Analysing Cryptography in the Wild
A Retrospective

Martin R. Albrecht[1] and Kenneth G. Paterson[2]

[1] King's College London
[2] ETH Zurich

Abstract. We reflect on our experiences analysing cryptography deployed "in the wild" and give recommendations to fellow researchers about this process.

1 Introduction

Cryptography is not in Kansas, anymore. From e-commerce, online banking, payment systems, mobile phones, government/military communication systems and authentication systems of old, it has spread to secure messaging, video conferencing, full disc encryption, encrypted cloud storage, cryptographic custody solutions, anonymous authentication tokens, private browser analytics and privacy-preserving contact tracing, to name but a few.

Meanwhile, academic cryptography, too, has moved on and expanded its scope. Additionally, the rigour with which we study cryptographic artefacts, from primitives (such as public-key encryption), schemes (such as modes of encryption) to protocols (such as TLS), has increased.

Yet, in all the excitement about deploying cryptography to secure our brave new digital world, those cryptographic solutions in practice rarely receive that "academic" level of scrutiny, either in private before deployment or in public post deployment. This leads to a glut of cryptographic technologies "out there, in the wild", often protecting the data of millions of people, where the veracity of their security promises is unclear.

Studying "cryptography in the wild", then, means to find examples of cryptography being used in standards, products or deployed systems. It means to analyse them by either finding vulnerabilities and reporting them or by building security models and proofs for the cryptographic cores of these systems. The end result is that those who use these systems gain greater assurance about the security of the systems on which they rely.

Cryptography in the wild as it is understood here is *not* about developing new cryptographic primitives and pushing them towards practice. It is not applied cryptography in the sense of applying cryptography to real-world settings. This is an incredibly valuable activity and is the traditional theory-to-practice transfer process. Rather, here we are studying current practice as is. The activities considered here can be defined as applied cryptanalysis, the analysis of cryptographic solutions after their realisation in the wild. In this article, we present our reflections on this activity. These are based on our combined experience of pursuing it over the last couple of decades, as well as our observation of the work of the small, but growing, community of researchers who have chosen to plough the same furrow as us.

The nature of this piece – a reflection on our own practice – implies a certain accumulation of references to our own work. We would like to stress that this is not at all reflective of this area, which is shaped by the works of many others such as [HDWH12, ABD$^+$15, GGK$^+$16, MBP$^+$19, DPS21] to name but a few.

2 Methodology

The methodology consists of selecting an object of study (a protocol, an implementation, a scheme), deciding on what it means for this object to be "secure" by identifying a suitable adversarial model and security goals, and analysing the extent to which those goals are met.[3]

Object Selection. The process of selecting an object to study can start from two directions. We can start from the cryptographic technology. Does a product use non-trivial and/or non-standard cryptographic constructions? Do the security claims of the vendor seem to match what we could reasonably expect from the advertised building blocks? Is the cryptographic processing easy to analyse in the form of available source code and whitepapers? If not, is it easy and legal to reverse engineer? What are the first impressions? On the other hand, we may also start from social or societal considerations. How big is the (claimed) "user base", i.e. people relying on this piece of technology, either in absolute terms or in some area, e.g. a particular country or in a particular group of people? Are there users in sometimes (e.g. protesters) or continuously (e.g. domestic abuse victims) particularly vulnerable situations?

Our analysis of Telegram [AMPS22] came about as a result of the second kind of consideration. A previous work [ABJM21] had highlighted (similarly to prior works in other settings) that protesters in Hong Kong were relying on Telegram to coordinate their activities. Crucially, the optional end-to-end encryption in Telegram, which had been considered somewhat in the cryptographic literature, played essentially no role in this setting. Instead, large group chats secured by Telegram's bespoke MTProto 2.0 protocol in lieu of TLS were at the centre of the communication infrastructure for these protests. Based on this [AMPS22] studied Telegram's secure channel protocol, found several vulnerabilities (which were fixed after being reported) and fashioned a formal statement of the security guarantees of the repaired protocol.

The latter is otherwise known as "a proof of security" in the literature. However, here we deliberately shun this language to avoid the impression of a

[3] The wider literature often uses the term "threat model" to also mean adversarial models as understood here. We avoid this term here because it only has a rather loose meaning and thus can give a semblance of rigour where it is lacking.

binary "yes/no" answer to the question of security. Rather, the relevant theorems establish certain security guarantees under certain conditions.

This conflation – a security proof is a binary statement rather than a statement of a particular security guarantee in a given model – is also one reason why object selection can be contentious at times. Producing a formal security statement for, say, a brittle protocol or a protocol in a product from a controversial company might be considered as providing a "seal of approval" for shoddy practices.[4] We argue that often this ship has sailed. If a considerable number of people rely on a given piece of technology, its security guarantees need to be understood, no matter how we might view the particular parties involved.

It is worth noting that an object having survived one or more security audits should not deter researchers from studying it. These audits are typically performed under a strict time limit and so tend to focus on more surface-level security issues than those uncovered by an in-depth cryptographic investigation of the type discussed here. Our own prior works referenced throughout this article all took several person-months to complete. Multiplying that time by the assumed day-rate of an experienced auditor results in an amount that is well beyond what most vendors are willing to pay. The flip-side of this observation is that we do not consider the existence of positive security audits as providing meaningful evidence of the soundness of a cryptographic solution. Going further, we strongly suggest that consumers of cryptography demand formal theorems and proofs to at least rule out large classes of attacks.

Adversarial Model. When studying cryptographic solutions in the wild, we have to settle on what adversarial model we select. In the case of [AMPS22] this meant not considering the Telegram server as the adversary, i.e. not targeting end-to-end encryption. This decision was based on prior works suggesting this was not the most pressing security concern. Often, defining such a model is straightforward: if end-to-end encryption is offered then it is fair to consider the service provider as an adversary. Sometimes, the marketing material used to promote a given technology will provide an indication of what the developers themselves see as being a suitable adversarial model, though non-standard terminology may be used to make these claims.[5]

However, the decision to choose a particular adversarial model can be non-trivial and, again, lead to controversy. Just because end-to-end encryption is built-in and even advertised as present, this feature may not be of central importance to the developers. Yet, some groups of people may rely on these guarantees. For example, Bridgefy advertised itself as a solution for reaching

[4] Here we do not mean misrepresenting the security guarantees, e.g. by omitting non-standard assumptions or the brittleness of the design. We simply mean formalising the security guarantees provided and the conditions under which these guarantees hold.

[5] For example, the term "zero-knowledge encryption" has become popular in the realm of cloud storage, meaning roughly that not even the storage provider should be able to access user data. However, this term does not match any standard technical definition in the research domain.

target groups in disaster areas thanks to its mesh networking technology. On the other hand, media reports claimed adoption of Bridgefy's flagship messenger in some protest and conflict settings, claims which Bridgefy were happy to amplify on their website and social media. However, while the Bridgefy developers advertised their product as being secure, this was not tended to with sufficient care: the protocol was broken, fixed, and broken again with practical attacks, all in the span of two years.

So the study of cryptography in the wild may require defining an adversarial model in the face of opposition from the providers of the technology in question; adversarial adversary definitions, so to speak. We consider it a responsibility of academic cryptographic research to not allow vendors to define adversarial models for their own technologies. As we will argue below, we see our responsibility as not to the technology producers but to those who (may be forced to) rely on it. Discharging this responsibility starts with defining what we consider to be a suitable adversarial model based on the actual use of the technology, i.e. the place of that technology in the world, not the way in which its designers prefer to think about it or intend it to be used.[6]

Ingesting. The next step is to digest the available information, a process that may take months to complete. This information might come from whitepapers, design documents, API documentation, security audit reports or source code. It may also involve reverse engineering readable source code from binary blobs or minified archives, a process that we caution may itself take weeks if not months to conduct.

A natural next step is then to build pseudo-code models describing the cryptographic "core" of the object under study. Defining this core, too, can be difficult. For example, cryptographic group management is typically considered out of scope/outside the model when formally studying (group) messaging, yet this is a routine source of vulnerabilities as any kind of secure group management is often simply absent (as for example is the case with Matrix and WhatsApp).

Another difficulty here is to find the right level of abstraction and to unify the possibly conflicting implementations in different implementations (for example mobile and desktop clients) into that one pseudo-code model. Continuing with our example of analysing Telegram from above, we looked at three different official Telegram clients and found that they all performed slightly different checks on messages to decide on whether to accept them or not, each choice having different security consequences. Even official Telegram clients deviated from Telegram's own implementation advice for developers. This left open the question of what the "correct" choice should be in a model.

[6] Living up to this responsibility can lead to highly complex questions more adequately answered by the social sciences rather than computer science: how do we understand actual use and how do we cryptographers translate that into a suitable adversarial model? A technology might have many places in the world, how do we decide on an adversarial model, which "place" is taken into consideration? Or do we consider different adversarial models for a piece of technology depending on its role in different social settings?

The choice of model can have a dramatic effect on whether relevant and practical attacks are discovered or not. Again, picking an example from our own work, our first work breaking (Open)SSH [APW09] succeeded despite it enjoying a proof of security [BKN02]. This was possible because our attack exploited a property of the protocol abstracted away in the security model. Among other works, our later work [ADHP16] produced a security theorem in a refined model. This, too, was later shown to be incorrect in a work presenting a practical attack on SSH [BBS23]. This attack, again, succeeded because a simplifying assumption – an abstraction – made in our security model turned out to be both false and significant.

Attacks. With pseudo-code models in hand, we can start to reason about potential avenues of attack (or proofs). This might entail considering known attacks from the literature: ECB mode, exotic or home-made encryption modes, lack of integrity mechanisms, improper use of integrity (e.g. Mac-then-Encrypt Encrypt-and-Mac), the presence of padding oracle vulnerabilities, nonce reuse, lack of proper key separation/key reuse, lack of domain separation, bad interactions between different protocols, use of weak PRNGs or home-brew randomness generation methods, compression combined with encryption, etc, etc.

Of course, a more rewarding direction is to come up with novel (variants of) attacks, possibly chaining together multiple attacks to achieve a given aim. This is a process sometimes endearingly referred to as "stunt cryptography" (a phrase we attribute to Thomas Ptacek[7]). This may be needed if the object under study does not allow us to apply known attacks as is, yet the object is – by inspection – insecure. Turning that inspection into a practical attack might then require significant new ideas.

Proof of Concepts. It is much easier to convince third parties of the seriousness of a vulnerability by providing a working proof-of-concept exploit. Indeed, most bug bounty programs require this of submitters. Yet, developing such exploits can be a laborious process. The obstacle to pulling off that stunt might be a computation taking 2^{60} steps and several person-months to carefully implement, cf. the construction of a special key pair needed to complete an attack in [PST23]. Even if a vulnerability should be exploitable without consuming too many resources (queries, time, memory), developing proof-of-concept exploits can be time and resource intensive. Quite often, the complexity arises from wrestling with the state-machine or parser of the considered protocol and not the cryptographic, and thus interesting, core.

Proofs. Finally, the cryptographic object under study might be amenable to a more formal analysis, i.e. the establishment of a formal theorem characterising its security in the previously defined adversarial model. This rules out large classes of attacks and thus gives greater confidence in the actual security properties of the studied object. The difficulties here often stem from needing to first develop

[7] `https://news.ycombinator.com/item?id=31829130`

models of security, the studied protocol attempting to use novel cryptographic functionalities, or the studied protocol using unconventional approaches to solve cryptographic problems with known solutions, e.g. by hashing a key together with some message in lieu of a MAC construction. For these bespoke constructions, we might be able to reduce their security to some standard and well-studied assumptions, or we might need to simply state the required (likely novel) assumptions on the underlying primitives needed for the proof to go through. In the latter case, opportunities are created for follow-up research to either support or invalidate these assumptions.

3 Responsibilities

A standard step in this line of research, if significant vulnerabilities are discovered, is to consider how to disclose these vulnerabilities in a way that minimises harm, defined in some way. The standard approach here is "coordinated vulnerability disclosure", previously known as "responsible disclosure". This involves privately disclosing vulnerabilities to vendors, typically with a 90-day deadline attached. If by the end of the disclosure period no remedy is made available, the vulnerability will be publicly announced regardless. Prior to this deadline, the invitation is for the vendor and the researchers to coordinate the public disclosure. In our experience, it may happen for the initial deadline to slip to 120 days but not longer. We strongly recommend that the decision on timing should be maintained under the complete control of the research team.

It is worth, though, mentioning that for some (classes of) vulnerabilities a 90 day disclosure vulnerability is *a priori* unrealistic. Examples that come to mind are hardware/micro-architecture level vulnerabilities. Put differently, the "90 days" rule is somewhat arbitrary but works fairly well for software-based vulnerabilities when the vendor has an established update process.

A common misconception here is that the research team has a responsibility to the vendor. Yet, no such particular responsibility exists, as typically there is no pre-established legal or commercial relationship between the parties. Rather, insofar we want to or can speak of responsibility, we see it as being to those who (have to) rely on the technology provided by the vendor in question.[8] The users of an insecure-by-design protocol may be better protected by them being warned against relying on it rather than attempts to patch it by an inexperienced team of developers. Yet, avoiding a given protocol may simply be impossible for many. This suggests cooperation with vendors as being a solid strategy. This may involve informally advising vendors on remediation strategies and reviewing their patches. We recommend that research teams discuss amongst themselves (or potentially with trusted mentors) to whom they consider themselves responsible before disclosing to vendors.

Researchers do have a responsibility to be realistic about the impact of any vulnerabilities discovered when communicating publicly. Insinuating the existence

[8] While we, the authors, certainly consider this our responsibility, we are mindful that we are in no position to prescribe such a responsibility for others.

of serious vulnerabilities in a robust system may deter some from using it, possibly encouraging them to pivot to insecure alternatives.

4 Reception

The style of works discussed here is received by three rather distinct audiences.

Vendors. Disclosing vulnerabilities to vendors is often a time-consuming and frustrating process. Some vendors may simply never have handled a vulnerability disclosure before and thus lack any processes for dealing with one when it arrives out of the blue. Other vendors enthusiastically start to make patches in public code repositories, forgetting everything that the word "coordinated" implies, and potentially allowing the patches to be reverse-engineered and exploited by others before new product versions can be distributed. Yet other vendors do not ever tell their users that their product is being updated for security reasons. More rarely, disclosure is a butter-smooth and pleasurable experience, in which case the vendor concerned should be publicly credited.

A worrying trend in this area is that vendors "outsource" this process to "bug bounty" service providers. These services are based around the idea of vulnerability disclosure being financially rewarded, often in return for some form of agreement on non-disclosure. This process is unsuited to the activities considered here; both with regard to their motivation – improving security independently of financial reward – and with regard to their content – cryptographic vulnerabilities are rarely disclosed and thus the industry around vulnerability disclosures and bug bounties is inexperienced in dealing with them.

We recommend avoiding any intermediaries in the disclosure process, since they may impose their own policies with negative consequences. For example, our disclosure of the previously mentioned plaintext recovery attack against (Open)SSH [APW09] was hampered by us going via the UK's Centre for the Protection of National Infrastructure, resulting in the following notification on the OpenSSH project webpages:

> *The OpenSSH team has been made aware of an attack against the SSH protocol version 2 by researchers at the University of London. Unfortunately, due to the report lacking any detailed technical description of the attack and CPNI's unwillingness to share necessary information, we are unable to properly assess its impact.*[9]

Vendors who ship cryptography may also overestimate their own understanding of it. Vendors might not be impressed with an attack on the IND-CPA security of their scheme but may expect key or plaintext recovery. Similarly, they might dismiss an attack taking 2^{60} steps as too expensive, underestimating the tendency of "attacks only getting better" with time. As an extreme illustration of this, consider the rapid evolution of attacks against MEGA, an end-to-end encrypted

[9] See `https://www.openssh.com/txt/cbc.adv`.

cloud storage service. The first analysis required a user to make 512 logins in order to be able to recover their private key [BHP23]. This was followed within a couple of months by a second, more sophisticated analysis needing only six logins [HR23]. Then just a few months later, a third analysis requiring only two logins was published [AHMP23]. This evolution somewhat undermined MEGA's claim that their users were secure because, according to their logs, none of them had logged in as many as 512 times. Here we must also recognise MEGA, however, as being one of those vendors who made disclosure and remediation a positive experience.

Moreover, vendors have an incentive to "downplay" the significance of vulnerabilities, fearing negative market perception. A shift towards downplaying is often seen as the agreed-upon disclosure date approaches and the conversation moves from the security or development team to a more senior party, such as a CEO. The flip side is that the disclosing researchers have an interest to "play up" the vulnerability for obvious prestige reasons.

Scientific Community. Reception in the scientific community is typically both rather negative and exceptionally positive, with the two reactions occurring simultaneously. First, strong attacks or detailed proofs of high-profile targets routinely win best paper awards at security research venues, highlighting the community's appreciation for this sort of work. At the same time, it is rare for these works to find a home in more traditional cryptographic venues, e.g. the flagship venues of the International Association for Cryptologic Research (IACR). This is because many reviewers perceive this endeavour as lacking scientific significance, as expressed in reviews asking for comparative studies, lessons learned for the field, a higher degree of novelty, attacks with greater technical depth, new primitives, new security models, etc.

In an ideal world, every system relying on cryptography would be thoroughly analysed before release, drawing on the wealth of scientific literature available, and there would be no need for the study of cryptography in the wild. This does happen sometimes, for example, with TLS 1.3 or more recently when Apple engaged two teams of academic researchers to review their new post-quantum version of the iMessage protocol prior to its release. However, this is rarely the case. Our contention is that we, the scientific community, can gain a lot from observing cryptography in the wild. Not only do we obtain valuable examples for use in the classroom (answering in concrete ways questions like "Why is key separation really needed?"), but we may also begin to understand why cryptography is seemingly so hard to get right in practice, why developers so often get it wrong, and how we can design new cryptographic primitives and protocols so that they can be more safely consumed.

Indeed, if we consider science to be the process by which we get to understand the world, then a security analysis of a significant object in the world – say a protocol relied upon by millions of people – is then in and of itself a valid scientific result. The situation here is not at all dissimilar to many other sciences which feature branches labelled "theoretical", branches labelled "applied" and branches labelled "experimental", "empirical" or "observed". We consider the study of cryptography as it exists in the world as "observed cryptography" in that sense.

"The Public". Eventually, the results of the analysis are made public. If those results are "only" a formal security statement then news of this barely reaches even IT-security-focused practitioner circles. However, if vulnerabilities are also disclosed and if the object is sufficiently high-profile then this may well produce a short burst of interest with news reporting and social media threads. In our own practice, we typically reach out to a journalist before disclosure to give them a chance to get a more accurate technical account of what is and what is not broken and what that means. This aids with communicating the findings clearly. In addition, we may set up a special website containing high-level and less-technical discussions of the vulnerabilities, the disclosure process and patch status. This, of course, creates an additional burden on the research team. We regard the production of logos and stickers as highly optional.

The typical social media response to vulnerabilities is then some variant of "don't roll your own crypto". While this truism is, well, true – people not trained in the development of cryptography should not develop it, just like people who are not trained to design bridges should avoid that activity – it glosses over what allows someone to "roll their own": careful modelling and cryptographic design, formal security analysis, and an appreciation for secure cryptographic implementation. Instead of this simple scientific message, we are typically left with a simple measurement contest – "who is better at it" – which does not help to improve the general state of cryptographic solutions.

Moreover, despite the heavy moralising typically found on social media about the failings of vendors to roll cryptography, as far as we can tell, this "name and shame" approach has little lasting effect. In our own experience, such approaches do not harm the popularity of the relevant vendor.

5 Conclusion

We have described some of our experiences with and approaches to studying cryptography in the wild. We encourage the further development and refinement of this folklore methodology. As the scope of application of cryptography continues to broaden, it becomes ever more deeply embedded as a foundation of privacy, trust and security in our digital society. So we anticipate the topic of cryptography in the wild to be of perennial interest and value.

Acknowledgements

We thank Rikke Bjerg Jensen for helpful discussions.

References

ABD⁺15. David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 5–17. ACM Press, October 2015. 1

ABJM21. Martin R. Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková. Collective information security in large-scale urban protests: the case of hong kong. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 3363–3380. USENIX Association, August 2021. 2

ADHP16. Martin R. Albrecht, Jean Paul Degabriele, Torben Brandt Hansen, and Kenneth G. Paterson. A surfeit of SSH cipher suites. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1480–1491. ACM Press, October 2016. 2

AHMP23. Martin R. Albrecht, Miro Haller, Lenka Mareková, and Kenneth G. Paterson. Caveat implementor! Key recovery attacks on MEGA. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 190–218. Springer, Heidelberg, April 2023. 4

AMPS22. Martin R. Albrecht, Lenka Mareková, Kenneth G. Paterson, and Igors Stepanovs. Four attacks and a proof for Telegram. In *2022 IEEE Symposium on Security and Privacy*, pages 87–106. IEEE Computer Society Press, May 2022. 2, 2

APW09. Martin R. Albrecht, Kenneth G. Paterson, and Gaven J. Watson. Plaintext recovery attacks against SSH. In *2009 IEEE Symposium on Security and Privacy*, pages 16–26. IEEE Computer Society Press, May 2009. 2, 4

BBS23. Fabian Bäumer, Marcus Brinkmann, and Jörg Schwenk. Terrapin attack: Breaking SSH channel integrity by sequence number manipulation. *CoRR*, abs/2312.12422, 2023. 2

BHP23. Matilda Backendal, Miro Haller, and Kenneth G. Paterson. MEGA: Malleable encryption goes awry. In *2023 IEEE Symposium on Security and Privacy*, pages 146–163. IEEE Computer Society Press, May 2023. 4

BKN02. Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Authenticated encryption in SSH: Provably fixing the SSH binary packet protocol. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 1–11. ACM Press, November 2002. 2

DPS21. Luca De Feo, Bertram Poettering, and Alessandro Sorniotti. On the (in)security of ElGamal in OpenPGP. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2066–2080. ACM Press, November 2021. 1

GGK⁺16. Christina Garman, Matthew Green, Gabriel Kaptchuk, Ian Miers, and Michael Rushanan. Dancing on the lip of the volcano: Chosen ciphertext attacks on apple iMessage. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 655–672. USENIX Association, August 2016. 1

HDWH12. Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In Tadayoshi Kohno, editor, *USENIX Security 2012*, pages 205–220. USENIX Association, August 2012. 1

HR23. Nadia Heninger and Keegan Ryan. The hidden number problem with small unknown multipliers: Cryptanalyzing MEGA in six queries and other applications. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 147–176. Springer, Heidelberg, May 2023. 4

MBP+19. Jens Müller, Marcus Brinkmann, Damian Poddebniak, Hanno Böck, Sebastian Schinzel, Juraj Somorovsky, and Jörg Schwenk. "Johnny, you are fired!" - Spoofing OpenPGP and S/MIME signatures in emails. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019*, pages 1011–1028. USENIX Association, August 2019. 1

PST23. Kenneth G. Paterson, Matteo Scarlata, and Kien T. Truong. Three lessons from Threema: Analysis of a secure messenger, 2023. 2