One Protocol to Rule Them All? On Securing Interoperable Messaging

Jenny Blessing[1] and Ross Anderson[1,2]

[1] University of Cambridge
[2] University of Edinburgh

{first.last}@cl.cam.ac.uk

Abstract. European lawmakers have ruled that users on different platforms should be able to exchange messages with each other. Yet messaging interoperability opens up a Pandora's box of security and privacy challenges. While championed not just as an anti-trust measure but as a means of providing a better experience for the end user, interoperability runs the risk of making the user experience worse if poorly executed. There are two fundamental questions: how to enable the actual message exchange, and how to handle the numerous residual challenges arising from encrypted messages passing from one service provider to another – including but certainly not limited to content moderation, user authentication, key management, and metadata sharing between providers. In this work, we identify specific open questions and challenges around interoperable communication in end-to-end encrypted messaging, and present high-level suggestions for tackling these challenges.

1 Introduction

Users of end-to-end encrypted (E2EE) messaging services have long existed in a world where they need to use the same service as another user in order to communicate. A Signal user can only talk to other Signal users, an iMessage user can only use iMessage to communicate with other iPhone users, and so on. Platform interoperability promises to change this: the vision is that a user of a messaging service would be able to use their platform of choice to send a message to a user on a different service — following the precedent of email and SMS. Proponents of this kind of open communication have argued that it will benefit both the end user and the market for services. If users can message each other using their preferred service, they can enjoy their user experience of choice, and if there is less pressure to use a service simply because others use it, this can eliminate network effects and market monopolies.

An interoperability mandate for end-to-end encrypted messaging systems is no longer hypothetical: the European Union's Digital Markets Act (DMA) came into force in November 2022, and Article 7 requires that the largest messaging platforms (termed "gatekeepers" by the DMA) allow users on smaller messaging platforms to communicate directly with users on the large platforms [26].[3] The mandate applies only to the gatekeepers, with any non-gatekeeper platforms free to choose whether they wish to interoperate with other platforms. In accordance with the DMA, the gatekeepers cannot deny any "reasonable" request. Notably, it leaves the technical implementation details for the platforms to determine. In the U.S., the 2021 ACCESS Act proposed similar requirements but has yet to make any headway in Congress.

In this paper, we survey and explicitly articulate the security and privacy trade-offs inherent to any meaningful notion of interoperability, focusing primarily on the supporting aspects of E2EE communication which are largely agnostic to the actual method of message exchange. Designing a system capable of securely encrypting and decrypting messages and associated data across different service providers raises many thorny questions and practical implementation compromises.[4]

We outline what current solutions exist and where existing protocols fall short, and propose high-level solutions for tackling some of these challenges. For the sake of the discussions that follow, we assume platforms make a genuine effort to develop a system that emphasizes security and usability, though in practice they may degrade the user experience for interoperability, whether as a matter of necessity (WhatsApp has more features than Signal) or choice (to maintain some degree of customer lock-in). But as we will discuss, these challenges exist even if platforms make real efforts to open up their systems.

We argue that while it is possible to achieve interoperable end-to-end encrypted communications with contemporary messaging services, this will require numerous new protocols and processes, both cryptographic and human, to maintain reasonable levels of security and usability. The conceptual simplicity of messages passing back and forth between services belies the difficulty of the problem. Interoperability doesn't just mean co-opting existing cryptographic protocols so that one service provider can pass messages along to another – it encompasses the many supporting features and protocols that make up contemporary E2EE applications. The resulting complexity of the system may inherently compromise the level of security due to the increased number of moving parts, just as key escrow mechanisms endanger cryptography even if the escrow keys are kept perfectly secure.

The DMA includes a purported safeguard that the "level of security, including the end-to-end encryption, must be maintained" in an interoperable service, but this raises as many questions as it answers. "Level of security" goes well beyond the mere fact of using an end-to-end key exchange protocol. A platform

[3] The European Commission will not officially designate which platforms will be considered gatekeepers (and therefore fall under the mandate) until September 2023, but it is reasonable to assume this will include WhatsApp and iMessage at the least.

[4] Messaging interoperability raises numerous systems challenges, such as the latency impact of transferring messages between service providers when users expect communication to be instantaneous, particularly with audio/video calls, impact of bridging on mobile device battery life, etc. We limit our scope to challenges with direct security impact.

may use a proprietary E2EE protocol that does not provide forward secrecy, a de facto standard in preventing compromise of past communications [15], does not regularly rotate encryption keys, or neglects various other cryptographic guarantees. Until recently, the Swiss messaging app Threema openly had no forward secrecy at the E2EE layer [85], a design which led to multiple security problems [68]. Will these be considered valid reasons for a gatekeeper to deny a request to interoperate?

How would a gatekeeper verify the requesting service's encryption protocol, along with the authentication and content moderation schemes used? Will they need to take the requesting service's word for it, or else invest the resources to do a proper security audit? All widely-used E2EE messaging services have significant differences in both protocol and implementation, including fundamentally different design decisions impacting security and usability [25]. For instance, WhatsApp and Signal, despite both being based on the Signal protocol, handle key changes when a message is in flight differently [40]. When a recipient's keys change in the course of a conversation (e.g., because they uninstalled the app), Signal discards any messages sent after the change, while WhatsApp chooses to deliver them once the recipient comes back online. Both designs are legitimate [89, 64].

The greatest challenges are non-technical: interoperability will require competing platforms to cooperate and communicate, both in the current design phase and after any agreed-upon interoperable scheme has been deployed. Each service provider will need to trust the others to provide authentic key material, enforce certain spam and abuse policies, and generally to develop secure software with minimal bugs. A vulnerability or outage in one service now propagates to all other services with which it interoperates. Existing cryptographic protocols mitigate, but do not eliminate, these trust requirements. There is simply no getting around the fact that interoperability represents a dramatic expansion in the degree of trust a user will need to place not only in their own messaging service but also in any used by their communication partners, a point to which we will return throughout the paper. Much of the rhetoric around messaging interoperability at a recent European Commission-hosted stakeholder workshop [27] and elsewhere calls to mind the quip "if you think cryptography is your solution, you don't understand your problem."[5]

As service providers begin to make plans to comply with the DMA by the March 2024 deadline, it is important to tackle open questions now that providers are grappling with them, rather than waiting until platforms have invested a major effort in developing new systems and protocols. The stated policy goals of interoperability are to improve the user experience and decrease network effects, yet a poor execution risks doing the opposite. Open communication by itself achieves little if it undermines the very reasons people use end-to-end encrypted messaging in the first place.

[5] This quotation has been alternately attributed to Roger Needham, Butler Lampson, and Peter Neumann [7, 32], and paraphrased by Phillip Rogaway [74].

2 Implementation Paths

There are two broad strategies to enable separate messaging platforms to talk to each other [14, 58]: either all platforms adopt a common communications protocol (native interoperability), or each platform publishes an open API allowing others to communicate with them through a bridge. There are more flavors than these, including many hybrid possibilities with varying levels of implementation and maintenance feasibility. A platform could, for instance, support multiple protocols (e.g., in addition to its custom version of the Signal protocol, WhatsApp could also support MLS).

2.1 Standard Protocol

There are several existing candidates for selecting a universally adopted end-to-end key establishment protocol. The Matrix Foundation has developed the federated and interoperable Matrix protocol [55]. The Signal protocol (formly TextSecure), has been around for roughly a decade now and is the only open-source E2EE protocol that has been deployed at a scale of billions of users. More recently, the IETF standardized a new E2EE message exchange protocol, MLS (Messaging Layer Security) [12], which is intended to provide more efficient group communications than in Signal and related protocols. In February 2023, the IETF created a new "More Instant Messaging Interoperability" (MIMI) working group, the IETF's latest messaging interoperability standardization effort, dedicated to establishing the "minimal set of mechanisms" needed to allow contemporary messaging services to interoperate [38]. Among other aspects of messaging standardization, MIMI will seek to to extend MLS to deal with user discovery ("the introduction problem") as well as content formats for data exchange [38]. Work is ongoing and expected to continue well into 2024.

Any choice here is not obvious.[6] Several of the largest messaging services already use variations of the Signal protocol, and Meta explicitly advocated for widespread adoption of Signal at the European Commission's stakeholder workshop [27]. Another possibility is some sort of hybrid option: Matrix has proposed Matrix-over-MLS as part of the IETF's MIMI working group [88]. But any agreed-upon standard would eventually have to support many of the features and functionalities of all widely used E2EE applications.

Switching to a standard communications protocol poses immense challenges given the variety of protocols currently in use. Signal, WhatsApp, Viber, Facebook Messenger, and others rely on some variation of the Signal protocol, though they have developed different implementations (and, particularly in the case of group communications, different protocol versions). Telegram, Threema, and iMessage use custom protocols for the end-to-end encrypted layer, with Threema even using a custom client-to-server protocol [92, 84]. Element uses the Matrix protocol [55]. Existing messaging services would need to either switch to a common protocol or support multiple protocols. Service providers not only have to agree on a

[6] See `https://xkcd.com/927/`.

single protocol, but in many cases would have to redesign their entire system and manage a major migration to the new interoperable version.

There are also valid concerns around hampering innovation: Moxie Marlinspike, one of the co-creators of the Signal protocol, has argued that centralized, unfederated protocols evolve more rapidly than decentralized ones, where the latter have to deal with a wide diversity of clients, implementations, and deployments [54, 53]. For these reasons and others, a common protocol seems less practical in the near future than client-side APIs, not least because of the time constraints imposed by the DMA.

2.2 Client-side Bridges

In the face of substantial political, economic and technical obstacles to a universal communication standard, the developer community has begun to gravitate towards the idea of providing interoperability via public APIs, at least in the short term [57]. Each service provider could largely keep their existing E2EE protocol and implementation, but would provide a client-side interface to allow other messaging services to interact via a bridge between the two services.

Depending on the precise architectural design, such an interface would entail decrypting messages locally on the recipient's device after they are received from the sender's service provider, and then re-encrypting them with the recipient service provider's protocol. If the bridge (and therefore the key establishment protocol) runs on the client, this does not, at least theoretically, break the general notion of end-to-end encryption. Only the endpoints (the client devices) see the plaintext message; neither of the two (or more, in the case of group chats) platform servers are able to decrypt messages at any point. While running a server-side bridge is a technical possibility, both Meta and Matrix have acknowledged that server-side bridging has been "largely dismissed" since it would require message decryption and re-encryption in view of the service (violating the core principle of end-to-end encryption) [59]. A hybrid of the two, in which only the E2EE protocol is run in a client-side bridge with a server-side bridge doing much of the actual message transport, is also a possibility to handle systems challenges that may arise from asking the client to do too much work.

Some service providers already offer similar interfaces, either to integrate with other services or to allow third-party clients access. iMessage, Apple's end-to-end encrypted messaging service which interoperates with SMS, provides an example of client-side bridging at scale. It is well worth noting, however, that the technical feasibility of the message exchange, even at large scale, does not mean that this can be smoothly applied to interoperating E2EE services. SMS suffers from serious privacy issues, including that all messages are unencrypted, and interface design choices (namely, bubble color) have led to widespread aversion to SMS conversations. We will return to both points in greater depth later on.

Both of these approaches are far from a panacea. Eric Rescorla has written and spoken at length about the challenges of writing, standardizing, and implementing protocol specifications [24]. Client-side bridging comes with its own set of challenges: each service requesting access will need to build a different bridge for each provider with whom they want to interoperate. This has inherent security implications arising from the amount of excess code a service would need to add to achieve interoperability with a meaningful number of services. And while client-side bridging may not break E2EE in the technical sense that the plaintext messages are only viewable on the client's device, it is indisputably not a conventional meaning of E2EE. Moreover, providing *an* interface is not the same as providing a *usable* interface, which is a challenge even when a provider is giving their best effort. To aid in providing a reasonable developer experience, Rescorla has suggested a set of main interface requirements, including unambiguous interface specifications, stable interfaces, test servers, and real-time support from engineers at the other service [24]. Needless to say, these are far easier said than done.

Restrictions on API access: In practice, a service provider's interface cannot truly be "open" due to the sensitivity of the data being exchanged. Large service providers will individually approve requests by smaller service providers. The process for filtering requests may vary by provider, though in accordance with the DMA, they cannot deny a "reasonable" request. An interoperable interface will likely use some sort of revocable access key to manage access. Providers then need to figure out how to manage key requests and key storage, including some sort of service-level identity indicator [39].

But API request filtering is not merely a gatekeeping measure: platforms need to be able to detect and block bulk spam and forwarding services in real time. WhatsApp relies heavily on behavioral features to detect spam clients at the time of account registration in its existing public-facing interfaces, stating that the majority of use cases of these interfaces are spam [56]. Will Cathcart, the current head of WhatsApp, identified the need to restrict or outright block certain services as one of the most important considerations under an interoperability mandate [16].

The EFF has raised the spectre of a malicious actor creating a fake messaging service with a number of fabricated users and requesting access [13]. Large service providers will need to set certain objective and justifiable thresholds that give them the flexibility to respond only to legitimate requests. This is seemingly within the bounds of the DMA, though of course this will depend on what types of requesting services the European Commission considers to endanger the "integrity" of a service [26].

Many of the data sharing and privacy concerns surrounding third-party access are not fundamentally new: Facebook's infamous Cambridge Analytica scandal was made possible through Facebook's external app API. What has changed with the DMA is that service providers have far less flexibility to refuse or revoke access, let alone in real time as a situation is unfolding. Service providers will need a fair amount of latitude in their ability to deny access requests to continue to guard against malicious data scraping and mining, regardless of whether interoperable message is implemented through client-side bridging or an open standard.

3 Open Challenges

The security and privacy considerations associated with trying to reconfigure end-to-end encrypted systems to communicate with each other are too numerous for us to attempt complete coverage. We focus on five general areas in particular: user identity, key distribution, user discovery, spam and abuse, and interface design. We try to keep the discussion high-level so that it is largely agnostic to the message exchange architecture (i.e., whether messaging platforms have adopted a standard protocol or opted for client-side bridges).

3.1 User Identity

End-to-end encryption is meaningless without sufficient verification of the authenticity of the ends. There are two layers to identifying users:

1. *Cryptographic Identity:* First, how do you know whether a given public identity key belongs to a user Alice's account?
2. *Real-world Identity:* Second, once you have verified the public key attached to Alice's account on this service, how do you know that the user "Alice Appleton" who has contacted you is the Alice Appleton you know in real life?

Both cryptographic identity and real-world identity are needed to assure a user that they are indeed talking to the right person. We discuss each of them in turn.

3.1.1 Cryptographic Identity: A user's public key forms their "cryptographic identity". Currently, each messaging provider maintains their own separate public key directory for their userbase. When Alice wants to talk to Bob on a given messaging application (which both Alice and Bob use), Alice's client queries their provider for Bob's public key. The provider looks up Bob in their centralized key database and uses this information to establish an end-to-end encrypted communication channel between Alice and Bob.

Existing key distribution protocols generally require users to trust the provider to store and distribute the correct keys, and are vulnerable to a malicious provider or compromised key server. Interoperability further complicates trust establishment, since users will now have to place some degree of trust in a separate service provider. We revisit this question in greater depth in §3.2.

3.1.2 Real-world Identity: Tying a cryptographic identity to a real-world identity is an even more challenging problem since security and privacy are somewhat in conflict here. Messaging services vary in the information they ask of users: WhatsApp and Signal both require users to provide a phone number at the time of account registration, though Signal is currently working on username-based discovery so that a user's phone number is known only to Signal [62].

iMessage uses the email address associated with a user's Apple ID by default, but can also be configured to identify a user by their phone number. The Swiss messaging app Threema, on the other hand, identifies users through a randomly generated 8-digit Threema ID, and does not require a phone number or any other information tying a user account to an real-world identity scheme [86]. How is a WhatsApp user to know the Threema user is who they claim to be? At the moment, the only option would be to verify identities through an out-of-band channel (e.g., SMS, email, or meeting in person), which users rarely do in practice [95, 94, 79]. And even if an identity assurance ceremony is performed once in person, many vectors of account takeover have been industrialised by the cybercrime community (such as SIM swapping) while others are widely available to state actors (such as SS7 hacking).

While a real-world identity mechanism like a phone number or email address may give some identity assurance, there are many valid reasons why a user might not want to tie their identity on a messaging service to their real name, so the use of handles or pseudonyms is a desirable property for some messaging platforms to offer. Given the importance of identity assurance to interoperable communication, however, it may be that identity assurance and anonymity cannot be reconciled absent out-of-band user verification. Under the DMA, would a service provider be allowed to reject a request for interoperability from a messaging service that does not collect some form of external identity from its users? It is difficult to argue that a service's "level of security" is maintained if platforms are obligated to interoperate with a service that does not rely on some suitably accredited external identity scheme. And if so, would this give platforms an incentive to remove support for pseudonymous account registration? Such a decision may interact with other EU provisions around identity (such as eIDAS, the European Union's electronic identity verification service) as well as broader policy questions (such as identity escrow and age verification). The eventual outcome might be a creeping 'real names' policy, to the disadvantage of users with a genuine reason to seek anonymity – from political dissidents and stigmatised minorities to survivors of intimate abuse. Such users might have to seek out grey market service providers that have opted not to interoperate with regulated platforms, which would be marginalized and perhaps pass some stigma to their users.

Importantly, one of the main attractions of these platforms is that they are simple to sign up for. In most cases, user identity is bootstrapped off of a user's phone number, where a provider sends the user an SMS message with a random string which the user then enters in the app to prove control of said phone number. This ease of use must be preserved in any interoperable scheme, for instance by using an identifier the user already has memorized.

3.2 Key Distribution

Contemporary end-to-end encrypted messaging systems maintain platform-specific, centralized key directories to store their users' encryption keys. When a new user registers an account, the user's device generates several public-private key pairs (the precise number and type of key pairs depend on the protocol used) and sends the public keys to the service provider to store in its directory.

From a security standpoint, this reliance on the service provider to store and distribute encryption keys is a major weakness in existing systems. Since the service provider controls the public key directory, a malicious provider could compromise end-to-end security by swapping a user's public key for one under their control, either of their own volition or because they were legally compelled or otherwise pressured to do so. The confidentiality of the communications, then, hinges on trusting that the service provider has provided the correct keys. Existing protocols fail to prioritize this: MLS does not tackle the storage or distribution of keys, only how they might be used to send and receive cross-platform messages.

Interoperable communication further complicates trust issues around key storage and distribution. How would one service provider share the identity keys of its users with another in a way that satisfies user privacy expectations? And how can one platform be certain that the other has shared the correct keys? A platform could conceivably trick a user of another service into talking to a different person than the one with whom they believe they are communicating. Each service provider has no choice but to trust the others. Any open communication ecosystem will need to design explicit and effective controls around accessing, sharing, and replacing keys. Users' identity keys change constantly—every time they delete and reinstall the app or get a new phone, their device generates a new set of keys and the provider updates their directory accordingly. Service providers will need an efficient way of conveying user key changes to other providers. There is some ongoing academic work to develop a decentralized key infrastructure based on existing digital ID systems [83], but these are still in the proof-of-concept stage.

3.2.1 Key Transparency: Key transparency is an area of active research that, in theory, would eliminate the need to trust the service provider to share the correct keys. The general idea is that a service provider will still maintain a large key directory, but this directory is now publicly accessible and auditable by independent parties (either a third-party or possibly the service providers themselves auditing each other). A provider cryptographically commits to a user's identity to publicly key mapping at the time of key generation (such that it cannot be changed without detection), and further periodically commits to the full directory version. Users (more precisely, users' messaging clients) can then verify both their own keys as well as their contacts' keys to detect a provider serving different versions of its key directory to different users. In practice, service providers would likely maintain separate key directories, but other providers would be able to query this directory in a privacy-preserving manner such that an external provider could only query for individual users, along with other privacy mitigations. Note that an auditable keystore system does not *prevent* a key-swapping attack from taking place, it only *detects* when such an attack has happened after the fact.

CONIKS [61], the first end-user key verification service, was proposed in 2016 but suffered from scalability issues due to the frequency with which users would need to check that their key is correct. Several other key transparency systems have been proposed and, in some cases, deployed, in the years since. Subsequent academic research has built on CONIKS, formalizing the notion of a verifiable key directory and mitigating scalability concerns by making the frequency of user key checks depend on the number of times a user's key has changed, along with providing a handful of other practical deployment improvements [17, 50]. For various practical reasons, key transparency has seen minimal adoption among the largest industry providers, though Google released an open-source Key Transparency initiative [75] in 2017, representing a variation of CONIKS that enables users to audit their own keys.

But deploying a verifiable key directory service at scale across multiple service providers large and small is another matter. The authors of CONIKS neatly outlined several of the main barriers to deployment back in 2016 based on discussions with engineers at Google, Yahoo, Apple, and Signal [51]. The most serious challenges boil down to the difficulty of distinguishing between legitimate and adversarial key changes. When Alice gets a new phone or forgets her iMessage password and resets her keys, how can Alice's service provider convince other providers accessing and auditing the key directory that they have legitimately identified Alice through some real-world identity mechanism, and that these new keys do in fact still belong to Alice? Will Alice need to somehow prove her identity not just to her own service provider, but to all other providers? We are not aware of any existing key transparency proposals that solve these problems adequately. In the absence of genuine end-to-end identity verification, will WhatsApp simply have to take Telegram's word for it?

And completely apart from the technical challenges, we cannot assume that platforms will agree on how such a key management service should work. Messaging providers have already made different design decisions around detecting false positives and when to send a user a security warning that one of their contacts' keys has changed, usually in an attempt to avoid inundating users with security warnings. Until very recently, iMessage did not even provide users with an option to manually verify their keys. Their new Contact Key Verification system claims to alert users if an "exceptionally advanced adversary, such as a state-sponsored attacker" has managed to secretly join a chat [9], but few technical specifics of how this works on the backend are publicly available. Many other widely used E2EE services inundate the user with in-chat notifications every time one of their contacts' keys has changed. The complexity of a distributed key verification service, however, makes it even more likely to issue false security warnings due to various synchronization problems.

3.3 User Discovery

We can now build on the above discussions of user identity and key management to consider how the process of learning which service(s) a user uses and/or prefers might work.

There are two separate but related design principles, advocated by multiple NGOs, that should be considered in the development of any discovery mechanism [81, 40]:

1. **Separate Communications:** Users must be able to keep their communications on different messaging apps separate if they choose. The Center for Democracy & Technology offered the analogy of using a work email and a personal email, a paradigm adopted by the vast majority of the general public [81]. Prior work has shown that some users likewise use different messaging apps for different purposes [67, 10, 34, 99], a feature that should be preserved in an interoperable world. In other words, users should retain the ability to sign up for a messaging service and opt to receive messages from users on that same service only.
2. **Opt-Out by Default:** Related to the first principle, users should be opted-out of discovery by default to maintain reasonable user privacy expectations.[7] When a user signs up for a messaging service, they consent to discovery within that service—and only that service. We see two high-level designs where this could be accomplished: using service-specific identifiers (as with email), or allowing users to change their discoverability preferences in a per-service basis in the settings of the app(s) they use (i.e., Alice can choose to be discoverable on iMessage but not Telegram, etc.).

Keeping these principles in mind, we have two general models for forming user identifiers [42]:

1. **Service-Independent:** Users use the same real-world identifier (e.g., phone number) across multiple services. Alice wants to contact Bob for the first time, but since Bob's identifier is not linked to any one specific service, Alice still needs to figure out where to send her message. Presumably, her messaging app will present some sort of app selection interface for Alice to choose which service to contact Bob on.[8] If Bob is discoverable on multiple services, either Alice asks Bob which service he prefers out-of-band (e.g., in-person, over email, etc.), or Alice simply selects one of the options from the interface based on her own preferences. If users do not want to select a service manually each time they begin a conversation, then there are various options for automation. The simplest might be for each user to have a priority list (e.g. try Signal, then iMessage, then WhatsApp), just like the negotiation of TLS ciphersuites or EMV credit card chip authentication methods. But if

[7] The discussion around opt-in versus opt-out discovery in messaging interoperability is often compared to email since email addresses are openly discoverable and contactable by anyone. But there are well-recognized social conventions and distinctions among email services that do not exist in the messaging ecosystem. For instance, given two of Alice's email addresses, "alice.appleton@gmail.com" and "alice.appleton@company.com", one can reasonably infer which types of communications Alice would like sent to each address without needing to discuss with Alice.

[8] Many interface design questions arise here. For now, we focus primarily on privacy concerns in user discovery, and revisit the user experience in §3.5.

a user wanted family messages on WhatsApp and work messages on Signal, then this would become more complex still.

2. **Service-Specific:** Alternatively, Bob's identifier could be linked in some way to a specific service such that Alice's service provider knows where to deliver messages addressed to Bob. This is analogous to email, where each identifier is scoped to a particular namespace (i.e., alice.appleton@gmail.com and alice.appleton@cam.ac.uk do not necessarily refer to the same person). In the case of messaging, Rescorla has suggested that this might look something like "1.415.555.0123@whatsapp.com" [22].

   Of course, asking the user to enter this mapping would be messy from a usability standpoint, and in many cases the user would still need to go through an out-of-band process to obtain the scoped identifier from the other user. To keep the user experience seamless, this identifier-to-service mapping could be hidden from the user and only used internally.

3.3.1 Centralized Directory Service: The tricky part is figuring out how each service provider learns which services are associated with a given phone number. Rescorla and others have floated the idea of a large-scale, centralized database of phone number-to-messenger mappings, similar to how the PSTN (Public Switched Telephone Network) maintains a large database mapping phone numbers to carrier [22]. At a high level, a user record would be added to the directory service or updated at the moment of app installation, once said user has gone through some real-world verification process and proved their identity to their service's satisfaction (e.g., provided a random code sent to their phone number). In theory, this could be the same directory used as part of a centralized key distribution service.

But creating such a database for E2EE messaging services is more complicated than the PSTN database for a number of reasons (as Rescorla acknowledges). First and foremost, there is no privacy to speak of in SMS, whether we're talking about discovery or message contents. Any carrier can query the database—which is, of course, one of the reasons SMS is overrun with spam. Second, number-to-carrier is a one-to-one mapping, while number-to-messaging service is a one-to-many mapping (assuming that a significant number of users continue to use more than one messaging service). As discussed above, there are several different design options for selecting a service, including letting the message sender select through an interface, letting the message recipient indicate a preferred service, or perhaps even asking users to provide a ranking of services by context and then choosing the highest-ranked service common to both. Each of these would require a different set of cryptographic protocols to maintain certain privacy-preserving attributes.

In contrast to the PSTN database, in which phone numbers are rarely changed or removed, users may frequently adjust their discoverability preferences for different services, for instance as they make a new acquaintance who wants to use a particular service or receive too many unwanted spam messages from a different service. We will need effective identity revocation mechanisms: if Bob changes his mind and no longer wants to be discoverable by other platforms, how can Bob's service provider communicate this to the other platforms and gain reasonable assurance that they have actually removed Bob and all associated data from their servers?

In particular, the centralized nature of such a service raises several security and privacy concerns, some, but not all, of which can be solved using known cryptographic schemes. The ability of an individual user to query this directory and learn which apps another user is associated with is probably the least concerning, assuming that users will need to explicitly opt in to being discoverable in the first place, since this is the case for several of the largest platforms today. On Signal, for instance, you need only know someone's phone number to learn whether they are also on Signal. While a user can now rapidly retrieve a list of apps used by someone else, instead of having to manually install and test each one, this has minimal impact on the attack surface compared to other challenges. Presumably the directory would be rate-limited to some extent to prevent mass scraping, though the success of such a mitigation will come down to how rigorously clients are identified. More worrying is the fact that the service would know precisely who is talking to whom based on user identity lookups. This might be mitigated through private information retrieval (PIR) or private set intersection (PSI) techniques for anonymous contact discovery [20, 45, 21], but while academic work has made great strides in improving the scalability of PIR, it is unclear if such schemes are workable on the order of billions of users.

The most serious concern is that this kind of centralized service would know all messaging platforms used by every individual, a fact with enormous implications for user privacy. A related point is the need to make any such lookup service compatible with users' ability to opt in to interoperability on a service-by-service basis. In other words, Alice, who uses S_A, could opt in to discovery by users on S_B, but not S_C. Would S_C still be able to access Alice's records? How could access be controlled such that each provider is only able to query a subset of user records, and who would enforce this? Perhaps instead of a universal lookup service, each set of providers that interoperates shares a lookup server with records of jointly discoverable users, though this still poses many of the same questions around trust, shared hosting responsibility, and scalability.

3.3.2 SPIN: To avoid having to use a centralized service, the IETF MIMI working group has introduced a high-level framework for a new identity mapping protocol called SPIN (Simple Protocol for Inviting Numbers) [43] that operates similarly to existing account setups. SPIN assumes that all users are identifiable by their phone number, communicating users already know each other's numbers (as WhatsApp and others operate today), and that users' devices are online at the moment the conversation is started. When Alice wants to send a message to Bob, who uses a different service, Alice's client sends an SMS message to Bob's device requesting the services Bob supports, and Bob's device replies. While this method of identity mapping avoids the problem of using a large-scale directory service, it brings its own downsides. It would require modifications to and the cooperation of the underlying mobile OS (namely, iOS or Android) in addition to the messaging platforms, and would effectively add an extra step (and possibly a small delay) to the normal communication process [22].

3.4 Spam and Abuse

Content moderation is a real challenge in deploying an interoperable network of networks. Detecting and handling spam and abuse effectively is an unsolved problem with plaintext content, let alone encrypted content, let alone across multiple complex distributed systems [60, 78, 8].

3.4.1 Existing Techniques

The existing providers have spent years building up their content moderation systems, training complex machine-learning models and hiring human moderators to resolve hard cases and feed back ground truth. The scale is astonishing: WhatsApp bans nearly 100 million accounts annually for violating WhatsApp's terms of service [27]. It is unreasonable to expect that providers will all adopt some new universal content moderation system, unless perhaps mandated to do so by governments (more on this in §3.4.2).

For now, let us assume an end-to-end encrypted system where only the users can access message contents. In such a setup, providers rely on two primary techniques for content moderation: user reporting and metadata [46, 71].

User Reporting: The most effective content moderation scheme at scale is user reporting [71]. How might reports work for users communicating through two different service providers? Suppose Alice is using S_A, to exchange messages with Bob, who uses S_B. Bob sends Alice an abusive message, prompting Alice to block Bob. Either S_A needs a way of passing this on to S_B, or S_A continues to receive further messages from Bob but simply opts not to display them to Alice. For instance, email handles blocked users by automatically redirecting any future emails from them to a user's spam folder.

Suppose Alice also reports Bob for good measure. Presumably S_B would be responsible for handling the report as Bob is their user, but since Alice has reported him on S_A's interface, S_A will need to pass along relevant information to enable S_B to take appropriate action. When a message is reported, the clients of several E2EE services, including WhatsApp and Facebook Messenger, automatically send the plaintext of the previous five messages in a conversation to the provider along with the reported message to provide additional context [46]. Will this policy be maintained in an interoperable context, such that in order to report Bob, Alice ends up sharing certain personal communications with a different service provider?

All of this also needs to be compatible with existing message franking protocols [35] (and their metadata-private counterpart, asymmetric message franking [90]), which prevent users from faking abuse reports by cryptographically stamping each message. Message franking has been deployed in Meta messaging services for several years now [29].

Metadata: Since user reporting is inherently retroactive (i.e., the harmful content has already been sent), service providers also rely extensively on metadata to monitor unusual communication patterns in real time. This is most obviously relevant for spam, where a service provider can detect unusual volumes or destinations of messages, but profile or chat descriptions can also be very useful for fighting abuse. WhatsApp bans over 300,000 accounts each month for CSAM sharing based on this approach [97].

From a privacy standpoint, this dependence on metadata raises numerous questions around what data would be viewable by each service provider in an interoperable communication. While the DMA specifies that only personal data that is "strictly necessary" for "effective interoperability" can be shared between providers, this can be quite expansive given providers' reliance on metadata. On the other hand, if a provider is overly limited in what data they can collect, this could have adverse effects on efforts to fight spam and CSAM, negatively impacting the user experience.

And it cannot be assumed that only two platforms would be involved. If Alice and Bob are on Signal, and Bob adds Carol and Dave on WhatsApp, and Dave then adds Eolina and Firoz on Telegram, and so on, do we maintain message forwarding limits, source tracing, and other moderation techniques end-to-end? WhatsApp, for instance, imposes restrictions on how many times a message can be forwarded to limit viral spread (as this is abused to promote disinformation, such as election-related hoaxes) [96]. Without an inter-provider limit, spam and disinformation can be laundered in just the same way that drug gangs launder their proceeds by sending them along a chain of bitcoin exchanges. But cross-platform forwarding limits will only work if everyone keeps proper records in compatible ways. If you believe that's going to happen, we have a bridge we'd like to sell you.

3.4.2 Content Detection Schemes

While user reporting and metadata analysis are already deployed by most platforms, some may additionally deploy other content moderation schemes which arguably undermine the confidentiality and end-to-end encrypted properties of the messages, whether of their own volition or because they are under a government mandate to do so.

Content Scanning: Suppose a platform using some kind of automated content detection system (such as perceptual hashing [48], which can detect known harmful content in encrypted communications but has been shown to be vulnerable to attack [41, 72]) requests to interoperate with a service that has consciously opted not to use such a system. Will the latter be allowed to refuse this request on the grounds that it would compromise the level of security they provide to users?

And perhaps it is the other way around, where it is the large platform that has deployed some type of client-side scanning mechanism, either voluntarily or under a government mandate. Interoperability can be co-opted by governments to undermine end-to-end encryption in various ways. There is a clear risk, given the Child Sex Abuse ("Chat Control") Regulation before the European Parliament [28] and the Online Safety Bill before the British one, that the agencies will mandate client-side scanning on precisely those large platforms on which the interoperability mandate falls. In that case, the mandated client-side bridge becomes a scanning gateway that is in effect under government control. Recent history is littered with examples where law enforcement and the technical community held rather different notions of what violates an existing level of security [2, 3, 37, 1, 49, 6].

Traceability: Message traceability is the idea that platforms should be able to identify the source of a heavily forwarded message [91]. WhatsApp and numerous civil society organizations have spoken out against traceability, arguing that it breaks end-to-end encryption by identifying content users have shared without the consent of the message sender or recipient(s) [98]. Will WhatsApp be required to interoperate with a platform that has deployed some form of message traceability?

Will the European Commission consider systems using these moderation techniques to endanger the integrity of a service that does not (and would potentially refuse the interoperability request), given that they are themselves pursuing such mandates? More to the point, how would a gatekeeper even know whether the requesting service has deployed these or other schemes? Meredith Whittaker, President of the Signal Foundation, has alluded to this challenge, stating that while Signal is open to the general notion of interoperability, they would need to ensure there are no "tricks on the backend" to compromise security or privacy [44].

3.5 User Interface Design

Interface design is critical if messaging interoperability is to enhance, rather than degrade, the user experience. We have already seen that interoperable communication inherently presents a reduction in security and privacy compared to communication taking place on a single service. A panelist at a recent DMA stakeholder workshop, however, suggested that "we have failed if we have to inform the user of something changing" [27]. This is a well-intentioned but misguided notion that understates the data-sharing implications of interoperability and the depth of the privacy protections promised by platforms. Privacy-preserving cryptographic techniques can only take us so far. Critically, some data *cannot* be hidden in order to allow for sufficiently accurate spam moderation. We will need clear and unambiguous ways of informing the user that their data and messages are leaving their service, and, by extension, that the security and privacy guarantees and features of their platform may no longer apply.

3.5.1 App Selection Interface

The Matrix Foundation, Cory Doctorow, and others have drafted preliminary mockups of what the interface for selecting the message recipient's service might look like [58, 18], with Matrix likening it to choosing which application to use to open a file on an OS ("Open with..."). If a user opts to be discoverable on a large number of services (say, $n > 5$), we risk the interface becoming cluttered, but we consider this a comparatively minor problem since a limited number of services will be designated gatekeepers by the EU to begin with. Rescorla [23] suggested the example of email provider selection, where a user is shown a window containing the five or so most common email providers along with a freeform "other" option to enter an alternative provider. Such an interface was mandated by the EU for default browser selection on Windows PCs from 2009–2014, with a random choice order, after an antitrust finding against Microsoft's Internet Explorer.

There are still many open design questions around user discovery that determine what would be displayed to the user. If Bob is discoverable on only one service, should Alice's client default to sending the message to Bob on that service? Or should Alice be shown a pop-up window with just one option that she still needs to manually select in order to ensure she understands and consents to starting a communication channel with Bob's service provider? If Bob is discoverable on multiple services, and one of them happens to be the same service that Alice uses, should the chat default to Alice's service? Suppose Bob has two distinct ongoing communication channels with Alice through different apps Bob uses. Will the messages be intermingled within the same interface (as is the case with iMessage's SMS interoperability), or will they be shown to the user as two separate chats?

If Bob designates a particular platform as his preferred service, would Alice's interface indicate Bob's preferred service (for instance, via a visual nudge to choose it), or would the chat simply default to it? Would Bob be able to change this after the fact, or would they have to start a new conversation? What if Bob wants his default service to be whatever service Alice is using, and to choose a different one only if Bob is not on Alice's service? Most importantly, can individual platforms make different decisions in response to these questions, or do they need to establish a set of agreed standards for the interoperable user experience? Design questions like these will have an enormous impact on the user experience.

3.5.2 Communicating Changes in Security Guarantees

Effectively communicating security and privacy risks to the user is arguably a more difficult problem than designing a new cryptographic protocol for data exchange. We can draw on decades of security usability research showing time and again that users struggle to comprehend and act on data access requests [31, 47, 77] and security warnings [82, 79, 73, 70]. Among other takeaways, researchers have concluded that to be effective, the text of warnings need to communicate clear and specific risks [80]. The user would presumably be given a textual warning with an option to read further details at the moment they opt to be discoverable by an alternative service.

The open question is what distinction, if any, would be shown on a chat-by-chat basis. Matrix has compared this to the WhatsApp Business API [65, 27], which is not end-to-end encrypted when a business uses a third-party vendor (Meta included) to host their API. But while ordinary WhatsApp chats have a small bubble at the top of their chat window informing them that their communications are encrypted end-to-end, a chat passing through the business API simply makes no mention of encryption at all. In other words, WhatsApp "informs" users of the reduction in security through the absence of a typical security indicator, a design which is likely ineffective in building user comprehension. To avoid inadvertent downgrade attacks, it will have to be abundantly clear to a user whether a given conversation is taking place within their own service only, or across multiple third-party services [4]. And visual indicators are no silver bullet: Signal recently removed SMS support for Android, citing, among other reasons, the need to avoid inadvertent user confusion given that SMS and Signal messages were both sent in the same interface. As Signal put it, they "can only do so much on the design side to prevent such misunderstandings" [66].

We know from existing usability research that whether a user takes a warning seriously depends heavily on the design [93]. In one example back in 2013, Akhawe et al. [5] found that Mozilla Firefox's SSL warnings were significantly more effective than those in other browsers due to variations in design: around 70% of users clicked through Chrome's SSL warnings, while only one-third clicked through Firefox's warnings. Chrome has since invested heavily in redesigning their SSL warnings based on what Felt et al. [30] termed "opinionated design", meaning that the user should be nudged towards the option perceived by the designer to be superior through visual cues (instead of text explanations). In practice, this is typically accomplished by varying colors to encourage the user to choose the safer option or requiring the user to click through an extra window before advancing. Of course, in industry such ideas have been co-opted to create interfaces that nudge users to select an option that is in the company's interest—the more well-known term for these design strategies is now "dark patterns" [33].

3.5.3 Blue Bubbles and Green Bubbles

Fortunately, we already have a real-world case study demonstrating the impact of interface design choices on user decisions in an interoperable setting. While Apple's iMessage interoperates with SMS/MMS, Apple uses visual contrasts to make the difference abundantly clear to the user. When two iPhone users communicate through Apple's default Messages app, the sender's messages appear blue (indicating that the communication is taking place over Apple's iMessage). In contrast, while an iPhone user is able to use the same interface to talk to an Android user, the iPhone user's sent messages now appear green, indicating that the messages are being sent over SMS. This distinction has given rise to social pressure to use an iPhone over an Android phone, thereby further consolidating

Apple's market control, particularly over younger generations [69, 87]. In the words of one user, "If that bubble pops up green, I'm not replying" [69].

While the bubble color is not the only difference the user experiences (green bubbles lack certain iMessage features like emoji reactions, group naming, typing indicators, etc.), the rapid propagation of the blue versus green bubbles phenomenon throughout popular culture demonstrates just how effective these types of visual cues can be in branding one option as "good" and the other "bad"—indeed, from Android's perspective, perhaps a little too effective [76].

The ongoing bubble war dispels any notion that interoperability on its own will defeat network effects. Far from opening up Apple's "walled garden", iMessage/SMS interoperability appears to have further solidified Apple's market power. At the same time, however, Apple has an obligation to its users to inform them of the difference for fundamental security reasons. The same is true even when two E2EE services interoperate: as Meredith Whittaker recently pointed out, simply being end-to-end encrypted is not "an end in itself" [63]. Messaging services have widely varying privacy policies, cloud backup schemes, jurisdictions in which they operate, receptiveness to client-side scanning, levels of cooperation with foreign governments, and so on. Rather, the goal must be to preserve a broader understanding of privacy and situational awareness of possible attacks. The wicked question here, and one for which we have no answer, is how to convey accurately and clearly to the user what is happening when they opt to interoperate with another service, while not needlessly discouraging them from doing so.

4 Discussion

Having explored in depth what an interoperable messaging solution might look like, we can see that the core message communication protocol is just the beginning. Interoperable systems will also need protocols to support the many other features that make secure communication possible, including cross-platform user authentication and tracking data exchanges between platforms. A harbinger of the technical difficulty may be the fact that Meta has been trying to interoperate WhatsApp and Facebook Messenger since 2019 – and this is for two services owned and operated by the same company, which has a strong business incentive to make it work [52, 11]. Meta's President of Global Affairs, Nick Clegg, acknowledged in a 2021 interview that inter-platform interoperability is "taking us a lot longer than we initially thought" [11].

In any serious discussion of security and privacy trade-offs, the yardstick needs to be how much the attack surface has increased compared to existing systems. For instance, disappearing messages, a feature allowing a user to set messages to be automatically deleted after some specified period of time, have been held up as an example of a feature that is difficult to ensure in an interoperable service [16, 36]. Even if both services ostensibly offer the feature, each has no guarantee that the other has actually complied with the deletion request. When the subject came up at the European Commission's stakeholder workshop, the panelists' response was that a user never really knows what happens to their communications once they've left their device anyway—the message recipient may take a screenshot, have malware on their device, and so on [27]. This rejoinder dodges the reality that interoperability would add a substantial new attack vector in the form of the interoperating service. That disappearing messages, and indeed many of the challenges discussed throughout the paper, are already concerns in existing systems is no reason to make things worse.

Finally, interoperability will have to respect users' moral or personal-security choices. Some users so dislike Meta that they refuse to use WhatsApp, and demand that their friends use Signal instead. Other users, for example in Ukraine, consider Telegram to be compromised by Russian state agencies and refuse to touch it [19]. It will have to be obvious to everyone in a group when someone joining the group is acting as a gateway to another service, and the notification will have to be transitive, so that every group member can see whether their communications will end up in a system they do not trust. But here again, there are trade-offs between security and usability: if all users in a group have to approve the addition of a new member who comes from an odd network, the effect will be like forcing people to opt in to ad tracking – it won't happen at scale.

Interoperability without robust moderation and interface design to make platforms pleasant to use is a nonstarter. Giving users a choice between platforms without giving them a platform they would want to spend time on is no choice at all. The challenges in this space, though, are all the more reason for researchers to devote effort and attention to it, to ensure that users continue to have secure channels of communication in an interoperable world.

5 Acknowledgements

We thank Alastair Beresford, Ian Brown, Jon Crowcroft, Phillip Hallam-Baker, Daniel Hugenroth, Alec Muffett, and Sam Smith for valuable feedback and/or discussions. These contributors do not necessarily agree with the arguments presented here.

References

1. Abelson, H., Anderson, R., Bellovin, S.M., Benaloh, J., Blaze, M., Callas, J., Diffie, W., Landau, S., Neumann, P.G., Rivest, R.L., et al.: Bugs in our pockets: The risks of client-side scanning. arXiv preprint arXiv:2110.07450 (2021)
2. Abelson, H., Anderson, R., Bellovin, S.M., Benaloh, J., Blaze, M., Diffie, W., Gilmore, J., Neumann, P.G., Rivest, R.L., Schiller, J.I., et al.: The risks of key recovery, key escrow, and trusted third-party encryption (1997)
3. Abelson, H., Anderson, R., Bellovin, S.M., Benaloh, J., Blaze, M., Diffie, W.W., Gilmore, J., Green, M., Landau, S., Neumann, P.G., et al.: Keys under doormats. Communications of the ACM **58**(10), 24–26 (2015)

4. Abu-Salma, R., Redmiles, E.M., Ur, B., Wei, M.: 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) (2018)
5. Akhawe, D., Felt, A.P.: Alice in warningland: A large-scale field study of browser security warning effectiveness. In: USENIX security symposium. vol. 13, pp. 257–272 (2013)
6. Alec Muffett: A Civil Society Glossary and Primer for End-to-End Encryption Policy in 2022. https://alecmuffett.com/alecm/e2e-primer/e2e-primer-web.html (2022)
7. Anderson, R.: Security Engineering: A Guide to Building Dependable Distributed Systems. John Wiley & Sons (2020)
8. Anderson, R.: Chat control or child protection? arXiv preprint arXiv:2210.08958 (2022)
9. Apple: Apple advances user security with powerful new data protections. https://www.apple.com/newsroom/2022/12/apple-advances-user-security-with-powerful-new-data-protections/ (2022)
10. Arnold, R., Schneider, A., Lennartz, J.: Interoperability of interpersonal communications services–a consumer perspective. Telecommunications Policy **44**(3), 101927 (2020)
11. Azeem Azhar: How to Regulate Facebook (with Nick Clegg). https://hbr.org/podcast/2021/06/how-to-regulate-facebook-with-nick-clegg (2021)
12. Barnes, R., Beurdouche, B., Millican, J., Omara, E., Cohn-Gordon, K., Robert, R.: The Messaging Layer Security (MLS) protocol. Internet Engineering Task Force, Internet-Draft draft-ietf-mls-architecture/ (2020)
13. Bennett Cyphers and Cory Doctorow: Privacy Without Monopoly: Data Protection and Interoperability. https://www.eff.org/wp/interoperability-and-privacy (February 2021)
14. BEREC: BEREC report on interoperability of Number-Independent Interpersonal Communication Services (NI-ICS) (December 2022)
15. Borisov, N., Goldberg, I., Brewer, E.: Off-the-record communication, or, why not to use pgp. In: Proceedings of the 2004 ACM workshop on Privacy in the electronic society. pp. 77–84 (2004)
16. Casey Newton: Three ways the European Union might ruin WhatsApp. https://www.platformer.news/p/three-ways-the-european-union-might?s=r
17. Chase, M., Deshpande, A., Ghosh, E., Malvai, H.: Seemless: Secure end-to-end encrypted messaging with less trust. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security. pp. 1639–1656 (2019)
18. Cory Doctorow: Interoperable Facebook: How to Ditch Facebook Without Losing Friends. https://www.eff.org/files/2022/09/12/interoperablefacebook-2022_0.pdf (September 2022)
19. Darren Loucaides: The Kremlin Has Entered the Chat. https://www.wired.com/story/the-kremlin-has-entered-the-chat/ (2023)
20. Davidson, A., Pestana, G., Celi, S.: Frodopir: Simple, scalable, single-server private information retrieval. Cryptology ePrint Archive (2022)
21. Demmler, D., Rindal, P., Rosulek, M., Trieu, N.: Pir-psi: scaling private contact discovery. Cryptology ePrint Archive (2018)
22. Eric Rescorla: Discovery Mechanisms for Messaging and Calling Interoperability. https://educatedguesswork.org/posts/messaging-discovery/ (2022)
23. Eric Rescorla: End-to-End Encryption and Messaging Interoperability. https://educatedguesswork.org/posts/messaging-e2e/ (2022)

24. Eric Rescorla: Architectural Options for Messaging Interoperability. https://educatedguesswork.org/posts/dma-interop/ (2023)
25. Ermoshina, K., Musiani, F., Halpin, H.: End-to-end encrypted messaging protocols: An overview. In: Internet Science: Third International Conference, INSCI 2016, Florence, Italy, September 12-14, 2016, Proceedings 3. pp. 244–254. Springer (2016)
26. European Commission: Digital Markets Act (DMA). https://competition-policy.ec.europa.eu/dma_en
27. European Commission: DMA workshop - The DMA and interoperability between messaging services. https://competition-policy.ec.europa.eu/dma/dma-workshops/interoperability-workshop_en (February 2023)
28. European Parliament: Combating child sexual abuse online. https://www.europarl.europa.eu/RegData/etudes/BRIE/2022/738224/EPRS_BRI(2022)738224_EN.pdf (May 2022)
29. Facebook: Messenger Secret Conversations: Technical Whitepaper. https://about.fb.com/wp-content/uploads/2016/07/messenger-secret-conversations-technical-whitepaper.pdf (2016)
30. Felt, A.P., Ainslie, A., Reeder, R.W., Consolvo, S., Thyagaraja, S., Bettes, A., Harris, H., Grimes, J.: Improving ssl warnings: Comprehension and adherence. In: Proceedings of the 33rd annual ACM conference on human factors in computing systems. pp. 2893–2902 (2015)
31. Felt, A.P., Ha, E., Egelman, S., Haney, A., Chin, E., Wagner, D.: Android permissions: User attention, comprehension, and behavior. In: Proceedings of the eighth symposium on usable privacy and security. pp. 1–14 (2012)
32. Gina Kolata: The Key Vanishes: Scientist Outlines Unbreakable Code. https://www.nytimes.com/2001/02/20/science/the-key-vanishes-scientist-outlines-unbreakable-code.html (February 2001)
33. Gray, C.M., Kou, Y., Battles, B., Hoggatt, J., Toombs, A.L.: The dark (patterns) side of ux design. In: Proceedings of the 2018 CHI conference on human factors in computing systems. pp. 1–14 (2018)
34. Griggio, C.F., Nouwens, M., Klokmose, C.N.: Caught in the Network: The Impact of WhatsApp's 2021 Privacy Policy Update on Users' Messaging App Ecosystems. In: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems. pp. 1–23 (2022)
35. Grubbs, P., Lu, J., Ristenpart, T.: Message Franking via Committing Authenticated Encryption. In: Annual International Cryptology Conference. pp. 66–97. Springer (2017)
36. Ian Brown: Private Messaging Interoperability in the EU Digital Markets Act. https://openforumeurope.org/wp-content/uploads/2022/11/Ian_Brown_Private_Messaging_Interoperability_In_The_EU_DMA.pdf (December 2022)
37. Ian Levy and Crispin Robinson: Principles for a More Informed Exceptional Access Debate. https://www.lawfareblog.com/principles-more-informed-exceptional-access-debate (2018)
38. IETF: More Instant Messaging Interoperability (MIMI). https://datatracker.ietf.org/wg/mimi/about/
39. Internet Society: White Paper: Considerations for Mandating Open Interfaces. https://www.internetsociety.org/wp-content/uploads/2020/12/ConsiderationsMandatingOpenInterfaces-03122020-EN.pdf (December 2020)
40. Internet Society: DMA and interoperability of encrypted messaging. https://www.internetsociety.org/wp-content/uploads/2022/03/ISOC-EU-DMA-interoperability-encrypted-messaging-20220311.pdf (March 2022)

41. Jain, S., Crețu, A.M., de Montjoye, Y.A.: Adversarial detection avoidance attacks: Evaluating the robustness of perceptual hashing-based client-side scanning. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 2317–2334 (2022)
42. Jonathan Rosenberg: A Taxonomy for More Messaging Interop (MIMI). https://www.internetsociety.org/wp-content/uploads/2022/03/ISOC-EU-DMA-interoperability-encrypted-messaging-20220311.pdf (March 2022)
43. Jonathan Rosenberg and Cullen Jennings and Alissa Cooper and Jon Peterson: Simple Protocol for Inviting Numbers (SPIN). https://www.ietf.org/archive/id/draft-rosenberg-mimi-spin-00.html (October 2022)
44. Julia Angwin: Back into the Trenches of the Crypto Wars: A conversation with Meredith Whittaker. https://themarkup.org/hello-world/2023/01/07/back-into-the-trenches-of-the-crypto-wars
45. Kales, D., Rechberger, C., Schneider, T., Senker, M., Weinert, C.: Mobile private contact discovery at scale. In: USENIX Security Symposium. pp. 1447–1464 (2019)
46. Kamara, S., Knodel, M., Llansó, E., Nojeim, G., Qin, L., Thakur, D., Vogus, C.: Outside Looking In: Approaches to Content Moderation in End-to-End Encrypted Systems. arXiv preprint arXiv:2202.04617 (2022)
47. King, J., Lampinen, A., Smolen, A.: Privacy: Is there an app for that? In: Proceedings of the Seventh Symposium on Usable Privacy and Security. pp. 1–20 (2011)
48. Kulshrestha, A., Mayer, J.R.: Identifying harmful media in end-to-end encrypted communication: Efficient private membership computation. In: USENIX Security Symposium. pp. 893–910 (2021)
49. Levy, I., Robinson, C.: Thoughts on child safety on commodity platforms. arXiv preprint arXiv:2207.09506 (2022)
50. Malvai, H., Kokoris-Kogias, L., Sonnino, A., Ghosh, E., Oztürk, E., Lewi, K., Lawlor, S.: Parakeet: Practical Key Transparency for End-to-End Encrypted Messaging. Cryptology ePrint Archive (2023)
51. Marcela Melara: Why Making Johnny's Key Management Transparent is So Challenging. https://freedom-to-tinker.com/2016/03/31/why-making-johnnys-key-management-transparent-is-so-challenging/ (2016)
52. Mark Zuckerberg: A Privacy-Focused Vision for Social Networking. https://www.nytimes.com/2019/03/06/technology/facebook-privacy-blog.html (2019)
53. Marlinspike, M.: Reflections: The ecosystem is moving. Signal Blog (2016)
54. Marlinspike, M.: The ecosystem is moving. In: 36th Chaos Communication Congress, Leipzig, Germany. https://www.youtube.com/watch?v=Nj3YFprqAr8&ab_channel=n99 (2019)
55. Matrix: Matrix Specification. https://spec.matrix.org/latest/
56. Matt Jones: How WhatsApp Reduced Spam While Launching End-to-End Encryption. https://www.youtube.com/watch?v=LBTOKlrhKXk&ab_channel=USENIXEnigmaConference (2017)
57. Matthew Hodgson: Interoperability without sacrificing privacy: Matrix and the DMA. https://matrix.org/blog/2022/03/25/interoperability-without-sacrificing-privacy-matrix-and-the-dma
58. Matthew Hodgson: How do you implement interoperability in a DMA world? https://matrix.org/blog/2022/03/29/how-do-you-implement-interoperability-in-a-dma-world (2023)
59. Matthew Hodgson: The DMA Stakeholder Workshop: Interoperability between messaging services. https://matrix.org/blog/2023/03/15/the-dma-stakeholder-workshop-interoperability-between-messaging-services (2023)

60. Mayer, J.: Content moderation for end-to-end encrypted messaging. Princeton University (2019)
61. Melara, M.S., Blankstein, A., Bonneau, J., Felten, E.W., Freedman, M.J.: {CONIKS}: Bringing key transparency to end users. In: 24th {USENIX} Security Symposium ({USENIX} Security 15). pp. 383–398 (2015)
62. Meredith Whittaker: `https://twitter.com/mer__edith/status/1582808091397005312` (2022)
63. Meredith Whittaker: `https://twitter.com/mer__edith/status/1629131348731478017` (February 2023)
64. Moxie Marlinspike: There is no WhatsApp 'backdoor'. `https://signal.org/blog/there-is-no-whatsapp-backdoor/` (January 2017)
65. Nate Cardozo: Making it Easier to Manage Business Conversations on WhatsApp. `https://about.fb.com/news/2020/10/privacy-matters-whatsapp-business-conversations/` (October 2020)
66. nina-signal: Removing SMS support from Signal Android (soon). `https://signal.org/blog/sms-removal-android/` (2022)
67. Nouwens, M., Griggio, C.F., Mackay, W.E.: " WhatsApp is for family; Messenger is for friends" Communication Places in App Ecosystems. In: Proceedings of the 2017 CHI conference on human factors in computing systems. pp. 727–735 (2017)
68. Paterson, K.G., Scarlata, M., Truong, K.T.: Three lessons from threema: Analysis of a secure messenger. Tech. rep., Technical report
69. Patrick McGee: How Apple captured Gen Z in the US — and changed their social circles. `https://www.ft.com/content/8a2e8442-449e-4dbd-bd6d-2656b4503526` (February 2023)
70. Petelka, J., Zou, Y., Schaub, F.: Put Your Warning Where Your Link Is: Improving and Evaluating Email Phishing Warnings. In: Proceedings of the 2019 CHI conference on human factors in computing systems. pp. 1–15 (2019)
71. Pfefferkorn, R.: Content-oblivious trust and safety techniques: Results from a survey of online service providers. Journal of Online Trust and Safety **1**(2) (2022)
72. Prokos, J., Fendley, N., Green, M., Schuster, R., Tromer, E., Jois, T., Cao, Y.: Squint hard enough: Attacking perceptual hashing with adversarial machine learning
73. Reeder, R.W., Felt, A.P., Consolvo, S., Malkin, N., Thompson, C., Egelman, S.: An experience sampling study of user reactions to browser warnings in the field. In: Proceedings of the 2018 CHI conference on human factors in computing systems. pp. 1–13 (2018)
74. Rogaway, P.: The moral character of cryptographic work. Cryptology ePrint Archive (2015)
75. Ryan Hurst and Gary Belvin: Security Through Transparency. `https://security.googleblog.com/2017/01/security-through-transparency.html` (2017)
76. Samantha Cole: Google Is Begging Apple to Make Life Better for Green Bubbles. `https://www.vice.com/en/article/wxngb9/google-is-begging-apple-to-make-life-better-for-green-bubbles` (September 2022)
77. Santos, C., Rossi, A., Sanchez Chamorro, L., Bongard-Blanchy, K., Abu-Salma, R.: Cookie banners, what's the purpose? analyzing cookie banner text through a legal lens. In: Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society. pp. 187–194 (2021)
78. Scheffler, S., Mayer, J.: Sok: Content moderation for end-to-end encryption. arXiv preprint arXiv:2303.03979 (2023)
79. Schröder, S., Huber, M., Wind, D., Rottermanner, C.: 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. pp. 1–7 (2016)

80. Stransky, C., Wermke, D., Schrader, J., Huaman, N., Acar, Y., Fehlhaber, A.L., Wei, M., Ur, B., Fahl, S.: On the limited impact of visualizing encryption: Perceptions of e2e messaging security. In: Seventeenth Symposium on Usable Privacy and Security. pp. 437–454 (2021)
81. Sukhi Gulati-Gilbert and Michal Luria: Designing Interoperable, Encrypted Messaging with User Journeys. `https://cdt.org/insights/designing-interoperable-encrypted-messaging-with-user-journeys/` (August 2022)
82. Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., Cranor, L.F.: Crying Wolf: An Empirical Study of SSL Warning Effectiveness. In: USENIX security symposium. pp. 399–416. Montreal, Canada (2009)
83. The Alan Turing Institute: Trustchain. `https://github.com/alan-turing-institute/trustchain`
84. Threema: Cryptography Whitepaper. `https://threema.ch/press-files/2_documentation/cryptography_whitepaper.pdf`
85. Threema: New Communication Protocol "Ibex" and Extended Protocol Suite. `https://threema.ch/en/blog/posts/ibex`
86. Threema: Anonymity – the ultimate privacy protection. `https://threema.ch/en/blog/posts/anonymity` (2019)
87. Tim Higgins: Why Apple's iMessage is Winning: Teens Dread the Green Text Bubble. `https://www.wsj.com/articles/why-apples-imessage-is-winning-teens-dread-the-green-text-bubble-11641618009` (January 2022)
88. Travis Ralston and Matthew Hodgson: Matrix Message Transport. `https://datatracker.ietf.org/doc/draft-ralston-mimi-matrix-transport/` (2022)
89. Tufekci, Z.: In Response to Guardian's Irresponsible Reporting on WhatsApp: A Plea for Responsible and Contextualized Reporting on User Security (2017), `https://technosociology.org/?page_id=1687`
90. Tyagi, N., Grubbs, P., Len, J., Miers, I., Ristenpart, T.: Asymmetric Message Franking: Content Moderation for Metadata-Private End-to-End Encryption. In: Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III 39. pp. 222–250. Springer (2019)
91. Tyagi, N., Miers, I., Ristenpart, T.: Traceback for end-to-end encrypted messaging. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 413–430 (2019)
92. Unger, N., Dechand, S., Bonneau, J., Fahl, S., Perl, H., Goldberg, I., Smith, M.: Sok: secure messaging. In: 2015 IEEE Symposium on Security and Privacy. pp. 232–249. IEEE (2015)
93. Utz, C., Degeling, M., Fahl, S., Schaub, F., Holz, T.: (Un) Informed Consent: Studying GDPR Consent Notices in the Field. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 973–990 (2019)
94. Vaziripour, E., Wu, J., O'Neill, M., Metro, D., Cockrell, J., Moffett, T., Whitehead, J., Bonner, N., Seamons, K.E., Zappala, D.: Action Needed! Helping Users Find and Complete the Authentication Ceremony in Signal. In: SOUPS@ USENIX Security Symposium. pp. 47–62 (2018)
95. Vaziripour, E., Wu, J., O'Neill, M., Whitehead, J., Heidbrink, S., Seamons, K., Zappala, D.: Is that you, Alice? A Usability Study of the Authentication Ceremony of Secure Messaging Applications. In: Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017). pp. 29–47 (2017)
96. WhatsApp: About forwarding limits. `https://faq.whatsapp.com/1053543185312573`

97. WhatsApp: How WhatsApp Helps Fight Child Exploitation. https://faq.whatsapp.com/5704021823023684/?locale=en_US
98. WhatsApp: What is traceability and why does WhatsApp oppose it? https://faq.whatsapp.com/1206094619954598
99. Wiewiorra, L., Steffen, N., Thoste, P., Fourberg, N., Taş, S., Kroon, P., Busch, C., Krämer, J.: Interoperability regulations for digital services: Impact on competition, innovation and digital sovereignty especially for platform and communication services. https://www.bundesnetzagentur.de/DE/Fachthemen/Digitalisierung/Technologien/Onlinekomm/Study_InteroperabilityregulationsDigiServices.pdf?__blob=publicationFile&v=1 (August 2022)