Cryptography, Trust and Privacy: It's Complicated

Ero Balsa
Cornell Tech
USA
ero.balsa@cornell.edu

Helen Nissenbaum
Cornell Tech
USA
helen.nissenbaum@cornell.edu

Sunoo Park
Cornell Tech
USA
sep243@cornell.edu

ABSTRACT

Privacy technologies support the provision of online services while protecting user privacy. Cryptography lies at the heart of many such technologies, creating remarkable possibilities in terms of functionality while offering robust guarantees of data confidentiality. The cryptography literature and discourse often represent that these technologies *eliminate the need to trust* service providers, i.e., they enable users to protect their privacy even against untrusted service providers. Despite their apparent promise, privacy technologies have seen limited adoption in practice, and the most successful ones have been implemented by the very service providers these technologies purportedly protect users from.

The adoption of privacy technologies by supposedly adversarial service providers highlights a mismatch between traditional models of trust in cryptography and the trust relationships that underlie deployed technologies in practice. Yet this mismatch, while well known to the cryptography and privacy communities, remains relatively poorly documented and examined in the academic literature—let alone broader media. This paper aims to fill that gap.

Firstly, we review how the deployment of cryptographic technologies relies on a chain of trust relationships embedded in the modern computing ecosystem, from the development of software to the provision of online services, that is not fully captured by traditional models of trust in cryptography. Secondly, we turn to two case studies—web search and encrypted messaging—to illustrate how, rather than *removing* trust in service providers, cryptographic privacy technologies *shift* trust to a broader community of security and privacy experts and others, which in turn enables service providers to implicitly build and reinforce their trust relationship with users. Finally, concluding that the trust models inherent in the traditional cryptographic paradigm elide certain key trust relationships underlying deployed cryptographic systems, we highlight the need for organizational, policy, and legal safeguards to address that mismatch, and suggest some directions for future work.

CCS CONCEPTS

• **Software and its engineering**; • **Security and privacy → Cryptography**; *Systems security*; • **Social and professional topics → Computing / technology policy**;

KEYWORDS

privacy, cryptography, trust, assumptions

ACM Reference Format:
Ero Balsa, Helen Nissenbaum, and Sunoo Park. 2022. Cryptography, Trust and Privacy: It's Complicated. In *Proceedings of the 2022 Symposium on Computer Science and Law (CSLAW '22), November 1–2, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3511265.3550443

CSLAW '22, November 1–2, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9234-1/22/11…$15.00
https://doi.org/10.1145/3511265.3550443

1 INTRODUCTION

Modern cryptography enables remarkably versatile uses of information while simultaneously maintaining (partial) secrecy of that information. In addition to good old encryption, modern techniques such as secure multiparty computation and homomorphic encryption have opened a new realm of possibilities in privacy technologies, enabling the design and development of previously impossible—and sometimes seemingly paradoxical—combinations of functionality and confidentiality. Examples include, among others, anonymous credentials, which can enable verification without requiring identification [14, 17]; homomorphic encryption, which can enable cloud services that conceal user content from cloud providers [58]; and private information retrieval, which keeps user consumption of digital information (e.g., web search, media streaming) confidential from the provider [29, 40].

Being at the heart of modern privacy technologies, cryptography has pushed the limits of what is possible in terms of *data minimization*, a core principle in privacy engineering and *privacy by design* [31]. Cryptography is instrumental to the realization of data minimization strategies such as minimum data collection and minimum data exposure, which in turn result in minimization of the *need for trust* [32]. In theory, by shielding data flows from unauthorized access and prying eyes *by design*, implemented through code, rather than contractual agreements or privacy policies, cryptography enables privacy-preserving systems that do not rely on the goodwill or good behavior of the service provider or system administrators, thus minimizing the need to trust them with the protection of users' privacy.

Yet in spite of the powerful privacy properties that cryptographic privacy technologies promise, few of these technologies have seen adoption in practice. Whereas cryptography for security has been largely successful, holding the key (no pun intended) to secure transactions online, cryptography for privacy has not shared the same fate [41]. Cryptography for security may address important privacy concerns (e.g., HTTPS); however, few organizations have adopted the kind of privacy technologies that protect their customers or users against the organization itself, in theory ridding users of the need to rely on service providers to protect their privacy [23, 30]. In the same vein, despite the fact that outcries about privacy invasions and state and corporate surveillance have become a mainstay in

contemporary media, few users have taken matters into their own hands and adopted these technologies to protect their privacy.[1]

Reasons for this lack of adoption have long puzzled and drawn the interest of the academic community. Assumptions and hypotheses about poor usability, user and organizational unawareness, economic incentives and inefficiency are regarded as a complex network of interacting factors that prevent the adoption of these technologies, either by end users or service providers [42, 67, 70]. Among these factors, there is a key mismatch between how trust in cryptosystems is modeled to operate in theory and how it operates in practice [42]. In cryptographic privacy technologies design, service providers are often considered to be the main adversary. In practice, however, the deployment of cryptographic privacy technologies often depends on the very service providers these technologies are meant to protect against. Even more, privacy by design *dictates* that service providers embed privacy technologies in their system designs.

Take popular instant messaging service WhatsApp, for example. WhatsApp implements end-to-end encryption to protect users' message confidentiality, protecting WhatsApp users against privacy intrusions by WhatsApp itself. The traditional cryptographic model of end-to-end encryption, in contrast, considers that users themselves install an encryption tool to protect their messages against the adversarial service provider (here, WhatsApp), in addition to assuming that the *client* application that performs encryption is trustworthy and independent from the adversarial service provider. This model does not match reality today: WhatsApp, the service provider, deploys the encryption code and controls the client, i.e., the WhatsApp application itself.

In light of this significant mismatch, we seek to examine whether and how cryptographic privacy technologies, when adopted by service providers, effectively eliminate trust in service providers. We seek to understand the implications of having the same party against whom cryptographic privacy technologies are meant to defend users being the one implementing these technologies, and whether or not this apparent contradiction undermines the privacy protection these technologies are meant to provide.

Such model limitations and mismatches are generally well understood by cryptographers, and thus may seem to lack novelty to most in the cryptography community. Our message is not to simply reiterate the existence of these limitations or to criticize the traditional cryptographic approach. Rather, we argue that the lack of systematic exposition thereon, and the effective reliance on those modeling assumptions by deployed technologies, have important and understudied consequences in practice. Critical attention to modeling limitations when maintaining claims about deployed cryptosystems' properties is essential: without it, we risk conveying an oversimplistic and false sense of security (quite possibly at odds with lay users' well-founded skepticism [54]). Academic claims taken out of their expert context may further embolden service providers to simplistically reiterate those claims.[2]

Such claims can be found across many cryptographic applications today. End-to-end encryption, already mentioned above, promises that only the sender and receiver can read a message, typically without mention of the fact that the messaging provider is likely the one implementing the encryption. The selling point of homomorphic encryption (HE) is to shield users from the prying eyes of the cloud provider, typically without mention of the fact that the cloud provider would likely be the one implementing HE and running the whole infrastructure [64]. Similar claims of *trustlessness* abound and sustain the hype of "crypto" in the blockchain context, even when *"a few economic players—such as the largest mining pools and mining farms, as well as the most popular online exchanges and blockchain explorers—have become centralized points of failure and control in the governance of many blockchain networks* [21]. This, in turn, undermines the deployment of and investment in sociotechnical measures that need to be in place to address this mismatch.

1.1 Relation to prior work

The academic literature is awash with papers that examine reasons which may explain the poor adoption of cryptographic privacy technologies. In a 2013 paper, Narayanan charts the main lines of inquiry, which he broadly identifies as *human factors*, *developer's lack of training* and *mismatched incentives and models* [42]. Within human factors, a vast body of work has studied *usability* issues that may hinder user adoption, from the seminal and now classic 1999 *Why Johnny can't encrypt* study that exposed the inability of lay people to encrypt e-mail with PGP [70], to the more recent studies that investigate users' perceptions and understanding of end-to-end encryption on instant messaging apps [2, 3, 22, 54, 66, 71]. These later studies are illuminating because they call into question previous premises around usability, i.e., people still *do not* understand how encryption works, the threats encryption is meant to protect them from, or the extent to which it mitigates those threats, yet they *are* using encryption, if only because it is embedded by default in the services they use [2, 3, 22, 66, 71]. This work also casts a new light on Narayanan's point about mismatched models; namely, that *"crypto protocols treat service providers as adversaries, a model that's nonsensical in the modern computing environment"*. And yet some studies evaluate usability by how well users understand this model, e.g., Schroder et al. argue that because users do not compare keys with their conversation partners for verification purposes, they are *"very likely to fall for attacks [such as] central services [exchanging] cryptographic keys"* [54]. Such analysis makes sense within the traditional cryptographic model; in practice, when those central services control the app, they can undermine the key verification implementation in ways effectively invisible even to users who do try to verify. In other words, the traditional model would have users *trust* the implementation of the key verification process that displays the encryption/decryption keys, yet *mistrust* the service provider because it may tamper with their keys. While this seems to be a contradiction, it logically follows from the traditional model of cryptosystem design: user device security is out of scope, client-side

[1] Notably, some cryptographic privacy technologies cannot be unilaterally adopted by users, as they require service providers to deploy them (e.g., privacy technologies based on private information retrieval).

[2] To mention just one example, WhatsApp claims that *"WhatsApp's end-to-end encryption [...] ensures only you and the person you're communicating with can read or listen to what is sent, and nobody in between, not even WhatsApp. [...] WhatsApp has no ability to see the content of messages or listen to calls that are end-to-end encrypted"* [68]. As we examine below, WhatsApp does retain that ability, as it controls the code of the app and can deactivate end-to-end encryption or push a side-channel through an update in a snap of a finger.

encrypting software is assumed to be trustworthy, and adversaries trying to intercept and decrypt messages in transit (i.e. *eavesdroppers*) are a threat. In practice, however, both tampering with keys and the key verification process are equally legitimate threats, as the party providing the encryption software *is* the adversarial party.

There has been extensive work on the study of software vulnerabilities that undermine the theoretical properties of cryptosystems, including the introduction of intentional or unintentional backdoors, zero-days, or the possibility of side-channel attacks [7, 53]. Similarly, scholars have drawn attention to how recent shifts in computing infrastructure and software engineering practices may alter the way we think about security and privacy [6, 27]. Other work has pointed out at the mismatch between the role of and trust assumptions about service providers in privacy law and privacy engineering [23]. However, there has been significantly less attention to the consequences of having service providers —whom traditional cryptographic threat models treat as adversaries— control or assist in the adoption of cryptographic privacy technologies [9]; or more generally how traditional cryptographic assumptions and models do not accurately reflect the current paradigm of provision of digital services [27, 33]. We aim to bridge this gap.

1.2 Summary of contributions

Our contributions are as follows.

(1) In Sect. 2, we describe how cryptographic privacy guarantees depend on broader sociotechnical arrangements that traditional trust models in cryptography abstract away from. Privacy guarantees ultimately depend not only on cryptographic specifications and cryptographic security analyses, but on a complex chain of trust relationships inherent in the modern computing ecosystem, from the development of software to the provision of online services, including control over user devices and the software that runs on them.
(2) In Sect. 3, we show how instead of eliminating the need to trust providers, cryptographic privacy technologies shift trust to a broader community of security and privacy experts, in turn enabling service providers to implicitly *build* and reinforce trust relationships with their user base. To that end, we perform a comparative analysis examining two paradigms of privacy protection: one, based on trusted parties, the second, based on cryptographic privacy technologies. We consider the following case studies.
   – In Sect. 3.1 we examine private web search, illustrating through a hypothetical dialogue between a cryptographer and a skeptical user how conceptions of trust diverge between a cryptographer's implicit model and notion of trust and a popular, *commonsense* notion of trust.
   – In Sect. 3.2 we provide a comparative security analysis between encrypted instant messaging systems to illustrate how the service provider's adoption of end-to-end encryption shifts and distributes trust away from service providers to a wider community of security, privacy, and cryptography experts.
(3) In Sect. 4, we emphasize the need for legal and policy strategies to support the deployment of cryptographic privacy technologies, and chart some directions for future work.

2 CRYPTOGRAPHY AND TRUST

This section overviews: (1) cryptographic terminology and assumptions; (2) cryptographic and colloquial notions of trust, and where they may diverge; and (3) chains of trust that arise in cryptography as deployed in practice, throughout the process of implementation and deployment of a cryptographic specification.

2.1 Cryptographic assumptions

Cryptographic systems (henceforth, "cryptosystems") are defined by an algorithmic *specification* of how each of their components must be programmed to function—much like an architectural or engineering blueprint serves as a specification of how each component of a physical structure must be constructed. Cryptosystems are typically accompanied by a *security analysis* or *security proof*, which demonstrates that if certain specified *conditions* hold, then the cryptosystem as described in the specification provides certain security guarantees (e.g., related to confidentiality, integrity, or availability). Such conditions are usually called *assumptions*, because the security analysis assumes that they are true. Some of these assumptions are explicitly specified in the cryptography literature, such as computational hardness assumptions[3] or assumptions about the behavior of parties or devices;[4] others, however, are often left implicit in the cryptography literature, especially those common to all or most cryptographic protocols. A key, commonly implicit condition is that the implementation of a cryptosystem—including hardware devices, software, and human interaction—adheres to the specification of the cryptosystem. This condition encompasses that the software implementation of a cryptosystem is free from bugs and that hardware devices work as expected. Regrettably, implementation of cryptosystems and human error pose serious challenges in practice, and are a far more likely source of failure in cryptosystems than violation of explicitly specified cryptographic assumptions [7].

To be clear, cryptographic security proofs are generally precise and rigorous: all they claim to demonstrate is that certain security guarantees hold in the cryptosystem *as described in the specification*. Moreover, security proofs are a critical component of ensuring the trustworthiness of cryptosystems in practice. Our purpose here is not to criticize cryptographic analyses for omitting these implicit assumptions, or to characterize them as imprecise or flawed: they are not. Rather, we wish to highlight that in order to fully understand cryptographic security claims once they are deployed in practice, it is essential to account for the software engineering problems and human aspects of correctly implementing a cryptographic specification that are widely treated as out of the scope of the field of cryptography, and thus are not expressed in security guarantees and trust requirements as stated in the cryptography literature.[5]

Finally, it also bears note that *colloquial* discussion of cryptography more often omits assumptions and even states unconditional guarantees, e.g., "in end-to-end encryption, nobody but the sender and the receiver can read the content of their messages" or "in Bitcoin, it is not possible for anyone to tamper with data once it is written into the blockchain." While such simplified descriptions

[3] E.g., hardness of integer factorization or inverting a discrete logarithm.
[4] E.g., involved parties adhere to the protocol specification.
[5] This is reasonable given the extent of subfield specialization in modern computer science. Software engineering and human factors are not within cryptographers' expertise and other computer science subfields are better equipped to deal with them.

can be helpful and more accessible in some contexts, they can also mislead as to the nature and strength of the guarantees cryptography provides: no cryptographic tool provides truly unconditional guarantees in practice.[6] A nuanced and more realistic analysis of the security and privacy properties of a cryptosystem will always depend on the underlying (explicit or implicit) assumptions.

2.2 Models of trust

Trust is a key concept in security and privacy. The term features prominently in cryptographic terminology, in technical analyses of privacy tools in practice, and in public perceptions of privacy and technology. However, the meaning of *trust* as a term of art in cryptography is not always intuitive based on its lay usage. Below, we briefly examine some semantic nuances, and disambiguate how the term is used in this paper.

A commonsense notion of trust. Trust is an extraordinarily rich concept and covers a variety of relationships; it may refer to certain beliefs about others, attitudes toward them, or even certain feelings they elicit. Generally, when we trust others, we believe they will not act, intentionally, contrary to our interests; they won't harm, cheat, or betray us. What it means to trust others, however, is more than believing they will not intentionally harm us; it means placing our fates in their hands, knowing we are vulnerable to their actions. This is different from circumstances in which we believe others will not intentionally harm us because our interests have been secured against these harms. When reviewing conditions that philosophers and social scientists have claimed as important for engendering trust, including interaction history, reputation, known personal characteristics, relationships of mutuality and reciprocity, familiar social and professional roles, as well as contextual norms, it is not surprising that the formation of trust in digital societies, particularly in digital transactions, over digital networks, and on digital platforms is challenged [26, 34, 38, 57]. More specifically, not the mere formation of trust but the formation of reliably grounded trust is challenged by such conditions as lack of persistent identities that enable an accrual of histories and reputations, the breakdown of reciprocity, newfangled social and commercial actors who may or may not stand in adversarial relations to us, and uncertainty about prevailing norms in contexts in transition [11, 44, 45]. It probably also is not surprising that absent such conditions, people might be reluctant to trust even as they need to engage in transactions and communications that typically benefit from a penumbra of trust. As a consequence, in digital realms, security has come to stand in for trust so people will continue to engage. This can create a spiral downward for trust because, if Philip Pettit is right [49], making oneself vulnerable is a necessary condition for the formation of trust: if one does not have the opportunity to test others by exposing oneself to the possibility of harm, how does one learn who is and is not trustworthy, and for what? Common sense tells us not to expose ourselves to catastrophic harms, where the cost of learning who can and cannot be trusted may be fatal—literally or figuratively. In digital societies, we will need to count on the development of novel social markers of trust, to augment those that have evolved in the past as well as the heuristics each of us has developed through individual experiences.

In cryptography. The cryptography literature refers to parties as *trusted* if they are assumed, for the purpose of a security analysis, to adhere perfectly to behavior that is defined in the specification of a cryptosystem. Parties are described as *untrusted* if the security analysis assumes they may deviate from the specified behavior.[7] In a nutshell, cryptographic security guarantees hold as long as trusted parties follow the specification, regardless of whether untrusted parties deviate. Within the context of a given cryptosystem, cryptographers may refer to the *need to trust a party* (e.g., a service provider) if the party is *trusted* and hence the relevant security guarantees rely on that party's adherence to specified behavior.

The literature usually (implicitly) treats parties as monolithic entities encompassing human(s), software, and hardware devices. For example, if Alice is trusted, then not only she herself but also her software and devices are assumed to behave correctly according to specification; if Company X is trusted, then all the human processes in the company and the software and devices that the company controls are assumed to adhere perfectly to the specification.

A trusted party represents a *single point of failure* with respect to a given security guarantee because that party has the potential to unilaterally undermine a cryptosystem's security. However, entirely eliminating reliance on trusted parties is usually impractical: the security guarantees achievable tend to be quite limited when *all* parties are modeled as untrusted. One approach cryptographers use to avoid a single point of failure is to *distribute trust* across multiple parties: this means that security guarantees hold provided at least one of these parties, across whom trust is distributed, adheres to the cryptosystem specification.

The term *trust* in cryptography is thus effectively a shorthand for assumptions about adherence to prescribed behavior as described in the cryptosystem specification. Certainly, there is a relationship between the cryptographic term of art and its lay usage: parties that are untrusted or untrustworthy in the colloquial sense may be more likely to deviate from the specification. However, there is not a perfect match. Trustworthy parties in the cryptographic sense may be untrustworthy in the colloquial sense. As an example, a cryptosystem may enable two parties to confidentially communicate over an insecure channel, e.g., sending encrypted e-mail, so that their e-mail and Internet providers cannot eavesdrop on their conversation. In this scenario, sender and recipient (often called Alice and Bob) are *trusted* in the cryptographic model and the potential eavesdroppers are *untrusted*. However, even if Alice and Bob follow the cryptosystem's specification (i.e., in terms of the protocol and code they run on their computers) and are therefore effectively *trusted* in the cryptographic sense, they may be *untrusted* outside of the cryptosystem's scope. Indeed, whether Alice can or should trust Bob with sensitive information—i.e., whether he is trustworthy in the colloquial sense—is out of scope, and unrelated to the fact that Bob is a trusted party in the cryptographic analysis, e.g. Bob may

[6] Even the classic "one-time pad" encryption scheme, widely known for its simplicity and *unconditional security*, relies on deployment details for security in practice. The term "unconditional security" in this context refers to the absence of cryptographic hardness or trust assumptions within the traditional cryptographic model.

[7] In the cryptography literature, trusted parties are also commonly referred to as *honest*, and untrusted parties are also commonly referred to as *dishonest* or *malicious*—terminology that can appear to evoke far stronger value judgments and to an audience outside the field of cryptography. We use only the trusted/untrusted terminology in this paper, but note the common synonyms here for completeness.

follow the cryptosystem specification (thus remaining "trusted" in the cryptographic sense) and still choose to reveal the contents of his communication with Alice through an alternative channel, outside of the cryptosystem's purview (thus being "untrusted" in the colloquial sense). In addition, even highly trustworthy parties in the colloquial sense cannot be considered foolproof, as deviations from a specification may occur without malicious intent.

In sum, the cryptographic usage of the term *trust* should be taken for just what it is: a technical term of art that has a precise, narrow meaning that is useful within a specialized field. Cryptographic trust is about adherence to prespecified behavior, not about trustworthiness, interpersonal relationships, or other aspects relevant to the lay usage of the term.

Parameters of a trust relationship. Trust is relational: it is important *who* trusts *whom* and *for what*. Or, in the cryptographic context: *who* relies upon *whom* for the correct functioning of *what*. The cryptography literature is often lax about these parameters, leaving one or more of the three implicit. This approach can be rigorous within the cryptography community, in which there are established norms about the kinds of trust assumptions intended by certain set phrases, but can be confusing if the language reaches a broader audience. This paper aims to specify all three parameters wherever relevant.

Usage in this paper. We use the terms *trusted party*, *trust assumption*, and *distributing trust* as in the cryptographic terms of art. Beyond that, unless otherwise specified, we use the term *trust* informally, and aim to specify all the relevant parameters (i.e., who, whom, and for what) each time.

2.3 Cryptography in practice: from specification to implementation and deployment

For cryptosystems to be useful in practice, in addition to designing them and analyzing their security, it is necessary to implement, deploy and adopt them.

Implementation means writing code that follows the cryptosystem's specification, so it can be executed as a component of a larger system that provides a service.[8] For example, the specification of TLS needs to be written in code and embedded in browsers and servers so that users can request HTTPS-encrypted websites.

Implementation and deployment may introduce vulnerabilities and reliance on third parties that typically fall out of the scope of cryptosystems' design. As mentioned above, most of the cryptography literature treats as out of scope the sociotechnical arrangements that enable users to reasonably believe that the code and the broader system in which it is embedded are trustworthy. Looking at the broader picture of a cryptosystem's lifecycle from specification to implementation to deployment, however, the reliability and credibility of security guarantees promised by cryptosystems depends on a complex *chain of trust*, as we outline next.

Firstly, even if a cryptosystem *specification* is deemed to be secure, its *implementation* in executable code may not be. This highlights the importance of open standards, disclosed specifications, and disclosed-source code. Open standards and disclosure of specifications and code make it easier for stakeholders to agree on and, at least in principle, verify and test an implementation.

Secondly, even if specification and code are public, most intended users are unable to test and verify the cryptosystem directly, and must therefore rely on proxies of trust in order to assess the credibility of claimed cryptographic privacy guarantees. For example, they may depend on a cryptographer or developer whom they trust, or on institutional endorsements (e.g., a NIST standard, or statements from reputable research institutes or nonprofit organizations) to support a belief that a cryptosystem—and the system in which it is embedded—is secure. Of course, reputable institutions may still make mistakes or behave in bad faith [13, 53].

Thirdly, even expert cryptographers and software engineers are typically unable to verify with perfect certainty that a particular cryptosystem implementation is secure. Current computing infrastructure represents a tremendous challenge to the verification of code. Software applications are networked and distributed, as well as constantly updated: developers routinely rely on third-party code packages, need to deal with legacy code and systems, and service providers continuously push updates which, even when including security patches, make it harder to test and verify by the broader community [27]. Perhaps surprisingly, the problem of *software verification*—i.e., verifying that a given piece of source code conforms to a given specification of functionality, is an unsolved problem in computer science—save for very simple or specialized pieces of code [65]. Furthermore, due to complexities in the compilation processes that convert the human-readable source code written by developers to the machine-readable executable code[9] that can be run on devices, it is notoriously challenging to verify with certainty that a given piece of executable code corresponds exactly to a given piece of source code.[10]

Lastly, even a perfect implementation in code can be undermined by human error (or intentional misbehavior) in necessary human interactions with the deployed cryptosystem.

Chain of trust. To further illustrate these friction points and complex trust dependencies, let us provide a walk-through of a typical application's development lifecycle.

(1) *Cryptographer designs cryptosystem* The first step starts with a cryptosystem specification, i.e., an algorithmic description of how the cryptosystem works, the steps and parties involved. If the specification and its constituent components are published, other people can verify them, distributing trust among the crypto community who is able to verify the privacy guarantees the initial design provides. The community thus reaches sufficient confidence to trust the cryptosystem design, even if some vulnerabilities may be hard to detect.

[8] In this paper we focus on software implementations, but acknowledge that hardware security provides the *root of trust* for any software implementation. In fact, hardware security clearly highlights how cryptosystems which are secure *on paper* become vulnerable to physical attacks (e.g., *side-channels*) as their specification abstracts away from the physics of the hardware that enables computing [52]. Hence, while for illustration purposes we restrict our observations to software implementations and software infrastructure, similar observations could (and should!) be drawn to account for the hardware that supports them (e.g., devices, physical networks).

[9] Sometimes called object code.

[10] E.g., Mozilla has been working for years on providing such verifiability for its Firefox browser [62].

(2) *Developer writes code.* A developer *implements* the cryptosystem specification into code, often embedding it within a broader system in which the cryptosystem is but a component. If the code of the implementation is disclosed, this distributes trust among the community who is able and willing to examine and test it (e.g., developers and security experts). Even if such a community reaches sufficient confidence to deem the code secure, the developer may have (un)intentionally introduced a vulnerability or backdoor that is hard to detect.
(3) *Developer compiles code and creates an installation package.* Developers rarely use their own compilers. Instead they use compilers that third parties have written and publicly released. If the compiler is buggy or has been tampered with,it could introduce vulnerabilities into the executable code. The developer could also (1) deliberately use a compiler that introduces a backdoor or (2) simply compile different source code from the code it publishes as open source.[11] Hence, there is reliance on both developer and compiler.[12]
(4) *Developer publishes installation package,* e.g., Apple's App Store, Google Play, or any other websites and repositories for non-mobile applications (henceforth, "app store"). To verify that the application offered for download on the app store is the same application that the developer uploaded, most users rely on the app store host (e.g., Apple or Google). For those with more technical expertise, such reliance on the app store host is not necessary, as developers can provide digital signatures showing the authenticity of applications, which expert users can verify themselves.[13] Some non-expert users may also rely on experts that they consider reputable to check digital signatures, thereby reducing the extent of their reliance on the app store hosts.
(5) *User installs app.* Users must trust that the operating system (OS) installs the downloaded app correctly, i.e., as opposed to installing a malicious or weakened version of the app.
(6) *User opens and uses app.* The OS runs the installed app. Users must trust that the OS executes the app correctly, that it does not tamper with its execution or leaks details about it (e.g., OS effectively has control over the app on the user device). Incorrect functioning of the OS or device (see footnote 8) may arise from errors (e.g., software bugs, hardware glitches) as well as malicious compromise (e.g., malware such as NSO's Pegasus [12]).

Further trust requirements arise after initial installation:

– Developers can push updates, prompting users to install them, thus requiring anew trust assumptions as described in items 3 to 5 above.
– Threats related to the OS and device (namely, items 5 and 6) may arise at any time. A compromised operating system or device can undermine the security even of perfectly implemented, perfectly installed apps.

These chains of trust illustrate how in spite of the initial specification of a cryptosystem, its implementation and deployment require a coordinated interplay of several actors and processes, all of which must be relied upon not to introduce vulnerabilities that undermine the system in which the cryptosystem is embedded—even if the cryptosystem specification, analyzed in isolation, is provably secure. In other words, a reasoned belief that an implemented and deployed cryptosystem provides the security guarantees promised by the cryptosystem specification depends on many factors beyond the scope of the cryptosystem's specification and security analysis. With this in mind, it is arguably neither surprising nor misguided that many users conflate security and privacy issues, and neither understand nor trust the protection that cryptography may offer to them [22].

3 TWO CASE STUDIES

We present two case studies where we compare trust assumptions across two paradigms of privacy protection. On the one hand, we consider services where, from the cryptographic point of view, privacy protection depends on the service provider as a *trusted party*. This paradigm has been previously referred to in the privacy engineering literature as *privacy-by-policy*, *process-oriented privacy* or *soft privacy*, highlighting that privacy guarantees are derived from a promise or contract, rather than a technological intervention [20, 36, 59]. On the other hand, we consider services where, in cryptographic terms, the service provider is an *untrusted party* and privacy protection relies on a cryptosystem that the service provider itself implements. This paradigm has been previously referred to in the privacy engineering literature as *privacy-by-architecture*, or *data-oriented privacy* and *hard privacy*, to highlight that privacy protections are embedded into the system architecture *by design* [20, 36, 59]. In the language of cryptographers and privacy engineers, the second paradigm would *remove the need to trust* the service provider: a phrase that means that it is the architecture and code of the system itself that guarantee privacy protection, rather than the promise and goodwill of the service provider.

We select two essential services, web search and messaging, as our case studies. In the first, we confront a cryptographer's trust assumption logic with that of an imagined skeptical user that faces a choice between a search engine that promises privacy protections exclusively *by policy* and one that implements a cryptosystem to blind itself from seeing users' search queries. In the second, we compare popular instant messaging (IM) services that implement end-to-end encryption with an imagined IM service that promises the same privacy guarantees without relying on cryptography. In both cases, we illustrate how neither cryptosystem truly eliminates trust in the provider. Rather, trust is shifted or distributed, sustained by a complex web of trust relationships underlying the implementation and deployment of any privacy technology.

[11] Recall (see Section 2) that it is currently impractical to verify with perfect confidence that a complex piece of executable code is really the result of compiling a given piece of source code.
[12] Alternatively, users could compile the code themselves, eliminating the dependence on the developer; however, this is unrealistic for most people, and even then users need to trust the compilation process.
[13] Note that relying on digital signatures further shifts trust from the app store host to the distributed network of trust on which the relevant public-key infrastructure is built. How trust is construed within public-key infrastructures has been amply discussed elsewhere [4, 15, 39].

3.1 Web search

Web search allows users to find online resources (e.g., websites, images, videos) by querying a *search engine* with keywords describing what they seek.[14] While the most well known search engine is Google, we focus on DuckDuckGo (DDG) as a privacy preserving alternative. We compare DDG to a fictitious search engine that relies on state-of-the-art cryptography to embed analogous privacy protections into the system architecture.

Web search and privacy. Search queries can be highly sensitive and revealing, especially when collected over extended periods of time. Web search data may reveal gender, age, location, health condition, religious and political affiliation, sexual orientation, daily routine and frequented locations, life struggles, hobbies and interests, and much more [10, 37, 60, 63]. Moreover, when search engines such as Google use search query data to personalize search results and target users with advertisements, users may be served hyperpersonalized queries and discriminated against, among other harms that stem from user profiling [47, 61].

Google, DuckDuckGo and "CryptoSearch". Google collects users' search queries and has a vast network of trackers across the web to analyze and monetize users' web browsing behavior. Conversely, DDG presents itself as an alternative *"search engine that doesn't track you"* [25]. To that end, DDG *chooses* not to collect any user data [24]. Note the emphasis on *chooses*: DDG does not implement any technology that makes it difficult for it to collect this data, it simply decides not to implement any code that logs users' search queries. Hence, in cryptographic terms, DDG is a *trusted party*: users must rely upon DDG to honor its promise and protect their privacy by not collecting any of their search queries, now or in future.

Let us now consider a hypothetical cryptographic alternative: *"CryptoSearch"*, a search engine that "encrypts" user queries and search results so that only the user can decrypt them, hiding this sensitive information even from the search provider itself. Cryptographers have developed sophisticated techniques for *private information retrieval* (PIR) [5, 18] that, in theory, provide functionality similar to this. Currently, these PIR techniques are not developed enough to efficiently implement the complex algorithms of modern search engines. However, our inquiry concerns not efficiency but the differences in trust requirements between cryptographic privacy technologies and promise-based alternatives. As such, we put aside the efficiency question for the sake of a thought experiment: the hypothetical CryptoSearch provides an illuminating contrast with DDG in the discussion that follows.[15]

Why not settle with a trusted party? DDG's stated assurances are very similar to what one would expect from CryptoSearch, but instead of relying on cryptography to provide those assurances, DuckDuckGo *promises* its users that it will not collect their information.[16] From a cryptographers' viewpoint, DDG is a *trusted party*. Yet, how does DDG's trust-based guarantee fundamentally differ from the guarantees that a service like CryptoSearch would offer? Let us consider the following hypothetical exchange between a skeptical user and a cryptographer:

Skeptical User: *What's the difference between DDG and CryptoSearch? If I wish to prevent the search engine (and other third parties) from using my search queries, which should I prefer?*

Cryptographer: *With DDG, the privacy of your search data is under DDG's control and you are relying entirely on their promise not to use it in ways you don't want. With CryptoSearch, you're relying on a technical guarantee, not some humans' promise. As long as the cryptosystem that CryptoSearch uses is sound, it is guaranteed that CryptoSearch cannot misuse your data even if it wants to, because it cannot "see" your queries—even if the provider changes its mind, breaks its promise, or is coerced or hacked.*

Indeed, CryptoSearch seems to do away with the search engine provider as a trusted party. Eliminating trusted parties (or eliminating trust assumptions) is such a common goal in cryptography that it needs hardly any justification in academic papers: the received wisdom is that trusted parties provide poorer security and can beneficially be replaced by cryptographic guarantees. But now, let us consider how the rest of the conversation might play out.

Skeptical User: *But if CryptoSearch changes its mind, or is coerced or hacked, won't it just stop using the cryptography?*

Cryptographer: *Perhaps, but at least you would know that the code had changed, and probably detect that the cryptography had been removed.*

Skeptical User: *How would I be able to tell?*

Cryptographer: *If the client-side (user) code were disclosed-source, then you could check the code to see what had changed. Otherwise you could still reverse-engineer the client code, or do some traffic analysis; you might be able to detect the change that way. Of course you could always get a security expert to do those things for you.*

Skeptical User: *Right, I wouldn't be able to do those things myself. I'd have to place my trust in some humans' promise after all. I have to trust cryptographers like you, both when you say that cryptography provides all these strong guarantees, and when you say that a particular service I'm using correctly deploys that cryptography. So in the end I would choose between DDG or CryptoSearch depending on whom I trust more.*

The above dialogue is not intended to be realistic but rather to make a point; namely, to illustrate that the language of *eliminating trust* is misleading, both to users and to cryptographers; *shifting* or *distributing* trust would probably be more apt. To users, *shifting trust* is a more accurate description of the choice they face between these two services. And to cryptographers, the framing of *eliminating trust* makes it seem as if cryptographic solutions really remove the need for trust, as if a cryptosystem would be a replacement for trust relationships. This is why a cryptographer's response to a layperson may not address the latter's concerns.

[14] This describes a *text-based* search engine. Search engines don't have to be text-based, but most of them are.

[15] We could have selected a different cryptographic technology which is currently efficiently deployable, such as encrypted messaging. However, we found it helpful to reference a real-world technology that offers promise-based privacy, of which DuckDuckGo appears to be a relatively rare example.

[16] Some of DuckDuckGo's publicly advertised guarantees are reminiscent of what one would expect from cryptography. In an interview with Wired, founder Gabriel Weinberg declared *"We protect your search history, even from us"* [16].

In the next case study, we further examine how the adoption of cryptographic privacy technologies by service providers shifts and distributes trust, rather than eliminating it.

3.2 Instant messaging

Instant messaging (IM) refers to online services that enable near-instantaneous communication between two or more people, e.g., WhatsApp and Signal.[17]

IM and privacy. People use messaging services like WhatsApp and Signal to communicate with intimate partners, family, friends and coworkers, among others. As such, IM provides a medium for some of people's most private and sensitive conversations.

Online communication was originally devoid of strong privacy protections. Messages exchanged on early IM services, such as AIM or MSN's Messenger, could be easily intercepted in transit [50, 51]. A big shift took place with the normalization of TLS (previously SSL) encryption and XMPP, enabling service providers to encrypt in-transit communication, thus preventing network eavesdroppers from reading users' messages. However, TLS only encrypts messages between a user's device and the service provider, thus still enabling service providers to read all of their users' messages. To cryptographically protect message confidentiality against service providers themselves, *end-to-end encryption* (e2ee) is required.

End-to-end encryption. E2ee, as the name suggests, encrypts messages "end to end," meaning that only the communicating users hold the necessary keys to encrypt and decrypt the messages they send to each other. This means that an IM service provider like WhatsApp or Signal should not have the ability to decrypt their users' messages. Cryptographers would say that e2ee removes the need to trust the service provider for message confidentiality: since the provider has no access to user messages' content to begin with, there is no need to rely on the provider's good behavior to guarantee that the messages will not be misused or mishandled.

E2ee does not address all privacy concerns in IM. Most notably, e2ee does not protect metadata, this is, information about a user's communication other than message content. Metadata includes users' contact lists, who they talk to, how often, for how long, when users are online and their location, among other types of information. WhatsApp and Signal also differ in their treatment of metadata: Signal promises (much like DuckDuckGo) to not collect metadata, whereas WhatsApp states that it may retain and use metadata for business and other purposes.

E2ee has become the gold standard for message confidentiality in IM. However, the deployment of e2ee in Whatsapp and Signal challenges some of the assumptions upon which the privacy guarantees of e2ee depend. When the IM provider itself implements e2ee—namely, the very party e2ee is supposed to protect against—an interesting conundrum arises: does e2ee truly remove the need to trust the service providers? The short answer is *no*. To better understand how exactly the adoption of e2ee by service providers shapes trust assumptions supporting message confidentiality guarantees, below we present a comparative privacy analysis between three IM services: WhatsApp and Signal, that implement e2ee and enable it by default. WhatsApp and Signal implement very similar cryptographic protocols, but are organizationally very different: Signal is a nonprofit whereas WhatsApp is owned by Meta (formerly Facebook), and Signal's source code is published whereas WhatsApp's is not.[18] Alongside WhatsApp and Signal, we also consider "TrustMeIM", a hypothetical IM service that encrypts messages in transit (using TLS) but *not* end-to-end, yet promises not to examine or store user messages—much like DuckDuckGo's promises in the previous case study.

Comparative security analysis.

Message confidentiality against service providers. TrustMeIM has access to messages in plaintext by default, as they traverse their servers unencrypted. Thus, message confidentiality requires reliance on TrustMeIM's promise not to use or access users' messages. There is no technology preventing TrustMeIM from reading users' messages.

As for Signal and WhatsApp, message confidentiality requires that the client software (i.e., the messaging application running on the user's phone), has no unintended functionality that enables the transfer of data to the service provider. Note that it is not necessary to *break encryption* in order to undermine message confidentiality: a few extra lines of code are all that is needed to instruct the app to send back unencrypted messages to the server.

In this regard, there is a key difference between Signal and WhatsApp. While the two services may implement the same protocol, Signal's code is public and available for anyone to review, whereas WhatsApp's is not. Secrecy of code substantially hinders the ability to audit and verify the implementation.

In sum, by publishing all its code, Signal partially distributes trust among the community of cryptographers and security experts that have the willingness and commitment to audit and verify this code. Conversely, by keeping its source code closed, WhatsApp hinders such independent auditing, thereby requiring heightened trust in its e2ee implementation.

Surreptitious policy changes. In all three IM services, there is little protection against a service provider determined to implement surreptitious changes to undermine message confidentiality Such surreptitious breaches of message confidentiality might target particular users' communications (as in wiretaps), or undermine message confidentiality across large groups of users: the former, targeted surveillance, would be harder to detect than the latter, mass surveillance. There is however a significant difference in the cost and effort required to implement surreptitious changes to message confidentiality if e2ee is implemented. Because of this, e2ee enables providers to claim inability to comply with wiretap requests.

TrustMeIM would be the easiest to wiretap, since it already has access to user messages. Users would have no way of finding out if they have been subjected to a wiretap, barring someone (e.g., an insider from law enforcement, the courts, or the service provider) revealing its existence. In fact, in all three services, employees in the know about these wiretapping programs could expose their employers.

Wiretapping WhatsApp or Signal, on the other hand, would require either exploiting existing flaws in the implementation (intentional or not), or creating and distributing specific vulnerabilities

[17] See https://www.whatsapp.com and https://signal.org.

[18] WhatsApp is built on the Signal protocol with some modifications.

(e.g., by forcing an update on selected users or devices). Although regular users would unlikely be able to detect such vulnerabilities, the larger cryptography and security community could potentially expose such flaws and vulnerabilities. This, however, can be an arduous process, and requires a commitment from the community that is often predicated on free labor. Vulnerabilities can go undetected for years, and those who do discover them may choose not to disclose them to benefit from a lucrative zero-days market [1, 55]. In any case, even targeted surveillance would be more detectable than TrustMeIM's wiretapping.

Moreover, users' devices and the OS running on them would be vulnerable to exploits by external adversaries (i.e., beyond the service provider) such as law enforcement and governmental intelligence agencies. A case in point is NSO's Pegasus, a Trojan horse that exploited OS vulnerabilities to gain access to cellphones' text messages, calls and their microphones and cameras, among other capabilities [12]. Hence, the adoption of e2ee forces adversaries to attack end-user devices, to some extent preventing the convenience of just collecting all messages being relayed by the service provider. In this sense, e2ee also makes targeted surveillance much harder.

To conduct *mass* surveillance, the strategic vulnerabilities or backdoors required to conduct targeted surveillance (as described above) would have to be deployed at scale, for the vast majority of the user base. This would render the vulnerabilities more visible, and the ease of detecting them higher.

Overt policy changes. In all three IM services, there is no protection against the service provider changing its policy and deciding to undermine message confidentiality. TrustMeIM could decide to start collecting and analyzing users' messages; WhatsApp and Signal could decide (willingly or under pressure or coercion) to switch off e2ee.

Factors such as reputation, willingness to moderate content, and willingness to cooperate with law enforcement would likely influence providers' approaches to message confidentiality. Arguably, it could be easier for TrustMeIM to revert their policy of message confidentiality, as privacy expectations may already be lower. Conversely, the implementation of e2ee signals a stronger commitment to message confidentiality. Hence, breaking or switching off e2ee might be seen as a harsher reversal of that commitment, and one that would anger and disappoint those users who adopted Signal or Whatsapp for their deployment of e2ee.

Additional factors could influence the willingness of a provider to shift policies. Signal, as a non-profit, stands in opposition to the data-extractive business model that most tech companies rely on today. Signal stores minimal user data about its users and has been consistently pushing the state of the art in e2ee. Conversely, while Whatsapp provides e2ee encryption, its privacy policy reveals that collection of metadata is extensive and open to monetization by Meta. Meta's own reputation cannot but color user's privacy expectations. Moreover, Signal's reputation and far smaller userbase suggests that its userbase may have more stringent expectations of privacy than the far larger and more heterogeneous WhatsApp userbase [28]. Signal's userbase may thus be far more unwilling to tolerate a regressive change to Signal's privacy policy [28]. On the other hand, competition from other more privacy-protective IM apps could conceivably discourage Whatsapp from discontinuing e2ee (e.g., a recent change to its privacy policy to allow companies to connect with customers on Whatsapp led to a public outcry and some users leaving Whatsapp for alternatives such as Signal and Telegram [43]).

Discussion. Adoption of e2ee *by the provider* thus has an impact on trust that differs from the traditional cryptography model, in which users would rely on e2ee to protect the confidentiality of their communications against any unauthorized parties, e.g., the government, the Internet service provider (ISP), *or the IM provider*. Certainly, technologically savvy TrustMeIM users could adopt e2ee on their own to protect themselves against TrustMeIM. However, this model does not translate well when instead of users resorting to e2ee to protect themselves against an IM provider, it is the IM provider itself implementing e2ee. In this setting, e2ee does not perform the same function as it does in the traditional cryptography model. Certainly, it serves to protect message confidentiality, but trust must still be placed squarely on the service provider. An IM provider does not deploy e2ee so that users do not have to trust it with the confidentiality of their messages; rather, e2ee *reinforces* the trust that users place on the provider. By deploying e2ee, the provider is essentially telling users and the wider community: *"I have no desire to read, analyze, or misuse your messages, and I am committed to implementing technological measures to make it difficult for myself and others to do so"*. In situations such as misbehaving employees, faulty system security that exposes the service's servers to external attackers or intelligence agencies over-relying on communications surveillance, e2ee introduces an additional barrier to protect the confidentiality of users' communications. Similarly, e2ee works as a shield for IM providers to refuse turning user data to law enforcement or contributing to a surveillance infrastructure.

Still, e2ee does not ultimately prevent a truly adversarial service provider from breaching message confidentiality if it wishes to. Hence, e2ee has a performative function on top of the privacy guarantees it provides. It is a token of goodwill, a way for the IM provider to keep itself honest, tie its hands behind its back—a sort of checks and balances. Thus, whereas for a cryptographer e2ee is a tool to avoid placing trust in the provider, in practice, when adopted by a provider, it becomes a tool to build trust.

Yet the provider's adoption of e2ee still *distributes trust*, relying on a delicate balance of institutional trust and experts willing to verify the provider's claims. The analysis above in fact shows that trust in practice is far more complex and multidimensional than the concept of trust traditionally inherent in cryptosystems' design. Trust is in the eye of the beholder. Our analysis shows how the perspective of a layperson, a regular user, unable to verify or test code themselves, is radically different from a crypto, security or privacy expert. Regular users cannot be expected to rely on the same relationships of trust as experts do to obtain privacy guarantees online, partly because they are unlikely to understand the properties of the cryptosystem and likely to conflate the links along the chain of trust (see Sect. 2.3). Experts, on the other hand, still rely on a delicate balance of trust relationships within the broader community: not every crypto expert can test and verify every cryptosystem currently in use, so we implicitly rely on the expertise and know-how of others.

3.3 Lessons

According to privacy-by-design principles, service providers should adopt organizational and technological solutions to protect, by design and by default, users' privacy. Cryptographic privacy technologies are at the forefront of privacy engineering and privacy by design, often promising privacy protections against the service provider itself. A cryptographer would phrase this as *eliminating the trust* on the service provider to protect users' privacy.

The case studies above provide a comparison between a purely organizational approach to privacy protection (provider as a *trusted party*) and a cryptographic approach that implicitly considers the service provider as an *untrusted party* that users must be protected from. Whereas a cryptographer's promise would be that adoption of cryptographic privacy technologies eliminates the need to rely on the untrusted service provider, our analysis illustrates why this does not happen in practice, *especially* when the provider itself implements these technologies. Instead, cryptography shifts and shapes a new delicate balance of trust between users, (security, privacy, cryptography) experts and service providers that users themselves are likely to remain oblivious to.

This mismatch between a cryptographer's promise of *eliminating trust* is partly predicated in outdated computing and user paradigms: a paradigm where users control their devices and adopt privacy technologies by themselves to fend off an adversarial, untrusted service provider. In reality, the modern computing paradigm makes it incredibly difficult for users to adopt these technologies on their own: even if users did not trust WhatsApp's e2ee, users cannot add an encrypting plugin or extension to WhatsApp to encrypt their messages because of current mobile OS sandboxing restrictions.

Moreover, users cannot unilaterally adopt certain privacy technologies even if they wished to. A tool like CryptoSearch *requires* service provider cooperation, i.e., the service provider is the only one who can deploy it. And so the provider retains the power to undermine it. It is therefore crucial that complementary measures are in place to support cryptographic solutions. Privacy protection requires reliance on humans and organizations, whether that involves implementing cryptographic code or making sure service providers abide by their implementing promises. We explore some of these measures next.

4 PATHS FORWARD

In this section we explore a range of complementary measures, both technical and non-technical, to support the deployment of cryptographic privacy technologies. In essence, we ask:

What legal, organizational, technical, or other measures could support cryptographic privacy technologies where traditional modeling assumptions are in doubt?

While a thorough study of this question is beyond the scope of a single paper, we highlight the question for future research, and outline preliminary directions for further exploration. In particular, we hope to promote explicit and context-specific consideration of this question when analysing existing or planned deployments of cryptographic privacy technologies.

Firstly, we note that technical measures may help, but they alone are insufficient. Hence, organizational and legal measures that enhance the *transparency*, *accountability*, and *enforceability* of promised privacy protection—cryptographic or otherwise—are essential to bolster the reliability of privacy guarantees at the inevitable points where their realization relies on third-party (human and organizational) behavior. As such, the following discussion is lighter on technical measures than legal and organizational ones.

The rest of this section discusses technical, organizational, policy, and legal measures in turn. While not all of these will be a perfect fit for every situation in which a privacy technology is deployed, we believe most have broad applicability.

4.1 Technical and organizational safeguards

- Avoid unnecessary complexity in system design.[19]
- Publish (crypto)system specifications.
- Publish source code.
- Provide signed code, binary transparency, and reproducible builds wherever practicable.
- Encourage or commission independent code audits.
- Promote, develop and adopt open standards and interoperability [35, 56].
- Enforce robust access-control policies for insiders.
- Maintain detailed activity logs and audit them routinely.
- Establish incident reporting and investigation processes.
- Establish a bug bounty program, and set clear guidelines as to what discoveries will be considered a privacy bug.
- Openly commit to whistleblower protection policies.
- Publish clear policies about allowed data use and access.
- Publish clear policies about any commitments to continue providing privacy services into the future.

4.2 Legal safeguards

To further support the privacy properties that cryptosystems afford, organizations may choose to make them credibly legally enforceable. Making contractual commitments (e.g., in terms of service) to the organizational and policy safeguards described above is useful, but only a starting point. Contracts may be as ineffective as cryptography unless contract violations are evident to the party harmed, and that party has the resources and patience to pursue legal action against the violating party.

Thus, to ensure meaningful *private enforceability*: (1) organizations should implement transparency and auditing processes that render it likely that deviations from contractual commitments would be readily detectable and demonstrable (perhaps reinforced by technical measures); and (2) organizations should commit to clearly defined consequences and penalties in case of discovered violations (e.g., reviews and reports after the fact, curtailment of certain activities, and/or monetary payouts to affected users or charities). Additionally, to enhance enforceability beyond the defaults provided by the legal system, organizations may commit to honor such penalties and consequences through extrajudicial processes, reducing the likelihood of costly litigation as a barrier to enforcement. For example, organizations might pledge to award a "privacy bounty" to those who report deviations from the organization's stated privacy commitments, and/or to put funds in escrow with

[19] E.g., the phrase "Keep It Simple, Stupid" (KISS) refers to a design principle originating from the U.S. Navy in 1960, which has since gained popularity in software development and security. The underlying idea is that simpler systems work more reliably and can be more reliably secured, so unnecessary complexity should be avoided [19, 69].

a neutral arbiter who is charged with paying out privacy-related claims. Finally, some of the voluntary safeguards thus far mentioned may be overridden by government subpoenas or other mandates; to reduce the likelihood of such activities going undetected, organizations may choose to implement canaries [46].

A further benefit of clear commitments by private parties to organizational privacy practices—and of establishing credible mechanisms to make violations evident to parties harmed—is that the U.S. Federal Trade Commission (FTC) would then have jurisdiction to investigate and penalize violations of the stated privacy practices, as they would fall within the scope of "unfair and deceptive practices" as defined by the FTC Act.[20] Though regulatory regimes differ internationally, written commitments to privacy practices and evidence of violations may similarly facilitate oversight by consumer protection and data protection authorities in other countries.

The legal measures discussed thus far depend on voluntary adoption by organizations. While this limits their scope, it also means that they could be instated in the immediate term upon the initiative of individual organizations. New legislative or regulatory measures are likely to take much longer to establish, but could provide stronger safeguards. In general, productive legislative or regulatory directions should encourage or mandate the adoption of technical and organizational measures such as outlined above. A few examples follow.

- Obliging cryptographic service providers that meet certain requirements (e.g., a threshold in user base or revenue) to demonstrate heightened trustworthiness, such as by undergoing independent privacy audits.
- Requiring providers to publish the results of any independent privacy audits—or, at the very least, to disclose them to a competent government agency with authority to pursue further investigation, such as the FTC.
- Designating and providing resources to a government agency or other organization to enforce and conduct privacy audits.
- Allocating public funds so that demonstrations of trustworthiness, such as independent privacy audits, are accessible to all interested cryptographic service providers (e.g., subsidies for nonprofits).
- Establishing a heightened duty of care for cryptographic service providers above a certain size (perhaps similar to some existing *information fiduciary* proposals [8]).
- Suggesting enhanced penalties for otherwise actionable harm related to providers' misuse or misrepresentation of privacy technologies.
- Creating a private right of action, rooted in consumer protection law, that users (or classes of users or nonprofits advocating on behalf of users) could raise against providers for misuse or misrepresentation of privacy technologies and/or cryptography, even if no otherwise actionable harm can be proven. The elements of such a claim might be scoped around reasonable reliance and exposure to risk: (1) that users reasonably relied on providers not to misuse or misrepresent the technology; (2) that users placed themselves at significant privacy or other risk based on such reliance, would not have done so but for such reliance, and the providers' misuse or misrepresentation aggravated that risk.
- Allocating public funds and convening independent expert bodies towards the development of open standards.
- Encouraging the use of, and interoperation via, open standards, e.g., by offering capped or reduced penalties for security or privacy incidents for organizations that use open standards according to software engineering and security best practices.

[20] Federal Trade Commission Act, 15 U.S.C. §41–58, at §45.

4.3 Discussion

Our emphasis on *providers* in the above is significant. The burden should be on providers, not users, to ensure privacy technologies are understood and used safely, and not to offer unreasonably risky products. Much as in consumer protection law, providers are better informed, and generally are positioned to be the "lesser cost avoider". Furthermore, where providers may be naturally incentivized to cut corners, they would often be able to do so in a way that would be difficult or impossible for consumers to detect; therefore, placing liability and auditing obligations on providers is essential to effectively achieving the privacy protections sought.

However, an important counterbalancing consideration when designing mandates for providers is to promote the broader policy goal underlying strengthened privacy regulation: namely, to encourage the adoption of privacy technologies. Imposing additional burdens on organizations that voluntarily adopt cryptographic privacy technologies could backfire by discouraging the adoption of such technologies in the first place—especially where, as now, existing privacy regulation does not mandate privacy protection at the level that cryptographic privacy technologies provide. Hence, the measures outlined above should be considered in combination with complementary measures that encourage (or at least do not make more difficult) the adoption of privacy technologies. These could include broader privacy protection requirements for more companies to "level the playing field" for those providing cryptographic protections, or other types of incentives, such as tax breaks or safe harbors for adopting organizations that comply with one or more measures listed above (e.g., published code and independent audits), and meet security best practices in their deployment.

Providers of supporting infrastructure for cryptographic privacy technologies, such as app stores and code repositories, are also critical to these technologies' trustworthiness, as discussed in detail in Sect. 2.2. Though the discussion above refers mainly to privacy technology providers, many of the suggested measures may be productively applied to supporting infrastructure providers too. Furthermore, since providers of centralized supporting infrastructure can sometimes function as effective gatekeepers to market entry by privacy technology providers, regulation to encourage (or not make more difficult) the adoption of privacy technology should also discourage or penalize infrastructure providers from imposing disproportionate barriers for privacy technology providers. Looking ahead longer term, these problems would be alleviated by reducing the extent of technological dependence and gatekeeper roles in the "chain of trust" (discussed in Sect. 2.2) that technologies today tend to rely on.

Finally, added obligations on *users* should be avoided. Naturally, if users cause otherwise actionable harm through the use of a privacy technology (e.g., harassment via private messaging), then they may be properly liable under existing law. However, there should be no liability for users for simply using a provider's product or service in a manner not anticipated or endorsed by the provider—even if such use entails privacy risk. In other words, companies should not be in a position to shift liability for a privacy incident to the user by claiming that the user "misused" a privacy technology which, if used correctly, would not have led to the incident. Not only does this approach again mirror consumer protection law, it also promotes essential research that may involve experimenting with privacy technologies: such research may discover weaknesses in existing technologies, from which we can learn, and is necessary to improving computer and Internet security over time [48].

5 CONCLUSION

Cryptography is at the heart of modern privacy technologies, enabling stringent data confidentiality properties. The cryptography literature often informally claims that cryptography eliminates the need for trust. In other words, cryptography enables users to no longer need to trust service providers to protect their privacy, as these tools protect them even *against* service providers. The narrative goes that privacy is thus protected by design and by default, embedded into system design, automatically enforced by code.

And yet, despite their apparent promise, cryptographic privacy technologies have seen limited adoption in practice, while, at the same time, the most popular cryptographic privacy technologies have been implemented by the very service providers these technologies purportedly protect users from. Hence, the adoption of privacy technologies by supposedly adversarial service providers highlights a mismatch between traditional models of trust in cryptography and the trust relationships that underlie deployed technologies in practice. While this mismatch is well known to experts in the cryptography community, its consequences have been understudied and understated in practice, emboldening service providers to peddle misleading claims and fostering a false sense of security.

This paper has sought to highlight, document and better understand this mismatch. To that end, we have provided a description of the divergence between conceptualizations of trust in cryptography and a *commonsense* notion of trust. We have described how cryptographic privacy guarantees operate within a limited model that abstracts away from complex relationships of trust in the modern computing ecosystem, from the development of software to the provision of online services, including control over user devices and the software that runs on them. Moreover, we have performed a comparative analysis between two paradigms of privacy protection: one, based on trusted parties, the second, based on cryptographic privacy technologies. This comparative analysis enables us to illustrate how, far from removing trust in the provider, adopting cryptographic privacy technologies shifts trust to a broader community of security and privacy experts, enabling in turn service providers to implicitly *build* and reinforce trust relationships with its user base.

Lastly, these observations have important implications for policy making and the broader regulation of privacy online. While a technocentric or cryptocentric approach may suggest that cryptographic privacy technologies are *the* solution to complex sociotechnical problems, the deployment of these technologies requires a robust arrangement of complementary technical and non-technical measures to support them. This paper has charted a few tentative steps in this direction.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their thoughtful and helpful reviews, which greatly contributed to improve this paper. This material is based upon work supported by DARPA under Agreement No. HR00112020021. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA. SP's research is additionally supported by a Computing Innovation Fellowship, funded by the National Science Foundation under Grant #2127309 to the Computing Research Association.

REFERENCES

[1] Lillian Ablon and Andy Bogart. 2017. *Zero days, thousands of nights: The life and times of zero-day vulnerabilities and their exploits.* Rand Corporation.
[2] Ruba Abu-Salma, Elissa M Redmiles, Blase Ur, and Miranda Wei. 2018. Exploring User Mental Models of End-to-End Encrypted Communication Tools. In *8th USENIX Workshop on Free and Open Communications on the Internet (FOCI 18).*
[3] Ruba Abu-Salma, M Angela Sasse, Joseph Bonneau, Anastasia Danilova, Alena Naiakshina, and Matthew Smith. 2017. Obstacles to the adoption of secure communication tools. In *2017 IEEE Symposium on Security and Privacy (S&P).* IEEE, 137–153.
[4] Carlisle Adams and Steve Lloyd. 1999. *Understanding public-key infrastructure: concepts, standards, and deployment considerations.* Sams Publishing.
[5] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. 2016. XPIR: Private information retrieval for everyone. *Proceedings on Privacy Enhancing Technologies* 2016, 2 (2016), 155–174.
[6] Jay Aikat, Aditya Akella, Jeffrey S. Chase, Ari Juels, Michael K. Reiter, Thomas Ristenpart, Vyas Sekar, and Michael Swift. 2017. Rethinking security in the era of cloud computing. *IEEE Security & Privacy* 15, 3 (2017), 60–69.
[7] Ross J. Anderson. 1994. Why Cryptosystems Fail. *Commun. ACM* 37, 11 (1994), 32–40. https://doi.org/10.1145/188280.188291
[8] Jack Balkin. 2014. Information Fiduciaries in the Digital Age. Balkinization. Online at https://balkin.blogspot.com/2014/03/information-fiduciaries-in-digital-age.html. Last retrieved on Aug 15, 2022.
[9] Ero Balsa, Filipe Beato, and Seda Gürses. 2014. Why Can't Online Social Networks Encrypt?. In *Proceedings of W3C Workshop Privacy UserCentric Controls.*
[10] Michael Barbaro and Tom Zeller. 2006. A face is exposed for AOL searcher no. 4417749. Online at https://www.nytimes.com/2006/08/09/technology/09aol.html. Last retrieved on Mar 14, 2022.
[11] Adam Barth, Anupam Datta, John C. Mitchell, and Helen Nissenbaum. 2006. Privacy and contextual integrity: Framework and applications. In *2006 IEEE symposium on security and privacy (S&P'06).* IEEE, 15–pp.
[12] Ronen Bergman and Mark Mazzetti. 2022. The Battle for the World's Most Powerful Cyberweapon. Online at https://www.nytimes.com/2022/01/28/magazine/nso-group-israel-spyware.html. Last retrieved on Mar 7, 2022..
[13] Daniel J Bernstein, Tanja Lange, and Ruben Niederhagen. 2016. Dual EC: A standardized back door. In *The New Codebreakers.* Springer, 256–281.
[14] Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. 2009. Anonymous credentials on a standard Java Card. In *Proceedings of the 16th ACM conference on Computer and communications security.* 600–610.
[15] Matt Blaze, Joan Feigenbaum, and Jack Lacy. 1996. Decentralized trust management. In *Proceedings 1996 IEEE Symposium on Security and Privacy (S&P).* IEEE, 164–173.
[16] Matt Burgess and Victoria Woollaston-Webber. 2017. DuckDuckGo: what is it and how does it work? Online at https://www.wired.co.uk/article/duckduckgo-anonymous-privacy. Last retrieved on Mar 13, 2022..
[17] David Chaum. 1985. Security without identification: Transaction systems to make Big Brother obsolete. *Commun. ACM* 28, 10 (1985), 1030–1044.
[18] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. 1998. Private Information Retrieval. *J. ACM* 45, 6 (1998), 965–981. https://doi.org/10.1145/293347.293350

[19] Tom Dalzell. 2009. *The Routledge Dictionary of Modern American Slang and Unconventional English*. Taylor & Francis. https://books.google.com/books?id=5F-YNZRv-VMC&pg=PA595 p. 595.

[20] George Danezis. 2007. Introduction to privacy technology. *Katholieke University Leuven, COSIC: Leuven, Belgium* (2007).

[21] Primavera De Filippi, Morshed Mannan, and Wessel Reijers. 2020. Blockchain as a confidence machine: The problem of trust & challenges of governance. *Technology in Society* 62 (2020), 101284.

[22] Sergej Dechand, Alena Naiakshina, Anastasia Danilova, and Matthew Smith. 2019. In encryption we don't trust: The effect of end-to-end encryption to the masses on user perception. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 401–415.

[23] Claudia Diaz, Omer Tene, and Seda Gürses. 2013. Hero or villain: The data controller in privacy law and technologies. *Ohio St. LJ* 74 (2013), 923.

[24] DuckDuckGo. 2012. Privacy policy. https://duckduckgo.com/privacy. Last retrieved on Mar 13, 2022.

[25] DuckDuckGo. 2022. About. https://duckduckgo.com/about. Last retrieved on Aug 11, 2022.

[26] Francis Fukuyama. 1995. *Trust*. New York: Free Press Paperbacks.

[27] Rafa Galvez and Seda Gürses. 2018. The Odyssey: Modeling privacy threats in a brave new world. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 87–94.

[28] Matthew Green. 2020. Why is Signal asking users to set a PIN, or "A few thoughts on Secure Value Recovery". Online at https://blog.cryptographyengineering.com/2020/07/10/a-few-thoughts-about-signals-secure-value-recovery/. Last retrieved on Mar 7, 2022..

[29] Trinabh Gupta, Natacha Crooks, Whitney Mulhern, Srinath Setty, Lorenzo Alvisi, and Michael Walfish. 2016. Scalable and private media consumption with Popcorn. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI'16)*. 91–107.

[30] Seda Gürses and Claudia Diaz. 2013. Two tales of privacy in online social networks. *IEEE Security & Privacy* 11, 3 (2013), 29–37.

[31] Seda Gürses, Carmela Troncoso, and Claudia Diaz. 2011. Engineering privacy by design. *Computers, Privacy & Data Protection* 14, 3 (2011), 25.

[32] Seda Gürses, Carmela Troncoso, and Claudia Diaz. 2015. Engineering privacy by design reloaded. In *Amsterdam Privacy Conference*, Vol. 21.

[33] Seda Gürses and Joris Van Hoboken. 2018. Privacy after the agile turn. In *The Cambridge Handbook of Consumer Privacy*, E. Selinger, J. Polonetsky, and O. Tene (Eds.). Cambridge University Press, 579–601.

[34] Russell Hardin. 1993. The street-level epistemology of trust. *Politics & society* 21, 4 (1993), 505–529.

[35] Matthew Hodgson. 2022. Interoperability without sacrificing privacy: Matrix and the DMA. Online at https://matrix.org/blog/2022/03/25/interoperability-without-sacrificing-privacy-matrix-and-the-dma. Last retrieved on Aug 11, 2022..

[36] Jaap-Henk Hoepman. 2014. Privacy design strategies. In *IFIP International Information Security Conference*. Springer, 446–459.

[37] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. 2007. "I know what you did last summer" — Query logs and user privacy. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. 909–914.

[38] Roderick M Kramer. 1999. Trust and distrust in organizations: Emerging perspectives, enduring questions. *Annual review of psychology* 50, 1 (1999), 569–598.

[39] Dimitrios Lekkas. 2003. Establishing and managing trust within the public key infrastructure. *Computer Communications* 26, 16 (2003), 1815–1825.

[40] Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. 2016. XPIR: Private information retrieval for everyone. *Proceedings on Privacy Enhancing Technologies* 2016 (2016), 155–174.

[41] Arvind Narayanan. 2013. What happened to the crypto dream?, part 1. *IEEE security & privacy* 11, 2 (2013), 75–76.

[42] Arvind Narayanan. 2013. What happened to the crypto dream?, part 2. *IEEE Security & Privacy* 11, 3 (2013), 68–71.

[43] Lily Hay Newman. 2021. WhatsApp's New Privacy Policy Just Kicked In. Here's What You Need to Know. Online at https://www.wired.com/story/whatsapp-privacy-policy-facebook-data-sharing/. Last retrieved on Mar 7, 2022..

[44] Helen Nissenbaum. 2001. Securing trust online: Wisdom or oxymoron? *BUL Rev.* 81 (2001), 635.

[45] Helen Nissenbaum. 2004. Will Security Enhance Trust online, or supplant it? In *Trust and distrust within organizations: Emerging perspectives, enduring questions*, R. Kramer and K. Cook (Eds.). Russell Sage Publications, 155–188.

[46] Kurt Opsahl. 2014. Warrant Canary Frequently Asked Questions. https://www.eff.org/deeplinks/2014/04/warrant-canary-faq.

[47] Eli Pariser. 2011. *The filter bubble: How the new personalized web is changing what we read and how we think*. Penguin.

[48] Sunoo Park and Kendra Albert. 2020. A Researcher's Guide to Some Legal Risks of Security Research. A joint publication of the Cyberlaw Clinic at Harvard Law School and the Electronic Frontier Foundation.

[49] Philip Pettit. 1995. The cunning of trust. *Philosophy & Public Affairs* 24, 3 (1995), 202–225.

[50] Scarlet Pruitt. 2003. AOL adds encryption to its corporate IM. Online at https://www.computerworld.com/article/2571336/aol-adds-encryption-to-its-corporate-im.html. Last retrieved on Mar 14, 2022..

[51] John Rittinghouse and James F. Ransome. 2005. *IM Instant Messaging Security*. Elsevier.

[52] Masoud Rostami, Farinaz Koushanfar, and Ramesh Karri. 2014. A primer on hardware security: Models, methods, and metrics. *Proc. IEEE* 102, 8 (2014), 1283–1295.

[53] Bruce Schneier, Matthew Fredrikson, Tadayoshi Kohno, and Thomas Ristenpart. 2015. Surreptitiously weakening cryptographic systems. *Cryptology ePrint Archive* (2015).

[54] Svenja Schröder, Markus Huber, David Wind, and Christoph Rottermanner. 2016. When Signal hits the fan: On the usability and security of state-of-the-art secure mobile messaging. In *European Workshop on Usable Security. IEEE*. 1–7.

[55] Guido Schryen. 2011. Is open source security a myth? *Commun. ACM* 54, 5 (2011), 130–140.

[56] Ross Schulman. 2022. We Don't Have to Sacrifice Encryption to Achieve Messaging Interoperability. Online at https://www.newamerica.org/oti/blog/we-dont-have-to-sacrifice-encryption-to-achieve-messaging-interoperability/. Last retrieved on Aug 11, 2022.

[57] Adam B. Seligman. 1997. *The Problem of Trust*. Princeton University Press.

[58] Sujoy Sinha Roy, Furkan Turan, Kimmo Jarvinen, Frederik Vercauteren, and Ingrid Verbauwhede. 2019. FPGA-based high-performance parallel architecture for homomorphic computing on encrypted data. In *2019 IEEE International symposium on high performance computer architecture (HPCA)*. IEEE, 387–398.

[59] Sarah Spiekermann and Lorrie Faith Cranor. 2008. Engineering privacy. *IEEE Transactions on software engineering* 35, 1 (2008), 67–82.

[60] Seth Stephens-Davidowitz. 2017. Everybody lies: how Google search reveals our darkest secrets. Online at https://www.theguardian.com/technology/2017/jul/09/everybody-lies-how-google-reveals-darkest-secrets-seth-stephens-davidowitz. Last retrieved on Mar 13, 2022..

[61] Latanya Sweeney. 2013. Discrimination in online ad delivery. *Commun. ACM* 56, 5 (2013), 44–54.

[62] Gregory Szorc. 2018. Deterministic Firefox Builds. https://gregoryszorc.com/blog/2018/06/20/deterministic-firefox-builds.

[63] Omer Tene. 2008. What Google knows: Privacy and Internet search engines. *Utah L. Rev.* (2008), 1433–1492.

[64] Marten Van Dijk and Ari Juels. 2010. On the Impossibility of Cryptography Alone for Privacy-Preserving Cloud Computing. In *5th USENIX Workshop on Hot Topics in Security (HotSec'10)*.

[65] Hans van Vliet. 2008. *Software Engineering: Principles and Practice* (3 ed.). Wiley.

[66] Elham Vaziripour, Justin Wu, Mark O'Neill, Daniel Metro, Josh Cockrell, Timothy Moffett, Jordan Whitehead, Nick Bonner, Kent Seamons, and Daniel Zappala. 2018. Action needed! Helping users find and complete the authentication ceremony in Signal. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. 47–62.

[67] Ari Ezra Waldman. 2021. *Industry Unbound: The Inside Story of Privacy, Data, and Corporate Power*. Cambridge University Press.

[68] Whatsapp. 2021. About end-to-end encryption. Online at https://faq.whatsapp.com/791574747982248/. Last retrieved on Aug 3, 2022.

[69] David A. Wheeler. 2015. *Secure Programming HOWTO* (3.72 ed.). 84 pages.

[70] Alma Whitten and J. Doug Tygar. 1999. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *USENIX Security Symposium*, Vol. 348. 169–184.

[71] Justin Wu and Daniel Zappala. 2018. When is a tree really a truck? Exploring mental models of encryption. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. 395–409.