Mind the Gap: Where Provable Security and Real-World Messaging Don't Quite Meet

Extended Abstract

Katriel Cohn-Gordon, Cas Cremers
me@katriel.co.uk, cas.cremers@cs.ox.ac.uk
Department of Computer Science, University of Oxford

Abstract

Secure messaging apps have enjoyed huge uptake, and with the headline figure of one billion active WhatsApp users there has been a corresponding burst of academic research on the topic. One might therefore wonder: how far is the academic community from providing concrete, applicable guarantees about the apps that are currently in widespread use?

We argue that there are still significant gaps between the security properties that users might expect from a communication app, and the security properties that have been formally proven. These gaps arise from dubious technical assumptions, tradeoffs in the name of reliability, or simply features out of scope of the analyses. We survey these gaps, and discuss where the academic community can contribute. In particular, we encourage more transparency about analyses' restrictions: the easier they are to understand, the easier they are to solve.

In the past few years, secure messaging apps have enjoyed explosive growth in uptake, perhaps best exemplified by WhatsApp's headline figure of a billion active users. There has been a corresponding burst of academic research into these secure messaging protocols [7, 10, 12, 18, 24], and in particular into Open Whisper Systems' *Signal Protocol*, which is fast becoming an industry standard. Kobeissi, Bhargavan, and Blanchet [18] and Cohn-Gordon et al. [7] both give formal security proofs of Signal.

One might imagine that, armed with the certainty provided by a mathematical proof of security, users could rest assured that their messages are only readable by the right people. In this extended abstract we discuss the myriad ways in which this may fail to hold despite these proofs. We aim (i) to clarify the limits of existing analyses, and (ii) to show how new research and collaboration can help bridge this gap.

We focus here on three main areas. In §I we discuss issues with the security theorems themselves, in §II topics out of scope of most analyses but necessary to deploy a practical protocol, and in §III additional features added by implementers.

I. Limitations and Assumptions in Security Models

Every security model makes some **assumptions** on the cryptographic primitives that it uses. In the symbolic methodology, these assumptions are generally clear, simple and too strong; for example, that encryption is a black box revealing no information about the plaintext, or that it provides message integrity. It is also not uncommon to perform bounded verification, in which the relevant property is checked only for a small number of agents or sessions (often as few as two or three).

Computational models use more fine-grained assumptions; sometimes these are fairly standard (e.g. given Diffie-Hellman values g^x and g^y, it is hard to compute g^{xy}, an assumption known as CDH), but modern key exchange protocols with complex key derivations often require assumptions such as Gap-DH, PRF-ODH [5], XDH or similar. These assumptions are much less well-understood and rely on internal properties of the underlying group, which may well not be true when instantiated. However, if the chosen assumptions does not hold in the chosen group, then the security proof does not prove anything. Computational models also often assume that hash functions are random oracles, despite the well-known problems with this assumption [6].

Computational **reductions** also merit highlighting, since they are "loose" for almost all protocol proofs [2]. This implies that the concrete security parameters needed to invoke a security theorem with reasonable guarantees on its protocol are generally much, much larger than those actually used when

the protocol is deployed. Seen one way, this is a technicality; seen another, it means that most proofs do not apply to actual instantiations of their protocols.

Almost all protocol security models assume that every agent knows its peers' **public keys**; this distribution must actually be performed when protocols are deployed. TLS uses the well-loved X.509 PKI, with all its attendant quirks, but messaging protocol implementations generally just nominate the service provider as a trusted third party for key distribution. (The Signal app includes an out-of-band "fingerprint" or "safety number" channel—but many users will not understand or perform this verification.) Of course, a malicious or coerced service provider could intercept Alice's messages by giving her malicious keys.

There are some new and exciting designs attempting to remove some trust in the service provider, based on Google's original Certificate Transparency [13]. Perhaps the closest to being in widespread use is Key Transparency [14], which is in turn based on CONIKS [21] but with some changes made by Google; other examples include [3, 17, 25, 26, 27]. However, one downside of the Merkle-tree family of solutions is that they are generally very complex, with a number of different subsystems: log servers, monitors, auditors, and gossip between clients.

Recent work on **detection** by Milner et al. [22] considers techniques for users to detect disagreement, for example, on the keys they have been provided. Their work could provide a simpler way for users to notice malicious key distributions, assuming a weaker-than-Dolev-Yao adversary.

II. Analysis Scope is Always Limited

We turn now to practical considerations which apply to all deployed secure messaging apps but are generally excluded from the scope of formal analyses.

Although there are a number of ways to measure and quantify resistance to **side channel** attacks, most widely-used protocol models do not consider them, restricting the adversary to "legitimate" communication channels with participants. However, many of the most significant reported vulnerabilities of the past few years fall into this category; padding oracles are a particularly fruitful class.

Some models allow the adversary to learn or affect the output of agents' **random number generators**, which overapproximates certain types of side channel attacks. However, many protocols assume that the random numbers generated during executions are perfect, and indeed fail if they are not. Signal and (all versions of) TLS fall into this category, although there is some research on how to do better.

On the **implementation** side, the app installed on users' devices does not necessarily correctly implement the abstract protocol it uses. Errors here are a major concern and source of employment for red teams: the Debian patch removing OpenSSL's entropy source is a famous example, but others abound. Many implementations of the Signal protocol use the Open Whisper Systems `libsignal-protocol` codebases in Java, Objective-C or Javascript (and thus will share any bugs that arise), but this has licensing restrictions and some vendors have reimplemented Signal from scratch. (This is a common pattern: Beurdouche et al. [4] reveal many different and buggy state machines arising from vendor reimplementations of TLS.)

Perhaps equally worrying, though, is the fact that the code actually executing on devices is generally an **opaque binary** compiled by the operating system vendor. This means that there is no way for a user to be certain that the code running on their device is actually the open-source code that they might trust. Indeed, even the Signal app uses Google's closed-source Play Services library to handle push notifications: if Google deployed or were coerced to deploy malicious code in this library, they could easily access message plaintexts in a way that would not be easy to detect.

There is a significant amount of research in this area, but few ideas have achieved widespread deployment. One simple approach is to allow users who do not trust OS-provided binaries to compile and run their own clients, but (for a different reason) the Signal developers do not allow federation [19].

III. Implementers Really Like Features

Finally, real-world messaging applications need to support more features than are usually described in basic protocol models, not just because users expect them but also because service providers wish to differentiate themselves in features. This leads to a large variety of extra options which, in formal analyses, are often described as outside the "cryptographic core" (in part because analyses of real-world protocols are already approaching the limits of scalability without them).

1) Retries: Abu-Salma et al. [1] conclude from multiple interviews that users actually value reliability over security, to the extent of using the former as a proxy for the latter. This has not gone unnoticed by service providers, who often implement retry logic to hide transient failures from users. This logic can go further than might be expected: WhatsApp will silently re-encrypt a sent message to a new key if it believes that the recipient has changed device, a feature originally branded a backdoor by the Guardian [11] but now referred to as a "tradeoff" [16].

2) Backups: For mass-market communication apps, users do not want to lose their message histories when they lose or replace a device; thus, some apps allow users to export or back up their messages. For example, WhatsApp allows scheduled (encrypted) backups to Google Drive, and Signal allows plaintext but not encrypted XML exports. Scheduled cloud backup in particular, while very user-friendly, means that an adversary who compromises the backup key can eavesdrop on users' later conversations via the backup, even though the frequent re-keying in Signal (leading to post-compromise security [8]) means that they cannot necessarily intercept messages in transit.

3) Franking: Any sufficiently popular messaging system will come with its share of abusive messages, which service providers wish to to detect and prevent. However, in an end-to-end encrypted context, providers cannot simply inspect messages to classify them as abusive. This led Facebook to build a franking system for their deployment of Signal, subsequently studied as an example of "committing authenticated encryption" by Grubbs, Lu, and Ristenpart [15].

4) GIF search: The Signal app currently supports privacy-aware GIF search [20], by proxying a TLS connection from the Signal app through Signal's servers to Giphy. This nominally allows users to maintain their privacy while still communicating with an external service. In a related feature, WhatsApp displays URL previews when users enter a URL and before sending a message, in the process revealing URLs character-by-character [23] to anyone on the network.

5) Group and Multidevice: It turns out that many Signal implementations allow users to send messages to multiple entities (either sets of users, or multiple devices of a single user) but provide, without notification, a weaker security property for these conversations. Specifically, apps using Signal's "sender key" design permit an adversary who compromises a single group member to intercept all of their subsequent communications.

There are many systems for secure group messaging, but generally these are inspired by the instant-messaging paradigm and require interactive, online broadcast rounds. In [9], we give a novel method for group messaging to achieve Signal's post-compromise security guarantees without interactivity or synchronicity, building on known tree-based designs but avoiding their broadcast rounds.

IV. Conclusion

In the abstract, we asked how far the community is from providing concrete guarantees about real-world messaging protocols. The many examples above show that there is still quite some distance to cover before we can truly claim to have proofs about deployed systems: many features that are frequently written off as "just engineering" or "out of scope" are in fact vitally important for a mass-market protocol

However, our key takeaway is that although there are many problems, there are also plenty of solutions coming from both industry and academia, and the frequent crossovers between the two domains paint an encouraging picture. Overall, we feel that there is scope for significant advances in the state of the art, and we hope that future research continues to address the challenges we describe.

Finally, we wish to encourage researchers once more to be transparent about the limitations of their analyses: the easier it is to understand the scope of a proof or the assumptions in a theorem, the easier it is to improve on the state of the art.

REFERENCES

[1] Ruba Abu-Salma, M. Angela Sasse, Joseph Bonneau, Anastasia Danilova, Alena Naiakshina, and Matthew Smith. "Obstacles to the Adoption of Secure Communication Tools". In: *2017 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2017, pp. 137–153.

[2] Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. "On the Impossibility of Tight Cryptographic Reductions". In: *EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Heidelberg, May 2016, pp. 273–304. DOI: 10.1007/978-3-662-49896-5_10.

[3] David A. Basin, Cas J. F. Cremers, Tiffany Hyun-Jin Kim, Adrian Perrig, Ralf Sasse, and Pawel Szalachowski. "ARPKI: Attack Resilient Public-Key Infrastructure". In: *ACM CCS 14*. Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. ACM Press, Nov. 2014, pp. 382–393.

[4] Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue. "A Messy State of the Union: Taming the Composite State Machines of TLS". In: *2015 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2015, pp. 535–552. DOI: 10.1109/SP.2015.39.

[5] Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. "PRF-ODH: Relations, Instantiations, and Impossibility Results". In: *CRYPTO 2017, Part III*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10403. LNCS. Springer, Heidelberg, Aug. 2017, pp. 651–681.

[6] Ran Canetti, Oded Goldreich, and Shai Halevi. "The Random Oracle Methodology, Revisited (Preliminary Version)". In: *30th ACM STOC*. ACM Press, May 1998, pp. 209–218.

[7] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. *A Formal Security Analysis of the Signal Messaging Protocol*. Cryptology ePrint Archive, Report 2016/1013. http://eprint.iacr.org/2016/1013. 2016.

[8] Katriel Cohn-Gordon, Cas Cremers, and Luke Garratt. *On Post-Compromise Security*. Cryptology ePrint Archive, Report 2016/221. http://eprint.iacr.org/2016/221. 2016.

[9] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. *On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees*. Cryptology ePrint Archive, Report 2017/666. http://eprint.iacr.org/2017/666. 2017.

[10] Tilman Frosch, Christian Mainka, Christoph Bader, Florian Bergsma, Joerg Schwenk, and Thorsten Holz. *How Secure is TextSecure?* Cryptology ePrint Archive, Report 2014/904. http://eprint.iacr.org/2014/904. 2014.

[11] Manisha Ganguly. *WhatsApp design feature means some encrypted messages could be read by third party*. 2017-01-13. URL: https://www.theguardian.com/technology/2017/jan/13/whatsapp-design-feature-encrypted-messages.

[12] Ian Goldberg, Berkant Ustaoglu, Matthew Van Gundy, and Hao Chen. "Multi-party off-the-record messaging". In: *ACM CCS 09*. Ed. by Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis. ACM Press, Nov. 2009, pp. 358–368.

[13] Google. *Certificate Transparency*. URL: https://www.certificate-transparency.org/ (visited on 10/2017).

[14] Google. *Key Transparency*. 2017. URL: https://github.com/google/keytransparency.

[15] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. "Message Franking via Committing Authenticated Encryption". In: *CRYPTO 2017, Part III*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10403. LNCS. Springer, Heidelberg, Aug. 2017, pp. 66–97.

[16] Frederic Jacobs. *On the "WhatsApp backdoor", Trade-Offs and Opportunistic Authentication*. Jan. 20, 2017. URL: https://www.fredericjacobs.com/blog/2017/01/20/whatsapp-backdoor/.

[17] Tiffany Hyun-Jin Kim, Lin-Shung Huang, Adrian Perrig, Collin Jackson, and Virgil Gligor. "Accountable Key Infrastructure (AKI): A Proposal for a Public-key Validation Infrastructure". In: ACM, 2013. DOI: 10.1145/2488388.2488448.

[18] N. Kobeissi, K. Bhargavan, and B. Blanchet. "Automated Verification for Secure Messaging Protocols and their Implementations: A Symbolic and Computational Approach". In: *IEEE European Symposium on Security and Privacy (EuroS&P)*. to appear. 2017.

[19] Moxie Marlinspike. *Reflections: The Ecosystem is Moving*. May 10, 2016. URL: https://signal.org/blog/the-ecosystem-is-moving/.

[20] Moxie Marlinspike. *Signal and GIPHY*. 2016-11-01. URL: https://signal.org/blog/giphy-experiment/.

[21] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. *CONIKS: Bringing Key Transparency to End Users*. Cryptology ePrint Archive, Report 2014/1004. http://eprint.iacr.org/2014/1004. 2014.

[22] Kevin Milner, Cas Cremers, Jiangshan Yu, and Mark Ryan. *Automatically Detecting the Misuse of Secrets: Foundations, Design Principles, and Applications*. Cryptology ePrint Archive, Report 2017/234. http://eprint.iacr.org/2017/234. 2017.

[23] Twitter post. *@mulander*. 2017. URL: https://twitter.com/mulander/status/874370124932943874.

[24] Paul Rösler, Christian Mainka, and Jörg Schwenk. *More is Less: How Group Chats Weaken the Security of Instant Messengers Signal, WhatsApp, and Threema*. Cryptology ePrint Archive, Report 2017/713. http://eprint.iacr.org/2017/713. 2017.

[25] Mark D. Ryan. "Enhanced certificate transparency and end-to-end encrypted mail". In: *In Network and Distributed System Security Symposium (NDSS). Internet Society*. 2014.

[26] Jiangshan Yu, Vincent Cheval, and Mark Ryan. *DTKI: a new formalized PKI with no trusted parties*. Cryptology ePrint Archive, Report 2014/600. http://eprint.iacr.org/2014/600. 2014.

[27] Jiangshan Yu, Mark Ryan, and Cas Cremers. "DECIM: Detecting Endpoint Compromise In Messaging". In: *IEEE Transactions on Information Forensics and Security* PP.99 (Aug. 2017). DOI: 10.1109/TIFS.2017.2738609.