Human Aspects in Secure Messaging

Kumulative Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

von

Sergej Dechand

Bonn, December 2024

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der
Rheinischen Friedrich-Wilhelms-Universität Bonn

Gutachter / Betreuer:	Prof. Dr. Matthew Smith
Gutachter:	Prof. Dr. Michael Meier
Tag der Promotion:	29.04.2025
Erscheinungsjahr:	2025

Abstract

The widespread adoption of digital communication demands robust security and privacy protections, particularly through secure messaging systems that can protect personal and sensitive information. Despite advancements in end-to-end security, encryption, and anonymity, significant gaps remain in usability and user trust, limiting widespread adoption. This cumulative dissertation examines the human aspects of secure messaging systems through four peer-reviewed studies, addressing fundamental challenges in usability, trust establishment, and practical implementations.

Diverse methodological approaches drive the research, including systematic protocol analysis with a focus on human aspects, large-scale empirical studies, and qualitative investigations, alongside the proposal and evaluation of improved technical implementations. First, a comprehensive systematization of knowledge establishes a unified framework for evaluating secure messaging protocols and "in-the-wild" tools, investigating critical gaps in current approaches. Second, an empirical study with 1047 participants examines fingerprint representation approaches for trust establishment. Third, qualitative research explores potential misconceptions in user mental models and trust for end-to-end security in general. Finally, a novel hardware-based approach utilizing NFC-enabled wearables demonstrates practical solutions for simplifying cryptographic key management while maintaining security.

Key findings indicate that (1) trust establishment remains the cornerstone of secure messaging, as it requires user interaction and underpins the entire security guarantees; failure in this area compromises the system entirely. (2) traditional hex-based fingerprint representations significantly underperform in both attack detection and perceived usability compared to the proposed sentence-based representation, but also numeric representation – as commonly used outside cryptographic contexts – also proving more effective; (3) users mistrust messaging platforms and security features in general and substantially overestimate attackers while underestimating cryptographic capabilities; and (4) less invasive security mechanisms as with using wearables show promise for broader adoption. The findings align with current developments in secure messaging applications, where similar verification approaches are used.

This work advances the field of usable security by bridging theoretical understanding with practical implementation, contributing to the development of more effective and accessible secure communication systems. The findings provide guidance for designing next-generation secure messaging solutions that balance robust security with user needs and capabilities.

Acknowledgements

Completing this thesis would not have been possible without the support of many individuals, especially my wife and family, to whom I am deeply grateful.

First and foremost, I extend my sincere appreciation to my "Doktorvater" Matthew Smith, for his exceptional academic guidance, constructive feedback, and dedication to advancing my research. His expertise and commitment to scientific rigor have been instrumental in shaping this work and my development as a researcher. Those long nights for paper deadlines fueled by coffee and determination will always be remembered.

I am particularly thankful to my collaborators and co-authors whose contributions have significantly enriched my research: Nik Unger, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, Dominik Schürmann, Karoline Busse, Yasemin Acar, Alena Naiakshina, Anastasia Danilova, Khaled Yakdan, Elmar Padilla and Lars Wolf. Their expertise and collaborative spirit have enhanced the quality and scope of this work.

This work has benefited from the collegiate atmosphere at the University of Bonn and the constructive academic discourse within the research community. And yes, to all the secure messaging apps, to ensure my procrastination was end-to-end encrypted.

Author's Publications

1 Covered Core Publications

where the thesis author made key contributions, forming the main thesis chapters:

- Sergej Dechand, Dominik Schürmann, Karoline Busse, Yasemin Acar, Sascha Fahl, and Matthew Smith: *An Empirical Study of Textual Key-Fingerprint Representations*, USENIX Security Symposium (2016) [1]
- Sergej Dechand, Alena Naiakshina, Anastasia Danilova, and Matthew Smith: *In encryption we don't trust: The effect of end-to-end encryption to the masses on user perception*, doi: 10.1109/EuroSP.2019.00037, 2019 IEEE European Symposium on Security and Privacy (2019) [2]
- Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith: *SoK: Secure Messaging*, doi: 10.1109/SP.2015.22, 2015 IEEE Symposium on Security and Privacy (S&P) (2015) [3]
  together with its extended version referenced in the publication
  Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith: *SoK: Secure Messaging (Extended Version)*, University of Waterloo (2015) [4]
- Dominik Schürmann, Sergej Dechand, and Lars Wolf: *OpenKeychain: An Architecture for Cryptography with Smart Cards and NFC Rings on Android*, doi: 10.1145/3130964, Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (2017) [5]

2 Referenced Supporting Contributions

where the thesis author significantly contributed by re-using methodologies or designing related systems outside the messaging context. These are only cited in the related chapters:

- Sascha Fahl, Sergej Dechand, Henning Perl, Felix Fischer, Jaromir Smrcek, and Matthew Smith: *Hey, NSA: Stay Away from my Market! Future Proofing App Markets against Powerful Attacker*, doi: 10.1145/2660267.2660311, Conference on Computer and Communications Security (2014) [6]
- Khaled Yakdan, Sergej Dechand, Elmar Gerhards-Padilla, and Matthew Smith: *Helping Johnny to Analyze Malware: A Usability-Optimized Decompiler and Malware Analysis User Study*, doi: 10.1109/SP.2016.18, Proceedings of the 37th IEEE Symposium on Security and Privacy (2016) [7]

3 Related Research

to the work on this dissertation, the thesis author has also participated in various related lines of research, such as usable security for developers and it professionals.

- Anil Kurmus, Sergej Dechand, and Rüdiger Kapitza: *Quantifiable run-time kernel attack surface reduction*, doi: 10.1007/978-3-319-08509-8_12, International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (2014)
- Marten Oltrogge, Yasemin Acar, Sergej Dechand, Matthew Smith, and Sascha Fahl: *To Pin or Not to Pin-Helping App Developers Bullet Proof Their TLS Connections*, USENIX Security Symposium (2015)
- Henning Perl, Sergej Dechand, Matthew Smith, Daniel Arp, Fabian Yamaguchi, Konrad Rieck, Sascha Fahl, and Yasemin Acar: *Vccfinder: Finding potential vulnerabilities in open-source projects to assist code audits*, doi: 10.1145/2810103.2813604, Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (2015)
- Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, Marco Herzog, Sergej Dechand, and Matthew Smith: *Why Do Developers Get Password Storage Wrong?: A Qualitative Usability Study*, doi: 10.1145/3133956.3134082, Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (2017)

Contents

CHAPTER 1

Introduction

The rapid adoption of digital communication platforms has transformed how people connect, collaborate, and express themselves. Yet, as reliance on these tools grows, so does the need to secure private conversations. From activists resisting authoritarian regimes to ordinary users seeking privacy, secure communication has emerged as a critical battleground in the fight for digital rights. The history of secure digital communication dates back to the early days of email, when privacy-conscious users recognized the need to protect their messages.

This need led to the 1991 release of Pretty Good Privacy (PGP) by Phil Zimmermann [12], the first widely available tool to bring public-key cryptography to the general public. Traditionally, users had to either directly exchange keys in advance or use the web-of-trust model for key validation. The latter relied on chains of validations to establish identity trust, but both approaches proved too complex for average users to understand and manage. Around the same time, RSA Data Security began developing S/MIME (Secure/Multipurpose Internet Mail Extensions) as a standard track alternative to PGP [13]. S/MIME's hierarchical trust model relied on certificate authorities – trusted third parties that verify and certify identities. Although this approach simplified trust establishment compared to PGP, it introduced dependencies on authorities, and it remained unclear where to retrieve the public keys before the first communication. Despite their different approaches – PGP's web of trust versus S/MIME's hierarchical trust model – both standards struggled with usability challenges that came to define early security tools [14–17]. Their need to manually exchange keys, verify identities, and understand complex cryptographic concepts created barriers preventing widespread adoption beyond technically skilled users. With less than 0.06 % of emails transmitted in 2022 being end-to-end secured [18], it is evident that email has failed to achieve widespread security adoption and is likely to remain inherently insecure.

While email encryption efforts stalled, instant messaging experienced its first major growth phase during the 1990s and early 2000s. During this early era of online communication, commercial services such as AIM, ICQ, and MSN messengers[1] – all now largely obsolete – and various IRC chat systems allowing group communication dominated the messaging landscape. These platforms prioritized real-time message delivery over security and privacy considerations, utilizing (mostly) unencrypted transmission protocols through centralized servers or peer-to-peer connections. This architecture made messages vulnerable to interception or surveillance by both service providers

[1] AIM (AOL Instant Messenger), ICQ, and MSN (Microsoft Network) messengers were popular early instant messaging services, while IRC (Internet Relay Chat) facilitated decentralized group communication. All have been largely replaced by modern platforms.

and potential attackers with access to the network infrastructure. As these platforms became widely used, privacy-conscious users began making early attempts at securing instant messaging.

1.1 Evolution of Secure Messaging

Extending the security definitions established by Goldberg et al. [19], secure messaging deals with three key security challenges mostly solved by cryptography (explored in detail in Chapter 2): trust establishment – ensuring users are communicating with their intended partners free of *man-in-the-middle* (MITM) attacks, conversation security – protecting message content's confidentiality and integrity, and transport privacy – preserving anonymity and concealing communication metadata. Early attempts to solve secure messaging typically involved retrofitting email encryption tools like `PGP`, again requiring users to manually exchange keys and verify identities. These approaches carried over the technical complexity and usability issues that had already hindered the adoption of encrypted email among non-technical users, ultimately leading to insecure communication practices.

In response to `PGP`'s security and usability limitations, Borisov, Goldberg, and Brewer introduced *Off-the-Record Messaging* (`OTR`) in 2004 as a new standard designed explicitly for secure instant messaging [20]. For easier adoption, the authors provided a client library to facilitate support for instant messaging client developers. Unlike the retrofitted email/messaging encryption with `PGP` and `S/MIME`, `OTR` even offered more advanced security features such as *backward/forward secrecy* and *deniability* (all definitions presented in detail in Chapter 2), which appealed to those seeking enhanced privacy in their conversations. More importantly, `OTR` achieved a significant usability improvement by adopting a *trust on first use* (TOFU) model similar to `OpenSSH`, i. e., securing all conversations with automatically-exchanged keys and only notifying the user when keys change. Since key verification is optional, even novice users can use it without requiring them to exchange any keys in advance or understand the complex web of trust relationships or certificate authorities. This meant that regular users could communicate securely by default, while security-conscious users retained the option to verify textual key fingerprints if they suspected manipulation or wished to ensure authenticated communication. `OTR` gained traction in niche communities, particularly among technically savvy users, yet despite these innovations, `OTR` still faced adoption hurdles similar to those of its predecessors, especially within the mainstream user base.

In the following years, a series of high-profile security incidents made the limitations of email encryption and early secure messaging solutions starkly apparent. Key incidents such as the Iranian Gmail breach in 2009, which exposed email data and targeted journalists [21], and the 2011 DigiNotar compromise, which enabled attackers to intercept and decrypt private Gmail communications using forged SSL certificates [22], underscored the critical need for end-to-end security in private communications. Further scandals like the Snowden revelations in 2013 [23], which exposed global mass surveillance programs, and the Sony Pictures hack in 2014 [24], where a massive leak of unencrypted email communications exposed private information and damaged the company's reputation, highlighted the inherent weaknesses of centralized trust models and the broader risks of state and corporate surveillance. These events underscored the importance of implementing end-to-end security and privacy at scale, while also highlighting the need to move beyond centralized trust models that leave communications vulnerable to breaches and surveillance.

The growing demand for secure messaging and the risks posed by centralized trust models have incentivized platform providers to adopt end-to-end encryption. A breach exposing billions of private conversations on a major messaging platform, similar to the aforementioned Email hacks [22, 24], would represent a catastrophic failure of user privacy. Platforms like `WhatsApp` and `iMessage` now actively promote end-to-end security features [25, 26], while open protocols such as `Signal protocol` and Matrix have emerged to democratize secure communication [27–29], making privacy accessible to a wider audience.

1.2 Human Aspects in Secure Messaging

Despite these advancements, the widespread adoption of secure messaging technologies has not been without hurdles. While end-to-end security has become a cornerstone of modern communication platforms, many users still struggle with understanding the underlying concepts to remain secure [2, 14–17]. Inconsistent implementations across platforms creates a fragmented landscape [3], complicating even expert users' judgment and understanding. Balancing security with usability remains critical, as overly complex systems risk alienating users or promoting insecure practices.

This thesis examines the intersection of technical security implementations and human behavior. As an article thesis – also known as cumulative dissertation in German academic systems – it investigates the human aspects of secure messaging, exploring how to design systems that users can and will use correctly while maintaining strong security guarantees. Here, it focuses on two fundamental security properties: trust establishment to the process of users verifying that they are actually communicating with the intended parties, and conversation security to protect message content for confidentiality and integrity. To address these security properties comprehensively, the thesis builds upon four peer-reviewed key publications presented in Chapters 2 to 5, investigating secure messaging. Each publication contributes to a deeper understanding of how, in the long term, secure messaging can be made more effective and accessible to the general public. While transport privacy is another critical aspect of secure messaging as outlined in Chapter 2, the rest of the thesis focuses primarily on trust establishment and conversation security, as transport privacy considerations were primarily addressed by collaborators. Bridging the gap between theory and practice, every approach is examined through three critical dimensions: the systematic analysis of security properties, the human factors affecting acceptance and usability, and the practical challenges during the adoption.

While the evolution from early email to modern secure messaging platforms represents significant progress, fundamental challenges persist in making secure communication both usable and effective. The historical difficulties with traditional security protocols highlight how technical solutions alone cannot ensure widespread adoption – user experience, trust, and perception all play crucial roles. This understanding shaped this thesis' research agenda, which examines secure messaging through multiple lenses: from systematic analysis of existing solutions to qualitative and empirical studies of user behavior and novel technical implementations. This analysis encompasses both commercial implementations and academic research and observed that both communities naturally learned from each other, leading to a mutually beneficial evolution of secure messaging practices. These historical developments led to an analysis of existing approaches and tools. Based on this list and obvious gaps between technical capabilities and real-world usage patterns, four big research questions emerged.

1.3 Research Questions

This thesis addresses four key research questions aimed at advancing secure messaging. Focusing on the research gaps will allow for a higher adoption of end-to-end security. The questions are addressed with the presented diverse research methods to examine the research from multiple angles comprehensively. The following questions guided the work throughout the research process:

RQ 1 **How can existing secure messaging solutions be systematically categorized and evaluated to identify major challenges in terms of security, usability, and adoption perspectives and what needs further improvements?** The goal here was to create a structured understanding of the current landscape of secure messaging tools, protocols, and approaches, examining their strengths and limitations.

A unified framework for categorizing and evaluating new approaches in this field helps identify problems and create a guide for the research community to move forward on this important topic.

This research question is addressed in *Publication 1* [3, 4], covered in Chapter 2. The results from this research directly influence the next three research questions.

RQ 2 **What are the primary challenges users face in trust establishment, and how these can be addressed through both manual and automated approaches?** This question aimed to explore methods that simplify the process by which users verify their communication partners, whether through manual verification techniques like key fingerprints or automated systems.

Evidence-based recommendations help design more effective and user-friendly trust establishment mechanisms, particularly focusing on key fingerprint representations.

This research question is addressed in *Publication 3* [1], covered in Chapter 4. The results from this research will help the community to verify the authority-based trust establishment.

RQ 3 **How do users understand secure messaging tools, and what misconceptions or gaps exist in their mental models of these technologies?** Understanding users' perceptions and knowledge of secure messaging was crucial for identifying usability issues that impact broader adoption.

Detailed characterization of user mental models and perception regarding secure messaging, including identification of common misconceptions and areas where user understanding diverges from technical reality, helps the community to address these in next-generation messengers.

This research question is addressed in *Publication 2* [2], covered in Chapter 3.

RQ 4 **How can users improve the cryptographic key management without sacrificing usability?** This question explored approaches to simplify key management while maintaining strong security properties, addressing a fundamental challenge in making security accessible to average users.

Novel technical architecture for integrating wearable devices into cryptographic operations demonstrates practical approaches to simplifying key management.

This research question is addressed in *Publication 4* [5], covered in Chapter 5.

While prior work extensively focused on technical protocols and cryptographic properties, the research questions Items RQ 1 to RQ 4 also consider the human and practical aspects of secure messaging - from understanding user perceptions to improving usability overall. Each question builds upon insights gained from addressing the previous questions, creating a cohesive research narrative that spans both theoretical and practical aspects of secure messaging. Their methodological diversity allows the examination of secure messaging challenges from multiple perspectives, providing both breadth and depth of understanding.

1.4 Research Overview and Contributions

The research presented in this dissertation revolves around the challenges and opportunities in secure messaging, particularly from a human-centric perspective. Ultimately, this work seeks to establish secure messaging as default, accessible to all rather than reserved for specialists. This research examined secure messaging during a pivotal evolution period (2015-2020), as major platforms like `WhatsApp` and `iMessage` started to transition to end-to-end encryption by default. This research period captured `WhatsApp`'s transition to end-to-end encryption, with its billion-user base, allowing empirical study of user adaptation patterns. The research benefited from collaboration between academic institutions and messaging developers, e. g., `Signal` developers and members from the EFF, allowing insights into both theoretical advances and practical deployment challenges. To this end, four representative publications or studies are introduced and organized in individual Chapters 2 to 5 presenting peer-reviewed *Publications 1-4* [1–3, 5], each per research question with a particular contribution.

Each chapter follows the same structure: starting with a short introduction highlighting the contributions and context of the study, it then presents the peer-reviewed publication in its original published format, embedding the respective venue's published layout, typography, and style guidelines. The embedding is designed so that the running head indicates the current chapter and section, helping the reader navigate the dissertation. Additionally, each page features two separate page numbering systems: an outer page number for the dissertation and an inner page number corresponding to the original publication. The original formatting, including internal references within the publications – such as (inner) page numbers, tables, and figures – is preserved relative to their respective sections to maintain consistency with the published versions. Bibliographic references cited within the publications remain publication-specific and unchanged, retaining their original numbering and formatting. Finally, each chapter concludes with a short summary that contextualizes the findings within the current state of the art and subsequent developments in the field.

The following chapters explore these studies in depth. Each publication addresses a distinct research question and contributes unique insights to the secure messaging landscape in a high-level overview. A more detailed discussion of the particular contributions can be found in the publications presented in every chapter:

Publication 1 | Systematization of Secure Messaging Solutions

Despite the mentioned availability of encryption protocols like `OpenPGP` and `S/MIME`, real-world adoption remains minimal due to usability barriers and the challenges of managing end-to-end security independently: Less than 0.06 % of emails are encrypted nowadays [18]. However, the rise

of instant messaging and a renewed focus on end-to-end security have catalyzed advancements in secure messaging:

Chapter 2, consisting of the *SoK: Secure Messaging* publication, published in *2015 IEEE Symposium on Security and Privacy (S&P)*, in its original form [3, 4], offers a comprehensive systematization of secure messaging solutions, incorporating insights from academic research and real-world implementations, referred to as "in-the-wild" projects in the publications. Presented as a Systematization of Knowledge (SoK) publication [30, 31], it establishes foundational definitions, categorizes approaches, and identifies challenges within secure messaging, establishing the groundwork and analytical framework for analyses in subsequent chapters. This forms the foundation, including the research background and technical definitions used in this thesis. In traditional monographic dissertations, this is also known as the background chapter.

The evaluation framework identifies three core challenges in secure messaging: trust establishment, conversation security, and transport privacy. The systematization evaluates approaches from both academia and practical implementations in terms of security capabilities, usability issues, and adoption implications.

Contribution The chapter's key contributions include: defining standardized security and privacy terms, systematizing both academic and real-world approaches, conducting a comparative evaluation, and highlighting unresolved challenges in secure messaging. The analysis is structured through a novel evaluation framework that examines security properties, usability aspects, and adoption challenges. Some of the unresolved areas in terms of the human aspects in trust establishment and conversation security will then be covered in the following chapters.

Methodology With the formation of a unified evaluation framework, this research employs a comprehensive systematic analysis of end-to-end encryption protocols from both academic literature and real-world implementations. The methodology particularly focuses on comparing how solutions address core challenges in trust establishment, conversation security, and transport privacy, while always considering the trade-offs between security properties, usability aspects, and adoption challenges.

Impact This work identified critical gaps in the secure messaging landscape, particularly in trust establishment and usability aspects, through systematically analyzing existing systems. This framework not only guided the research direction of subsequent publications presented in this thesis but has also been utilized by other researchers to evaluate new secure messaging protocols and implementations. The systematic categorization of security properties and usability challenges provides a common language for discussing and comparing secure messaging solutions, contributing to more structured research in this area.

Publication 2 | Evaluation of Trust Establishment

As outlined in *Publication 1* presented in Chapter 2, trust establishment in secure messaging remains a fundamental challenge, as systems must balance the conflicting goals of usability and security. It criticizes the predominant reliance on centralized trust authorities for prioritizing ease over security. Chapter 3 consists of the *An Empirical Study of Textual Key-Fingerprint Representations* peer-reviewed publication in 2016 at the *25th USENIX Security Symposium*, in its original form [1]. While manual

fingerprint comparison is acknowledged as the most secure approach in decentralized systems, its design and usability flaws have prevented widespread adoption.

The empirical study of Publication 2 presented in Chapter 3 directly addresses these concerns by offering actionable insights into improving manual verification through better fingerprint representation. This chapter critiques these traditional approaches, highlighting their vulnerability to partial preimage attacks and unnecessary cognitive burdens on users. Addressing these concerns, the study empirically evaluates six textual key fingerprint schemes, including numeric, word lists, and sentence-based encodings, through a large-scale online experiment with 1,047 participants. Findings reveal that the established hex comparisons perform worst in terms of usability and security, and the study proposes tested and proven alternatives.

Contribution The study contributes by systematically evaluating the usability and security trade-offs of textual key fingerprint representations, introducing sentence-based encodings as a practical and highly effective alternative. It demonstrates how these alternatives achieve superior attack detection rates, usability ratings, and user confidence compared to traditional hexadecimal representations. This work provides actionable recommendations for replacing outdated formats with inclusive, user-friendly solutions, laying the foundation for more secure and accessible trust establishment mechanisms in secure messaging systems. Some of the proposed approaches are even used in popular messengers nowadays.

Methodology The research employs a large-scale quantitative empirical study (n=1,047) through a carefully designed online experiment. The online experiment uniquely incorporated realistic attack scenarios based on contemporary computational capabilities for generating partial preimage collisions, mirroring real-world threat models. This approach enabled direct comparison of verification methods under practical security conditions, measuring both user performance in detecting attacks and subjective aspects like user confidence and perceived usability.

Impact By combining thorough experimental design with realistic security parameters, the study provides actionable insights into the effectiveness of different verification approaches in real-world scenarios. The empirical evaluation of fingerprint representation schemes in *Publication 2* demonstrated that sentence-based approaches achieved the highest performance in attack detection by 98 %, and user confidence in terms of trustworthiness and usability, while numeric representations emerged as the most practical solution when considering language barriers. This research highlighted a significant disconnect in secure messaging implementations and other security tools, such as `OpenSSH`, at the time when platforms predominantly used hexadecimal representations despite their demonstrated worse performance in both security and usability metrics. Throughout the research process, the findings were shared with key stakeholders in the messaging community, including developers of the Signal application and members of the Electronic Frontier Foundation (EFF). These exchanges served as a feedback mechanism, ensuring that the research remained aligned with practical considerations and real-world applications.

In alignment with this research, today's popular messengers – `WhatsApp`, which implemented the `Signal protocol` in the meantime, and others – all employ numeric representations in their optional verification systems, validating our findings and the research conclusions about the optimal balance between security, usability, and practical deployment constraints. The detailed implications and adoption of these findings by the main platforms are discussed in Chapter 3.

Publication 3 | Analysis of User Mental Models

Chapter 4 builds on *Publication 1*'s observation that the understanding of how end-users perceive secure messaging remains underexplored, despite its critical role in designing usable and effective solutions. Chapter 4 consists of the *In Encryption We Don't Trust: The Effect of End-to-End Encryption to the Masses on User Perception* peer-reviewed publication, published in *2019 IEEE European Symposium on Security and Privacy* (*EuroS&P*), in its original form [2]. As secure messaging technologies like WhatsApp's implementation of the Signal Protocol made end-to-end encryption widely accessible, the assumption was that user trust and awareness would naturally improve. However, the extent to which these advancements have translated into accurate user mental models and trust in encryption technologies has remained unclear.

The chapter presents the first longitudinal qualitative study exploring the evolution of user perceptions of secure messaging, comparing mental models before and after the mainstream adoption of end-to-end encryption. The study examines user understanding of secure messaging, identifies common misconceptions, and provides recommendations for improving user interfaces and messaging practices. To capture users' perception and understanding, a qualitative user study based on interviews was conducted to illustrate average users' mental models regarding secure messaging, i. e., how they see the messaging world. The goal is identifying aspects that help messenger developers implement usable and secure software without contradicting the average user's understanding.

Contribution The study identifies persistent gaps in understanding, including misconceptions about the capabilities and limitations of encryption. For example, while explicit encryption notifications have become common in apps like WhatsApp and iMessage, most users remain unaware of what these mechanisms entail or how they function. Furthermore, many participants demonstrated overestimated perceptions of attackers' capabilities while underestimating the protections afforded by cryptography. Finally, the findings contribute actionable recommendations for designing next-gen secure messaging protocols that address critical gaps in user understanding.

Methodology The research employs a two-phase qualitative investigation examining user perceptions before and after the mainstream adoption of end-to-end encryption. Unstructured focus groups formed the development of semi-structured interviews, enabling systematic exploration of users' mental models and security understanding. The final interviews then provided insights into how widespread encryption deployment impacts user trust and perception.

Impact As the first-of-its-kind mental model in the area of secure messaging, it how users perceived the security of popular messengers and whether they understand how to use security features correctly. Users significantly overestimate attackers' capabilities and underestimate cryptography ("no technical solution to stop skilled attackers from getting data exists"). Most users didn't even comprehend what the security info bubbles and warnings meant and stated a general feeling of "being not well protected." Based on the findings, the work offers recommendations for how the security messaging community can address these challenges.

Publication 4 | Repurposing Wearables for Cryptographic Security

Building Publication 1's critique that current secure messaging systems often neglect the usability-security balance in conversation security, Chapter 5 presents a novel approach to automating

authentication via NFC-based security tokens rather than unlocking cryptographic keys with passwords or smartcards. Chapter 5 consists of the *OpenKeychain: An Architecture for Cryptography with Smart Cards and NFC Rings on Android* peer-reviewed journal publication at the *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT) 2017*, in its original form [5]. The approach addresses the vulnerabilities associated with on-device key storage by offloading cryptographic keys and operations to external tokens on wearable devices. Leveraging NFC technology, it provides a more user-friendly and secure mechanism for key management, effectively reducing user interaction with complex cryptographic operations. The built architecture for NFC-based cryptography on Android via external security tokens allows messaging tool developers to incorporate end-to-end security protection in their applications.

Contribution This is one of the first implementations to integrate wearable devices, such as NFC-enabled rings, for cryptographic operations in real-world messaging and email applications. The chapter's usability lab study, conducted under enterprise field conditions with 40 participants, evaluates the architecture's practical performance, revealing its potential to significantly enhance the usability and security methods in emails or secure messaging.

Methodology The research combines technical implementation with empirical evaluation, developing a novel NFC-based authentication architecture for Android and assessing it through controlled usability studies in enterprise settings. Our implementation extends existing cryptographic frameworks with wearable device integration, while the evaluation, which involved 40 participants under realistic working conditions, provides insights into practical deployment scenarios and usability implications.

Impact The mental model findings from *Publication 3* have informed how messaging platforms communicate security features to users, while the authentication architecture from *Publication 4* has been integrated into OpenKeychain, the most used OpenPGP implementation on Android. This research established principles for NFC-based authentication through wearables, which are reflected in current developments in security mechanisms. As organizations are increasingly adopting two-factor authentication with smartcards or security tokens such as YubiKey, the integration of these features into wearable devices can be seen as a natural progression, same as these are now used for device unlocking.

Synthesis | Interconnection of Publications

While each publication provides a standalone contribution, together they address complementary aspects of secure messaging. This framework is built on diverse methodologies, ranging from systematic protocol analysis to simulated attacks under realistic conditions and from empirical user studies to novel implementations. These four publications represent a comprehensive investigation of secure messaging that bridges theoretical understanding and practical implementation.

Publication 1's evaluation framework sets the stage by defining the problem space and identifying gaps in current solutions regarding user interaction and security. In particular, it identified key challenges in trust establishment, which directly motivated and led to the empirical investigation of trust establishment mechanisms in *Publication 2*, in particular, textual key fingerprint representations. The usability issues outlined in *Publication 1* and the findings about user difficulties with

technical security concepts in *Publication 2* revealed the need to better understand how users conceptualize secure messaging, leading to the qualitative in-depth exploration of user mental models in *Publication 3*. Finally, based on the understanding gained from studying user perceptions and behaviors, where users tend to use noninvasive features without actually knowing it, experimented with adding minimal invasive methods for key storage and verification in *Publication 4*.

CHAPTER 2

Background and Systematization

Secure messaging has become increasingly important as individuals and organizations seek to protect their communications from surveillance and other threats. The rise of instant messaging and the revelations of widespread surveillance and security breaches, however, sparked renewed interest in end-to-end security protection, referred to as secure messaging in general, in both practical applications and academic research.

This chapter incorporates, in its entirety, a previously published technical report [4], which serves as an extended version of the peer-reviewed publication [3]. The technical report, referenced in the peer-reviewed publication, was chosen for inclusion because it retains the same semantics as the peer-reviewed version while adding further explanations and detailed definitions to assist readers less familiar with the secure messaging landscape. For the sake of completeness, the peer-reviewed version has been included in Appendix A.1.

Building on the research questions introduced in Chapter 1, this chapter provides a thorough systematic analysis of existing secure messaging approaches and their limitations, drawing on both academic research and approaches that have emerged in actual application implementations using their modified cryptographic protocols (referred to as "in-the-wild" projects). A notable example analyzed in this systematization is the `Signal protocol`[1], which built upon `OTR`'s security properties while addressing its limitations regarding asynchronous communication. Despite introducing significant cryptographic advancements as early as 2013, the protocol initially received limited attention from the academic community until this systematization work began.

As a Systematization of Knowledge (SoK) publication [30, 31], this work perfectly serves as the foundational background for the dissertation, addressing the key challenges and systematizing the current state-of-the-art approaches in secure messaging systems. It provides a structured framework for understanding and evaluating messaging systems regarding security, usability, and adoption, laying the groundwork for the analyses and contributions presented in later chapters. The upcoming Chapters 3 to 5 are based on the definitions and problem areas outlined in this chapter.

The primary contributions of this publication include: 1) establishing standardized definitions for security and privacy features in secure messaging; 2) systematizing approaches to secure

[1] `Signal` formerly `TextSecure` and its protocol `Axolotl` underwent a renaming to `Signal` and `Signal protocol` at the end of 2015. The publication presented in this chapter uses the old naming, whereas the rest of the dissertation sticks with the new naming.

messaging from academic research and real-world implementations, referred to as *in-the-wild* projects; 3) conducting a comparative evaluation of these approaches; and 4) identifying current research challenges to guide future work. These contributions provide a comprehensive perspective that supports the subsequent chapters and informs about ongoing efforts in the field.

2.1 Peer-Reviewed Publication 1 | SoK: Secure Messaging

Authors' Contributions

The published technical report [4], an extended version of a peer-reviewed publication at the 2015 IEEE Symposium on Security and Privacy (DOI: 10.1109/SP.2015.22) [3], one of the premier venues in computer security, attached in Appendix A.1, is included in its entirety in original form in this section. This work synthesizes and evaluates secure messaging approaches using a consistent framework – particularly important given the rapid evolution of messaging systems following the data breaches and surveillance revelations 2010 - 2015. The authors' contributions, with two joint first authors that are relevant to the contents of this chapter, are as follows:

- **Sergej Dechand** I took the lead in the systematization, review, and evaluation of existing trust establishment protocols and conversation security mechanisms, although with less focus on group chats, which is less relevant for this dissertation. These sections form the core focus and research areas in this dissertation and represent the primary technical background chapter of this work.
- **Nik Unger** responsible for the systematization, review, and evaluation of metadata protection mechanisms, as well as addressing the group chat aspects of communication security, while Sergej focused mainly on the other aspects. Nik's contributions complemented the overall scope of the research and provided critical insights into the multi-user communication framework.
- **Joseph Bonneau** provided the initial list of various "in-the-wild" secure messaging tools considered in our evaluation. His feedback during all phases of the work significantly enhanced the technical rigor and comprehensiveness of the report.
- **Ian Goldberg, Sascha Fahl, Henning Perl, Matthew Smith** reviewed the work at various stages, offering valuable feedback, cryptographic reviews, and suggestions that greatly improved the quality and clarity of the final report.

The following pages present an in-depth analysis through the inclusion of a peer-reviewed publication that systematizes secure messaging approaches, providing a foundation for the analyses presented in later chapters. This systematization is essential to identify recurring challenges in trust establishment, conversation security, and transport privacy, thereby setting the stage for further empirical investigation in the remainder of this dissertation.

SoK: Secure Messaging[1]

Nik Unger*, Sergej Dechand†
Joseph Bonneau‡§, Sascha Fahl¶, Henning Perl¶
Ian Goldberg*, Matthew Smith†
* University of Waterloo, † University of Bonn, ‡ Stanford University, § Electronic Frontier Foundation, ¶ Fraunhofer FKIE

***Abstract*—Motivated by recent revelations of widespread state surveillance of personal communication, many products now claim to offer secure and private messaging. This includes both a large number of new projects and many widely adopted tools that have added security features. The intense pressure in the past two years to deliver solutions quickly has resulted in varying threat models, incomplete objectives, dubious security claims, and a lack of broad perspective on the existing cryptographic literature on secure communication.**

In this paper, we evaluate and systematize current secure messaging solutions and propose an evaluation framework for their security, usability, and ease-of-adoption properties. We consider solutions from academia, but also identify innovative and promising approaches used "in the wild" that are not considered by the academic literature. We identify three key challenges and map the design landscape for each: trust establishment, conversation security, and transport privacy. Trust establishment approaches offering strong security and privacy features perform poorly from a usability and adoption perspective, whereas some hybrid approaches that have not been well studied in the academic literature might provide better trade-offs in practice. In contrast, once trust is established, conversation security can be achieved without any user involvement in most two-party conversations, though conversations between larger groups still lack a good solution. Finally, transport privacy appears to be the most difficult problem to solve without paying significant performance penalties.

I. Introduction

Most popular messaging tools used on the Internet do not offer end-to-end security. Even though protocols such as OpenPGP and S/MIME have been available for decades, they have failed to achieve widespread adoption and have been plagued by usability issues [2]–[5]. However, recent revelations about mass surveillance by intelligence services have highlighted the lack of security and privacy in messaging tools and spurred demand for better solutions. A recent Pew Research poll found that 80% of Americans are now concerned about government monitoring of their electronic communications. A combined 68% of respondents reported feeling "not very secure" or "not at all secure" when using online chat and 57% felt similarly insecure using email [6]. Consequently, many new applications claiming to offer secure communication are being developed and adopted by end users.

Despite the publication of a large number of secure messaging protocols in the academic literature, tools are being released with new designs that fail to draw upon this knowledge, repeat known design mistakes, or use cryptography in insecure ways. However, as will become clear over the course of this paper, the academic research community is also failing to learn some lessons from tools in the wild.

Furthermore, there is a lack of coherent vision for the future of secure messaging. Most solutions focus on specific issues and have different goals and threat models. This is compounded by differing security vocabularies and the absence of a unified evaluation of prior work. Outside of academia, many products mislead users by advertising with grandiose claims of "military grade encryption" or by promising impossible features such as self-destructing messages [7]–[10]. The recent EFF Secure Messaging Scorecard evaluated tools for basic indicators of security and project health [11] and found many purportedly "secure" tools do not even attempt end-to-end encryption.

We are motivated to systematize knowledge on secure messaging due to the lack of a clear winner in the race for widespread deployment and the persistence of many lingering unsolved research problems. Our primary goal is to identify where problems lie and create a guide for the research community to help move forward on this important topic. A further goal in this work is to establish evaluation criteria for measuring security features of messaging systems, as well as their usability and adoption implications. We aim to provide a broad perspective on secure messaging and its challenges, as well as a comparative evaluation of existing approaches, in order to provide context that informs future efforts. Our primary contributions are: (1) establishing a set of common security and privacy feature definitions for secure messaging; (2) systematization of secure messaging approaches based both on academic work and "in-the-wild" projects; (3) comparative evaluation of these approaches; and (4) identification and discussion of current research challenges, indicating future research directions.

After defining terminology in Section II, we present our systematization methodology in Section III. In subsequent sections (Sections IV–VI), we evaluate each of the proposed problem areas (namely *trust establishment*, *conversation security* and *transport privacy*) in secure messaging. Our findings are discussed and concluded in Section VII.

II. Background and Definitions

Secure messaging systems vary widely in their goals and corresponding design decisions. Additionally, their target audiences often influence how they are defined. In this section, we

[1]This is an extended version of our paper in the 2015 IEEE Symposium on Security and Privacy [1]. This document was last updated on 2015-04-14.

define terminology to differentiate these designs and provide a foundation for our discussion of secure messaging.

A. Types of specification

Secure messaging systems can be specified at three different broad levels of abstraction:

Chat protocols: At the most abstract level, chat protocols can be defined as sequences of values exchanged between participants. This mode of specification deals with high-level data flows and often omits details as significant as the choice of cryptographic protocols (e.g., key exchanges) to use. Academic publications typically specify protocols this way.

Wire protocols: Complete wire protocols aim to specify a binary-level representation of message formats. A wire protocol should be complete enough that multiple parties can implement it separately and interoperate successfully. Often these are specific enough that they have versions to ensure compatibility as changes are made. Implicitly, a wire protocol implements some higher-level chat protocol, though extracting it may be non-trivial.

Tools: Tools are concrete software implementations that can be used for secure messaging. Implicitly, a tool contains a wire protocol, though it may be difficult and error-prone to derive it, even from an open-source tool.

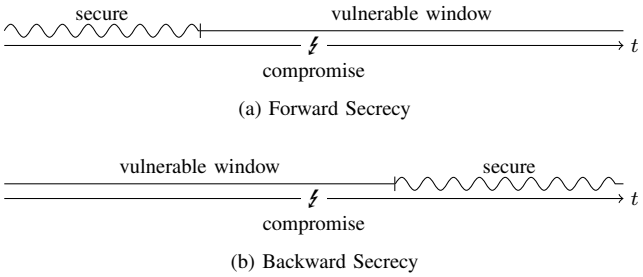
B. Synchronicity

A chat protocol can be *synchronous* or *asynchronous*. Synchronous protocols require all participants to be online and connected at the same time in order for messages to be transmitted. Systems with a peer-to-peer architecture, where the sender directly connects to the recipient for message transmission, are examples of synchronous protocols. Asynchronous protocols, such as SMS (text messaging) or email, do not require participants to be online when messages are sent, utilizing a third party to cache messages for later delivery.

Due to social and technical constraints, such as switched-off devices, limited reception, and limited battery life, synchronous protocols are not feasible for many users. Mobile environments are also particularly prone to various transmission errors and network interruptions that preclude the use of synchronous protocols. Most popular instant messaging (IM) solutions today provide asynchronicity in these environments by using a *store-and-forward* model: a central server is used to buffer messages when the recipient is offline. Secure messaging protocols designed for these environments need to consider, and possibly extend, this store-and-forward model.

C. Deniability

Deniability, also called *repudiability*, is a common goal for secure messaging systems. Consider a scenario where Bob accuses Alice of sending a specific message. Justin, a judge, must decide whether or not he believes that Alice actually did so. If Bob can provide evidence that Alice sent that message, such as a valid cryptographic signature of the message under Alice's long-term key, then we say that the action is *non-repudiable*. Otherwise, the action is *repudiable* or *deniable*.



(a) Forward Secrecy

(b) Backward Secrecy

Fig. 1. Session keys are protected from long-term key compromise.

We can distinguish between *message repudiation*, in which Alice denies sending a specific message, and *participation repudiation* in which Alice denies communicating with Bob at all. The high-level goal of repudiable messaging systems is to achieve deniability similar to real-world conversations.

A fundamental problem of deniability is that Justin may simply trust Bob even with no technical evidence due to Bob's reputation or perceived indifference. In a group chat, this problem may be even worse as Alice may need to convince Justin that a number of accusers are all colluding to frame her. It is not possible to construct a messaging system that overcomes this fundamental social problem; the best that can be done is to provide no stronger evidence than the word of the accusers. Some technical systems clearly offer more evidence; for example, signed PGP emails offer strong evidence that Alice really was the sender.

The cryptographic literature has produced many definitions of "deniability" since deniable encryption was first formally proposed [12]. For example, we can draw a distinction between an offline and online judge: in the offline case, the accuser attempts to convince the judge of an event after the conversation has already concluded; in the online case, the judge exchanges private communications with the accuser while the conversation is still taking place. Existing work defines online repudiation in incompatible ways, and very few protocols attempt to achieve meaningful online repudiation [13], [14]. Thus, in this work we only consider the offline setting.

D. Forward/Backward Secrecy

In systems that use the same static keys for all messages, a key compromise allows an attacker to decrypt the entire message exchange. A protocol provides *forward secrecy* if the compromise of a long-term key does not allow ciphertexts encrypted with previous session keys to be decrypted (Figure 1a). If the compromise of a long-term key does not allow subsequent ciphertexts to be decrypted by passive attackers, then the protocol is said to have *backward secrecy* (Figure 1b). However, tools with *backward secrecy* are still vulnerable to active attackers that have compromised long-term keys. In this context, the "self-healing" aspect of *backward secrecy* has also been called *future secrecy*. The terms are controversial and vague in the literature [15]–[17].

III. Systematization Methodology

Over the years, hundreds of secure messaging systems have been proposed and developed in both academia and industry. An exhaustive analysis of all solutions is both infeasible and undesirable. Instead, we extract recurring secure messaging techniques from the literature and publicly available messaging tools, focusing on systematization and evaluation of the underlying concepts and the desirable secure messaging properties. In this section, we explain our precise methodology.

A. Problem Areas

While most secure messaging solutions try to deal with all possible security aspects, in our systematization, we divide secure messaging into three nearly orthogonal problem areas addressed in dedicated sections: the *trust establishment* problem (Section IV), ensuring the distribution of cryptographic long-term keys and proof of association with the owning entity; the *conversation security* problem (Section V), ensuring the protection of exchanged messages during conversations; and the *transport privacy* problem (Section VI), hiding the communication metadata.

While any concrete tool must decide on an approach for each problem area, abstractly defined protocols may only address some of them. Additionally, the distinction between these three problem areas is sometimes blurred since techniques used by secure messaging systems may be part of their approach for multiple problem areas.

B. Threat Model

When evaluating the security and privacy properties in secure messaging, we must consider a variety of adversaries. Our threat model includes the following attackers:

Local Adversary (active/passive): An attacker controlling local networks (e.g., owners of open wireless access points).

Global Adversary (active/passive): An attacker controlling large segments of the Internet, such as powerful nation states or large Internet service providers.

Service providers: For messaging systems that require centralized infrastructure (e.g., public-key directories), the service operators should be considered as potential adversaries.

Note that our adversary classes are not necessarily exclusive. In some cases, adversaries of different types might collude. We also assume that all adversaries are participants in the messaging system, allowing them to start conversations, send messages, or perform other normal participant actions. We assume that the endpoints in a secure messaging system are secure (i.e., malware and hardware attacks are out of scope).

C. Systematization Structure

Sections IV–VI evaluate *trust establishment*, *conversation security*, and *transport privacy* approaches, respectively. For each problem area, we identify desirable properties divided into three main groups: *security and privacy features*, *usability features*, and *adoption considerations*. Each section starts by defining these properties, followed by the extraction of generic approaches used to address the problem area from existing secure messaging systems. Each section then defines and evaluates these approaches, as well as several possible variations, in terms of the already-defined properties. Concrete examples of protocols or tools making use of each approach are given whenever possible. The sections then conclude by discussing the implications of these evaluations.

In each section, we include a table (Tables I, II, and III) visualizing our evaluation of approaches within that problem area. Columns in the tables represent the identified properties, while rows represent the approaches. Groups of rows begin with a generic concept, specified as a combination of cryptographic protocols, followed by extension rows that add or modify components of the base concept. Whenever possible, rows include the name of a representative protocol or tool that uses the combination of concepts. Representatives may not achieve all of the features that are possible using the approach; they are merely included to indicate where approaches are used in practice. Each row is rated as providing or not providing the desired properties. In some cases, a row might only partially provide a property, which is explained in the associated description.

For each problem area, we identify desirable properties in three main categories:

1) Security and Privacy Properties: Most secure messaging systems are designed using standard cryptographic primitives such as hash functions, symmetric encryption ciphers, and digital signature schemes. When evaluating the security and privacy features of a scheme, we assume cryptographic primitives are securely chosen and correctly implemented. We do not attempt to audit for software exploits which may compromise users' security. However, if systems allow end users to misuse these cryptographic primitives, the scheme is penalized.

2) Usability Properties: Usability is crucial for the use and adoption of secure messaging services. Human end users need to understand how to use the system securely and the effort required to do so must be acceptable for the perceived benefits.

In previous research, various secure messaging tools have been evaluated and weaknesses in the HCI portion of their design have been revealed. The seminal paper "Why Johnny Can't Encrypt" [2] along with follow-up studies evaluating PGP tools [3], [4] and other messaging protocols [18]–[22] have also showed users encountering severe problems using encryption securely. However, these studies focused on UI issues unique to specific implementations. This approach results in few generic insights regarding secure messenger protocol and application design. Given the huge number of secure messaging implementations and academic approaches considered in our systematization, we opted to extract generic concepts. Because we focus on usability consequences imposed by generic concepts, our results hold for any tool that implements these concepts.

To evaluate the usability of secure messaging approaches, we examine the additional user effort (and decisions), security-related errors, and reduction in reliability and flexibility that they introduce. Our usability metrics compare this extra effort

to a baseline approach with minimal security or privacy features. This is a challenging task and conventional user studies are not well suited to extract such high-level usability comparisons between disparate tools. We opted to employ expert reviews to measure these usability properties, which is consistent with previous systematization efforts for security schemes in other areas [23], [24]. To consider usability and adoption hurdles in practice, we combined these expert reviews with cognitive walkthroughs of actual implementations based on Nielsen's usability principles [25]–[27] and already known end-user issues discovered in previous work [2]–[5], [19]–[21], [28]. These usability results supplement our technical systematization and highlight potential trade-offs between security and usability.

3) Ease of Adoption: Adoption of secure messaging schemes is not only affected by their usability and security claims, but also by requirements imposed by the underlying technology. Protocols might introduce adoption issues by requiring additional resources or infrastructure from end users or service operators. When evaluating the adoption properties of an approach, we award a good score if the system does not exceed the resources or infrastructure requirements of a baseline approach that lacks any security or privacy features.

IV. Trust Establishment

One of the most challenging aspects of messaging security is *trust establishment*, the process of users verifying that they are actually communicating with the parties they intend. *Long-term key exchange* refers to the process where users send cryptographic key material to each other. *Long-term key authentication* (also called *key validation* and *key verification*) is the mechanism allowing users to ensure that cryptographic long-term keys are associated with the correct real-world entities. We use *trust establishment* to refer to the combination of *long-term key exchange* and *long-term key authentication* in the remainder of this paper. After contact discovery (the process of locating contact details for friends using the messaging service), end users first have to perform trust establishment in order to enable secure communication.

A. Security and Privacy Features

A trust establishment protocol can provide the following security and privacy features:

Network MitM Prevention: Prevents Man-in-the-Middle (MitM) attacks by local and global network adversaries.

Operator MitM Prevention: Prevents MitM attacks executed by infrastructure operators.

Operator MitM Detection: Allows the detection of MitM attacks performed by operators after they have occurred.

Operator Accountability: It is possible to verify that operators behaved correctly during trust establishment.

Key Revocation Possible: Users can revoke and renew keys (e.g., to recover from key loss or compromise).

Privacy Preserving: The approach leaks no conversation metadata to other participants or even service operators.

B. Usability Properties

Most trust establishment schemes require key management: user agents must generate, exchange, and verify other participants' keys. For some approaches, users may be confronted with additional tasks, as well as possible warnings and errors, compared to classic tools without end-to-end security. If a concept requires little user effort and introduces no new error types, we award a mark for the property to denote good usability. We only consider the minimum user interaction required by the protocol instead of rating specific implementations.

Automatic Key Initialization: No additional user effort is required to create a long-term key pair.

Low Key Maintenance: Key maintenance encompasses recurring effort users have to invest into maintaining keys. Some systems require that users sign other keys or renew expired keys. Usable systems require no key maintenance tasks.

Easy Key Discovery: When new contacts are added, no additional effort is needed to retrieve key material.

Easy Key Recovery: When users lose long-term key material, it is easy to revoke old keys and initialize new keys (e.g., simply reinstalling the app or regenerating keys is sufficient).

In-band: No *out-of-band* channels are needed that require users to invest additional effort to establish.

No Shared Secrets: Shared secrets require existing social relationships. This limits the usability of a system, as not all communication partners are able to devise shared secrets.

Alert-less Key Renewal: If other participants renew their long-term keys, a user can proceed without errors or warnings.

Immediate Enrollment: When keys are (re-)initialized, other participants are able to verify and use them immediately.

Inattentive User Resistant: Users do not need to carefully inspect information (e.g., key fingerprints) to achieve security.

C. Adoption Properties

Multiple Key Support: Users should not have to invest additional effort if they or their conversation partners use multiple public keys, making the use of multiple devices with separate keys transparent. While it is always possible to share one key on all devices and synchronize the key between them, this can lead to usability problems.

No Service Provider Required: Trust establishment does not require additional infrastructure (e.g., key servers).

No Auditing Required: The approach does not require auditors to verify correct behavior of infrastructure operators.

No Name Squatting: Users can choose their names and can be prevented from reserving a large number of popular names.

Asynchronous: Trust establishment can occur asynchronously without all conversation participants online.

Scalable: Trust establishment is efficient, with resource requirements growing logarithmically (or smaller) with the the total number of participants in the system.

D. Evaluation

1) Opportunistic Encryption (Baseline): We consider *opportunistic encryption*, in which an encrypted session is established without any key verification, as a baseline. For

TABLE I
TRADE-OFFS FOR COMBINATIONS OF TRUST ESTABLISHMENT APPROACHES. SECURE APPROACHES OFTEN SACRIFICE USABILITY AND ADOPTION.

Scheme	Example	Security Features						Usability									Adoption					
		Network MitM Prevented	Operator MitM Prevented	Operator MitM Detected	Operator Accountability	Key Revocation Possible	Privacy Preserving	Automatic Key Initialization	Low Key Maintenance	Easy Key Discovery	Easy Key Recovery	In-Band	No Shared Secrets	Alert-less Key Renewal	Immediate Enrollment	Inattentive User Resistant	Multiple Key Support	No Service Provider	No Auditing Required	No Name Squatting	Asynchronous	Scalable
Opportunistic Encryption†*	TCPCrypt	-	-	-	-	-	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
+TOFU (Strict)†	-	◐	◐	◐	◐	-	●	●	●	●	-	●	●	-	●	●	-	●	●	●	●	●
+TOFU†*	TextSecure	◐	◐	◐	◐	-	●	●	●	●	●	●	●	-	●	-	-	●	●	●	●	●
Key Fingerprint Verification†*	Threema	●	●	●	●	◐	●	-	-	-	-	-	●	-	-	-	-	●	●	●	●	●
+Short Auth Strings (Out-of-Band)†*	SilentText	●	●	●	●	◐	●	-	-	-	-	-	●	-	-	-	-	-	●	●	-	●
+Short Auth Strings (In-Band/Voice/Video)†*	ZRTP	●	●	●	●	◐	●	-	-	-	-	◐	●	-	-	-	-	●	●	●	-	●
+Socialist Millionaire (SMP)†*	OTR	●	●	●	●	◐	●	-	-	-	-	●	-	-	-	-	-	●	●	●	-	●
+Mandatory Verification†*	SafeSlinger	●	●	●	●	◐	●	-	-	-	-	●	-	-	●	●	-	●	●	●	-	●
Key Directory†*	iMessage	●	-	-	-	●	-	●	●	●	●	●	●	●	●	●	●	-	●	●	◐	●
+Certificate Authority†*	S/MIME	●	-	-	-	-	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
+Transparency Log	-	●	-	◐	◐	-	-	●	●	●	●	●	●	●	◐	●	●	●	-	●	●	●
+Extended Transparency Log†	-	●	-	◐	●	●	-	●	●	●	●	●	●	●	◐	●	●	●	-	●	●	●
+Self-Auditable Log†	CONIKS	●	-	◐	●	●	●	●	●	●	●	●	●	●	●	●	●	●	◐	●	●	●
Web-of-Trust†*	PGP	●	●	●	◐	◐	-	-	-	◐	◐	-	-	-	-	-	●	●	●	●	●	●
+Trust Delegation†*	GnuNS	●	●	●	◐	◐	●	-	-	◐	◐	-	-	-	-	-	●	●	●	●	●	●
+Tracking[*]	Keybase	●	◐	◐	-	◐	-	◐	◐	◐	◐	-	-	-	●	-	●	-	●	◐	●	●
Pure IBC†	SIM-IBC-KMS	●	-	-	-	-	●	●	●	●	●	●	●	●	●	●	-	-	●	-	●	●
+Revocable IBC†	-	●	-	-	-	◐	●	●	●	●	●	●	●	●	●	●	-	-	●	-	●	●
Blockchains[*]	Namecoin	●	●	●	●	-	●	●	◐	●	-	●	●	-	●	●	●	●	-	-	●	-
Key Directory+TOFU+Optional Verification†*	TextSecure	◐	◐	◐	◐	●	-	●	●	●	●	●	●	-	●	-	◐	-	●	●	●	●
Opportunistic Encryption+SMP†*	OTR	◐	◐	◐	◐	●	●	-	●	-	-	◐	◐	-	●	-	●	●	●	●	-	●

● = provides property; ◐ = partially provides property; - = does not provide property; †has academic publication; *end-user tool available

instance, this could be an OTR encryption session without any authentication. The main goal of opportunistic encryption is to counter passive adversaries; active attackers can easily execute MitM attacks. From a usability perspective, this approach is the baseline since it neither places any burden on the user nor generates any new error or warning messages.

2) TOFU: Trust-On-First-Use (TOFU) extends opportunistic encryption by remembering previously seen key material [29]. The *network MitM prevented* and *infrastructure MitM prevented* properties are only partially provided due to the requirement that no attacker is present during the initial connection. TOFU *requires no service provider* since keys can be exchanged by the conversation participants directly. TOFU does not define a mechanism for *key revocation*. TOFU can be implemented in strict and non-strict forms. The strict form fails when the key changes, providing *inattentive user resilience* but preventing *easy key recovery*. The non-strict form prompts users to accept key changes, providing *easy key recovery* at the expense of *inattentive user resilience*.

TOFU-based approaches, like the baseline, do not require any user interaction during the initial contact discovery. This yields good scores for all user-effort properties except for the *key revocation* property, which is not defined, and *alert-less key renewal*, since users cannot distinguish benign key changes from MitM attacks without additional verification methods. For instance, TextSecure shows a warning that a user's key has changed and the user must either confirm the new key or apply manual verification to proceed (shown in Figure 2). If the user chooses to accept the new key immediately, it is possible to perform the verification later. The motivation behind this approach is to provide more transparency for more experienced or high-risk users, while still offering an "acceptable" solution for novice end users. Critically, previous work in the related domain of TLS warnings has shown that frequent warning messages leads to higher click-through rates in dangerous situations, even with experienced users [30].

From an adoption perspective, TOFU performs similarly to the baseline, except for *key recovery* in the strict version and *multiple key support* in both versions. The multiple key support problem arises from the fact that if multiple keys are used, the protocol cannot distinguish between devices. An attacker can claim that a new device, with the attacker's key, is being used.

3) Key Fingerprint Verification: Manual verification requires users to compare some representation of a cryptographic hash of their partners' public keys out of band (e.g., in person or via a separate secure channel).

Assuming the fingerprint check is performed correctly by end users, manual verification provides all desirable security properties with the exception of only partial *key revocation* support, as this requires contacting each communication part-

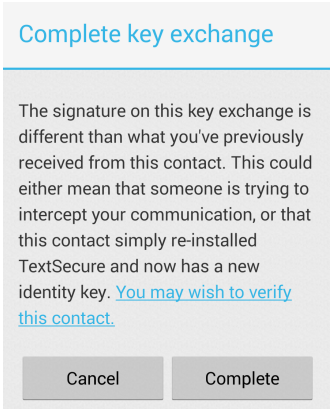


Fig. 2. TextSecure warning for key changes: the user must either accept the new key by selecting "complete", or perform manual verification [31].

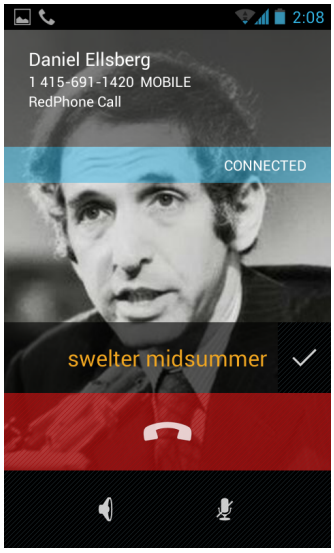


Fig. 3. Users read random words during SAS verification in RedPhone [31].

ner out-of-band. The approaches differ only in their usability and adoption features.

Fingerprint verification approaches introduce severe usability and adoption limitations: users have to perform manual verification before communicating with a new partner (and get them to do the same) to ensure strong authentication. Thus, manual verification does not offer *automatic key initialization*, *easy key discovery*, or *immediate enrollment*. In addition, new keys introduce an alert on key renewal, resulting in a *key maintenance* effort. Fingerprints complicate *multiple key support* since each device might use a different key.

While it is possible to improve the usability of key fingerprint verification by making it optional and combining it with other approaches, we postpone discussion of this strategy until the discussion.

4) Short Authentication Strings: To ease fingerprint verification, shorter strings can be provided to the users for comparison. A short authentication string (SAS) is a truncated cryptographic hash (e.g., 20–30 bits long) of all public parts of the key exchange. It is often represented in a format aimed to be human friendly, such as a short sequence of words. All participants compute the SAS based on the key exchange they observed, and then compare the resulting value with each other. The method used for comparison of the SAS must authenticate the entities using some underlying trust establishment mechanism.

Several encrypted voice channels, including the ZRTP protocol and applications like RedPhone, Signal, and SilentPhone, use the SAS method by requiring participants to read strings aloud [32], [33]. Figure 3 shows an example of SAS verification during establishment of a voice channel in RedPhone. For usability reasons, RedPhone and SilentPhone use random dictionary words to represent the hash. Because these tools require the user to end the established call manually if the verification fails, they are not *inattentive user resistant*.

SAS systems based on voice channels anchor trust in the ability of participants to recognize each other's voices. Users who have never heard each other's voices cannot authenticate using this method. Even for users that are familiar with each other, the security provided by voice identification has been the subject of controversy [34], [35]. Recent work [36] suggests that, with even a small number of samples of a target user's speaking voice, audio samples can be synthesized that are indistinguishable from the genuine user's voice with typical levels of background noise. We should expect that artificial voice synthesis will improve in cost and accuracy, while human auditory recognition will not improve.

For this reason, we consider voice-based SAS verification to be obsolescent from a security standpoint. In Table I, we assume that users verify the SAS with a method providing stronger security (e.g., using audio and video channels with careful inspection during the SAS verification). If the communication channel (e.g., text messaging) does not support a mechanism to establish trust, the SAS must be compared out of band (e.g., as recommended by SilentText).

The SAS approach sacrifices *asynchronicity*, since mutual authentication must be done with all users at the same time. Due to the short size of the SAS, the naïve approach is vulnerable to a MitM attack by an adversary that attempts to select key exchange values that produce a hash collision for the two connections. To mitigate this problem, the attacker can be limited to a single guess by forcing them to reveal their chosen keys before observing the keys of the honest parties. This can be accomplished by requiring that the initiator of the key exchange release a commitment to their key, and then open the commitment after the other party reveals theirs.

5) Secret-based Zero-Knowledge Verification: The Socialist Millionaire Protocol (SMP) is a zero-knowledge proof of knowledge protocol that determines if secret values held by two parties are equal without revealing the value itself. This protocol is used in OTR as the recommended method for user verification [37], [38]. Alice poses a question based on shared knowledge to Bob *in-band* and secretly records her answer.

After Bob answers the question, the two parties perform the SMP to determine if their answers match, without revealing any additional information. Users are expected to choose secure questions with answers based on shared knowledge that attackers would be unable to know or guess.

The SMP used in OTR is performed on a cryptographic hash of the session identifier, the two parties' fingerprints, and their secret answers. This prevents MitM and replay attacks.

Since a MitM must perform an online attack and can only guess once, even low min-entropy secrets achieve strong security [38], [39]. However, use of the SMP sacrifices *asynchronicity* since all participants must be online during the verification. If the protocol fails, the end users do not know whether their answers did not match, or if a MitM attacker exists and has made an incorrect guess.

6) Mandatory Verification: The previously defined verification methods are prone to *inattentive users*. *Mandatory verification* approaches counter user negligence by requiring that users enter the correct fingerprint strings instead of merely confirming that they are correct. Of course, entering the fingerprints takes user effort. In practice, QR-Codes and NFC are popular methods to ease this process.

In SafeSlinger the user must choose the correct answer among three possibilities to proceed [40]. Physically co-located users form a group and exchange ephemeral keys. Each device hashes all received information and displays the hash as a sequence of three common words. Two additional sequences are randomly generated. The users communicate to determine the sequence that is common to all devices and select it to verify the ephemeral keys, preventing users from simply clicking an "OK" button. These keys are then used to exchange contact information within the group with security guarantees including confidentiality and authenticity.

Mandatory verification is a technique that is applied to another trust establishment scheme; the resulting approach inherits the usability properties of the underlying scheme. Incorporating mandatory verification sacrifices *asynchronicity* to ensure *inattentive user resistance*.

7) Authority-based Trust: In authority-based trust schemes, public keys must be vouched for by one or more trusted authorities.

During key initialization, authorities can verify ownership of public keys by the claimed subjects through various means, such as password-based authentication or email validation. The authority then asserts the keys' validity to other users. Two well-known examples of authority-based trust establishment are public-key directories and certificate authority schemes.

A Certificate Authority (CA) may issue signed certificates of public keys to users, who can then present them directly to other users without needing to communicate further with the authority. This model has been widely deployed on the web with the X.509 Public Key Infrastructure (PKIX) for HTTPS. While the S/MIME standard uses this model for secure email, it has seen less widespread deployment than PGP.

Alternatively, users may look up keys directly from an online *public-key directory* over a secure channel. This is common in several proprietary messaging applications such as Apple iMessage and BlackBerry Protected Messenger. In contrast to CA schemes, where the conversation partner directly provides an ownership assertion from the CA, the authority is directly asked for ownership assertions in key directory schemes.

From the security point of view, the two schemes only differ in *key revocation* and *privacy preservation*. While key updates in key directories imply the revocation of old keys, in the CA approach, certificates signed by the authority are trusted by default; revocation lists have to be maintained separately. However, CA-based revocation lists used in web browsers are known to have issues with effectiveness and practicality [24], [41], [42]. Since certificates may be exchanged by peers directly, the CA-based approach can be *privacy preserving*.

With either system, users are vulnerable to MitM attacks by the authority, which can vouch for, or be coerced to vouch for, false keys. This weakness has been highlighted by recent CA scandals [43], [44]. Both schemes can also be attacked if the authority does not verify keys before vouching for them. Authorities in messaging services often rely on insecure SMS or email verification, enabling potential attacks.

The two approaches both support good usability. Well-known systems using public-key directories, such as iMessage, work without any user involvement.

8) Transparency Logs: A major issue with trusted authorities is that they can vouch for fraudulent keys in an attack. The *Certificate Transparency* protocol [45] requires that all issued web certificates are included in a public log.

This append-only log is implemented using a signed Merkle tree with continual proofs of consistency [45]. Certificates are only trusted if they include cryptographic proof that they are present in the log. This ensures that any keys the authority vouches for will be visible in the log and evidence will exist that the authority singed keys used in an attack.

Certificate Transparency is a specific proposal for logging PKIX certificates for TLS, but the general idea can be applied to authority-based trust establishment in secure messaging. We refer to the general concept as *transparency logs* for the remainder of the paper. While there are no known deployments to date, Google plans to adapt *transparency logs* for user keys in End-to-End, its upcoming email encryption tool [46]. In the absence of a concrete definition, we evaluate *transparency logs* based on the certificate transparency protocol.

The main security improvement of the two schemes consists of *operator accountability* and the *detection of operator MitM attacks* after the fact. The remaining security features are inherited from authority-based trust systems.

However, these schemes introduce new and unresolved usability and adoption issues. For instance, the logs must be audited to ensure correctness, negating the *no auditing required* property. The auditing services require gossip protocols to synchronize the view between the monitors and prevent attack bubbles (e.g., where different views are presented to different geographical regions) [45]. Also, since only identity owners are in a position to verify the correctness of their long-term

keys, they share responsibility for verifying correct behavior of the log. Previous research has shown that users often neglect such security responsibilities [30], [47], so this task should be performed automatically by client applications. However, if a client detects a certificate in the log that differs from their version, it is not clear whether the authorities have performed an attack, an adversary has successfully impersonated the subject of the certificate to the authorities, or if the subject actually maintains multiple certificates (e.g., due to installing the app on a second device). Ultimately, end users have to cope with additional security warnings and errors, and it remains to be seen whether they can distinguish between benign and malicious log discrepancies without training. In addition, *transparency logs* might hamper *immediate enrollment* due to delays in log distribution.

Ryan [48] proposed extending the *transparency logs* concept using two logs: one of all certificates in chronological order of issuance, and one of currently valid certificates sorted lexicographically. This enables a form of revocation by making it efficient to query which certificates are currently valid for a given username.

Melara et al. [49] proposed CONIKS, using a series of chained commitments to Merkle prefix trees to build a key directory that is self-auditing, that is, for which individual users can efficiently verify the consistency of their own entry in the directory without relying on a third party. This "self-auditing log" approach makes the system partially have *no auditing required* (as general auditing of non-equivocation is still required) and also enables the system to be *privacy preserving* as the entries in the directory need not be made public. This comes at a mild bandwidth cost not reflected in our table, estimated to be about 10 kilobytes per client per day for self-auditing.

Both Ryan's Extended Certificate Transparency and CONIKS also support a *proof-of-absence*, which guarantees the absence of an identifier or key in the log.

9) Web of Trust: In a *web of trust* scheme, users verify each other's keys using manual verification and, once they are satisfied that a public key is truly owned by its claimed owner, they sign the key to certify this. These certification signatures might be uploaded to key servers. If Alice has verified Bob's key, and Bob certifies that he has verified Carol's key, Alice can then choose to trust Carol's key based on this assertion from Bob. Ideally, Alice will have multiple certification paths to Carol's key to increase her confidence in the key's authenticity.

The user interface for web of trust schemes tends to be relatively complex and has never been fully standardized. The scheme also requires a well-connected social graph, hence the motivation for "key-signing parties" to encourage users to form many links within a common social context.

Assuming that the web of trust model performs correctly, MitM attacks by network and operator adversaries are limited due to distribution of trust. However, since key revocations and new keys might be withheld by key servers, the model offers only partial *operator accountability* and *key revocation*. Since the web of trust model produces a public social graph, it is not *privacy preserving*.

The key-initialization phase requires users to get their keys signed by other keys, so the system does not offer *automatic key initialization*, *alert-less key renewal*, or *immediate enrollment*, and is not *inattentive user resistant*. Because users must participate in key-signing parties to create many paths for trust establishment, users have a high *key maintenance* overhead and a need for an *out-of-band* channel. Even worse, users must understand the details of the PKI and be able to decide whether to trust a key.

PGP typically uses a web of trust for email encryption and signing. In practice, the PGP web of trust consists of one strongly connected component and many unsigned keys or small connected components, making it difficult for those outside the strongly connected component to verify keys [50].

A simplification of the general web of trust framework is SDSI [51] (Simple Distributed Security Infrastructure) later standardized as SPKI [52], [53] (Simple Public Key Infrastructure). With SDSI/SPKI, Bob can assert that a certain key belongs to "Carol" and, if Alice has verified Bob's key as belonging to "Bob", that key will be displayed to Alice as "Bob's Carol" until Alice manually verifies Carol's key herself (which she can then give any name she wants, such as "Carol N."). We refer to these approaches as *trust delegation*. A modern implementation is the GNU Name System (GNS) [54], [55], which implements SDSI/SPKI-like semantics with a key server built using a distributed hash table to *preserve privacy*.

10) Keybase: Keybase is a trust establishment scheme allowing users to find public keys associated with social network accounts. It is designed to be easily integrated into other software to provide username-based trust establishment. If a user knows a social network username associated with a potential conversation partner, they can use Keybase to find the partner's public key.

During key initialization, all users register for accounts with the Keybase server. They then upload a public key and proof that they own the associated private key. Next, the user can associate accounts on social networks or other services with their Keybase account. Each external service is used to post a signature proving that the account is bound to the named Keybase account.

When looking up the key associated with a given user, the Keybase server returns the public key, a list of associated accounts, and web addresses for the external proofs. The client software requests the proofs from the external services and verifies the links. The user is then prompted to verify that the key belongs to the expected individual, based on the verified social network usernames. To avoid checking these proofs for every cryptographic operation, the user can sign the set of accounts owned by their partner. This signature is stored by the Keybase server so that all devices owned by the user can avoid verifying the external proofs again. This process is known as *tracking*. Tracking signatures created by other users are also accessible, providing evidence of account age. Old tracking signatures provide confidence that a user's accounts

have not been recently compromised, but does not protect against infrastructure operator attacks.

Keybase provides partial *operator MitM protection* since attacks require collusion between multiple operators. The scheme also provides easier *key initialization* and *key maintenance* than web-of-trust methods.

11) Identity-Based Cryptography: In identity-based cryptography (IBC), plaintext identifiers (such as email or IP addresses) are mapped to public keys. A trusted third party, the Private Key Generator (PKG), publishes a PKG public key that is distributed to all users of the system. Public keys for an identifier are computed using a combination of the identifier and the PKG public key. The owner of the identity requests the private key for that identity from the PKG while providing proof that they own the identity. The advantage of this system is that users do not need to contact any other entity in order to retrieve the public key of a target user, since the public key is derived from the identifier.

There are two main problems with basic IBC schemes: they lack any *operator MitM prevention* and *key revocation* is not possible. Since the PKG can generate private keys for any user, the operator of the PKG can break the security properties of all conversations. While this fundamental problem cannot be overcome without using hybrid encryption schemes, key revocation support can be added. Revocable IBC approaches [56]–[58] add timestamps to the public key derivation process, regularly refreshing key material.

IBC schemes are normally deployed in situations where the trustworthiness of the PKG operator is assumed, such as in enterprise settings. Few pure-IBC schemes have been proposed for end-user messaging [59], [60].

12) Blockchains: The Bitcoin cryptocurrency utilizes a novel distributed consensus mechanism using pseudonymous "miners" to maintain an append-only log [61]. Voting power is distributed in proportion to computational resources by using a probabilistic proof-of-work puzzle. For the currency application, this log records every transaction to prevent double-spending. Miners are rewarded (and incentivized to behave honestly) by receiving money in proportion to the amount of computation they have performed. The success of Bitcoin's consensus protocol has led to enthusiasm that similar approaches could maintain global consensus on other types of data, such as a mapping of human-readable usernames to keys.

Namecoin, the first fork of Bitcoin, allows users to claim identifiers, add arbitrary data (e.g., public keys) as records for those identifiers, and even sell control of their identifiers to others [62]. Namecoin and similar name-mapping blockchains are denoted by the *blockchain* entry in Table I. Unlike most other schemes, Namecoin is strictly "first-come, first-served", with any user able to purchase ownership of any number of unclaimed names for a small, fixed fee per name. This price is paid in Namecoins—units of currency that are an inherent part of the system. A small maintenance fee is required to maintain control of names, and small fees may be charged by miners to update data or transfer ownership of names.

From the security perspective, blockchain schemes achieve similar results to manual verification, except that instead of exchanging keys, the trust relies on the username only. Once users have securely exchanged usernames, they can reliably fetch the correct keys.

However, various shortcomings arise from a usability and adoption perspective. The primary usability limitation is that if users ever lose the private key used to register their name (which is not the same as the communication key bound to that name), they will permanently lose control over that name (i.e., *key recovery* is not possible). Similarly, if the key is compromised, the name can be permanently and irrevocably hijacked. Thus, the system requires significant *key management* effort and burdens users with high responsibility. If users rely on a web-based service to manage private keys for them, as many do with Bitcoin in practice, the system is no longer truly end-to-end. The system requires users to pay to reserve and maintain names, sacrificing *low key maintenance* and *automatic key initialization*. Users also cannot instantly issue new keys for their identifiers (i.e., there is no *immediate enrollment*) but are required to wait for a new block to be published and confirmed. In practice, this can take 10–60 minutes depending on the desired security level.

On the adoption side, for the system to be completely trustless, users must store the entire blockchain locally and track its progress. Experience from Bitcoin shows that the vast majority of users will not do this due to the communication and storage requirements and will instead trust some other party to track the blockchain for them. This trusted party cannot easily insert spurious records, but can provide stale information without detection. In any case, the system is not highly *scalable* since the required amount of storage and traffic consumption increases linearly with the number of users.

Finally, there are serious issues with *name squatting*, which have plagued early attempts to use the system. Because anybody can register as many names as they can afford, a number of squatters have preemptively claimed short and common names. Given the decentralized nature of blockchains, this is hard to address without raising the registration fees, which increases the burden on all users of the system.

E. Discussion

As Table I makes evident, no trust establishment approach is perfect. While it is common knowledge that usability and security are often at odds, our results show exactly where the trade-offs lie. Approaches either sacrifice security and provide a nearly ideal user experience, or sacrifice user experience to achieve nearly ideal security scores. Authority-based trust (whether in the form of a single authority or multiple providers) and TOFU schemes are the most usable and well-adopted, but only offer basic security properties. Not surprisingly, authority-based trust (particularly app-specific key directories) is predominant among recently developed apps in the wild, as well as among apps with the largest userbases (e.g., iMessage, BlackBerry Protected, TextSecure, and Wickr). By contrast, no approach requiring specific user action to manage keys, such as web-of-trust, Keybase, GNS,

or blockchains, has seen significant adoption among non-technically-minded users.

In practice, we may be faced with the constraint that *none* of the usability properties can be sacrificed in a system that will achieve mass adoption. Higher-security schemes may be useful within organizations or niche communities, but defending against mass surveillance requires a communication system that virtually all users can successfully use. Thus, it may be wise to start from the basic user experience of today's widely deployed communication apps and try to add as much security as possible, rather than start from a desired security level and attempt to make it as simple to use as possible. The recent partnership between WhatsApp and TextSecure [95] is an example of a successful application of this approach.

There appears to be considerable room for security improvements over authoritative key directories even without changes to the user experience. Transparency logs might provide more accountability with no interaction from most users. Because this approach has not yet been deployed, it remains to be seen how much security is gained in practice. The insertion of new keys in the log does not provide public evidence of malicious behavior if insecure user authentication methods (e.g., passwords) are used to authorize key changes, as we fully expect will be the case. Still, the possible loss of reputation may be enough to keep the server honest.

Another promising strategy is a layered design, with basic security provided by a central key directory, additional trust establishment methods for more experienced users (e.g., visual fingerprint verification or QR-codes), and TOFU warning messages whenever contacts' keys have changed. TextSecure and Threema, among others, take such a layered approach (represented by the second-to-last row in Table I). In contrast, OTR uses opportunistic encryption with the ability to perform the SMP to ensure trust (represented by the last row in Table I).

Conversely, the approaches with good security properties should focus on improving usability. There has been little academic work studying the usability of trust establishment. Further research focusing on end-users' mental models and perception for trust establishment could help to develop more sophisticated and understandable approaches.

V. Conversation Security

After *trust establishment* has been achieved, a *conversation security* protocol protects the security and privacy of the exchanged messages. This encompasses how messages are encrypted, the data and metadata that messages contain, and what cryptographic protocols (e.g., ephemeral key exchanges) are performed. A conversation security scheme does not specify a trust establishment scheme nor define how transmitted data reaches the recipient.

In Table II, we compare the features of existing approaches for conversation security. Rows without values in the "group chat" columns can only be used in a two-party setting.

A. Security and Privacy Features

Confidentiality: Only the intended recipients are able to read a message. Specifically, the message must not be readable by a server operator that is not a conversation participant.

Integrity: No honest party will accept a message that has been modified in transit.

Authentication: Each participant in the conversation receives proof of possession of a known long-term secret from all other participants that they believe to be participating in the conversation. In addition, each participant is able to verify that a message was sent from the claimed source.

Participant Consistency: At any point when a message is accepted by an honest party, all honest parties are guaranteed to have the same view of the participant list.

Destination Validation: When a message is accepted by an honest party, they can verify that they were included in the set of intended recipients for the message.

Forward Secrecy: Compromising all key material does not enable decryption of previously encrypted data.

Backward Secrecy: Compromising all key material does not enable decryption of succeeding encrypted data.

Anonymity Preserving: Any anonymity features provided by the underlying transport privacy architecture are not undermined (e.g., if the transport privacy system provides anonymity, the conversation security level does not deanonymize users by linking key identifiers).

Speaker Consistency: All participants agree on the sequence of messages sent by each participant. A protocol might perform consistency checks on blocks of messages during the protocol, or after every message is sent.

Causality Preserving: Implementations can avoid displaying a message before messages that causally precede it.

Global Transcript: All participants see all messages in the same order.

Not all security and privacy features are completely independent. If a protocol does not authenticate participants, then it offers participation repudiation (since no proof of participation is ever provided to anyone). Similarly, no authentication of message origin implies message repudiation as well as message unlinkability. Note that the implications are only one way: repudiation properties might be achieved together with authentication. Additionally, a global transcript order implies both speaker consistency and causality preservation since all transcripts are identical.

Conversation security protocols may provide several different forms of deniability. Based on the definitions from Section II-C, we define the following deniability-related features:

Message Unlinkability: If a judge is convinced that a participant authored one message in the conversation, this does not provide evidence that they authored other messages.

Message Repudiation: Given a conversation transcript and all cryptographic keys, there is no evidence that a given message was authored by any particular user. We assume that the accuser has access to the session keys because it is trivial to deny writing a plaintext message when the accuser cannot demonstrate that the ciphertext corresponds to this plaintext.

TABLE II
CONVERSATION SECURITY PROTOCOLS AND THEIR USABILITY AND ADOPTION IMPLICATIONS. NO APPROACH REQUIRES ADDITIONAL USER EFFORT.

Scheme	Example	Security and Privacy														Adoption					Group Chat				
		Confidentiality	Integrity	Authentication	Participant Consistency	Destination Validation	Forward Secrecy	Backward Secrecy	Anonymity Preserving	Speaker Consistency	Causality Preserving	Global Transcript	Message Unlinkability	Message Repudiation	Particip. Repudiation	Out-of-Order Resilient	Dropped Message Resilient	Asynchronicity	Multi-Device Support	No Additional Service	Computational Equality	Trust Equality	Subgroup Messaging	Contractable	Expandable
TLS+Trusted Server†*	Skype	-	-	-	-	-	-	-	-	-	-	-	●	●	●	●	●	●	●	-	●	●	●	◐	◐
Static Asymmetric Crypto†*	OpenPGP, S/MIME	●	●	●	-	-	-	-	●	-	-	-	-	-	-	●	●	●	●	●					
+IBE†	Wang et al.	-	●	●	-	-	-	-	●	-	-	-	-	-	-	●	●	●	●	-					
+Short Lifetime Keys	OpenPGP Draft	●	●	●	-	-	◐	◐	●	-	-	-	-	-	-	●	●	●	●	-					
+Non-Interactive IBE†	Canetti et al.	●	●	●	-	-	●	-	●	-	-	-	-	-	-	◐	●	●	●	●					
+Puncturable Encryption†	Green and Miers	●	●	●	-	-	●	-	●	-	-	-	-	-	-	●	●	●	●	●					
Key Directory+Short Lifetime Keys†	IMKE	●	●	◐	-	●	◐	◐	-	-	-	-	●	●	●	●	●	-	-	-					
+Long-Term Keys†	SIMPP	●	●	◐	-	●	◐	◐	-	-	-	-	●	●	-	●	●	-	-	-					
Authenticated DH†*	TLS-EDH-MA	●	●	●	●	●	◐	◐	●	-	-	-	●	●	◐	●	●	-	-	●					
+Naïve KDF Ratchet*	SCIMP	●	●	●	●	●	●	◐	●	◐	-	-	●	●	◐	◐	◐	-	-	●					
+DH Ratchet†*	OTR	●	●	●	●	●	◐	●	●	◐	◐	-	●	●	◐	◐	◐	-	-	●					
+Double Ratchet†*	Axolotl	●	●	●	●	●	●	●	●	◐	◐	-	●	●	◐	●	◐	-	-	●					
+Double Ratchet+3DH AKE†*	-	●	●	●	●	●	●	●	◐	◐	◐	-	●	●	●	●	◐	-	-	●					
+Double Ratchet+3DH AKE+Prekeys†*	TextSecure	●	●	●	●	●	●	●	-	◐	●	-	●	●	●	◐	◐	●	-	-					
Key Directory+Static DH+Key Transport†	Kikuchi et al.	●	●	-	-	●	◐	◐	-	-	-	-	●	●	-	●	●	●	-	-	-	-	-	●	●
+Authenticated EDH+Group MAC†	GROK	●	●	◐	-	●	◐	◐	●	-	-	-	●	●	-	●	●	●	-	-	-	-	-	●	●
GKA+Signed Messages+Parent IDs†	OldBlue	●	●	●	●	●	◐	◐	●	●	-	-	-	-	-	●	●	◐	-	●	●	●	-	-	-
Authenticated MP DH+Causal Blocks†*	KleeQ	●	●	◐	◐	◐	●	●	◐	◐	●	-	●	●		●	●	◐	-	●	●	●	-	-	●
OTR Network+Star Topology†	GOTR (2007)	●	●	-	-	-	◐	●	-	-	-	-	●	●	●	◐	●	◐	-	●	-	-	-	●	●
+Pairwise Topology†		●	●	●	●	●	◐	●	●	-	-	-	●	●	●	◐	●	◐	-	●	●	●	●	●	●
+Pairwise Axolotl+Multicast Encryption*	TextSecure	●	●	●	-	●	●	●	-	●	●	-	●	●	●	●	●	●	-	-	●	●	●	●	●
DGKE+Shutdown Consistency Check†	mpOTR	●	●	●	●	●	◐	◐	●	◐	-	-	-	●	●	●	●	-	-	●	●	●	-	-	-
Circle Keys+Message Consistency Check†	GOTR (2013)	●	●	●	●	●	◐	◐	●	●	●	●	●	●	●	●	-	-	-	◐	●	●	-	●	●

● = provides property; ◐ = partially provides property; - = does not provide property; †has academic publication; *end-user tool available

We also assume that the accuser does not have access to the accused participant's long-term secret keys because then it is simple for the accuser to forge the transcript (and thus any messages are repudiable).

Participation Repudiation: Given a conversation transcript and all cryptographic key material for all but one accused (honest) participant, there is no evidence that the honest participant was in a conversation with any of the other participants.

Several additional features are only meaningful for group protocols (i.e., protocols supporting chats between three or more participants):

Computational Equality: All chat participants share an equal computational load.

Trust Equality: No participant is more trusted or takes on more responsibility than any other.

Subgroup messaging: Messages can be sent to a subset of participants without forming a new conversation.

Contractible Membership: After the conversation begins, participants can leave without restarting the protocol.

Expandable Membership: After the conversation begins, participants can join without restarting the protocol.

When a participant joins a secure group conversation, it is desirable for the protocol to compute new cryptographic keys so that the participant cannot decrypt previously sent messages. Likewise, keys should be changed when a participant leaves so that they cannot read new messages. This is trivial to implement by simply restarting the protocol, but this approach is often computationally expensive. Protocols with *expandable / contractible membership* achieve this without restarts.

There are many higher-level security and privacy design issues for secure group chat protocols. For example, the mechanisms for inviting participants to chats, kicking users out of sessions, and chat room moderation are all important choices that are influenced by the intended use cases. We do not cover these features here because they are implemented at a higher level than the secure messaging protocol layer.

B. Usability and Adoption

In classic messaging tools, users must only reason about two simple tasks: sending and receiving messages. However, in secure communication, additional tasks might be added. In old secure messaging systems, often based on OpenPGP, users could manually decide whether to encrypt and/or sign messages. Many studies have shown that this caused usability problems [2]–[5], [21]. However, during our evaluation, we found that most recent secure messenger apps secure all messages by default without user interaction. Since all implementations can operate securely once the trust establishment

is complete, we omit the user-effort columns in Table II. However, we take other usability and adoption factors, such as resilience properties, into account:

Out-of-Order Resilient: If a message is delayed in transit, but eventually arrives, its contents are accessible upon arrival.

Dropped Message Resilient: Messages can be decrypted without receipt of all previous messages. This is desirable for asynchronous and unreliable network services.

Asynchronous: Messages can be sent securely to disconnected recipients and received upon their next connection.

Multi-Device Support: A user can participate in the conversation using multiple devices at once. Each device must be able to send and receive messages. Ideally, all devices have identical views of the conversation. The devices might use a synchronized long-term key or distinct keys.

No Additional Service: The protocol does not require any infrastructure other than the protocol participants. Specifically, the protocol must not require additional servers for relaying messages or storing any kind of key material.

C. Two-party Chat Evaluation

1) Trusted central servers (baseline): The most basic conversation security features that a secure chat protocol can provide are confidentiality and integrity. This can be easily implemented without adversely affecting usability and adoption properties by using a central server to relay messages and securing connections from clients to the central server using a transport-layer protocol like TLS. This also allows the central server to provide presence information. Since this approach does not negatively affect usability, it is no surprise that this architecture has been adopted by some of the most popular messaging systems today (e.g., Skype, Facebook Chat, Google Hangouts) [63]–[67]. We do not consider these protocols further because they allow the central server to decrypt messages and thus do not meet our stronger end-to-end definition of *confidentiality*—that messages cannot be read by anyone except the intended recipient(s). We include this approach as a baseline in Table II in order to evaluate the effects of various designs.

Note that the baseline protocols provide all *repudiation* features, since there is no cryptographic proof of any activity. Additionally, these protocols are highly resilient to errors since there are no cryptographic mechanisms that could cause problems when messages are lost. The use of a trusted central server makes *asynchronicity* and *multi-device support* trivial.

2) Static Asymmetric Cryptography: Another simple approach is to use participants' static long-term asymmetric keypairs for signing and encrypting.

OpenPGP and S/MIME are two well-known and widely implemented standards for message protection, mostly used for email but also in XMPP-based tools [63], [68]–[70].

While this approach provides *confidentiality*, message *authentication*, and *integrity*, it causes a loss of all forms of *repudiation*. Additionally, care must be taken to ensure that *destination validation* and *participant consistency* checks are performed. Without destination validation, *surreptitious forwarding* attacks are possible [71]. Without participant consistency, *identity misbinding* attacks might be possible [72]. Defenses against replay attacks should also be included. These considerations are particularly relevant since the OpenPGP and S/MIME standards do not specify how to provide these features, and thus most implementations remain vulnerable to all of these attacks [68], [70].

To simplify key distribution, several authors have proposed the use of identity-based cryptography in the same setting. The SIM-IBC-KMS protocol acts as an overlay on the MSN chat network with a third-party server acting as the PKG [60]. Messages are encrypted directly using identity-based encryption. The protocol from Wang et al. [73] operates similarly, but distributes the PKG function across many servers with a non-collusion assumption in order to limit the impact of a malicious PKG. These protocols partially sacrifice *confidentiality* since an attacker with access to the PKG private key could surreptitiously decrypt communications.

A second issue with naïve asymmetric cryptography is the lack of *forward* or *backward secrecy*. One way to address this issue is to use keys with very short lifetimes (e.g., changing the key every day). Brown et al. propose several extensions to OpenPGP based on this principle [74]. In the most extreme proposal, conversations are started using long-term keys, but each message includes an ephemeral public key to be used for replies. This method provides *forward* and *backward secrecy* for all messages except those used to start a conversation.

From a usability and adoption perspective, static key approaches achieve the same properties as the baseline. Apart from the non-transparent trust establishment, iMessage is a prominent example of how static asymmetric cryptography can achieve end-to-end conversation security with no changes to the user experience. Since the same long-term keys are used for all messages, *message order resilience*, *dropped message resilience*, *asynchronicity*, and *multi-device-support* are provided. *No additional services* are required.

3) FS-IBE: In traditional PKI cryptography, *forward secrecy* is achieved by exchanging ephemeral session keys or by changing keypairs frequently. The use of key agreement protocols makes *asynchronicity* difficult, whereas frequently changing keypairs requires expensive key distribution. Forward Secure Identity Based Encryption (FS-IBE) allows keypairs to be changed frequently with a low distribution cost. Unlike traditional identity-based encryption schemes, the private key generators (PKG) in FS-IBE are operated by the end users and not by a server. Initially, each participant generates a PKG for an identity-based cryptosystem. Participants generate N private keys (SK_i), one for each time period i, by using their PKG, and then immediately destroy the PKG. Each private key SK_i is stored encrypted by the previous private key SK_{i-1} [15], [75]. The participant then distributes the public key of the PKG. Messages sent to the participant are encrypted for the private key corresponding to the current time period. When a time period concludes, the next secret key is decrypted and the expired key is deleted. Thus, if intermediate keys are compromised, the attacker can only retrieve corresponding

future private keys; *forward secrecy*, but not *backward secrecy*, is provided. In contrast to generating key pairs for each time period, which requires distribution of N keys, only a single public master key is published; however, the generation still needs to be repeated after all time periods expire.

Canetti, Halevi, and Katz were the first to construct a non-interactive forward secrecy scheme based on hierarchical IBE with logarithmic generation and storage costs [75]. In addition, they showed how their scheme can be extended to an unbounded number of periods (i.e., the private keys do not have to be generated in advance), removing the need for *additional services* to distribute new keys at the cost of increasing computational requirements over time. This scheme provides non-interactive *asynchronous forward secrecy* without relying on *additional services*. However, if messages arrive out of order, their corresponding private keys might have already been deleted. As a mitigation, expired keys might be briefly retained, providing partial *out-of-order resilience*.

Green and Miers proposed *puncturable encryption*, a modification of attribute-based encryption (built using a hierarchical IBE scheme) in which each message is encrypted with a randomly chosen "tag" and the recipient can update their private key to no longer be able to decrypt messages with that tag after receipt [76]. This approach provides arbitrary *out-of-order resilience*, although to make the scheme efficient in practice requires periodically changing keys.

Computational costs and storage costs increase over time for both FS-IBE and puncturable encryption, introducing *scalability* concerns. To our knowledge, neither approach has been deployed and they thus merit further development.

4) Short lifetime key directories: Several protocols make use of a central server for facilitating chat session establishment. In these systems, users authenticate to the central server and upload public keys with short lifetimes. The server acts as a key directory for these ephemeral public keys. Conversations are initiated by performing key exchanges authenticated with the short-term keys vouched for by the key directory. Messages are then encrypted and authenticated using a MAC. IMKE [77] is a protocol of this type where the server authenticates users through the use of a password. SIMPP [78]–[80] operates similarly, but uses long-term keys to authenticate instead.

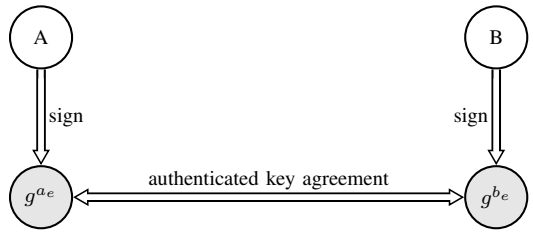
These protocols achieve *confidentiality* and *integrity*, but lack *authentication* of participants since the central server can vouch for malicious short-term keys. Since session keys are exchanged on a per-conversation basis, these protocols achieve *forward* and *backward secrecy* between conversations. Since SIMPP uses signatures during the login procedure, it loses *participation repudiability*; the accuser cannot forge their response to the server's challenge.

5) Authenticated Diffie-Hellman: While the use of central servers for presence information and central authentication is fundamental to systems such as IMKE and SIMPP, there is an alternative class of solutions that instead performs end-to-end authenticated Diffie-Hellman (DH) key exchanges. By default, the authenticated DH key agreement does not rely on central servers. In an authenticated key exchange (AKE) such as authenticated DH, the participants generate an ephemeral session key and authenticate the exchange using their long-term keys. The resulting session key is used to derive symmetric encryption and MAC keys, which then protect messages using an encrypt-then-MAC approach. This basic design provides *confidentiality*, *integrity*, and *authentication*. TLS with an ephemeral DH cipher suite and mutual authentication (TLS-EDH-MA) is a well-known example of this approach. Note that further protections are required during key exchange to protect against *identity misbinding* attacks violating *participant consistency* [38], [72], such as those provided by SIGMA protocols [82].

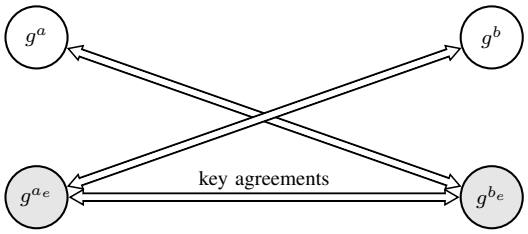
The use of ephemeral session keys provides *forward* and *backward secrecy* between conversations. *Message unlinkability* and *message repudiation* are provided since messages are authenticated with shared MAC keys rather than being signed with long-term keys. At a minimum, messages can be forged by any chat participants. Some protocols, such as OTR, take additional measures, such as publication of MAC keys and the use of malleable encryption, to expand the set of possible message forgers [83]. If the participants simply sign all AKE parameters, then this approach does not provide *participation repudiation*. However, if participants only sign their own ephemeral keys, these signatures can be reused by their conversation partners in forged transcripts. Figure 4a shows the authenticated key exchange used by OTRv1 (more recent versions use a SIGMA key exchange). Conversation partners are able to reuse ephemeral keys signed by the other party in forged transcripts, thereby providing partial *participation repudiation*. OTR users can increase the number of possible forgers by publishing previously signed ephemeral keys in a public location, thereby improving their *participation repudiation*.

Once the AKE has been performed, the encrypt-then-MAC approach allows messages to be exchanged asynchronously with *out-of-order* and *dropped message resilience*. However, since a traditional AKE requires a complete handshake before actual messages can be encrypted, this basic approach requires *synchronicity* during conversation initialization. Additionally, since key agreements can only be performed with connected devices, there is no trivial *multi-device support*.

6) Key Evolution: A desirable property is *forward secrecy* for individual messages rather than for entire conversations. This is especially useful in settings where conversations can last for the lifetime of a device. To achieve this, the session key from the initial key agreement can be evolved over time through the use of a *session key ratchet* [17]. A simple approach is to use key derivation functions (KDFs) to compute future message keys from past keys. This naïve approach, as used in SCIMP [84], provides *forward secrecy*. However, it does not provide *backward secrecy* within conversations; if a key is compromised, all future keys can be derived using the KDF. *Speaker consistency* is partially obtained since messages cannot be surreptitiously dropped by an adversary without also dropping all future messages (otherwise, recipients would not be able to decrypt succeeding messages). If messages are



(a) OTRv1 DH handshake. The session key is derived from the key agreement based on signed ephemeral keys: $s = \text{DH}(g^{a_e}, g^{b_e})$



(b) 3-DH handshake. The session key is a combination of all key agreements: $s = \text{KDF}(\text{DH}(g^{a_e}, g^{b_e})||\text{DH}(g^{a_e}, g^b)||\text{DH}(g^a, g^{b_e}))$

Fig. 4. TLS/OTRv1 handshake vs. 3-DH handshake (figures derived from [81]). Gray nodes represent ephemeral keys, white nodes represent long-term keys.

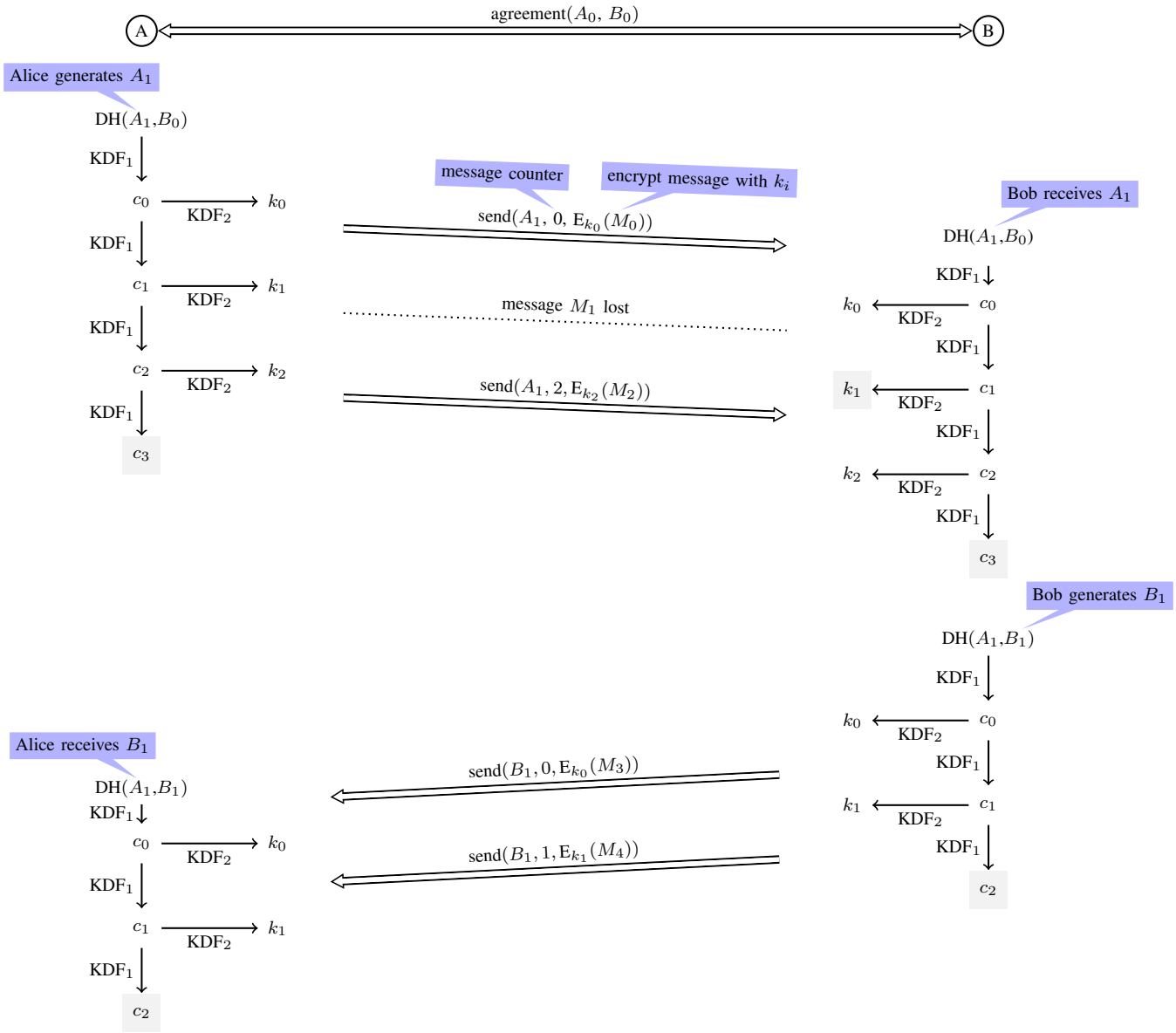


Fig. 5. Simplified version of Axolotl: c_i denote chain keys, k_i message keys, KDF_i arbitrary key derivation functions, E_{k_i} an encryption function using k_i, and $A_i = g^{a_i}$ and $B_i = g^{b_i}$ as public DH values. Gray key nodes denote keys held in memory after Alice receives message M_4.

dropped or arrive out of order, the recipient will notice since the messages are encrypted with an unexpected key. To handle this, the recipient must store expired keys so that delayed or re-transmitted messages can still be decrypted, leaving a larger window of compromise than necessary. Thus, *out-of-order* and *dropped message resilience* are only partially provided.

7) Diffie-Hellman Ratchet: A different ratcheting approach, introduced by OTR, is to attach new DH contributions to messages [83]. With each sent message, the sender advertises a new DH value. Message keys are then computed from the latest acknowledged DH values. This design introduces *backward secrecy* within conversations since a compromised key will regularly be replaced with new key material. *Causality preservation* is partially achieved since messages implicitly reference their causal predecessors based on which keys they use. The same level of *speaker consistency* as the naïve KDF solution can be provided by adding a per-speaker monotonic counter to messages. A disadvantage of the DH ratchet is that session keys might not be renewed for every message (i.e., *forward secrecy* is only partially provided). Like the KDF-based ratchet, the DH ratchet lacks *out-of-order resilience*; if a message arrives after a newly advertised key is accepted, then the necessary decryption key was already deleted.

8) Double-Ratchet (Axolotl): To improve the forward secrecy of a DH ratchet, both ratchet approaches can be combined: session keys produced by DH ratchets are used to seed per-speaker KDF ratchets. Messages are then encrypted using keys produced by the KDF ratchets, frequently refreshed by the DH ratchet on message responses. The resulting *double ratchet*, as implemented by Axolotl [85], provides *forward secrecy* across messages due to the KDF ratchets, but also *backward secrecy* since compromised KDF keys will eventually be replaced by new seeds. To achieve *out-of-order resilience*, the Axolotl ratchet makes use of a second derivation function within its KDF ratchets. While the KDF ratchets are advanced normally, the KDF keys are passed through a second distinct derivation function before being used for encryption.

Figure 5 depicts the double ratchet used in Axolotl. The secondary KDF, denoted as KDF_2, allows the chain keys (c_i) to be advanced without sacrificing forward secrecy; each c_i is deleted immediately after being used to derive the subsequent chain key c_{i+1} and the corresponding message key (k_i) for encryption. If messages arrive out of order, this system provides a mechanism for decrypting the messages without compromising forward secrecy. For example, if Bob is expecting message M_1 and is storing c_1 in memory, but then receives M_2 instead, he uses c_1 to compute k_1, c_2, k_2, and c_3. Bob uses k_2 to decrypt the newly received message, and then he deletes c_1 and c_2 from memory, leaving only k_1 and c_3. When the missing M_1 eventually arrives, Bob can use k_1 to decrypt it directly. However, if an attacker compromises Bob's system at this moment, they will be unable to derive k_2 to decrypt M_2. A similar situation is depicted in Figure 5, where gray key nodes denote keys held in memory after Alice was able to receive M_4.

Axolotl also simplifies the use of its outer DH ratchet. In OTR, a chain of trust, allowing trust in new DH key exchanges to be traced back to the original AKE, is provided through the use of DH key advertisements and acknowledgments. To speed up this process, Axolotl instead derives a *root key* from the initial AKE in addition to the initial DH keys. Each subsequent DH secret is derived by using the sender's latest DH key, the latest DH key received from the other participant, and the current root key. Each time the DH ratchet is advanced, a new root key is derived in addition to a new chain key. Since deriving the chain keys requires knowledge of the current root key, newly received DH keys can be trusted immediately without first sending an acknowledgment. Despite these improvements, the double ratchet still requires *synchronicity* for the initial AKE.

9) 3-DH Handshake: A triple DH (3-DH) handshake is a different AKE scheme that provides stronger *participation repudiation*. Specifically, transcripts of conversations between any two participants can be forged by anyone knowing nothing more than the long-term public keys of the participants. Figure 4b depicts a 3-DH AKE. Triple DH is an implicitly authenticated key agreement protocol—a category that has been extensively examined in the literature [86]–[92]. Note that in this simplified version, if an attacker's ephemeral key was used (possible since ephemeral keys are not signed), the attacker would be able to calculate the session key retrospectively assuming the delayed possession of the corresponding long-term key. Thus, in practice the key exchange requires further protection mechanisms against ephemeral keys chosen by an attacker. Assuming that Alice and Bob both have long-term DH keys g^a and g^b and ephemeral keys g^{a_e} and g^{b_e}, the 3-DH shared secret s is computed as $s = \text{KDF}(\text{DH}(g^{a_e}, g^{b_e})||\text{DH}(g^a, g^{b_e})||\text{DH}(g^{a_e}, g^b))$ [85]. If a secure key derivation function is used, a MitM attacker must either know a and a_e, or b and b_e. Kudla et al. have shown that the 3-DH key exchange provides the same *authentication* level as achieved with the authenticated versions of DH key agreements [93]. 3-DH achieves full *participation repudiation* since anybody is able to forge a transcript between any two parties by generating both a_e and b_e and performing DH key exchanges with g^a and g^b. Assuming that Mallory uses g^m as her long-term DH value and g^{m_e} as her ephemeral key agreement value, and that she knows Alice's long-term DH value g^a, she is able to forge a transcript by calculating $s = \text{KDF}(\text{DH}(g^{a_e}, g^{m_e})||\text{DH}(g^a, g^{m_e})||\text{DH}(g^{a_e}, g^m))$ as the common HMAC and encryption secrets. Mallory can do this without ever actually interacting with Alice. Since the secret is partially derived from the long-term public keys, 3-DH also provides *participant consistency* without the need to explicitly exchange identities after a secure channel has been established. Unfortunately, this also causes a partial loss of *anonymity preservation* since long-term public keys are observable during the initial key agreement (although future exchanges can be protected by using past secrets to encrypt these identities). It is possible to regain *anonymity preservation* by encrypting key identifiers with the given ephemeral keys.

10) Prekeys: While a *double ratchet* does not provide *asynchronicity* by itself, it can be combined with a *prekey* scheme to create an asynchronous version of the protocol. Prekeys are one-time ephemeral public DH contributions that have been uploaded in advance to a central server. Clients can complete a DH key exchange with a message recipient by requesting their next prekey from the server. When combined with a 3-DH exchange, this is sufficient to complete an asynchronous AKE as part of the first message. In comparison to time-window based FS-IBE approaches (cf. Section V-C3), this approach requires the precomputation of a number of ephemeral keys; otherwise, *forward secrecy* is weakened. However, this scheme also permits the destruction of the private ephemeral values immediately after receiving a message using them, instead of keeping a key until a time window expires.

TextSecure [31] is a popular Android app that combines Axolotl, prekeys, and 3-DH to provide an *asynchronous* user experience while sacrificing the *no additional service* property. It has gained considerable attention recently after being incorporated into WhatsApp [94], [95]. Assuming Axolotl is used on two devices, the key material can evolve independently for each device. However, if one of those devices remains offline for a long time, a key compromise on that device is problematic: if the device can use its outdated keys to read messages that were sent when it was offline, then this compromise defeats *forward secrecy*; if the device cannot read the old messages, then the protocol does not achieve complete *multi-device support*. Deciding how long a device may be offline before it can no longer read buffered messages is an adoption consideration requiring further study of user behavior.

D. Group Chat Evaluation

1) Trusted central servers (baseline): The baseline protocol described in Section V-C1, where clients simply connect to a trusted central server using TLS, can trivially support group chats. While it is easy to add and remove group participants in this system, the only thing preventing participants from reading messages sent before or after they are part of the group is the trustworthiness of the server. This fact is indicated by half circles for *expandable / contractible membership*. SILC [96] in its default mode is an example of a protocol using this design. While SILC's architecture involves a network of trusted servers similar to the IRC protocol, for analysis purposes this network can be considered as one trusted entity.

To improve the security and privacy of these systems, participants can simply encrypt and authenticate messages before sending them to the server by using a pre-shared secret key for the group. This approach is useful because it can be applied as a layer on top of any existing infrastructure. SILC has built-in support for this method in its "private mode"; users can provide a password for a channel that is used to derive a pre-shared key unknown to the server. While this design provides *confidentiality* and *integrity*, it does not provide *authentication*.

2) Key transport: Rather than relying on users to exchange a secret password out-of-band, it is far better to automatically exchange a new secret for each conversation. A simple proposed method for doing this is to have one participant generate a session key and securely send it to the other participants. These systems begin by establishing secure channels between participants. The conversation initiator then generates a group key and sends it to the other participants using the pairwise channels. This design provides *forward* and *backward secrecy* since a new group key is randomly generated for each conversation. Due to the use of a group leader, *computational* and *trust equality* are also lost. However, groups are easily *expandable* and *contractible* by having the initiator generate and distribute a new group key.

An early design of this type, proposed by Kikuchi et al. [97], suggests using a key directory to store static DH public keys for users. When group chats are formed, these keys and are used to derive pairwise session keys for the participants. A modified DH exchange is used in order to allow the server to reduce the required computation for the clients. *Participation repudiation* is lost due to the design of the key exchange mechanism, whose security properties have not been rigorously verified. An improvement, used in the GROK protocol [98], is to use standard DH exchanges for the pairwise channels, authenticated using long-term public keys stored in the key directory. This improvement provides *authentication* and *anonymity preservation*, but still suffers from the inherent inequality of key transport approaches.

3) Causality preservation: One issue that is rarely addressed in the design of conversation security protocols is causality preservation. The user interface of the chat application must make design choices such as whether to display messages immediately when they are received, or to buffer them until causal predecessors have been received. However, the conversation security protocol must provide causality information in order to allow the interface to make these choices.

OldBlue [99] is a protocol that provides *speaker consistency* and *causality preservation*. An authenticated group key agreement (GKA) protocol is executed at the start of the conversation. Messages are encrypted with the group key and then signed with long-term asymmetric keys. This approach to signatures eliminates *message repudiation*. To preserve causality, messages include a list of identifiers of messages that causally precede them. The OldBlue protocol conservatively assumes that any message received by a user might influence the messages they subsequently send. Therefore, all received messages are considered to causally precede subsequently transmitted messages. Message identifiers are hashes of the sender, the list of preceding identifiers, and the message contents. When a message has been lost, the client continuously issues resend requests to the other clients.

A different approach is employed by KleeQ [100], a protocol designed for use by multiple trusted participants with tenuous connectivity. An authenticated multi-party DH exchange is performed to initiate the protocol. By authenticating the parameters in a manner similar to OTR, *participation*

repudiation can be provided. The group can be easily *expanded* by incorporating the DH contribution of a new member into the multi-party DH exchange, deriving a new group key. However, the group is not *contractible* without restarting the protocol. When two conversation participants can establish a connection, they exchange the messages that the other is missing using a patching algorithm. All messages are encrypted and authenticated with a MAC using keys derived from the group secret, providing *message repudiation*. Messages are sealed into blocks, which are sequences of messages having the property that no messages could possibly be missing. After each block is sealed, rekeying is performed using the previous keys and the block contents. A mechanism is provided to seal blocks even if some users are inactive in the conversation. *Speaker consistency* is not guaranteed until the messages have been sealed in a block. While participants are *authenticated* during group formation, message contents are not authenticated until after they have been sealed into a block. The block sealing mechanism indirectly provides *participant consistency* and *destination validation*. If malicious participants send differing messages to others, this will be uncovered during the block sealing phase. Manual resolution is required to identify malicious participants.

4) OTR networks: Since OTR [83] provides desirable features for two-party conversations, it is natural to extend it to a group setting by using OTR to secure individual links in a network. A basic strategy is to enlist a trusted entity to relay messages and then secure client links to this entity using OTR. This is the approach taken by the GOTR protocol released in 2007 (we write the year to distinguish it from a different protocol with the same name from 2013). GOTR (2007) [101] selects a participant to act as the relay, forming a star topology of pairwise connections with the selected participant acting as the hub. All *authentication* properties, *speaker consistency*, and *causality preservation* are lost because they do not persist across the relay node. Since the relay server can buffer messages, *asynchronicity* is provided as long as the relay node remains online. All other properties are inherited from OTR. Groups can be *expanded* and *contracted* simply by establishing new OTR connections to the relay.

Instead of using a star topology, pairwise OTR connections between all participants can be established. This approach restores *authentication* and *anonymity preservation*, as well as *equal trust* between members. It is also possible to send messages to *subgroups* by only transmitting the message across selected OTR links. The downside of this approach is that it does not *preserve causality* or provide *speaker consistency*; participants can send different messages to different people. This design also incurs significant computational overhead. It would be desirable to achieve these security properties without this level of additional cost.

5) OTR for groups: Several protocols have been proposed to achieve OTR-like *repudiation* properties for group conversations. The TextSecure protocol can be naturally extended to groups by sending messages to each recipient using the two-party TextSecure protocol [102]. Multicast encryption is used for performance: a single encrypted message is sent to a central server for relaying to recipients while the decryption key for the message is sent pairwise using TextSecure. In practice, the wrapped decryption keys are attached to the same message for broadcasting. It is also possible to accomplish this task using one of the many existing broadcast encryption schemes [103]. This design does not provide any guarantees of *participant consistency*, but it inherits the *asynchronicity* of the two-party TextSecure protocol. *Speaker consistency* and *causality preservation* are achieved by attaching preceding message identifiers to messages. A message identifier is a hash of the sender, the list of preceding identifiers, and the message contents.

A *repudiable* group chat scheme can also be designed by utilizing a deniable group key exchange (DGKE) protocol, as in the mpOTR protocol [104], [105]. When completed, the DGKE provides each participant with a shared secret group key and individual ephemeral signing keys. This information is authenticated with long-term keys in a manner providing *participation repudiation* while still *authenticating* participants—participants receive proof of each other's identities, but this proof cannot be used to convince outsiders. All parties must be online to complete the DGKE, so the protocol does not support *asynchronicity*. Messages are encrypted with the shared group key and signed with the ephemeral keys. The ephemeral signatures provide proof of authorship to others in the group but, because outsiders cannot be certain that these ephemeral signing keys correspond to specific long-term keys, *message repudiation* is preserved. However, since all messages from an individual are signed with the same (ephemeral) key, the protocol does not have *message unlinkability*. When the conversation has concluded, each participant hashes all messages received from each other participant. The hashes are then compared to ensure that everyone received the same set of messages, providing *speaker consistency*. If this check fails, messages must be individually compared to uncover discrepancies. This approach, where a consistency check is performed only once at the conclusion of the conversation, does not work if a network adversary disconnects users from the conversation before the consistency check can be completed. In this worst-case scenario, the only information received by users is that something went wrong at some point during the protocol, but nothing more specific. Unfortunately, in many scenarios it is unclear how users should respond to this limited information. In this scheme, *subgroup messaging* is not possible since all messages share a single encryption key. The group is also not *expandable* or *contractible* without performing a new DGKE.

A completely different approach is taken by the GOTR (2013) protocol. GOTR (2013) [14] is built using a "hot-pluggable" group key agreement (GKA) protocol, allowing members to join and drop out of the conversation with little overhead. This system involves the use of "circle keys": sets of public keys having the property that a shared secret key can be computed by anyone with a private key matching a public key in the set. The key exchange mechanism in this protocol is relatively complex; we refer the interested reader

to the original publication for details [14]. Pairwise secure channels are set up between participants to send consistency check messages. These consistency channels have the effect of providing *global transcript order*, but all participants are required to be online to receive messages. The system otherwise provides features similar to mpOTR but with *flexible group membership* and *message unlinkability*.

E. Discussion

Similar to our study of trust establishment, Table II makes immediately clear that no conversation security protocol provides all desired properties. Since most of the properties in the table are not mutually exclusive, however, there is significant room for improvement by combining protocol designs and this should be seen as a tangible and important call to action for the research community.

Sadly, the most widely adopted solutions also have the worst security and privacy properties, with most non-security-focused applications providing only basic static asymmetric cryptography. This does not appear to be due to the usability drawbacks of the more secure protocols: once the trust establishment has been done, all of the conversation security approaches we studied can be automated without any additional effort for the user. An exception is enabling *asynchronous* communication while still providing *forward* and *backward secrecy*; the only solution for this problem that appears to have any significant deployment in practice is the prekeys approach implemented by TextSecure. This requires relatively complicated infrastructure compared to a simple key server, introduces problems for *multi-device support*, and is prone to denial-of-service attacks if it is used in anonymous communication. This approach is poorly studied in the academic literature. The FS-IBE scheme discussed in Section V-C3 promises to resolve the issues of server complexity and denial of service, but introduces new challenges such as scalability and performance issues [75]. Unlike prekeys (Section V-C10), this scheme has received a considerable amount of follow-up research and academic citations, but we are unaware of any practical tool implementing it. In addition, a time-window based FS-IBE scheme requires holding the ephemeral keys for a certain amount of time to allow decryption of delayed messages. One possible mitigation is to rely on an additional server maintaining window counters where every window number is used once, analogous to the prekeys approach. Improving the practicality of FS-IBE and puncturable encryption schemes warrants further research.

Another outstanding concern that limits adoption of secure conversation security protocols is the limited support for multiple devices. Despite a vast number of users owning multiple devices, only the most insecure protocols support this property without requiring users to perform pairing procedures. Device pairing has proved extremely difficult for users in practice [106], [107] and allowing users to register multiple devices with distinct keys is a major usability improvement. Although extremely difficult, implementing usable device pairing is not necessarily an insurmountable problem. Additional work in this area is needed.

When it comes to group chat properties, we can identify several areas for improvement in Table II. Classic protocols often do not provide *participant consistency* or *destination validation*, making them potentially vulnerable to surreptitious forwarding or identity misbinding attacks. However, these are sometimes addressed in concrete implementations. The double ratchet used in Axolotl improves *forward secrecy* with low cost in performance, implementation complexity, and resilience, but it has not yet been thoroughly evaluated in an academic context. Additionally, decentralized group chat systems inherently permit a participant to send different messages to different people. Due to network conditions, users can also end up observing significantly different transcripts. Despite these intrinsic weaknesses, surprisingly few protocols explicitly consider *speaker consistency* or *causality preservation*. The recently proposed (n+1)sec protocol [108] is an example of new work in this area. (n+1)sec builds off of Abdalla et al.'s flexible group key exchange protocol [109] to provide a DGKE and checks for transcript consistency.

Existing solutions achieve mixed results concerning *repudiation*. It is often debated whether repudiation is a desirable feature and, if it is, whether or not it is worth pursuing. There are situations in which the mere suspicion of authoring a given message is potentially harmful to an author; in these cases, repudiation is not useful, and participants should make use of a protocol providing anonymity instead. Even in scenarios where repudiation is traditionally thought to be useful, such as during criminal trials in societies requiring proof of authorship "beyond a reasonable doubt", there are no prominent examples of repudiable messaging properties influencing legal decisions. Nonetheless, if it is possible to maintain repudiation within a secure messaging protocol without substantial cost, we believe that it remains a desirable property. Users typically think of real-world conversations as "off-the-record", so it is natural to desire (or expect) this property from a secure messaging protocol. For the definitions of *participation repudiation* and *message repudiation* used in this work, the two-party protocols based on authenticated DH key exchanges and the OTR-like group protocols provide inexpensive solutions.

There are also additional adoption constraints imposed by many modern secure group chat protocols. Group protocols often choose to employ either a trusted participant or an additional service to improve protocol performance, which can lead to security concerns or introduce additional costs for deployment. Very few group protocols support *subgroup messaging* or changing group membership after the conversation has started without incurring the substantial costs of a new protocol run. Additionally, many proposed designs require synchronicity in order to simplify their protocols, which largely precludes their use on current mobile devices.

VI. Transport Privacy

The transport privacy layer defines how messages are exchanged, with the goal of hiding message metadata such as

TABLE III
TRANSPORT PRIVACY SCHEMES. EVERY PRIVACY-ENHANCING APPROACH CARRIES USABILITY AND/OR ADOPTION COSTS.

Scheme	Example	Privacy					Usability					Adoption						
		Sender Anonymity	Recipient Anonymity	Particip. Anonymity	Unlinkability	Global Adv. Resistant	Contact Discovery	No Message Delays	No Message Drops	Easy Initialization	No Fees Required	Topology Independent	No Additional Service	Spam/Flood Resistant	Low Storage	Low Bandwidth	Low Computation	Asynchronous	Scalable
Store-and-Forward[†*]	Email/XMPP	-	-	-	-	-	●	◐	●	●	●	●	-	-	●	●	●	●	●
+DHT Lookup[†*]	Kademlia	◐	◐	-	-	-	●	◐	●	●	●	●	●	◐	●	◐	●	●	●
Onion Routing+Message Padding[†*]	Tor	●	-	●	●	-	-	◐	●	●	●	●	◐	-	●	●	●	-	●
+Hidden Services[*]	Ricochet	●	●	●	◐	-	-	◐	●	●	●	●	◐	-	●	●	●	-	●
+Inbox Servers[†]	-	●	-	●	●	-	-	◐	●	●	●	●	-	-	●	●	●	●	●
+Random Delays[†*]	Mixminion	●	-	●	●	◐	-	-	●	●	●	●	-	-	◐	●	●	●	●
+Hidden Services+Delays+Inboxes+ZKGP[*]	Pond	●	-	●	●	◐	-	-	●	●	●	●	-	●	◐	●	●	●	●
DC-Nets[†*]	-	●	●	-	-	●	-	-	●	●	●	-	●	-	●	●	●	-	-
+Silent Rounds[†]	Anonycaster	●	●	-	-	●	-	-	●	●	●	-	●	◐	●	●	●	-	-
+Shuffle-Based DC-Net+Leader[†]	Dissent	●	●	-	-	●	-	-	●	●	●	-	●	●	●	●	●	-	-
+Shuffle-Based DC-Net+Anytrust Servers[†]	Verdict	●	●	-	-	●	-	-	●	●	●	-	-	●	●	●	●	-	◐
Message Broadcast[†]	-	-	●	●	●	●	●	●	●	●	-	●	●	-	-	-	◐	-	-
+Blockchain	-	◐	●	●	●	●	●	-	-	●	-	●	◐	●	-	-	-	●	-
PIR[*]	Pynchon Gate	-	●	●	●	●	●	-	●	◐	●	●	-	-	-	◐	◐	●	◐

● = provides property; ◐ = partially provides property; - = does not provide property; [†]has academic publication; [*]end-user tool available

the sender, receiver, and conversation to which the message belongs. Some transport privacy architectures impose topological structures on the conversation security layer, while others merely add privacy to data links between entities. The transport privacy schemes may also be used for privacy-preserving contact discovery. In this section, we compare approaches for transport privacy in terms of the privacy features that they provide, as well as usability concerns and other factors that limit their adoption. Table III compares the various schemes.

A. Privacy Features

We make the distinction between *chat messages*, which are the user-generated payloads for the messaging protocol to exchange, and *protocol messages*, which are the underlying data transmissions dictated by the upper protocol layers. We define following privacy properties:

Sender Anonymity: When a chat message is received, no non-global entities except for the sender can determine which entity produced the message.

Recipient Anonymity: No non-global entities except the receiver of a chat message know which entity received it.

Participation Anonymity: No non-global entities except the conversation participants can discover which set of network nodes are engaged in a conversation.

Unlinkability: No non-global entities except the conversation participants can discover that two protocol messages belong to the same conversation.

Global Adversary Resistant: Global adversaries cannot break the anonymity of the protocol.

B. Usability Properties

Contact Discovery: The system provides a mechanism for discovering contact information.

No Message Delays: No long message delays are incurred.

No Message Drops: Dropped messages are retransmitted.

Easy Initialization: The user does not need to perform any significant tasks before starting to communicate.

No Fees Required: The scheme does not require monetary fees to be used.

C. Adoption Properties

Topology Independent: No network topology is imposed on the conversation security or trust establishment schemes.

No Additional Service: The architecture does not depend on availability of any infrastructure beyond the chat participants.

Spam/Flood Resistant: The availability of the system is resistant to denial-of-service attacks and bulk messaging.

Low Storage Consumption: The system does not require a large amount of storage capacity for any entity.

Low Bandwidth: The system does not require a large amount of bandwidth usage for any entity.

Low Computation: The system does not require a large amount of processing power for any entity.

Asynchronous: Messages sent to recipients who are offline will be delivered when the recipient reconnects, even if the sender has since disconnected.

Scalable: The amount of resources required to maintain system availability scales linearly with the number of users.

D. Evaluation

1) Store-and-Forward (baseline): To evaluate the effectiveness and costs of different transport privacy architectures in Table III, we compare the solutions to a baseline. For the baseline protocol, we assume a simple store-and-forward messaging protocol. This method is employed by email and text messaging, causing minor message delays and storage

requirements for intermediate servers. Since email headers contain sender and recipient information, a simple store-and-forward mechanism does not provide any privacy properties.

2) Peer-to-Peer Solutions: Instead of relying on centralized servers for message storage and forwarding, peer-to-peer based schemes try to establish a direct message exchange between the participants. Since end users frequently change their IP addresses, these systems often use Distributed Hash Tables (DHTs) to map usernames to IP addresses without a central authority. Examples of popular DHT systems are Chord, Kademlia (used by BitTorrent), and GNUnet [110]–[112]. In addition to acting as an IP address lookup table, it is possible to store exchanged messages directly in a DHT. Various query privacy extensions have been proposed to prevent other users from learning what data is being requested. They can be used in advanced DHT overlays allowing anonymous queries and message exchange [113]–[115].

Global network adversaries are still able to see the traffic flow between participants during message exchange. Thus, clients have two options to protect the data flow: fake message transmissions, or use anonymization techniques. End-user clients might use services such as onion routing, which is evaluated in the next section, to hide their identities.

From the usability and adoption perspective, peer-to-peer networks require a synchronous environment. DHTs can be used for *contact discovery* with *easy initialization*, but they introduce *message delays* and *message drops*.

In practice, various end-user applications use the BitTorrent or GNUnet networks for their secure messaging service. For instance, Tox, Bleep, and other messengers use BitTorrent for message exchange. The GNUnet Name Service (GNS) offers privacy-preserving name queries for contact discovery [54].

3) Onion Routing: Onion routing is a method for communicating through multiple proxy servers that complicates end-to-end message tracing [116]. In onion routing, senders send messages wrapped in multiple layers of encryption through preselected paths—called *circuits*—of proxy servers. These servers unwrap layers of encryption until the original message is exposed, at which point it is relayed to the final destination. Each node in the path only knows the immediate predecessor and successor in the path. The routing process adds some latency to messages, but otherwise retains the baseline usability features. An onion routing protocol, such as the widely used Tor protocol [117], provides *sender anonymity*, *participant anonymity*, and *unlinkability* against network attackers with limited scope.

Global network adversaries are still able to break the anonymity properties of simple onion routing designs by performing statistical analysis incorporating features such as content size, transmission directions, counts, and timing, among others. The success of such an adversary can be limited by individually eliminating these features. Protection can be added, for example, by introducing random delays to transmissions. The longer the allowed delays, the less statistical power is available to the adversary. Of course, this imposes potentially long *message delays* and *additional storage requirements* for relays, making it unusable for synchronous instant messaging.

Unfortunately, random delays do not completely defeat global adversaries. The only way to do so is to make transmission indistinguishable from no transmission (e.g., by saturating the bandwidth of all connections). However, in practice, this is likely infeasible. Additionally, concrete implementations such as Tor often provide weaker anonymity guarantees than idealized onion routing schemes. Several prominent attacks against Tor have been based on implementation defects, limited resources, weaknesses introduced by performance trade-offs, and predictability of the content being transmitted [118]–[121]. Adoption of onion routing is limited by the requirement to establish a large network of nodes to provide a sufficient anonymity set and cover traffic.

In the default mode, onion routing systems do not attempt to provide *recipient anonymity*. However, Tor can be extended to achieve this property using an extension called *hidden services*. To create a Tor hidden service, the recipient uses traditional Tor circuits to upload a set of *introduction points* and a public key to a public database. The sender later uses a circuit to acquire this information from the database. The sender chooses a *rendezvous point* and sends it along with a nonce to the recipient through an introduction point. The recipient and sender both connect to the rendezvous point, which uses the nonce to establish a communication channel by matching up the sender and recipient circuits. Without the aforementioned defenses, this scheme is also vulnerable to global adversaries.

To provide *asynchronous* communication support, store-and-forward servers can be incorporated into the onion routing model. Each user is associated with a Tor hidden service that remains online. To send a message, the sender constructs a circuit to the recipient's server and transmits the message. Users periodically poll their own servers to determine if any messages are queued. Ricochet is an example of this approach [122].

Pond uses this design for its transmission architecture [123] but adds random delays between connections, all of which transmit the same amount of data, to weaken statistical analysis by network adversaries. While some protection against global network adversaries is provided by the onion routing model, this protection is strictly weaker than Tor because connections are made directly from senders to recipient mail servers. This design requires *storage commitments* by servers and also introduces very *high latency*.

Without additional protections, this scheme is also highly vulnerable to denial-of-service attacks because connection delays and fixed transmission sizes artificially limit bandwidth to very low levels. Pond addresses this by requiring users to maintain group lists secured by zero-knowledge-group-proof schemes (ZKGP). This way, recipients can upload contact lists without revealing their contacts. Simultaneously, senders can authenticate by providing zero-knowledge proofs that they are in this list. The BBS signature scheme [124] is currently used by Pond to achieve this. Additional work is underway to provide a similar mechanism in more efficient manner by using one-time delivery tokens [123].

The ZKGP schemes used by Pond are related to secret handshake protocols. Secret handshakes enable authentication between parties that share some attributes, while keeping identities hidden from others [125].

4) DC-nets: Dining Cryptographer networks (DC-nets) are anonymity systems that are often compared to onion routing schemes. Since they are primarily used as a general-purpose transport privacy mechanism, many varieties have been proposed [126]–[134]. In our brief overview, we focus on recently introduced schemes that explicitly list secure messaging as an intended use case.

DC-nets are group protocols that execute in rounds. At the start of each round, each participant either submits a secret message or no message. At the end of the round, all participants receive the xor of all secret messages submitted, without knowing which message was submitted by which participant. In this way, DC-nets provide *sender anonymity* while also achieving *global adversary resilience*—no statistical analysis can reveal the sender of a message. *Recipient anonymity* can be achieved by using the protocol to publish an ephemeral public key. Messages encrypted with this key are then sent and, since the owner of the matching private key is unknown, the participant able to decrypt the messages cannot be determined. Since messages are sent in rounds, DC-nets add message latency and do not support *asynchronous* communication; dropped messages prevent the protocol from advancing. Messages are easily *linked* by observing which network nodes participate in a round. Additionally, DC-nets have limited *scalability* due to requiring pairwise communication.

The basic DC-net design has a problem with collisions: if two parties submit a message in the same round, the result will be corrupted. A malicious participant can exploit this to perform an anonymous denial-of-service attack by submitting garbled messages each round. Worse still, an active network attacker can also perform this attack by perturbing transmitted bits. There are several approaches to mitigate this problem. Anonycaster [131] adds pseudorandomly determined "silent rounds" where all members know that no message should be contributed. Receipt of a message during a silent round indicates a denial-of-service attack by an active network attacker. However, malicious participants can still launch attacks by sending garbled messages only during non-silent rounds.

Dissent [130], [132], [134] and Verdict [133] take a different approach by constructing a DC-net system through the use of a verifiable shuffle and bulk transfer protocol. Shuffle-based DC-nets can include a blame protocol to pinpoint the entity that caused a round to fail. Dissent appoints one participant as a leader to manage round timing, the blame protocol, and exclusion of disconnected members from rounds, thereby restoring support for *asynchronicity*. Verdict uses an alternative approach where the DC-net protocol is executed by a set of central servers that clients connect to, providing greater *scalability* and maintaining security as long as any one server is honest.

While DC-nets are primarily a transport privacy mechanism, they are distinguished from other schemes by their use of rounds and the fact that every network node is also a participant in the conversation. When using DC-nets to transmit higher-level conversation security protocols, it is important for designers to consider how these properties affect the overall security of the scheme (e.g., the use of synchronous rounds creates a *global transcript*, and the details of the DC-net key exchanges may cause a loss of *participation repudiation*).

5) Broadcast Systems: There is a simple approach to providing recipient anonymity against all attackers, including global adversaries: distributing messages to everyone. This approach provides *recipient anonymity*, *participation anonymity*, and *unlinkability* against all network attackers. It also provides a natural way to *discover contacts* because requests for contact data can be sent to the correct entity without knowledge of any addressing information. However, there are some serious downsides that hinder adoption: broadcasting a message to everyone in the network requires high bandwidth, there is no support for *asynchronicity*, and it has extreme *scalability* issues. Additionally, it is easy to attack the availability of the network through flooding. Bitmessage [135], a broadcast-based transport system, either *requires monetary fees* or a proof of work to send messages in order to limit spam, adding *computation requirements* and *message delays* as represented by the *blockchains* row in Table III. It is also possible to alleviate *scalability* problems by clustering users into smaller broadcast groups, at the cost of reduced anonymity set sizes.

6) PIR: Private Information Retrieval (PIR) protocols allow a user to query a database on a server without enabling the server to determine what information was retrieved. These systems, such as the Pynchon Gate [136], can be used to store databases of message inboxes, as well as databases of contact information. *Recipient anonymity* is provided because, while the server knows the network node that is connecting to it, the server cannot associate incoming connections with protocol messages that they retrieve. For the same reason, the protocols offer *participation anonymity* and *unlinkability*. By default, there is no mechanism for providing *sender anonymity*. These systems are naturally *asynchronous*, but they result in *high latency* because inboxes must be polled. The servers also incur a *high storage cost* and are vulnerable to flooding attacks.

PIR schemes can also be used to privately retrieve presence information, which can be useful for augmenting synchronous protocols lacking this capability. For example, DP5 [137] uses PIR to privately provide presence data for a secure messaging protocol; DP5 does not facilitate message transmission itself.

PIR implementations can be divided into computational schemes, which rely on computational limitations of the server, information-theoretic schemes, which rely on non-collusion of servers, and hybrid schemes that combine properties of both. There is also a class of PIR schemes that make use of secure coprocessors, which require users to trust that the coprocessor has not been compromised. PIR implementations differ in their bandwidth, computation, and initialization costs, as well as their scalability. PIR is not widely adopted in practice because one or more of these costs is usually prohibitively high.

E. Discussion

If messages are secured end to end, leaving only identifiers for anonymous inboxes in the unencrypted header, then metadata is easily hidden from service operators. Assuming that each message is sent using new channels, an adversary is not able to link single messages to conversations. However, such schemes introduce adoption and usability issues; they are prone to spam, flooding, and denial-of-service attacks, or require expensive operations such as zero-knowledge authentication, posing barriers to adoption. Worse still, hiding metadata from a global adversary in these schemes necessitates serious usability problems such as long delays.

In contrast, decentralized schemes either exhibit synchronicity issues or have serious scalability problems. Most decentralized projects, especially BitTorrent-based approaches, lack detailed documentation that is required for complete evaluation. Some tools claiming to hide metadata only do so in the absence of global network adversaries, which recent surveillance revelations suggest may exist.

Broadcast-based schemes can achieve the best privacy properties, but exhibit serious usability issues, such as lost or delayed messages, in addition to apparently intractable scalability issues. Even if anonymous transmission schemes are adopted, they require a large user base to provide a high degree of anonymity, potentially discouraging early adopters. Finally, care must be taken when selecting a conversation security scheme to avoid leaking cryptographic material or identifiers that might lead to deanonymization.

VII. Concluding Remarks

The vast majority of the world's electronic communication still runs over legacy protocols such as SMTP, SMS/GSM, and centralized messengers, none of which were designed with end-to-end security in mind. We encourage the research community to view the high-profile NSA revelations in the United States as a golden opportunity to encourage the adoption of secure systems in their place. As the old adage goes: "never let a crisis go to waste".

Unfortunately, while we have seen considerable progress in practical tools over the past two years, there is little evidence suggesting that academic research on secure messaging has dramatically increased. This is unfortunate for two reasons: First, many interesting problems of practical importance remain unresolved. In particular, apparent practical deployment constraints, including limitations for asynchronous communication, multiple independent devices, and zero user effort, are not fully appreciated in most published research papers. Second, many theoretically *solved* problems are not considered in practice, whether because developers are unaware of their existence, or because they cannot immediately translate the cryptographic publications into working systems.

Our effort to systematize existing knowledge on secure messaging suggests three major problems must be resolved: *trust establishment*, *conversation security*, and *transport privacy*. The schemes can largely be chosen independently, yielding a vast design space for secure messaging systems. Yet we also caution against a proliferation of a-la-carte systems for specific niches. The main purpose of communication networks is to interact with others and there is considerable value in having a small number of popular protocols that connect a large number of users. Currently, many people fall back to email despite its insecurity.

We also note that, disappointingly, most of the exciting progress being made right now is by protocols that are either completely proprietary (e.g., Apple iMessage) or are open-source but lack a rigorously specified protocol to facilitate interoperable implementations (e.g., TextSecure). An open standard for secure messaging, combining the most promising features identified by our survey, would be of immense value.

Inevitably, trade-offs have to be made. We conclude that secure approaches in trust establishment perform poorly in usability and adoption, while more usable approaches lack strong security guarantees. We consider the most promising approach for trust establishment to be a combination of central key directories, transparency logs to ensure global consistency of the key directory's entries, and a variety of options for security-conscious users to verify keys out of band to put pressure on the key directory to remain honest.

Our observations on the conversation security layer suggest that asynchronous environments and limited multi-device support are not fully resolved. For two-party conversation security, per-message ratcheting with resilience for out-of-order messages combined with deniable key exchange protocols, as implemented in Axolotl, can be employed today at the cost of additional implementation complexity with no significant impact on user experience. The situation is less clear for secure group conversations; while no approach is a clear answer, the TextSecure group protocol provides pragmatic security considerations while remaining practical. It may be possible to achieve other desirable properties, such as participant consistency and anonymity preservation, by incorporating techniques from the other systems. It remains unclear exactly what consistency properties are required to match users' expectations and usability research is sorely needed to guide future protocol design. Finally, transport privacy remains a challenging problem. No suggested approaches managed to provide strong transport privacy properties against global adversaries while also remaining practical.

We consider this systematization to be a useful assessment of published research and deployment experience. We have uncovered many open challenges and interesting problems to be solved by the research community. The active development of secure messaging tools offers a huge potential to provide real-world benefits to millions; we hope this paper can serve as an inspiration and a basis for this important goal.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments, and Trevor Perrin and Henry Corrigan-Gibbs for their discussion and excellent feedback. Joseph Bonneau is supported by a Secure Usability Fellowship from the Open Technology Fund and Simply Secure. We

gratefully acknowledge the support of NSERC and the Ontario Research Fund.

REFERENCES

[1] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith, "SoK: Secure Messaging," in *Symposium on Security and Privacy*. IEEE, 2015.

[2] A. Whitten and J. D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," in *Security Symposium*. USENIX, 1999.

[3] S. L. Garfinkel and R. C. Miller, "Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express," in *Symposium on Usable Privacy and Security*. ACM, 2005, pp. 13–24.

[4] S. L. Garfinkel, D. Margrave, J. I. Schiller, E. Nordlander, and R. C. Miller, "How to Make Secure Email Easier To Use," in *SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2005, pp. 701–710.

[5] K. Renaud, M. Volkamer, and A. Renkema-Padmos, "Why Doesn't Jane Protect Her Privacy?" in *Privacy Enhancing Technologies*. Springer, 2014, pp. 244–262.

[6] M. Madden. (2014, Nov) Public Perceptions of Privacy and Security in the Post-Snowden Era. [Online]. Available: http://www.pewinternet.org/2014/11/12/public-privacy-perceptions/

[7] GoldBug Project. GoldBug - Secure Instant Messenger. [Online]. Available: http://goldbug.sourceforge.net/

[8] Telegram. Telegram Messenger. [Online]. Available: https://telegram.org/

[9] Wickr. Wickr – Top Secret Messenger. [Online]. Available: https://wickr.com/

[10] Confide. Confide - Your Off-the-Record Messenger. [Online]. Available: https://getconfide.com/

[11] Electronic Frontier Foundation. Secure Messaging Scorecard. [Online]. Available: https://www.eff.org/secure-messaging-scorecard

[12] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, "Deniable Encryption," in *Advances in Cryptology–CRYPTO*. Springer, 1997, pp. 90–104.

[13] Y. Dodis, J. Katz, A. Smith, and S. Walfish, "Composability and On-Line Deniability of Authentication," in *Theory of Cryptography*. Springer, 2009, pp. 146–162.

[14] H. Liu, E. Y. Vasserman, and N. Hopper, "Improved Group Off-the-Record Messaging," in *Workshop on Privacy in the Electronic Society*. ACM, 2013, pp. 249–254.

[15] R. Anderson, "Two Remarks on Public Key Cryptology," 1997, available from https://www.cl.cam.ac.uk/users/rja14.

[16] R. Shirey, "Internet Security Glossary," RFC 2828 (Informational), Internet Engineering Task Force, 2000, obsoleted by RFC 4949. [Online]. Available: http://tools.ietf.org/rfc/rfc2828.txt

[17] O. W. Systems. Advanced cryptographic ratcheting. [Online]. Available: https://whispersystems.org/blog/advanced-ratcheting/

[18] S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer, "The Emperor's New Security Indicators: An evaluation of website authentication and the effect of role playing on usability studies," in *Symposium on Security and Privacy*. IEEE, 2007, pp. 51–65.

[19] R. Stedman, K. Yoshida, and I. Goldberg, "A User Study of Off-the-Record Messaging," in *Symposium on Usable Privacy and Security*. ACM, 2008, pp. 95–104.

[20] S. Clark, T. Goodspeed, P. Metzger, Z. Wasserman, K. Xu, and M. Blaze, "Why (Special Agent) Johnny (Still) Can't Encrypt: A Security Analysis of the APCO Project 25 Two-way Radio System," in *Security Symposium*. USENIX, 2011.

[21] S. Fahl, M. Harbach, T. Muders, M. Smith, and U. Sander, "Helping Johnny 2.0 to Encrypt His Facebook Conversations," in *Symposium on Usable Privacy and Security*. ACM, 2012.

[22] S. Ruoti, N. Kim, B. Burgon, T. van der Horst, and K. Seamons, "Confused Johnny: When Automatic Encryption Leads to Confusion and Mistakes," in *Symposium on Usable Privacy and Security*. ACM, 2013.

[23] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes," in *Symposium on Security and Privacy*. IEEE, 2012. [Online]. Available: http://www.jbonneau.com/doc/BHOS12-IEEESP-quest_to_replace_passwords.pdf

[24] J. Clark and P. C. van Oorschot, "SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements," in *Symposium on Security and Privacy*. IEEE, 2013, pp. 511–525.

[25] J. Nielsen, "Finding Usability Problems Through Heuristic Evaluation," in *SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1992, pp. 373–380.

[26] ——, "Usability inspection methods," in *Conference Companion on Human Factors in Computing Systems*. ACM, 1994, pp. 413–414.

[27] B. E. John and M. M. Mashyna, "Evaluating a Multimedia Authoring Tool with Cognitive Walkthrough and Think-Aloud User Studies," DTIC, Tech. Rep., 1995.

[28] S. Sheng, L. Broderick, C. A. Koranda, and J. J. Hyland, "Why Johnny Still Can't Encrypt: Evaluating the Usability of Email Encryption Software," in *Symposium On Usable Privacy and Security*. ACM, 2006.

[29] D. Wendlandt, D. G. Andersen, and A. Perrig, "Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing," in *Annual Technical Conference*. USENIX, 2008, pp. 321–334.

[30] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor, "Crying Wolf: An Empirical Study of SSL Warning Effectiveness," in *Security Symposium*. USENIX, 2009, pp. 399–416.

[31] O. W. Systems. Open WhisperSystems. [Online]. Available: https://whispersystems.org/

[32] P. Zimmermann, A. Johnston, and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP," RFC 6189 (Informational), Internet Engineering Task Force, 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6189.txt

[33] E. A. Blossom, "The VP1 Protocol for Voice Privacy Devices Version 1.2," Communication Security Corporation, 1999.

[34] P. Gupta and V. Shmatikov, "Security Analysis of Voice-over-IP Protocols," in *Computer Security Foundations Symposium*. IEEE, 2007, pp. 49–63.

[35] M. Petraschek, T. Hoeher, O. Jung, H. Hlavacs, and W. N. Gansterer, "Security and Usability Aspects of Man-in-the-Middle Attacks on ZRTP," *Journal of Universal Computer Science*, vol. 14, no. 5, pp. 673–692, 2008.

[36] M. Shirvanian and N. Saxena, "Wiretapping via Mimicry: Short Voice Imitation Man-in-the-Middle Attacks on Crypto Phones," in *Conference on Computer and Communications Security*. ACM, 2014, pp. 868–879.

[37] M. Jakobsson and M. Yung, "Proving Without Knowing: On Oblivious, Agnostic and Blindfolded Provers," in *Advances in Cryptology–CRYPTO*. Springer, 1996, pp. 186–200.

[38] C. Alexander and I. Goldberg, "Improved User Authentication in Off-The-Record Messaging," in *Workshop on Privacy in the Electronic Society*. ACM, 2007, pp. 41–47.

[39] F. Boudot, B. Schoenmakers, and J. Traoré, "A fair and efficient solution to the socialist millionaires' problem," *Discrete Applied Mathematics*, vol. 111, no. 1, pp. 23–36, 2001.

[40] M. Farb, Y.-H. Lin, T. H.-J. Kim, J. McCune, and A. Perrig, "SafeSlinger: Easy-to-Use and Secure Public-Key Exchange," in *International Conference on Mobile Computing & Networking*. ACM, 2013, pp. 417–428.

[41] M. Marlinspike, "More tricks for defeating SSL in practice," in *Black Hat USA*, 2009.

[42] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov, "The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software," in *Conference on Computer and Communications Security*. ACM, 2012, pp. 38–49.

[43] VASCO. (2011, Sep) DigiNotar reports security incident. [Online]. Available: https://www.vasco.com/company/about_vasco/press_room/news_archive/2011/news_diginotar_reports_security_incident.aspx

[44] A. Langley. (2013) Enhancing digital certificate security. [Online]. Available: http://googleonlinesecurity.blogspot.de/2013/01/enhancing-digital-certificate-security.html

[45] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," RFC 6962 (Experimental), Internet Engineering Task Force, 2013. [Online]. Available: http://tools.ietf.org/rfc/rfc6962.txt

[46] Google. End-To-End. [Online]. Available: https://github.com/google/end-to-end

[47] B. B. Anderson, C. B. Kirwan, J. L. Jenkins, D. Eargle, S. Howard, and A. Vance, "How Polymorphic Warnings Reduce Habituation in the Brain—Insights from an fMRI Study," in *CHI*. ACM, 2015.

[48] M. D. Ryan, "Enhanced Certificate Transparency and End-to-end Encrypted Mail," in *Network and Distributed System Security Symposium*. Internet Society, 2014.

[49] M. S. Melara, A. Blankstein, J. Bonneau, M. J. Freedman, and E. W. Felten, "CONIKS: A Privacy-Preserving Consistent Key Service for Secure End-to-End Communication," Cryptology ePrint Archive Report 2014/1004, 2014. [Online]. Available: https://eprint.iacr.org/2014/1004

[50] A. Ulrich, R. Holz, P. Hauck, and G. Carle, "Investigating the OpenPGP Web of Trust," in *Computer Security–ESORICS*. Springer, 2011, pp. 489–507.

[51] R. L. Rivest and B. Lampson, "SDSI – A Simple Distributed Security Infrastructure," 1996, manuscript.

[52] C. M. Ellison, "Establishing Identity Without Certification Authorities," in *Security Symposium*. USENIX, 1996, pp. 67–76.

[53] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, "SPKI Certificate Theory," RFC 2693 (Experimental), Internet Engineering Task Force, 1999. [Online]. Available: http://tools.ietf.org/rfc/rfc2693.txt

[54] M. Wachs, M. Schanzenbach, and C. Grothoff, "A Censorship-Resistant, Privacy-Enhancing and Fully Decentralized Name System," in *Cryptology and Network Security*. Springer, 2014, pp. 127–142.

[55] ——, "On the Feasibility of a Censorship Resistant Decentralized Name System," in *Foundations and Practice of Security*. Springer, 2014, pp. 19–30.

[56] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based Encryption with Efficient Revocation," in *Conference on Computer and Communications Security*. ACM, 2008, pp. 417–426.

[57] B. Libert and D. Vergnaud, "Adaptive-ID Secure Revocable Identity-Based Encryption," in *Topics in Cryptology–CT-RSA*. Springer, 2009, pp. 1–15.

[58] C. Wang, Y. Li, X. Xia, and K. Zheng, "An Efficient and Provable Secure Revocable Identity-Based Encryption Scheme," *PLOS ONE*, 2014.

[59] A. Thukral and X. Zou, "Secure Group Instant Messaging Using Cryptographic Primitives," in *Networking and Mobile Computing*. Springer, 2005, pp. 1002–1011.

[60] Z. Bin, F. Meng, X. Hou-ren, and H. Dian-you, "Design and Implementation of Secure Instant Messaging System Based on MSN," in *International Symposium on Computer Science and Computational Technology*, vol. 1. IEEE, 2008, pp. 38–41.

[61] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008, self-published.

[62] N. Project. Namecoin. [Online]. Available: https://namecoin.info/

[63] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," RFC 6120 (Proposed Standard), Internet Engineering Task Force, 2011. [Online]. Available: http://tools.ietf.org/rfc/rfc6120.txt

[64] S. Schrittwieser, P. Frühwirt, P. Kieseberg, M. Leithner, M. Mulazzani, M. Huber, and E. R. Weippl, "Guess Who's Texting You? Evaluating the Security of Smartphone Messaging Applications," in *Network and Distributed System Security Symposium*. Internet Society, 2012.

[65] Microsoft. (2014) Does Skype use encryption? [Online]. Available: https://support.skype.com/en/faq/FA31/does-skype-use-encryption

[66] Google. (2014) Google Hangouts - Video Conferencing & Meeting for Business. [Online]. Available: https://www.google.com/work/apps/business/products/hangouts/

[67] Facebook. (2014) Facebook Help Center. [Online]. Available: https://www.facebook.com/help/

[68] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, "OpenPGP Message Format," RFC 4880 (Proposed Standard), Internet Engineering Task Force, 1999, updated by RFC 5581. [Online]. Available: http://tools.ietf.org/rfc/rfc4880.txt

[69] D. Fomin and Y. Leboulanger. Gajim, a Jabber/XMPP client. [Online]. Available: https://gajim.org/

[70] B. Ramsdell and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification," RFC 5751 (Proposed Standard), Internet Engineering Task Force, 2010. [Online]. Available: http://tools.ietf.org/rfc/rfc5751.txt

[71] D. Davis, "Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML," in *Annual Technical Conference, General Track*. USENIX, 2001, pp. 65–78.

[72] W. Diffie, P. C. van Oorschot, and M. J. Wiener, "Authentication and Authenticated Key Exchanges," *Designs, Codes and Cryptography*, vol. 2, no. 2, pp. 107–125, 1992.

[73] C.-J. Wang, W.-L. Lin, and H.-T. Lin, "Design of An Instant Messaging System Using Identity Based Cryptosystems," in *International Conference on Emerging Intelligent Data and Web Technologies*. IEEE, 2013, pp. 277–281.

[74] I. Brown, A. Back, and B. Laurie, "Forward Secrecy Extensions for OpenPGP," Draft, Internet Engineering Task Force, 2002. [Online]. Available: https://tools.ietf.org/id/draft-brown-pgp-pfs-03.txt

[75] R. Canetti, S. Halevi, and J. Katz, "A Forward-Secure Public-Key Encryption Scheme," in *Advances in Cryptology–EUROCRYPT*. Springer, 2003, pp. 255–271.

[76] M. Green and I. Miers, "Forward Secure Asynchronous Messaging from Puncturable Encryption," in *Symposium on Security and Privacy*. IEEE, 2015.

[77] M. Mannan and P. C. van Oorschot, "A Protocol for Secure Public Instant Messaging," in *Financial Cryptography and Data Security*. Springer, 2006, pp. 20–35.

[78] C.-H. Yang and T.-Y. Kuo, "The Design and Implementation of a Secure Instant Messaging Key Exchange Protocol," 2007, available from http://crypto.nknu.edu.tw/psnl/publications/2007Technology_SIMPP.pdf.

[79] C.-H. Yang, T.-Y. Kuo, T. Ahn, and C.-P. Lee, "Design and Implementation of a Secure Instant Messaging Service based on Elliptic-Curve Cryptography," *Journal of Computers*, vol. 18, no. 4, pp. 31–38, 2008.

[80] C.-P. Lee and C.-H. Yang, "Design and Implement of a Secure Instant Messaging Service with IC Card," 2009, available from http://crypto.nknu.edu.tw/psnl/publications/2009CPU_SIMICCard.pdf.

[81] O. W. Systems. Simplifying OTR deniability. [Online]. Available: https://whispersystems.org/blog/simplifying-otr-deniability

[82] H. Krawczyk, "SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and its Use in the IKE protocols," in *Advances in Cryptology–CRYPTO 2003*. Springer, 2003, pp. 400–425.

[83] N. Borisov, I. Goldberg, and E. Brewer, "Off-the-Record Communication, or, Why Not To Use PGP," in *Workshop on Privacy in the Electronic Society*. ACM, 2004, pp. 77–84.

[84] V. Moscaritolo, G. Belvin, and P. Zimmermann, *Silent Circle Instant Messaging Protocol Protocol Specification*, 2012.

[85] T. Perrin. (2013) Axolotl Ratchet. [Online]. Available: https://github.com/trevp/axolotl/wiki

[86] A. J. Menezes, M. Qu, and S. A. Vanstone, "Some New Key Agreement Protocols Providing Implicit Authentication," in *Selected Areas in Cryptography*, 1995, pp. 22–32.

[87] R. Ankney, D. Johnson, and M. Matyas, "The Unified Model," Contribution to X9F1, 1995.

[88] N. S. Agency. SKIPJACK and KEA Algorithm Specifications. [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/documents/skipjack/skipjack.pdf

[89] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, "An Efficient Protocol for Authenticated Key Agreement," *Designs, Codes and Cryptography*, vol. 28, no. 2, pp. 119–134, 2003.

[90] H. Krawczyk, "HMQV: A High-Performance Secure Diffie-Hellman Protocol," in *Advances in Cryptology–CRYPTO*. Springer, 2005, pp. 546–566.

[91] K. Lauter and A. Mityagin, "Security Analysis of KEA Authenticated Key Exchange Protocol," in *Public Key Cryptography – PKC 2006*. Springer, 2006, pp. 378–394.

[92] B. LaMacchia, K. Lauter, and A. Mityagin, "Stronger Security of Authenticated Key Exchange," in *Provable Security*. Springer, 2007, pp. 1–16.

[93] C. Kudla and K. G. Paterson, "Modular Security Proofs for Key Agreement Protocols," in *Advances in Cryptology–ASIACRYPT*. Springer, 2005, pp. 549–565.

[94] T. Frosch, C. Mainka, C. Bader, F. Bergsma, J. Schwenk, and T. Holz, "How Secure is TextSecure?" Cryptology ePrint Archive Report 2014/904, 2014. [Online]. Available: https://eprint.iacr.org/2014/904

[95] O. W. Systems. Open Whisper Systems partners with WhatsApp to provide end-to-end encryption. [Online]. Available: https://whispersystems.org/blog/whatsapp/

[96] SILC Project. SILC – Secure Internet Live Conferencing. [Online]. Available: http://silcnet.org/

[97] H. Kikuchi, M. Tada, and S. Nakanishi, "Secure Instant Messaging Protocol Preserving Confidentiality against Administrator," in *International Conference on Advanced Information Networking and Applications*. IEEE, 2004, pp. 27–30.

[98] J. A. Cooley, R. I. Khazan, B. W. Fuller, and G. E. Pickard, "GROK: A Practical System for Securing Group Communications," in *International Symposium on Network Computing and Applications*. IEEE, 2010, pp. 100–107.
[99] M. D. Van Gundy and H. Chen, "OldBlue: Causal Broadcast In A Mutually Suspicious Environment (Working Draft)," 2012, available from http://matt.singlethink.net/projects/mpotr/oldblue-draft.pdf.
[100] J. Reardon, A. Kligman, B. Agala, and I. Goldberg, "KleeQ: Asynchronous Key Management for Dynamic Ad-Hoc Networks," University of Waterloo, Tech. Rep., 2007.
[101] J. Bian, R. Seker, and U. Topaloglu, "Off-the-Record Instant Messaging for Group Conversation," in *International Conference on Information Reuse and Integration*. IEEE, 2007, pp. 79–84.
[102] O. W. Systems. Private Group Messaging. [Online]. Available: https://whispersystems.org/blog/private-groups/
[103] A. Fiat and M. Naor, "Broadcast Encryption," in *Advances in Cryptology–CRYPTO'93*. Springer, 1994, pp. 480–491.
[104] I. Goldberg, B. Ustaoğlu, M. D. Van Gundy, and H. Chen, "Multi-party Off-the-Record Messaging," in *Conference on Computer and Communications Security*. ACM, 2009, pp. 358–368.
[105] M. Van Gundy, "Improved Deniable Signature Key Exchange for mpOTR," 2013, available from http://matt.singlethink.net/projects/mpotr/improved-dske.pdf.
[106] R. Kainda, I. Flechais, and A. W. Roscoe, "Usability and Security of Out-Of-Band Channels in Secure Device Pairing Protocols," in *Symposium on Usable Privacy and Security*. ACM, 2009.
[107] B. Warner. (2014) Pairing Problems. [Online]. Available: https://blog.mozilla.org/warner/2014/04/02/pairing-problems/
[108] eQualit.ie. (2015) (n+1)sec. [Online]. Available: learn.equalit.ie/wiki/Np1sec
[109] M. Abdalla, C. Chevalier, M. Manulis, and D. Pointcheval, "Flexible Group Key Exchange with On-Demand Computation of Subgroup Keys," in *Progress in Cryptology–AFRICACRYPT*. Springer, 2010, pp. 351–368.
[110] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," *SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.
[111] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.
[112] J. A. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The BitTorrent P2P File-Sharing System: Measurements and Analysis," in *Peer-to-Peer Systems IV*. Springer, 2005, pp. 205–216.
[113] A. Kapadia and N. Triandopoulos, "Halo: High-Assurance Locate for Distributed Hash Tables," in *Network and Distributed System Security Symposium*. Internet Society, 2008.
[114] Q. Wang and N. Borisov, "Octopus: A Secure and Anonymous DHT Lookup," in *International Conference on Distributed Computing Systems*. IEEE, 2012, pp. 325–334.
[115] M. Backes, I. Goldberg, A. Kate, and T. Toft, "Adding Query Privacy to Robust DHTs," in *Symposium on Information, Computer and Communications Security*. ACM, 2012, pp. 30–31.
[116] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous Connections and Onion Routing," *Selected Areas in Communications*, vol. 16, no. 4, pp. 482–494, 1998.
[117] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," DTIC, Tech. Rep., 2004.
[118] S. J. Murdoch and G. Danezis, "Low-Cost Traffic Analysis of Tor," in *Symposium on Security and Privacy*. IEEE, 2005, pp. 183–195.
[119] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-Resource Routing Attacks Against Tor," in *Workshop on Privacy in the Electronic Society*. ACM, 2007, pp. 11–20.
[120] N. S. Evans, R. Dingledine, and C. Grothoff, "A Practical Congestion Attack on Tor Using Long Paths," in *Security Symposium*. USENIX, 2009, pp. 33–50.
[121] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website Fingerprinting in Onion Routing Based Anonymization Networks," in *Workshop on Privacy in the Electronic Society*. ACM, 2011, pp. 103–114.
[122] Ricochet Project. (2014) Anonymous and serverless instant messaging that just works. [Online]. Available: https://github.com/ricochet-im/ricochet
[123] A. Langley. Pond. [Online]. Available: https://pond.imperialviolet.org/
[124] D. Boneh, X. Boyen, and H. Shacham, "Short Group Signatures," in *Advances in Cryptology–CRYPTO*. Springer, 2004, pp. 41–55.
[125] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H.-C. Wong, "Secret Handshakes from Pairing-Based Key Agreements," in *Symposium on Security and Privacy*. IEEE, 2003, pp. 180–196.
[126] D. Chaum, "The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability," *Journal of Cryptology*, vol. 1, no. 1, pp. 65–75, 1988.
[127] M. Waidner and B. Pfitzmann, "The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability," in *Advances in Cryptology–EUROCRYPT*. Springer, 1989.
[128] S. Goel, M. Robson, M. Polte, and E. Sirer, "Herbivore: A Scalable and Efficient Protocol for Anonymous Communication," Cornell University, Tech. Rep. TR2003-1890, 2003.
[129] P. Golle and A. Juels, "Dining Cryptographers Revisited," in *Advances in Cryptology–EUROCRYPT*. Springer, 2004, pp. 456–473.
[130] H. Corrigan-Gibbs and B. Ford, "Dissent: Accountable Anonymous Group Messaging," in *Conference on Computer and Communications Security*. ACM, 2010, pp. 340–350.
[131] C. C. Head, "Anonycaster: Simple, Efficient Anonymous Group Communication," 2012, available from https://blogs.ubc.ca/computersecurity/files/2012/04/anonycaster.pdf.
[132] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in Numbers: Making Strong Anonymity Scale," in *Conference on Operating Systems Design and Implementation*. USENIX, 2012, pp. 179–182.
[133] H. Corrigan-Gibbs, D. I. Wolinsky, and B. Ford, "Proactively Accountable Anonymous Messaging in Verdict," arXiv e-prints, Tech. Rep. arXiv:1209.4819, 2012.
[134] E. Syta, H. Corrigan-Gibbs, S.-C. Weng, D. Wolinsky, B. Ford, and A. Johnson, "Security Analysis of Accountable Anonymity in Dissent," *Transactions on Information and System Security*, vol. 17, no. 1, p. 4, 2014.
[135] B. Project. Bitmessage. [Online]. Available: https://bitmessage.org/
[136] L. Sassaman, B. Cohen, and N. Mathewson, "The Pynchon Gate: A Secure Method of Pseudonymous Mail Retrieval," in *Workshop on Privacy in the Electronic Society*. ACM, 2005, pp. 1–9.
[137] N. Borisov, G. Danezis, and I. Goldberg, "DP5: A Private Presence Service," CACR, Tech. Rep. 2014-10, 2014.

2.2 Implications and Future Directions

This chapter provides a systematic evaluation of secure messaging protocols and tools, identifying gaps in security, usability, and adoption. Here, the results are critically evaluated in terms of their implications for the broader adoption of secure messaging tools.

The open research was systematized into three problem areas: conversation security (Section V, Table II), trust establishment (Section IV, Table I), and transport privacy (Section VI, Table III). The study concludes that conversation security, where once trust is established, is on a good path to being solved in practice even when implementing advanced security features such as perfect forward secrecy. Not only their security features have been proven, but popular messaging systems, namely `WhatsApp` and `iMessage` have already implemented the `Signal` protocol consisting of the double ratchet, pre-keys and 3DH (and their improved successor versions [27]) as their default [32].

The study emphasizes that the other two areas, trust establishment and transport privacy, still remain a big challenge in balancing trade-offs between privacy/security and usability. This concludes that no single approach perfectly balances both.

In current widespread systems, trust is predominantly established through authority-based authentication and the leverage of *trust on first use* (TOFU) schemes, which contradicts the end-to-end security assumptions in case the authorities are involved in the surveillance. The study concludes with two potential solutions for future work, which have been addressed by the community and this thesis.

a) Transparency logs, as with key transparency, might provide more provider accountability without interaction from most users similar to [33–35]. Notably, the work by Fahl, Dechand, et al., co-authored by the author of this thesis [6], demonstrates a system for Android applications featuring a transparency log, which could be adapted for secure messaging platforms by replacing apps with user public keys. Later in 2023, Meta announced the deployment of key transparency for `WhatsApp` [36].

b) additional trust establishment methods for more experienced users as with improved key fingerprint verification schemes and the usage of QR-codes); This can be accompanied by warning messages whenever contacts' keys have changed, which motivated our research in Chapter 3. `WhatsApp` and `iMessage` both offer the extended trust establishment as hidden in the expert settings so that advanced users can go beyond authority-based trust to verify their counterparts manually.

This systematization identifies and discusses additional usability challenges, including the limited understanding and mental models of non-technical users when engaging with end-to-end security, which guided the research presented in Chapter 4. It also highlights issues related to key storage and management, thereby informing the investigations discussed in Chapter 5.

CHAPTER 3

Empirical Study of Textual Key Fingerprints

Building on the systematization and identified open challenges in Chapter 2, trust establishment remains a big challenge in secure messaging. Traditionally, tools relying on end-to-end security, such as SSH and OpenPGP, relied on a manual key-fingerprint comparison which also became an extended feature in popular messaging systems. This chapter builds on these findings by empirically evaluating alternative textual fingerprint representations to address these usability and security gaps.

This study focuses on the usability and security of manual key-fingerprint comparison, a foundational authentication component in secure messaging and other cryptographic systems. With the advent of end-to-end encryption in mainstream applications like `WhatsApp`, the question of improving fingerprint representations for better usability and security has gained renewed attention.

This work presents the first large-scale empirical evaluation of different fingerprint representation formats, examining how design choices impact both security and user experience. For this, the study evaluates six textual key-fingerprint representation schemes, including hexadecimal, numeric, and language-based options such as word lists and sentence-based encodings. The online study involved 1047 participants, where participants experienced our designed simulated attacks and consisted of two parts: (1) an experiment where it was measured how fast and accurate participants perform for different schemes under a realistic attack scenario, and (2) a survey about their perception and sentiment.

The research demonstrates that the widely used hexadecimal fingerprint representation, which is the current standard in tools like SSH and OpenPGP, performs significantly worse than alternative approaches in terms of both usability and attack detection. Specifically, over 10 % of attacks went undetected with hexadecimal representation.

The study introduces and validates novel fingerprint representation approaches, particularly showing that sentence-based encodings achieve the highest attack detection rate (97.97 %) while maintaining good usability scores. The research also found that numeric representations outperform traditional alphanumeric and especially hexadecimal approaches.

The research offers actionable recommendations for improving fingerprint representations in secure messaging applications, demonstrating that significant usability and security improvements are possible through better representation choices without requiring new hardware or complex software changes.

3.1 Peer-Reviewed Publication 2 | An Empirical Study of Textual Key Fingerprints

Authors' Contributions

The work presented in this chapter is based on the publication at the 25th USENIX Security Symposium (USENIX Security 16, TX, Texas) [1] and is included in its entirety in original form in this section. The authors' contributions that are relevant to the contents of this chapter are as follows:

- **Sergej Dechand** I served as the main author and was primarily responsible for the overall conceptualization, design, and execution of the study. This included designing the research methodology, implementing the fingerprint representations, building the evaluation test bench, and conducting the comprehensive quantitative study with 1,047 participants. I oversaw all critical aspects of the project, ensuring its successful completion and scientific rigor.
- **Dominik Schürmann** contributed to implementation and verification efforts, especially for the performed attacks, but also integrating fingerprint representations into OpenKeychain. Dominik's contributions provided helpful technical context and complemented the study's practical applications.
- **Yasemin Acar** provided expert support on statistical framework and methodology and analysis of the quantitative results ensuring their statistical validity. Additionally, she offered valuable feedback throughout the project that enhanced the overall quality of the research.
- **Caroline Busse, Sascha Fahl and Matthew Smith** provided valuable feedback throughout all project phases, actively contributing to discussions regarding the research direction. Their input included offering structural and conceptual advice to enhance the quality, clarity, and presentation of the publication.

An Empirical Study of Textual Key-Fingerprint Representations

Sergej Dechand
USECAP, University of Bonn

Dominik Schürmann
IBR, TU Braunschweig

Karoline Busse
USECAP, University of Bonn

Yasemin Acar
CISPA, Saarland University

Sascha Fahl
CISPA, Saarland University

Matthew Smith
USECAP, University of Bonn

Abstract

Many security protocols still rely on manual fingerprint comparisons for authentication. The most well-known and widely used key-fingerprint representation are hexadecimal strings as used in various security tools. With the introduction of end-to-end security in WhatsApp and other messengers, the discussion on how to best represent key-fingerprints for users is receiving a lot of interest.

We conduct a 1047 participant study evaluating six different textual key-fingerprint representations with regards to their performance and usability. We focus on textual fingerprints as the most robust and deployable representation.

Our findings show that the currently used hexadecimal representation is more prone to partial preimage attacks in comparison to others. Based on our findings, we make the recommendation that two alternative representations should be adopted. The highest attack detection rate and best usability perception is achieved with a sentence-based encoding. If language-based representations are not acceptable, a simple numeric approach still outperforms the hexadecimal representation.

1 Introduction

Public key cryptography is a common method for authentication in secure end-to-end communication and has been a part of the Internet throughout the last two decades [7, 11]. While security breaches have shown that systems based on centralized trusted third parties such as Certificate Authorities and Identity Based Private Key Generators are prone to targeted attacks [42], decentralized approaches such as Web of Trust and Namecoin struggle with beeing adopted in practice due to usability issues [7, 13, 30]. Certificate transparency systems, such as CONIKS and others [24, 39, 27], aim to solve a subset of these issues by providing an auditable directory of all user keys. Still, manual key verification, i. e., the link between public keys and the entities, such as hostnames or people, remains a challenging subject, especially in decentralized systems without pre-defined authorities, such as SSH, OpenPGP, and secure messaging [12, 41].

Many traditional authentication systems still rely on manual key-fingerprint comparisons [17]. Here, key-fingerprints are generated by encoding the (hashed) public key material into a human readable format, usually encoded in hexadecimal representation. A variety of alternatives such as QR Codes, visual fingerprints, Near Field Communication (NFC), and Short Authentication Strings (SAS) have been proposed. Most of these systems offer specific benefits, e. g., QR codes and NFC do not require users to compare strings, but they also come with specific disadvantages, e. g., they require hardware and software support on all devices. While advances are being made in these areas, the text-based representation is still the dominant form in most applications.

However, due to the recent boom of secure messaging tools, the debate of how to best represent and evaluate textual fingerprints has opened up again and there are many very active discussions among security experts [28, 33]. In April 2016, WhatsApp serving over one billion users enabled end-to-end encryption as default by implementing the Signal protocol. Key verification is optional and can be done by using QR codes or comparing numeric representations, in their case 60-digit numbers [43]. However, it is not clear whether their solution is more usable than traditional representations.

In this paper, we present an evaluation of different textual key-fingerprint representation schemes to aid in the secure messenger discussion. The requirements posed to the developers are as follows:

- The fingerprint representation scheme should provide offline support and work asynchronously. One reason for this is that fingerprints are often printed on business cards or exchanged by third parties.

- The fingerprint should be transferable via audio channels, e. g., it should be possible to compare fingerprint over the phone.
- The representation scheme should be as technically inclusive as possible. No special hardware or software should be required to verify the fingerprints: both require a concerted and coordinated effort between many actors to get enough coverage for a comparison mechanism to be worthwhile for users to adopt.
- The representation should be as inclusive as possible, i. e., excluding as few people with sensory impairments (visual, color, audio, etc.) as possible.

The above requirements exclude many proposed representation schemes and offer an explanation why they have not seen any adoption outside of academia. For this reason, we focus exclusively on textual fingerprint representations in our study. Textual key-fingerprints do not require hardware support and work in synchronous and asynchronous scenarios, i. e., they can be compared via voice or printed on business cards. Depending on the scheme, they even could be recalled from memory and exchanged over a voice channel.

This paper presents our study testing the usability of various textual key-fingerprint representation schemes. Our study consists of two parts: (1) an experiment where we measured how fast and accurate participants perform for different schemes, and (2) a survey about their perception and sentiment. These also contained a direct comparison between the representations.

Our findings suggest that the most adopted alphanumeric approaches such as the Hexadecimal and Base32 scheme perform worse than other alternatives: under a realistic threat model, more than 10% of the users failed to detect attacks targeting Hexadecimal representations, whereas our best system had failure rates of less than 3%. While the best system for accuracy is not the fastest, it is the system which received the highest usability rating and is preferred by users.

In the following sections, we discuss related work followed by an analysis of current implementations deploying in-persona key-fingerprint representation techniques and discuss our evaluated representation schemes. Then, we describe our experiment evaluating text-based key-fingerprint verification techniques with regards to their attack-detection accuracy and speed. Our experiment was conducted as an online study with 1047 participants recruited via the Amazon Mechanical Turk (MTurk) platform. We consider the scenario outlined above, where a user compares two key-fingerprint strings encoded by the different representation schemes. In addition to the implicit measurements of accuracy and speed, we also evaluate the self-reported user perception to get feedback about which systems are preferred by end users. Finally, we present our results, discuss their implications and takeaways, and conclude our work.

```
alice@localhost:~$ ssh alice@example.com
The authenticity of host 'example.com (93.184.216.34)'
  can't be established.
RSA key fingerprint is
  6f:85:66:da:e3:7a:02:c6:5e:62:3f:36:b7:d9:b4:2c.
Are you sure you want to continue connecting (yes/no)?
```

(a) OpenSSH: Lowercase Hexadecimal with Colons

```
alice@localhost:~$ gpg --fingerprint Bob
pub   2048R/00012282 2015-01-01 [expires: 2020-01-01]
      Key fingerprint =
      73EE 2314 F65F A92E C239  0D3A 718C 0701 0001 2282
uid                  Bob <bob@example.com>
```

(b) GnuPG: Uppercase Hexadecimal with Spaces

Figure 1: Alphanumeric Fingerprints Used in Practice

2 Related Work

Various key-fingerprint representations have been proposed in academia and industry. Various cryptographic protocol implementations still rely on manual fingerprint comparisons, while the hexadecimal representation is used in most of them. However, previous work suggests that fingerprint verifications are seldom done in practice [17, 37].

2.1 Key-Fingerprint Representations

Previous work has shown that users struggle with comparing long and seemingly "meaningless" fingerprints and it is suspected that they even might perform poorly in this task [19]. While most previous work has focused on the family of visual fingerprints [35, 32, 19, 10], to our knowledge, none of those focused on the differences between various different textual fingerprint representations.

Hsiao et al. have conducted a study with some textual and visual representation methods for hash verification [19]. They compared Base32 and simple word list representations with various algorithms for visual fingerprints and hash representation with Asian character sets (a subset of Chinese, Japanese Hiragana, and Korean Hangul, respectively). A within-subjects online study with 436 participants revealed that visual fingerprints score very well in both accuracy and speed, together with the Base32 text representation. Hsiao et al. conclude that depending on the available computation power and display size, either Base32 or one of the visual fingerprinting schemes should be used. They explicitly did not evaluate hexadecimal representation or digits

"because that scheme is similar to Base32 and known to be error-prone" [19]. However, our work shows that numeric representations actually perform significantly better than Base32 and is less error prone. In addition, our results suggest that language-based schemes, e. g., generated sentences achieve excellent results comparable to visual schemes. At the same time, textual approaches are more flexible (can be read out loud) and do not exclude people with sensory impairments.

Another study by Olembo et al. also focused mainly on the topic of visual fingerprints [32]. They developed a new family of visual fingerprints and compared them against a Base32 representation. The Base32 strings were twelve characters long and displayed without chunking. The participants performed better with the visual fingerprints than with Base32, regarding both accuracy and speed. Olembo et al. conclude that the Base32 representation is far away from optimal when it comes to manual key-fingerprint verification. We test this claim by comparing Base32 representation with other textual key-fingerprint representation and eventually prove it wrong.

Regarding chunking, Miller et al. have published *The magical number seven* and succeeding work that shows that most people can recall 7 ± 2 items from their memory span [29]. It has been shown that although there are slight differences between numbers, letters and words (numbers perform slightly better than letters, and letters slightly better than words), they perform similar in studies. More recent studies have shown that human working memory easily remembers up to 6 digits, 5.6 letters and 5.2 words [1, 6, 8]. Adjusting chunk sizes to these numbers can help users when comparing hashes.

While all of the above studies offer interesting insights into different (mainly visual) fingerprint representations, to the best of our knowledge there is not work focusing on which textual representation performs the best. However, this knowledge would be extremely important to help in the current debate in the secure messaging community. The representations currently being put forward and implemented are far from optimal and the results of our study can help improve the accuracy and usability of fingerprint representations. Unlike the above studies we conduct our study with a more realistic attacker strenth, as presented in subsection 4.1).

2.2 Passwords and Passphrases

A passphrase is basically a password consisting of a series of words rather than characters. In academic literature, passphrases are often considered as a potentially more memorable and more secure alternative to passwords and are often recommended by system administrators [23, 40]. In contrast to most passphrase-based systems, key-fingerprints cannot be chosen by the end-user and thus are more related to the system-assigned passphrases field: Bonneau et al. have shown that users are able to memorize 56-bit passwords [4]. miniLock[1] and its commercial successor Peerio[2] use system-assigned passphrases to generate cryptographic key pairs easing key backup and synchronization among multiple devices.

Scheme	Example
Hexadecimal	`18e2 55fd b51b c808` `601b ee5c 2d69`
Base32	`ddrf l7nv dpea` `qya3 5zoc 22i`
Numeric	`2016 507 6420 1070 394` `1136 2973 991 70`
PGP	locale voyager waffle disable Belfast performance slingshot Ohio spearhead coherence hamlet liberty reform hamburger
Peerio	bates talking duke rummy slurps iced farce pound day
Sentences	Your line works for this kind power cruelly. That lazy snow agrees upon our tall offer.

Table 1: Examples for different textual key-fingerprint representations for the same hash value

Contrary to widespread expectations, Shay et al. were not able to find any significant recall differences between system-assigned passphrases and system-assigned passwords [40]. However, they reported reduced usability due to longer submission times due to typing.

Similar to passphrases, the usage of language-based key-fingerprint representations is claimed to provide better memorability than just an arbitrary series of character strings despite the lack of empirical evidence. In our study, we measure the performance of the different approaches and also collect perception and feedback from end users.

3 Background

In the past years, various textual key-fingerprint representations have been proposed. In this section, we analyze currently practised in-persona key verification techniques in well-known applications. For comparison, Table 1 lists the approaches we used in our evaluation generated from the same hash value.

Only applications requiring manual key-fingerprint

[1] `https://minilock.io`
[2] `https://peerio.com`

verification are considered. In mechanisms like S/MIME or X.509, fingerprints play only a secondary role because certificates are verified via certificate chains.

In the following, $SHA\text{-}1(x)^{16}$ defines the execution of 16 rounds of nested *SHA-1* on x, a truncation to the leftmost 16 bits is defined by $x[0,\ldots,16]$, and pk is used as an abbreviation for the values of a public key (differs for RSA, DSA, or ECC).

3.1 Numeric

Numeric representation describes the notation of data using only numeric digits (0-9). The primary advantage of a such system is that Arabic numerals are universally understood, and in addition, numeric key-fingerprints show a similarity to phone numbers. The encoding is achieved by splitting a binary hash into chunks of equal length and expressing each chunk as a decimal number, e. g., by simply switching the representation base from 2 to 10.

The messaging and data exchange application SafeSlinger[3] implements this as a fallback scheme for unsupported languages [14]. A 24 bit SAS in SafeSlinger (cf. Figure 2a) can be expressed by three decimal encoded 8-bit numbers.

In the messaging platform WhatsApp, a fingerprint is calculated by $SHA\text{-}256(pk)^{5200}[0,\ldots,240]$. This fingerprint is split up into six chunks, where each chunk is represented by a five digits long number modulo 100,000 [43]. Concatenating this fingerprint with the fingerprint of the communication partner results in the displayed representation, e. g.,

```
77658 87428 72099 51303
34908 23247 95615 27317
09725 59699 62543 54320
```

3.2 Alphanumeric

Alphanumeric approaches use numbers and letters to represent data. Depending on the representation type and its parameters, the letters can be presented either in lower-case or in upper-case. The string can be chunked into groups of characters, which are usually of equal length. Chunking does not alter the information contained, while changing lower-case letters to upper-case letters (and vice versa) may does, depending on the coding scheme. Commonly used representations are Hexadecimal, Base32, and Base64.

3.2.1 Hexadecimal

Hexadecimal digits use the letters A-F in addition to numerical digits and are a common representation for key-fingerprints and primarily used in SSH and OpenPGP.

[3] `https://www.cylab.cmu.edu/safeslinger`

Note that the case of the letters do not make any difference. Regarding chunking, both spaces (cf. Figure 1b) and colons (cf. Figure 1a) are commonly used as separation characters.

Key fingerprints in OpenPGP version 4 are defined in RFC 4880 [7] by

$$Hex(SHA\text{-}1(0x99 \,\|\, len \,\|\, 4 \,\|\, creation\_time \,\|\, algo \,\|\, pk))$$

where *len* is the length of the packet, *creation_time* is the time the key has been created and *algo* is unique identifier for the public-key algorithm. While the inclusion of *creation_time* makes sure that even two keys with the same key material have different fingerprints, it allows an attacker to iterate through possible past times to generate similar fingerprints skipping the key generation step [5]. The actual representation of OpenPGP fingerprints is not defined in RFC 4880, but most implementations chose to encode them in hexadecimal form, e. g., GnuPG displays them uppercase in 16 bit blocks separated by whitespaces with an additional whitespace after 5 blocks (cf. Figure 1b), e. g.,

```
73EE 2314 F65F A92E C239  0D3A 718C 0701 0001 2282
```

Other implementations, such as OpenKeychain, deviate only slightly, for example by displaying them lowercase or with colored letters to ease comparison but still provide compatibility with GnuPG.

SSH fingerprint strings, as defined in RFC 4716 and RFC 4253 [15, 44], are calculated by

$$Hex(MD5(Base64(algo \,\|\, pk)))$$

where *algo* is a string indicating the algorithm, for example “ssh-rsa”. Fingerprints are displayed as “hexadecimal with lowercase letters and separated by colons” [15] (cf. Figure 1a), e. g.,

```
6f:85:66:da:e3:7a:02:c6:5e:62:3f:36:b7:d9:b4:2c
```

3.2.2 Base32

Base32 uses the Latin alphabet (A-Z) without the letters O and I (due to the confusion with numbers 1 and 0). There is no difference between lower-case letters and upper-case letters. In addition, a special padding character “=” is used, since the conversion algorithm processes blocks of 40 bit (5 Byte) in size. The source string is padded with zeroes to achieve a compatible length and sections containing only zeroes are represented by “=” [20, 21].

The ZRTP key exchange scheme for real-time applications is based on a Diffie-Hellman key exchange extended by a preceding hash commitment that allows for very short fingerprints, called Short Authentication

Strings (SAS) without compromising security [45]. The Base32 encoding used in ZRTP uses a special alphabet to produce strings that are easier to read out loud. VoIP applications such as CSipSimple[4] use this Base32 option, usually named "B32" inside the protocol. Here, the leftmost 20 bits of the 32 bit SAS value are encoded as Base32. , e. g.,

```
5 e m g
```

3.2.3 Base64

There exist a number of specifications for encoding data into the Base64 format, which uses the Latin alphabet in both lower-case and upper-case (a-z, A-Z) as well as the digits 0-9 and the characters "+", "/", and "=" to represent text data. Again, the character "=" is used to encode padded input [20]. Starting with OpenSSH 6.8 a new fingerprint format has been introduced that uses SHA-256 instead of MD5 and Base64 instead of hexadecimal representation. In addition the utilized hash algorithm is prepended, e. g.,

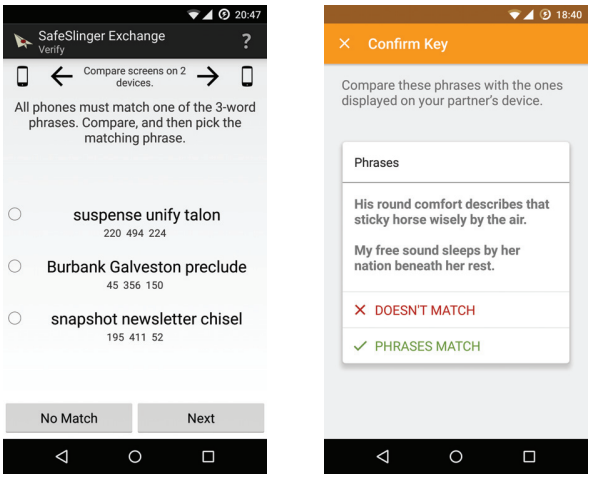
```
SHA256:mVPwvezndPv/ARoIadVY98vAC0g+P/5633yTC4d/wXE
```

3.3 Unrelated Words

Instead of (alpha)numeric representation, fingerprints can be mapped to lists of words. Here, the binary representation is split into chunks, where each possible value of a chunk is assigned to a word in a dictionary. To increase readability, such a dictionary usually contains no pronouns, articles, prepositions and such. Word lists, such as the PGP Word List [22] and the Basic English word list compiled by K.C. Ogden [31], are primarily used for verification mechanisms based on SAS. Key-Fingerprints represented by words have been implemented for VoIP applications based on the ZRTP key exchange and other real-time communication protocols. Examples are Signal[5], and the messaging and contact sharing application SafeSlinger [14] (cf. Figure 2). Besides their use in SAS based mechanisms, miniLock and Peerio utilize unrelated words for passphrase generation.

An example for a modern VoIP implementation that utilizes ZRTP for key exchange over Secure Real-Time Transport Protocol (SRTP) is Signal's private calling feature, previously distributed as Redphone. The developers chose to implement only a specific subset of the ZRTP specification [45], namely Diffie-Hellmann key exchange via P-256 elliptic curves using "B256" SASs, i. e., Base256 encoding that maps to the leftmost 16 bits of the 32 bit SAS values to the previously introduced PGP Word List [22], e. g.,

[4] https://github.com/r3gis3r/CSipSimple
[5] https://github.com/WhisperSystems/Signal-Android



(a) SafeSlinger: List of words (b) OpenKeychain: Sentences

Figure 2: Language-based fingerprint representations

```
quota holiness
```

The messaging application SafeSlinger is based on a Group Diffie-Hellman protocol [14] implementing a key verification with SASs for up to 10 participants. In SafeSlinger the leftmost 24 bits of a SHA-1 hash is used to select 3 words from the PGP Word List, e. g.,

```
suspense unify talon.
```

Besides this, two other 3 word triples are selected to force users to make a selection before proceeding (cf. Figure 2a).

In contrast to Signal and SafeSlinger, Peerio (based on miniLock) does not use any SAS based verification mechanism. It uses pictures for verification and word lists for code generation. The word list is generated from most occurring words in movie subtitles. Besides key verification, these are also used to generate so called passphrases, which are used to derive their ECC private keys.

3.4 Generated Sentences

The words from the previous dictionaries can also be used to generate syntactically correct sentences as proposed by previous research: Goodrich et al. proposed to use a "syntactically-correct English-like sentence" representation for exchanging hash-derived fingerprints over audio by using *text-to-speech* (TTS) [16]. Michael Rogers et al. implemented a simple deterministic sentence generator [16, 38][6] Though the sentences from both approaches rarely make sense in a semantic fashion, they are syntactically correct and are claimed to pro-

[6] https://github.com/akwizgran/basic-english

vide good memorability. In our study, we used Michael Roger's approach for our sentence generator.

We implemented this method for PGP fingerprints in OpenKeychain 3.6[7] (cf. Figure 2b). To the best of the authors' knowledge, to this date, it is the first integration of key verification via sentences although other projects are considering to change their fingerprint encoding scheme [38, 36].

4 Methodology

In order to evaluate the effect and perception of the different textual key-fingerprint representations, we conducted an online study on Amazon's Mechanical Turk (MTurk) crowdsourcing service. Our Universities do not have an IRB, but the study conformed to the strict data protection law of Germany and informed consent was gathered from all participants. Our online study is divided into two parts: The experiment for performance evaluation followed by a survey extracting self-reported data from users. The survey ended with demographic questions.

4.1 Security Assumptions

In this section, we define the underlying security assumptions of our study, such as fingerprint method, length, and strength against an adversary. The fingerprint method and parameters are utilized consistently for all experiments in our study to offer comparability between all possible fingerprint representations. This attack model is important for the usability since an unrealistically strong or weak attacker could skew the results. Obviously, if the fingerprint strength is not kept equal between the systems this would also skew the results.

4.1.1 Fingerprint Method

To decide upon a fingerprint method for humanly verifiable fingerprints in our study, we first have to differentiate between human and machine verification to illustrate their differences. While a full fingerprint comparison can be implemented for machine verification, humans can fall for fingerprints that match only partially. Additionally, machine comparison can work with long values, whereas for human verification the length must be kept short enough to fit on business cards and to keep the time needed for comparison low.

For machine comparison, full SHA-256 hashes should be calculated binding a unique *ID* to the public key material. The probability of finding a preimage or collision attack is obviously negligible, but the fingerprints can still be computed fast in an ad-hoc manner when needed.

[7] `https://www.openkeychain.org`

It is important to note that collision resistance is not required for our scenarios. It is required for infrastructure-based trust models such as X.509, where certificates are verified by machines and trust is established by authority. In these schemes, a signature generated by a trusted authority can be requested for a certificate by proving the control over a domain, but then reused maliciously for a different certificate/domain. This is already possible with a collision attack, without targeting a full preimage. In contrast, the direct human-based trust schemes considered in this study only need to be protected against preimage attacks, because no inherently trusted authority is involved here.

While machine comparison needs to be done fast, e. g., on key import, manual fingerprint verification by humans is done asynchronously in person or via voice. Thus, we can use a key derivation function to provide a proof-of-work, effectively trading calculation time for a shorter fingerprint length. Secure messaging applications such as Signal or OpenPGP-based ones could pre-calculate the fingerprints after import and cache these before displaying them for verification later.

Thus, modern memory-hard key derivation functions such as *scrypt* [34] or *Argon2* [3] can be utilized to shorten the fingerprint length. These key derivation functions are parametrized to allow for different work factors. Suitable parameters need to be chosen by implementations based on their targeted devices and protocol.

As discussed in Section 3.2.1, while the generation of new fingerprints consists of the creation of a new key pair and the key derivation step, an attacker can potentially skip the key creation. Thus, in the following we only consider the key derivation performance as the limiting factor for brute force attacks.

When utilizing a properly parametrized key derivation function for bit stretching, the security of a 112 bit long fingerprint can be increased to require a brute force attack comparable to a classical 2^{128} brute force attacker. Consequently, a fingerprint length of 112 bit is assumed throughout our study.

4.1.2 Attacker Strength for Partial Preimages

In our user study, we assume an average attacker trying to impersonate an existing *ID* using our fingerprint method. Thus, an attacker would need to find a 112 bit preimage for this existing fingerprint using a brute force search executing the deployed key derivation function in each step. Due to the work factor, we consider this to be infeasible and instead concentrate on partial preimages. For comparability and to narrow the scope of our study, an attacker is assumed that can control up to 80 bits of the full 112 bit fingerprint.

Attackers might aim to find partial preimages where

the uncontrolled bits occur at positions that are more easily missed by inattentive users. First, the bits at the beginning and the end should be fixed as users often begin their comparison with these bits. Thus, we assume that, for any representation method, the first 24 and last 24 bits are controlled by the attacker and thus the same as in the existing fingerprint. Based on the feedback from our pre-study participants and reports from related work, this can be considered best-practice [17, 37]. Second, of the remaining 64 bits in the middle of our 112 bit fingerprint, we assume that 32 bits are controlled by the attacker in addition to the first 24 and last 24 bits. In total, we assume that 80 bits are controlled by the attacker, i. e., are the same as in the existing fingerprint, and 32 bit are uncontrolled.

The probability of finding such a partial preimage for a fingerprint when executing 2^{49} brute force steps is calculated approximately by

$$1-\left(\frac{2^{112}-\sum_{k=1}^{32}\binom{64}{k}}{2^{112}}\right)^{2^{49}} \approx 0.66.$$

The inner parentheses of this equation define the probability that no partial preimage exists for one specific bit permutation. Instead of using $\binom{64}{32}$, a sum over 32 variations has been inserted to include permutations with more than the uncontrolled 32 bit that are also valid partial preimages. Finally, the probability to find a partial preimage is defined by the inverse of the exponentiation. Assuming the scrypt key derivation function parametrized with $(N,r,p) = (2^{20},8,1)$, Percival calculates the computational costs of a brute force attack against 2^{38} ($\approx 26^8$) hashed passwords with \$610k and 2^{53} ($\approx 95^8$) with \$16B [34]. These costs can be considered a lower and upper bound for our attacker, which we assume to have average capabilities and resources. While 2^{38} has a probability of finding a partial preimage of only 0.05%, with 2^{42} the probability reaches nearly 1%, and with 2^{49}, as in our example, a partial preimage is found with over 50%.

In our study, we simulate attacks by inverting the bits from the existing fingerprint which are uncontrolled by the attacker, while the controlled bits are unchanged. For our theoretical approximation, we assume that the first 24 and last 24 bits should be controlled as well as 32 bits from the middle. In our study, we simulate an even more careful selection of appropriate fingerprints from the ones that an attacker would brute force. A general criteria here is to minimize the influence of uncontrolled bits on the entire fingerprint: For numeric and alphanumeric representations all bits affecting a character or digit are inverted together. For unrelated words, all bits affecting a word are changed. Sentences are never changed in a way that would alter the sentence structure.

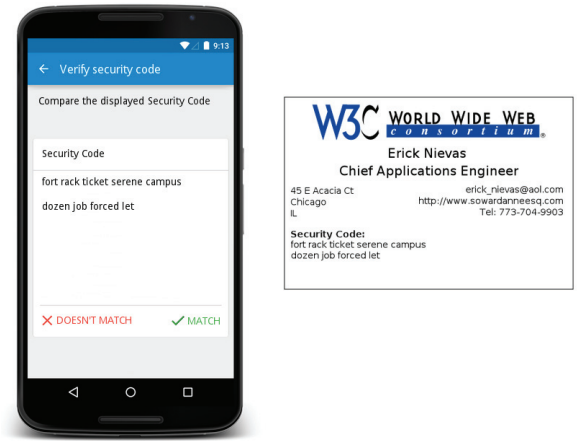


Figure 3: A screenshot of the actual task a user had to perform in the experiment. A user rates whether the *security codes* match, in this case with the Peerio word list approach, by clicking on the corresponding buttons shown on the phone.

4.2 Pre-Study

To get additional feedback from participants and evaluate our study design for flaws and misunderstandings, we conducted two small pre-studies: A lab study with 15 participants and an MTurk experiment with 200 participants, all required to perform 10 comparisons for each representation scheme (totally 60 comparisons in a randomized order). In our lab-study, we mainly focused on qualitative feedback, whereas the main goal of the MTurk pre-study was to find flaws in the presentation and task descriptions, as well as to check whether our proposed methodology is received as expected.

The biggest problem we found regarding the study design was that participants were uncertain if they should check for spelling mistakes in the words and sentence-based representation or if the all attacks would change entire words. To clarify this, a speech bubble was included in the task description that the participants do not have to look for spelling mistakes for language-based approaches.

We tested different rates of attack during the pre-study. The results showed that participants who were exposed to frequently occurring attacks were more aware and had a much higher attack detection rate. For our main study, we reduced the number of attacks to 40 comparisons with 4 attacks to have a good balance between true positives and false negatives. We received feedback that attacks on anchor parts of the strings, i. e., in the beginning, end, and at line breaks could be easily detected. Many users had problems with distinguishing the hexadecimal from the Base32 representation as well as distinguishing different word list approaches (Peerio vs. OpenPGP word list). Thus, we opted for a mixed factorial study

design where users test *only one* scheme of each type. We grouped the hexadecimal and Base32 scheme for the alphanumeric type and the PGP and Peerio for the word-list type together. These two groups were tested between-subjects in a split-plot design, i. e., the participants test either hexadecimal or Base32 for the alphanumeric type. See Table 2 for a graphical representation of our condition assignment design.

Type (Within-Group)	Scheme (Between-Group)
Alphanumeric	Hexadecimal `XOR` Base32
Numeric	Numeric
Unrelated Words	PGP `XOR` Peerio
Generated Sentences	Generated Sentences

Table 2: To avoid confusion between too similar approaches (cf. Section 4.2), in our condition assignment, scheme types (left column) can consist of multiple representation schemes (right column). Each participant tests *only one* randomly assigned scheme of each type in a randomized order. .

4.3 Experiment Design

The main part of our online study is the experiment part where users perform actual fingerprint comparisons. Here, we conducted two separate experiments with a distinct set of participants: (1) our main experiment testing different textual high-level representation schemes against each other and (2) a secondary experiment testing different chunk sizes for the hexadecimal representation. We opted for two distinct experiments due to the exponential growth of experiment conditions, as described in Section 4.3.1.

Before letting the participants start our experiment, we explained the scenario:

> "With this HIT, we are conducting an academic usability study followed by a short survey about different types of security codes used in the IT world. Security codes are often used in encrypted communications to identify the participants in a communication. If the security codes match, you are communicating securely. If they don't match, an eavesdropper may be intercepting your communication".

On MTurk, the term *Human Intelligence Task*, or *HIT* stands for a self-contained task that a worker can work on, submit answers, and get a reward for completing. Since our participants might not be familiar with the *key-fingerprint representation* term, we replaced it with *security codes* for the sake of the study.

We opted not to obfuscate the goal of the study since our research aims at finding the best possible representation for the comparison of key-fingerprints in a security context. This is closest to how users interact with fingerprints in the real world — their secure messaging applications also ask them to compare the strings for security purposes. The question how to motivate users to compare fingerprints is an entirely different research question. So in our case, we believe it was not necessary or desirable to use deception and since deception should be used as sparingly as possible we opted for the "honest" approach.

After agreeing the terms, participants are shown a fictitious business card next to a mobile phone, both displaying a security code (as shown in Figure 3). To become more familiar with the task, the experiment is started with 4 training tasks (each method once) not considered in the evaluation. The user's only task is to rate whether the shown fingerprints match by clicking on *Match* or *Doesn't Match* on the phone. Based on the condition assignment, participants see different approaches in a *randomized order*. We measure whether their answer was correct and their speed, i. e., the amount of time spent on the comparison. The experiment is concluded with a survey collecting feedback on the used approaches and the tasks and demographic information discussed in the "Results" section.

4.3.1 Variables and Conditions

In the main experiment, the used *representation scheme* is our controlled independent variable whereas its values define our experiment conditions. In our additional chunking experiment, the *chunking size* is our controlled independent variable instead of the representation algorithm. During all tasks, we measure how fast participants perform with their given conditions and whether they are able to detect attacks by rating "incorrect" (*speed* and *accuracy* as our measured dependent variables).

In both experiments, each user had to perform 46 comparisons in total. To detect users clicking randomly, 2 obviously distinct comparisons were added to test a participant's attention. Training comparisons and attention tests are not included in the evaluation. Based on the feedback in our pre-study, we added tooltips during the training comparisons giving hints for language-based approaches telling the user that spelling attacks would not occur. We set the number of attacks to six: two obvious attacks where all bits are altered serving as control questions and 4 actual attacks with partial 80-bit preimages (one for each representation scheme). Participants failing at the control attacks are not considered in the evaluation but still received a payment if finishing all tasks. The major challenge in the study design is a high attack detection rate in general: most users perform comparisons correctly for the given attacker strength.

To avoid side effects, we chose fixed font size, color

Figure 4: A screenshot showing a statement rating in the post-experiment survey. Since the participants might not distinguish the different types, we have provided an example from their previous task.

and style, i. e., the same typeface for all fingerprint representations. In addition, we set fixed line breaks for sentences and word lists. In the main experiment, the same chunking style was used for all representations: For (alpha)numeric approaches a chunk consists of four characters separated by spaces. For word lists, we opted for a line break every four words. In the generated sentences representation, one sentence per line is displayed. We are aware that all these design decisions can have an effect on the comparison of the representations. However, our pre-study results show a significantly lower effect size. More importantly, we are mainly interested in comparing the concepts, therefore we did not vary any of the visual attributes like font size or style. In particular, differences resulting from the font's typeface have not been evaluated. Lund showed in his meta-analysis that there are no significant legibility differences between serif and sans serif typefaces [25].

Chunk-Size Testing A question was raised whether the chunking of a hexadecimal string plays a greater role in comparison to the different approaches. Thus, in addition to the main experiment testing different representation types, we conducted a second experiment with new participants testing different chunk sizes for the hexadecimal representation. Here, we used chunk-sizes ranging from 2 to 8 in addition to "zero-chunk size" (8 cases). The zero-chunk size means that no spaces have been included. To make the results more comparable, we opted for a similar design as done in the major experiment, i. e., we required the same amount of comparisons, used the same font settings, and had the same amount of attacks. For each participant, we assigned 4 out of 8 different chunk-sized randomly. Same as in the major experiment, all participants had to compare 46 fingerprints whereas the first 4 are considered as training comparisons, 4 attacks (one for each chunk size), and 2 control attacks with obviously distinct fingerprints.

The major experiment is followed by a survey focusing on self-reported user perception and opinions about the different approaches. This is the main reason we opted to compare as much as possible in a within-groups fashion and only selected a small number of conditions in total. Since users might not notice the difference between the various dictionary or alphabet approaches, we designed a mixed factorial design where the users would only get one of the alphabets/dictionaries (between-subjects) but they would test all different high-level systems (within-group) as depicted in Table 2. The between-group conditions have been assigned randomly with a uniform distribution. Since participants from our pre-study had difficulties to distinguish the different chunking approaches, we skipped the survey part in the chunk-size experiment.

4.3.2 Online Survey

The experiment was followed by an online survey gathering self-reported data and demographics from participants. To measure perception, we asked the participants whether they agreed with statements discussed in subsection 5.2 on a 5 point Likert scale: from strongly disagree to neural strongly agree as shown in Figure 4. Participants had to rate each representation type for all statements. Since users might not distinguish the different representation schemes, we provide an example from their previously finished task.

4.3.3 Statistical Testing

We opted for the common significance level of $\alpha = 0.05$. To counteract the multiple comparisons problem, we use the Holm-Bonferronicorrection for our statistical significance tests [18]. Consequently, all our p-values are reported in the corrected version.

We test the comparison duration with the Mann-Whitney-Wilcoxon (MWW) test (two-tailed). We opt for this significance test due to a few outliers, consequently a

Scheme	Speed				Accuracy				Total	
	mean [s]	med [s]	stdev	p-val	fail-rate	p-val	f-pos	fails	attacks	tests
Hexadecimal	11.2	10.0	6.4		10.44		0.49	50	479	4765
Hexadecimal – Base32	1.0	1.1	0.0	<0.001	−1.94	0.690	−2.09	12	32	269
Hexadecimal – Numeric	0.6	0.5	0.6	<0.001	−4.10	0.048	0.21	−9	−452	−4527
Hexadecimal – PGP	−1.8	−1.2	−1.0	<0.001	−1.65	0.690	−0.01	11	35	340
Hexadecimal – Peerio	2.5	2.7	0.8	<0.001	−4.69	0.048	0.08	22	−8	−91
Hexadecimal – Sentences	−1.1	−0.7	−0.6	<0.001	−7.45	<0.001	−0.99	22	−457	−4518
Base32	10.2	8.9	6.4		8.50		2.58	38	447	4496
Base32 – Hexadecimal	−1.0	−1.1	−0.0	<0.001	1.94	0.690	2.09	−12	−32	−269
Base32 – Numeric	−0.4	−0.6	0.6	<0.001	−2.16	0.404	2.30	−21	−484	−4796
Base32 – PGP	−2.8	−2.3	−1.0	<0.001	0.28	0.714	2.08	−1	3	71
Base32 – Peerio	1.5	1.6	0.8	<0.001	−2.75	0.404	2.17	10	−40	−360
Base32 – Sentences	−2.1	−1.8	−0.6	<0.001	−5.51	<0.001	1.10	10	−489	−4787
Numeric	10.6	9.5	5.8		6.34		0.28	59	931	9292
Numeric – Hexadecimal	−0.6	−0.5	−0.6	<0.001	4.10	0.048	−0.21	9	452	4527
Numeric – Base32	0.4	0.6	−0.6	<0.001	2.16	0.404	−2.30	21	484	4796
Numeric – PGP	−2.4	−1.7	−1.6	<0.001	2.45	0.404	−0.22	20	487	4867
Numeric – Peerio	1.9	2.2	0.2	<0.001	−0.59	0.714	−0.13	31	444	4436
Numeric – Sentences	−1.7	−1.2	−1.2	<0.001	−3.35	0.004	−1.20	31	−5	9
PGP	13.0	11.2	7.4		8.78		0.50	39	444	4425
PGP – Hexadecimal	1.8	1.2	1.0	<0.001	1.65	0.690	0.01	−11	−35	−340
PGP – Base32	2.8	2.3	1.0	<0.001	−0.28	0.714	−2.08	1	−3	−71
PGP – Numeric	2.4	1.7	1.6	<0.001	−2.45	0.404	0.22	−20	−487	−4867
PGP – Peerio	4.3	3.9	1.8	<0.001	−3.03	0.337	0.09	11	−43	−431
PGP – Sentences	0.7	0.5	0.4	<0.001	−5.79	<0.001	−0.98	11	−492	−4858
Peerio	8.7	7.3	5.6		5.75		0.41	28	487	4856
Peerio – Hexadecimal	−2.5	−2.7	−0.8	<0.001	4.69	0.048	−0.08	−22	8	91
Peerio – Base32	−1.5	−1.6	−0.8	<0.001	2.75	0.404	−2.17	−10	40	360
Peerio – Numeric	−1.9	−2.2	−0.2	<0.001	0.59	0.714	0.13	−31	−444	−4436
Peerio – PGP	−4.3	−3.9	−1.8	<0.001	3.03	0.337	−0.09	−11	43	431
Peerio – Sentences	−3.6	−3.4	−1.4	<0.001	−2.76	0.075	−1.07	0	−449	−4427
Sentences	12.3	10.7	7.0		2.99		1.48	28	936	9283
Sentences – Hexadecimal	1.1	0.7	0.6	<0.001	7.45	<0.001	0.99	−22	457	4518
Sentences – Base32	2.1	1.8	0.6	<0.001	5.51	<0.001	−1.10	−10	489	4787
Sentences – Numeric	1.7	1.2	1.2	<0.001	3.35	0.004	1.20	−31	5	−9
Sentences – PGP	−0.7	−0.5	−0.4	<0.001	5.79	<0.001	0.98	−11	492	4858
Sentences – Peerio	3.6	3.4	1.4	<0.001	2.76	0.075	1.07	0	449	4427

Table 3: Our experiment results showing the differences between the representation schemes. The top rows of each row group separated by a rule, show the raw performance of a baseline scheme, followed by italic rows showing a direct comparison delta. Greyed-out values are not backed by statistical significance. The columns *fail-rate* (undetected attacks) and *false-pos* (same string rated as an attack) display percentage values.

slightly skewed normal distribution, and a large amount of collected data. The *common language effect size* is shown by mean and median comparisons [26].

The attack detection rate is tested with a pairwise Holm-Bonferroni-corrected Barnard's exact test (`Exakt` package in `R`) achieving one of highest statistical power for 2x2 contingency tables [2].

Survey ratings are, again, tested by using the MWW significance test (two-tailed test). As has been shown in previous research [9], it is most suitable for 5-point Likert scales, especially if not multimodal distributed as in our survey results. In case two fingerprint representation schemes are statistically tested against each other, only participants encountering both schemes were considered.

5 Results

In this section, we present our results: our online study with 1047 participants has been conducted in August and September 2015. The study for testing the chunk size has been conducted in February 2016 with 400 participants. Starting with our online experiment evaluation showing the raw performance of users, we then present user perception results from the follow-up survey. Finally, we discuss the demographics of our participants.

5.1 Online Experiment

Participants who have not finished all comparisons or failed the attention tests were excluded from our eval-

Scheme	Speed			Accuracy				Total	
	mean [s]	med [s]	p-val	fail-rate	p-val	false-pos	fails	*attacks*	*tests*
Hexadecimal (4)	12.3	10.4		6.78		0.38	16	236	2360
hex (4) – hex (0)	−2.4	−2.6	<0.001	0.33	1.000	−0.28	−2	−17	−170
hex (4) – hex (2)	−0.3	−0.9	<0.001	1.37	1.000	0.00	−3	3	30
hex (4) – hex (3)	−0.3	0.1	0.362	−0.64	1.000	0.09	2	8	80
hex (4) – hex (5)	−1.4	−1.2	<0.001	1.01	1.000	−0.40	−2	5	50
hex (4) – hex (6)	−1.9	−1.8	<0.001	2.43	1.000	0.09	−5	8	80
hex (4) – hex (7)	−1.7	−1.8	<0.001	3.35	1.000	0.19	−8	−1	−10
hex (4) – hex (8)	−2.8	−3.2	<0.001	1.35	1.000	−0.12	−4	−10	−100

Table 4: Comparison of the chunking experiment results showing the differences between the representation schemes. The top row shows the raw performance of the hexadecimal scheme with a four-character chunking, followed by italic rows showing a direct comparison delta. Greyed-out values are not backed by statistical significance. The columns *fail-rate* (undetected attacks) and *false-pos* (same string rated as an attack) display percentage values.

uation: all participant compared 46 security codes in a randomized order, whereas 40 (10 of each scheme) were considered in the evaluation. The four training samples and the control questions are excluded. Few comparisons done in less than 2 seconds and more than one minute have been excluded. The reason for such can either be multiple clicks during the page load, or external interruptions of the participants. None of the attack could be successfully detected in under 4 seconds.

Our experiment results, summarized in Table 3, show the raw performance of all schemes regarding their speed, accuracy and false-positive rate. The top rows of each row group, separated by a rule, show the raw performance of a representation scheme as baseline (negative values indicate lower values than the baseline). The following rows show a direct comparison delta between between two schemes. The speed column group consists of the mean and median (in seconds), the standard deviation and the according p-values for a direct comparison. The fail-rate column shows the rate of the undetected attacks with the according p-values for a direct comparison. The total column group simply shows the total numbers of tests, attacks and undetected attacks.

The results show that the average time spent on comparisons plays only a minor role among the schemes: 4.3 s difference between the best and the worst scheme. Note that the Peerio word-list scheme performed best with 8.7 s mean whereas the PGP word list performed worst with 13 s mean ($p < 0.001$).

However, there is a clear effect regarding the attack detection rate (see Table 3). All alternative key-fingerprint representations performed better than the state-of-the-art hexadecimal representation, where 10.1% of attacks have not been detected by the users. Previous work shows similar numbers for Base32 [19]. To our surprise, the numeric approach performs better in both categories: it features an attack detection rate of 93.57% ($p < 0.01$) and an average speed of 10.6*s* ($p < 0.001$). Generated sentences achieved the highest attack detection rate of 97.97% with a similar average speed as the hexadecimal scheme. On the downside, this scheme has produced a slightly higher false-positive rate. We found that the false positives occurred mostly with longer sentences where there has been a line break on the phone mock-up due to portrait orientation. This is a realistic problem of this system if used with portrait orientation and not a problem with our mock-up in itself. Improvements on making the sentences shorter could mitigate this situation.

Chunk-Size Experiment

Table 4 summarizes the results of our secondary chunk-size experiment. As can be seen, no statistically significant results have been achieved for the *attack detection fail-rate* (undetected attacks by end users). However, we observed that the chunk sizes with 3 and 4 characters performed best in speed, even though the effect sizes were minor: only 3.3 seconds difference with similar standard deviations between the best and worst chunk size setting.

Firstly, we notice that despite the same attack strength as in our major experiment, participants were able to detect more attacks. We suspect that the higher attack detection rate is based on (1) a higher learning effect due to the same scheme for all comparisons and (2) in contrast to our major study, participants had a slightly higher drop-out rate and thus only more motivated participants were considered. This is supported by the numbers in the total tests column of Table 4: here, we can see that for the zero-chunking and chunking with 8 characters less tests have been performed. This is based on the fact that although the chunk sizes have been assigned almost uniformly, participants assigned with harder chunk settings often dropped out before even finishing their entire task.

More importantly, our results also support the claim from our pre-study: The chunking parameter in hexadecimal strings plays only a minor role in the *attack detection fail-rate*.

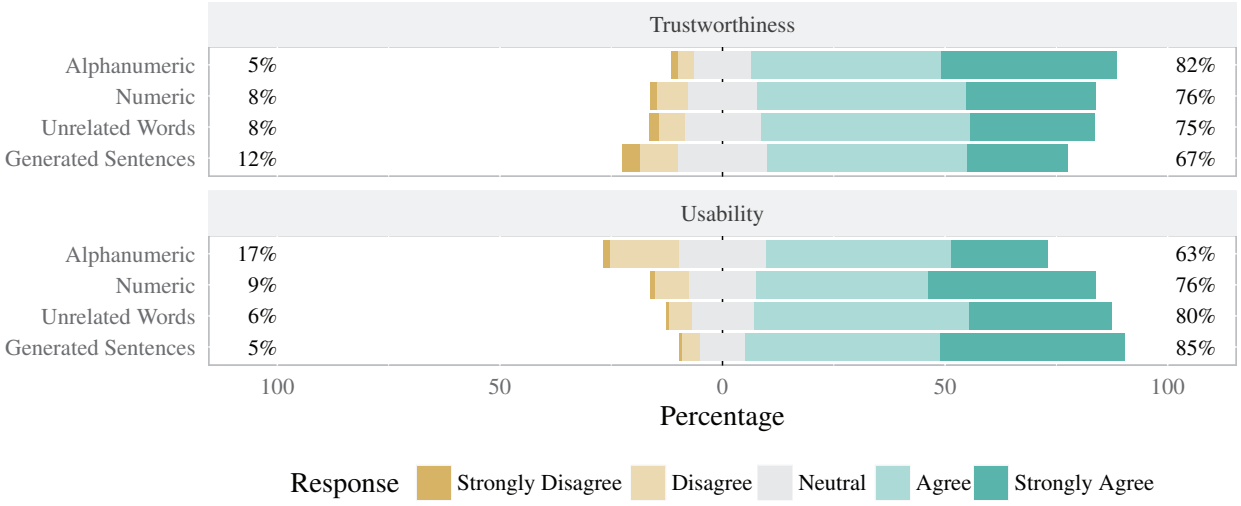


Figure 5: Aggregated survey results for statement rating regarding the usability and trustworthiness.

5.2 Online Survey

To measure the usability and trustworthiness of all representation schemes, we asked our participants whether they agreed with the following statements:

S_1 The comparisons were easy for me with this method

S_2 I am confident that I can make comparisons using this method without making mistakes

S_3 I think making comparisons using this method would help me keep my communications secure

S_4 I was able to do the comparisons very quickly with this method

S_5 I found this method difficult to use

S_6 Overall, I liked this method

We mixed positive and negative statements, e. g., S_1 and S_5, to create a more robust measure. S_6 is used to calculate the overall ranking of the different representation schemes.

Figure 5 shows the aggregated results where the usability statements are grouped to one usability feature and the trustworthiness derived from the rating on the statement S_3. Negative statement ratings have been inverted for a better comparison. Figure 6 shows the rating results for each specific statement in the survey. The order of the tested schemes has been chosen randomly, but was kept consistent across all statements. Same as in our online experiment evaluation, the pairwise statistical tests are Holm-Bonferroni corrected. In case of a direct statistical test between two schemes, only users encountering both schemes have been considered. All in all, the usability perception of the participants is almost consistent with the performance results from the experiment.

To measure the perception of the task difficulty, we asked the participants whether they agreed with the statements S_1, S_2 and S_3 respectively. As illustrated in Figure 6 in the Appendix A, the effect size between the different approaches is low. However, the participants were more likely to agree that language-based representation schemes are easier to use. For instance, we see that in comparison to the alphanumeric schemes (average rating of 3.4), word list (average rating of 3.9, $p < 0.001$) and generated sentence schemes (average rating of 4.2, $p < 0.001$) are rated to be easier by our participants (S_1, S_5). While the experiment results of the sentence generators clearly outperformed all other approaches, they also were rated better by the participants. Same applies for the low-performing hexadecimal and Base32 schemes which clearly received lower ratings. Consistently with the surprising performance results in the experiment, the numeric scheme is also considered to be easier by many participants: average rating of 3.9 and $p < 0.001$.

The sentence generator scheme achieved the highest user confidence rating "making comparisons without any mistakes" (S_2, $p < 0.001$ for all pairwise comparisons). The participants' perception is consistent with the experiment results where the word-list-based and sentence generator schemes lead to higher attack detection rates.

The ratings for S_4 illustrate that more complex representation schemes from the user's point of view, such as hexadecimal and Base32, are considered to be more secure by participants, even though all approaches provide the same level of security.

5.3 Demographics

A total of 1047 users participated in the online study while only 1001 have been considered in the evaluation due to our two control questions. Out of the evaluated participants, 534 participants were male, 453 were female, 4 chose other while the rest opted to not give any information. No significant difference between genders could be found, with a subtle trend of a higher accuracy for women and higher speed among men. The median age was 34 (34.4 average) years, while 34 participants chose not to answer (no statistically significant differences between ages).

A total of 39 people reported to have "medical conditions that complicated the security code comparisons (e. g., reading disorders, ADHD, visual impairments, etc.)" with a slightly higher undetected attack rate (statistically insignificant due to small sample size and thus low statistical power).

The majority of the participants stated to have a Bachelor's degree (399 of 1047) as their highest education whereas 34% chose not to answer. 931 participants have started our HIT but stopped early during the experiment (mostly after the first few comparisons). 160 users reported the general task to be annoying.

6 Discussion

The results of our study show that while there are subtle speed variations among all approaches, the attack detection rate and user perception for the current state-of-the-art hexadecimal key-fingerprint representation is significantly lower than those of most alternative representation schemes. Language-based representations (with the exception of the PGP word list) show improved user behaviour leading to a higher detection rate of attacks. To improve the usability of key-fingerprints, we propose the following takeaways based on our study results.

6.1 Takeaways

Our results show that all representation schemes achieve a high accuracy (high attack detection rate) and can be performed quickly by users. As expected, language-based fingerprint representations are more resilient against attacks (higher attack detection rate) and achieve better usability scores. Among all conditions, alphanumeric approaches performed worse and have been outperformed. For instance, the *numeric* representation was more suitable than hexadecimal and Base32. The raw performance results suggest a similar speed for the numeric representation with a higher attack detection rate, and it also has received better usability ratings from end-users.

Our chunking experiment has shown that chunk-sizes play only a minor role in improving attack detection rates (we could not find statistically significant differences). However, if a hexadecimal representation is used chunks of 3 and 4 characters perform best.

As shown by the word list representations, the comparison speed can be increased by larger dictionaries leaving room for improvement in this area. Even though all representation schemes provide the same level of security, exotic looking solutions are considered to be more secure by end users.

6.2 Limitations

Most importantly, our study design *does not test* whether end users *are actually willing to compare any fingerprints* in practice. We only aim to study how easy different representations are to compare from the users' point of view.

As with any user study conducted with MTurk, there is concern about the external validity of the results: users in the real world might show different behaviour. This is mainly because of two reasons: (1) in practice fingerprint comparisons will seldom occur in a such frequency, and (2) when performed in practice play a more important role than just participating in an anonymous online study. Additionally, MTurkers have been shown to be more tech-savvy and are better in solving textual and visual tasks in comparison to the average population. Thus, they could have performed better in most of the comparison conditions than the average population. It is also known that some MTurkers just "click through" studies to get the fee and thus distort study results. Our counterbalanced study design with included control questions and statistical significance tests mitigate this effect. For instance, we excluded 46 out of 1047 participants from our main study part based on these questions being answered incorrectly.

Due to the within-group part of our factorial design, many parameter choices such as different fonts, font sizes, attack rates, etc. could not be considered. These are, however, interesting avenues for future work. As shown in our additional chunking experiment, another challenge in testing different parameters is the high attack detection rate, where subtle changes would require a high amount of users to produce statistically significant results.

Due to the anonymous nature of online studies, it is

also impossible to reliably tell which languages a participant is fluent in. We specified that we only wanted participants from English-speaking countries, however we had no way of checking compliance except by relying on self-reported data. Language-based representation approaches might induce additional barriers for non-native speakers, e. g., due to unknown or unfamiliar words.

7 Conclusion and Future Work

We evaluated six different key-fingerprint representation types with regards to their comparison speed, attack detection accuracy and usability, which encompasses attack detection but also resilience against human errors in short-term memory. An online study with 1047 participants was conducted to compare numeric, alphanumeric (Hexadecimal and Base32), word lists (PGP and Peerio), as well as generated sentences representation schemes for key-fingerprint verification. All fingerprint representations were configured to offer the same level of security with the same attacker strength.

Our results show that usage of the large word lists (as used in Peerio) lead to the fastest comparison performance, while generated sentences achieved highest attack detection rates. In addition, we found that additional parameters such as chunking of characters plays only a minor role in the overall performance. The widely-used hexadecimal representation scheme performed worst in all tested categories which indicates that it should be replaced by more usable schemes. Unlike proposals which call for radically new fingerprint representations, we studied only textual fingerprint representations, which means that the results of our work can be directly applied to various encryption applications with minimal changes needed. Specifically, no new hardware or complex software is required: applications merely need to replace the strings they output to achieve a significant improvement in both attack-detection accuracy and usability.

There are various interesting areas of future work. Firstly, we chose to study only a selected sample from the design space of fingerprint representations in a within-subjects design, so we could facilitate a direct comparison between the different classes of fingerprints. Further work exploring line breaks, font settings, dictionaries, different attacker strengths, etc. will likely lead to further improvement possibilities.

While this work shows that there are better ways to represent key-fingerprints than currently being used, it does not explore what can be done to motivate more users to actually compare the fingerprints in the first place. Follow-up studies to research this important question are naturally an interesting and vital area of research.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments, and Trevor Perrin, Jake McGinty, Tom Ritter and Skylar Nagao for their discussion and excellent feedback.

References

[1] BADDELEY, A. Working memory. *Science 255*, 5044 (1992), 556–559.

[2] BARNARD, G. Significance tests for 2×2 tables. *Biometrika* (1947), 123–138.

[3] BIRYUKOV, A., DINU, D., AND KHOVRATOVICH, D. Argon2: the memory-hard function for password hashing and other applications. Tech. rep., Password Hashing Competition (PHC), December 2015.

[4] BONNEAU, J., AND SCHECHTER, S. Towards reliable storage of 56-bit secrets in human memory. In *Proceedings of the 23rd USENIX Security Symposium* (August 2014).

[5] BREITMOSER, V. pgp-vanity-keygen. `https://github.com/Valodim/pgp-vanity-keygen`, 2014.

[6] BUCKNER, R. L., PETERSEN, S. E., OJEMANN, J. G., MIEZIN, F. M., SQUIRE, L., AND RAICHLE, M. Functional anatomical studies of explicit and implicit memory retrieval tasks. *The Journal of Neuroscience 15*, 1 (1995), 12–29.

[7] CALLAS, J., DONNERHACKE, L., FINNEY, H., SHAW, D., AND THAYER, R. OpenPGP Message Format. RFC 4880 (Proposed Standard), Nov. 2007. Updated by RFC 5581.

[8] CRANNELL, C., AND PARRISH, J. A comparison of immediate memory span for digits, letters, and words. *The Journal of Psychology 44*, 2 (1957), 319–327.

[9] DE WINTER, J. C., AND DODOU, D. Five-point Likert items: t test versus Mann-Whitney-Wilcoxon. *Practical Assessment, Research & Evaluation 15*, 11 (2010), 1–12.

[10] DHAMIJA, R. Hash visualization in user authentication. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems* (New York, NY, USA, 2000), CHI EA '00, ACM, pp. 279–280.

[11] DIERKS, T., AND RESCORLA, E. The Transport Layer Security Protocol Version 1.2. RFC 5246, Aug. 2008. Updated by RFCs 5746, 5878, 6176.

[12] ELECTRONIC FRONTIER FOUNDATION. Secure Messaging Scorecard. `https://www.eff.org/secure-messaging-scorecard`, 2014.

[13] ELLISON, C., ET AL. Establishing identity without certification authorities. In *USENIX Security Symposium* (1996), pp. 67–76.

[14] FARB, M., LIN, Y.-H., KIM, T. H.-J., MCCUNE, J., AND PERRIG, A. Safeslinger: easy-to-use and secure public-key exchange. In *Proceedings of the 19th annual international conference on Mobile computing & networking* (2013), ACM, pp. 417–428.

[15] GALBRAITH, J., AND THAYER, R. The Secure Shell (SSH) Public Key File Format. RFC 4716 (Informational), Nov. 2006.

[16] GOODRICH, M. T., SIRIVIANOS, M., SOLIS, J., TSUDIK, G., AND UZUN, E. Loud and clear: Human-verifiable authentication based on audio. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on* (2006), IEEE, pp. 10–10.

[17] GUTMANN, P. Do users verify SSH keys? *USENIX;login: 36*, 4 (2011).

[18] HOLM, S. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics* (1979), 65–70.

[19] HSIAO, H.-C., LIN, Y.-H., STUDER, A., STUDER, C., WANG, K.-H., KIKUCHI, H., PERRIG, A., SUN, H.-M., AND YANG, B.-Y. A study of user-friendly hash comparison schemes. In *Computer Security Applications Conference, 2009. ACSAC'09. Annual* (2009), IEEE, pp. 105–114.

[20] JOSEFSSON, S. The Base16, Base32, and Base64 Data Encodings. RFC 3548 (Informational), July 2003.

[21] JOSEFSSON, S. The Base16, Base32, and Base64 Data Encodings. RFC 4648, Oct. 2006.

[22] JUOLA, P. Whole-word phonetic distances and the PGPFone alphabet. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on* (1996), vol. 1, IEEE, pp. 98–101.

[23] KEITH, M., SHAO, B., AND STEINBART, P. A behavioral analysis of passphrase design and effectiveness. *Journal of the Association for Information Systems 10*, 2 (2009), 2.

[24] LAURIE, B., LANGLEY, A., AND KASPER, E. Certificate Transparency. RFC 6962 (Experimental), June 2013.

[25] LUND, O. *Knowledge construction in typography: the case of legibility research and the legibility of sans serif typefaces.* PhD thesis, The University of Reading, Department of Typography & Graphic Communication., 1999.

[26] MCGRAW, K. O., AND WONG, S. A common language effect size statistic. *Psychological bulletin 111*, 2 (1992), 361.

[27] MELARA, M. S., BLANKSTEIN, A., BONNEAU, J., FREEDMAN, M. J., AND FELTEN, E. W. CONIKS: A Privacy-Preserving Consistent Key Service for Secure End-to-End Communication. Tech. Rep. 2014/1004, Cryptology ePrint Archive, December 2014.

[28] [MESSAGING] MAILING-LIST ARCHIVE. Usability of Public-Key Fingerprints. `https://moderncrypto.org/mail-archive/messaging/2014/000004.html`, 2014.

[29] MILLER, G. A. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review 63*, 2 (1956), 81.

[30] NAMECOIN PROJECT. Namecoin. `http://namecoin.info`, Nov. 2014.

[31] OGDEN, C. K., ET AL. System of Basic English. *Self-published* (1934).

[32] OLEMBO, M. M., KILIAN, T., STOCKHARDT, S., HÜLSING, A., AND VOLKAMER, M. Developing and testing a visual hash scheme. In *EISMC* (2013), pp. 91–100.

[33] [OPENPGP] IETF MAIL ARCHIVE. Fingerprints. `https://mailarchive.ietf.org/arch/msg/openpgp/2C9gTsxTgh29W8VX8x70OYZqfUY`, 2015.

[34] PERCIVAL, C. Stronger key derivation via sequential memory-hard functions. *Self-published* (2009).

[35] PERRIG, A., AND SONG, D. Hash visualization: A new technique to improve real-world security. In *International Workshop on Cryptographic Techniques and E-Commerce* (1999), pp. 131–138.

[36] PINGEL, I., IRVING, A., GENERALMANAGER, WIKINAUT, TINLOAF, FARB, M., AND JPOPPLEWELL. Fingeprint exchange - issue #826 - whispersystems/textsecure - github.

[37] PLASMOID. Fuzzy Fingerprints: Attacking Vulnerabilities in the Human Brain. `http://www.thc.org/papers/ffp.html`, Oct. 2003.

[38] ROGERS, M., AND PERRIN, T. Key-Fingerprint Poems. `https://moderncrypto.org/mail-archive/messaging/2014/000125.html`, 2014.

[39] RYAN, M. D. Enhanced certificate transparency and end-to-end encrypted mail. In *NDSS* (2014), NDSS.

[40] SHAY, R., KELLEY, P. G., KOMANDURI, S., MAZUREK, M. L., UR, B., VIDAS, T., BAUER, L., CHRISTIN, N., AND CRANOR, L. F. Correct horse battery staple: Exploring the usability of system-assigned passphrases. In *Proceedings of the Eighth Symposium on Usable Privacy and Security* (2012), ACM, p. 7.

[41] UNGER, N., DECHAND, S., BONNEAU, J., FAHL, S., PERL, H., GOLDBERG, I., AND SMITH, M. Sok: Secure messaging. In *Security and Privacy (SP), 2015 IEEE Symposium on* (2015), IEEE, pp. 232–249.

[42] VASCO.COM. `http://www.vasco.com/company/about_vasco/press_room/news_archive/2011/news_diginotar_reports_security_incident.aspx`, Sept. 2011.

[43] WHATSAPP. Encryption Overview. `https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf`, Apr. 2016.

[44] YLONEN, T., AND LONVICK, C. The Secure Shell Transport Layer Protocol. RFC 4253, Jan. 2006. Updated by RFC 6668.

[45] ZIMMERMANN, P., JOHNSTON, A., AND CALLAS, J. ZRTP: Media Path Key Agreement for Unicast Secure RTP. RFC 6189 (Informational), Apr. 2011.

A Appendix

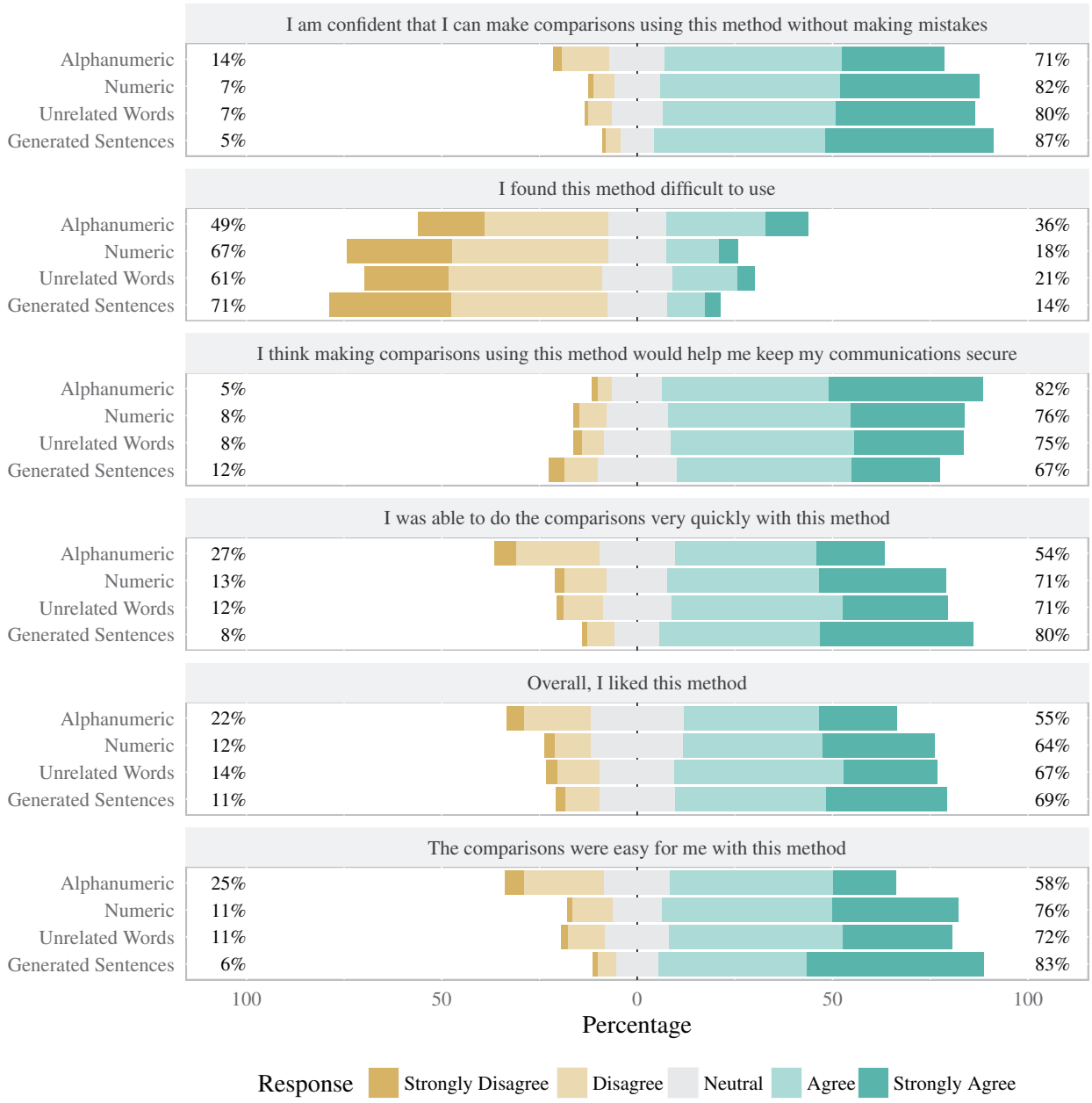
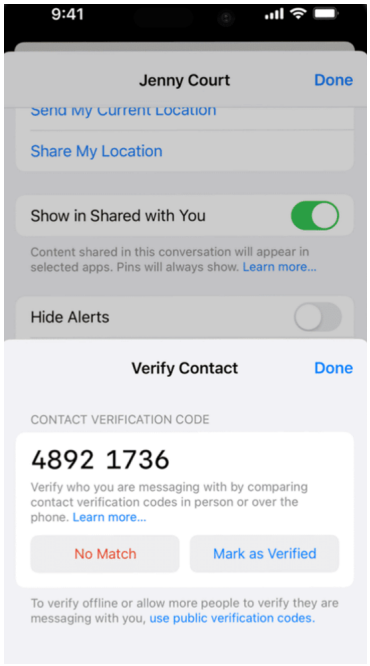


Figure 6: Survey results for all statement ratings

3.2 Implications and Future Directions

(a) `WhatsApp`'s key verification with the possibility to use the camera

(b) `Signal`'s key verification with the possibility to use the camera



(c) `iMessage` key verification, although using SAS to shorten the fingerprint in synchronous modes [37]

Figure 3.1: Manual key fingerprint verification in popular messengers in 2024. Screenshots taken from the EFF tool guide and Apple's official `iMessage` documentation [26, 38]

The findings highlight significant usability and security gaps in traditional hexadecimal formats. The recommendation is to adopt sentence-based or numeric representations, demonstrating higher attack detection rates and better user acceptance. Some participants expressed frustration with hexadecimal / `base32` representations, citing difficulty in recognizing errors during comparison tasks. In contrast, sentence-based encodings were described as "intuitive" and "easier to relate to," indicating a clear preference for language-based formats that align better with everyday cognitive patterns. By addressing these usability challenges, the study contributes to the broader effort of making secure communication more accessible and reliable for users.

While sentence-based and numeric representations demonstrate significant improvements in usability and security, their scalability in real-world deployments requires further exploration. Key considerations include backward compatibility with existing tools, integration with current cryptographic libraries, and potential resource constraints for low-power devices.

Backward compatibility and tool integration might also be why traditional security tools such as `OpenSSH` and `OpenPGP` remained unchanged with their TOFU combined with the hexadecimal key fingerprint representation. By contrast, most instant messaging tools adopted the numeric representation as their default, although it is mainly hidden deep in the expert settings. As depicted in Fig. 3.1, today, most of the popular messengers, such as `WhatsApp`, use the numeric fingerprint format, which validates our findings that the numeric representation seems to be the best trade-off approach when facing language barriers. To reduce potential mistakes, these interfaces are typically enhanced with the capability to scan QR codes or exchange files over NFC directly.

In cases where the spoken languages can be derived from context, sentence-based encodings reduce susceptibility to phishing attacks by presenting fingerprints in a familiar, non-technical format that is harder to spoof convincingly (see Fig. 3.1(c) where just one word would offer the same security). Moreover, the improved attack detection rates observed in our study suggest that these encodings offer enhanced protection in scenarios where users must verify authenticity under adversarial conditions.

The study's methodology and online experiment platform have proved valuable beyond the secure messaging domain. While requiring substantial adaptations and the creation of a new study design, the core experimental platform with the statistical framework was successfully reused for usability studies in malware analysis with Yakdan et al. [7], demonstrating how systematic user study approaches can benefit security research across different domains.

CHAPTER 4

Secure Messaging Mental Models

As outlined in Chapter 2, the analysis questions whether there might be significant gaps in how end-to-end security features are presented and communicated to end users. This chapter builds on these findings by examining how users conceptualize secure messaging systems and their features, focusing on understanding and addressing misconceptions. The qualitative study presented in this paper explores the evolution of user perception and mental models of secure messaging following the widespread adoption of end-to-end security in popular applications like `WhatsApp` or `iMessage`. With `WhatsApp` 's implementation of the `Signal protocol` as the default in 2016, end-to-end encryption became accessible to the masses, accompanied by explicit notifications informing users about the encryption of their communication. Despite this significant shift in messaging security, the paper investigates whether users' understanding, trust, and awareness of encryption have improved.

Using a qualitative methodology, the research compares users' mental models before and after the introduction of *mass messenger encryption* (MME, pre- and post-MME). Findings reveal a persistent lack of trust in encryption technologies among users, largely driven by misconceptions about its capabilities and the strength of cryptographic protections. Furthermore, the majority of participants remained unaware of the encryption mechanisms by default in `WhatsApp`, despite abundant in-app notifications and media attention. This lack of awareness and understanding underscores the need for improved communication about encryption and highlights barriers to user confidence in secure messaging systems. These findings are particularly relevant given the growing importance of secure communication in both personal and professional environments, where users' trust and understanding directly impacts the effective use of security features. The research contributes valuable insights into the interplay between usability, trust, and awareness in adopting secure technologies.

This in-depth qualitative study is the first to investigate how users conceptualize, understand and perceive secure messaging over time. It examines their mental models – the internal constructs people form to understand and explain how a system works and predict its behavior. Comparing mental models from two points in time offers a unique longitudinal perspective on how widespread encryption impacts user understanding and trust. The study analyzes the discrepancies between users' conceptual understanding and established security practices implemented in widely adopted messaging applications. It also addresses a general mistrust in security capabilities, with users overestimating attackers' capabilities and underestimating the effectiveness of cryptographic protections.

4.1 Peer-Reviewed Publication 3 | In Encryption We Don't Trust: The Effect of End-To-End Encryption to the Masses on User Perception

Authors' Contributions

The work presented in this chapter is based on our paper (In Encryption We Don't Trust: The Effect of End-to-End Encryption to the Masses on User Perception) published at the 2019 IEEE European Symposium on Security and Privacy (EuroS&P, DOI: 10.1109/EuroSP.2019.00037) [2]. The authors' contributions that are relevant to the contents of this chapter are as follows:

- **Sergej Dechand** I served as the main author of this work, leading all aspects of the research, including the conceptualization, design, and execution of the study. I was responsible for designing the user study, analyzing the results, and creating the overall structure of the published work. My coordination and contributions helped ensure the project's successful completion and its scientific impact.
- **Alena Naiakshina**, **Anastasia Danilova** carried out the user interviews and diligently processed and organized the interview data. Their efforts and attention to detail were essential for collecting high-quality qualitative data and supporting the analysis.
- **Matthew Smith** Provided valuable feedback throughout all phases of the project, offering conceptual input and suggestions that improved the structure and clarity.

In Encryption We Don't Trust: The Effect of End-To-End Encryption to the Masses on User Perception

Sergej Dechand
University of Bonn
Bonn, Germany
dechand@cs.uni-bonn.de

Alena Naiakshina
University of Bonn
Bonn, Germany
naiakshi@cs.uni-bonn.de

Anastasia Danilova
University of Bonn
Bonn, Germany
danilova@cs.uni-bonn.de

Matthew Smith
University of Bonn, Fraunhofer FKIE
Bonn, Germany
smith@cs.uni-bonn.de

***Abstract*—With WhatsApp's adoption of the Signal Protocol as its default, end-to-end encryption by the masses happened almost overnight. Unlike iMessage, WhatsApp notifies users that encryption is enabled, explicitly informing users about improved privacy. This rare feature gives us an opportunity to study people's understandings and perceptions of secure messaging pre- and post-mass messenger encryption (pre/post-MME). To study changes in perceptions, we compared the results of two mental models studies: one conducted in 2015 pre-MME and one in 2017 post-MME. Our primary finding is that users do not trust encryption as currently offered. When asked about encryption in the study, most stated that they had heard of encryption, but only a few understood the implications, even on a high level. Their consensus view was that no technical solution to stop skilled attackers from getting their data exists. Even with a major development, such as WhatsApp rolling out end-to-end encryption, people still do not feel well protected by their technology. Surprisingly, despite WhatsApp's end-to-end security info messages and the high media attention, the majority of the participants were not even aware of encryption. Most participants had an almost correct threat model, but don't believe that there is a technical solution to stop knowledgeable attackers to read their messages. Using technology made them feel vulnerable.**

I. Introduction

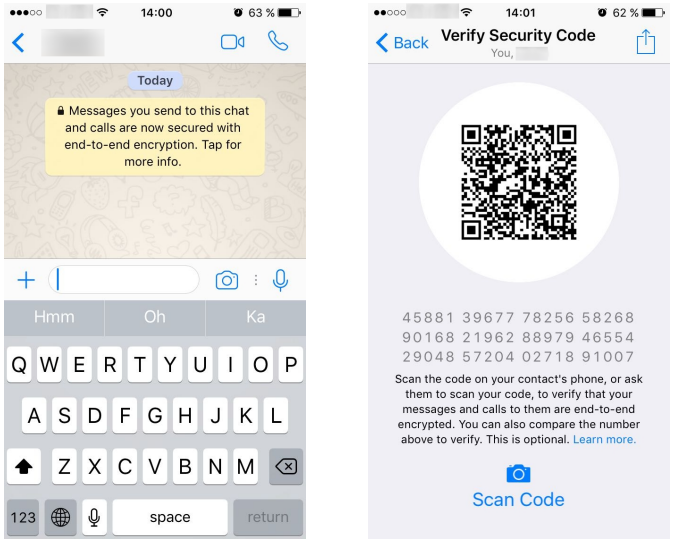
Before 2016, most mobile communication, such as short messaging services (SMS) or messaging apps, did not provide any end-to-end encryption. One popular exception was `iMessage`. However, it still lacked a user interface for key authentication, thus creating a vulnerability to man-in-the-middle attacks. It also did not advertise any of its security features. With `WhatsApp`'s introduction of the `Signal` protocol as its default, authenticated end-to-end message encryption suddenly became available to the masses [43]. `WhatsApp` notifies users that their messages are end-to-end encrypted every time a new conversation is opened and offers an optional authentication process based on quick response codes or key fingerprints [43, 13].

Even though various cryptography protocols as OpenPGP, off-the-record messaging, and Tor, have been available for decades [8, 5, 2], they have all failed to achieve widespread adoption due to usability issues, such as key management and key authentication and sometimes even unreliable message delivery. Previous work has shown that end users struggle with security tools for email encryption [44, 16, 32, 24]. The general perception is still that usability problems are to blame for the woes of encryption solutions. However, we propose that perceptions and mental model issues might remain hindrances even when security mechanisms provide good usability. The introduction of a very usable end-to-end encryption solution provides the ideal opportunity to study this aspect.

To capture users' perception and understanding, we conducted a qualitative user study based on interviews to illustrate average users' mental models *before* and *after* the *mass messenger encryption* event (pre- vs. post-MME). The methodology and the first 11 interviews were conducted in July and August 2015. In January and February 2017, approximately nine months after `WhatsApp`'s introduction of end-to-end encryption, we conducted another set of interviews (post-MME) to be able to compare mental models before and after the introduction of this widespread encryption mechanism. In 2017, we re-invited some participants from the 2015 group and invited 11 new participants to enable a direct comparison. The results of both groups were analyzed individually. We chose this mixed set because taking part in the *pre-MME* study was very likely to have shaped the participants' opinions. Recruiting both old and new participants allowed us to see both a within group and a between group view of this event.

Our study was aimed at identifying aspects helping messenger developers implement usable and secure software, so it was important to understand how people imagine the process of sending and receiving mobile messages. We considered two main methods: classic text messages (SMS) and `WhatsApp`. Rather than technical details, we focused on whether end users understand high-level concepts and more importantly, the implications of encryption. The first part of our study was started before the widespread adoption of end-to-end encryption services, so we considered the transition of users' mental models after the introduction of *mass messenger encryption*. Our key findings are:

1) **Users do not trust encryption**. Most of our participant were well aware of who is theoretically capable of eavesdropping on their communication. However, they *overestimated the capabilities* of potential attackers, e.g.,



(a) WhatsApp Encryption (b) WhatsApp Authentication

Fig. 1: WhatsApp's implementation of end-to-end encryption based on the Signal protocol [35].

skilled neighbours could break the encryption and read their messages. At the same time, they *underestimated cryptographic capabilities*, sharing the consensus that there is no technical solution stopping skilled attackers from breaking encryption. Most stated that they had heard about encryption, but only a few understood the implications, even on a high level.

2) **Users lack awareness**. Despite WhatsApp's introduction of end-to-end encryption in April 2016, most users were still unaware of it nine months later. More surprisingly, when directly asked about the info message (see Figure 1a), many participants reported noticing it but chose to ignore it or failed to understand the implications correctly.
3) **SMS is more secure than WhatsApp.** Almost all the participants believed that SMS is more secure than WhatsApp. The participants justified this belief with two reasons: first, they considered the Internet to be *evil* and second, they held opinion that SMS messages seldom cross country borders.
4) **Users do not feel targeted.** In general, government and special service surveillance was perceived critically, especially if executed by foreign countries. However, most participants believed that they would never be targeted. A few participants saw benign, exceptional cases for the police in fighting crime and terrorism but rejected the idea of mass surveillance. Finally, the majority thought that governments and hackers are able to break any kind of encryption.
5) **Study participation raises awareness and attention.** The participants who took part in our 2015 study were more aware of encryption and security in the second interview. For instance, some used alternative messengers such as Threema, Signal and Telegram. This was mainly because the participants were critical of WhatsApp's acquisition. In contrast, the newly-invited participants never used other apps except for LINE.

The rest of the paper is structured in the following way: in Section II we review related work. Section III describes our methodology and its limitations. In Section IV we present the results of our focus groups. In Sections V and VI we compare the results of the pre- and post-MME single interviews. An extra section (Section VII) is dedicate to the 4 re-invited participants. Finally, in Sections VIII and IX we discuss our results, provide recommendations, and conclude.

II. Related Work

In this section, we discuss the usage of current security solutions in messaging and prior attempts to improve their usability. We also discuss work on gathering user perceptions and obtaining people's mental models in the human-computer interaction and especially the usable security and privacy community.

A. Communication Security

Starting with the commercially available email encryption PGP [8], a large body of security communication tools (e.g., Tor, Off-the-Record Messaging (OTR) and others) followed [14, 37, 5]. However, the famous Johnny user studies and follow-up work confirmed that end users are overwhelmed with most security tools [44, 16, 32, 15, 24]. For instance, some required hard key management [44, 8, 2], and others had session issues with less reliable mobile environments [5, 2, 33, 38, 34].

Consequently, up to the beginning of 2016 the majority of exchanged personal messages including email and messengers remained unsecured. With the growing popularity of the Signal messenger and its protocol providing similar security features as OTR and a better integration in mobile environments [36, 34, 38], WhatsApp integrated it in early 2016 [43]. Similarly to iMessage, end-to-end encryption is activated by default. Additionally, it provides an optional authentication verifying the absence of man-in-the-middle attacks and can be done by using QR codes or comparing numeric hash representation with 60-digit numbers, as shown in Figure 1b [43, 13].

B. Mental Models

To elicit users' understanding and perception in an area, a commonly used method in psychology, and recently in the usable security and privacy area, is based on mental models [9, 28]. More specifically, mental models describe what exactly users think about a specific topic or problem; they disclose that people form diagrams in their minds, which help them to build their understanding of the world and to solve the problems that emerge when they have to interact with complex systems [20, 28].

Various usable security researchers adopted mental models of users' understanding regarding technologies based on the Internet [42, 41, 21, 31, 21, 22, 29]. Having unrealistic mental models regarding communication may induce a risk based on

users' activities. Previous work specifically named some of the misunderstandings [41, 40]. We propose that developers should take users' mental models in consideration when designing messaging protocols rather than assuming that their users understand cryptographic details. This approach could lead to a greater acceptance of secure messaging software as users will not be confused or frustrated by the complexity of public key encryption.

Asgharpour et al. [3] found that security experts and non-experts have different mental models of security risks. They proposed that risk communication should not be designed based on experts' mental models, as it is mostly done, but rather should be designed based on end users' point of view.

Bravo-Lillo et al. [7] designed a mental model in order to understand how users think about computer security warnings. The study compared advanced and novice user's behavior dealing with computer security warnings and was supposed to help developers to improve their warning design.

Wash [41] conducted a study in order to understand the users' mental models of attackers and security technologies, and to explain why users strictly follow some security advice from computer security experts and ignore others.

Furthermore, a multi-method user study was conducted by Vaniea et al. [39] to get a better understanding of how people make decisions about software updates.

C. User Perceptions of Secure Messaging

Adoption criteria of secure messaging tools and services as well as social influence on users decision of security tools and security behavior have been intensively investigated in the past [17, 11, 23, 12]. For example, De Luca et al. [23] conducted an online survey with 1500 participants, making a quantitative analysis of *how much of a role* security played in people's decisions to use a mobile messenger. As often suspected in the usable security field, their results suggest that security plays a minor role, and the available contacts in a messenger clearly dominate their decision making.

Furthermore, Bai et al. [4] investigated whether users are able to understand the difference between a *key-exchange* and a *key-directory* model. Their results show that users are well aware of usability and security trade-off, but favored key directories.

Renauld et al. [31] investigated why the usage of end-to-end encrypted emails is limited. Their study revealed issues such as incomplete threat models, misaligned incentives, and a general absence of understanding of email architecture. In contrast to their work, we did not stop the interview sessions if the term encryption was not mentioned by participants. Rather, we asked our participants about encryption and how they imagine it is supposed to work. This procedure ensures that participants, who have a general idea of encryption, but might not conceive of the term, could still express their knowledge.

Abu-Salma et al. [1] conducted 60 qualitative interviews analyzing the obstacles to the adoption of mobile messaging applications. Their participants were additionally advised to explain encryption. Interestingly, the participants indicated telephony as a service, which is more secure than text messaging. Although the study of Abu-Salma et al. was conducted with another methodology and in the UK, the findings show remarkable similarities to our study. For instance, their participants observed SMS as more secure than instant messaging, an opinion which was also shared by our participants in Germany.

III. Methodology

To capture end users' understanding of mobile communication, we conducted semi-structured interviews with 22 participants from Germany. We conducted our interviews before and after `WhatsApp`'s introduction of end-to-end encryption, so we could observe changes in end users' mental models. The *pre-mass messenger encryption (pre-MME)* interviews were conducted with 11 participants in July and August 2015. The *post-mass messenger encryption (post-MME)* interviews were conducted with 11 new invited and 4 re-invited participants in January and February 2017.

Our study was designed to offer insights into the following aspects of users' perceptions of encrypted messaging:

- **Understanding of the architecture:** how users imagine mobile communication works and whether they distinguish between SMS and instant messaging.
- **Threat model:** who users assume is able to eavesdrop and whether there are ways to prevent it.
- **Understanding of encryption:** how do users imagine encryption and authentication and whether they understand the implications.
- **Impact of end-to-end encryption:** change in users' understanding of mobile communication after `WhatsApp`'s introduction of end-to-end encryption (post-MME).

A. Interview Guideline Design

A major challenge in designing semi-structured interviews to capture users' understandings and mental models is to design an interview guideline covering the necessary topics. Previous work in this area either opted to use more open questions with a large unstructured part, or refined their interviews based on pre-studies [1]. To improve this process, we used *focus groups*, to iteratively develop a guideline for individual interviews [30]. Focus groups are an approved methodology with a long history of use in psychological studies to collect qualitative data [26, 19]. A skilled moderator leads an interactive group discussion with multiple participants by using a guideline of a set of carefully predetermined structured questions. We used the focus group methodology as a more refined and structured way of a pre-study justifying the final interview guideline [30]. We performed 3 focus group iterations, adapted and improved our interview guideline with each iteration. In addition to the moderated discussion, we asked our participants to support their thoughts with drawings, in order to let them visualize their perceptions [31, 21, 22, 29].

B. Focus Groups

We conducted 3 focus group iterations in 2015 [27]. For the recruitment of participants we used the theoretical sampling

approach [30]. The first focus group was recruited at our university. Two student assistants invited participants who were on campus. To get a more diversity for the second and third focus groups, we recruited students of non-technical degrees from other universities and older people with less academic background. Participation was rewarded with snacks and refreshments. A single, skilled researcher led and moderated the focus group discussions by using a carefully predetermined semi-structured guideline. The moderator was accompanied by an assistant researcher, who served as a note-taker for important observations.

The first part of the guideline was used to gain a first glimpse of the participants' mental model of mobile messaging communication. An empty sheet of paper with only two individuals, Alice and Bob on it, was handed out to the focus groups. The participants were asked to describe how Alice and Bob could communicate with each other and to draw stations important for their communication. The second part covered perceptions of encryption and authentication. The participants were asked to think about eavesdroppers and to highlight threatening spots in their drawings. They were also asked how eavesdropping might be prevented. To gain insight into the participants' perceptions of Man-in-the-Middle scenarios, we asked how Alice could be sure that the person she was communicating with was really Bob, in other words, how it can be guaranteed that communication partners are the persons whom they claim to be. All the questions were open-ended to encourage the participants to express their thoughts in detail. The guideline was refined after each focus group, e.g., by adding relevant questions and removing less relevant questions.

The first group's participants were undergraduate computer science students (first and second year), 1 female and 6 male with an average age of 21 years. The second focus group had 6 participants (2 male and 4 female) with an average age of 24 years. The group consisted of students (e.g., medicine, architecture) and employees (e.g., logopedics) in fields less related to computer science. Considering the younger age and the student status of most participants, we invited older participants with less technical backgrounds (cleaner, scholar, housewife, pastor, vendor, geriatric nurse, mechanic and a carpenter) to join the third focus group. This group consisted of 8 people, 3 male and 5 female with an average age of 47.5 years. Each focus group discussion lasted approximately 60 minutes.

Based on the three iterations of focus groups, we created a semi-structured guideline for individual interviews (see Appendix A). For instance, instead of using unfamiliar persons such as Alice and Bob, we decided to ask the participants how they thought communication with their friends worked. Additionally, the focus group participants mentioned SMS and `WhatsApp` as communication applications they used often. So, we concentrated on these applications in our individual interviews. Finally, most participants in the focus groups were confused by the expression *authentication*. We, therefore, neither mentioned the term *authentication* nor used a helping scenario in the individual interviews.

C. Individual Interview Participants

For the individual interviews, we recruited participants by placing advertisements on *eBay Kleinanzeigen* (a German private marketplace website similar to *Craigslist*). We also asked the participants in the 2015-group whether they would like to participate in our study in 2017 again. Those interested were invited to individual interviews to our university. Participation in the interviews was rewarded with 15 Euro. A table showing the demographics of the participants can be found in the Appendix E.

Pre-MME (2015): 11 participants were invited to talk about messaging architecture and security. The sample consisted of 7 females and 4 males with an average age of 29 years. The average self-reported level of technical knowledge (see Section III-E) was 8 (medium). All participants had diverse professions, like students, office assistants, food service workers, scientific assistant etc. The student participants studied business informatics, computer science, economics and media management.

Post-MME (2017): We established two groups in 2017: (1) 4 re-invited participants from the 2015 study[1] and (2) 11 new participants. 6 female and 5 male, with an average age of 31 years. Four of the new invited participants were students in computer science unrelated programs including economics, medicine, agriculture, and teaching. Three participants were self-employed in the beauty treatment, economics and health care industries. One participant reported to be working as a paramedic and another participant was unemployed. Their average self-reported level of technical knowledge was 6 (medium).

In both studies, the coders discussed after each evaluated interview whether theoretical saturation [6, 10] was reached or whether more participants were needed to be sampled. In case of a disagreement between the coders, further participants were invited. Saturation was reached when both coders agreed that no new codes/themes emerged.

D. Individual Interview Procedure

We conducted 11 individual interviews in 2015 before `WhatsApp`'s introduction of end-to-end encryption in 2017. After this, we conducted further 15 individual interviews to identify the mental model changes. All the interviews were conducted in German and followed the same methodology and guideline. To avoid experimenter bias, the same interviewer conducted them. Each interview lasted from 30 to 40 minutes.

First, the participants had to answer demographic questions and how often they use SMS and `WhatsApp` (see Appendix B). Next, the participants were presented a brief scenario:

> *You would like to communicate with a friend using your mobile phone. You would like to send a text message to your friend via Short Messaging Service*

[1]We invited all participants from the original group in order to examine whether their mental models changed after the introduction of E2E encryption. Only 4 of 11 participants were willing to join the new study. We did not receive any answer from the other 7.

(SMS) or via WhatsApp. Please draw how you think the communication looks like.

The interview technique was designed to give as little hints as possible. As in the focus groups, the participants were handed a sketch of two individuals. If participants indicated differences in their mental models of SMS and WhatsApp, they were requested to make two drawings. Also, in the case of space limitations, we handed out another blank sheet of paper for the second service. The participants were also asked what parties they thought could eavesdrop communication. They indicated stations where these parties could eavesdrop. If participants did not mention encryption, we asked them whether they had heard of it and requested that they explain and draw how they imagined encryption. The participants had to indicate whether they believed that WhatsApp or SMS messages are encrypted and if they knew of or used any apps providing encryption. Finally, the participants were asked how a message could be explicitly assigned to a person. The guideline for the individual interviews can be found in the Appendix A.

E. Evaluation

We conducted a qualitative analysis for the single interviews and focus groups. All interviews were transcribed in German. Two researchers independently coded and evaluated every transcription following a three-step procedure: (1) using *open coding* to develop concepts and categories, (2) developing connections among the categories, and (3) drawing conclusions by assigning users' individual statements to the categories [19]. Differences between the two coders were resolved through discussion to avoid interpretation bias. Relevant statements and expressions from the participants' mental models were translated into English by the same researchers.

To report statements by the participants from single interviews, we label them P1-P11 in the pre-MME (2015) group and from N1-N11 for the post-MME (2017) group. The 4 re-invited participants are referred to with their initial labels from 2015: P1, P5, P6, and P9. In relevant cases, we also report how many participants stated specific themes to indicate their frequency and distribution, although we do not aim to generate quantitative results.

To assess participants' technical experience, they rated statements on a 5-point Likert-Scale (see Appendix C). We calculated a ratio indicating the technical background of participants from 0 to 12 (low: 0-3, medium: 4-8, high: 9-12). More details on the calculation can also be found in the Appendix C.

F. Limitations

Despite high individual demographic differences, our sample contains only participants with a German cultural background. The results might not be the same in different cultures. Based on historic background, language nuances but also local media exposure in Germany, the results in this paper might not be applied to other cultures directly.

Furthermore, we used *eBay Kleinanzeigen* (similar to *Craigslist*) to recruit our participants. Thus, we have a self-selection bias and could only sample in the population of people using this service.

A qualitative approach does not claim to provide generalization. Qualitative work such as this is only the first step on the road to generalizable results as it can help exploring reasoning and views of participants.

G. Ethics

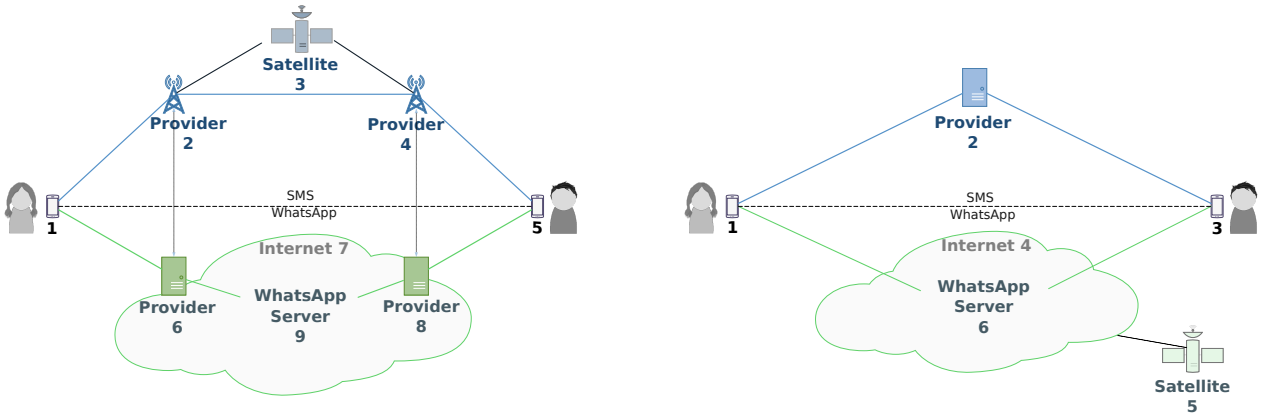
Since our study was conducted in Germany, it was not required to pass an IRB review. However, our study complies with the strict German privacy regulations. At the end of the first study we asked if participants would be willing to be contacted again. The contact data was stored separately. The data was collected anonymously and the participants were informed about withdrawing their data during or after the study.

IV. Focus Group Results

Although the participants in the focus groups were from different age ranges and professions, all the groups mentioned some general points [27]. Most participants appeared skeptical of and expressed insecurity when using mobile messaging communication. They believed that almost anyone could eavesdrop on their messages at every station if someone, whether companies or individuals, was eager enough or had the proficiency to do so. Although all the participants had security concerns they nevertheless still actively used services they mistrusted on their mobile phones. However, a number of participants stated that they did not send any sensitive information by mobile phone. For that, they preferred to meet in person or write emails. Several reasons motivated this behavior: some participants did not understand the system behind SMS and WhatsApp, and others did not know how to protect their messages from eavesdropping.

Various participants were concerned about WhatsApp and its security, but did not switch to secure messengers because all their contacts still used WhatsApp. Nevertheless, the participants stated that if more of their contacts and other people started to use secure messaging apps, they would too. All the groups mentioned encryption in connection to preventing eavesdropping.

Some believed that not even encryption can protect their messages. Most importantly, the second and third groups showed that the majority of users were not familiar with public-key cryptography. The only concepts they could imagine involved passwords and symmetric encryption. The participants suggested that the key exchange should involve a personal meeting or sharing a secret. One interesting aspect mentioned by the second group was that mobile numbers are connected to WhatsApp. The second group seemed to trust this kind of control. In contrast the first focus group stated that this is not a method trustworthy at all. The third focus group became aware of the potential threat from the man-in-the-middle attacks by going through a simplified scenario with Alice, Bob and Eve. This indicates that users are capable to understand the threats without effort, but didn't realize the threat on their own.



(a) Before End-To-End Security

(b) After End-To-End Security

Fig. 2: Aggregated Mental Models of Individual Interviews.
The upper parts of the illustration (blue) refer to messaging via SMS whereas the lower illustration (green) refers to messaging via `WhatsApp`. Stations mentioned by our participants were indicated with individual numbers (see Appendix D).

Importantly, all the focus groups mentioned the NSA and governments as potential adversaries. Obviously, most the participants had noticed the NSA spying scandal and realized that their messages could easily be captured by different parties, particularly the U.S. government. This finding shows that news and stories influence mental models.

V. Individuals' Architecture Understanding

Most importantly, our analysis shows that even though end-to-end security did not gain high attention among our users, all the participants were aware of the interception risks of both systems (`WhatsApp` and SMS). We present the results in subsections focusing on different areas of mobile communication. In each subsection, we describe the results from our first iteration *before the introduction of end-to-end security to the masses (2015)*. If the users' mental models changed in the follow-up iteration *after the introduction of end-to-end security to the masses (2017)*, we also discuss the differences in a separate paragraph. For the comparison of 2015 and 2017, we consider the results of the 11 participants in 2015 and the 11 new participants in 2017.

A. Messaging Architecture

Our main goal was to illustrate the users' understandings of messaging infrastructure. Therefore, we asked participants of how they imagined messaging communication between two persons. The mental model drawings of the individual participants can be found online.[2] Ignoring minor details discussed in the following, all participants' mental models had great similarities. We, therefore, summed the individual mental models into a single aggregated model of messaging architecture by including stations mentioned by our participants, and summarizing terms, such as, server, provider, and radio mast referred to as providers' base stations by the participants (see Appendix D). Figure 2a shows the summarized mental model of all the 2015 participants and Figure 2b for 2017. Almost all the participants distinguished between communication via SMS and `WhatsApp`.

1) Pre-MME (2015): In 2015, the participants mentioned different intermediate stations. First, all the participants mentioned a mobile provider base station (specifically referred as a radio mast by P4, P7, P8, P10, and P11) as a point, through which an SMS message passed on its way to the other communicating peer. P1, P2, P5, P6, P7, P10, and P11 included one or more mobile providers or ISPs (Internet Service Providers) in their visualizations for SMS. Moreover, P1, P3, P5, P6, P8, P9, and P11 mentioned servers as an intermediate stop in the architecture. P3 and P6 distinguished between servers in foreign countries and Germany. Some participants (P2, P8, and P10) believed that satellites were involved in mobile communication.

Asked to extend their drawing of SMS or make another diagram for messaging via `WhatsApp`, almost all the participants identified the Internet as the basic difference between the two (Figure 2a, lower illustration):

> "Yes, in my opinion, it looks different because [the text messages] are sent through the Internet, which is an external service. Thus, it definitely works in a different way." (P1)

P1, P8, and P9 differentiated between mobile servers for SMS and Internet servers for `WhatsApp`. P8 stated that the Internet works via satellites. Additionally, the `WhatsApp` provider often referred to as a server by the participants, was specified as the core difference from SMS.

[2]The original mental model drawings of the individual participants were handwritten and in German. In order to ensure participants' anonymity we transcribed all drawings. Additionally, they were translated in English: https://net.cs.uni-bonn.de/fileadmin/user_upload/smith/EuroSP19_MentalModel_Drawings.pdf

Adversary	Stations Pre-MME	#P	Stations Post-MME	#N
Mobile Provider	2, 2 ↔ 4, 3, 6	4	1 ↔ 2, 2, 2 ↔ 3	4
German Government	2, 3, 4, *	5	2	3
German Law Enforcement	1, 1 ↔ 2, 2	3	1 ↔ 2, 2 , 2 ↔ 3	2
US Law Enforcement	2	2	3	1
German Intelligence Agencies	2 ↔ 4, 2	4	1 ↔ 2, 2↔ 3	2
Foreign Intelligence Agencies	2	2	2 ↔ 3	3
Hackers	1, 2, 2 ↔ 4, 3, 7	7	1 ↔ 2 , 2 , 2 ↔ 3, 3	8

TABLE I: SMS adversaries mentioned by our participants before and after the introduction of MME. *Stations Pre-MME* (related to Figure 2a) and *Stations Post-MME* (related to Figure 2b) show the stations where participants (#P of 11/#N of 11) expect adversaries to eavesdrop on their messages. Attackers indicated between two stations were marked by arrows; * indicates "everywhere".

Adversary	Stations Pre-MME	#P	Stations Post-MME	#N
WhatsApp	7, 9	9	1 ↔ 3 , 4 , 5 , 6	9
Mobile Provider	6	2	1 ↔ 4	2
German Government	1, 2, 9, *	3	2 ↔ 3	3
Foreign Governments	6 ↔ 9, 9, *	6	1 ↔ 3, 6	4
German Law Enforcement	1 ↔ 9, 2, 9	3	1 ↔ 3, 3	1
Foreign Law Enforcement	9	3	1, 3	1
German Intelligence Agencies	2	5	6	3
Foreign Intelligence Agencies	6 ↔ 9, 7,	2	4, 6	5
Hackers	1, 2, 9	6	1, 3, 3 ↔ 6, 4, 6	9
Commercial Companies	7 ↔ 9, 9	5	3 ↔ 6, 6	2
Facebook/Zuckerberg	7 ↔ 9	4	1, 3 ↔ 5, 3 ↔ 6, 6	4
Spying Apps (e.g., Viruses)	1	4		0

TABLE II: `WhatsApp` adversaries mentioned by our participants before and after the introduction of MME. Same table structure as in Table I.

2) Post-MME (2017): We found only minor differences between the mental models of the participants in 2015 and 2017 (new participants). The 2015 participants often called several provider base stations radio masts or towers. This changed slightly in 2017, when most of the participants specified only one provider base station for SMS (N1, N3, N4, N6, N7, N9, and N11). N2, N5, and N8, however, still identified more than one provider base station. For example, N2's mobile communication was represented by only three single radio masts connected to each other. Finally, in 2015 *satellites* were associated with mobile communication for text messages, while in 2017 the participants N2-N4 believed satellites were also required for the Internet.

B. Threat Model

Table I and Table II summarize the threat model for SMS and `WhatsApp` shared by the participants. To give an impression of how often adversaries were mentioned, we also provide the number of participants (#P/#N of 11) who mentioned them.

1) Pre-MME (2015): When investigating the participants' threat models, they all agreed that eavesdropping was possible at various stations involved in communication channels. This position captured the generally skeptical sentiments all the participants have shown when speaking of security of mobile communication.

The most significant result of our study was that messages sent via the SMS were felt to be more secure than `WhatsApp` messages. Various reasons were mentioned. First, because `WhatsApp` messages are sent via the Internet, participants often stated that `WhatsApp` is more prone to eavesdropping. Second, banks and post offices also use SMS in order to send important data (P7). Third, P7, P8, P9 and P10 believed that SMS is more difficult to be hacked. Almost all participants mentioned `WhatsApp` and hackers as potential adversaries.

However, P6 and P10 believed that ordinary people are not likely to be targeted for surveillance. They stated that either rich, famous people, politicians or criminals are targeted: "I do not think that everybody is being monitored. I'm an unimportant person and don't write dangerous or bad stuff" (P6).

In the following, we discuss the adversaries mentioned by the participants in more detail.

Mobile Providers and ISPs: P2 assumed that mobile providers were not allowed to access the messages. In contrast, P7 explained that mobile providers maintained relations with governments and spies, and P7 argued that governments did not necessarily need to cooperate with mobile providers to access the messages. Also, P8 added that hackers and governments could access radio providers.

Government and Law Enforcement: Half of the participants mentioned German and foreign governments (e.g., U.S. and Russia) as potential adversaries. Most participants stated that the German government could read their messages when they use SMS for mobile communication. P2 speculated that the German government could only read SMS messages at satellites

and that it is impossible to protect SMSes from governments because it is unfeasible to control the satellites. Furthermore, P8 noted that German government could install malware, such as key loggers and trojans, on mobile devices.

P3 and P8 mentioned the Pentagon as an eavesdropping party in the U.S. government. P8 and P9 both mentioned Asian and Chinese governments. P9 also assumed that the Russian government could eavesdrop. Additionally, the participants mentioned the U.S. law enforcement agencies: FBI, CSI, CIA.

Some participants (P8, P10, and P11) distinguished between permitted data access and illegitimate data access: when someone collects data but does nothing with it vs. access data without your knowledge and deliberately spies on the data. For example, German and U.S. law enforcement and the German government were mentioned as eavesdroppers in context of terrorism and crime prevention (P3, P5, P6, and P9-P11). P10 believed that key words such as *bomb* and *terrorist* are logged and counted. If a log shows an unusual large amount of such key words, the sender is monitored and controlled.

Intelligence Agencies: P3 remembered that the NSA eavesdropped on German Chancellor Angela Merkel. The German media reported on mass surveillance. More than half of the participants (P1, P7-P11) mentioned the NSA as a potential eavesdropper in `WhatsApp` communication. The German Federal Intelligence Service (BND) was also explicitly mentioned by P8 and P10.

Hackers: Almost all participants (P3, P4, P6-P11) identified hackers as a party, which can eavesdrop text messages. In particular, P4 mentioned Russian hackers. Further, P6 differentiated between "good hackers" (searching for exploits and vulnerabilities) and "bad hackers" (stalker and criminals in general). P7, on the other side, referred to hackers as "teenager without friends" or "[someone] connected to the mafia." P9 explained that some hackers sell the data, which they obtain. Hackers who are collecting data for marketing were also mentioned by P10. A larger part of our participants believed hackers are able to access the messages on all stations.

Commercial Companies: P5 believed that text messages are scanned. Further, some participants assumed that commercial companies like Amazon (P2), newspaper (P2) or advertisement agencies (P5, P9, P10) buy data from the `WhatsApp` company in order to advertise. However, they are only capable to access the data through contacting the `WhatsApp` company.

Smartphones, Malware and the Internet: P10 named updates as a security threat because users are not aware of what they are downloading and do not know whether mobile devices send information to app providers. P6 and P11 mentioned malware, such as trojans. While P6 reported that malware could come from the governments, P11 believed the malware is downloaded when surfing on web pages with smartphones. Furthermore, P6 assumed that smartphones were easier to break into than into older mobile devices. Additionally, P6 stated that SMS is more secure than `WhatsApp` because it does not use the Internet. Similarly, P4 opined that "the Internet is evil. Everyone can listen to and read everything."

Fusion of Facebook and WhatsApp: P2 and P5 commented critically on the fusion of `WhatsApp` and Facebook in 2014:

> "I think it is defined in general terms and conditions that `WhatsApp` can use pictures etc. that are sent. Therefore, I believe that companies, which could profit from eavesdropping, [can read the messages]." (P2)

These participants assumed that Facebook is capable of accessing all messages from `WhatsApp`: For instance, P2 equated Facebook with Mark Elliot Zuckerberg: "The founder Zuckerberg [can read messages]. He also started up a company for that" (P2).

WhatsApp: For `WhatsApp` communication almost all of the participants mentioned the `WhatsApp` provider, hackers, the U.S. government, and the Secret Service as potential eavesdroppers. The participants had great doubts about the Internet technology as well as about `WhatsApp` as a company:

> "In principle, I assume that `WhatsApp` is able to decrypt messages. `WhatsApp` can read the messages [within their server]." (P1)

In addition to that, P5 believed that `WhatsApp` scans the messages for important information. P5 also remarked that `WhatsApp` reveals details about the communicating party, e.g., the online status or profile pictures.

2) Post-MME (2017): When comparing the adversaries, the new participants mentioned the same adversaries as the old participants. More interestingly, even though six participants (N3, N4, N7, N9-N11) noticed that `WhatsApp` introduced end-to-end encryption, they still considered `WhatsApp` to be capable of reading and accessing the messages.

Additionally, all participants except N7 reported that they did not change their behavior after they saw the notification for the end-to-end encryption. Furthermore, our participants stated that they did not understand the message. Even though six participants noticed the encryption of `WhatsApp` only three participants (N1, N5, N9) thought `WhatsApp` to be more secure than SMS. Our participants lack trust in encryption as well as lack knowledge of end-to-end encryption technology.

Finally, comparing pre-MME and post-MME it is notable that post-MME no participants thought that spying apps are included in the threat model for `WhatsApp`.

VI. Individuals' Cryptography Understanding

A. Pre-MME (2015)

The participants were asked whether they believed that SMS or `WhatsApp` messages were encrypted. Almost all participants (P1-P10) assumed that SMS messages are encrypted but again indicated that the mobile provider and the German government could decrypt their messages. Regarding

WhatsApp, participants had mixed feelings. P1, P3, P6, P9 and P10 stated that WhatsApp messages are encrypted, the rest stated that they are not encrypted. P1, P3, P6 and P9 assumed that WhatsApp has access to their messages independently of whether the messages are encrypted or not.

Four participants were able to name alternative secure messaging apps to WhatsApp, mentioning Threema (P7-P9, P11) and Telegram (P8). P1 stated that Tor could be used for secure email communication. However, eight participants (P1-P7, P10) stated that they have never used secure messaging apps. Although P4 used the app MyEnigma, which advertises end-to-end encryption, she did not state to use an encryption app. P9 and P11 tested Threema, but only P11 used the app for a longer time. Finally, most of the participants stated that they would be interested in a secure messaging app. However, they do not trust the app provider offering such services.

Encryption: Three participants (P3, P5, P10) needed an encryption tip, but could explain the idea of encryption after that very well. Participants defined encryption in the following way:

- A kind of a secret code (P5-P8, P10).
- Every message gets an encryption code, key or password.
- A change of data, so it is only possible to read the message with special information.
- A secret language in which all the letters are exchanged by numbers or symbols.
- Encryption is a "blockade" or "barrier", which is difficult to break like an anti-virus software or a firewall.
- Analogy with a ZIP-file: "Both parties need to have the protection app. [The text message] is passing through the Internet, but not to an extra company. Directly after sending the message, it has to be packed. Imagine it like ZIP-files. For the text messages you would also need a program, which is able to unpack" (P2).

Only one participant (P11) mentioned "end-to-end encryption" as well as illustrated asymmetrical encryption with private and public keys (student of business informatics).

Some participants believed that even if encryption is used it can be decrypted: "Actually nothing is secure, because software exist, which can decrypt everything" (P6). Therefore, the majority of the participants believed that the provider of a secure app handles the encryption and thus has access to the code. Consequently, the app provider is able to read all of their messages. To prevent eavesdropping by the secure app provider, P4 mentioned an idea of separate services for encryption and messaging. P6 expressed another idea: when developing an encryption application, the developer team splits the code in several parts. Then every member would be responsible for one part of the encryption. If the team combines the individual parts of the program they can break the code. Also P10 assumed that if the user wants to be secure, he needs to handle the encryption manually. It seems that users expect to put a certain amount of work in order to be/feel secure.

Authentication: In the next part, the participants were requested to think about authentication, specifically, how a message can be assigned to a unique person. Eight participants (P1-P5, and P7-P9) mentioned mobile phone numbers, but assumed that mobile providers could also imitate mobile phone numbers. Alternatively, a key and a password were mentioned by three participants (P1, P2, and P11). Two participants (P1, P10) commented that the only alternative was a personal meeting for comparing a password or exchanging a key:

> "There are classical methods in which each side owns a key. It would only be completely safe, if you meet physically and exchanged the keys. [...] Meeting physically means that you exchange the keys without using an electronic device. You can never be sure whether the post office or anybody else reads along. [...] I believe exchanging a key privately is the only solution providing full security" (P1).

Other participants (P3, P6, P10) mentioned that they would agree on a word and would append it to the message or simply ask the other person something, which only the person could know. P8 noticed that encryption is a contrast to authentication. In summary, most of the participants did not understand why authentication was needed:

> "But I have not sent [the message to a third party], but to my friend. I assume that my friend and I have accounts, Alex27 and Kati07, for example. Then I send a message to them. Why should Pia23 read along? The Internet does not know my password, only me and my friend do." (P2)

B. Post-MME (2017)

By comparing mental models of encryption between pre- and post-MME, we recognized that they did not change. Participants described encryption in a similar way, using symmetrical passwords or codes. The majority of participants said that WhatsApp is able to access the messages even if they deploy end-to-end encryption. Participants had troubles with understanding the term end-to-end encryption and feel like it is impossible that the WhatsApp company is not able to access the messages because they provide the encryption in the first place.

N10 mentioned that she uses WhatsApp but did not encrypt because it requires to scan the QR code of the other party. Obviously, even though she noticed the notification WhatsApp gives at the beginning of a conversation, she believed that the message is not automatically encrypted and that she needs to put an effort to encrypt her messages.

Our participants in 2017 still fail to understand how authentication can be achieved. Some participants believe they would notice if a certain message does not come from the sender (different style of writing or language). Further, all participants except one (N10) did not know about the security code verification. Even worse, the only participant who successfully scanned the code (N10) thought afterwards that WhatsApp does not offer encryption as default. As mentioned above, she

thought that only after scanning the code the messages are encrypted.

VII. Re-invited Participants Post-MME (2017)

Four participants (P1, P5, P6, P9) agreed to repeat our study giving us the chance to compare their mental models of 2015 and 2017. The re-joint participants all reported that they use WhatsApp regularly, compared to 2015 where some reported to use it rather often.

Messaging Architecture: In terms of mobile messaging architecture the participants stuck to their remaining mental models. P1 and P6 established a more detailed view on the architecture. P1 mentioned satellites and P6 added a new drawing for WhatsApp with a WhatsApp server as an intermediate station.

Threat Model: The threat model and the possible adversaries remained as strong as in 2015. While in 2015 P1 thought everybody could read WhatsApp messages, in 2017 he was sure that no one except the WhatsApp company can read WhatsApp messages any longer. He also did not mention the German or American government as eavesdroppers any more. In 2017 P5 additionally named the NSA as eavesdropper for every station (SMS and WhatsApp), but no longer the German government.

Although all participants recognized that WhatsApp introduced encryption, P1, P5 and P9 were sure that WhatsApp could access the keys. Further, P9 believed that "everyone who is a bit clever can decrypt". P6 was the only one from our re-invited participants, who believed that WhatsApp's encryption protects users' messages from eavesdroppers. She indicated that WhatsApp is probably using extra software for encryption preventing also the company from reading the messages. However, it should be considered that P6 is a Computer Science student and thus, we expect her to be more knowledgeable than lay people.

Encryption: In 2015 all four participants thought SMS were encrypted, while in 2017 only P9 still believed that SMS are encrypted (but using an universal encryption key). P5 was the only one in 2017 still thinking SMS is more secure than WhatsApp.

All participants were aware that WhatsApp introduced encryption. We assume that our study increased the awareness of the participants. Still, the introduction of end-to-end encryption in WhatsApp did not change the behavior of our four participants:

> "I still would not send a PIN over WhatsApp because it is personal information. If I *had* to, I would not feel differently before and after WhatsApp did it." (P9)

Authentication: Regarding two-party authentication in 2017 there are still misconceptions of WhatsApp's QR code: "I only used the QR code for WhatsApp web" (P6). P9 did not know about the QR code in WhatsApp because she has not found it in the app. However, she was aware that Threema uses the QR code for authentication.

VIII. Discussion and Implications

In previous work on mental models and security perception, researchers examined the current state at a given point in time. To the best of our knowledge, this is the first work studying behavior and understanding changes of users between two points in time separated by an important event: the introduction of end-to-end message encryption to the masses.

In this section we elaborate on the implications of our results for the security community. In addition to our takeaways based on our findings, we make concrete recommendations what messenger and protocol developers could do in the future.

One of the key differences between the encryption introduced by WhatsApp (and iMessage) is that they are extremely usable - to a point that it is almost impossible to make a mistake. This stands in strong contrast to *classical* approaches such as PGP, S/MIME and others, which suffer from many usability problems. Studies have shown that the majority of users made catastrophic mistakes in those cases.

However, our study results show that the introduction and usage of a highly usable security mechanism did not lead to higher perception or valuation of the security technology. In our view, this means that it was not just poor usability, which hindered the adoption of past end-to-end encryption schemes but also the unsurprising fact that users are overwhelmed by technology in general and consider themselves to be helpless and vulnerable against skilled attackers.

The fact that encryption was turned on without any user interaction and further, that the use does not require any additional interaction is what made this role-out a success. However, this success comes at a price. Unlike with PGP the WhatsApp and iMessage solutions rely on trusted first-parties, namely the message providers. The providers can change keys and mount attacks with little risk of being exposed since there is still limited awareness of the underlying end-to-end security mechanism.

A. Takeaways and Recommendations

Our most important takeaway is that end users do not trust encryption, and this has not been changed by the widespread adoption of usable encryption mechanisms. Even though the participants had an almost correct threat model, they felt vulnerable when using technology and were skeptical to any technical solutions to prevent the attacks. When asked about encryption, most stated they had heard of encryption but only few understood the implications even on a high level. Their consensus was that there is no technical solution stopping skilled attackers from eavesdropping. The fact that encryption increases the effort required to do so was not raised in any significant manner. More surprisingly, despite WhatsApp's end-to-end encryption information messages that pop up with every new communication partner and the high media attention surrounding iMessage and WhatsApp encryption, the majority of our participants were still not aware of it.

The success of WhatsApp's and iMessage's security solutions is based on the fact that end users do not have to understand what is happening. This seems like a very sensible

way forward: "Users should not have to care about security - it should just be there for them." The failure of the introduction and usage of end-to-end encryption to raise awareness or increase the desire for protection shows that continuing down the route of trying to educate users is likely to fail as it has in the past.

This leaves some big challenges for the community to tackle. One of the main reason why `WhatsApp` and `iMessage` have succeeded where previous solutions have failed is the fact that the products are under control of a single entity acting as a trusted entity. This makes the creation of a usable solution significantly easier. However, it also introduces the challenge of preventing the key manager from tampering with the keys. This confirms the demand for user-interaction-free solutions, such as key transparency and CONIKS [18, 25]. On the other hand, email encryption has to battle with the fact that dozens of email clients and hundred of providers are in use and without any concept for key management.

The results of our study suggest that attempts to provide security against key-change attacks by informing the user of changed keys might not succeed. For the time being we recommend that the community accepts the fact that there is a vendor lock-in since the effort to create a usable version of PGP has shown that the engineering effort to create a multi-client, multi-provider solution is a Herculean engineering task and in our view focusing on the less challenging problem. Thus, based on the results of our study, we propose that the open challenge is to fulfill the following criteria in a closed-ecosystem to start with:

- *Work on users' trust.* Our study suggests that end users underestimate the power of cryptography. As stated by P5, "Even Mrs. Merkel was hacked and all messages decrypted. So decrypting messages of an end user, the little customer, is probably not a big deal. [...] In the end, everything can be decrypted, interpreted or read."
- *Speak users language.* Despite this being a well known recommendation and part of Nielsen's usability principles this is still an issue in modern apps. Improving the notification in `WhatsApp` by using a more user-friendly language, e.g., understandable wording of end-to-end encryption. In our study, the participants stated that they ignored this message due to unfamiliar terms. For instance, N3 stated when asked about `WhatsApp`'s end-to-end encryption: "Oh that was what this annoying notification was about?"
- *Keep it simple for users.* As already adopted in various mobile messengers, users should not need to do anything for security. This was confirmed by our participants questioning the need to authenticate contacts manually: "It would be too inconvenient to scan QR codes of all contacts" (N10).
- *Fewer notification, which the user does not understand.* The party managing the keys should not be able to mount a man-in-the-middle attack by pushing new keys to a client. Only in this event a user should be notified. User struggle to understand the notification shown by `WhatsApp`.

IX. Conclusion and Future Work

The main goal of this paper was to capture and compare end-user mental models of message encryption. For this, we conducted an in-depth qualitative study consisting of two parts, each with 11 participants: pre- and post-MME. Although `WhatsApp` introduced end-to-end encryption more than nine months before our post-MME study, many participants still were not aware of it and there had been only minor changes in their mental models. Our results suggest that even though users were able to describe the threat model almost correctly, they do not believe that there is any technical possible to stop the attackers. More importantly, misconceptions about cryptography have lead to a lack of trust in encryption in general. The participants reported feeling vulnerable and simply assumed that they cannot protect themselves from attackers. Based on our findings, we make recommendations for how the community can address these challenges.

Our future work consists of studying cultural differences in detail and conducting a quantitative experiment testing users' understanding of different high-level encryption concepts.

X. Acknowledgment

We would like to thank Johanna Deuter for transcribing the original mental model drawings of the individual participants.

References

[1] Ruba Abu-Salma et al. "Obstacles to the Adoption of Secure Communication Tools". In: *Security and Privacy (SP), 2017 IEEE Symposium on (SP'17). IEEE Computer Society*. San Jose, CA: IEEE, 2017.

[2] Chris Alexander and Ian Goldberg. "Improved User Authentication in Off-The-Record Messaging". In: *Workshop on Privacy in the Electronic Society*. ACM, 2007, pp. 41–47.

[3] Farzaneh Asgharpour, Debin Liu, and L Jean Camp. "Mental models of security risks". In: *Financial Cryptography and Data Security*. Springer, 2007, pp. 367–377.

[4] Wei Bai et al. "An Inconvenient Trust: User Attitudes toward Security and Usability Tradeoffs for Key-Directory Encryption Systems". In: *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*. Denver, CO: USENIX Association, 2016, pp. 113–130. ISBN: 978-1-931971-31-7. URL: https://www.usenix.org/conference/soups2016/technical-sessions/presentation/bai.

[5] Nikita Borisov, Ian Goldberg, and Eric Brewer. "Off-the-Record Communication, or, Why Not To Use PGP". In: *Workshop on Privacy in the Electronic Society*. ACM. 2004, pp. 77–84.

[6] Glenn A Bowen. "Naturalistic inquiry and the saturation concept: a research note". In: *Qualitative research* 8.1 (2008), pp. 137–152.

[7] Cristian Bravo-Lillo et al. "Bridging the gap in computer security warnings: A mental model approach". In: *IEEE Security & Privacy* 9.2 (2011), pp. 18–26.

[8] Jon Callas et al. *OpenPGP message format*. Tech. rep. IETF Working Group, 2007.

[9] L Jean Camp. "Mental models of privacy and security". In: *Available at SSRN 922735* (2006).

[10] Kathy Charmaz. *Constructing grounded theory: A practical guide through qualitative analysis*. Sage, 2006.

[11] Sauvik Das et al. "Increasing security sensitivity with social proof: A large-scale experimental confirmation". In: *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. ACM. 2014, pp. 739–749.

[12] Sauvik Das et al. "The effect of social influence on security sensitivity". In: *Proc. SOUPS*. Vol. 14. 2014.

[13] Sergej Dechand et al. "An Empirical Study of Textual Key-Fingerprint Representations". In: *USENIX Security Symposium*. USENIX, 2016.

[14] Roger Dingledine, Nick Mathewson, and Paul Syverson. *Tor: The Second-Generation Onion Router*. Tech. rep. DTIC, 2004.

[15] Facebook. *Facebook Help Center*. 2014. URL: https://www.facebook.com/help/ (visited on 11/03/2014).

[16] Simson L Garfinkel and Robert C Miller. "Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express". In: *Symposium on Usable Privacy and Security*. ACM. 2005, pp. 13–24.

[17] Shirley Gaw, Edward W Felten, and Patricia Fernandez-Kelly. "Secrecy, flagging, and paranoia: adoption criteria in encrypted email". In: *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM. 2006, pp. 591–600.

[18] Google. *Safe email - Transparency Report. Email encryption in transit*. Google, June 4, 2014. URL: https://www.google.com/transparencyreport/saferemail (visited on 11/02/2014).

[19] Bryan Jenner et al. *A companion to qualitative research*. Sage, 2004.

[20] Natalie Jones et al. "Mental models: an interdisciplinary synthesis of theory and methods". In: *Ecology and Society* 16.1 (2011).

[21] Ruogu Kang et al. "My Data Just Goes Everywhere:" User Mental Models of the Internet and Implications for Privacy and Security". In: *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*. Ottawa: USENIX Association, 2015, pp. 39–52. ISBN: 978-1-931971-249. URL: https://www.usenix.org/conference/soups2015/proceedings/presentation/kang.

[22] Predrag Klasnja et al. "When i am on wi-fi, i am fearless: privacy concerns & practices in eeryday wi-fi use". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2009, pp. 1993–2002.

[23] Alexander De Luca et al. "Expert and Non-Expert Attitudes towards (Secure) Instant Messaging". In: *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*. Denver, CO: USENIX Association, 2016, pp. 147–157. ISBN: 978-1-931971-31-7. URL: https://www.usenix.org/conference/soups2016/technical-sessions/presentation/deluca.

[24] Susan E McGregor et al. "Investigating the Computer Security Practices and Needs of Journalists." In: *USENIX Security*. USENIX, 2015, pp. 399–414.

[25] Marcela S. Melara et al. *CONIKS: A Privacy-Preserving Consistent Key Service for Secure End-to-End Communication*. Cryptology ePrint Archive Report 2014/1004. 2014. URL: https://eprint.iacr.org/2014/1004.

[26] David L Morgan. "Focus groups". In: *Annual review of sociology* (1996), pp. 129–152.

[27] Alena Naiakshina et al. "Poster: Mental Models-User understanding of messaging and encryption". In: *Proceedings of European Symposium on Security and Privacy. http://www. ieee-security. org/TC/EuroSP2016/posters/number18. pdf*. 2016.

[28] Jakob Nielsen. *Mental Models*. http://www.nngroup.com/articles/mental-models. 3.3.2015. Oct. 2010.

[29] Erika Shehan Poole et al. "More than meets the eye: transforming the user experience of home network management". In: *Proceedings of the 7th ACM conference on Designing interactive systems*. ACM. 2008, pp. 455–464.

[30] Richard A Powell and Helen M Single. "Focus groups". In: *International journal for quality in health care* 8.5 (1996), pp. 499–504.

[31] Karen Renaud, Melanie Volkamer, and Arne Renkema-Padmos. "Why Doesn't Jane Protect Her Privacy?" In: *Privacy Enhancing Technologies*. Springer. 2014, pp. 244–262.

[32] Steve Sheng et al. "Why Johnny Still Can't Encrypt: Evaluating the Usability of Email Encryption Software". In: *Symposium On Usable Privacy and Security*. ACM. 2006.

[33] Ryan Stedman, Kayo Yoshida, and Ian Goldberg. "A User Study of Off-the-Record Messaging". In: *Symposium on Usable Privacy and Security*. ACM. 2008, pp. 95–104.

[34] Open Whisper Systems. *Advanced cryptographic ratcheting*. Nov. 2014. URL: https://whispersystems.org/blog/advanced-ratcheting/ (visited on 11/02/2014).

[35] Open Whisper Systems. *Open Whisper Systems partners with WhatsApp to provide end-to-end encryption*. Nov. 2014. URL: https://whispersystems.org/blog/whatsapp/ (visited on 12/23/2017).

[36] Open Whisper Systems. *Simplifying OTR deniability*. Nov. 2014. URL: https://whispersystems.org/blog/simplifying-otr-deniability (visited on 11/02/2014).

[37] Threema GmbH. *Threema.Seriously secure messaging*. https://threema.ch/de/. 01.04.2015. Apr. 2015.

[38] Nik Unger et al. "SoK: Secure Messaging". In: *Security and Privacy (SP), 2017 IEEE Symposium on (SP'17). IEEE Computer Society*. IEEE. San Jose, CA, 2015.

[39] Kami Vaniea, Emilee Rader, and Rick Wash. "Mental models of software updates". In: *International Communication Association, Seattle, WA, USA, in May* (2014).

[40] Matthias Wachs, Martin Schanzenbach, and Christian Grothoff. "A Censorship-Resistant, Privacy-Enhancing and Fully Decentralized Name System". In: *Cryptology and Network Security*. Springer, 2014, pp. 127–142.

[41] Rick Wash. "Folk models of home computer security". In: *Proceedings of the Sixth Symposium on Usable Privacy and Security*. ACM. 2010, p. 11.

[42] Rick Wash and Emilee Rader. "Influencing mental models of security: a research agenda". In: *Proceedings of the 2011 workshop on New security paradigms workshop*. ACM. 2011, pp. 57–66.

[43] WhatsApp. *Encryption Overview*. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf. Apr. 2016.

[44] Alma Whitten and J Doug Tygar. "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0". In: *USENIX Security Symposium*. USENIX, 1999.

APPENDIX A
FINAL GUIDELINE

1) **Scenario (5 minutes)**:
   You would like to communicate with a friend via your mobile phone. Therefore, you would like to send a text message via Short Messaging Service (SMS) or via WhatsApp to your friend. Please draw how you think a communication via SMS looks like. Which intermediate stations does the message pass through until it reaches your friend?
   - Does the communication for WhatsApp look similar or different?
     - If YES: Please prepare a second drawing or complete the first one.
2) Can other parties eavesdrop the message?
   - If YES: At which stations in the communication can other parties eavesdrop the message? Please draw it in your sketch. Which parties?
     Lately, there have been reports of "Eavesdropping of messages" in the media. Is there something that comes to your mind concerning this topic?
     - If NO: Have you heard about "Snowden" or the "NSA surveillance scandal?" If NO: proceed with (3, Tip)
3) You assumed that other parties might eavesdrop your message. Are there any countermeasures to prevent that? (1 minute) Tip: Have you heard of encryption?
4) How can somebody encrypt? Could you please explain how you think encryption works? Please draw it in your sketch. (5 minutes)
5) How can you assign a message uniquely to one person? (3 minutes)
   - If answer = "Personal password": Then you would have to agree with your communication partner on an individual password. This could be quite difficult. Can you imagine an alternative?
   - If answer= "Phone number": Would this be sufficient? (Tip: Or could the provider imitate the telephone number?)
6) Do you think SMS messages are encrypted? (1 minute)
   - If YES: Who could eavesdrop the messages? There was no password exchange.
7) Do you think WhatsApp messages are encrypted? (1 minute)
   - If YES: Who could eavesdrop the messages? There was no password exchange.
8) Have you heard of apps offering encryption? (1 minute)
   - If YES: Which can you name?
9) Have you used apps offering encryption? (1 minute)
   - If YES: Which ones?
   - If NO: Would you use apps offering encryption?
10) Are you currently using apps offering encryption? (2 minutes)
    - If YES: Which ones?
    - If NO but already used: Why are you not using the app(s) any longer?

APPENDIX B
DEMOGRAPHIC SURVEY QUESTIONS

1) Gender: *Female / Male*, Age: ... Years
2) How often do you send text messages via `SMS`/ `WhatsApp`? *Regularly - Often - Rarely - Never*
3) What is your highest completed level of education?
4) Recent professional status: Student, subject:...; Employed, profession:...; Unemployed

APPENDIX C
TECHNICAL SCORE

We calculated the technical skill score (0-12) of our participants as follows: (T1 + reverse(T2) + T3) - 3. A value of 0-3 was considered as a **low** technical score, 4-8 was considered a **medium** technical score, and 9-12 was considered a **high** technical score.

- **T1:** I have a good understanding of Computers and the Internet: *1: I disagree - 5: I agree*
- **T2:** I often ask other people for help when I am having problems with my computer. *1: I disagree - 5: I agree*
- **T3:** I am often asked for help when other people have problems with their computer. *1: I disagree - 5: I agree*

APPENDIX D
AGGREGATED MENTAL MODELS OF INDIVIDUAL INTERVIEWS

Stations mentioned by participants before end-to-end security from Figure 2a:

1) **Mobile Phone:** P3, P4, P5, P6, P8, P9, P10
2) **Provider:** P1, P2, P4, P5, P6, P7, P8, P10, P11
3) **Satellite:** P2, P8, P10
4) **Provider:** P1, P2, P7, P8, P10, P11
5) **Mobile Phone:** P3, P4, P5, P6, P8, P9, P10
6) **Provider:** P1, P8, P10, P11
7) **Internet:** P1, P2, P3, P4, P5, P6, P8, P10, P11
8) **Provider:** P1, P8, P11
9) **WhatsApp Server:** P1, P3, P5, P6, P7, P8, P9, P11

Stations mentioned by participants after end-to-end security from Figure 2b:

1) **Mobile Phone:** N2, N3, N4, N5, N6, N7, N10
2) **Provider:** N1, N2, N3, N4, N5, N6, N7, N8, N9, N11
3) **Mobile Phone:** N2, N3, N4, N5, N6, N10
4) **Internet:** N3, N4, N5, N6, N11
5) **Satellite:** N2, N3, N4
6) **WhatsApp Server:** N3, N4, N5, N6, N7, N8, N9, N11

APPENDIX E
DEMOGRAPHICS

The demographics of our participants can be found in Table III.

Participant	Sex	Age	Highest completed level of education	Current employment	Technical Score	How often do you send text messages via SMS	How often do you send text messages via WhatsApp	Used encryption apps	Encryption-tip
P1	Male	33	Master of Applied Biotechnology	Office Assistant	high	Regularly	Rarely	N	N
P2	Female	24	A-Levels	Biotechnical Assistent	medium	Rarely	Regularly	N	N
P3	Female	27	Trained Hairdresser	Caterer / Waitres	low	Rarely	Regularly	N	Y
P4	Female	47	Computer Science	Self-employed	high	Regularly	Rarely	Y	N
P5	Male	26	Advanced technical college entrance qualification	Student Business Administration	high	Rarely	Regularly	N	Y
P6	Female	26	Advanced technical college entrance qualification	Student Computer Science	medium	Rarely	Often	N	N
P7	Female	19	A-Levels	Student Media Informatics	medium	Rarely	Regularly	N	N
P8	Male	26	Diploma Financial Consultant	Public servant Fincancial Administation	high	Often	Regularly	Y	N
P9	Female	46	State examination	Research Assistant Medicin	medium	Regularly	Regularly	Y	N
P10	Female	25	Advanced qualification to enroll for a technical college+ Trained Educator	Unemployed	medium	Regularly	Never	N	Y
P11	Male	23	A-Levels	Student Business Informatics	high	Rarely	Regularly	Y	N
NP1	Male	48	Certificate of Secondary Education + Trained cook	Unemployed	high	Rarely	Regularly	N	N
NP2	Male	22	Advanced qualification to enroll for a technical college	Self-employed Salesperson Health Sector	low	Regularly	Regularly	N	N
NP3	Male	21	A-Levels	Student Aggricultural Studies	high	Regularly	Regularly	N	Y
NP4	Female	27	A-Levels	Student Medicin	medium	Regularly	Regularly	Y (WhatsApp)	N
NP5	Male	53	Trained forwarding merchant	Self-employed Trader	medium	Regularly	Regularly	N	N
NP6	Female	24	Bachelor Education	Student Master of Education	low	Rarely	Regularly	N	
NP7	Male	35	Master Logistics and E-Business	Part-time employed Business Admin.	medium	Regularly	Often	Y (WhatsApp)	Y
NP8	Female	53	Advanced qualification to enroll for a technical college	Self-Employed (Wellness Consulter)	medium	Rarely	Regularly	N	Y
NP9	Female	21	A-Levels	Paramedic	low	Regularly	Often	Y (WhatsApp)	Y
NP10	Female	21	A-Levels	Pharmaceutic, technical Assistant	high	Regularly	Regularly	Y (Line, WhatsApp)	N
NP11	Female	20	A-Levels	Employed in Business Administration	medium	Regularly	Regularly	Y (WhatsApp)	N
2017:P1 from 2015	Male	34	same as in 2015	Assistant of the CEO	high	Regularly	Regularly	Y (WhatsApp)	N
2017:P5 from 2015	Male	28	same as in 2015	same as in 2015	medium	Often	Regularly	N	N
2017:P6 from 2015	Female	27	same as in 2015	same as in 2015	medium	Often	Regularly	Y (Signal)	N
2017:P9 from 2015	Female	47	same as in 2015	same as in 2015	medium	Often	Regularly	Y	N

TABLE III: Demographics of 22 participants

4.2 Implications and Future Directions

The focus was on examining how users conceptualize secure messaging systems and their features to find and address users' misconceptions leading to lower security. The findings of this study reveal a significant gap between secure messaging systems' technical capabilities and users' perceptions of their effectiveness. Despite the adoption of end-to-end encryption by mainstream platforms like `WhatsApp`, user awareness of these advancements remains low. Most users did not even notice the introduction of end-to-end security. The study highlights a persistent mistrust in encryption technologies, rooted in misconceptions about their capabilities. Participants often overestimated attackers' abilities while underestimating encryption's protective power and immensely overestimating the capabilities of the attackers ("every encryption can be broken by secret services"). This skepticism led to a general sense of vulnerability ("Using technology made them feel vulnerable" [2]), with users believing that no technology could prevent skilled attackers from accessing their messages. Future studies could investigate whether user trust and understanding of encryption differ by cultural background or technological exposure, enabling developers to tailor communication strategies to specific audiences.

This mistrust underscores the limitations of usability-focused security implementations alone. While features like end-to-end encryption were seamlessly integrated into messaging platforms, they failed to shift user confidence in the technology. The findings suggest that technical transparency, intuitive communication, and targeted education are necessary to bridge the gap between user mental models and the actual security guarantees these systems offer. Integrating the study's recommendations into widely-used platforms would require a phased approach, starting with user testing of simplified terminology and incremental updates to the UI that prioritize intuitive encryption awareness. Emerging technologies like AI chatbots or decentralized systems could complicate user mental models since even for experts it is not always clear what remains locally and what might be forwarded to cloud-based models. Developers must preemptively address these shifts by ensuring transparency in how these features align with encryption standards. Building trust in encryption requires effective implementation and user-centered design that addresses misconceptions and fosters confidence in secure communication.

CHAPTER 5

Repurposing Wearables for Cryptographic Security

With smartphones increasingly serving as primary devices for sensitive operations such as email encryption, authentication, and secure communication in general, the protection of cryptographic keys becomes a central concern. As outlined in the analysis in Chapters 2 and 4, users can handle encryption if it is handled automatically, but the handling of security keys remains a significant problem for users since most average users do not entirely grasp what it means. Furthermore, traditional approaches, like storing keys directly on the device, are susceptible to theft, compromise, or software vulnerabilities.

This chapter, incorporating a previously published peer-reviewed journal article in its entirety [5], introduces a novel architecture that offloads key storage and cryptographic operations to external wireless hardware security tokens, such as smart cards or wearables. Leveraging the Near Field Communication (NFC) protocol enables a secure and user-friendly mechanism for key storage and management without requiring users to handle complex cryptographic operations manually. The proposed NFC-based token architecture can be integrated into existing tools like `OpenKeychain` by leveraging existing APIs for key management while ensuring backward compatibility with current cryptographic standards, especially `OpenPGP`. A quantitative lab study within a large enterprise company in Germany evaluates the practicality, usability, and acceptance of this approach.

This is one of the first implementations for offloading cryptographic key storage to wearable devices on Android, such as rings, which are used in common messaging applications or email. Rather than providing APIs for authentication only, this work offers a more high-level approach, which not only includes cryptographic primitives but also a variety of pre-defined user interface components for common end-user interactions. This approach separates key management from the mobile device, ensuring that sensitive data is stored securely on external tokens.

A lab study with 40 participants conducted under field conditions evaluates the implementation's usability and practicability, as well as the usability and security of the architecture in general. The empirical evaluation assesses the practical feasibility of NFC-based wearables for cryptographic operations and provides insights into the advantages and challenges of using such tokens in real-world scenarios.

5.1 Publication 4 | OpenKeychain: An Architecture for Cryptography with Smart Cards and NFC Rings on Android

Authors' Contributions

The work presented in this chapter is based on our publication *OpenKeychain: An Architecture for Cryptography with Smart Cards and NFC Rings on Android* published at the Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, Vol. 1, No. 3 (DOI 10.1145/3130964) [5]. The chapter text is, in large part, taken and adapted from this paper. The authors' contributions that are relevant to the contents of this chapter are as follows:

- **Sergej Dechand** I designed the study, conducted the sessions with participants and was responsible for processing and evaluating the study results. My contributions ensured the methodological rigor and the analysis of the findings presented in the paper.
- **Dominik Schürmann** served as the main author, primarily responsible for the software implementations and the integration of cryptographic features into OpenKeychain. Dominik also managed the interactions with study participants, ensuring the smooth execution of the software-related aspects of the study.
- **Lars Wolf** provided valuable feedback throughout all phases of the work and offered guidance on structuring the paper, contributing to its clarity and quality.

OpenKeychain: An Architecture for Cryptography with Smart Cards and NFC Rings on Android

DOMINIK SCHÜRMANN, TU Braunschweig, Germany
SERGEJ DECHAND, University of Bonn, Germany
LARS WOLF, TU Braunschweig, Germany

While many Android apps provide end-to-end encryption, the cryptographic keys are still stored on the device itself and can thus be stolen by exploiting vulnerabilities. External cryptographic hardware solves this issue, but is currently only used for two-factor authentication and not for communication encryption.

In this paper, we design, implement, and evaluate an architecture for NFC-based cryptography on Android. Our high-level API provides cryptographic operations without requiring knowledge of public-key cryptography. By developing OpenKeychain, we were able to roll out this architecture for more than 100,000 users. It provides encryption for emails, messaging, and a password manager. We provide a threat model, NFC performance measurements, and discuss their impact on our architecture design. As an alternative form factor to smart cards, we created the prototype of an NFC signet ring. To evaluate the UI components and form factors, a lab study with 40 participants at a large company has been conducted. We measured the time required by the participants to set up the system and reply to encrypted emails. These measurements and a subsequent interview indicate that our NFC-based solutions are more user friendly in comparison to traditional password-protected keys.

CCS Concepts: • **Security and privacy → Usability in security and privacy**; *Key management*; *Hardware-based security protocols*; • **Human-centered computing** → *Mobile devices*;

Additional Key Words and Phrases: NFC, near-field communication, smart card, ring

ACM Reference Format:
Dominik Schürmann, Sergej Dechand, and Lars Wolf. 2017. OpenKeychain: An Architecture for Cryptography with Smart Cards and NFC Rings on Android. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 99 (September 2017), 24 pages.
https://doi.org/10.1145/3130964

1 INTRODUCTION

Nowadays, it is widely acknowledged that smartphones hold the most sensitive information, such as emails, short messages, and photos. Besides private usage, the same devices are often used for accessing privileged company information due to Bring Your Own Device (BYOD) policies. Privacy-aware app developers take this into account and provide apps for secure messaging, encrypted cloud storage, and other use cases. Unfortunately, secret keys generated by these apps are unprotected and stored on the internal flash memory. Thus, an attacker can fully compromise end-to-end security by retrieving secret keys through privilege escalation exploits or direct physical access. While this is a well-known problem, the security of many apps and the mobile operating system is subpar [82]. Since the Stagefright-bug [85], Google started rolling out monthly Over-The-Air (OTA)

2474-9567/2017/9-ART99 $15
https://doi.org/10.1145/3130964

updates [53] documented in Nexus Security Bulletins [4]. Unfortunately, not all fixes are backported for devices of other Original Equipment Manufacturers (OEMs). Even if the system is up-to-date, widespread vulnerabilities in secure email and messaging apps have been found, potentially exposing secret keys to attackers [78, 82]. Even worse, app developers often lack knowledge of cryptography, which leads to insecure implementations [25, 29].

Traditional desktop encryption software provides protection against key theft by encrypting keys using high-entropy passwords. One of the main reasons this is not done on mobile devices is due to the difficulty of entering passwords on mobile on-screen keyboards. One often recommended password alternative are biometric fingerprints. These can be used since Android 6 with the Keystore API. While they are suitable for locking devices, they should not be used alone to protect secret keys. They do not provide enough entropy to reach the same strength as passwords or hardware-backed solutions [49, 56].

For desktop systems, external hardware in form of smart cards exist which can be used with USB card readers. They replace password-protected key files with external cards and corresponding PINs. In contrast to traditional key files, which are stored on the same device where the password is entered, the secret key of a smart card is stored physically separated from the device where the PIN is entered. For smartphones, On-The-Go USB cables or other external peripherals, such as hardcovers with card slots, exist. Because these are unwieldy and, thus, unsuitable for day-to-day usage, Near-Field Communication (NFC) has been proposed for smart card communication. A small selection of NFC tokens, which are capable of encryption and signature generation and not only authentication is available. Still, no high-level cryptographic NFC API was available.

1.1 Contributions

In this paper, we introduce and evaluate an architecture for NFC-based cryptography on Android via external security tokens, such as smart cards. First, we discuss the problem setting by looking into currently available NFC hardware and software libraries to derive a clear set of requirements. In the threat model we consider for this work, three main attack areas have been identified: NFC, security token, and smartphone. All attacks are evaluated in comparison to traditional password-protected secret keys. Based on our requirements and threat model, we carefully design our architecture and, thus, make the following contributions in this paper:

(1) Our main contribution is the design of a high-level API for Android, which not only includes cryptographic primitives but also a variety of pre-defined UI components for common end-user interactions. Previously available cryptographic NFC APIs have been designed for authentication only.

(2) For developers, we provide a set of API methods, which can be used without knowledge of public-key cryptography. Keys are handled transparently independent of their storage location (password-protected key file or via NFC). In addition to the card's PIN authentication, we provide a security layer via an Android app that handles the PIN input and allows only certain apps access to cryptographic methods, which have been explicitly granted by the user. This way, under the assumptions of Android's security model, only one trusted app handles the PIN input, while many semi-trusted apps can access cryptographic methods.

(3) As one of the main developers of *OpenKeychain*, an encryption app for Android, we were able to roll out this architecture for more than 100,000 devices. Its API is used by several other apps, such as *K-9 Mail*, the XMPP client *Conversations*, and the password manager *Password Store*.

(4) As an alternative form factor, we consider a NFC ring. Because no NFC rings with a cryptographic processor are available, we created our own prototype using an NXP Integrated Circuit (IC) and a 3D printer.

(5) To evaluate the end-user usability of our architecture in combination with password-protected keys, NFC cards, and NFC rings, we conducted a usability lab study with 40 participants outside of the university environment in cooperation with the IT security department of a large company. As a use case scenario we implemented end-to-end encrypted email communication. Finally, an interview has been conducted to evaluate user perception and form factor acceptance.

Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, Vol. 1, No. 3, Article 99. Publication date: September 2017.

2 RELATED WORK

The work presented in this paper touches the research areas of (a) usability of end-to-end encryption, (b) cryptographic API design, and (c) usability of security tokens, especially in conjunction with smartphones.

While our API supports a wide range of differing use cases, end-to-end email encryption is one prominent example, which we also included as an experiment in our study. The famous "Why Johnny Can't Encrypt" [81] publication discusses a case study on PGP 5.0 and concluded that end-to-end email encryption software was still not usable enough for most end-users. This led to several user studies over the few last years, e.g., by Garfinkel et al. [35, 36] and Fahl et al. [30]. They studied larger groups and proposed how email encryption or Facebook encryption can be made usable.

While there is a broad range of research on usable security for end-users, developer usability is often neglected. It has been shown that in a set of 12000 apps at least 88 % include at least one cryptographic error [25]. Similarly, developers often override safe defaults due to being unaware of the implications as shown in the evaluation of SSL apps by Fahl et al. [31]. To prevent these incidents from happening, high-level cryptographic APIs, such as NaCl [7], Sodium [20], and Keyczar [21], have been published in the last years. High-level 'crypto-box'-methods, based on a fixed set of algorithms, are provided that execute several steps at once, which are usually done individually when using low-level APIs such as Java Cryptography Extension (JCE) or Bouncy Castle [11]. Still, these libraries provide no automated way of handling password/PIN input and are not designed to fit into the developer ecosystem of mobile operating systems such as Android.

Research on the usability of security tokens for asymmetric cryptography in the context of encryption of emails or instant messages is limited. Most publications focus on authentication not encryption with security tokens, in particular smart cards. Sasse concludes that smart cards can offer usability benefits compared to password authentication [75]. Strouble et al. conducted a survey, answered by 300 participants, evaluating the usability of smart cards [79]. A notable result is that 67 % left their smart cards in the USB reader at least once, which increases the possibility of theft. Paul et al. conducted a field study with 24 participants over 10 weeks to evaluate the user behavior and perceptions in regards to smart cards [67]. They conclude that "The greatest perceived benefit was the use of an easy-to-remember PIN in replacement of complicated passwords. The greatest perceived drawback was the lack of smartcard-supported applications" [67]. Recently, Google compared their 'Security Keys' with other hardware tokens, passwords alone, and two factor authentications using smartphone apps [52] using the usability framework by Bonneau et al. [10]. They measured the raw performance and found out that users are twice as fast authenticating via 'Security Keys' in comparison to in-app One Time Passwords. Failure rates dropped from 3 % with OTP to 0 %. Taking into account the burden of physically carrying around authentication hardware, Mare et al. found that participants greatly differ in their preference regarding form factors [54]. To the best of the authors' knowledge, no studies are considering the usage of NFC security tokens, in our case NFC cards and rings, for end-to-end encryption instead of authentication on smartphones.

Available security tokens, which are qualified for our architecture, must include a cryptographic processor for asymmetric operations, such as RSA or ECC, and a NFC interface. For desktop systems with USB or card readers, tokens such as NitroKey [62], variants of YubiKey [83] (w/ OpenPGP support), and Java Card OpenPlatform (JCOP) smart cards with ICs by NXP are available. A number of smartphone accessories exist, e.g., back covers like the BlackBerry Smart Card Reader [72], modern variants like the Smart Card Reader by Precise Biometrics [69] (~100 EUR), and the Smart Fold Android Contact Smart Card Reader [46]. Due to the cost and bulkiness of external peripherals, NFC security tokens are considered. Only a limited number of the presented ones have an NFC interface, such as YubiKey NEO [83], the Fidesmo Card [32], and NXP developer cards with dual interface.

In addition to evaluating the traditional card form factor, we consider the usage of wearable NFC rings. A first example from 1998 is the Java Ring by Dallas Semiconductor [18] running an early version of the Java Card environment. During the following years, this form factor has been proposed for different communication

scenarios [16, 60, 71, 73]. A famous commercially available NFC ring has been designed by McLear and marketed via Kickstarter [58]. To the best of the authors' knowledge, no consumer-ready NFC ring supporting asymmetric cryptography over NFC together with an implementation was available until the first author started the work outlined in this paper.

3 PROBLEM SETTING

As discussed in the introduction, important secret keys should not be kept on internal flash storage as they can be stolen using widely available exploits. Instead, it is good practice to store those on external security tokens. In this section, we discuss related security and usability problems to derive a set of requirements for our architecture.

Cryptographic operations should be executed via a connection between smartphone and token, while the secret key itself should never be exposed to the smartphone. This connection is traditionally established via card readers, which require On-The-Go USB cables or external peripherals to work with modern smartphones. Also, smart cards are often left inside the reader [79], which poses a security risk. Communication over NFC has been proposed as a mobile alternative but no API is available for encryption/digital signature generation. Even though a small selection of security tokens with NFC and cryptographic processor are available, these are designed as credit cards or USB sticks. While other form factors have been proposed for simple NFC tags with read/write capability, these are not available with cryptographic processors. Furthermore, while the usage of security tokens has been evaluated for two-factor authentication, no studies exist in the context of end-to-end encryption. Thus, it is not known how users perceive the usage of NFC in the context of email encryption for example. It is also not known if other form factors could improve the acceptance of tokens or their usability.

When using NFC, security tokens must be held against the device's NFC antenna for differing durations depending on the cryptographic operation. Users can easily get frustrated if these are too long. Access control to operations should be done by entering a numeric PIN on the device that is easy to remember. To restrict brute force attempts, the token should deactivate after a number of failed attempts and a special Admin PIN must allow the user to re-activate the token. In available implementations for desktop operating systems with card readers, the users are not properly guided through the selection of an appropriate PIN and Admin PIN. For example, in GnuPG [26] and Enigmail [14] users are not forced to change the default PINs after key generation.

Not all apps should have access to the PIN to execute arbitrary cryptographic operations while the security token is held against the NFC antenna. Also, in traditional desktop implementations, it is not possible to restrict the execution of cryptographic operations to a specific set of client applications. Thus, client applications often handle PIN input directly, even though they may not be trusted fully. Furthermore, usage of secret keys is not restricted in any way, as long as a contact smart card is inserted into the reader. Secret keys for special purposes or higher levels of classification are not protected differently than other keys. Currently, no high-level API exists for smartphones that supports cryptographic operations, but also securely handles PIN/password caching and provides user interaction for common functionality, such as public key import. All available low-level APIs integrated in mobile operating systems such as Java's Cipher API [2] on Android, whose internals are based on Bouncy Castle/OpenSSL, as well as iOS Cryptographic Services [5] require a substantial effort from the developer. This naturally leads to vulnerabilities implemented over and over in many apps as well as to re-implementations of the same functionality in different contexts, such as password input as well as caching layers. Furthermore, they do not provide re-usable UI components. Even high-level APIs, such as NaCl [7], Sodium [20], and Keyczar [21], require a substantial amount of app-specific code to handle password input and caching, migrations from older key algorithms, and user to key mappings.

In addition, these are designed without special hardware in mind, i.e., no external security tokens are supported out of the box. Thus, keys stored on security tokens must be handled completely different and require more complexity than keys stored on the device, e.g., when using Android APIs [27].

3.1 Requirements

Based on the outlined problem setting, a set of architectural requirements is derived:

- R_1 Support for multiple form factors without external smartphone accessories
- R_2 Short durations of cryptographic NFC operations
- R_3 PIN input and caching solely handled by a trusted entity on the smartphone
- R_4 High-level versionable cryptographic API including UI components for common user interactions, secure defaults, and standardized packet formats
- R_5 Access control on the token-side by numeric PIN/Admin PIN and on the app-side by restriction of secret keys to specific clients
- R_6 Transparent handling of secret keys independent of their storage location

3.2 Threat Model

The main advantage of security tokens is that its key storage is physically separated from the mobile device. The hardware and firmware of the security token does not provide an API to retrieve secret keys, only cryptographic operations are exposed executed on the processor of the token. In the following, the discussed threat model is subdivided by the attacked entity, i.e., NFC, security token, and smartphone. While we evaluate all scenarios relevant for the whole architecture, an emphasis lies on mitigations provided by our own contributions.

3.2.1 NFC. First, we discuss attacks against the NFC connection itself that is established between the smartphone as an active initiator and the security token as a passive target.

Denial of Service Since radio jamming in general is difficult to prevent, NFC lacks sophisticated countermeasures. However, simply preventing communication is of low value to an attacker. In particular, signing and decrypting emails is not a time-critical activity and can, thus, tolerate short-term disruptions. More importantly, this attack does not put the security of the secret key at risk.

Relay Attack This attack is also called a wormhole or Mafia attack. In the field of NFC payment and authentication systems, a connection is established between the victim's smart card and an attacker's NFC reader. This connection is relayed over the Internet to a second device of the attacker to actually authenticate or pay at a different NFC reader physically far away [34]. This attack can potentially be executed unnoticed by holding the attacker's NFC reader against the victim's pocket containing the smart card. In our architecture, the security token is protected by a PIN. Thus, NFC relay attacks can only be executed if the PIN has been compromised beforehand.

Eavesdropping Kortvedt and Mjolsnes were able to eavesdrop on NFC in a range of up to 29 cm [51]. Brown et al. experimentally showed that eavesdropping capabilities largely depend on the amount of background noise [12]. Thus, if the attacker is very close to the victim and has the required equipment, it might be possible to extract the following information: In case of signature generation, a hash is transmitted and a signed hash is received. In case of decryption, an encrypted session key packet is transmitted and the decrypted session key is received. It is important to note that the plaintext that should be signed or the ciphertext that should be decrypted is never transmitted. Furthermore, the secret key never leaves the security token. Therefore, to decrypt an email with an eavesdropped session key, the encrypted email must also be intercepted at the corresponding email provider. Still, this is a valid attack scenario against targeted individuals. While our current prototype assumes channel security, for a future version we consider deploying the NFC-SEC standard [23, 24] that provides an Elliptic Curve Diffie Hellman key exchange with AES encryption. An application-level alternative for securing NFC has been proposed by Hölzl et al. using the Secure Remote Password (SRP-6a) protocol authenticated by a user-provided password [45].

Man-in-the-Middle (MitM) While eavesdropping might be possible in a certain range, MitM are extremely difficult because the attacker needs to block existing NFC. In detail, these attacks differ depending on the type of NFC connection: With an active-passive or passive-active connection, an attacker has to both block the originator's channel and to create an own RF field with perfect timing [42]. This is hard to achieve in practice and can possibly be detected by the user. In case of active-active connections, the interceptor has to completely block the communication between both partners without them noticing. Usually, NFC should abort if two RF fields exists, but is has been shown for the EMV protocol that some implementations do not follow the standard and it is possible to win the timing race [59]. Yet, this has not been shown for other protocols besides EMV. Conclusively, while Haselsteiner and Breitfuß [42] consider MitM attacks practically impossible, we at least consider them extremely difficult.

3.2.2 Security Token. In general, access to the security token is protected by a PIN with a length of 6 digits to protect against attackers in physical proximity. Thus, every cryptographic operation requires an authentication step. For the remaining attacks, we assume that the security token is protected against manipulations until it is received by the end user. Hence, the manufacturing process, warehousing, and shipping are considered to be secure, such that the initial key generation by the user is uncompromised. From this point on, though, the security token is vulnerable to theft and loss. As stated in the Introduction, there is no pre-deployed secret key on the security token, but the keys are generated by the user. In the following, we will mainly focus on attacks against the authentication step and hardware.

Brute Force PIN For memorability, we let the user chose the PIN, but prevent certain commonly chosen combinations, such as 123456. An attacker gaining access to the device can brute force up to 3 possible combinations of the PIN. After this, the security token is locked to prevent further brute forcing, and can only be unlocked entering the Admin PIN. In our architecture, the Admin PIN is not chosen by the user, instead it is securely-generated from random.

Physical attacks Due to theft or while the owner leaves the token unattended, an attacker can gain access to the security token. Physical attacks [80] aim to read, to modify or to erase data on the security token. Examples are provoking a power outage, examination with a probe station, chip re-wiring, as well as addition and cutting of a track. Given the physical access to the security token, these are generally difficult to defeat completely. Yet, they are typically expensive, destructive, and time consuming, especially since attacks are very target dependent. Additional protections against physical attacks, such as additional metal layers, bus scrambling, or on-board sensors can also be implemented on the hardware side. For these countermeasures, we rely on the security of the utilized NXP IC.

Side-Channel Attacks While being in close proximity to the owner, who currently uses the token with her smartphone, information about the cryptographic operation can be leaked by the token and smartphone. Timing attacks, for instance, exploit that the computing time of an operation differs with the used parameters, which in turn, can then be derived. As with physical attacks, we rely on the countermeasures provided by the IC. As discussed for an attacker who eavesdrop on NFC communication, also for side-channel attacks against the hardware, it could provide an additional advantage to monitor the corresponding email communication to correlate the decryption process with a particular message.

3.2.3 Smartphone. Recent vulnerabilities, such as the Stagefright bug [85], show the limited security on mobile devices. While local and remote software/firmware vulnerabilities are considered, we assume state-of-the-art cryptographic algorithms to be secure.

Physical Access An attacker, who gains physical access to a smartphone, while the owner leaves it unattended, is assumed to be able to download all data. The secret key, however, is never stored on the phone and, thus, not at risk. If no sophisticated attacks are performed, such as flashing a whole new operating

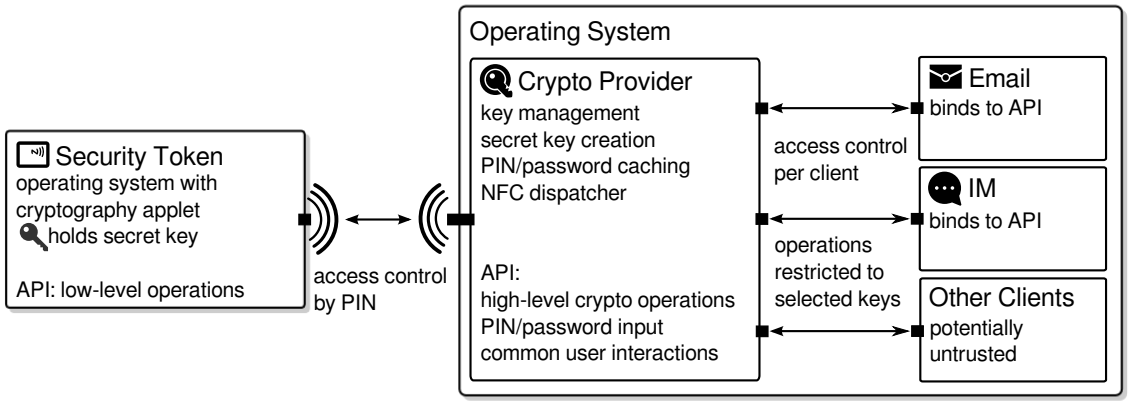


Fig. 1. Architecture Overview.

system, Android's system security prevents the installation of malware with root access. Still malicious apps with normal privileges could be installed. Due to the fact that PINs/passwords are only entered via a single trusted cryptography provider, malicious apps without root access cannot intercept these.

Vulnerabilities in Client Apps An attacker can try to exploit vulnerabilities in client apps that use our API. As described before, passwords and PINs cannot directly be retrieved as these are handled by the trusted cryptography provider only. Some vulnerabilities could potentially allow the attacker to trick the user into decrypting different content than originally opened by the user. In this case the password/PIN is properly entered by the user and the attack succeeds. To reduce the privacy impact in this scenario, a client app's API access is restricted to specific secret keys. Thus, only the keys selected for this client can be misused by an attacker.

Vulnerabilities in Android Critical vulnerabilities in Android can lead to exploits being used by an attacker to install malware with system access. In this case all installed apps are potentially insecure, even the PIN/password input and caching. Still, after detecting such a breach and removing the malware, future communications are secure, because the attacker was not able to retrieve the secret key.

UI Spoofing/Task Hijacking A malicious app could start a PIN/password dialog overlaying the original one and mimicking its design to intercept user secrets. This could be done by installing malware or a patched version of our cryptography provider. More sophisticated attacks building upon this scenario exploiting Android specific mechanisms are discussed by Cooley et al. [17] and Ren et al. [70]. As a future countermeasure we are plan to allow the user to set a personal image that will appear in all trustworthy UI components of our architecture. This has been implemented for example by Mailvelope [55].

GUI Side Channel Attacks As a special subcategory of attacks on Android, side channel attacks on the GUI of Android apps and hardware interrupts can potentially leak PINs/passwords to an attacker [15, 22]. As Diao et al. [22] remark, these side channels need to be closed on a system level by providing less runtime statistics to installed apps.

4 ARCHITECTURE

Considering the requirements and threat model, we propose a security architecture as depicted schematically in Figure 1. A security token, e.g., in credit card format, runs a smart card operating system together with an implementation of the cryptographic operations. The user's secret key is stored solely on this external token. It

receives power via induction while holding it against the device and communicates with the operating system over NFC. The device's NFC interface is standardized and, thus, works with security tokens of multiple form factors (R_1). The token's API is protected via PIN authentication and provides all required cryptographic operations such as key generation, signature creation, and decryption. A single trusted cryptography provider is installed as an app on the smartphone's operating system. It implements all required low-level cryptographic operations as well as communication over NFC, optimized for short durations (R_2) and usability. PIN/password input and caching is done solely by this trusted cryptography provider (R_3). In addition to PIN/password input, several other common user interactions are supported to prevent re-implementations of the same interactions in different clients and decrease implementation complexity for client developers. An API is exposed to client developers providing high-level versioned cryptographic methods, e.g., a method for encryption combined with signatures, in the standardized OpenPGP message format (R_4). Access to this API is granted per app by user choice (R_5). The cryptography provider provides a key generation wizard that includes a secure selection of PIN/Admin PIN. Furthermore, it provides a unified way to transparently use secret keys without exposing their storage location or asymmetric algorithms (R_6).

4.1 Prerequisites

Our architecture is based on several existing technologies that are discussed briefly as prerequisites.

OpenPGP To integrate with existing protocols, the OpenPGP standard [13] has been chosen. It provides standardized email [28] and instant messaging [76, 77] encryption. Thus, it supports most common use cases with extensions for standardized communication protocols. OpenPGP support for smart cards has been standardized for ISO-compatible card operating systems [68] and primarily three open source implementations exist [33, 61, 84].

NFC Typically operating at 13.56 MHz, NFC is a wireless transmission technology for short ranges allowing active-active and active-passive modes. In our case, an active-passive connection is established, where the smartphone serves as the initiator and the security token is the passive target. The ISO 14443-4 [47] standard is used as the physical/link layer protocol between initiator and target and ISO 7816-4 [48] defines the basic structure of commands and Application Protocol Data Units (APDUs).

Operating System Support NFC support in Apple's current iOS 9 only supports NFC store loyalty cards as part of the Apple Pay API [6] and Windows Phone has only limited support for smart cards [1]. In contrast, Android's NFC API allows to exchange APDUs. It supports a foreground dispatch mode (≥ Android 2.3.3) and reader mode (≥ Android 4.4) that allow an app to manage the NFC connection without interfering with other installed NFC apps [3].

4.2 API Design

The presented architecture has been fully implemented for the Android operating system as part of OpenKeychain [65], an app implementing the OpenPGP standard. Currently, OpenKeychain has over 100,000 installations on Google Play and is also available via alternative stores, such as F-Droid. Besides the API proposed in this paper, OpenKeychain also provides encryption/decryption as well as signature generation/verification functionality of messages and files within the app. Since October 2015, a version has been released that was audited externally [43].

The following design provides a high-level API that complies with all requirements and can be used by other installed apps in a convenient way: In agreement with the OpenIntents project [64], the API definition lives in the namespace "org.openintents.openpgp". In contrast to similar architectures like JCE, these can also be chosen at runtime via app settings, i.e., the cryptographic backend for an email app can be provided by multiple implementations of the same high-level API.

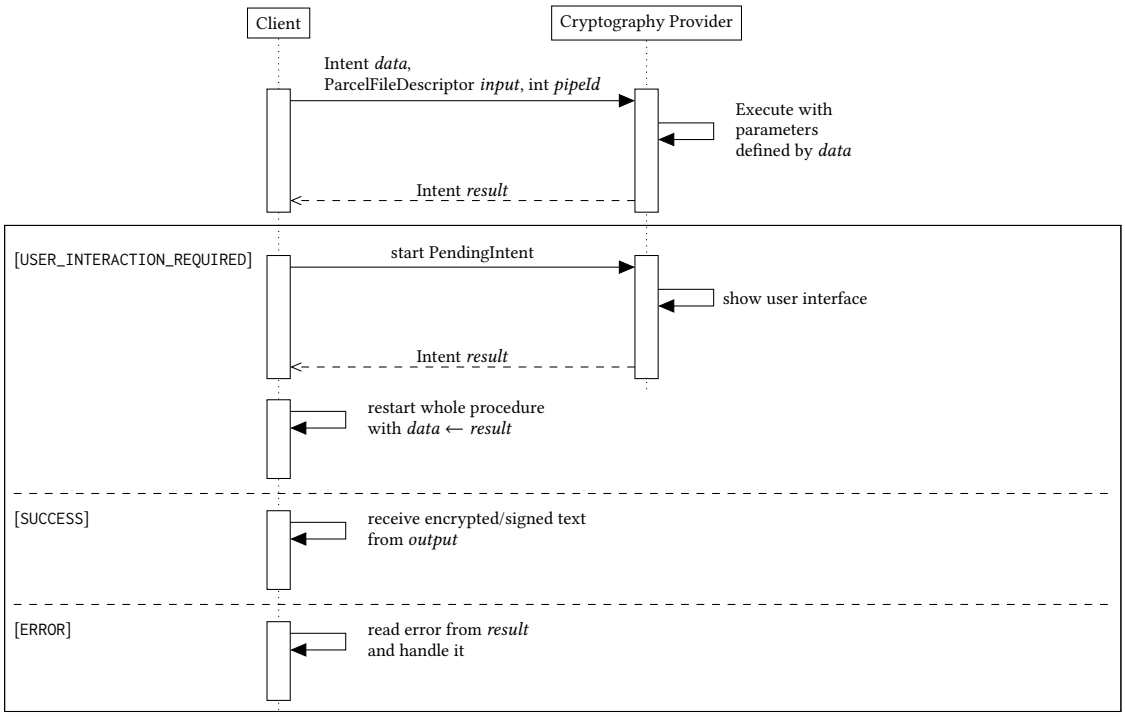
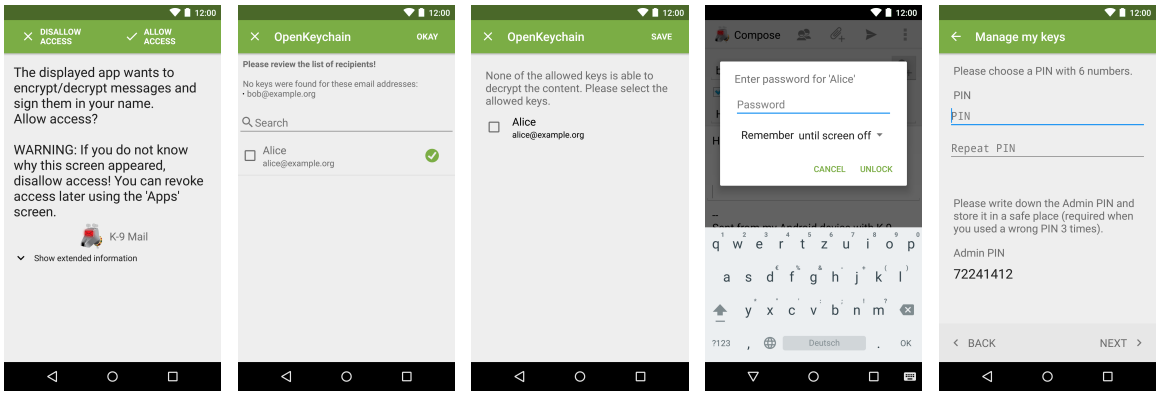


Fig. 2. Control flow of the proposed API.

Instead of providing an API via exported Android Activities, the API has been defined with the Android Interface Definition Language (AIDL). This allows for streaming of larger content via file descriptors. While this allows for performant access to shared memory between two apps, the immutable definition of the methods' type signatures makes it difficult to maintain backward compatibility with API definitions included in older client apps. Thus, instead of defining the method name and parameters directly as part of the type signature, these are defined using an Intent with a specific action (method name) and extras (method parameters), which are well-known to Android developers. Using Intents allows for easier backward compatibility and more flexible method definitions, which are not constrained to a specific parameter combination. To satisfy the requirement of backward compatibility, these are made versionable by including a version field together with a size calculated over the remaining fields at the first position when flattening the object for serialization.

Following Figure 2, after binding to the service, a client can execute a remote method. If a parameter is not specific enough or a required parameter is missing that can be provided by the user, the operation is canceled and the `USER_INTERACTION_REQUIRED` result code is returned together with a immutable PendingIntent. This PendingIntent can be started by the client at an appropriate time and is executed in the cryptography provider's process sandbox to handle interaction using appropriate UI components. One common use case is that a public key is missing for a given email address and must be downloaded. After user interactions, the operation is executed again with the parameters from the first execution combined with those retrieved from user input via the *result* Intent. The client app holds all parameters and decides by itself at which point in its control flow to



(a) Access control per app via user decision.

(b) Missing public key.

(c) Restriction of allowed keys per app.

(d) Password input for password-protected keys.

(e) PIN selection during key creation.

Fig. 3. A selection of UI components shown via PendingIntent for the `USER_INTERACTION_REQUIRED` state.

Table 1. API specification. No understanding of public-key cryptography is required.

Action	Req. Extras	Description
`SIGN_AND_ENCRYPT`	`USER_IDS`	Encrypt to email addresses (`USER_IDS`) and generate signature
`DECRYPT_VERIFY`	-	Decrypt and verify signature

start the PendingIntent for user interaction. On `SUCCESS`, the encrypted and signed text has been streamed into the file descriptor given by *pipeId*.

For an API it is desirable to be stateless, i.e., the cryptography provider should be implementable without caching parameters or method calls for connected clients. Provider-side caching is unnecessary because the *result* Intent (previously passed through the PendingIntent), which is required for a second execution, is returned to the client after every user interaction (`USER_INTERACTION_REQUIRED` case). While this is possible for most parameters, PIN and passwords should never be exposed to the client and, thus, cannot be returned via the *result* Intent. Therefore, a PIN/password cache has been implemented using key IDs as unique identifiers. No session management is required inside the cryptography provider due to this architectural design. Due to their high abstraction, the exposed API methods work independently from the storage location of the secret key.

4.3 API

We provide a simple API specification in Table 1. A combined signature generation with encryption can be executed by creating an Intent with `SIGN_AND_ENCRYPT` with at least one extra holding the email addresses of the recipients named `USER_IDS`. The plaintext is streamed into the file descriptor and read from the file descriptor previously opened by the client. On first execution, the operation will result in the `USER_INTERACTION_REQUIRED` state three times before finishing with the ciphertext in `SUCCESS`. As shown in Figure 3a, the cryptography provider asks the end-user to allow the requesting client access to the API. Afterward, the user is asked to select her own key (secret key) by another UI component of the provider. If no key is available for the requested

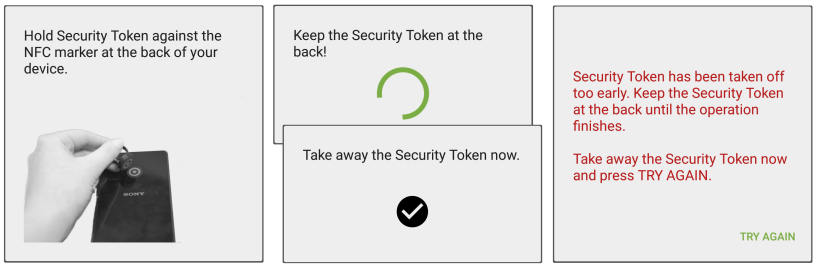


Fig. 4. Users are guided through the usage of security tokens over NFC.

email address a screen for key selection and retrieval is displayed, as shown in Figure 3b. Finally, based on the selection of the secret key, either a password (cf. Figure 3d) or a PIN is requested (similar to Figure 3d but with numeric keypad). This example shows that even without knowledge of public-key cryptography, a developer can effectively encrypt and sign data that can only be read by the recipient. Also, the secret key storage location is handled automatically and either a password input or NFC interaction with PIN input is returned. A second execution of the same Intent for different data will succeed earlier because access has now already been allowed, the public key is available and the PIN/password is already cached.

Regardless of whether the input is only encrypted, only signed, or a combined encryption with signature, an Intent with `DECRYPT_VERIFY` can be started to process the cryptographic input. No additional extras are required. Based on the PIN/password caching status of the specific secret key, the required screens are shown, e.g., Figure 3d. A special screen allows the user to restrict the secret keys that can be used by a particular client app (cf. Figure 3c). Besides the plaintext, two Parcelable objects are always returned indicating the decryption and signature result. These include the information if the given input was signed and/or encrypted.

For security tokens, an appropriate PIN and Admin PIN must be chosen (cf. Figure 3e). Here, we prevent the user from chosing one of the top 20 common PIN combinations following Berry's PIN number analysis [9], e.g., 123456, 000000, and similar ones. An attacker can try up to 3 different PINs until the security token locks itself and can only be unlocked by the Admin PIN. We provided a trade-off between usability and security by letting the user select her favorite PIN, but securely generate an Admin PIN that should be written down. Our design decisions are similar to current practices of PIN/PUK selection for SIM cards.

Advanced API calls, for example to generate backups or detached signatures, can be found in OpenKeychain's API documentation [65].

4.4 NFC UI Component

We conducted a pre-study with 12 participants using a preliminary design of our NFC UI component. In this pre-study, we mainly focused on qualitative feedback, whereas the main goal of this pre-study was to find flaws in the UI design and user experience. We provided a Sony Xperia Z3 smartphone and a white NFC smart card that has been pre-configured for the scenario and asked the participants to send an encrypted email. We observed them during this task, especially their interaction with the NFC UI components. Finally, we interviewed them about their experience.

We found out that it is important to give clear instructions to guide the users through the steps of using a security token. In previous versions, users took away the security token too early or were confused when the dialog closed automatically after a successful operation. Thus, we improved the process by dividing it into three steps shown in Figure 4: 1) clearly depict how to hold the token against the device, 2) display a progress indicator together with the instruction to keep the token at the back, and finally 3) display the instruction that the token can

(a) IC extracted from NXP J3D081.

(b) Circular coil as new NFC antenna.

(c) 3D printed ring prototype.

Fig. 5. Components of our NFC signet ring prototype (photos have the same proportions).

be taken away. Our implementation keeps polling for the established NFC connection after successful operations to detect when the token is taken away. When the token indicates with an error that a wrong PIN has been used for authentication, the previous cached one for this key will be cleared for the next try.

4.5 NFC Smart Cards and Signet Rings

In addition to evaluating our architecture, in particular our proposed cryptographic API for developers, we want to study the impact of a different form factor for end-users. Widely available form factors are that of smart cards or USB sticks. Available NFC rings solely include read/write NFC tags (cf. Section 2) No challenge-response or asymmetric cryptographic protocols are supported. Thus, they do not satisfy the requirements in this paper to support full asymmetric cryptography.

Due to the unavailability of such rings, we created our own prototypes. Because smart card ICs are only sold to smart card manufacturers, we bought blank *NXP J3D081* developer cards with dual interface support for NFC. The IC, depicted in Figure 5a, has been extracted using acetone [74]). A new induction coil functioning as the NFC antenna has been created using magnet wire to fit the form factor (cf. Figure 5b). The number of turns for an NFC antenna operating at 13.56 MHz depends on the IC configuration. Thus, to estimate the correct number of turns, we measured the frequency of the original antenna with a signal generator and oscilloscope. The original antenna's frequency has been measured as 875 kHz, thus, the inductance can be calculated as $L = \frac{4.57}{875\,\mathrm{kHz}} \approx 5.223\,\mu\mathrm{H}$ [19]. According to NXP [63], the number of turns for circular coils can be calculated with

$$L = \frac{24.6 \cdot N^2 \cdot D}{1 + 2.75 \cdot \frac{s}{D}}\,.$$

The magnet wire has been wrapped around a metal cylinder with a diameter of $D = 1.53\,\mathrm{cm}$ resulting in a circular coil with $s = 0.2\,\mathrm{cm}$. Choosing the number of turns with $N = 14$ results in an inductance close to the original one:

$$5.426\,\mu\mathrm{H} \approx \frac{24.6 \cdot 14^2 \cdot 1.53\,\mathrm{cm}}{1 + 2.75 \cdot \frac{0.2\,\mathrm{cm}}{1.53\,\mathrm{cm}}}$$

The resulting coil has been soldered with the IC and inserted into a 3D printed ring prototype as depicted in Figure 5c. It should be noted that the cylinder height on top of the ring can be reduced drastically by a more sophisticated production process.

5 EVALUATION

In this section we evaluate different aspects of our architecture by API comparison, NFC performance measurements, and a user study of our UI components.

5.1 Methodology

The evaluation of our architecture consists of several parts. (1) To understand its designated use for developers, a comparison with existing APIs, implemented as libraries and applications, has been done. (2) The raw performance of the cryptographic operations over NFC have been measured in a controlled environment. This helps to understand the technical constraints we needed to design UI components for. (3) As mentioned in Section 4.4, we conducted walkthroughs with 12 users from our university. The results indicated that users did not know when to keep NFC tokens at the back or when to take them away, which led to an improved UI for NFC operations. (4) Finally, we recruited 40 participants from a large company to test the full architecture from an end-user perspective including the UI components of our API and different NFC security tokens in a real-world environment. Furthermore, the usability and user perception of our NFC signet ring prototype as an alternative form factor has been evaluated.

5.2 API Comparison

Many cryptographic APIs are available that have been designed with different features and goals in mind. Thus, not every API suites every purpose and when designing communication systems the selection of an appropriate one largely depends on the following: If the design should be interoperable with existing standards, the API must support *standardized formats*. If it operates in a closed ecosystem, modern *high-level APIs* can be chosen where less programming errors can be made [7]. For higher security standards, especially in cooperate environments, the API should *support security tokens*. Furthermore, a category of APIs exist supporting functionality that go beyond cryptographic methods and require storage and GUI, such as *PIN/password cache*, *key management*, and *Graphical User Interfaces (GUI)*. These features provide complex functionality via API calls and reduce the burden on client developers who otherwise need to implement these on their own.

We selected prominent representatives for the categories of traditional low-level APIs, modern high-level APIs, and fully integrated systems and compared them in regards to the discussed functionality in Table 2. Only APIs for supporting encryption and signature generation for end-to-end security are considered, no authentication or transport security APIs are included. While more APIs exist, they typically fall into one of these categories and are thus evaluated similar to the selected ones. While modern libraries such as libsodium or Keyczar provide 'crypto-box'-methods with a fixed set of algorithms, GnuPG's selected algorithms depend on local configuration files and preferred algorithms defined in public key files. In our implementation, similar high-level operations with fixed algorithms exist that do not even require the knowledge of public-key cryptography due to additional UI components. While libraries such as Bouncy Castle must be integrated with additional libraries, such as OpenSC, to support security tokens, in our architecture security token support is an integral part. Modern libraries often lack a standard format and a corresponding key/algorithm migration path.

The features that require either support by the operating system or depend on specific GUI toolkits are typically not found in libraries, but in apps/integrated systems. An exception is Keyczar that provides basic command line tools for key management. One of the main goals of our system is to provide common user interactions via UI components. GnuPG specifies a 'UI Server Protocol' [40] that has similar goals and is implemented for Kleopatra and GPA. The library GpgME makes accessing this API easier [40]. In comparison to our implementation, the specification is not stateless and the implementation in Kleopatra does not provide dialogs for security tokens. Its Inter-Process Communication (IPC) is based on libassuan for platform-independent sockets. GNOME's Seahorse,

Table 2. Feature comparison of cryptographic APIs for end-to-end security. Libraries that only offer authentication or transport security are not considered here.

			High-Level API w/ Secure Defaults	Supports Security Tokens	Standardized Formats	Cross-Platform	PIN/Password Cache	Key Management	GUI
Low-Level APIs	libcrypto	[66]	○	○	●	●	○	○	○
	Bouncy Castle	[11]	○	○	●	●	○	○	○
	OpenSC	[37]	○	●	●	◐	○	○	○
High-Level APIs	NaCl/libsodium	[8]	●	○	○	●	○	○	○
	Keyczar	[21]	●	○	○	●	○	◐	○
Fully Integrated Systems	GnuPG	[40]	○	●	●	◐	●	●	○
	GNU Privacy Assistant (GPA)[a]	[41]	○	●	●	◐	●	●	●
	Kleopatra[a]	[50]	○	●	●	◐	●	●	●
	GNOME Keyring[a]	[38]	○	●	●	○	●	●	●
	Our work		●	●	●	○	●	●	●

[a] uses GnuPG as its backend

which is the frontend to GNOME Keyring, provides similar capabilities using a dbus service. Still, it misses functionality such as searching for and importing keys when choosing recipients [39].

While our API is not used as widely as the listed competitors, several client applications have already been released with an active user base. Its usage spans easy use cases, such as password managers (*Password Store*), as well as sophisticated ones, such as instant messaging (*Conversations*) and email clients (*K-9 Mail*) [65]. Most client apps are developed by third-parties and available on Google Play.

5.3 NFC Performance

We measured the performance of executing cryptographic operations over NFC using a Sony Xperia Z3 and the NXP J3D081 smart card running Yubico's OpenPGP app version 1.0.10. The average durations can be found in Table 3. Besides generating secret keys, which is only done once for new users, our measurements show average durations below 1 s for day-to-day operations. Only the asymmetric operations are executed on the smart card: For signatures, the hash of the input is generated on the smartphone, only the RSA signature is calculated on the smart card. For decryption, AES is executed on the smartphone, only the session key is decrypted on the smart card. 2048 bit RSA keys have been transferred and generated. ECC has not been evaluated, because no OpenPGP applet with ECC support was available for JCOP operating system during this work.

Table 3. Mean durations (w/ standard deviation) of cryptographic operations over NFC (10 experiments per operation).

Operation	Duration	σ
Signature calculation	787.9 ms	3.18
Decrypt session key	830.9 ms	55.86

Operation	Duration	σ
Transfer existing secret key	711.9 ms	32.66
Generate secret key on-token[a]	9476.2 ms	2297.71

[a] Roughly, only every third key generation succeeded

Generating keys on the smart card turned out to be unreliable. Only roughly every third generation succeeded, while all other operations canceled by losing the connection. Even when having the card lying on a flat surface with the smartphone on top, we were never able to generate three keys in a row that makes this method unsuitable in practice. The same issues have been encountered with different smartphone-token combinations. By building a self-contained implementation, we ruled out issues in our architecture design. Instead, we suspect that the induction does not provide a perfectly stable energy supply, which is required by the key generation process. Because on-token key generation was too unreliable, in our current version keys are generated on-smartphone. We will investigate this issue further and will fix this in an upcoming version, possibly using ECC providing faster key generation methods.

5.4 User Study

To evaluate the usability of NFC ring and card form factors in comparison to password-protected keys, we conducted an end-user study with 40 participants at a large company outside of the university environment. The main goal of our study is to test the usability of the NFC-based approaches in comparison with state-of-the-art password protection of secret key material. In this section, we present our study design and discuss the corresponding results.

5.4.1 Participant Recruitment. Our 40 participants were recruited in the IT department of a large company based in Germany. Taking part in the study was considered as working time, i.e., the participants were paid their normal hourly wages. Due to restrictions in their employment contract, we were not allowed to pay additional money on top. Our university does not have an Institutional Review Board (IRB), but the study conformed to the strict data protection law of Germany and informed consent was gathered from all participants.

5.4.2 Study Design. Our conducted study consists of two parts: (1) a lab experiment observing objective measurements such as *setup time*, *decryption time* and (2) a follow-up user survey for analyzing end-user perception.

The experiment consists of different tasks to be performed with different approaches.

5.4.3 Variables, Conditions and Participant Assignment. Our independent variable among all tasks was the chosen authentication type with following conditions *password*, *NFC card*, and *NFC ring*. The in the evaluation relevant dependent variables (objective measurements) were (1) duration to measure the efficiency of each task and (2) user perception based on a follow-up survey. The effectiveness was not considered as a separate dependent variable in our evaluation since all users were able to perform the tasks. For the condition assignment, we opted for a within-group design where all study participants had to perform tasks from all approaches: *password*, *NFC card*, and *NFC ring*. We did not test against other methods commonly used for authentication, such as biometric fingerprints or pattern-based techniques. Generally, these provide a much lower security level than passwords satisfying modern length requirements, let alone security tokens [56] and are thus not suitable for end-to-end encryption. Our design allows us to gather user perception at the end of the study where users give feedback and

(a) Ring with smartphone. (b) Card with smartphone.

Fig. 6. Handling of NFC rings and cards together with smartphones. A sticker has been used to indicate the best NFC spot on smartphone and ring.

ratings for all approaches. To mitigate learning and fatigue effects in our within-group study design, the order in which participants were asked to perform the approaches was randomized.

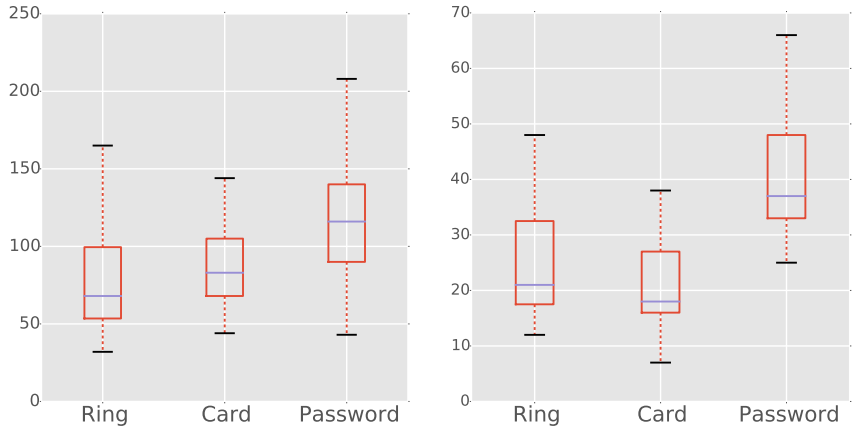
5.4.4 Tasks. The performed tasks are:

Task 1 When users start the application for the first time, they have to follow a wizard to create a key pair for usage with the app. They are guided through the process, which consists of entering a name and an email address, select a password/PIN and depending on the approach to hold an NFC security token against the smartphone. The actual key generation is indicated by a progress bar, while the user has to wait until it finishes.

Task 2 In the second task, the participants are asked to receive and read an encrypted email. Depending on the approach, users might be asked to enter a PIN or password, or hold an NFC device against their smartphone. To avoid bias due to variable password/PIN complexities, during this step we provide a pre-defined password/PIN.

Task 3 At last, the participants are asked to reply with a secure email by writing an appropriate response text and sending it.

To begin with the study, we provided a detailed explanation of the concept and the procedure to participants. We gave them a Sony Xperia Z3 smartphone and optionally depending on the approach, either the NFC ring or NFC card to let them get accustomed to the hardware themselves. As depicted in Figure 6, the usage patterns between the NFC ring and card differ due to their physical size. Before conducting the study, a sticker has been attached to the smartphone and ring indicating the best spot and the affiliation between these objects. Right after this, the participants continued with the key creation wizard of the first tested approach followed by the other remaining tasks. After completing the first tested approach, the other approaches follow. At the end of the study, we interviewed the participants for their ratings with regards to the single approaches and additional feedback. Finally, they were asked to participate in an anonymized questionnaire to collect demographic statistics.



(a) Setup time.

(b) Decryption time.

Fig. 7. Time measurements (in seconds, no outliers, lower is better).

5.4.5 Statistical Testing. For the statistical hypothesis testing, we opted for the common significance level of $\alpha = 0.05$. To account for multiple testing, all our study p-values are reported in the Holm-Bonferroni corrected version [44]. All time intervals and user-ratings are tested with the *nonparametric* (applicable to unknown statistical distributions) *Mann–Whitney U* test (two-tailed, Holm-Bonferroni corrected). We opted for this nonparametric statistical test due to the ordinal nature of our data and to avoid any statistical distribution assumptions. All our effect sizes are reported by mean comparisons and the usage of the *common language effect size* method [57], i.e., the meaning of the effect size is explained in plain English.

5.4.6 Objective Measurements. We measured following objective measurements in our experiment:

Setup time We measure the entire time of the setup process in Task 1. This includes input of name and email address, password/PIN selection, key generation on-smartphone, and optionally transfer to security token.

Decryption time Here, we measure only the time where the users have to perform an action related to the cryptographic operation of Task 2, i.e., password input and on-smartphone asymmetric decryption or PIN input and on-token operation (requiring holding the token against the smartphone's back side).

Sign/Encryption time Again, we measure the time where the users have to perform an action in Task 3. Due to PIN/password caching, no input is required for signing. Thus, we only measured the time required for executing the NFC operations by holding the security tokens against the smartphone's back side.

Figure 7a shows a box plot with a time distribution overview for the setup process. Our main hypothesis is that passwords are less efficient (especially on smartphones) in comparison to NFC-based approaches which is also a common belief in the usable security community. As can be clearly seen, the password-based approach tends to require extra time: a median of 114.5 seconds indicated by the blue line in box ($p < 0.0001$ in comparison to the NFC-based approaches supporting our main hypothesis). NFC-based approaches, on the other hand, have shown a better performance during the wizard process: a median of 83.5 and 68.5 seconds ($p = 0.083$ indicating that based on our sample size, we could not observe a significant difference between those). For instance, only 14 people were able to type in a valid password on the first try. By comparison, 22 people were able to position the ring correctly and choose a valid PIN on the first try.

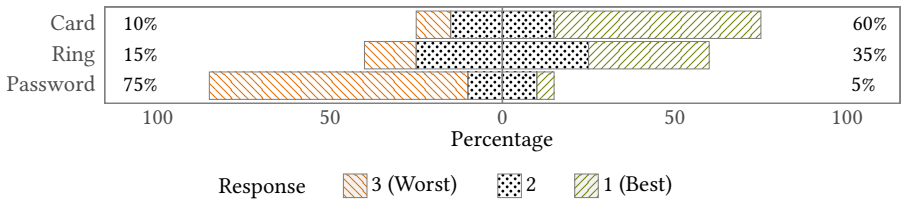


Fig. 8. Aggregated user perception showing the ranking choices in the interview.

Figure 7b shows a box plot with a time distribution overview for decrypting an email. This process consists of the following steps: when a secured email is opened, the cryptographic API immediately starts with the decryption process. To be able to decrypt, depending on the approach, either a password input or the positioning of the NFC device in combination with a PIN input is required. Note, that during this task we provided the participants a pre-defined password/PIN to avoid bias due to variable complexities. As can be clearly seen, the password-based protection not only performs worst in security and setup time, but also requires additional time for reading encrypted emails: median of 37.0 seconds (18.5 and 22.0 for NFC-based solutions) as indicated by the blue lines in the boxes ($p < 0.001$ supporting our main hypothesis). In the follow-up writing of a secured email where the password is cached, NFC-based approaches require additional time for positioning of the NFC devices: mean of 9.6 seconds (11 seconds for the ring and 8.3 for the card).

5.4.7 User Perception. To measure user perception, after completing all tasks in the experiment, we asked the users to answer questions with regards to their tested approaches in an interview. Here, we distinguish between *quantitative* results aggregated from users' ratings and *qualitative* open-end questions asking the users for their feedback and justifications. Our hypothesis is based on a common belief that passwords are troublesome on mobile devices. The full interview can be seen in the Appendix A. Furthermore, we asked them to fill out a short questionnaire form with additional details about their demographics, education, computer literacy, and previous usage of security apps. The full questionnaire can be seen in the Appendix B.

Quantitative. As our major quantitative question in the interview, we asked our participants to rank their tested approaches: 1 as best and 3 as worst. As depicted in Figure 8, the majority of our participants ranked NFC-based approaches to be superior to the password-based approach. The color coding shows the ranking level and the percentage numbers on the y-axes summarize the percentage of participants who ranked an approach as best (1) and as worst (3) respectively. A pairwise comparison between the approaches shows a high statistical significance between the token-based approaches and password-based protection ($p < 0.0001$ in both cases). However, we could not observe significant results between the NFC card and the ring ($p = 0.073$). The NFC card approach achieved a slightly higher mean of 1.5 in comparison the NFC ring that achieved a mean of 1.8. By comparison, password-based key storage achieved a mean rating of 2.7 where only 5 % of the users consider this approach to be best.

Qualitative. During the interview we asked the participants to describe the advantages and disadvantages of each approach (cf. Appendix A). While the participants accepted passwords as a well-known approach, most participants agreed that "imagining good passwords is incredibly difficult" and "good passwords are difficult to enter on smartphone keyboards". Whereas, cards and rings have the advantage of "requiring only a short PIN instead of a complicated password". In general, participants were in favor of cards, due to their common form factor, which allows to "store them easily in the wallet". Some mentioned that it is annoying to constantly take it out of the wallet, thus they prefer wearing the card attached to their belt. Many participants remarked that the card was more easily be placed below the smartphone and then worked perfectly with NFC and did not

require precise positioning like the ring. Participants who favored the ring found the idea great and described it as a "cool gadget". Some noted that "rings are more secure than cards because they are more difficult to steal than wallets" and their "security purpose is not immediately obvious to an outsider". Interestingly, participants assessed it differently if cards or rings are more easily lost. Some argued that "rings can easily be forgotten on a bedside cabinet while not worn at night", while they argued that they "never forget the wallet in the morning before work". Others said that "cards are easily misplaced as they are not constantly worn on the body".

Demographics. A total of 40 users from the same company participated in our user study. 33 participants were male, 6 participants were female and one participant opted not to disclose the gender. The mean and median age of the participants was 34. In the quantitative analysis, we could not find any statistically significant differences between the genders, although women tend to prefer wearable NFC devices over cards. During the interview we also noted different reasons for liking/disliking particular approaches. For instance, 9 out of 33 men preferred cards instead of rings simply because they usually do not wear rings at all and are not accustomed to it. Some of them proposed the usage of watches or wristbands as an alternative form factor. A woman argued that, because dresses are often worn without belts, she "prefer[s] to wear cards attached to a necklace". Naturally, she and two other woman preferred the ring as it can be worn as a fashion accessory and has a smaller size.

14 participants did not have a university degree (3 of them were students planning to complete a degree). 8 participants completed a Bachelor's degree or similar, 17 a Master's degree or similar and 1 a doctorate's degree. We could not observe statistical significant differences between the degrees.

Our question set (cf. Appendix A) indicates a high level of technical background. On a scale from 1 (novice user) to 20 (experienced) the participants achieved a mean score of 17.1 wheres the participant with least knowledge achieved a score of 13.

Limitations. First-off, our study does not test whether end users will actually switch to NFC-based encryption or even start encrypting their emails during daily work. As with any lab study, more issues might arise in the field and thus an actual field study is an important future work. Our 40 participants were recruited in the IT department in a large company based in Germany, which is not the representative of the general population in Germany. As mentioned before, our questionnaire (cf. Appendix A) indicates a high level of technical background.

6 CONCLUSION

We proposed and implemented an architecture for NFC-based cryptography on Android devices. Our architecture includes a high-level cryptographic API especially designed for developers accustomed to Android's IPC mechanisms. It allows for cryptographic operations without knowledge of public-key cryptography, works transparently with password-protected key files as well as NFC security tokens, and provides carefully designed user interactions. In addition to traditional NFC smart cards, our NFC signet ring prototype represents an alternative form factor for end-users. Performance measurements show that cryptographic operations over NFC can be executed fast enough to be usable for day-to-day use. In our lab study with 40 participants, 95 % chose one of the NFC solutions as the best approach. Conclusively, we have shown the advantages of our architecture for NFC-based cryptography.

A INTERVIEW

The original questions were asked in German.

- Which operating system are you using on your own smartphone?
- For every approach {password, smart card, signet ring} (in the order the approaches have been tested by the participant):
  - What did you think was good?

  - What did you think was bad?
- Which approach was the best in your opinion? Please create an order by assigning the numbers 1, 2, 3.

Approach	Order
Password	[ ]
Smart Card	[ ]
Signet Ring	[ ]

- Would you consider replacing passwords by a signet ring?
- Would you consider replacing smart cards by a signet ring?

B QUESTIONNAIRE

The original questionnaire was written in German. Furthermore, it included in addition gender, age, and qualification questions (Question 1–3), which are not displayed in this appendix.

Question 4

Please rate how much you agree (or disagree) with the statements below.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I have a very good understanding of computers and the Internet.	☐	☐	☐	☐	☐
I often ask others for help when I have computer problems.	☐	☐	☐	☐	☐
Others often ask me for help when they have computer problems.	☐	☐	☐	☐	☐
I have a very good understanding of computer security.	☐	☐	☐	☐	☐

Question 5

Are you already using apps for encryption or secure communication on your *computer* in your *private* life?

☐ Yes, in particular: .....
☐ No

Question 6

Are you already using apps for encryption or secure communication on your *smartphone* in your *private* life?

☐ Yes, in particular: .....
☐ No

Question 7

Are you already using apps for encryption or secure communication on your *smartphone* in your *job*?

☐ Yes, in particular: .....
☐ No

Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, Vol. 1, No. 3, Article 99. Publication date: September 2017.

Question 8

Have you already used NFC before this study?

□ Yes, in particular for: .....
□ No

ACKNOWLEDGMENTS

The authors would like to thank Vincent Breitmoser, Joey Castillo, and Signe Rüsch for contributing to the NFC-related code, Johannes van Balen and Yannic Schröder for their help creating the signet ring prototype, as well as Linda Fliss and Stephan Merker for their assistance and coordination during the lab study. Furthermore, we would like to thank all participants of the study.

REFERENCES

[1] 'Alex288'. 2014. NFC Smart Card Reader PC/SC Library: Project Description. (2014). Retrieved July 2017 from https://nfcsmartcardreader.codeplex.com
[2] Android Documentation. 2017. Cipher class. (2017). Retrieved July 2017 from http://developer.android.com/reference/javax/crypto/Cipher.html
[3] Android Documentation. 2017. Near Field Communication. (2017). Retrieved July 2017 from http://developer.android.com/guide/topics/connectivity/nfc/index.html
[4] Android Open Source Project. 2017. Nexus Security Bulletins. (2017). Retrieved July 2017 from https://source.android.com/security/bulletin
[5] Apple Inc. 2017. About Cryptographic Services. (2017). Retrieved July 2017 from https://developer.apple.com/library/ios/documentation/Security/Conceptual/cryptoservices/Introduction/Introduction.html
[6] Apple Inc. 2017. PassKit Package Format Reference. (2017). Retrieved July 2017 from https://developer.apple.com/library/ios/documentation/UserExperience/Reference/PassKit_Bundle/Chapters/TopLevel.html
[7] Daniel J. Bernstein, Tanja Lange, and Peter Schwabe. 2012. *Progress in Cryptology – LATINCRYPT 2012: 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings.* Springer Berlin Heidelberg, Berlin, Heidelberg, Chapter The Security Impact of a New Cryptographic Library, 159–176. https://doi.org/10.1007/978-3-642-33481-8_9
[8] Daniel J. Bernstein, Tanja Lange, and Peter Schwabe. 2012. *The Security Impact of a New Cryptographic Library.* Springer Berlin Heidelberg, Berlin, Heidelberg, 159–176. https://doi.org/10.1007/978-3-642-33481-8_9
[9] Nick Berry. 2012. PIN analysis. (Sept. 2012). Retrieved July 2017 from http://datagenetics.com/blog/september32012/index.html
[10] J. Bonneau, C. Herley, P. C. v. Oorschot, and F. Stajano. 2012. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *IEEE Symposium on Security and Privacy*. 553–567. https://doi.org/10.1109/SP.2012.44
[11] Bouncy Castle Inc. 2017. The Legion of the Bouncy Castle. (2017). Retrieved July 2017 from http://www.bouncycastle.org
[12] T. W. C. Brown, T. Diakos, and J. A. Briffa. 2013. Evaluating the eavesdropping range of varying magnetic field strengths in NFC standards. In *7th European Conference on Antennas and Propagation (EuCAP)*. 3525–3528.
[13] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. 2007. OpenPGP Message Format. RFC 4880 (Proposed Standard). (Nov. 2007).
[14] 'Cane', 'Topo', and 'Orso'. 2017. Privacy-Handbuch: GnuPG-SmartCard und NitroKey. (2017). Retrieved July 2017 from https://www.privacy-handbuch.de/handbuch_32r.htm
[15] Qi Alfred Chen, Zhiyun Qian, and Z. Morley Mao. 2014. Peeking into Your App without Actually Seeing It: UI State Inference and Novel Android Attacks. In *23rd USENIX Security Symposium (USENIX Security)*. USENIX Association, San Diego, CA, 1037–1052.
[16] Yongsoon Choi, Jordan Tewell, Yukihiro Morisawa, Gilang A. Pradana, and Adrian David Cheok. 2014. Ring*U: A Wearable System for Intimate Communication Using Tactile Lighting Expressions. In *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology (ACE '14)*. ACM, Article 63, 4 pages. https://doi.org/10.1145/2663806.2663814
[17] Brett Cooley, Haining Wang, and Angelos Stavrou. 2014. *Activity Spoofing and Its Defense in Android Smartphones.* Springer International Publishing, Cham, 494–512. https://doi.org/10.1007/978-3-319-07536-5_29
[18] Stephen M. Curry. 1998. An introduction to the Java Ring. (April 1998). Retrieved July 2017 from http://www.javaworld.com/article/2076641/learn-java/an-introduction-to-the-java-ring.html
[19] Ronald Dekker. 2017. A Simple Method to Measure Unknown Inductors. (2017). Retrieved July 2017 from http://www.dos4ever.com/inductor/inductor.html
[20] Frank Denis. 2017. The Sodium crypto library (libsodium). (2017). Retrieved July 2017 from https://libsodium.org

[21] Arkajit Dey and Stephen Weis. 2008. Keyczar: A Cryptographic Toolkit. (Aug. 2008). Retrieved July 2017 from http://keyczar.googlecode.com/files/keyczar05b.pdf

[22] W. Diao, X. Liu, Z. Li, and K. Zhang. 2016. No Pardon for the Interruption: New Inference Attacks on Android Through Interrupt Timing Analysis. In *IEEE Symposium on Security and Privacy (SP)*. 414–432. https://doi.org/10.1109/SP.2016.32

[23] ECMA International. 2015. NFC-SEC-01: NFC-SEC Cryptography Standard using ECDH and AES, 4rd edition. ECMA-386. (June 2015). http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-386.pdf

[24] ECMA International. 2015. NFC-SEC: NFCIP-1 Security Services and Protocol, 4rd edition. ECMA-385. (June 2015). http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-385.pdf

[25] Manuel Egele, David Brumley, Yanick Fratantonio, and Christopher Kruegel. 2013. An Empirical Study of Cryptographic Misuse in Android Applications. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS '13)*. ACM, 73–84. https://doi.org/10.1145/2508859.2516693

[26] Rebecca Ehlers, Thorsten Ehlers, Werner Koch, and Matthias Kirschner. 2006. The GnuPG Smartcard HOWTO. (June 2006). Retrieved July 2017 from https://www.gnupg.org/howtos/card-howto/en/smartcard-howto.html

[27] Nikolay Elenkov. 2014. *Android Security Internals: An In-Depth Guide to Android's Security Architecture.* No Starch Press.

[28] M. Elkins, D. Del Torto, R. Levien, and T. Roessler. 2001. MIME Security with OpenPGP. RFC 3156 (Proposed Standard). (Aug. 2001).

[29] Sascha Fahl, Marian Harbach, Thomas Muders, Lars Baumgärtner, Bernd Freisleben, and Matthew Smith. 2012. Why Eve and Mallory love Android: An analysis of Android SSL (in) security. In *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 50–61.

[30] Sascha Fahl, Marian Harbach, Thomas Muders, Matthew Smith, and Uwe Sander. 2012. Helping Johnny 2.0 to Encrypt His Facebook Conversations. In *Proceedings of the Eighth Symposium on Usable Privacy and Security (SOUPS '12)*. ACM, Article 11, 17 pages. https://doi.org/10.1145/2335356.2335371

[31] Sascha Fahl, Marian Harbach, Henning Perl, Markus Koetter, and Matthew Smith. 2013. Rethinking SSL Development in an Appified World. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS '13)*. ACM, 49–60. https://doi.org/10.1145/2508859.2516655

[32] Fidesmo. 2017. Card App Store. (2017). Retrieved July 2017 from http://www.fidesmo.com

[33] 'Fluffy'. 2017. OpenPGP-Card. (2017). Retrieved July 2017 from https://github.com/FluffyKaon/OpenPGP-Card

[34] Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. 2010. *Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones.* Springer Berlin Heidelberg, Berlin, Heidelberg, 35–49. https://doi.org/10.1007/978-3-642-16822-2_4

[35] Simson L. Garfinkel, David Margrave, Jeffrey I. Schiller, Erik Nordlander, and Robert C. Miller. 2005. How to Make Secure Email Easier to Use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. ACM, 701–710. https://doi.org/10.1145/1054972.1055069

[36] Simson L. Garfinkel and Robert C. Miller. 2005. Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express. In *Proceedings of the 2005 Symposium on Usable Privacy and Security (SOUPS '05)*. ACM, 13–24. https://doi.org/10.1145/1073001.1073003

[37] GitHub. 2017. OpenSC/OpenSC. (2017). Retrieved July 2017 from https://github.com/OpenSC/OpenSC

[38] GNOME. 2014. Keyring. (2014). Retrieved July 2017 from https://wiki.gnome.org/action/show/Projects/GnomeKeyring

[39] GNOME. 2014. Seahorse Roadmap. (2014). Retrieved July 2017 from https://wiki.gnome.org/Apps/Seahorse/Roadmap

[40] GnuPG authors. 2017. Appendix A The GnuPG UI Server Protocol. (2017). Retrieved July 2017 from https://www.gnupg.org/documentation/manuals/gpgme/UI-Server-Protocol.html

[41] GnuPG authors. 2017. GPA - The Gnu Privacy Assistant. (2017). Retrieved July 2017 from https://www.gnupg.org/software/gpa/index.html

[42] Ernst Haselsteiner and Klemens Breitfuß. 2006. Security in Near Field Communication (NFC). In *Printed Handout of Workshop on RFID Security (RFIDSec)*. Philips Semiconductors.

[43] Mario Heiderich, Jann Horn, Abraham Aranguren, Jonas Magazinius, and Dario Weißer. 2015. Pentest-Report OpenKeychain. (Aug. 2015). https://cure53.de/pentest-report_openkeychain.pdf.

[44] Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics* (1979), 65–70.

[45] Michael Hölzl, Endalkachew Asnake, René Mayrhofer, and Michael Roland. 2014. Mobile Application to Java Card Applet Communication using a Password-authenticated Secure Channel. In *12th International Conference on Advances in Mobile Computing and Multimedia (MoMM)*. ACM Press, New York, NY, USA, 147–156. https://doi.org/10.1145/2684103.2684128

[46] Identiv. 2015. uTrust 2910 R Data Sheet. (Feb. 2015). http://www.identiv.com/pdf/technicaldata/technical-datasheets/uTrust_2910R_Reader_DS_2015_02.pdf

[47] ISO/IEC. 2008. *ISO/IEC 14443-4: Identification cards – Contactless integrated circuit cards – Proximity cards – Part 4: Transmission protocol.*

[48] ISO/IEC. 2013. *ISO/IEC 7816-4: Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange.*

[49] A. K. Jain, A. Ross, and S. Pankanti. 2006. Biometrics: a tool for information security. *IEEE Transactions on Information Forensics and Security* 1, 2 (June 2006), 125–143. https://doi.org/10.1109/TIFS.2006.873653

[50] KDE. 2017. Kleopatra - Certificate Manager and Unified Crypto GUI. (2017). Retrieved July 2017 from https://www.kde.org/applications/utilities/kleopatra/

[51] Henning Kortvedt and S Mjolsnes. 2009. Eavesdropping near field communication. In *The Norwegian Information Security Conference (NISK)*, Vol. 27.

[52] Juan Lang, Alexei Czeskis, Dirk Balfanz, Marius Schilder, and Sampath Srinivas. 2017. *Security Keys: Practical Cryptographic Second Factors for the Modern Web*. Springer Berlin Heidelberg, Berlin, Heidelberg, 422–440. https://doi.org/10.1007/978-3-662-54970-4_25

[53] Frederic Lardinois. 2015. Google And Samsung Will Now Release Monthly OTA Android Security Updates. (Aug. 2015). http://techcrunch.com/2015/08/05/google-and-samsung-will-now-release-monthly-ota-android-security-updates

[54] Shrirang Mare, Mary Baker, and Jeremy Gummeson. 2016. A Study of Authentication in Daily Life. In *Twelfth Symposium on Usable Privacy and Security (SOUPS)*. USENIX Association, Denver, CO, 189–206.

[55] Mario Heiderich and Krzysztof Kotowicz. 2013. Pentest-Report Mailvelope 12.2012 - 02.2013. (2013). Retrieved July 2017 from https://cure53.de/pentest-report_mailvelope.pdf

[56] Mindi McDowell, Jason Rafail, and Shawn Hernan. 2009. Cyber Security Tip ST04-002. US-CERT. (2009). Retrieved July 2017 from http://www.us-cert.gov/cas/tips/ST04-002.html

[57] Kenneth O McGraw and SP Wong. 1992. A common language effect size statistic. *Psychological bulletin* 111, 2 (1992), 361.

[58] John McLear. 2013. NFC Ring - One Smart Ring, Unlimited Possibilities. (July 2013). Retrieved July 2017 from https://www.kickstarter.com/projects/mclear/nfc-ring

[59] Maryam Mehrnezhad, Mohammed Aamir Ali, Feng Hao, and Aad van Moorsel. 2016. *NFC Payment Spy: A Privacy Attack on Contactless Payments*. Springer International Publishing, Cham, 92–111. https://doi.org/10.1007/978-3-319-49100-4_4

[60] MOTA. 2017. MOTA DOI SmartRing. (2017). Retrieved July 2017 from http://shop.mota.com/mota-doi-smartring.html

[61] Nitrokey. 2017. OpenPGP support. (2017). Retrieved July 2017 from https://github.com/Nitrokey

[62] Nitrokey. 2017. Secure your digital life. (2017). Retrieved July 2017 from https://www.nitrokey.com

[63] NXP Semiconductors. 2010. AN1445: Antenna design guide for MFRC52x, PN51x and PN53x. (Oct. 2010). http://data.nxp.com/doc/published_files/1270733179751

[64] OpenIntents. 2017. Where applications unite. (2017). Retrieved July 2017 from http://www.openintents.org

[65] OpenKeychain. 2017. Easy PGP. (2017). Retrieved July 2017 from https://www.openkeychain.org

[66] OpenSSL. 2016. Libcrypto API. (2016). Retrieved July 2017 from https://wiki.openssl.org/index.php/Libcrypto_API

[67] Celeste Lyn Paul, Emile Morse, Aiping Zhang, Yee-Yin Choong, and Mary Theofanos. 2011. A field study of user behavior and perceptions in smartcard authentication. In *Human-Computer Interaction–INTERACT 2011*. Springer, 1–17.

[68] A. Pietig. 2009. Functional Specification of the OpenPGP application on ISO Smart Card Operating Systems. (April 2009). http://www.g10code.com/docs/openpgp-card-3.0.pdf

[69] Precise Biometrics. 2017. Smart Card Readers for Convenient and Secure Access. (2017). Retrieved July 2017 from http://precisebiometrics.com/smart-card-reader

[70] Chuangang Ren, Yulong Zhang, Hui Xue, Tao Wei, and Peng Liu. 2015. Towards Discovering and Understanding Task Hijacking in Android. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., 945–959.

[71] Arne Renkema-Padmos, Jerome Baum, Melanie Volkamer, and Karen Renaud. 2014. Shake Hands to Bedevil: Securing Email with Wearable Technology.. In *Proceedings of the Eighth International Symposium on. Human Aspects of Information Security & Assurance (HAISA 2014)*. 90–100.

[72] Research In Motion Limited. 2007. Smart Card Security Solved: The BlackBerry Smart Card Reader. (2007). Retrieved July 2017 from http://www.blackberry.com/newsletters/connection/it/i5-2007/smart-card-reader.shtml

[73] RINGLY. 2017. Smart Jewelry and Accessories. (2017). Retrieved July 2017 from https://ringly.com

[74] Michael Roland and Michael Hölzl. 2015. Evaluation of Contactless Smartcard Antennas. (July 2015). http://arxiv.org/abs/1507.06427

[75] Martina Angela Sasse. 2005. Usability and trust in information systems. In *Trust and Crime in Information Societies*, R Mansell and B Collins (Eds.). Edward Elgar, Cheltenham, UK, 319–348.

[76] Florian Schmaus, Dominik Schürmann, and Vincent Breitmoser. 2016. *XEP-0373: OpenPGP for XMPP*. Technical Report. XMPP Standards Foundation, http://xmpp.org/extensions/xep-0373.html.

[77] Florian Schmaus, Dominik Schürmann, and Vincent Breitmoser. 2016. *XEP-0374: OpenPGP for XMPP Instant Messaging*. Technical Report. XMPP Standards Foundation, http://xmpp.org/extensions/xep-0374.html.

[78] Dominik Schürmann and Lars Wolf. 2016. Surreptitious Sharing on Android. In *Sicherheit 2016 (Lecture Notes in Informatics)*, Vol. P-256. Gesellschaft für Informatik, Bonn, Germany, 137–148. http://www.ibr.cs.tu-bs.de/papers/schuermann-sicherheit2016.pdf

[79] Dennis D Strouble, GM Schechtman, and Alan S Alsop. 2009. Productivity and usability effects of using a two-factor security system. *Proceedings of SAIS* (2009), 196–201.

[80] Michael Tunstall. 2006. Attacks on Smart Cards. (2006). http://www.cs.bris.ac.uk/home/tunstall/presentation/AttacksonSmartCards.pdf

[81] Alma Whitten and J. Doug Tygar. 1999. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *Proceedings of the 8th Conference on USENIX Security Symposium - Volume 8 (SSYM'99)*. USENIX Association.

[82] Meng Xu, Chengyu Song, Yang Ji, Ming-Wei Shih, Kangjie Lu, Cong Zheng, Ruian Duan, Yeongjin Jang, Byoungyoung Lee, Chenxiong Qian, Sangho Lee, and Taesoo Kim. 2016. Toward Engineering a Secure Android Ecosystem: A Survey of Existing Techniques. *ACM*

Comput. Surv. 49, 2, Article 38 (Aug. 2016), 47 pages. https://doi.org/10.1145/2963145

[83] Yubico. 2017. Trust the Net with YubiKey Strong Two-Factor Authentication. (2017). Retrieved July 2017 from https://www.yubico.com

[84] Yubico. 2017. YubiKey NEO's OpenPGP app. (maintained fork of "Java Card OpenPGP Card"). (2017). Retrieved July 2017 from https://github.com/Yubico/ykneo-openpgp

[85] Zimperium. 2015. Experts Found a Unicorn in the Heart of Android. (July 2015). Retrieved July 2017 from https://blog.zimperium.com/experts-found-a-unicorn-in-the-heart-of-android

Received May 2017; accepted July 2017.

5.2 Implications and Future Directions

This study successfully demonstrates the feasibility and practicality of offloading cryptographic key storage to wearable NFC devices, such as rings, for use in common applications like email. The architecture effectively separates key management from mobile devices with minimal user effort, enhancing security by utilizing external tokens. Lab evaluations under field conditions validate the proposed approach's usability, performance, and security, providing actionable insights for real-world adoption and future developments in wearable cryptographic solutions. The use of NFC-enabled wearables for cryptographic operations can extend to enterprise environments for secure document access or financial systems for transaction authentication, demonstrating the architecture's versatility.

The increasing utilization of wearables with smart devices expands their potential for secure authentication and key management [39]. Initial use cases, such as smartwatch unlocking and two-factor authentication [40, 41], lay the foundation for further research into biometric authentication, decentralized identity management, and interoperability with IoT ecosystems, enabling scalable and user-centric cryptographic solutions.

CHAPTER 6

Conclusions

Secure messaging has evolved from a niche domain into an essential component of modern digital communication infrastructure. Today, these systems serve over two billion users daily, spanning both personal and professional contexts. This dissertation examined four interrelated challenges central to the development and adoption of secure messaging systems, each investigated through a peer-reviewed publication. The challenges are the systematic evaluation of secure messaging protocols (RQ1), the design and assessment of usable key fingerprint verification for trust establishment (RQ2), the exploration of user perceptions and mental models of end-to-end security and encryption in general (RQ3), the integration and acceptance of hardware-based authentication mechanisms to simplify and strengthen key management (RQ4). This dissertation situates these studies within real-world ecosystems, demonstrating the necessity of a holistic, human-centered approach to secure messaging. The findings emphasize that robust technical security must be balanced with user understanding and practical usability to ensure communication adoption and effective protection.

Together, these interconnected studies provide new theoretical insights and practical guidance on integrating human aspects in secure messaging, illustrating how robust cryptographic methodologies and user-centered design principles must work in tandem. The following sections summarize the key findings, contributions, unresolved challenges, real-world impact, and directions for future work. As an initial step, we now focus on the key findings and principal contributions of the conducted research, shedding light on the core insights gained:

6.1 Summary of Research Outcomes

Our research yielded significant insights across our four presented research questions (RQ1 - RQ4):

RQ 1 | Systematization of Secure Messaging The analysis presented in Chapter 2 systematically mapped how various secure messaging protocols and tools address core aspects of secure communication – trust establishment, conversation security, and transport privacy– thereby establishing a rigorous foundation with the potential assessing and developing secure messaging protocols, while always considering the human aspects. This evaluation framework not only introduced standardized terminology and synthesized approaches spanning academic research and industry practice but also illuminated open research challenges, challenges such as enabling user-friendly

authority-based verification mechanisms. These findings laid the groundwork for subsequent research into improving secure messaging usability and adoption.

RQ 2 | Trust Establishment and Key Verification The empirical study presented in Chapter 3 conducted the first large-scale investigation of secure messaging key verification through a large-scale online experiment (n=1047 participants) that simulated realistic attack scenarios, where a plausible preimage collision attack was designed to test the different representations. The findings demonstrated that alternative fingerprint representations, particularly the newly developed sentence-based approach (but also the existing numeric schemes), significantly improve the usability and security of manual fingerprint verification compared to traditional and most adopted hexadecimal strings, which performed worst. These improvements manifested in enhanced user comprehension and significantly higher detection rates of attack attempts under realistic conditions, providing evidence-based guidance for secure messaging applications. Finally, concrete design recommendations for implementing more effective trust establishment mechanisms in secure messaging applications were recommended.

RQ 3 | Mental Models and User Perception of Secure Messaging The study in Chapter 4 revealed complex and often contradictory user attitudes in secure messaging toward end-to-end security and encryption in general. Through qualitative interviews (n=22), this work uncovered that despite the widespread deployment of end-to-end encryption in popular messaging apps, users heard about it for the first time and even exhibited significant skepticism and mistrust. Notably, even when users were aware of encryption features, they often dismissed their effectiveness or questioned their trustworthiness. The analysis identified several key factors contributing to this mistrust: limited understanding of encryption mechanisms (and underestimating its potential), skepticism about service providers' motivations, and a significant overestimation of attackers' capabilities. These findings challenge the assumption that merely implementing and advertising security features is sufficient for user acceptance, suggesting instead that secure messaging applications must address fundamental issues of trust and transparency.

RQ 4 | Hardware-Based Key Storage by using Wearables The study in Chapter 5 presented a novel architecture for integrating external cryptographic hardware in the form of an NFC-enabled ring as wearable into mobile security applications to handle key management. This work demonstrated how wearables and other external cryptography-capable devices, particularly NFC-enabled rings and smartcards, can enhance the security of key management and storage. It also has the potential to increase usability by abstracting the key management problem from the user, thereby bridging the gap between usability and security. While the results demonstrated the technical feasibility and security benefits of hardware-based approaches, the analysis revealed promising user feedback and important considerations for mass-market adoption, particularly regarding usability and ecosystem integration.

6.2 Synthesis and Broader Impact

During the course of this research, secure messaging underwent a significant transformation. The period began with encrypted messaging as a niche feature used primarily by privacy-conscious individuals.

This thesis advances the field of Secure Messaging through several key human aspect insights. It provides a unified evaluation framework for secure messaging protocols while emphasizing usability and adoption metrics. This helps categorize and evaluate existing and future messaging protocols. Building on this foundation, the research developed two distinct methodologies. The first developed a quantitative experimental approach testing how users actually behave when faced with security decisions in realistic scenarios. This helped to systematically observe and measure real-world security behaviors rather than relying solely on theoretical models or self-reported data. The testbed enables empirical validation of security and usability implementations in practical settings. The second methodology employed qualitative research techniques to deeply understand how users conceptualize and think about security in messaging applications. This approach revealed users' mental models, assumptions, and perceptions regarding end-to-end security through qualitative, semi-structured, in-depth interviews. Finally, the findings highlight significant gaps between technical security implementations and user comprehension, informing future secure system designs.

The research also produced two new concrete approaches to bridge these gaps:

Key Fingerprint Verification First, the sentence-based key fingerprint verification system serves as a foundational approach for trust establishment that can be adapted beyond messaging to various secure communication contexts, from SSH connections to IoT device authentication. Throughout the research process, the findings were shared with key stakeholders in the messaging community, including collaboration with developers of the `Signal protocol`, which became the security base for `WhatsApp` (among others), and members of the Electronic Frontier Foundation (EFF). These exchanges provided a feedback mechanism, ensuring that the research remained aligned with practical considerations and real-world applications. Eventually, the two most popular messengers `WhatsApp` and `iMessage`, transitioned to end-to-end security by default, while the public key handling comes from them as "trusted authority", there is still the option to verify the keys if needed (assuming that their apps are backdoor-free), gives us hope that a mass leakage or surveillance of private messaging conversations becomes less likely.

Key Management in Wearables The `OpenKeychain` NFC implementation demonstrates how hardware-based security through wearables can simplify key management while also increasing security guarantees. This work anticipated the growing role of wearable technology in security, a trend now evident in the widespread adoption of wearables for authentication and access control. The architecture and design patterns developed provide a blueprint for implementing secure key management in emerging wearable platforms.

Similarly, with the increasing prevalence of wearables, such as Apple Watch and Oura Ring, and already established security features using these, e. g., smartphone unlocking [40] and two-factor authentication [41], suggests that further application similar to our implementation become widely adopted in the future.

6.3 Open Challenges and Future Work

Despite the advances demonstrated through our research, several fundamental challenges in secure messaging remain that warrant future investigation. While this work's systematization presented various potent improvements to trust establishment by using authority-based trust with

key transparency, which has even been adopted in popular messengers, this process still requires basic end-to-end security knowledge and little effort from its users. Further research could identify and empirically validate approaches that users not only comprehend but feel genuinely motivated to employ in their daily communication practices. Furthermore, it could explore implementing and evaluating sentence-based key fingerprinting across diverse authentication scenarios beyond messaging – from SSH server connections and WIFI networks to screenless devices – where asynchronous trust establishment is essential.

As discussed in Chapter 2, Transport privacy and further social-graph-related metadata privacy continue to pose significant challenges, particularly regarding hiding social graphs and susceptibility to timing attacks revealing communication patterns. While numerous academic approaches have been presented in this work, the major platforms have not yet adopted improvements for billions of users due to potential scalability or practicability limitations (e. g., potential Denial of Service (DoS) attacks or legislative constraints), even though `Signal` pioneered in aspects significantly reducing metadata about related to social graphs with *Private Contact Discovery* and *Sealed Senders* [42, 43], which emerge after the release of *Publication 1*. Research addressing these practical constraints could significantly advance metadata protection in real-world applications. These technical challenges increasingly intersect with cultural and ethical considerations. While some users value absolute anonymity, providers and governments often prioritize accountability. Whether messaging providers pursue certificate-transparency approaches or focus on concealing communication patterns requires broader societal discussions to influence the design and capabilities of next-generation secure communication tools.

The integration of hardware security mechanisms presents another promising direction. With wearables already established for the discussed security use cases like unlocking and two-factor authentication, cryptographic key management can be further automated and secured. Mass evaluation through field studies would strengthen automated cryptographic key management through wearables in real-world settings.

Finally, expanding studies on user interaction with secure systems remains crucial to better align technical capabilities with human behaviors. Addressing these persistent challenges can help secure messaging solutions achieve broader adoption and foster deeper user trust without compromising security guarantees. The evolution of secure messaging from a niche technology to a mainstream medium not only reflects the progress made so far but emphasizes the importance of understanding and addressing human aspects to achieve secure, private, and widely adopted messaging systems.

Bibliography

[1] Sergej Dechand, Dominik Schürmann, Karoline Busse, Yasemin Acar, Sascha Fahl, and Matthew Smith: *An Empirical Study of Textual Key-Fingerprint Representations*, USENIX Security Symposium, USENIX Association, 2016 193, ISBN: 978-1-931971-32-4, (cit. on pp. v, 4–6, 40).

[2] Sergej Dechand, Alena Naiakshina, Anastasia Danilova, and Matthew Smith: *In encryption we don't trust: The effect of end-to-end encryption to the masses on user perception*, doi: 10.1109/EuroSP.2019.00037, 2019 IEEE European Symposium on Security and Privacy, IEEE, 2019 401, DOI: 10.1109/EuroSP.2019.00037 (cit. on pp. v, 3–5, 8, 60, 76).

[3] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith: *SoK: Secure Messaging*, doi: 10.1109/SP.2015.22, 2015 IEEE Symposium on Security and Privacy (S&P), IEEE, 2015 232, DOI: 10.1109/SP.2015.22 (cit. on pp. v, 3–6, 11, 12).

[4] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith: *SoK: Secure Messaging (Extended Version)*, tech. rep. 2015-02, University of Waterloo, 2015, https://cacr.uwaterloo.ca/techreports/2015/cacr2015-02.pdf (cit. on pp. v, 4, 6, 11, 12).

[5] Dominik Schürmann, Sergej Dechand, and Lars Wolf: *OpenKeychain: An Architecture for Cryptography with Smart Cards and NFC Rings on Android*, doi: 10.1145/3130964, Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies **1** (2017) 99:1, ISSN: 2474-9567, DOI: 10.1145/3130964 (cit. on pp. v, 4, 5, 9, 77, 78).

[6] Sascha Fahl, Sergej Dechand, Henning Perl, Felix Fischer, Jaromir Smrcek, and Matthew Smith: *Hey, NSA: Stay Away from my Market! Future Proofing App Markets against Powerful Attacker*, doi: 10.1145/2660267.2660311, Conference on Computer and Communications Security, ACM, 2014 13, DOI: 10.1145/2660267.2660311 (cit. on pp. v, 38).

[7] Khaled Yakdan, Sergej Dechand, Elmar Gerhards-Padilla, and Matthew Smith: *Helping Johnny to Analyze Malware: A Usability-Optimized Decompiler and Malware Analysis User Study*, doi: 10.1109/SP.2016.18, Proceedings of the 37th IEEE Symposium on Security and Privacy, San Jose, California, USA, 2016, DOI: 10.1109/SP.2016.18 (cit. on pp. v, 58).

[8] Anil Kurmus, Sergej Dechand, and Rüdiger Kapitza: *Quantifiable run-time kernel attack surface reduction*, doi: 10.1007/978-3-319-08509-8_12, International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2014 212, DOI: 10.1007/978-3-319-08509-8_12 (cit. on p. vi).

[9] Marten Oltrogge, Yasemin Acar, Sergej Dechand, Matthew Smith, and Sascha Fahl: *To Pin or Not to Pin-Helping App Developers Bullet Proof Their TLS Connections*, USENIX Security Symposium, 2015 239, (cit. on p. vi).

[10] Henning Perl, Sergej Dechand, Matthew Smith, Daniel Arp, Fabian Yamaguchi, Konrad Rieck, Sascha Fahl, and Yasemin Acar: *Vccfinder: Finding potential vulnerabilities in open-source projects to assist code audits*, doi: 10.1145/2810103.2813604, Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, 2015 426, DOI: 10.1145/2810103.2813604 (cit. on p. vi).

[11] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, Marco Herzog, Sergej Dechand, and Matthew Smith: *Why Do Developers Get Password Storage Wrong?: A Qualitative Usability Study*, doi: 10.1145/3133956.3134082, Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2017 311, DOI: 10.1145/3133956.3134082 (cit. on p. vi).

[12] Philip R Zimmermann: *The Official PGP User's Guide*, ISBN: 0-262-74017-6, MIT Press (1995) (cit. on p. 1).

[13] B. Ramsdell and S. Turner: *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification* (2010), RFC 5751, https://tools.ietf.org/rfc/rfc5751.txt (cit. on p. 1).

[14] Alma Whitten and J Doug Tygar: *Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0*, Proceedings of the 8th Conference on USENIX Security Symposium - Volume 8, SSYM'99, Washington, D.C.: USENIX Association, 1999 (cit. on pp. 1, 3).

[15] Simson L Garfinkel and Robert C Miller: *Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express*, doi: 10.1145/1073001.1073003, Proceedings of the 2005 Symposium on Usable Privacy and Security, SOUPS '05, ACM, New York, NY, USA: ACM, 2005 13, ISBN: 1-59593-178-3, DOI: 10.1145/1073001.1073003 (cit. on pp. 1, 3).

[16] Simson L Garfinkel, David Margrave, Jeffrey I Schiller, Erik Nordlander, and Robert C Miller: *How to Make Secure Email Easier to Use*, doi: 10.1145/1054972.1055069, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '05, ACM, New York, NY, USA: ACM, 2005 701, ISBN: 1-58113-998-5, DOI: 10.1145/1054972.1055069 (cit. on pp. 1, 3).

[17] Karen Renaud, Melanie Volkamer, and Arne Renkema-Padmos: *Why Doesn't Jane Protect Her Privacy?* Privacy Enhancing Technologies, Springer, 2014 244 (cit. on pp. 1, 3).

[18] Christian Stransky, Oliver Wiese, Volker Roth, Yasemin Acar, and Sascha Fahl: *27 Years and 81 Million Opportunities Later: Investigating the Use of Email Encryption for an Entire University*, To appear in 43rd IEEE Symposium on Security & Privacy (SP'22), IEEE Computer Society, 2022 (cit. on pp. 1, 5).

[19] Ian Goldberg, Berkant Ustaoğlu, Matthew D Van Gundy, and Hao Chen:
Multi-party Off-the-Record Messaging,
Conference on Computer and Communications Security, ACM, 2009 358 (cit. on p. 2).

[20] Nikita Borisov, Ian Goldberg, and Eric Brewer:
Off-the-Record Communication, or, Why Not To Use PGP,
Workshop on Privacy in the Electronic Society, ACM, 2004 77 (cit. on p. 2).

[21] Google Security Blog: *Attackers Forge SSL Certificates for Gmail in Iran*, Accessed: 2023-06-01, 2009, https://security.googleblog.com/2009/06/attackers-forge-ssl-certificates-for.html (cit. on p. 2).

[22] Salah Albakri: *The DigiNotar Certificate Authority breach: Analysis and implications*,
Communications of the ACM **54** (2011) 24 (cit. on pp. 2, 3).

[23] Glenn Greenwald: *NSA collecting phone records of millions of Verizon customers daily*,
Accessed: 2024-11-20, 2013,
https://www.theguardian.com/world/2013/jun/06/nsa-phone-records-verizon-court-order (cit. on p. 2).

[24] Andrea Peterson: *Sony Pictures hack: the whole story*,
https://www.washingtonpost.com/news/the-switch/wp/2014/12/18/the-sony-pictures-hack-explained/,
Accessed: 2023-06-01, 2014 (cit. on pp. 2, 3).

[25] WhatsApp Inc.: *How Does WhatsApp Use End-to-End Encryption?* Accessed: 2024-12-08, 2023,
https://faq.whatsapp.com/820124435853543 (cit. on p. 3).

[26] Apple Inc.: *iMessage Contact Key Verification*,
Apple Security Engineering and Architecture Blog,
Apple Security Engineering and Architecture, 2023,
https://security.apple.com/blog/imessage-contact-key-verification/ (visited on 12/02/2024) (cit. on pp. 3, 57).

[27] Moxie Marlinspike and Trevor Perrin:
The Signal Protocol: The X3DH Key Agreement Protocol and Double Ratchet Algorithm,
Accessed: 2024-12-08. Includes X3DH and Double Ratchet specifications., 2016,
https://signal.org/docs/specifications/x3dh/x3dh.pdf
(cit. on pp. 3, 38).

[28] Trevor Perrin and Moxie Marlinspike: *The Double Ratchet Algorithm*,
Accessed: 2024-12-08. Official specification of the Double Ratchet Algorithm., 2016,
https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf (cit. on p. 3).

[29] Matrix.org Foundation: *Matrix Specification: Decentralized Communication Standard*,
version 1.12, Accessed: 2024-12-08. Official specification of the Matrix Protocol., 2023,
https://spec.matrix.org/v1.12/ (cit. on p. 3).

[30] Journal of Systems Research: *Systemization of Knowledge (SoK) Papers*,
https://www.jsys.org/type_SoK/, Accessed: 2024-11-27, 2024
(cit. on pp. 6, 11).

[31] Oakland: *Systematization of Knowledge (SoK) FAQ*, https://oakland31.cs.virginia.edu/sokfaq.html, Accessed: 2024-11-20, 2024 (cit. on pp. 6, 11).

[32] WhatsApp: *Encryption Overview*, https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf, 2016 (cit. on p. 38).

[33] Marcela S Melara, Aaron Blankstein, Joseph Bonneau, Edward W Felten, and Michael J Freedman: {*CONIKS*}*: Bringing key transparency to end users*, 24th USENIX Security Symposium (USENIX Security 15), 2015 383 (cit. on p. 38).

[34] Mark D Ryan: *Enhanced Certificate Transparency and End-to-end Encrypted Mail*, Network and Distributed System Security Symposium, NDSS, Internet Society, 2014 (cit. on p. 38).

[35] Julia Len, Melissa Chase, Esha Ghosh, Kim Laine, and Radames Cruz Moreno: *OPTIKS: An Optimized Key Transparency System*, 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA: USENIX Association, 2024 4355, ISBN: 978-1-939133-44-1, (cit. on p. 38).

[36] Sean Lawlor and Kevin Lewi: *WhatsApp Key Transparency*, https://engineering.fb.com/2023/04/13/security/whatsapp-key-transparency/, Accessed: 2023-12-01, 2023 (cit. on p. 38).

[37] Serge Vaudenay: *Secure communications over insecure channels based on short authenticated strings*, Annual International Cryptology Conference, Springer, 2005 309 (cit. on p. 57).

[38] Electronic Frontier Foundation (EFF): *Tool Guides - Surveillance Self-Defense*, Accessed: 2024-11-27, 2024, https://ssd.eff.org/module-categories/tool-guides (cit. on p. 57).

[39] Ashwini Nagappan, Adriana Krasniansky, and Madelyn Knowles: *Patterns of Ownership and Usage of Wearable Devices in the United States, 2020-2022: Survey Study*, doi: 10.2196/56504, J Med Internet Res **26** (2024) e56504, ISSN: 1438-8871, DOI: 10.2196/56504, https://www.jmir.org/2024/1/e56504 (cit. on p. 103).

[40] Apple Inc.: *Use your Apple Watch to unlock your iPhone when Face ID detects a face with a mask*, Accessed: 2024-01-20, 2021, https://support.apple.com/en-us/HT212208 (cit. on pp. 103, 106).

[41] Twilio Inc.: *Twilio Authy Authenticator*, Accessed: 2024-01-20, 2023, https://www.twilio.com/docs/authy/authy-app-documentation (cit. on pp. 103, 106).

[42] Signal Foundation: *Private Contact Discovery for Signal*, https://signal.org/blog/private-contact-discovery/, Accessed: 2024-12-12, 2017 (cit. on p. 107).

[43] Signal Foundation: *Sealed Sender for Signal*, https://signal.org/blog/sealed-sender/, Accessed: 2024-12-12, 2018 (cit. on p. 107).

APPENDIX A

Incorporated Articles Part of the Thesis

A.1 Publication 1 | SoK: Secure Messaging

2015 IEEE Symposium on Security and Privacy

SoK: Secure Messaging[1]

Nik Unger*, Sergej Dechand†
Joseph Bonneau‡§, Sascha Fahl¶, Henning Perl¶
Ian Goldberg*, Matthew Smith†
* University of Waterloo, † University of Bonn, ‡ Stanford University, § Electronic Frontier Foundation, ¶ Fraunhofer FKIE

***Abstract*—Motivated by recent revelations of widespread state surveillance of personal communication, many solutions now claim to offer secure and private messaging. This includes both a large number of new projects and many widely adopted tools that have added security features. The intense pressure in the past two years to deliver solutions quickly has resulted in varying threat models, incomplete objectives, dubious security claims, and a lack of broad perspective on the existing cryptographic literature on secure communication.**

In this paper, we evaluate and systematize current secure messaging solutions and propose an evaluation framework for their security, usability, and ease-of-adoption properties. We consider solutions from academia, but also identify innovative and promising approaches used "in-the-wild" that are not considered by the academic literature. We identify three key challenges and map the design landscape for each: trust establishment, conversation security, and transport privacy. Trust establishment approaches offering strong security and privacy features perform poorly from a usability and adoption perspective, whereas some hybrid approaches that have not been well studied in the academic literature might provide better trade-offs in practice. In contrast, once trust is established, conversation security can be achieved without any user involvement in most two-party conversations, though conversations between larger groups still lack a good solution. Finally, transport privacy appears to be the most difficult problem to solve without paying significant performance penalties.

I. Introduction

Most popular messaging tools used on the Internet do not offer end-to-end security. Even though protocols such as OpenPGP and S/MIME have been available for decades, they have failed to achieve widespread adoption and have been plagued by usability issues [2]–[5]. However, recent revelations about mass surveillance by intelligence services have highlighted the lack of security and privacy in messaging tools and spurred demand for better solutions. A recent Pew Research poll found that 80% of Americans are now concerned about government monitoring of their electronic communications. 68% of respondents reported feeling "not very secure" or "not at all secure" when using online chat and 57% felt similarly insecure using email [6]. Consequently, many new applications claiming to offer secure communication are being developed and adopted by end users.

Despite the publication of a large number of secure messaging protocols in the academic literature, tools are being released with new designs that fail to draw upon this knowledge, repeat known design mistakes, or use cryptography in insecure ways. However, as will become clear over the course of this paper, the academic research community is also failing to learn some lessons from tools in the wild.

Furthermore, there is a lack of coherent vision for the future of secure messaging. Most solutions focus on specific issues and have different goals and threat models. This is compounded by differing security vocabularies and the absence of a unified evaluation of prior work. Outside of academia, many products mislead users by advertising with grandiose claims of "military grade encryption" or by promising impossible features such as self-destructing messages [7]–[10]. The recent EFF Secure Messaging Scorecard evaluated tools for basic indicators of security and project health [11] and found many purportedly "secure" tools do not even attempt end-to-end encryption.

We are motivated to systematize knowledge on secure messaging due to the lack of a clear winner in the race for widespread deployment and the persistence of many lingering unsolved research problems. Our primary goal is to identify where problems lie and create a guide for the research community to help move forward on this important topic. A further goal in this work is to establish evaluation criteria for measuring security features of messaging systems, as well as their usability and adoption implications. We aim to provide a broad perspective on secure messaging and its challenges, as well as a comparative evaluation of existing approaches, in order to provide context that informs future efforts. Our primary contributions are: (1) establishing a set of common security and privacy feature definitions for secure messaging; (2) systematization of secure messaging approaches based both on academic work and "in-the-wild" projects; (3) comparative evaluation of these approaches; and (4) identification and discussion of current research challenges, indicating future research directions.

We present our systematization methodology in Section II. In subsequent sections (Sections III–V), we evaluate each of the proposed problem areas (namely *trust establishment*, *conversation security* and *transport privacy*) in secure messaging. Our findings are discussed and concluded in Section VI.

II. Systematization Methodology

Over the years, hundreds of secure messaging systems have been proposed and developed in both academia and industry. An exhaustive analysis of all solutions is both infeasible and undesirable. Instead, we extract recurring secure messaging techniques from the literature and publicly available messaging tools, focusing on systematization and evaluation of the under-

[1] An extended version of this paper is available [1].

DOI 10.1109/SP.2015.22

lying concepts and the desirable secure messaging properties. In this section, we explain our precise methodology.

A. Problem Areas

While most secure messaging solutions try to deal with all possible security aspects, in our systematization, we divide secure messaging into three nearly orthogonal problem areas addressed in dedicated sections: the *trust establishment* problem (Section III), ensuring the distribution of cryptographic long-term keys and proof of association with the owning entity; the *conversation security* problem (Section IV), ensuring the protection of exchanged messages during conversations; and the *transport privacy* problem (Section V), hiding the communication metadata.

While any concrete tool must decide on an approach for each problem area, abstractly defined protocols may only address some of them. Additionally, the distinction between these three problem areas is sometimes blurred since techniques used by secure messaging systems may be part of their approach for multiple problem areas.

B. Threat Model

When evaluating the security and privacy properties in secure messaging, we must consider a variety of adversaries. Our threat model includes the following attackers:

Local Adversary (active/passive): An attacker controlling local networks (e.g., owners of open wireless access points).

Global Adversary (active/passive): An attacker controlling large segments of the Internet, such as powerful nation states or large internet service providers.

Service providers: For messaging systems that require centralized infrastructure (e.g., public-key directories), the service operators should be considered as potential adversaries.

Note that our adversary classes are not necessarily exclusive. In some cases, adversaries of different types might collude. We also assume that all adversaries are participants in the messaging system, allowing them to start conversations, send messages, or perform other normal participant actions. We assume that the endpoints in a secure messaging system are secure (i.e., malware and hardware attacks are out of scope).

C. Systematization Structure

Sections III–V evaluate *trust establishment*, *conversation security*, and *transport privacy* approaches, respectively. For each problem area, we identify desirable properties divided into three main groups: *security and privacy features*, *usability features*, and *adoption considerations*. Each section starts by defining these properties, followed by the extraction of generic approaches used to address the problem area from existing secure messaging systems. Each section then defines and evaluates these approaches, as well as several possible variations, in terms of the already-defined properties. Concrete examples of protocols or tools making use of each approach are given whenever possible. The sections then conclude by discussing the implications of these evaluations.

In each section, we include a table (Tables I, II, and III) visualizing our evaluation of approaches within that problem area. Columns in the tables represent the identified properties, while rows represent the approaches. Groups of rows begin with a generic concept, specified as a combination of cryptographic protocols, followed by extension rows that add or modify components of the base concept. Whenever possible, rows include the name of a representative protocol or tool that uses the combination of concepts. Representatives may not achieve all of the features that are possible using the approach; they are merely included to indicate where approaches are used in practice. Each row is rated as providing or not providing the desired properties. In some cases, a row might only partially provide a property, which is explained in the associated description. Due to space limitations, the discussion of some schemes listed in the tables has been omitted. Details of these and their evaluations are included in the extended version of this paper [1].

For each problem area, we identify desirable properties in three main categories:

1) Security and Privacy Properties: Most secure messaging systems are designed using standard cryptographic primitives such as hash functions, symmetric encryption ciphers, and digital signature schemes. When evaluating the security and privacy features of a scheme, we assume cryptographic primitives are securely chosen and correctly implemented. We do not attempt to audit for software exploits which may compromise users' security. However, if systems allow end users to misuse these cryptographic primitives, the scheme is penalized.

2) Usability Properties: Usability is crucial for the use and adoption of secure messaging services. Human end users need to understand how to use the system securely and the effort required to do so must be acceptable for the perceived benefits.

In previous research, various secure messaging tools have been evaluated and weaknesses in the HCI portion of their design have been revealed. The seminal paper "Why Johnny Can't Encrypt" [2] along with follow-up studies evaluating PGP tools [3], [4] and other messaging protocols [12]–[16] have also showed users encountering sever problems using encryption securely. However, these studies focused on UI issues unique to specific implementations. Because we focus on usability consequences imposed by generic concepts, our results hold for any tool that implements these concepts.

To evaluate the usability of secure messaging approaches, we examine the additional user effort (and decisions), security-related errors, and reduction in reliability and flexibility that they introduce. Our usability metrics compare this extra effort to a baseline approach with minimal security or privacy features. This is a challenging task and conventional user studies are not well suited to extract such high-level usability comparisons between disparate tools. We employed expert reviews in the form of cognitive walkthroughs of actual implementations to extract and systematize usability aspects based on Nielsen's usability principles [17]–[19], which is consistent with previous systematization efforts for security schemes in

other areas [20], [21]. These usability results supplement our technical systematization and highlight potential trade-offs between security and usability.

3) Ease of Adoption: Adoption of secure messaging schemes is not only affected by their usability and security claims, but also by requirements imposed by the underlying technology. Protocols might introduce adoption issues by requiring additional resources or infrastructure from end users or service operators. When evaluating the adoption properties of an approach, we award a good score if the system does not exceed the resources or infrastructure requirements of a baseline approach that lacks any security or privacy features.

III. Trust Establishment

One of the most challenging aspects of messaging security is *trust establishment*, the process of users verifying that they are actually communicating with the parties they intend. *Long-term key exchange* refers to the process where users send cryptographic key material to each other. *Long-term key authentication* (also called *key validation* and *key verification*) is the mechanism allowing users to ensure that cryptographic long-term keys are associated with the correct real-world entities. We use *trust establishment* to refer to the combination of *long-term key exchange* and *long-term key authentication* in the remainder of this paper. After contact discovery (the process of locating contact details for friends using the messaging service), end users first have to perform trust establishment in order to enable secure communication.

A. Security and Privacy Features

A trust establishment protocol can provide the following security and privacy features:

Network MitM Prevention: Prevents Man-in-the-Middle (MitM) attacks by local and global network adversaries.

Operator MitM Prevention: Prevents MitM attacks executed by infrastructure operators.

Operator MitM Detection: Allows the detection of MitM attacks performed by operators after they have occurred.

Operator Accountability: It is possible to verify that operators behaved correctly during trust establishment.

Key Revocation Possible: Users can revoke and renew keys (e.g., to recover from key loss or compromise).

Privacy Preserving: The approach leaks no conversation metadata to other participants or even service operators.

B. Usability Properties

Most trust establishment schemes require key management: user agents must generate, exchange, and verify other participants' keys. For some approaches, users may be confronted with additional tasks, as well as possible warnings and errors, compared to classic tools without end-to-end security. If a concept requires little user effort and introduces no new error types, we award a mark for the property to denote good usability. We only consider the minimum user interaction required by the protocol instead of rating specific implementations.

Automatic Key Initialization: No additional user effort is required to create a long-term key pair.

Low Key Maintenance: Key maintenance encompasses recurring effort users have to invest into maintaining keys. Some systems require that users sign other keys or renew expired keys. Usable systems require no key maintenance tasks.

Easy Key Discovery: When new contacts are added, no additional effort is needed to retrieve key material.

Easy Key Recovery: When users lose long-term key material, it is easy to revoke old keys and initialize new keys (e.g., simply reinstalling the app or regenerating keys is sufficient).

In-band: No *out-of-band* channels are needed that require users to invest additional effort to establish.

No Shared Secrets: Shared secrets require existing social relationships. This limits the usability of a system, as not all communication partners are able to devise shared secrets.

Alert-less Key Renewal: If other participants renew their long-term keys, a user can proceed without errors or warnings.

Immediate Enrollment: When keys are (re-)initialized, other participants are able to verify and use them immediately.

Inattentive User Resistant: Users do not need to carefully inspect information (e.g., key fingerprints) to achieve security.

C. Adoption Properties

Multiple Key Support: Users should not have to invest additional effort if they or their conversation partners use multiple public keys, making the use of multiple devices with separate keys transparent.

No Service Provider Required: Trust establishment does not require additional infrastructure (e.g., key servers).

No Auditing Required: The approach does not require auditors to verify correct behavior of infrastructure operators.

No Name Squatting: Users can choose their names and can be prevented from reserving a large number of popular names.

Asynchronous: Trust establishment can occur asynchronously without all conversation participants online.

Scalable: Trust establishment is efficient, with resource requirements growing logarithmically (or smaller) with the the total number of participants in the system.

D. Evaluation

1) Opportunistic Encryption (Baseline): We consider *opportunistic encryption*, in which an encrypted session is established without any key verification, as a baseline. For instance, this could be an OTR encryption session without any authentication. The main goal of opportunistic encryption is to counter passive adversaries; active attackers can easily execute MitM attacks. From a usability perspective, this approach is the baseline since it neither places any burden on the user nor generates any new error or warning messages.

2) TOFU: Trust-On-First-Use (TOFU) extends opportunistic encryption by remembering previously seen key material [22]. The *network MitM prevented* and *infrastructure MitM prevented* properties are only partially provided due to the requirement that no attacker is present during the initial connection. TOFU *requires no service provider* since keys can be exchanged by the conversation participants directly. TOFU does not define a mechanism for *key revocation*. TOFU can be

TABLE I
TRADE-OFFS FOR COMBINATIONS OF TRUST ESTABLISHMENT APPROACHES. SECURE APPROACHES OFTEN SACRIFICE USABILITY AND ADOPTION.

Scheme	Example	Security Features						Usability								Adoption					
		Network MitM Prevented	Operator MitM Prevented	Operator MitM Detected	Operator Accountability	Key Revocation Possible	Privacy Preserving	Automatic Key Initialization	Low Key Maintenance	Easy Key Discovery	Easy Key Recovery	In-Band	No Shared Secrets	Alert-less Key Renewal	Immediate Enrollment	Inattentive User Resistant	Multiple Key Support	No Service Provider	No Auditing Required	No Name Squatting	Asynchronous	Scalable
Opportunistic Encryption†*	TCPCrypt	-	-	-	-	-	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
+TOFU (Strict)†	-	◐	◐	◐	◐	-	●	●	●	●	-	●	●	-	●	●	-	●	●	●	●	●
+TOFU†*	TextSecure	◐	◐	◐	◐	-	●	●	●	●	●	●	●	-	●	-	-	●	●	●	●	●
Key Fingerprint Verification†*	Threema	●	●	●	●	◐	●	-	-	-	-	-	●	-	-	-	-	●	●	●	●	●
+Short Auth Strings (Out-of-Band)†*	SilentText	●	●	●	●	◐	●	-	-	-	-	-	●	-	-	-	-	-	●	●	-	●
+Short Auth Strings (In-Band/Voice/Video)†*	ZRTP	●	●	●	●	◐	●	-	-	-	-	◐	●	-	-	-	-	●	●	●	-	●
+Socialist Millionaire (SMP)†*	OTR	●	●	●	●	◐	●	-	-	-	-	●	-	-	-	-	-	●	●	●	-	●
+Mandatory Verification†*	SafeSlinger	●	●	●	●	◐	●	-	-	-	-	●	-	-	●	●	-	●	●	●	-	●
Key Directory†*	iMessage	●	-	-	-	●	-	●	●	●	●	●	●	●	●	●	●	-	●	●	◐	●
+Certificate Authority†*	S/MIME	●	-	-	-	-	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
+Transparency Log	-	●	-	◐	◐	-	-	●	●	●	●	●	●	●	◐	●	●	●	-	●	●	●
+Extended Transparency Log†	-	●	-	◐	●	●	-	●	●	●	●	●	●	●	◐	●	●	●	-	●	●	●
+Self-Auditable Log†	CONIKS	●	-	◐	●	●	●	●	●	●	●	●	●	●	●	●	●	●	◐	●	●	●
Web-of-Trust†*	PGP	●	●	●	◐	◐	-	-	-	◐	◐	-	-	-	-	-	●	●	●	●	●	●
+Trust Delegation†*	GnuNS	●	●	●	◐	◐	●	-	-	◐	◐	-	-	-	-	-	●	●	●	●	●	●
+Tracking[*]	Keybase	●	◐	◐	-	◐	-	◐	◐	◐	◐	-	-	-	●	-	●	-	●	◐	●	●
Pure IBC†	SIM-IBC-KMS	●	-	-	-	-	●	●	●	●	●	●	●	●	●	●	-	-	●	-	●	●
+Revocable IBC†	-	●	-	-	-	◐	●	●	●	●	●	●	●	●	●	●	-	-	●	-	●	●
Blockchains[*]	Namecoin	●	●	●	●	-	●	●	◐	●	-	●	●	-	●	●	●	●	-	-	●	-
Key Directory+TOFU+Optional Verification†*	TextSecure	◐	◐	◐	◐	●	-	●	●	●	●	●	●	-	●	-	◐	-	●	●	●	●
Opportunistic Encryption+SMP†*	OTR	◐	◐	◐	◐	●	●	-	●	-	-	◐	◐	-	●	-	●	●	●	●	-	●

● = provides property; ◐ = partially provides property; - = does not provide property; †has academic publication; *end-user tool available

implemented in strict and non-strict forms. The strict form fails when the key changes, providing *inattentive user resilience* but preventing *easy key recovery*. The non-strict form prompts users to accept key changes, providing *easy key recovery* at the expense of *inattentive user resilience*.

TOFU-based approaches, like the baseline, do not require any user interaction during the initial contact discovery. This yields good scores for all user-effort properties except for the *key revocation* property, which is not defined, and *alert-less key renewal*, since users cannot distinguish benign key changes from MitM attacks without additional verification methods.

From an adoption perspective, TOFU performs similarly to the baseline, except for *key recovery* in the strict version and *multiple key support* in both versions. The multiple key support problem arises from the fact that if multiple keys are used, the protocol cannot distinguish between devices. An attacker can claim that a new device, with the attacker's key, is being used.

3) Key Fingerprint Verification: Manual verification requires users to compare some representation of a cryptographic hash of their partners' public keys out-of-band (e.g., in person or via a separate secure channel).

Assuming the fingerprint check is performed correctly by end users, manual verification provides all desirable security properties with the exception of only partial *key revocation* support, as this requires contacting each communication partner out-of-band. The approaches differ only in their usability and adoption features.

Fingerprint verification approaches introduce severe usability and adoption limitations: users have to perform manual verification before communicating with a new partner (and get them to do the same) to ensure strong authentication. Thus, manual verification does not offer *automatic key initialization*, *easy key discovery*, or *immediate enrollment*. In addition, new keys introduce an alert on key renewal, resulting in a *key maintenance* effort. Fingerprints complicate *multiple key support* since each device might use a different key.

4) Short Authentication String (SAS): To ease fingerprint verification, shorter strings can be provided to the users for comparison. A SAS is a truncated cryptographic hash (e.g., 20–30 bits long) of all public parts of the key exchange. It is often represented in a format aimed to be human-friendly, such as a short sequence of words. All participants compute the SAS based on the key exchange they observed, and then compare the resulting value with each other. The method used for comparison of the SAS must authenticate the entities using some underlying trust establishment mechanism.

ZRTP, and several earlier products, use the SAS method by requiring participants to read strings aloud and thus anchors trust in the ability of participants to recognize each other's voices [23], [24]. Users who have never heard each other's voices cannot authenticate using this method. Even for users that are familiar with each other, the security provided by voice identification has been the subject of controversy [25], [26]. Recent work [27] suggests that, even with a small number of

samples of a target user's speaking voice, audio samples can be synthesized which are indistinguishable from the genuine user's voice with typical levels of background noise.

For this reason, we consider voice-based SAS verification to be obsolescent from a security standpoint. In Table I, we assume that users verify the SAS with a method providing stronger security (e.g., using audio and video channels with careful inspection during the SAS verification). If the communication channel (e.g., text messaging) does not support a mechanism to establish trust, the SAS must be compared out of band (e.g., as recommended by SilentText).

The SAS approach sacrifices *asynchronicity*, since mutual authentication must be done with all users at the same time. Due to the short size of the SAS, the naive approach is vulnerable to a MitM attack by an adversary that attempts to select key exchange values that produce a hash collision for the two connections. To mitigate this problem, the attacker can be limited to a single guess by forcing them to reveal their chosen keys before observing the keys of the honest parties. This can be accomplished by requiring that the initiator of the key exchange release a commitment to their key, and then open the commitment after the other party reveals theirs.

5) Secret-based Zero-Knowledge Verification: The Socialist Millionaire Protocol (SMP) is a zero-knowledge proof of knowledge protocol that determines if secret values held by two parties are equal without revealing the value itself. This protocol is used in OTR as the recommended method for user verification [28], [29]. Alice poses a question based on shared knowledge to Bob *in-band* and secretly records her answer. After Bob answers the question, the two parties perform the SMP to determine if their answers match, without revealing any additional information. Users are expected to choose secure questions with answers based on shared knowledge that attackers would be unable to know or guess.

Since a MitM must perform an online attack and can only guess once, even low min-entropy secrets achieve strong security [29], [30]. However, use of the SMP sacrifices *asynchronicity* since all participants must be online during the verification. If the protocol fails, the end users do not know whether their answers did not match, or if a MitM attacker exists and has made an incorrect guess.

6) Mandatory Verification: The previously defined verification methods are prone to *inattentive users*. *Mandatory verification* approaches counter user negligence by requiring that users enter the correct fingerprint strings instead of merely confirming that they are correct. Of course, entering the fingerprints takes user effort. In practice, QR-Codes and NFC are popular methods to ease this process.

In SafeSlinger the user must choose the correct answer among three possibilities to proceed [31]. After the protocol is completed, each device receives a copy of contact information shared with other participants with security guarantees including confidentiality and authenticity.

Mandatory verification inherits the usability properties of the underlying scheme. Incorporating mandatory verification sacrifices *asynchronicity* to ensure *inattentive user resistance*.

7) Authority-based Trust: In authority-based trust schemes, public keys must be vouched for by one or more trusted authorities. Two well-known examples are public-key directories and certificate authority schemes.

From the security point of view, the two schemes only differ in *key revocation* and *privacy preservation*. While key updates in key directories imply the revocation of old keys, in the CA approach, certificates signed by the authority are trusted by default; revocation lists have to be maintained separately. However, CA-based revocation lists used in web browsers are known to have issues with effectiveness and practicality [21], [32], [33]. Since certificates may be exchanged by peers directly, the CA-based approach can be *privacy preserving*.

With either system, users are vulnerable to MitM attacks by the authority, which can vouch for, or be coerced to vouch for, false keys. This weakness has been highlighted by recent CA scandals [34], [35]. Both schemes can also be attacked if the authority does not verify keys before vouching for them. Authorities in messaging services often rely on insecure SMS or email verification, enabling potential attacks.

The two approaches both support good usability. Well-known systems using public-key directories, such as iMessage, work without any user involvement.

8) Transparency Logs: A major issue with trusted authorities is that they can vouch for fraudulent keys in an attack. The *Certificate Transparency* protocol [36] requires that all issued web certificates are included in a public log.

Certificate Transparency is a specific proposal for logging PKIX certificates for TLS, but the general idea can be applied to authority-based trust establishment in secure messaging. We refer to the general concept as *transparency logs* for the remainder of the paper. While there are no known deployments to date, Google plans to adapt *transparency logs* for user keys in End-to-End, its upcoming email encryption tool [37]. In the absence of a concrete definition, we evaluate *transparency logs* based on the certificate transparency protocol.

The main security improvement of the two schemes consists of *operator accountability* and the *detection of operator MitM attacks* after the fact. The remaining security features are inherited from authority-based trust systems.

However, these schemes introduce new and unresolved usability and adoption issues. For instance, the logs must be audited to ensure correctness, negating the *no auditing required* property. The auditing services require gossip protocols to synchronize the view between the monitors and prevent attack bubbles (e.g., where different views are presented to different geographical regions) [36]. Also, since only identity owners are in a position to verify the correctness of their long-term keys, they share responsibility for verifying correct behavior of the log. Previous research has shown that users often neglect such security responsibilities [38], so this task should be performed automatically by client applications. However, if a client detects a certificate in the log that differs from their version, it is not clear whether the authorities have performed an attack, an adversary has successfully impersonated the subject of the certificate to the authorities, or if the subject

actually maintains multiple certificates (e.g., due to installing the app on a second device). Ultimately, end users have to cope with additional security warnings and errors, and it remains to be seen whether they can distinguish between benign and malicious log discrepancies without training. In addition, *transparency logs* might hamper *immediate enrollment* due to delays in log distribution.

Enhanced Certificate Transparency [39] and CONIKS [40] both improve on the basic *transparency logs* concept, but neither has yet been deployed in practice.

9) Blockchains: The Bitcoin cryptocurrency utilizes a novel distributed consensus mechanism using pseudonymous "miners" to maintain an append-only log [41]. Voting power is distributed in proportion to computational resources by using a probabilistic proof-of-work puzzle. For the currency application, this log records every transaction to prevent double-spending. The success of Bitcoin's consensus protocol has led to enthusiasm that similar approaches could maintain global consensus on other types of data, such as a mapping of human-readable usernames to keys.

Namecoin, the first fork of Bitcoin, allows users to claim identifiers, add arbitrary data (e.g., public keys) as records for those identifiers, and even sell control of their identifiers to others [42]. Namecoin and similar name-mapping blockchains are denoted by the *blockchain* entry in Table I. Unlike most other schemes, Namecoin is strictly "first-come, first-served", with any user able to purchase ownership of any number of unclaimed names for a small, fixed fee per name. This price is paid in Namecoins — units of currency that are an inherent part of the system. A small maintenance fee is required to maintain control of names, and small fees may be charged by miners to update data or transfer ownership of names.

From the security perspective, blockchain schemes achieve similar results to manual verification, except that instead of exchanging keys, the trust relies on the username only. Once users have securely exchanged usernames, they can reliably fetch the correct keys.

However, various shortcomings arise from a usability and adoption perspective. The primary usability limitation is that if users ever lose the private key used to register their name (which is not the same as the communication key bound to that name), they will permanently lose control over that name (i.e., *key recovery* is not possible). Similarly, if the key is compromised, the name can be permanently and irrevocably hijacked. Thus, the system requires significant *key management* effort and burdens users with high responsibility. If users rely on a web-based service to manage private keys for them, as many do with Bitcoin in practice, the system is no longer truly end-to-end. The system requires users to pay to reserve and maintain names, sacrificing *low key maintenance* and *automatic key initialization*. Users also cannot instantly issue new keys for their identifiers (i.e., there is no *immediate enrollment*) but are required to wait for a new block to be published and confirmed. In practice, this can take 10–60 minutes depending on the desired security level.

On the adoption side, for the system to be completely trustless, users must store the entire blockchain locally and track its progress. Experience from Bitcoin shows that the vast majority of users will not do this due to the communication and storage requirements and will instead trust some other party to track the blockchain for them. This trusted party cannot easily insert spurious records, but can provide stale information without detection. In any case, the system is not highly *scalable* since the required amount of storage and traffic consumption increases linearly with the number of users.

Finally, there are serious issues with *name squatting*, which have plagued early attempts to use the system. Because anybody can register as many names as they can afford, a number of squatters have preemptively claimed short and common names. Given the decentralized nature of blockchains, this is hard to address without raising the registration fees, which increases the burden on all users of the system.

E. Other Approaches

Due to space constraints, we omit evaluations of some approaches in Table I. Readers unfamiliar with web-of-trust schemes can refer to Appendix B. Identity Based Encryption (IBE) and Keybase are discussed in the extended paper [1].

F. Discussion

As Table I makes evident, no trust establishment approach is perfect. While it is common knowledge that usability and security are often at odds, our results show exactly where the trade-offs lie. Approaches either sacrifice security and provide a nearly ideal user experience, or sacrifice user experience to achieve nearly ideal security scores. Authority-based trust and TOFU schemes are the most usable and well-adopted, but only offer basic security properties. Not surprisingly, authority-based trust (particularly app-specific key directories) is predominant among recently developed apps in the wild, as well as among apps with the largest userbases (e.g., iMessage, BlackBerry Protected, and Wickr).

In practice, we may be faced with the constraint that *none* of the usability properties can be sacrificed in a system that will achieve mass adoption. Higher-security schemes may be useful within organizations or niche communities, but defending against mass surveillance requires a communication system that virtually all users can successfully use. Thus, it may be wise to start from the basic user experience of today's widely deployed communication apps and try to add as much security as possible, rather than start from a desired security level and attempt to make it as simple to use as possible.

There appears to be considerable room for security improvements over authoritative key directories even without changes to the user experience. Transparency logs might provide more accountability with no interaction from most users. Because this approach has not yet been deployed, it remains to be seen how much security is gained in practice. The insertion of new keys in the log does not provide public evidence of malicious behavior if insecure user authentication methods (e.g., passwords) are used to authorize key changes, as we fully

expect will be the case. Still, the possible loss of reputation may be enough to keep the server honest.

Another promising strategy is a layered design, with basic security provided by a central key directory, additional trust establishment methods for more experienced users (e.g., visual fingerprint verification or QR-codes), and TOFU warning messages whenever contacts' keys have changed. TextSecure and Threema, among others, take such a layered approach (represented by the second-to-last row in Table I). In contrast, OTR uses opportunistic encryption with the ability to perform the SMP to ensure trust (represented by the last row in Table I).

Conversely, the approaches with good security properties should focus on improving usability. There has been little academic work studying the usability of trust establishment. Further research focusing on end-users' mental models and perception for trust establishment could help to develop more sophisticated and understandable approaches.

IV. Conversation Security

After *trust establishment* has been achieved, a *conversation security* protocol protects the security and privacy of the exchanged messages. This encompasses how messages are encrypted, what data is attached to them, and what cryptographic protocols (e.g., ephemeral key exchanges) are performed. A conversation security scheme doesn't specify a trust establishment scheme nor define how transmitted data reaches the recipient.

In Table II, we compare the features of existing approaches for conversation security. Rows without circles in the "group features" columns can only be used in a two-party setting.

A. Security and Privacy Features

Confidentiality: Only the intended recipients are able to read a message. Specifically, the message must not be readable by a server operator that is not a conversation participant.

Integrity: No honest party will accept a message that has been modified in transit.

Authentication: Each participant in the conversation receives proof of possession of a known long-term secret from all other participants that they believe to be participating in the conversation. In addition, each participant is able to verify that a message was sent from the claimed source.

Participant Consistency: At any point when a message is accepted by an honest party, all honest parties are guaranteed to have the same view of the participant list.

Destination Validation: When a message is accepted by an honest party, they can verify that they were included in the set of intended recipients for the message.

Forward Secrecy: Compromising all key material does not enable decryption of previously encrypted data.

Backward Secrecy: Compromising all key material does not enable decryption of succeeding encrypted data. This property is also often called *future secrecy* [43]. The terms are controversial and vague in literature [44]–[46].

Anonymity Preserving: Any anonymity features provided by the underlying transport privacy architecture are not undermined (e.g., if the transport privacy system provides anonymity, the conversation security level does not deanonymize users by linking key identifiers).

Speaker Consistency: All participants agree on the sequence of messages sent by each participant. A protocol might perform consistency checks on blocks of messages during the protocol, or after every message is sent.

Causality Preserving: Implementations can avoid displaying a message before messages that causally precede it.

Global Transcript: All participants see all messages in the same order. Note that this implies speaker consistency.

Conversation security protocols may provide several different forms of deniability. For a detailed treatment of deniability in various contexts, see Appendix A. We define the following deniability-related features:

Message Unlinkability: If a judge is convinced that a participant authored one message in the conversation, this does not provide evidence that they authored other messages.

Message Repudiation: Given a conversation transcript and all cryptographic keys, there is no evidence that a given message was authored by any particular user. We assume that the accuser has access to the session keys because it is trivial to deny writing a plaintext message when the accuser cannot demonstrate that the ciphertext corresponds to this plaintext. We also assume that the accuser does not have access to the accused participant's long-term secret keys because then it is simple for the accuser to forge the transcript (and thus any messages are repudiable).

Participation Repudiation: Given a conversation transcript and all cryptographic key material for all but one accused participant, there is no evidence that the honest participant was in a conversation with any of the other participants.

Several additional features are only meaningful for group protocols (i.e., protocols supporting chats between three or more participants):

Computational Equality: All chat participants share an equal computational load.

Trust Equality: No participant is more trusted or takes on more responsibility than any other.

Subgroup messaging: Messages can be sent to a subset of participants without forming a new conversation.

Contractible Membership: After the conversation begins, participants can leave without restarting the protocol.

Expandable Membership: After the conversation begins, participants can join without restarting the protocol.

When a participant joins a secure group conversation, it is desirable for the protocol to compute new cryptographic keys so that the participant cannot decrypt previously sent messages. Likewise, keys should be changed when a participant leaves so that they cannot read new messages. This is trivial to implement by simply restarting the protocol, but this approach is often computationally expensive. Protocols with *expandable / contractible membership* achieve this without restarts.

There are many higher-level security and privacy design

TABLE II
CONVERSATION SECURITY PROTOCOLS AND THEIR USABILITY AND ADOPTION IMPLICATIONS. NO APPROACH REQUIRES ADDITIONAL USER EFFORT.

Scheme	Example	Security and Privacy														Adoption					Group Chat				
		Confidentiality	Integrity	Authentication	Participant Consistency	Destination Validation	Forward Secrecy	Backward Secrecy	Anonymity Preserving	Speaker Consistency	Causality Preserving	Global Transcript	Message Unlinkability	Message Repudiation	Particip. Repudiation	Out-of-Order Resilient	Dropped Message Resilient	Asynchronicity	Multi-Device Support	No Additional Service	Computational Equality	Trust Equality	Subgroup Messaging	Contractable	Expandable
TLS+Trusted Server†*	Skype	-	-	-	-	-	-	-	-	-	-	-	●	●	●	●	●	●	●	-	●	●	●	◐	◐
Static Asymmetric Crypto†*	OpenPGP, S/MIME	●	●	●	-	-	-	-	●	-	-	-	-	-	-	●	●	●	●	●					
+IBE†	Wang et al.	-	●	●	-	-	-	-	●	-	-	-	-	-	-	●	●	●	●	-					
+Short Lifetime Keys	OpenPGP Draft	●	●	●	-	-	◐	◐	●	-	-	-	-	-	-	●	●	●	●	-					
+Non-Interactive IBE†	Canetti et al.	●	●	●	-	-	●	-	●	-	-	-	-	-	-	◐	●	●	●	●					
+Puncturable Encryption†	Green and Miers	●	●	●	-	-	●	-	●	-	-	-	-	-	-	●	●	●	●	●					
Key Directory+Short Lifetime Keys†	IMKE	●	●	◐	-	●	◐	◐	-	-	-	-	●	●	●	●	●	-	-	-					
+Long-Term Keys†	SIMPP	●	●	◐	-	●	◐	◐	-	-	-	-	●	●	-	●	●	-	-	-					
Authenticated DH†*	TLS-EDH-MA	●	●	●	●	●	◐	◐	●	-	-	-	●	●	◐	●	●	-	-	●					
+Naïve KDF Ratchet*	SCIMP	●	●	●	●	●	●	◐	●	◐	-	-	●	●	◐	◐	◐	-	-	●					
+DH Ratchet†*	OTR	●	●	●	●	●	◐	●	●	◐	◐	-	●	●	◐	◐	◐	-	-	●					
+Double Ratchet†*	Axolotl	●	●	●	●	●	●	●	●	◐	◐	-	●	●	◐	●	◐	-	-	●					
+Double Ratchet+3DH AKE†*	-	●	●	●	●	●	●	●	◐	◐	◐	-	●	●	●	●	◐	-	-	●					
+Double Ratchet+3DH AKE+Prekeys†*	TextSecure	●	●	●	●	●	●	●	-	◐	●	-	●	●	●	◐	◐	●	-	-					
Key Directory+Static DH+Key Transport†	Kikuchi et al.	●	●	-	-	●	◐	◐	-	-	-	-	●	●	-	●	●	●	-	-	-	-	-	●	●
+Authenticated EDH+Group MAC†	GROK	●	●	◐	-	●	◐	◐	●	-	-	-	●	●	-	●	●	●	-	-	-	-	-	●	●
GKA+Signed Messages+Parent IDs†	OldBlue	●	●	●	●	●	◐	◐	●	●	-	-	-	-	-	●	●	◐	-	●	●	●	-	-	-
Authenticated MP DH+Causal Blocks†*	KleeQ	●	●	◐	◐	◐	●	●	●	◐	◐	●	-	●	●	●	●	◐	-	●	●	●	-	-	●
OTR Network+Star Topology†	GOTR (2007)	●	●	-	-	-	◐	●	-	-	-	-	●	●	●	◐	●	◐	-	●	-	-	-	●	●
+Pairwise Topology†		●	●	●	●	●	◐	●	●	-	-	-	●	●	●	◐	●	◐	-	●	●	●	●	●	●
+Pairwise Axolotl+Multicast Encryption*	TextSecure	●	●	●	-	●	●	●	-	●	●	-	●	●	●	●	●	●	-	-	●	●	●	●	●
DGKE+Shutdown Consistency Check†	mpOTR	●	●	●	●	●	◐	◐	●	◐	-	-	-	●	●	●	●	-	-	●	●	●	-	-	-
Circle Keys+Message Consistency Check†	GOTR (2013)	●	●	●	●	●	◐	◐	●	●	●	●	●	●	●	●	-	-	-	◐	●	●	-	●	●

● = provides property; ◐ = partially provides property; - = does not provide property; †has academic publication; *end-user tool available

issues for secure group chat protocols. For example, the mechanisms for inviting participants to chats, kicking users out of sessions, and chat room moderation are all important choices that are influenced by the intended use cases. We do not cover these features here because they are implemented at a higher level than the secure messaging protocol layer.

B. Usability and Adoption

In classic messaging tools, users must only reason about two simple tasks: sending and receiving messages. However, in secure communication, additional tasks might be added. In old secure messaging systems, often based on OpenPGP, users could manually decide whether to encrypt and/or sign messages. Many studies have shown that this caused usability problems [2]–[5], [15]. However, during our evaluation, we found that most recent secure messenger apps secure all messages by default without user interaction. Since all implementations can operate securely once the trust establishment is complete, we omit the user-effort columns in Table II. However, we take other usability and adoption factors, such as resilience properties, into account:

Out-of-Order Resilient: If a message is delayed in transit, but eventually arrives, its contents are accessible upon arrival.

Dropped Message Resilient: Messages can be decrypted without receipt of all previous messages. This is desirable for asynchronous and unreliable network services.

Asynchronous: Messages can be sent securely to disconnected recipients and received upon their next connection.

Multi-Device Support: A user can participate in the conversation using multiple devices at once. Each device must be able to send and receive messages. Ideally, all devices have identical views of the conversation. The devices might use a synchronized long-term key or distinct keys.

No Additional Service: The protocol does not require any infrastructure other than the protocol participants. Specifically, the protocol must not require additional servers for relaying messages or storing any kind of key material.

C. Two-party Chat Evaluation

1) Trusted central servers (baseline): The most basic conversation security features that a secure chat protocol can provide are confidentiality and integrity. This can be easily implemented without adversely affecting usability and adoption properties by using a central server to relay messages and securing connections from clients to the central server using a transport-layer protocol like TLS. This also allows the central server to provide presence information. Since this approach does not negatively affect usability, it is no surprise that this architecture has been adopted by some of the most popular messaging systems today (e.g., Skype, Facebook Chat, Google Hangouts) [47]–[51]. We do not consider these protocols further because they do not meet our stronger end-to-end

definition of *confidentiality* — that messages cannot be read by anyone except the intended recipient(s). We include this approach as a baseline in Table II in order to evaluate the effects of various designs.

Note that the baseline protocols provide all *repudiation* features, since there is no cryptographic proof of any activity. Additionally, these protocols are highly resilient to errors since there are no cryptographic mechanisms that could cause problems when messages are lost. The use of a trusted central server makes *asynchronicity* and *multi-device support* trivial.

2) Static Asymmetric Cryptography: Another simple approach is to use participants' static long-term asymmetric keypairs for signing and encrypting.

OpenPGP and S/MIME are two well-known and widely implemented standards for message protection, mostly used for email but also in XMPP-based tools [47], [52]–[54].

While this approach provides *confidentiality*, message *authentication*, and *integrity*, it causes a loss of all forms of *repudiation*. Additionally, care must be taken to ensure that *destination validation* and *participant consistency* checks are performed. Without destination validation, *surreptitious forwarding* attacks are possible [55]. Without participant consistency, *identity misbinding* attacks might be possible [44]. Defenses against replay attacks should also be included. These considerations are particularly relevant since the OpenPGP and S/MIME standards do not specify how to provide these features, and thus most implementations remain vulnerable to all of these attacks [52], [53].

A second issue with naive asymmetric cryptography is the lack of *forward* or *backward secrecy*. One way to address this issue is to use keys with very short lifetimes (e.g., changing the key every day). Brown et al. propose several extensions to OpenPGP based on this principle [56]. In the most extreme proposal, conversations are started using long-term keys, but each message includes an ephemeral public key to be used for replies. This method provides *forward* and *backward secrecy* for all messages except those used to start a conversation.

From a usability and adoption perspective, static key approaches achieve the same properties as the baseline. Apart from the non-transparent trust establishment, iMessage is a prominent example of how static asymmetric cryptography can achieve end-to-end conversation security with no changes to the user experience. Since the same long-term keys are used for all messages, *message order resilience*, *dropped message resilience*, *asynchronicity*, and *multi-device-support* are provided. *No additional services* are required.

3) FS-IBE: In traditional PKI cryptography, *forward secrecy* is achieved by exchanging ephemeral session keys or by changing keypairs frequently. The use of key agreement protocols makes *asynchronicity* difficult, whereas frequently changing keypairs requires expensive key distribution. Forward Secure Identity Based Encryption (FS-IBE) allows keypairs to be changed frequently with a low distribution cost. Unlike traditional identity-based encryption schemes, the private key generators (PKG) in FS-IBE are operated by the end users and not by a server. Initially, each participant generates a PKG for an identity-based cryptosystem. Participants generate N private keys (SK_i), one for each time period i, by using their PKG, and then immediately destroy the PKG. Each private key SK_i is stored encrypted by the previous private key SK_{i-1} [45], [57]. The participant then distributes the public key of the PKG. Messages sent to the participant are encrypted for the private key corresponding to the current time period. When a time period concludes, the next secret key is decrypted and the expired key is deleted. Thus, if intermediate keys are compromised, the attacker can only retrieve corresponding future private keys; *forward secrecy*, but not *backward secrecy*, is provided. In contrast to generating key pairs for each time period, which requires distribution of N keys, only a single public master key is published; however, the generation still needs to be repeated after all time periods expire.

Canetti, Halevi and Katz were the first to construct a non-interactive forward security scheme based on hierarchical IBE with logarithmic generation and storage costs [57]. In addition, they showed how their scheme can be extended to an unbounded number of periods (i.e., the private keys do not have to be generated in advance), removing the need for *additional services* to distribute new keys at the cost of increasing computational requirements over time. This scheme provides non-interactive *asynchronous forward secrecy* without relying on *additional services*. However, if messages arrive out of order, their corresponding private keys might have already been deleted. As a mitigation, expired keys might be briefly retained, providing partial *out-of-order resilience*.

A similar approach is *puncturable encryption* [58], in which a recipient can update their private key to prevent future decryption of a specific message identified by an (arbitrary) tag. Computational costs and storage costs increase over time for both systems, introducing *scalability* concerns. To our knowledge, neither scheme has been deployed and they thus merit further development.

4) Authenticated Diffie-Hellman: Many conversation security schemes make use of an authenticated Diffie-Hellman (DH) key exchange to initialize the conversation. In an authenticated key exchange (AKE) such as authenticated DH, the participants generate an ephemeral session key and authenticate the exchange using their long-term keys. The resulting session key is used to derive symmetric encryption and MAC keys, which then protect messages using an encrypt-then-MAC approach. This basic design provides *confidentiality*, *integrity*, and *authentication*. TLS with an ephemeral DH cipher suite and mutual authentication (TLS-EDH-MA) is a well-known example of this approach. Note that further protections are required during key exchange to protect against *identity misbinding* attacks violating *participant consistency*, such as those provided by the SIGMA protocol [29], [44].

The use of ephemeral session keys provides *forward* and *backward secrecy* between conversations. *Message unlinkability* and *message repudiation* are provided since messages are authenticated with shared MAC keys rather than being signed with long-term keys. At a minimum, messages can be forged by any chat participants. Some protocols, such as

OTR, take additional measures, such as publication of MAC keys and the use of malleable encryption, to expand the set of possible message forgers [59]. If the participants simply sign all AKE parameters, then this approach does not provide *participation repudiation*. However, if participants only sign their own ephemeral keys, these signatures can be reused by their conversation partners in forged transcripts. OTR uses this approach to obtain partial *participation repudiation* due to a limited set of possible forgers.

Once the AKE has been performed, the encrypt-then-MAC approach allows messages to be exchanged asynchronously with *out-of-order* and *dropped message resilience*. However, since a traditional AKE requires a complete handshake before actual messages can be encrypted, this basic approach requires *synchronicity* during conversation initialization. Additionally, since key agreements can only be performed with connected devices, there is no trivial *multi-device support*.

5) Key Evolution: A desirable property is *forward secrecy* for individual messages rather than for entire conversations. This is especially useful in settings where conversations can last for the lifetime of a device. To achieve this, the session key from the initial key agreement can be evolved over time through the use of a *session key ratchet* [43]. A simple approach is to use key derivation functions (KDFs) to compute future message keys from past keys. This naive approach, as used in SCIMP [60], provides *forward secrecy*. However, it does not provide *backward secrecy* within conversations; if a key is compromised, all future keys can be derived using the KDF. *Speaker consistency* is partially obtained since messages cannot be surreptitiously dropped by an adversary without also dropping all future messages (otherwise, recipients would not be able to decrypt succeeding messages). If messages are dropped or arrive out of order, the recipient will notice since the messages are encrypted with an unexpected key. To handle this, the recipient must store expired keys so that delayed or re-transmitted messages can still be decrypted, leaving a larger window of compromise than necessary. Thus, *out-of-order* and *dropped message resilience* are only partially provided.

6) Diffie-Hellman Ratchet: A different ratcheting approach, introduced by OTR, is to attach new DH contributions to messages [59]. With each sent message, the sender advertises a new DH value. Message keys are then computed from the latest acknowledged DH values. This design introduces *backward secrecy* within conversations since a compromised key will regularly be replaced with new key material. *Causality preservation* is partially achieved since messages implicitly reference their causal predecessors based on which keys they use. The same level of *speaker consistency* as the naive KDF solution can be provided by adding a per-speaker monotonic counter to messages. A disadvantage of the DH ratchet is that session keys might not be renewed for every message (i.e., *forward secrecy* is only partially provided). Like the KDF-based ratchet, the DH ratchet lacks *out-of-order resilience*; if a message arrives after a newly advertised key is accepted, then the necessary decryption key was already deleted.

7) Double-Ratchet (Axolotl): To improve the forward secrecy of a DH ratchet, both ratchet approaches can be combined: session keys produced by DH ratchets are used to seed per-speaker KDF ratchets. Messages are then encrypted using keys produced by the KDF ratchets, frequently refreshed by the DH ratchet on message responses. The resulting *double ratchet*, as implemented by Axolotl [61], provides *forward secrecy* across messages due to the KDF ratchets, but also *backward secrecy* since compromised KDF keys will eventually be replaced by new seeds. To achieve *out-of-order resilience*, the Axolotl ratchet makes use of a second derivation function within its KDF ratchets. While the KDF ratchets are advanced normally, the KDF keys are passed through a second distinct derivation function before being used for encryption. If a message arrives out of order, the KDF ratchet can be moved forward while temporarily storing the old derived message key; if this key is compromised, it does not impact *forward secrecy*. Despite these improvements, the double ratchet still requires *synchronicity* for the initial AKE.

8) 3-DH Handshake: A triple DH (3-DH) handshake is a different AKE scheme that provides stronger *participation repudiation*. Assuming that Alice and Bob both have long-term DH keys g^a and g^b and ephemeral keys g^{a_e} and g^{b_e}, the 3-DH shared secret s is computed as $s = \text{KDF}(\text{DH}(g^{a_e}, g^{b_e})||\text{DH}(g^a, g^{b_e})||\text{DH}(g^{a_e}, g^b))$ [61]. If a secure key derivation function is used, a MitM attacker must either know a and a_e, or b and b_e. Kudla et al. have shown that the 3-DH key exchange provides the same *authentication* level as achieved with the authenticated versions of DH key agreements [62]. 3-DH achieves full *participation repudiation* since anybody is able to forge a transcript between any two parties by generating both a_e and b_e and performing DH key exchanges with a and b. Since the secret is partially derived from the long-term public keys, 3-DH also provides *participant consistency* without the need to explicitly exchange identities after a secure channel has been established. Unfortunately, this also causes a partial loss of *anonymity preservation* since long-term public keys are always observable during the initial key agreement (although future exchanges can be protected by using past secrets to encrypt these identities).

9) Prekeys: While a *double ratchet* does not provide *asynchronicity* by itself, it can be combined with a *prekey* scheme to create an asynchronous version of the protocol. Prekeys are one-time ephemeral public DH contributions that have been uploaded in advance to a central server. This allows clients to complete a DH key exchange with a message recipient by requesting their next prekey from the server. When combined with a 3-DH exchange, this is sufficient to complete an asynchronous AKE as part of the first message.

TextSecure [63] is a popular Android app that combines Axolotl, prekeys, and 3-DH to provide an *asynchronous* user experience while sacrificing the *no additional service* property. It has gained considerable attention recently after being incorporated into WhatsApp [64], [65]. Assuming Axolotl is used on two devices, the key material can evolve independently for each device. However, if one of those devices remains

offline for a long time, a key compromise on that device is problematic: if the device can use its outdated keys to read messages that were sent when it was offline, then this compromise defeats *forward secrecy*; if the device cannot read the old messages, then the protocol does not achieve complete *multi-device support*. Deciding how long a device may be offline before it can no longer read buffered messages is an adoption consideration requiring further study of user behavior.

D. Group Chat Evaluation

1) OTR for groups: Several protocols have been proposed to achieve OTR-like *repudiation* properties for group conversations. The TextSecure protocol can be naturally extended to groups by sending messages to each recipient using the two-party TextSecure protocol [66]. Multicast encryption is used for performance: a single encrypted message is sent to a central server for relaying to recipients while the decryption key for the message is sent pairwise using TextSecure. This design does not provide any guarantees of *participant consistency*, but it inherits the *asynchronicity* of the two-party TextSecure protocol. *Speaker consistency* and *causality preservation* are achieved by attaching preceding message identifiers to messages. A message identifier is a hash of the sender, the list of preceding identifiers, and the message contents.

A *repudiable* group chat scheme can also be designed by utilizing a deniable group key exchange (DGKE) protocol, as in the mpOTR protocol [67], [68]. When completed, the DGKE provides each participant with a shared secret group key and individual ephemeral signing keys. This information is authenticated with long-term keys in a manner providing *participation repudiation* while still *authenticating* participants — participants receive proof of each other's identities, but this proof cannot be used to convince outsiders. Messages are encrypted with the shared group key and signed with the ephemeral keys. The ephemeral signatures provide proof of authorship to others in the group but, because outsiders cannot be certain that these ephemeral signing keys correspond to specific long-term keys, *message repudiation* is preserved. However, since all messages from an individual are signed with the same (ephemeral) key, the protocol does not have *message unlinkability*. To provide *speaker consistency*, a check is performed on shutdown whereby hashes of messages sent by each participant are exchanged. If this check fails, messages must be individually compared to uncover discrepancies. In this scheme, *subgroup messaging* is not possible since all messages share a single encryption key. The group is also not *expandable* or *contractible* without performing a new DGKE.

A completely different approach is taken by the GOTR protocol released in 2013 (we write the year to distinguish it from a different protocol with the same name from 2007). GOTR (2013) [69] is built using a "hot-pluggable" group key agreement (GKA) protocol, allowing members to join and drop out of the conversation with little overhead. This system involves the use of "circle keys": sets of public keys having the property that a shared secret key can be computed by anyone with a private key matching a public key in the set. The key exchange mechanism in this protocol is relatively complex; due to space constraints, we refer the interested reader to the original publication for details [69]. Pairwise secure channels are set up between participants to send consistency check messages. These consistency channels have the effect of providing *global transcript order*, but all participants are required to be online to receive messages. The system otherwise provides features similar to mpOTR but with *flexible group membership* and *message unlinkability*.

E. Other Approaches

Due to space constraints, there are several approaches in Table II for which we omit detailed evaluations. Protocols such as IMKE [70] and SIMPP [71]–[73] use a central server to exchange ephemeral keys. GROK [74] and the protocol of Kikuchi et al. [75] are early attempts to support secure group chat, and involve a conversation leader transmitting an ephemeral group key to others with the help of a central server. Group chat protocols such as OldBlue [76] and KleeQ [77] provide mechanisms to preserve causality of messages sent over unreliable networks. These protocols are evaluated in the extended paper [1]. Other designs include pairwise OTR connections, or a trusted server that is connected to using OTR, as in the GOTR (2007) scheme [78]. Appendix C discusses these designs. The recently proposed (n+1)sec protocol [79] provides a DGKE and checks for transcript consistency.

F. Discussion

Similar to our study of trust establishment, Table II makes immediately clear that no conversation security protocol provides all desired properties. Since most of the properties in the table are not mutually exclusive, however, there is significant room for improvement by combining protocol designs and this should be seen as a tangible and important call to action for the research community.

Sadly, the most widely adopted solutions also have the worst security and privacy properties, with most non-security-focused applications providing only basic static asymmetric cryptography. This does not appear to be due to the usability drawbacks of the more secure protocols: once the trust establishment has been done, all of the conversation security approaches we studied can be automated without any additional effort for the user. An exception is enabling *asynchronous* communication while still providing *forward* and *backward secrecy*; the only solution for this problem that appears to have any significant deployment in practice is the prekeys approach implemented by TextSecure. This requires relatively complicated infrastructure compared to a simple key server, introduces problems for *multi-device support*, and is prone to denial-of-service attacks if it is used in anonymous communication. This approach is poorly studied in the academic literature. The FS-IBE scheme discussed in Section IV-C3 promises to resolve the issues of server complexity and denial of service, but introduces new challenges such as scalability and performance issues [57]. Unlike prekeys (Section IV-C9),

this scheme has received a considerable amount of follow-up research and academic citations, but we are unaware of any practical tool implementing it. In addition, a time-window based FS-IBE scheme requires holding the ephemeral keys for a certain amount of time to allow decryption of delayed messages. Improving the practicality of FS-IBE and puncturable encryption schemes warrants further research.

Another outstanding concern that limits adoption of secure conversation security protocols is the limited support for multiple devices. Despite a vast number of users owning multiple devices, only the most insecure protocols support this property without requiring users to perform pairing procedures. Device pairing has proved extremely difficult for users in practice [80], [81] and allowing users to register multiple devices with distinct keys is a major usability improvement.

When it comes to group chat properties, we can identify several areas for improvement in Table II. Classic protocols often do not provide *participant consistency* or *destination validation*, making them potentially vulnerable to surreptitious forwarding or identity misbinding attacks. However, these are sometimes addressed in concrete implementations. The double ratchet used in Axolotl improves *forward secrecy* with low cost in performance, implementation complexity, and resilience, but it has not yet been thoroughly evaluated in an academic context. Additionally, decentralized group chat systems inherently permit a participant to send different messages to different people. Due to network conditions, users can also end up observing significantly different transcripts. Despite these intrinsic weaknesses, surprisingly few protocols explicitly consider *speaker consistency* or *causality preservation*.

Existing solutions achieve mixed results concerning *repudiation*. Only the OTR-like protocols, namely the two-party protocols based on authenticated DH key exchanges and the OTR-like group protocols, offer both *message* and *participant repudiation* while also providing *authentication*.

There are also additional adoption constraints imposed by many modern secure group chat protocols. Group protocols often choose to employ either a trusted participant or an additional service to improve protocol performance, which can lead to security concerns or introduce additional costs for deployment. Very few group protocols support *subgroup messaging*, and just as few support changing group membership after the conversation has started without incurring the substantial costs of a new protocol run. Additionally, many proposed designs require synchronicity in order to simplify their protocols, which largely precludes their use on current mobile devices.

V. Transport Privacy

The transport privacy layer defines how messages are exchanged, with the goal of hiding message metadata such as the sender, receiver, and conversation to which the message belongs. Some transport privacy architectures impose topological structures on the conversation security layer, while others merely add privacy to data links between entities. The transport privacy schemes may also be used for privacy-preserving contact discovery. In this section, we compare approaches for transport privacy in terms of the privacy features that they provide, as well as usability concerns and other factors that limit their adoption. Table III compares the various schemes.

A. Privacy Features

We make the distinction between *chat messages*, which are the user-generated payloads for the messaging protocol to exchange, and *protocol messages*, which are the underlying data transmissions dictated by the upper protocol layers. We define following privacy properties:

Sender Anonymity: When a chat message is received, no non-global entities except for the sender can determine which entity produced the message.

Recipient Anonymity: No non-global entities except the receiver of a chat message know which entity received it.

Participation Anonymity: No non-global entities except the conversation participants can discover which set of network nodes are engaged in a conversation.

Unlinkability: No non-global entities except the conversation participants can discover that two protocol messages belong to the same conversation.

Global Adversary Resistant: Global adversaries cannot break the anonymity of the protocol.

B. Usability Properties

Contact Discovery: The system provides a mechanism for discovering contact information.

No Message Delays: No long message delays are incurred.

No Message Drops: Dropped messages are retransmitted.

Easy Initialization: The user does not need to perform any significant tasks before starting to communicate.

No Fees Required: The scheme does not require monetary fees to be used.

C. Adoption Properties

Topology Independent: No network topology is imposed on the conversation security or trust establishment schemes.

No Additional Service: The architecture does not depend on availability of any infrastructure beyond the chat participants.

Spam/Flood Resistant: The availability of the system is resistant to denial-of-service attacks and bulk messaging.

Low Storage Consumption: The system does not require a large amount of storage capacity for any entity.

Low Bandwidth: The system does not require a large amount of bandwidth usage for any entity.

Low Computation: The system does not require a large amount of processing power for any entity.

Asynchronous: Messages sent to recipients who are offline will be delivered when the recipient reconnects, even if the sender has since disconnected.

Scalable: The amount of resources required to maintain system availability scales linearly with the number of users.

TABLE III
TRANSPORT PRIVACY SCHEMES. EVERY PRIVACY-ENHANCING APPROACH CARRIES USABILITY AND/OR ADOPTION COSTS.

Scheme	Example	Privacy					Usability					Adoption						
		Sender Anonymity	Recipient Anonymity	Particip. Anonymity	Unlinkability	Global Adv. Resistant	Contact Discovery	No Message Delays	No Message Drops	Easy Initialization	No Fees Required	Topology Independent	No Additional Service	Spam/Flood Resistant	Low Storage	Low Bandwidth	Low Computation	Asynchronous	Scalable
Store-and-Forward†*	Email/XMPP	-	-	-	-	-	●	◐	●	●	●	●	-	-	●	●	●	●	●
+DHT Lookup†*	Kademlia	◐	◐	-	-	-	●	◐	●	●	●	●	●	◐	●	◐	●	●	●
Onion Routing+Message Padding†*	Tor	●	-	●	●	-	-	◐	●	●	●	●	◐	-	●	●	●	-	●
+Hidden Services*	Ricochet	●	●	●	◐	-	-	◐	●	●	●	●	◐	-	●	●	●	-	●
+Inbox Servers†	-	●	-	●	●	-	-	◐	●	●	●	●	-	-	●	●	●	●	●
+Random Delays†*	Mixminion	●	-	●	●	◐	-	-	●	●	●	●	-	-	◐	●	●	●	●
+Hidden Services+Delays+Inboxes+ZKGP*	Pond	●	-	●	●	◐	-	-	●	●	●	●	-	●	◐	●	●	●	●
DC-Nets†*	-	●	●	-	-	●	-	-	●	●	●	-	●	-	●	●	●	-	-
+Silent Rounds†	Anonycaster	●	●	-	-	●	-	-	●	●	●	-	●	◐	●	●	●	-	-
+Shuffle-Based DC-Net+Leader†	Dissent	●	●	-	-	●	-	-	●	●	●	-	●	●	●	●	●	-	-
+Shuffle-Based DC-Net+Anytrust Servers†	Verdict	●	●	-	-	●	-	-	●	●	●	-	-	●	●	●	●	-	◐
Message Broadcast†	-	-	●	●	●	●	●	●	●	●	●	●	●	-	-	-	◐	-	-
+Blockchain	-	◐	●	●	●	●	●	-	-	●	-	●	◐	●	-	-	-	●	-
PIR*	Pynchon Gate	-	●	●	●	●	●	-	●	◐	●	●	-	-	-	◐	◐	●	◐

● = provides property; ◐ = partially provides property; - = does not provide property; †has academic publication; *end-user tool available

D. Evaluation

1) Store-and-Forward (baseline): To evaluate the effectiveness and costs of different transport privacy architectures in Table III, we compare the solutions to a baseline. For the baseline protocol, we assume a simple store-and-forward messaging protocol. This method is employed by email and text messaging, causing minor message delays and storage requirements for intermediate servers. Since email headers contain sender and recipient information, a simple store-and-forward mechanism does not provide any privacy properties.

2) Onion Routing: Onion routing is a method for communicating through multiple proxy servers that complicates end-to-end message tracing [82]. In onion routing, senders send messages wrapped in multiple layers of encryption through preselected paths of proxy servers. These servers unwrap layers of encryption until the original message is exposed, at which point it is relayed to the final destination. Each node in the path only knows the immediate predecessor and successor in the path. The routing process adds some latency to messages, but otherwise retains the baseline usability features. An onion routing protocol, such as the widely used Tor protocol [83], provides *sender anonymity*, *participant anonymity*, and *unlinkability* against network attackers with limited scope. Tor also includes an extension called *hidden services* that provides *recipient anonymity*.

Global network adversaries are still able to break the anonymity properties of simple onion routing designs by performing statistical analysis incorporating features such as content size, transmission directions, counts, and timing, among others. The success of such an adversary can be limited by individually eliminating these features. Protection can be added, for example, by introducing random delays to transmissions. The longer the allowed delays, the less statistical power is available to the adversary. Of course, this imposes potentially long *message delays* and *additional storage requirements* for relays, making it unusable for synchronous instant messaging. Mixminion is an implementation using this technique [84].

Unfortunately, random delays do not completely defeat global adversaries. The only way to do so is to make transmission indistinguishable from no transmission (e.g., by saturating the bandwidth of all connections). However, in practice, this is likely infeasible. Adoption of onion routing is limited by the requirement to establish a large network of nodes to provide a sufficient anonymity set and cover traffic.

To provide *asynchronous* communication support, store-and-forward servers can be incorporated into the onion routing model. Each user is associated with a Tor hidden service that remains online. To send a message, the sender constructs a circuit to the recipient's server and transmits the message. Users periodically poll their own servers to determine if any messages are queued. Ricochet is an example of this approach.

Pond uses this design for its transmission architecture [85] but adds random delays between connections, all of which transmit the same amount of data, to weaken statistical analysis by network adversaries. This design requires *storage commitments* by servers and also introduces very *high latency*.

Without additional protections, this scheme is also highly vulnerable to denial-of-service attacks because connection delays and fixed transmission sizes artificially limit bandwidth to very low levels. Pond addresses this by requiring users to maintain group lists secured by zero-knowledge-group-proof schemes (ZKGP). This way, recipients can upload contact lists without revealing their contacts. Simultaneously, senders can authenticate by providing zero-knowledge proofs that they are in this list. The BBS signature scheme [86] is currently used by Pond to achieve this.

3) DC-nets: Dining Cryptographer networks (DC-nets) are anonymity systems that are often compared to onion routing

schemes. DC-nets are group protocols that execute in rounds. At the start of each round, each participant either submits a secret message or no message. At the end of the round, all participants receive the xor of all secret messages submitted, without knowing which message was submitted by which participants. In this way, DC-nets provide *sender anonymity* while also achieving *global adversary resilience* — no statistical analysis can reveal the sender of a message. *Recipient anonymity* can be achieved by using the protocol to publish an ephemeral public key. Messages encrypted with this key are then sent and, since the owner of the matching private key is unknown, the participant able to decrypt the messages cannot be determined. Since messages are sent in rounds, DC-nets add message latency and do not support *asynchronous* communication; dropped messages prevent the protocol from advancing. Messages are easily *linked* by observing which network nodes participate in a round. Additionally, DC-nets have limited *scalability* due to requiring pairwise communication.

The basic DC-net design has a problem with collisions: if two parties submit a message in the same round, the result will be corrupted. A malicious participant can exploit this to perform an anonymous denial-of-service attack by submitting garbled messages each round. Worse still, an active network attacker can also perform this attack by perturbing transmitted bits. There are several approaches to mitigate this problem. Anonycaster [87] adds pseudorandomly determined "silent rounds" where all members know that no message should be contributed. Receipt of a message during a silent round indicates a denial-of-service attack by an active network attacker. However, malicious participants can still launch attacks by sending garbled messages only during non-silent rounds.

Dissent [88]–[90] and Verdict [91] take a different approach by constructing a DC-net system through the use of a verifiable shuffle and bulk transfer protocol. Shuffle-based DC-nets can include a blame protocol to pinpoint the entity that caused a round to fail. Dissent appoints one participant as a leader to manage round timing, the blame protocol, and exclusion of disconnected members from rounds, thereby restoring support for *asynchronicity*. Verdict uses an alternative approach where the DC-net protocol is executed by a set of central servers that clients connect to, providing greater *scalability* and maintaining security as long as any one server is honest.

While DC-nets are primarily a transport privacy mechanism, they are distinguished from other schemes by their use of rounds and the fact that every network node is also a participant in the conversation. When using DC-nets to transmit higher-level conversation security protocols, it is important for designers to consider how these properties affect the overall security of the scheme (e.g., the use of synchronous rounds creates a *global transcript*, and the details of the DC-net key exchanges may cause a loss of *participation repudiation*).

4) Broadcast Systems: There is a simple approach to providing recipient anonymity against all attackers, including global adversaries: distributing messages to everyone. This approach provides *recipient anonymity*, *participation anonymity*, and *unlinkability* against all network attackers. It also provides a natural way to *discover contacts* because requests for contact data can be sent to the correct entity without knowledge of any addressing information. However, there are some serious downsides that hinder adoption: broadcasting a message to everyone in the network requires high bandwidth, there is no support for *asynchronicity*, and it has extreme *scalability* issues. Additionally, it is easy to attack the availability of the network through flooding. Bitmessage [92], a broadcast-based transport system, either requires a proof of work or monetary fees to send messages in order to limit spam, adding *computation requirements* and *message delays* as represented by the *blockchains* row in Table III. It is also possible to alleviate *scalability* problems by clustering users into smaller broadcast groups, at the cost of reduced anonymity set sizes.

5) PIR: Private Information Retrieval (PIR) protocols allow a user to query a database on a server without enabling the server to determine what information was retrieved. These systems, such as the Pynchon Gate [93], can be used to store databases of message inboxes, as well as databases of contact information. *Recipient anonymity* is provided because, while the server knows the network node that is connecting to it, the server cannot associate incoming connections with protocol messages that they retrieve. For the same reason, the protocols offer *participation anonymity* and *unlinkability*. By default, there is no mechanism for providing *sender anonymity*. These systems are naturally *asynchronous*, but they result in *high latency* because inboxes must be polled. The servers also incur a *high storage cost* and are vulnerable to flooding attacks. PIR implementations can be divided into computational schemes, which rely on computational limitations of the server, information-theoretic schemes, which rely on non-collusion of servers, and hybrid schemes that combine properties of both. PIR implementations differ in their bandwidth, computation, and initialization costs, as well as their scalability. PIR is not widely adopted in practice because one or more of these costs is usually prohibitively high.

E. Discussion

If messages are secured end-to-end, leaving only identifiers for anonymous inboxes in the unencrypted header, then metadata is easily hidden from service operators. Assuming that each message is sent using new channels, an adversary is not able to link single messages to conversations. However, such schemes introduce adoption and usability issues; they are prone to spam, flooding, and denial-of-service attacks, or require expensive operations such as zero-knowledge authentication posing barriers to adoption. Worse still, hiding metadata from a global adversary in these schemes necessitates serious usability problems such as long delays.

In contrast, decentralized schemes either exhibit synchronicity issues or have serious scalability problems. Most decentralized projects, especially BitTorrent-based approaches, lack detailed documentation that is required for complete evaluation. Some tools claiming to hide metadata only do so in the absence of global network adversaries, which recent surveillance revelations suggest may exist.

Broadcast-based schemes can achieve the best privacy properties, but exhibit serious usability issues, such as lost or delayed messages, in addition to apparently intractable scalability issues. Finally, care must be taken when selecting a conversation security scheme to avoid leaking cryptographic material or identifiers that might lead to deanonymization.

VI. Concluding Remarks

The vast majority of the world's electronic communication still runs over legacy protocols like SMTP, SMS/GSM, and centralized messengers, none of which were designed with end-to-end security in mind. We encourage the research community to view the high-profile NSA revelations in the United States as a golden opportunity to encourage the adoption of secure systems in their place. As the old adage goes: "never let a crisis go to waste."

Unfortunately, while we have seen considerable progress in practical tools over the past two years, there is little evidence suggesting that academic research on secure messaging has dramatically increased. This is unfortunate for two reasons: First, many interesting problems of practical importance remain unresolved. In particular, apparent practical deployment constraints, including limitations for asynchronous communication, multiple independent devices, and zero user effort, are not fully appreciated in most published research papers. Second, many theoretically *solved* problems are not considered in practice, whether because developers are unaware of their existence, or because they cannot immediately translate the cryptographic publications into working systems.

Our effort to systematize existing knowledge on secure messaging suggests three major problems must be resolved: *trust establishment*, *conversation security* and *transport privacy*. The schemes can largely be chosen independently, yielding a vast design space for secure messaging systems. Yet we also caution against a proliferation of a-la-carte systems for specific niches. The main purpose of communication networks is to interact with others and there is considerable value in having a small number of popular protocols that connect a large number of users. Currently, everyone uses email and hence many people fall back to this method despite its insecurity.

We also note that, disappointingly, most of the exciting progress being made right now is by protocols that are either completely proprietary (e.g., Apple iMessage) or are open-source but lack a rigorously specified protocol to facilitate interoperable implementations (e.g., TextSecure). An open standard for secure messaging, combining the most promising features identified by our survey, would be of immense value.

Inevitably, trade-offs have to be made. We conclude that secure approaches in trust establishment perform poorly in usability and adoption, while more usable approaches lack strong security guarantees. We consider the most promising approach for trust establishment to be a combination of central key directories, transparency logs to ensure global consistency of the key directory's entries, and a variety of options for security-conscious users to verify keys out of band to put pressure on the key directory to remain honest.

Our observations on the conversation security layer suggest that asynchronous environments and limited multi-device support are not fully resolved. For two-party conversation security, per-message ratcheting with resilience for out-of-order messages combined with deniable key exchange protocols, as implemented in Axolotl, can be employed today at the cost of additional implementation complexity with no significant impact on user experience. The situation is less clear for secure group conversations; while no approach is a clear answer, the TextSecure group protocol provides pragmatic security considerations while remaining practical. It may be possible to achieve other desirable properties, such as participant consistency and anonymity preservation, by incorporating techniques from the other systems. It remains unclear exactly what consistency properties are required to match users' expectations and usability research is sorely needed to guide future protocol design. Finally, transport privacy remains a challenging problem. No suggested approaches managed to provide strong transport privacy properties against global adversaries while also remaining practical.

We consider this systematization to be a useful assessment of published research and deployment experience. We have uncovered many open challenges and interesting problems to be solved by the research community. The active development of secure messaging tools offers a huge potential to provide real-world benefits to millions; we hope this paper can serve as an inspiration and a basis for this important goal.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments, and Trevor Perrin and Henry Corrigan-Gibbs for their discussion and excellent feedback. Joseph Bonneau is supported by a Secure Usability Fellowship from the Open Technology Fund and Simply Secure. We gratefully acknowledge the support of NSERC and the Ontario Research Fund.

References

[1] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith, "SoK: Secure Messaging," CACR, Tech. Rep. 2015-02, 2015. [Online]. Available: http://cacr.uwaterloo.ca/techreports/2015/cacr2015-02.pdf

[2] A. Whitten and J. D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," in *Security Symposium*. USENIX, 1999.

[3] S. L. Garfinkel and R. C. Miller, "Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express," in *Symposium on Usable Privacy and Security*. ACM, 2005, pp. 13–24.

[4] S. L. Garfinkel, D. Margrave, J. I. Schiller, E. Nordlander, and R. C. Miller, "How to Make Secure Email Easier To Use," in *SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2005, pp. 701–710.

[5] K. Renaud, M. Volkamer, and A. Renkema-Padmos, "Why Doesn't Jane Protect Her Privacy?" in *Privacy Enhancing Technologies*. Springer, 2014, pp. 244–262.

[6] M. Madden. (2014, Nov) Public Perceptions of Privacy and Security in the Post-Snowden Era. [Online]. Available: http://www.pewinternet.org/2014/11/12/public-privacy-perceptions/

[7] GoldBug Project. GoldBug - Secure Instant Messenger. [Online]. Available: http://goldbug.sourceforge.net/

[8] Telegram. Telegram Messenger. [Online]. Available: https://telegram.org/

[9] Wickr. Wickr – Top Secret Messenger. [Online]. Available: https://wickr.com/

[10] Confide. Confide - Your Off-the-Record Messenger. [Online]. Available: https://getconfide.com/

[11] Electronic Frontier Foundation. Secure Messaging Scorecard. [Online]. Available: https://www.eff.org/secure-messaging-scorecard

[12] S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer, "The Emperor's New Security Indicators: An evaluation of website authentication and the effect of role playing on usability studies," in *Symposium on Security and Privacy*. IEEE, 2007, pp. 51–65.

[13] R. Stedman, K. Yoshida, and I. Goldberg, "A User Study of Off-the-Record Messaging," in *Symposium on Usable Privacy and Security*. ACM, 2008, pp. 95–104.

[14] S. Clark, T. Goodspeed, P. Metzger, Z. Wasserman, K. Xu, and M. Blaze, "Why (Special Agent) Johnny (Still) Can't Encrypt: A Security Analysis of the APCO Project 25 Two-way Radio System," in *Security Symposium*. USENIX, 2011.

[15] S. Fahl, M. Harbach, T. Muders, M. Smith, and U. Sander, "Helping Johnny 2.0 to Encrypt His Facebook Conversations," in *Symposium on Usable Privacy and Security*. ACM, 2012.

[16] S. Ruoti, N. Kim, B. Burgon, T. van der Horst, and K. Seamons, "Confused Johnny: When Automatic Encryption Leads to Confusion and Mistakes," in *Symposium on Usable Privacy and Security*. ACM, 2013.

[17] J. Nielsen, "Finding Usability Problems Through Heuristic Evaluation," in *SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1992, pp. 373–380.

[18] ——, "Usability inspection methods," in *Conference Companion on Human Factors in Computing Systems*. ACM, 1994, pp. 413–414.

[19] B. E. John and M. M. Mashyna, "Evaluating a Multimedia Authoring Tool with Cognitive Walkthrough and Think-Aloud User Studies," DTIC, Tech. Rep., 1995.

[20] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes," in *Symposium on Security and Privacy*. IEEE, 2012. [Online]. Available: http://www.jbonneau.com/doc/BHOS12-IEEESP-quest_to_replace_passwords.pdf

[21] J. Clark and P. C. van Oorschot, "SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements," in *Symposium on Security and Privacy*. IEEE, 2013, pp. 511–525.

[22] D. Wendlandt, D. G. Andersen, and A. Perrig, "Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing," in *Annual Technical Conference*. USENIX, 2008, pp. 321–334.

[23] P. Zimmermann, A. Johnston, and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP," RFC 6189 (Informational), Internet Engineering Task Force, 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6189.txt

[24] E. A. Blossom, "The VP1 Protocol for Voice Privacy Devices Version 1.2," Communication Security Corporation, 1999.

[25] P. Gupta and V. Shmatikov, "Security Analysis of Voice-over-IP Protocols," in *Computer Security Foundations Symposium*. IEEE, 2007, pp. 49–63.

[26] M. Petraschek, T. Hoeher, O. Jung, H. Hlavacs, and W. N. Gansterer, "Security and Usability Aspects of Man-in-the-Middle Attacks on ZRTP," *Journal of Universal Computer Science*, vol. 14, no. 5, pp. 673–692, 2008.

[27] M. Shirvanian and N. Saxena, "Wiretapping via Mimicry: Short Voice Imitation Man-in-the-Middle Attacks on Crypto Phones," in *Conference on Computer and Communications Security*. ACM, 2014, pp. 868–879.

[28] M. Jakobsson and M. Yung, "Proving Without Knowing: On Oblivious, Agnostic and Blindfolded Provers," in *Advances in Cryptology–CRYPTO*. Springer, 1996, pp. 186–200.

[29] C. Alexander and I. Goldberg, "Improved User Authentication in Off-The-Record Messaging," in *Workshop on Privacy in the Electronic Society*. ACM, 2007, pp. 41–47.

[30] F. Boudot, B. Schoenmakers, and J. Traoré, "A fair and efficient solution to the socialist millionaires' problem," *Discrete Applied Mathematics*, vol. 111, no. 1, pp. 23–36, 2001.

[31] M. Farb, Y.-H. Lin, T. H.-J. Kim, J. McCune, and A. Perrig, "SafeSlinger: Easy-to-Use and Secure Public-Key Exchange," in *International Conference on Mobile Computing & Networking*. ACM, 2013, pp. 417–428.

[32] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov, "The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software," in *Conference on Computer and Communications Security*. ACM, 2012, pp. 38–49.

[33] M. Marlinspike, "More tricks for defeating SSL in practice," in *Black Hat USA*, 2009.

[34] VASCO. (2011, Sep) DigiNotar reports security incident. [Online]. Available: https://www.vasco.com/company/about_vasco/press_room/news_archive/2011/news_diginotar_reports_security_incident.aspx

[35] A. Langley. (2013) Enhancing digital certificate security. [Online]. Available: http://googleonlinesecurity.blogspot.de/2013/01/enhancing-digital-certificate-security.html

[36] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," RFC 6962 (Experimental), Internet Engineering Task Force, 2013. [Online]. Available: http://tools.ietf.org/rfc/rfc6962.txt

[37] Google. End-To-End. [Online]. Available: https://github.com/google/end-to-end

[38] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor, "Crying Wolf: An Empirical Study of SSL Warning Effectiveness," in *Security Symposium*. USENIX, 2009, pp. 399–416.

[39] M. D. Ryan, "Enhanced Certificate Transparency and End-to-end Encrypted Mail," in *Network and Distributed System Security Symposium*. Internet Society, 2014.

[40] M. S. Melara, A. Blankstein, J. Bonneau, M. J. Freedman, and E. W. Felten, "CONIKS: A Privacy-Preserving Consistent Key Service for Secure End-to-End Communication," Cryptology ePrint Archive Report 2014/1004, 2014. [Online]. Available: https://eprint.iacr.org/2014/1004

[41] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008, self-published.

[42] N. Project. Namecoin. [Online]. Available: https://namecoin.info/

[43] O. W. Systems. Advanced cryptographic ratcheting. [Online]. Available: https://whispersystems.org/blog/advanced-ratcheting/

[44] W. Diffie, P. C. van Oorschot, and M. J. Wiener, "Authentication and Authenticated Key Exchanges," *Designs, Codes and Cryptography*, vol. 2, no. 2, pp. 107–125, 1992.

[45] R. Anderson, "Two Remarks on Public Key Cryptology," 1997, available from https://www.cl.cam.ac.uk/users/rja14.

[46] R. Shirey, "Internet Security Glossary," RFC 2828 (Informational), Internet Engineering Task Force, 2000, obsoleted by RFC 4949. [Online]. Available: http://tools.ietf.org/rfc/rfc2828.txt

[47] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," RFC 6120 (Proposed Standard), Internet Engineering Task Force, 2011. [Online]. Available: http://tools.ietf.org/rfc/rfc6120.txt

[48] S. Schrittwieser, P. Frühwirt, P. Kieseberg, M. Leithner, M. Mulazzani, M. Huber, and E. R. Weippl, "Guess Who's Texting You? Evaluating the Security of Smartphone Messaging Applications," in *Network and Distributed System Security Symposium*. Internet Society, 2012.

[49] Microsoft. (2014) Does Skype use encryption? [Online]. Available: https://support.skype.com/en/faq/FA31/does-skype-use-encryption

[50] Google. (2014) Google Hangouts - Video Conferencing & Meeting for Business. [Online]. Available: https://www.google.com/work/apps/business/products/hangouts/

[51] Facebook. (2014) Facebook Help Center. [Online]. Available: https://www.facebook.com/help/

[52] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, "OpenPGP Message Format," RFC 4880 (Proposed Standard), Internet Engineering Task Force, 1999, updated by RFC 5581. [Online]. Available: http://tools.ietf.org/rfc/rfc4880.txt

[53] B. Ramsdell and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification," RFC 5751 (Proposed Standard), Internet Engineering Task Force, 2010. [Online]. Available: http://tools.ietf.org/rfc/rfc5751.txt

[54] D. Fomin and Y. Leboulanger. Gajim, a Jabber/XMPP client. [Online]. Available: https://gajim.org/

[55] D. Davis, "Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML," in *Annual Technical Conference, General Track*. USENIX, 2001, pp. 65–78.

[56] I. Brown, A. Back, and B. Laurie, "Forward Secrecy Extensions for OpenPGP," Draft, Internet Engineering Task Force, 2002. [Online]. Available: https://tools.ietf.org/id/draft-brown-pgp-pfs-03.txt

[57] R. Canetti, S. Halevi, and J. Katz, "A Forward-Secure Public-Key Encryption Scheme," in *Advances in Cryptology–EUROCRYPT*. Springer, 2003, pp. 255–271.

[58] M. Green and I. Miers, "Forward Secure Asynchronous Messaging from Puncturable Encryption," in *Symposium on Security and Privacy*. IEEE, 2015.
[59] N. Borisov, I. Goldberg, and E. Brewer, "Off-the-Record Communication, or, Why Not To Use PGP," in *Workshop on Privacy in the Electronic Society*. ACM, 2004, pp. 77–84.
[60] V. Moscaritolo, G. Belvin, and P. Zimmermann, *Silent Circle Instant Messaging Protocol Protocol Specification*, 2012.
[61] T. Perrin. (2013) Axolotl Ratchet. [Online]. Available: https://github.com/trevp/axolotl/wiki
[62] C. Kudla and K. G. Paterson, "Modular Security Proofs for Key Agreement Protocols," in *Advances in Cryptology–ASIACRYPT*. Springer, 2005, pp. 549–565.
[63] O. W. Systems. Open WhisperSystems. [Online]. Available: https://whispersystems.org/
[64] T. Frosch, C. Mainka, C. Bader, F. Bergsma, J. Schwenk, and T. Holz, "How Secure is TextSecure?" Cryptology ePrint Archive Report 2014/904, 2014. [Online]. Available: https://eprint.iacr.org/2014/904
[65] O. W. Systems. Open Whisper Systems partners with WhatsApp to provide end-to-end encryption. [Online]. Available: https://whispersystems.org/blog/whatsapp/
[66] ——. Private Group Messaging. [Online]. Available: https://whispersystems.org/blog/private-groups/
[67] I. Goldberg, B. Ustaoğlu, M. D. Van Gundy, and H. Chen, "Multi-party Off-the-Record Messaging," in *Conference on Computer and Communications Security*. ACM, 2009, pp. 358–368.
[68] M. Van Gundy, "Improved Deniable Signature Key Exchange for mpOTR," 2013, available from http://matt.singlethink.net/projects/mpotr/improved-dske.pdf.
[69] H. Liu, E. Y. Vasserman, and N. Hopper, "Improved Group Off-the-Record Messaging," in *Workshop on Privacy in the Electronic Society*. ACM, 2013, pp. 249–254.
[70] M. Mannan and P. C. van Oorschot, "A Protocol for Secure Public Instant Messaging," in *Financial Cryptography and Data Security*. Springer, 2006, pp. 20–35.
[71] C.-H. Yang and T.-Y. Kuo, "The Design and Implementation of a Secure Instant Messaging Key Exchange Protocol," 2007, available from http://crypto.nknu.edu.tw/psnl/publications/2007Technology_SIMPP.pdf.
[72] C.-H. Yang, T.-Y. Kuo, T. Ahn, and C.-P. Lee, "Design and Implementation of a Secure Instant Messaging Service based on Elliptic-Curve Cryptography," *Journal of Computers*, vol. 18, no. 4, pp. 31–38, 2008.
[73] C.-P. Lee and C.-H. Yang, "Design and Implement of a Secure Instant Messaging Service with IC Card," 2009, available from http://crypto.nknu.edu.tw/psnl/publications/2009CPU_SIMICCard.pdf.
[74] J. A. Cooley, R. I. Khazan, B. W. Fuller, and G. E. Pickard, "GROK: A Practical System for Securing Group Communications," in *International Symposium on Network Computing and Applications*. IEEE, 2010, pp. 100–107.
[75] H. Kikuchi, M. Tada, and S. Nakanishi, "Secure Instant Messaging Protocol Preserving Confidentiality against Administrator," in *International Conference on Advanced Information Networking and Applications*. IEEE, 2004, pp. 27–30.
[76] M. D. Van Gundy and H. Chen, "OldBlue: Causal Broadcast In A Mutually Suspicious Environment (Working Draft)," 2012, available from http://matt.singlethink.net/projects/mpotr/oldblue-draft.pdf.
[77] J. Reardon, A. Kligman, B. Agala, and I. Goldberg, "KleeQ: Asynchronous Key Management for Dynamic Ad-Hoc Networks," University of Waterloo, Tech. Rep., 2007.
[78] J. Bian, R. Seker, and U. Topaloglu, "Off-the-Record Instant Messaging for Group Conversation," in *International Conference on Information Reuse and Integration*. IEEE, 2007, pp. 79–84.
[79] (2015) (n+1)sec. [Online]. Available: learn.equalit.ie/wiki/Np1sec
[80] R. Kainda, I. Flechais, and A. W. Roscoe, "Usability and Security of Out-Of-Band Channels in Secure Device Pairing Protocols," in *Symposium on Usable Privacy and Security*. ACM, 2009.
[81] B. Warner. (2014) Pairing Problems. [Online]. Available: https://blog.mozilla.org/warner/2014/04/02/pairing-problems/
[82] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous Connections and Onion Routing," *Selected Areas in Communications*, vol. 16, no. 4, pp. 482–494, 1998.
[83] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," DTIC, Tech. Rep., 2004.
[84] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a Type III Anonymous Remailer Protocol," in *Symposium on Security and Privacy*. IEEE, 2003, pp. 2–15.
[85] A. Langley. Pond. [Online]. Available: https://pond.imperialviolet.org/
[86] D. Boneh, X. Boyen, and H. Shacham, "Short Group Signatures," in *Advances in Cryptology–CRYPTO*. Springer, 2004, pp. 41–55.
[87] C. C. Head, "Anonycaster: Simple, Efficient Anonymous Group Communication," 2012, available from https://blogs.ubc.ca/computersecurity/files/2012/04/anonycaster.pdf.
[88] H. Corrigan-Gibbs and B. Ford, "Dissent: Accountable Anonymous Group Messaging," in *Conference on Computer and Communications Security*. ACM, 2010, pp. 340–350.
[89] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in Numbers: Making Strong Anonymity Scale," in *Conference on Operating Systems Design and Implementation*. USENIX, 2012, pp. 179–182.
[90] E. Syta, H. Corrigan-Gibbs, S.-C. Weng, D. Wolinsky, B. Ford, and A. Johnson, "Security Analysis of Accountable Anonymity in Dissent," *Transactions on Information and System Security*, vol. 17, no. 1, p. 4, 2014.
[91] H. Corrigan-Gibbs, D. I. Wolinsky, and B. Ford, "Proactively Accountable Anonymous Messaging in Verdict," arXiv e-prints, Tech. Rep. arXiv:1209.4819, 2012.
[92] B. Project. Bitmessage. [Online]. Available: https://bitmessage.org/
[93] L. Sassaman, B. Cohen, and N. Mathewson, "The Pynchon Gate: A Secure Method of Pseudonymous Mail Retrieval," in *Workshop on Privacy in the Electronic Society*. ACM, 2005, pp. 1–9.
[94] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, "Deniable Encryption," in *Advances in Cryptology–CRYPTO*. Springer, 1997, pp. 90–104.
[95] Y. Dodis, J. Katz, A. Smith, and S. Walfish, "Composability and On-Line Deniability of Authentication," in *Theory of Cryptography*. Springer, 2009, pp. 146–162.
[96] A. Ulrich, R. Holz, P. Hauck, and G. Carle, "Investigating the OpenPGP Web of Trust," in *Computer Security–ESORICS*. Springer, 2011, pp. 489–507.
[97] R. L. Rivest and B. Lampson, "SDSI – A Simple Distributed Security Infrastructure," 1996, manuscript.
[98] C. M. Ellison, "Establishing Identity Without Certification Authorities," in *Security Symposium*. USENIX, 1996, pp. 67–76.
[99] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, "SPKI Certificate Theory," RFC 2693 (Experimental), Internet Engineering Task Force, 1999. [Online]. Available: http://tools.ietf.org/rfc/rfc2693.txt
[100] M. Wachs, M. Schanzenbach, and C. Grothoff, "A Censorship-Resistant, Privacy-Enhancing and Fully Decentralized Name System," in *Cryptology and Network Security*. Springer, 2014, pp. 127–142.
[101] ——, "On the Feasibility of a Censorship Resistant Decentralized Name System," in *Foundations and Practice of Security*. Springer, 2014, pp. 19–30.
[102] Keybase. Keybase – understanding tracking. [Online]. Available: https://keybase.io/docs/tracking

APPENDIX A
DENIABILITY

Deniability, also called *repudiability*, is a common goal for secure messaging systems. Consider a scenario where Bob accuses Alice of sending a specific message. Justin, a judge, must decide whether or not he believes that Alice actually did so. If Bob can provide evidence that Alice sent that message, such as a valid cryptographic signature of the message under Alice's key, then we say that the action is *non-repudiable*. Otherwise, the action is *repudiable* or *deniable*. We can distinguish between *message repudiation*, in which Alice denies sending a specific message, and *participation repudiation* in which Alice denies communicating with Bob at all. The high-level goal of repudiable messaging systems is to achieve deniability similar to real-world conversations.

A fundamental problem of deniability is that Justin may simply trust Bob even with no technical evidence due to Bob's reputation or perceived indifference. In a group chat, this problem may be even worse as Alice may need to convince Justin that a number of accusers are all colluding to frame her. It is not possible to construct a messaging system that overcomes this fundamental social problem; the best that can be done is to provide no stronger evidence than the word of the accusers. Some technical systems clearly offer more evidence; for example, signed PGP emails offer strong evidence that Alice really was the sender.

The cryptographic literature has produced many definitions of "deniability" since deniable encryption was first formally proposed [94]. For example, we can draw a distinction between an offline and online judge: in the offline case, the accuser attempts to convince the judge of an event after the conversation has already concluded; in the online case, the judge exchanges private communications with the accuser while the conversation is still taking place. Existing work defines online repudiation in incompatible ways, and very few protocols attempt to achieve meaningful online repudiation [69], [95]. Thus, in this work we only consider the offline setting.

Appendix B
Web of Trust

In a *web of trust* scheme, users verify each other's keys using manual verification and, once they are satisfied that a public key is truly owned by its claimed owner, they sign the key to certify this. These certification signatures might be uploaded to key servers. If Alice has verified Bob's key, and Bob certifies that he has verified Carol's key, Alice can then choose to trust Carol's key based on this assertion from Bob. Ideally, Alice will have multiple certification paths to Carol's key to increase her confidence in the key's authenticity.

The user interface for web of trust schemes tends to be relatively complex and has never been fully standardized. The scheme also requires a well-connected social graph, hence the motivation for "key-signing parties" to encourage users to form many links within a common social context.

Assuming that the web of trust model performs correctly, MitM attacks by network and operator adversaries are limited due to distribution of trust. However, since key revocations and new keys might be withheld by key servers, the model offers only partial *operator accountability* and *key revocation*. Since the web of trust model produces a public social graph, it is not *privacy preserving*.

The key initialization phase requires users to get their keys signed by other keys, so the system does not offer *automatic key initialization*, *alert-less key renewal*, or *immediate enrollment*, and is not *inattentive user resistant*. Because users must participate in key-signing parties to create many paths for trust establishment, users have a high *key maintenance* overhead and a need for an *out-of-band* channel. Even worse, users must understand the details of the PKI and be able to decide whether to trust a key.

PGP typically uses a web of trust for email encryption and signing. In practice, the PGP web of trust consists of one strongly connected component and many unsigned keys or small connected components, making it difficult for those outside the strongly connected component to verify keys [96].

A simplification of the general web of trust framework is SDSI [97] (Simple Distributed Security Infrastructure) later standardized as SPKI [98], [99] (Simple Public Key Infrastructure). With SDSI/SPKI, Bob can assert that a certain key belongs to "Carol" and, if Alice has verified Bob's key as belonging to "Bob," that key will be displayed to Alice as "Bob's Carol" until Alice manually verifies Carol's key herself (which she can then give any name she wants, such as "Carol N."). We refer to these approaches as *trust delegation*. A modern implementation is the GNU Name System (GNS) [100], [101], which implements SDSI/SPKI-like semantics with a key server built using a distributed hash table to *preserve privacy*. Other web of trust approaches such as SDSI [97], [98], SPKI [99], Keybase [102] and GNS [100] are described and evaluated in the extended version of this paper [1].

Appendix C
OTR Networks

Since OTR [59] provides desirable features for two-party conversations, it is natural to extend it to a group setting by using OTR to secure individual links in a network. A basic strategy is to enlist a trusted entity to relay messages and then secure client links to this entity using OTR. This is the approach taken by the GOTR (2007) protocol. GOTR (2007) [78] selects a participant to act as the relay, forming a star topology of pairwise connections with the selected participant acting as the hub. All *authentication* properties, *speaker consistency*, and *causality preservation* are lost because they do not persist across the relay node. Since the relay server can buffer messages, *asynchronicity* is provided as long as the relay node remains online. All other properties are inherited from OTR. Groups can be *expanded* and *contracted* simply by establishing new OTR connections to the relay.

Instead of using a star topology, pairwise OTR connections between all participants can be established. This approach restores *authentication* and *anonymity preservation*, as well as *equal trust* between members. It is also possible to send messages to *subgroups* by only transmitting the message across selected OTR links. The downside of this approach is that it does not *preserve causality* or provide *speaker consistency*; participants can send different messages to different people. This design also incurs significant *computational overhead*.