Interoperable Symmetric Message Franking

Carolina Ortega Pérez
Cornell University
Ithaca, NY, USA

Thomas Ristenpart
Cornell Tech
New York, NY, USA

Julia Len
UNC Chapel Hill
Chapel Hill, NC, USA

Abstract

The recent Digital Markets Act (DMA), a regulation passed by the European Union in 2022, requires messaging applications with large user bases to support interoperable end-to-end encrypted (E2EE) communication. This raises numerous questions about how to adapt cryptographic protocols to this setting in a way that preserves security and privacy. This question is not only limited to the main messaging protocols, but also extends to protocols for abuse mitigation such as the symmetric message franking protocol first proposed by Facebook. The latter uses symmetric cryptography to enable reporting abusive E2EE messages in a way that allows the platform to cryptographically verify the report's veracity.

In this paper, we initiate a formal treatment of interoperable symmetric message franking (IMF). We focus on a server-to-server messaging flow, where messages are routed sequentially through the sender's and recipient's service providers, but allow the recipient to dynamically choose who to send a report to. We formalize the security definitions for IMF including adapting the sender and recipient binding definitions into various reportability and unforgeability definitions that take into account one of the service providers misbehaving. We also prove relations among these new definitions. Finally, we detail an IMF construction that satisfies the security definitions, and include a discussion of users' identity privacy goals and deployment considerations.

CCS Concepts

• **Security and privacy → Symmetric cryptography and hash functions**.

Keywords

message franking; interoperable messaging; end-to-end encrypted messaging

ACM Reference Format:
Carolina Ortega Pérez, Thomas Ristenpart, and Julia Len. 2025. Interoperable Symmetric Message Franking. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3719027.3744864

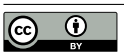
1 Introduction

End-to-end encrypted (E2EE) messaging systems are used by billions of people worldwide across major communication platforms like WhatsApp [57], Facebook Messenger [49], Apple iMessage [3], and Google Messages [29]. Recently, the European Union passed the Digital Markets Act (DMA) [26], which mandates large messaging applications must support interoperable E2EE communication. To be interoperable, E2EE platforms must allow one provider's users to communicate to another's, while retaining the E2E security guarantees of single-platform systems. This could be enabled by client-to-server deployments in which clients from different platforms communicate directly to messaging servers of another platform, or via a server-to-server deployment in which platforms operate independent servers that relay messages on behalf of their users. Major vendors and standardization bodies are now in the planning stages for near-term support of interoperability, including WhatsApp and Facebook Messenger [16, 32, 37, 41].

One aspect of interoperability that requires more attention is how abuse mitigations will work. Unfortunately, users face a flurry of spam, harassment, or other abuse on E2EE messaging platforms, and single-platform deployments have released various tools to curb it (c.f., [27, 56]). One important approach is *verifiable abuse reporting*, in which a user may report a message they received to the platform for moderation. To prevent fake reports from becoming an abuse vector themselves, the provider must be able to cryptographically verify that a report authentically came from the alleged sender. In the single-provider setting, Facebook Messenger initiated work here with deployment of their *message franking* feature [27]. Because the core message franking functionality uses just symmetric primitives, it is often referred to as symmetric message franking (SMF). Subsequent academic work formalized the cryptographic primitive [31], showed attacks against the handling of message attachments [23], and provided a variety of new protocol suggestions [23, 25, 36, 38, 55, 58].

SMF schemes would seem to work readily in the client-to-server architecture for interoperability, merely requiring implementation effort to ensure that clients support it. But the server-to-server setting raises a number of basic questions about how SMF could work, and what features we want from them. Given the importance of this setting—it offers potential privacy benefits [43], users may trust their own providers more for moderation, and it is going to be supported by at least WhatsApp [16]—we are motivated to provide a first treatment of interoperable symmetric message franking.

The server-to-server setting: challenges. The server-to-server setting works as follows. Each platform manages servers to which their clients connect. To send an encrypted message from a client on platform A to a client on platform B, the encrypted message is sent to a delivery service operated by A that we'll refer to as the sending provider (SP), which processes and then forwards it to a server operated by platform B, called the recipient provider (RP). Each platform in fact operates both an SP and RP service, to handle messages from and to their clients, respectively. This server-to-server architecture offers a number of beneficial features, as discussed by Len et al. [43].



CCS '25, Taipei, Taiwan

ACM ISBN 979-8-4007-1525-9/2025/10
https://doi.org/10.1145/3719027.3744864

In SMFs for single platforms, the delivery service and moderator are the same entity (implementations might run them on separate servers, but they are run by the same organization and are assumed to share any secret keys and state). One obvious approach for interoperable SMFs would be to just have each messaging platform run its own SMF instance and allow processing reports separately. However, this approach works only if all providers behave honestly, which ignores that platforms may misbehave or even be complicit in abuse. Indeed, WhatsApp has noted that one of its primary concerns with adopting interoperability is lack of trust in other providers to curb abuse [50]. For instance, a malicious SP (resp. RP) might be motivated to render encrypted messages unreportable to the RP (resp. SP) in order to protect their reputation. Conversely, a malicious RP might collude with recipients to forge reports to the SP. Thus, while most prior treatments of SMF treats the (single) delivery service as trusted, we must more broadly revisit the threat modeling for verifiable reporting in this setting.

Interoperable symmetric message franking. We initiate the formal treatment of interoperable symmetric message franking, or IMF, where we focus on the server-to-server architecture for sending messages. IMF empowers the recipient to choose the provider for reporting instead of designating a static moderator in advance as in prior work.

For IMF, defining security is complicated by the richer set of possible adversarial scenarios as compared with prior work on SMF. E2E confidentiality and authenticity is most similar to prior treatments, because here we can treat the SP and RP as a monolithic adversary. More complex is the landscape of definitions to capture ensuring that: (1) received messages are reportable and (2) reports are not forged to implicate someone as sending a message that they did not send. In the SMF literature, these goals were referred to as sender binding and receiver binding [31], but we will refer to them as reportability and unforgeability, respectively.

While with SMFs there were only two notions, we instead must contend with a variety of threat models, one for each meaningful subset of the four parties (sender, SP, RP, recipient) being adversarial, and for each of the two goals. We explore this space formally, detailing five reportability security definitions and six distinct unforgeability security definitions. For example, our notion (S)-RPT corresponds most closely to prior sender binding notions: a malicious sender tries to convince an honest recipient into not being able to report a received message to either the SP or RP, both modeled as honest. In contrast, (S,RP)-RPT captures instead a malicious sender colluding with a malicious RP to prevent a received message from being reported to an honest SP. We similarly explore the space of unforgeability notions, for example (S,R)-UNF corresponds to a malicious sender and recipient colluding to report a message to either the SP or RP (both honest) that was not sent. We establish relationships between the resulting 11 definitions.

Our main construction. We go on to detail an IMF scheme that achieves the strongest versions of our confidentiality, integrity, reportability, and unforgeability security goals. The starting point for the scheme is "parallel" use of a standard SMF protocol: the sender encrypts a message with a committing AEAD scheme, and the SP and RP separately compute MACs over the ciphertext (along with the sender and recipient identities) using independent provider-chosen keys. The full ciphertext and tags are sent to the recipient, who verifies (only) the validity of the ciphertext. This approach suffices if both SP and RP are always honest, but if, for example, a malicious RP modifies the SP's MAC tag, the recipient cannot detect it and reportability to the SP is forfeit.

One could use digital signatures here, but this would be relatively expensive at scale and we would like to stick to symmetric primitives if possible. Therefore, we rely on MACs to detect potential tampering instead. Computing and checking additional MACs at various stages of the sending flow does not add much overhead, assuming the relevant shared key material is available. However, this requires setting up such key material. Specifically, we want shared keys not only between the sender and recipient, but also between the sender and RP (preventing manipulations by the SP) and the SP and recipient (preventing manipulations by the RP). More generally, these keys would effectively allow building authenticated channels between the respective parties, even though they do not interact with each other directly in the architecture flow.

We detail how to setup the key material between all the four parties, including account registration, key distribution, and key agreement between parties. For this, we give a concrete example for how to compose message franking with the widely used Signal Protocol [51, 52], by adapting a simplified version of key exchange and secure messaging [2, 43] to our setting. The Signal protocol is used by WhatsApp and Messenger, the platforms mandated to support interoperability. We also discuss how one can realize such pairwise authentication channels, and by our results above, the significantly stronger level security of security, while allowing the privacy benefits of the server-to-server setting discussed in [43].

Summary. In summary, we initiate the formalization of interoperable symmetric message franking (IMF) for a server-to-server architecture. In more detail, we:

- introduce the notion of interoperable symmetric message franking (IMF) and formalize its correctness and security, including confidentiality, integrity, reportability, and unforgeability for various provider trust models;
- prove relations among the resulting 11 reportability and unforgeability definitions;
- detail a practical IMF construction and prove that it satisfies the security definitions described above.

2 Setting Overview

Verifiable abuse reporting enables a recipient to report abusive messages to service providers in a cryptographically verifiable manner. Verifiability is important in end-to-end encrypted (E2EE) messaging; message contents are hidden from the service providers, and so moderators need some way to check that a reported message was indeed sent by the indicated sender. Without verifiability, abuse reporting mechanisms are vulnerable to false accusations.

A prominent approach to verifiable abuse reporting for E2EE is *message franking*. Here the cryptographic abstraction, starting with [31], models the platform as both a ciphertext delivery service and moderator to which reports are submitted. As such, one can, roughly speaking, have the platform simply compute a MAC over

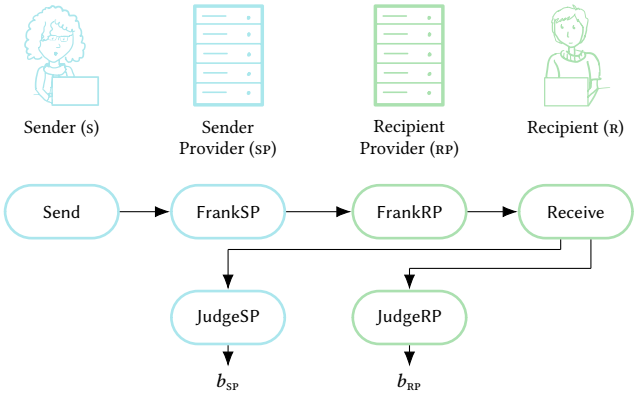


Figure 1: Architecture and flows for message franking in an interoperable setting. During the reporting stage the recipient can choose to report to the SP, the RP, or both.

each ciphertext and, combined with E2EE using appropriately key committing encryption, a report can simply submit back the key, ciphertext, and platform tag. This allows verification.

We consider a new setting for message franking: *interoperable E2EE messaging*. In this setting, there is a sender S, who is registered on a platform SP (sender provider) and wants to send a message to a recipient R, who is a user of the recipient provider RP. Len et al. [43] discuss two possible architectures for interoperable messaging: the *server-to-server* setting and the *client-to-server* setting. The first architecture requires users to always communicate through their service provider, and the second has users communicate directly with other service providers. Here, we consider a hybrid of both models: the messaging stage occurs on a server-to-server architecture, while the reporting operates on a client-to-server architecture to allow clients to communicate abuse reports with the provider they trust directly. We show the architecture in Figure 1.

Compared to work on traditional (single-server) message franking, this more complex architecture renders unclear how message franking should work. For instance, with two providers, we need to make decisions about which would receive reports of abusive messages. Moreover, while prior work captures security when users misbehave, it is now also possible that the service providers misbehave to exploit the system, e.g., by seeking to harm another provider's reputation and hurt their public trust.

We capture these various complexities in the new cryptographic primitive we introduce called Interoperable Symmetric Message Franking (IMF). We formalize IMF in Section 3 but first provide an overview of the setting we consider. Abuse reporting consists of three stages: the *setup*, the *sending flow*, and the *reporting flow*.

Setup. During an initial setup, users register with their respective service provider. At this stage, the sender S retrieves the keying material of the recipient from its provider RP by sending a request channeled through the sender provider SP. All key exchange protocols are performed during this stage. Throughout the protocol, we make reference to S and R, as strings that identify the sender and the recipient. We discuss these strings more in Section 7.

Sending flow. The sending flow enables S to send a message to the recipient. For the sending flow, we consider the server-to-server architecture, which we depict in Figure 1. This architecture requires users to always communicate through their service provider. Therefore, the sender S sends its encrypted message to the sender provider SP, which then forwards it to the recipient provider RP, which finally delivers the message to the recipient R.

Reporting flow. The reporting flow allows a recipient to report a message to one or both providers. The recipient can choose which provider(s) will act as the judge for verifying the report. This is a departure from prior work [23, 27, 31] which allows for only a single judge designated in advance of sending any messages. For this flow, we consider the more traditional client-to-server architecture, in which users contact other providers directly (i.e. not routing messages through their service providers). We chose this to ensure the recipient can choose to report dynamically, e.g., to whichever provider has stronger moderation policies. This gives R the chance to report to at least one honest provider, without being hindered by an RP who is malicious or who does not implement message franking. For instance, it is trivial for RP to drop or otherwise tamper with reports from the recipient to SP.

Alternative architecture. We note that the client-to-server setting is the default architecture used by both WhatsApp's and Messenger's interoperable messaging [16, 48], while the server-to-server model is optional. We do not consider the client-to-server setting for the complete flow because, from a verifiable abuse reporting perspective, it reduces to the traditional single-provider setting. While this may be seen as a feature of the client-to-server setting, it means we forego the other benefits of server-to-server, such as improved user privacy and ease of deployability [43]. Additionally, sending messages through both providers allows either one to serve as the moderator later on. We therefore view our work as a first step towards showing that IMF ameliorates the concern that server-to-server messaging cannot work well with verifiable abuse reporting.

Threat model. We start with a high level overview of our threat modeling, which will guide our formalization in later sections. Broadly, we target cryptographically verifiable abuse reporting. Prior work on message franking [21, 31, 36] targets two types of security goals:

- Sender binding: the inability of a malicious sender to trick a recipient into accepting a message that is not reportable.
- Receiver binding: the inability of a malicious recipient to report a message that was not sent by an honest sender.

First we point out that the second guarantee seems weaker than could be achievable: even if a malicious sender and recipient collude, the recipient should not be able to report different messages for any given transmitted ciphertext. We will target this stronger goal.

Given our server-to-server setting component, we consider what happens when service providers are malicious. This is important—with interoperability, providers may not be mutually trustworthy, and so just like we target end-to-end confidentiality and integrity guarantees, we also want cryptographic guarantees for IMF even in the face of a malicious SP or RP. That said, in this initial work we do assume some limitations on adversarial provider behavior.

For instance, a provider may simply drop a message or report, and we do not consider such denial of service. Finally, we do not consider replay attacks, e.g., a recipient repeatedly reporting the same received ciphertext. However, it is easy to extend our model to have providers include timestamps during franking, similar to Messenger [27], or some other sort of authenticated identifier that allows providers to have a strike list.

Finally, we assume secure communication channels (e.g., via TLS) between providers and their respective clients, as well as between providers. This prevents attacks by traditional network adversaries.

3 Syntax and Correctness

We introduce a new cryptographic primitive, *interoperable symmetric message franking* (IMF), which provides the algorithms needed for verifiable abuse reporting in the interoperable messaging setting. First, we present an overview of the setup and then the formal syntax and semantics of our sending and reporting algorithms. Finally, we describe the correctness goals for IMF. In Section 4, we present the formal security definitions for IMF.

Formally, an interoperable symmetric message franking scheme IMF = (Setup, Send, FrankSP, FrankRP, Receive, JudgeRP, JudgeSP) is a tuple of seven algorithms. Associated to an IMF scheme is a a message space $\mathcal{M} \subseteq \Sigma^*$, an associated data space $\mathcal{H} \subseteq \Sigma^*$, a ciphertext space $\mathcal{C} \subseteq \Sigma^*$, a tag space $\mathcal{T} \subseteq \Sigma^*$, and an identity space $\mathcal{I} \subseteq \Sigma^*$. Also associated to an IMF scheme are key spaces $\mathcal{EK}, \mathcal{TK} \subseteq \Sigma^*$, which we describe next in more detail.

IMF setup. For simplicity, we encapsulate contact discovery, key agreement and key distribution steps into our setup algorithm. The IMF key generation, is a randomized algorithm $(K_{\text{S,R}}, K_{\text{S,RP}}, K_{\text{SP,R}}, K_{\text{SP}}, K_{\text{RP}}) \leftarrow\$ \text{Kg}$, which outputs a tuple of five secret keys. The key $K_{\text{S,R}} \in \mathcal{EK}$ is a symmetric key shared between the sender and the recipient. The keys $K_{\text{S,RP}}$ and $K_{\text{SP,R}} \in \mathcal{TK}$ are symmetric keys shared between the respective parties indicated by the subscript text. The keys $K_{\text{SP}}, K_{\text{RP}} \in \mathcal{TK}$ are used by each provider, respectively, for generating message franking tags. For the security definitions we consider in Section 4, we assume all keys are independently and randomly chosen from their respective key spaces, such that all keys are independent.

Looking ahead, we will consider $K_{\text{S,R}}$ a one-time key that rotates after every message. This follows common deployed secure messaging models [51]. We assume the other keys are static. Since they do not rotate, there are no forward secrecy or post-compromise guarantees. An alternative design where keys do rotate would require additional logic and longer term storage to enable reporting messages signed with an older key. We discuss the key agreement in our messaging integration proposal (Section 6).

Sending and reporting algorithms. Below, we explain the syntax of each algorithm used in the sending and reporting flows:

- $(C, T_{\text{S,RP}}) \leftarrow\$ \text{Send}((K_{\text{S,R}}, K_{\text{S,RP}}), h, \text{S}, \text{R}, M)$: The randomized sending algorithm takes as input two keys associated to the sender $K_{\text{S,R}}, K_{\text{S,RP}}$, associated data $h \in \mathcal{H}$, the identifiers associated to the sender and the recipient $(\text{S}, \text{R}) \in \mathcal{I}^2$, and a message $M \in \mathcal{M}$. It outputs a ciphertext $C \in \mathcal{C}$ and a tag $T_{\text{S,RP}} \in \mathcal{T}$.

$\underline{\text{CORR}_{\text{IMF}}(\vec{K}, h, \text{S}, \text{R}, M)}$:
$(K_{\text{S,R}}, K_{\text{S,RP}}, K_{\text{SP,R}}, K_{\text{SP}}, K_{\text{RP}}) \leftarrow \vec{K}$
$(C, T_{\text{S,RP}}) \leftarrow\$ \text{IMF.Send}((K_{\text{SP}}, K_{\text{SP,R}}), h, \text{S}, \text{R}, M)$
$(T_{\text{SP}}, T_{\text{SP,R}}) \leftarrow\$ \text{IMF.FrankSP}((K_{\text{SP,R}}, K_{\text{SP}}), h, \text{S}, \text{R}, (C, T_{\text{S,RP}}))$
$T_{\text{RP}} \leftarrow\$ \text{IMF.FrankRP}((K_{\text{S,RP}}, K_{\text{RP}}), h, \text{S}, \text{R}, (C, T_{\text{S,RP}}, T_{\text{SP}}, T_{\text{SP,R}}))$
if $(T_{\text{SP}}, T_{\text{SP,R}}) = \bot \vee T_{\text{RP}} = \bot$ then return 0
$M' \leftarrow \text{IMF.Receive}((K_{\text{S,R}}, K_{\text{SP,R}}), h, \text{S}, \text{R}, (C, T_{\text{SP}}, T_{\text{SP,R}}, T_{\text{RP}}))$
if $M' \neq M$ then return 0
$(b_{\text{RP}}, M_{\text{RP}}) \leftarrow \text{IMF.JudgeRP}((K_{\text{S,RP}}, K_{\text{RP}}), K_{\text{S,R}}, h, \text{S}, \text{R}, (C, T_{\text{RP}}))$
$(b_{\text{SP}}, M_{\text{SP}}) \leftarrow \text{IMF.JudgeSP}((K_{\text{SP,R}}, K_{\text{SP}}), K_{\text{S,R}}, h, \text{S}, \text{R}, (C, T_{\text{SP}}))$
return $b_{\text{RP}} \wedge b_{\text{SP}} \wedge (M = M_{\text{RP}} = M_{\text{SP}})$

Figure 2: Correctness for an IMF scheme.

- $(T_{\text{SP}}, T_{\text{SP,R}}) \leftarrow\$ \text{FrankSP}((K_{\text{SP,R}}, K_{\text{SP}}), h, \text{S}, \text{R}, (C, T_{\text{S,RP}}))$: The randomized sender provider's franking algorithm takes as input the keys associated to the sender provider $K_{\text{SP,R}}, K_{\text{SP}}$, associated data h, two user identifiers S, R, a ciphertext C, and a tag $T_{\text{S,RP}}$. It outputs two tags $T_{\text{SP}}, T_{\text{SP,R}} \in \mathcal{T}^2$ or $\bot$.
- $T_{\text{RP}} \leftarrow\$ \text{FrankRP}((K_{\text{S,RP}}, K_{\text{RP}}), h, \text{S}, \text{R}, (C, T_{\text{S,RP}}, T_{\text{SP}}, T_{\text{SP,R}}))$: The randomized recipient provider's franking algorithm takes as input the recipient provider keys $K_{\text{S,RP}}, K_{\text{RP}}$, associated data h, two user identifiers S, R, a ciphertext C, and three tags $T_{\text{S,RP}}, T_{\text{SP}}, T_{\text{SP,R}}$. It outputs a franking tag $T_{\text{RP}} \in \mathcal{T}$ or $\bot$.
- $M \leftarrow \text{Receive}((K_{\text{S,R}}, K_{\text{SP,R}}), h, \text{S}, \text{R}, (C, T_{\text{SP}}, T_{\text{SP,R}}, T_{\text{RP}}))$: The deterministic receiving algorithm is run by the recipient to recover the message given the ciphertext. It takes as input the recipient's keys $K_{\text{S,R}}, K_{\text{SP,R}}$, associated data h, identifiers S, R, a ciphertext C, and three tags $T_{\text{SP}}, T_{\text{SP,R}}, T_{\text{RP}}$. It outputs a message $M \in \mathcal{M}$ if the message is recovered successfully, else $\bot$.
- $(b_{\text{RP}}, M_{\text{RP}}) \leftarrow \text{JudgeRP}((K_{\text{S,RP}}, K_{\text{RP}}), K_{\text{S,R}}, h, \text{S}, \text{R}, (C, T_{\text{RP}}))$: The deterministic recipient provider judging algorithm takes in keys $K_{\text{S,RP}}, K_{\text{RP}}, K_{\text{S,R}}$, associated data h, two user identifiers S, R, a ciphertext C, and a tag T_{RP}. It outputs a bit b_{RP} and a message M_{RP}.
- $(b_{\text{SP}}, M_{\text{SP}}) \leftarrow \text{JudgeSP}((K_{\text{SP,R}}, K_{\text{SP}}), K_{\text{S,R}}, h, \text{S}, \text{R}, (C, T_{\text{SP}}))$: The deterministic sender provider judging algorithm takes as input keys $K_{\text{S,R}}, K_{\text{SP,R}}, K_{\text{SP}}$, associated data h, two user identifiers S, R, a ciphertext C, and a tag T_{SP}. It outputs a bit b_{SP} and a message M_{SP}.

Modeling flexible reporting. A chief benefit of our formalization is that it enables more flexibility in reporting a message. The recipient can decide to report to SP via JudgeSP, to RP with JudgeRP, or both by calling both algorithms. This also allows the recipient to report a message without telling one of the providers. Thus, IMF supports a variety of reporting scenarios, including when a recipient selects a different judge for each message reported.

Properties of judging. We want that a judging algorithm outputs a bit 1 only when $M \neq \bot$, and otherwise returns bit 0. We capture this desired property below.

Definition 1. *An* IMF *scheme has valid judging algorithms JudgeSP and JudgeRP if when JudgeP for* $P \in \{\text{SP}, \text{RP}\}$ *returns* $b_P = 1$, $M_P \neq \bot$ *and when it returns* $b_P = 0$, $M_P = \bot$.

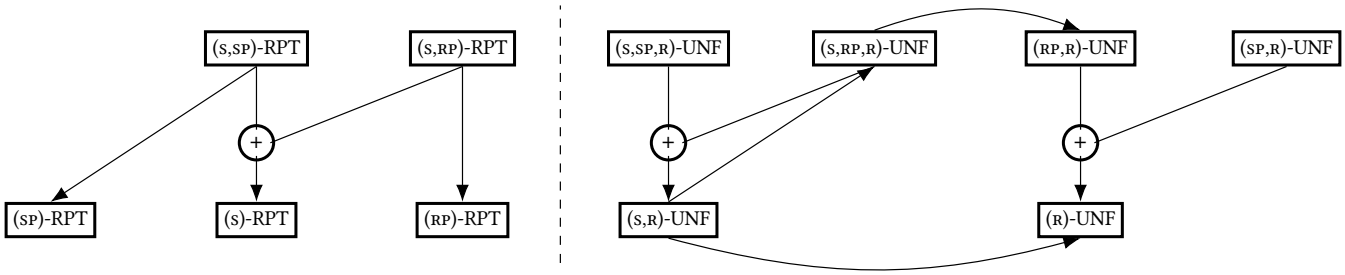


Figure 3: Diagram displaying the security games for reportability (RPT) and unforgeability (UNF) and the relations between them. The malicious parties on each game prefix each game on a parenthesis. The circle with a + sign denotes that the combination of definitions implies the resulting definition.

Correctness. We require that our schemes be correct. This means informally that an honest sending flow yields successful recovery of the message by the recipient, and that an honest reporting flow leads to the message being accepted by the relevant judge. Formally, an IMF scheme is correct if it holds that for all messages $M \in \mathcal{M}$, associated data $h \in \mathcal{H}$, identities $(\mathrm{S}, \mathrm{R}) \in \mathcal{I}^2$, and tuples of keys $\vec{K} = (K_{\mathrm{S,R}}, K_{\mathrm{S,RP}}, K_{\mathrm{SP,R}}, K_{\mathrm{SP}}, K_{\mathrm{RP}}) \in \mathcal{EK}^3 \times \mathcal{TK}^2$, running $\mathrm{CORR}_{\mathrm{IMF}}$ (Figure 2) it holds that

$$\Pr\left[\mathrm{CORR}_{\mathrm{IMF}}(\vec{K}, h, \mathrm{S}, \mathrm{R}, M) \Rightarrow 1\right] = 1\,,$$

where the probability is over the random coins used in Send, FrankSP, and FrankRP.

4 Security Goals

During the sending phase, an IMF scheme should provide confidentiality and ciphertext integrity. We formalize this with an all-in-one confidentiality and ciphertext integrity definition.

During the reporting phase, IMFs should (1) prevent an honestly generated report for a successfully received message from failing verification by the designated honest judge, and (2) prevent a maliciously generated report for a message that was not sent from being successfully verified by the designated honest judge. We refer to this first goal as *message reportability*. This follows the notion of sender binding in prior work [31]. We formalize the second goal as *report unforgeability*, similar to receiver binding from prior work.

Previous work on messaged franking only models the users as malicious or honest. However, in an interoperable setting, service providers can have conflicting interests, or even try to harm each other. For instance, one platform might try to inflate toxicity statistics on the other platform, by forging malicious reports. A platform could also try to make honest reports unreportable, to protect a malicious sender, or to give recipients a bad user experience and motivate them to use a different platform.

Therefore, for both reportability and unforgeability, we formalize a set of definitions where we model the four different parties—the sender (S), the sender provider (SP), the recipient provider (RP), and the recipient (R)—as malicious or honest. Figure 3 summarizes the games and relations. We label each game with a prefix indicating which are the malicious parties in that setting. We always assume at least one of the service providers is honest, else there are no guarantees about reporting.

$\underline{\mathrm{LEFT}^{\mathcal{A}}_{\mathrm{IMF}}}$:
$(\mathrm{S}, \mathrm{R}, K_{\mathrm{S,RP}}, K_{\mathrm{SP,R}}) \leftarrow\$\ \mathcal{A}$
$(K_{\mathrm{S,R}}, \cdot, \cdot, \cdot, \cdot) \leftarrow\$\ \mathrm{IMF.Kg}$
$\vec{K}_{\mathrm{S}} \leftarrow (K_{\mathrm{S,R}}, K_{\mathrm{S,RP}})\,;\ \vec{K}_{\mathrm{R}} \leftarrow (K_{\mathrm{S,R}}, K_{\mathrm{SP,R}})$
$\mathcal{Y} \leftarrow \{\}$
$b \leftarrow\$\ \mathcal{A}^{\mathsf{Send},\mathsf{Receive}}$
return b

$\underline{\mathsf{Send}(h, M_0, M_1)}$:
$(C, T_{\mathrm{S,RP}}) \leftarrow\$\ \mathrm{IMF.Send}(\vec{K}_{\mathrm{S}}, h, \mathrm{S}, \mathrm{R}, M_0)$
$\mathcal{Y} \leftarrow \mathcal{Y} \cup \{(h, C)\}$
return $C, T_{\mathrm{S,RP}}$

$\underline{\mathsf{Receive}(h, C, T_{\mathrm{SP}}, T_{\mathrm{SP,R}}, T_{\mathrm{RP}})}$:
if $(h, C) \in \mathcal{Y}$ then return $\perp$
$M \leftarrow \mathrm{IMF.Receive}(\vec{K}_{\mathrm{R}}, h, \mathrm{S}, \mathrm{R}, (C, T_{\mathrm{SP}}, T_{\mathrm{SP,R}}, T_{\mathrm{RP}}))$
return M

$\underline{\mathrm{RIGHT}^{\mathcal{A}}_{\mathrm{IMF}}}$:
$(\mathrm{S}, \mathrm{R}, K_{\mathrm{S,RP}}, K_{\mathrm{SP,R}}) \leftarrow\$\ \mathcal{A}$
$(K_{\mathrm{S,R}}, \cdot, \cdot, \cdot, \cdot) \leftarrow\$\ \mathrm{IMF.Kg}$
$\vec{K}_{\mathrm{S}} \leftarrow (K_{\mathrm{S,R}}, K_{\mathrm{S,RP}})$
$b \leftarrow\$\ \mathcal{A}^{\mathsf{Send},\mathsf{Receive}}$
return b

$\underline{\mathsf{Send}(h, M_0, M_1)}$:
$(C, T_{\mathrm{S,RP}}) \leftarrow\$\ \mathrm{IMF.Send}(\vec{K}_{\mathrm{S}}, h, \mathrm{S}, \mathrm{R}, M_1)$
return $C, T_{\mathrm{S,RP}}$

$\underline{\mathsf{Receive}(h, C, T_{\mathrm{SP}}, T_{\mathrm{SP,R}}, T_{\mathrm{RP}})}$:
return $\perp$

Figure 4: All-in-one left-or-right game for confidentiality and ciphertext integrity for scheme IMF **and adversary** $\mathcal{A}$**.**

Now we formally define these security goals. We assume IMF encryption key $K_{\mathrm{S,R}}$ is one-time, to follow the common secure messaging models which use the Double Ratchet [51]. We apply this assumption to unforgeability games where at least one user is honest. Reportability games are already a single oracle, while for the all-in-one confidentiality and ciphertext integrity we allow multiple queries to support a stronger adversary.

All-in-one confidentiality and ciphertext integrity. We want our IMF schemes to enjoy both confidentiality and ciphertext integrity. We adapt the standard all-in-one definition from the AE literature [11] to IMF. We call this notion LOR-CTXT, and it is formalized in the games $\mathrm{LEFT}^{\mathcal{A}}_{\mathrm{IMF}}$ and $\mathrm{RIGHT}^{\mathcal{A}}_{\mathrm{IMF}}$ shown in Figure 4. We formalized confidentiality as a left-or-right game instead of real-or-random because part of the output from Send, particularly $T_{\mathrm{S,RP}}$, is verified by RP and, thus, is not indistinguishable from random. We define the LOR-CTXT advantage of an adversary $\mathcal{A}$ against a scheme IMF by

$$\mathrm{Adv}^{\text{LOR-CTXT}}_{\mathrm{IMF}}(\mathcal{A}) = \left|\Pr\left[\mathrm{LEFT}^{\mathcal{A}}_{\mathrm{IMF}} \Rightarrow 1\right] - \Pr\left[\mathrm{RIGHT}^{\mathcal{A}}_{\mathrm{IMF}} \Rightarrow 1\right]\right|.$$

Multi-opening security. Multi-opening security [31] captures the goal of preserving confidentiality even after reporting occurs, when the key is revealed to the judge. We note that we intentionally do not target this notion because it is unnecessary for our setting,

$\underline{(\text{S,SP})\text{-RPT}^{\mathcal{A}}_{\text{IMF}}}$:
$(\cdot,\cdot,\cdot,\cdot,K_{\text{RP}}) \leftarrow\$ \text{Kg}$
$(K_{\text{S,R}}, K_{\text{SP,R}}, K_{\text{SP,R}}, h, \text{S}, \text{R}, C, T_{\text{S,RP}}, T_{\text{SP}}, T_{\text{SP,R}}) \leftarrow\$ \mathcal{A}$
$T_{\text{RP}} \leftarrow\$ \text{IMF.FrankRP}((K_{\text{S,RP}}, K_{\text{RP}}), h, \text{S}, \text{R}, (C, T_{\text{S,RP}}, T_{\text{SP}}, T_{\text{SP,R}}))$
if $T_{\text{RP}} = \perp$ then return 0
$M \leftarrow \text{IMF.Receive}((K_{\text{S,R}}, K_{\text{SP,R}}), h, \text{S}, \text{R}, (C, T_{\text{SP}}, T_{\text{SP,R}} T_{\text{RP}}))$
if $M = \perp$ then return 0
$(b_{\text{RP}}, M_{rp}) \leftarrow \text{IMF.JudgeRP}((K_{\text{S,RP}}, K_{\text{RP}}), K_{\text{S,R}}, h, \text{S}, \text{R}, (C, T_{\text{RP}}))$
return $(b_{\text{RP}} = 0)$

$\underline{(\text{S,RP})\text{-RPT}^{\mathcal{A}}_{\text{IMF}}}$:
$(\cdot,\cdot,K_{\text{SP,R}},K_{\text{SP}},\cdot) \leftarrow\$ \text{IMF.Kg}$
$(st_1, K_{\text{S,R}}, h, \text{S}, \text{R}, C, T_{\text{S,RP}}) \leftarrow\$ \mathcal{A}_1$
$(T_{\text{SP}}, T_{\text{SP,R}}) \leftarrow\$ \text{IMF.FrankSP}((K_{\text{SP,R}}, K_{\text{SP}}), h, \text{S}, \text{R}, (C, T_{\text{S,RP}}))$
$(h', \text{S}', \text{R}', C', T'_{\text{SP}}, T'_{\text{SP,R}}, T_{\text{RP}}) \leftarrow\$ \mathcal{A}_2(st_1, T_{\text{SP}}, T_{\text{SP,R}})$
$M \leftarrow \text{IMF.Receive}((K_{\text{S,R}}, K_{\text{SP,R}}), h', \text{S}', \text{R}', (C', T'_{\text{SP}}, T'_{\text{SP,R}}, T_{\text{RP}}))$
if $M = \perp$ then return 0
$(b_{\text{SP}}, M_{sp}) \leftarrow \text{IMF.JudgeSP}((K_{\text{SP,R}}, K_{\text{SP}}), K_{\text{S,R}}, h', \text{S}', \text{R}', (C', T'_{\text{SP}}))$
return $(b_{\text{SP}} = 0)$

Figure 5: Message reportability for a malicious sender and a malicious provider for a scheme IMF **and an adversary** $\mathcal{A}$**.**

in which we assume one-time keys for encrypting messages. This allows us to simplify notation and present the actual security required in practice for schemes that make use of such an assumption, e.g., those using the Double Ratchet [51]. The changes required to capture multi-opening security are mainly syntactic: instead of having $K_{\text{S,R}}$, there would be two separate keys—i.e., a secret encryption key and a franking key that may be revealed—just as in prior work [31]. We provide further details in the full version.

Message reportability. Here we consider two variants of message reportability (RPT) where different parties are malicious: both the sender and its provider ((S,SP)-RPT), and both the sender and the recipient provider ((S,RP)-RPT). The games are formally specified in Figure 5. We also consider variants where only the sender ((S)-RPT), only the sender provider ((SP)-RPT), and only the recipient provider ((RP)-RPT) are malicious. However, these three settings are implied by ((S,SP)-RPT) and ((S,RP)-RPT), therefore we include their games (and proof of these implications) in the full version of the paper. Since the reports are honestly generated, we do not consider a setting where the recipient is malicious.

The two games allow the adversary to choose the sender keys, the associated data h, the sender's and recipient's identifiers S, R, and the ciphertext. (S,SP)-RPT additionally allows the adversary to choose the sender provider's keys and SP's franking tag T_{SP}. (S,RP)-RPT allows the adversary to choose the recipient provider's keys. It also allows the adversary to see the honestly generated $T_{\text{SP}}, T_{\text{S,RP}}$, and then select a new set of associated data, identifiers, ciphertext, and franking tags. This is to capture that RP sees the tags as part of the sending flow and may tamper with the information before delivering it to R. The rest of the sending and reporting flows execute normally based on the adversarial inputs. We note that each game only executes one of the judging algorithms (the one from the honest service provider). The goal of the adversary is to generate the output values we describe above such that Receive succeeds but Judge rejects.

For an adversary $\mathcal{A}$ and message franking scheme IMF we define the (S,SP)-RPT advantage of $\mathcal{A}$ against IMF as

$$\text{Adv}^{(\text{S,SP})\text{-RPT}}_{\text{IMF}}(\mathcal{A}) = \Pr\left[(\text{S,SP})\text{-RPT}^{\mathcal{A}}_{\text{IMF}} \Rightarrow 1\right] .$$

For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and message franking scheme IMF we define the (S,RP)-RPT advantage of $\mathcal{A}$ against IMF as

$$\text{Adv}^{(\text{S,RP})\text{-RPT}}_{\text{IMF}}(\mathcal{A}) = \Pr\left[(\text{S,RP})\text{-RPT}^{\mathcal{A}}_{\text{IMF}} \Rightarrow 1\right] .$$

Report unforgeability. Here we consider three variants of report unforgeability (UNF) where different parties are malicious: both users and SP ((S,SP,R)-UNF), both users and RP ((S,RP,R)-UNF), and the recipient and SP ((SP,R)-UNF). The games are formally specified in Figure 6. Due to space constraints we formalize the other three games ((R)-UNF, (RP,R)-UNF, (S,R)-UNF) in the full version. However, those games are implied by the games described here.

The games allow the adversary to pick the shared keys (except in (SP,R)-UNF, where the game creates $K_{\text{S,RP}}$). In (SP,R)-UNF the adversary has access to a Send oracle that takes as input associated data, two identifiers, and a message, and output a ciphertext and a tag. Since S is honest, $K_{\text{S,R}}$ is one-time, and so is the Send oracle. The other games do not need this oracle because we assume that S is malicious, so the adversary has access to the keys necessary to run IMF.Send.

In all the games, the adversary has access to a franking oracle that takes as input associated data h, identifiers S, R, a ciphertext C, and some tags depending on the case. The oracle outputs the franking tag(s) from the honest service provider.

Then, in (S,SP,R)-UNF and (SP,R)-UNF, the adversary has access to a JudgeRP oracle that takes two tuples of a key, associated data, and two identifiers, and a ciphertext and with a franking tag. The honest recipient provider will judge the ciphertext and franking tags using both tuples. To win, the adversary must either (1) convince the provider to accept the report using one of the tuples, such that the identifiers and the ciphertext were not seen before, or (2) have the provider accept both reports such that the identifiers and the resulting message are different. To verify whether the identifiers and the ciphertext were seen before, the franking (send) oracle keep track of their queries inputs (and outputs). Similarly, in (S,RP,R)-UNF has access to an equivalent JudgeSP oracle.

For adversary $\mathcal{A}$ and message franking scheme IMF we define the (S,SP,R)-UNF, (S,RP,R)-UNF, and (SP,R)-UNF advantage of $\mathcal{A}$ against IMF, respectively, as

$$\text{Adv}^{(\text{S,SP,R})\text{-UNF}}_{\text{IMF}}(\mathcal{A}) = \Pr\left[(\text{S,SP,R})\text{-UNF}^{\mathcal{A}}_{\text{IMF}} \Rightarrow 1\right] ,$$
$$\text{Adv}^{(\text{S,RP,R})\text{-UNF}}_{\text{IMF}}(\mathcal{A}) = \Pr\left[(\text{S,RP,R})\text{-UNF}^{\mathcal{A}}_{\text{IMF}} \Rightarrow 1\right] , \text{ and}$$
$$\text{Adv}^{(\text{SP,R})\text{-UNF}}_{\text{IMF}}(\mathcal{A}) = \Pr\left[(\text{SP,R})\text{-UNF}^{\mathcal{A}}_{\text{IMF}} \Rightarrow 1\right] .$$

Prior works which targeted report unforgeability capture the case when the recipient, and not the sender, is malicious. To follow the prior literature, we formalize report unforgeability for when only the recipient is malicious and when the recipient and RP are malicious in the full version and show that the definitions with malicious users imply these. Report unforgeability with honest senders captures malicious recipients which may try to lie and frame the sender for sending abusive content. Alternatively, when

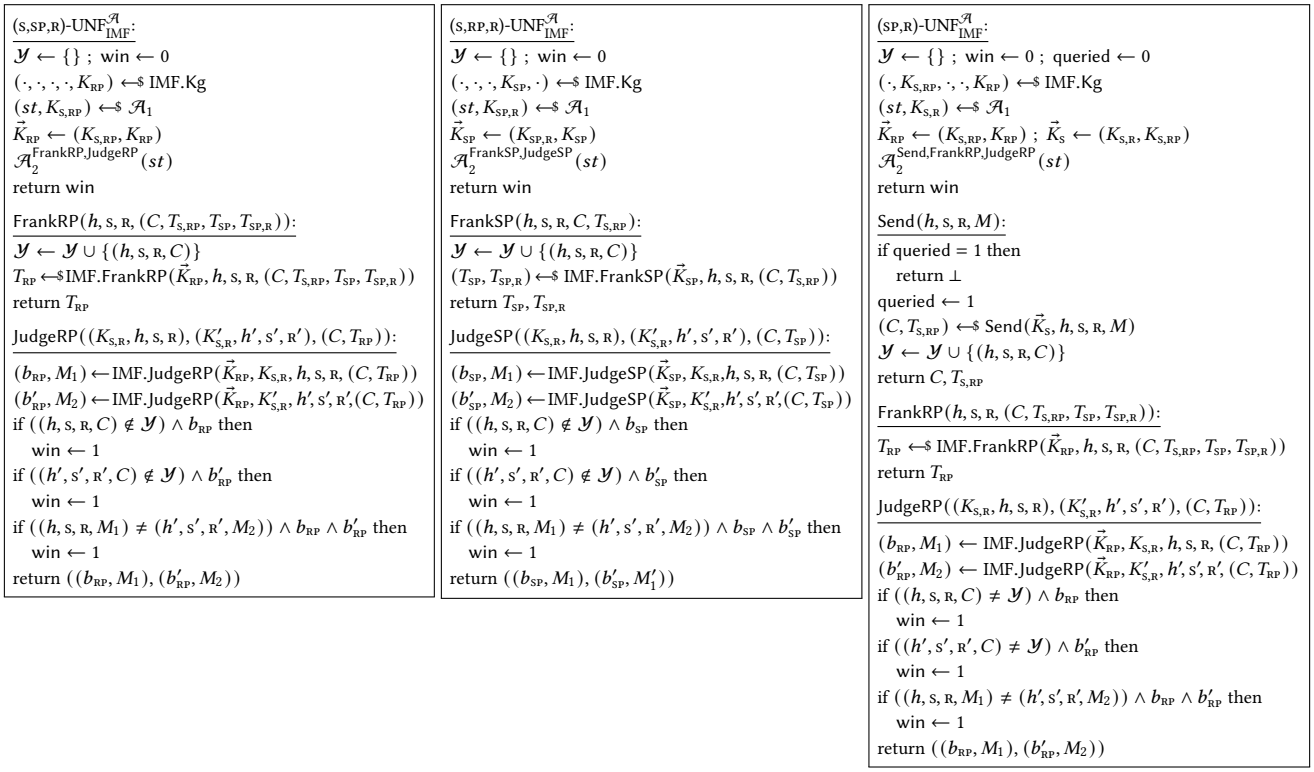


Figure 6: Report unforgeability for malicious sender and recipient and different configurations of malicious providers for a scheme IMF **and an adversary** $\mathcal{A}$**.**

both users are malicious, report unforgeability prevents such users from attempting to, for example, inflate the toxicity statistics of one or both providers.

Note that the setting when the recipient and SP are malicious ((SP,R)-UNF) is not implied by the setting where all the users and SP are malicious ((S,SP,R)-UNF). The intuition behind this is that the latter does not capture the case when a malicious sender provider and a recipient collude to pretend that S sent a message it did not; we assume that all ciphertexts that pass by the providers were sent. In the games in the full version, SP is honest and the malicious parties cannot pretend to be S because we assume S is authenticated to SP. Therefore, this attack does not arise in those settings either.

We do not consider a setting where only the sender provider or only the recipient provider is malicious because we assume authenticated channels between service providers and users during the report flow.

Relations between reportability and unforgeability. Due to space constraints, here we do not include the variations of the games in Figure 5 and Figure 6. Such variations are needed to prove the full separation between all reportability and unforgeability games shown in Figure 3, hence, we do not prove the full separation here. Instead, we sketch the separation between the games in Figure 5 and Figure 6. We defer the complete set of definitions, relations, and proofs to the full version.

Below, Theorem 1 shows that the reportability games from Figure 5 do not imply the unforgeability games from Figure 6. Then, Theorem 2 shows that the converse also holds.

Theorem 1. *We construct IMF scheme* Π_{rpt} *and show that for any (s,sp)-RPT adversary* $\mathcal{A}_{SP}$ *and for any (s,rp)-RPT adversary* $\mathcal{A}_{RP}$, $\text{Adv}^{(\text{S,SP})\text{-RPT}}_{\Pi_{rpt}}(\mathcal{A}_{SP}) = 0$ *and* $\text{Adv}^{(\text{S,RP})\text{-RPT}}_{\Pi_{rpt}}(\mathcal{A}_{RP}) = 0$. *Additionally, we give adversaries* $\mathcal{B}_{SP}$, $\mathcal{B}_{RP}$, $\mathcal{B}_{SPr}$ *such that* $\text{Adv}^{(\text{S,SP,R})\text{-UNF}}_{\Pi_{rpt}}(\mathcal{B}_{SP}) = 1$, $\text{Adv}^{(\text{S,RP,R})\text{-UNF}}_{\Pi_{rpt}}(\mathcal{B}_{RP}) = 1$, *and* $\text{Adv}^{(\text{SP,R})\text{-UNF}}_{\Pi_{rpt}}(\mathcal{B}_{SPr}) = 1$.

Sketch. This can be trivially achieved by a protocol where JudgeRP and JudgeSP always output 1, so we do not provide any further details.

Theorem 2. *Let* Π *be an IMF scheme. We construct IMF scheme* Π_{unf}*, such that for any (s,sp,r)-UNF adversary* $\mathcal{A}_{SP}$ *against* Π_{unf}*, we give adversary* $\mathcal{B}_{SP}$ *against* Π*, such that* $\text{Adv}^{(\text{S,SP,R})\text{-UNF}}_{\Pi_{unf}}(\mathcal{A}_{SP}) \leq \text{Adv}^{(\text{S,SP,R})\text{-UNF}}_{\Pi}(\mathcal{B}_{SP})$*. Furthermore, for any (s,rp,r)-UNF adversary* $\mathcal{A}_{RP}$ *against* Π_{unf}*, we give adversary* $\mathcal{B}_{RP}$ *against* Π*, such that* $\text{Adv}^{(\text{S,RP,R})\text{-UNF}}_{\Pi_{unf}}(\mathcal{A}_{RP}) \leq \text{Adv}^{(\text{S,RP,R})\text{-UNF}}_{\Pi}(\mathcal{B}_{RP})$*. Additionally, for any (sp,r)-UNF adversary* $\mathcal{A}_{SPr}$ *against* Π_{unf}*, we give adversary* $\mathcal{B}_{SPr}$ *against* Π*, such that* $\text{Adv}^{(\text{SP,R})\text{-UNF}}_{\Pi_{unf}}(\mathcal{A}_{SPr}) \leq \text{Adv}^{(\text{SP,R})\text{-UNF}}_{\Pi}(\mathcal{B}_{SPr})$*. Finally, we give adversaries* C_{SP} *and* C_{RP} *such that* $\text{Adv}^{(\text{S,SP})\text{-RPT}}_{\Pi_{unf}}(C_{SP}) = 1$ *and* $\text{Adv}^{(\text{S,RP})\text{-RPT}}_{\Pi_{unf}}(C_{RP}) = 1$.

Proof. We design Π_{unf} to run like Π, except that while Π.FrankSP outputs $(T_{\text{SP,R}}, T_{\text{SP}})$, Π_{unf}.FrankSP outputs $(T_{\text{SP,R}} \,\|\, b, T_{\text{SP}})$, where $b = 0$. Π_{unf}.FrankRP simply ignores b. In Π_{unf}.Receive, if $b = 0$, it discards the bit and proceeds as in Π. However, if the bit is 1, it outputs a string of 1s as M. Finally, b is not passed to the judging algorithms. However, we do modify these algorithms requiring them to fail (output $0, \perp$) if $C = \perp$. Notice that whenever $b = 0$, Π_{unf} outputs the same as Π. Since Send sets $b = 0$, Π_{unf} is correct.

Looking into the advantage of the unforgeability games, intuitively, b does not affect the behavior of the oracles. Let $\mathcal{A}_{\text{SP}}$ be an (S,SP,R)-UNF adversary against Π_{unf}. Then, we construct (S,SP,R)-UNF adversary $\mathcal{B}_{\text{SP}}$ against Π which works as follows. Whenever $\mathcal{A}_{\text{SP}}$ queries FrankRP, $\mathcal{B}_{\text{SP}}$ uses the same input to query its FrankRP, except that it removes the extra bit b from $T_{\text{SP,R}}$. Then, $\mathcal{B}_{\text{SP}}$ gets a tag T_{RP} and returns it to $\mathcal{A}_{\text{SP}}$. Whenever $\mathcal{A}_{\text{SP}}$ queries JudgeRP, $\mathcal{B}_{\text{SP}}$ sends the same query to its JudgeRP oracle and sends the output back to $\mathcal{A}_{\text{SP}}$. By construction, the contents of $\mathcal{Y}$ are identical in both games, as are the winning conditions, except that $\mathcal{A}_{\text{SP}}$ cannot win with a query where $C = \perp$. Therefore,

$$\mathsf{Adv}^{\text{(S,SP,R)-UNF}}_{\Pi_{unf}}(\mathcal{A}_{\text{SP}}) \leq \mathsf{Adv}^{\text{(S,SP,R)-UNF}}_{\Pi}(\mathcal{B}_{\text{SP}}),$$

as we wanted.

Now let $\mathcal{A}_{\text{SP}r}$ be an (SP,R)-UNF adversary against Π_{unf} and $\mathcal{B}_{\text{SP}r}$ the reduction to Π's (SP,R)-UNF security. We can use the same reduction as above and add that $\mathcal{B}_{\text{SP}r}$ also redirects queries to Send to its own oracle. This would also yield

$$\mathsf{Adv}^{\text{(SP,R)-UNF}}_{\Pi_{unf}}(\mathcal{A}_{\text{SP}r}) \leq \mathsf{Adv}^{\text{(SP,R)-UNF}}_{\Pi}(\mathcal{B}_{\text{SP}r}).$$

Meanwhile, let $\mathcal{A}_{\text{RP}}$ be an (S,RP,R)-UNF adversary against Π_{unf}. We construct (S,RP,R)-UNF adversary against Π, call it $\mathcal{B}_{\text{RP}}$, which forwards any query from $\mathcal{A}_{\text{RP}}$ to FrankSP to its own FrankSP oracle. After getting back $(T_{\text{SP}}, T_{\text{SP,R}})$, $\mathcal{B}_{\text{RP}}$ appends a 0 to $T_{\text{SP,R}}$ before returning the tags to $\mathcal{A}_{\text{RP}}$. The handling JudgeSP works the same as JudgeRP in the previous reductions. Again, $\mathcal{Y}$ in both reductions have the same contents and $\mathcal{B}_{\text{RP}}$ wins whenever $\mathcal{A}_{\text{RP}}$ does, so

$$\mathsf{Adv}^{\text{(S,RP,R)-UNF}}_{\Pi_{unf}}(\mathcal{A}_{\text{RP}}) \leq \mathsf{Adv}^{\text{(S,RP,R)-UNF}}_{\Pi}(\mathcal{B}_{\text{RP}}).$$

Finally, an (S,SP)-RPT adversary $\mathcal{C}_{\text{SP}}$ can create all of its inputs correctly, except that it sets $C = \perp$ and sets $b = 1$. Then, by construction, while Π_{unf}.Receive outputs a string of 1s instead of $\perp$, Π_{unf}.JudgeRP outputs $(0, \perp)$ and $\mathcal{C}_{\text{SP}}$ wins the game. An equivalent $\mathcal{C}_{\text{RP}}$ adversary can be described for (S,RP)-RPT. □

5 An IMF Protocol

Now we describe our scheme proposal Φ for IMF and prove it satisfies the security definitions from the previous section. Let AE = (Kg, Enc, Dec) be a key-committing AEAD scheme, and let MAC = (Kg, Tag, Ver) be a collision resistant (CR) MAC scheme.

At first glance, constructing an IMF protocol may seem simple. For instance, consider the following straw proposal. The sender encrypts the message to the recipient and has each provider MAC the ciphertext. Then, the recipient decrypts the message, and if they wish to report it, they send back the ciphertext, the encryption key, and the respective MAC tag to a provider for judging. However, a malicious provider could compromise reportability. For instance, a malicious RP could tamper with the MAC tag created by SP. So the recipient would be able to receive the message, but unable to report it to any provider. Concretely, this would violate (S,RP)-RPT.

To achieve our security goals in the stronger threat model that we consider, we must create authenticated channels between all the parties, to guarantee integrity of all communications. Pseudocode for the protocol Φ is shown in Figure 7. For the security proofs we assume the symmetric keys are generated randomly and independently of each other. Also, we rely on some traditional security notions associated to these building blocks. Due to space constrains, we include such definitions in the full version. We also assume that if any check fails, the protocol aborts.

In Section 6, we include the flow for sending a first message integrating a Signal protocol abstraction [2, 43] with ours. There, we also include a discussion of the properties required for our key exchange, since it is not limited to X3DH [52]. We keep the generation and usage of identifiers S, R black-boxed here and defer further discussion to Section 7.

Setup. For the setup, Kg runs the key generation algorithm for AE once, to create the encryption key $K_{\text{S,R}}$ and the one for MAC four times, to create $K_{\text{S,RP}}, K_{\text{SP,R}}, K_{\text{SP}}$, and K_{RP}. Having the shared keys between the sender and the recipient provider and between the sender provider and the recipient allow for the existence of authenticated channels between such parties. While having these channels makes messages from the same user linkable, the channels allow the recipient to reject messages if RP tampers with the message in such a way that it cannot later be reported to SP.

Messaging. To start, in Send, S uses $K_{\text{S,R}}$ to encrypt the message M with associated data h. The sender also uses key $K_{\text{S,RP}}$ to create a tag $T_{\text{S,RP}}$ for RP, to allow it to detect if SP tampers with any information it is sending. It then outputs $C, T_{\text{S,RP}}$. Then, the sender provider runs FrankSP, where it MACs the associated data h, both identifiers S, R, and the ciphertext using K_{SP} to create tag T_{SP}. It also uses $K_{\text{SP,R}}$ to MAC T_{SP} into another tag $T_{\text{SP,R}}$. It returns both tags. Next, through FrankRP, RP verifies $T_{\text{S,RP}}$, to ensure SP did not tamper with the information sent by S. Finally, it creates a new MAC tag T_{RP} of C and the identifiers using K_{RP}. It returns the new tag T_{RP}. Finally, the recipient, through Receive, verifies $T_{\text{SP,R}}$. This ensures that R can report the message to an honest SP, and decrypts C to retrieve M.

Reporting. If M is reported to either SP or RP, the respective judging algorithm verifies that the tag it created early correctly verifies and if so, decrypts C to retrieve M. If the verification and decryption succeed, it returns $(1, M)$, else $(0, \perp)$. From there, whichever provider(s) received a report, can verify whether M violates platform policies. Further penalties, such as blocking a user, are out-of-scope for this work.

Security evaluation. Due to the correctness of AE and MAC, it is easy to see by inspection that Φ satisfies the correctness condition. Now we prove that Φ satisfies the security definitions enumerated in Section 4. For the rest of the paper, we define $t(A, l)$ as the runtime of algorithm A on input of length l, and assume ℓ is the maximum length of $\mathcal{A}$ inputs.

Recall that our LOR-CTXT definition is based on traditional left-or-right confidentiality notions [7] as well as ciphertext integrity notions for authenticated encryption [11]. The theorem below shows that Φ satisfies LOR-CTXT via a reduction to LOR-CPA and INT-CTXT. See the full version for the proof.

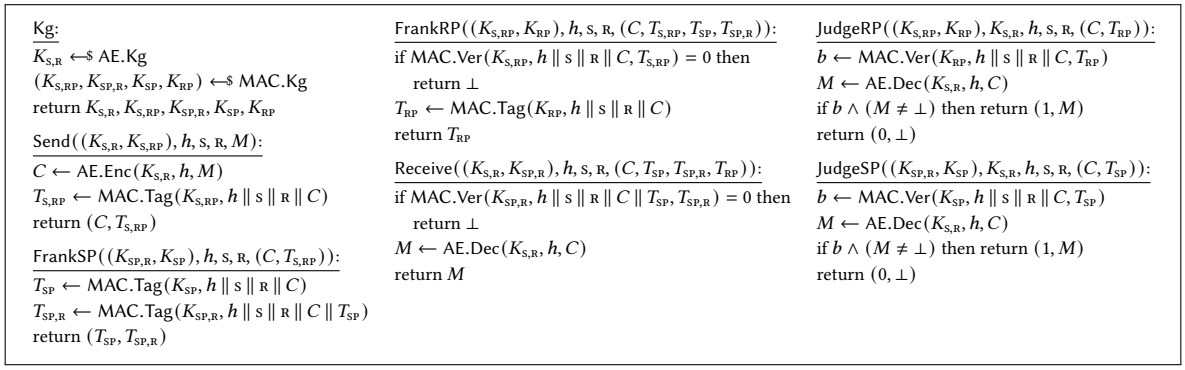


Figure 7: Interoperable symmetric message franking protocol Φ.

Theorem 3. *Let $\mathcal{A}$ be an LOR-CTXT adversary for scheme Φ using schemes* AE *and* MAC*, which performs q_s Send queries and q_r Receive queries. Then we give INT-CTXT adversary $\mathcal{B}$ and LOR-CPA adversary $\mathcal{C}$ against* AE

$$\mathsf{Adv}^{\text{LOR-CTXT}}_{\Phi}(\mathcal{A}) \leq \mathsf{Adv}^{\text{INT-CTXT}}_{\mathsf{AE}}(\mathcal{B}) + \mathsf{Adv}^{\text{LOR-CPA}}_{\mathsf{AE}}(\mathcal{C}) ,$$

where $\mathcal{B}, \mathcal{C}$ each performs at most $q_s + q_r$ queries. Then, $\mathcal{B}$ runs in time of $\mathcal{A}$ plus $q_s \cdot t(\mathsf{MAC.Tag}, \ell) + q_r \cdot t(\mathsf{MAC}.Ver, \ell) + O(1)$, while $\mathcal{C}$ runs in time of $\mathcal{A}$ plus $q_s \cdot t(\mathsf{MAC.Tag}, \ell) + O(1)$.

Now, regarding the reportability definitions, we only need to prove that Φ satisfies (S,SP)-RPT and (S,RP)-RPT, since these two imply the additional games included in the full version.

Theorem 4. *Let $\mathcal{A}$ be an (S,SP)-RPT adversary for scheme Φ. Then, $\mathsf{Adv}^{\text{(S,SP)-RPT}}_{\Phi}(\mathcal{A}) = 0$, meaning that Φ achieves perfect (S,SP)-RPT security.*

Proof. We will prove that (S,SP)-RPT always outputs 0. Let $(K_{S,R}, K_{S,RP}, K_{SP,R}, h, S, R, C, T_{S,RP}, T_{SP}, T_{SP,R})$ be the values returned by $\mathcal{A}$. During Φ.FrankRP, if $\mathsf{MAC.Ver}(K_{S,RP}, h \| S \| R \| C, T) = 0$, then (S,SP)-RPT returns 0. Thus, let us assume that with $\mathcal{A}$'s strategy this does not happen. Then Φ.FrankRP computes MAC.Tag over $h \| S \| R \| C$ using K_{RP}. Let this tag be T_{RP}. Next, during Φ.Receive, if $\mathsf{MAC.Ver}(K_{SP,R}, h \| S \| R \| C \| T_{SP}, T_{SP,R}) = 0$, then (S,SP)-RPT returns 0. Let us also assume that with $\mathcal{A}$ this does not happen. Then Φ.Receive computes and returns $\mathsf{AE.Dec}(K_{S,R}, h, C)$. Let us again assume that the output of this computation is $M \neq \perp$, else (S,SP)-RPT returns 0. Finally, Φ.JudgeRP verifies $\mathsf{MAC.Ver}(K_{RP}, h \| S \| R \| C, T_{RP}) = 1$, which by the correctness of MAC happens with probability 1. The computation of $M' \leftarrow \mathsf{AE.Dec}(K_{S,R}, h, C)$ is the same as that in Φ.Receive, such that when Φ.Receive outputs a message that is not $\perp$ it means that $M' \neq \perp$. Thus, we have that JudgeRP returns $(1, M')$ when Φ.Receive does not return $\perp$. □

Theorem 5. *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an (S,RP)-RPT adversary for scheme Φ using scheme* MAC*. Then we give adversary $\mathcal{B}$, running in time that of $\mathcal{A}$, such that*

$$\mathsf{Adv}^{\text{(S,RP)-RPT}}_{\Phi}(\mathcal{A}) \leq \mathsf{Adv}^{\text{UNF}}_{\mathsf{MAC}}(\mathcal{B}) .$$

Additionally, $\mathcal{B}$ runs in time of $\mathcal{A}$ plus $t(\mathsf{MAC.Tag}, \ell) + t(\mathsf{MAC}.Ver, \ell) + t(\mathsf{AE}.Dec, \ell) + O(1)$ and performs at most 2 oracle queries.

Proof. Let the values output by $\mathcal{A}_1$ and input to $\mathcal{A}_2$ be $(h, S, R, C, T_{SP}, T_{SP,R})$ and let $\mathcal{A}_2$ return $(h', S', R', C', T'_{SP}, T'_{SP,R}, T_{RP})$. We consider two games G_0, which equivalent to (S,RP)-RPT_{Φ}, and G_1, which is the same as G_0 except that if any of $h, S, R, C, T_{SP}, T_{SP,R}$ is different to $h', S', R', C', T'_{SP}, T'_{SP,R}$, the game returns 0. Now, by definition,

$$\begin{aligned}\mathsf{Adv}^{\text{(S,RP)-RPT}}_{\Phi}(\mathcal{A}) &= \Pr[G_0 \Rightarrow 1] \\ &= \Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1] + \Pr[G_1 \Rightarrow 1]\end{aligned}$$

Since, G_0 and G_1 are identical until the final check, using the fundamental lemma of game playing, we can bound

$$|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq \Pr[G_1 \text{ sets bad}] .$$

Additionally, we can bound $\Pr[G_1 \text{ sets bad}]$ by reducing to the UF-CMA security of MAC. Call this reduction $\mathcal{B}$, which works as follows. When $\mathcal{A}_1$ provides its inputs to the game $K_{S,R}, h, S, R, C, T_{S,RP}$, $\mathcal{B}$ samples a key K_{SP}, and uses it to run MAC.Tag to get T_{SP}. Then, it queries its Tag oracle on input $(h \| S \| R \| C \| T_{SP})$, which returns $T_{SP,R}$. After $\mathcal{A}_2$ provides its inputs, $\mathcal{B}$ queries its Ver oracle on inputs $((h' \| S' \| R' \| C' \| T'_{SP}), T'_{SP,R})$. Notice that $\mathcal{A}$ needs $T'_{SP,R}$ to verify (among others) in order to win the game. After this, $\mathcal{B}$ simulates the rest of the game, since it knows all the required keys. Now, in the event that $\mathcal{A}$ makes a query which sets bad to 1, $\mathcal{B}$'s query to Ver sets the UF-CMA flag win to 1, since it means it is querying values it had not queried before and the verification succeeded.

On the other hand, note that if $(h, S, R, C, T_{SP}, T_{SP,R}) = (h', S', R', C', T'_{SP}, T'_{SP,R})$, then due to the correctness of MAC, MAC.Ver inside Φ.Receive and Φ.JudgeSP succeed. Additionally, the decryption in Φ.Receive and Φ.JudgeSP are identical. Notice that $\mathcal{A}$ wins only when AE.Dec outputs $M \neq \perp$. However, if AE.Dec succeeds in Φ.Receive, it means that both MAC.Ver and AE.Dec succeed in Φ.JudgeSP. Therefore, $\Pr[G_1 \Rightarrow 1] = 0$ and so

$$\mathsf{Adv}^{\text{(S,RP)-RPT}}_{\Phi}(\mathcal{A}) \leq \mathsf{Adv}^{\text{UNF}}_{\mathsf{MAC}}(\mathcal{B})$$

as we wanted.

Overall, in addition to running $\mathcal{A}$, $\mathcal{B}$ also runs a key generation, which we will assume takes $O(1)$, and MAC.Tag, a MAC.Ver, and two AE.Dec (although they are the same, so $\mathcal{B}$ can run it only once). Therefore, $\mathcal{B}$ runs in time of $\mathcal{A}$ plus $t(\mathsf{MAC.Tag}, \ell) + t(\mathsf{MAC}.Ver, \ell) + t(\mathsf{AE}.Dec, \ell) + O(1)$. It queries each of its oracles once, hence $\mathcal{B}$ performs only 2 oracle queries. □

At last, looking into the unforgeability definitions, we prove that Φ satisfies (S,SP,R)-UNF, (S,RP,R)-UNF, and (SP,R)-UNF. Due to space constrains, we show here a sketch of the proofs and include the full proofs in the full version.

Theorem 6. *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an (S,SP,R)-UNF adversary for scheme Φ using schemes* MAC *and* AE*, which performs q_f FrankRP queries and q_j JudgeRP queries. Then we give UF-CMA adversary $\mathcal{B}$, CMT adversary $\mathcal{C}$, and CR adversary $\mathcal{D}$ such that*

$$\mathsf{Adv}^{\text{(S,SP,R)-UNF}}_{\Phi}(\mathcal{A}) \leq \mathsf{Adv}^{\text{UF-CMA}}_{\mathsf{MAC}}(\mathcal{B}) + \mathsf{Adv}^{\text{cmt}}_{\mathsf{AE}}(C) + \mathsf{Adv}^{\text{cr}}_{\mathsf{MAC}}(\mathcal{D}) ,$$

where $\mathcal{B}$ performs $q_f + 2q_j$ oracle queries while $\mathcal{C}$ and $\mathcal{D}$ perform one. $\mathcal{B}$ runs in time of $\mathcal{A}$ plus $q_f \cdot t(\mathsf{MAC}.\mathit{Ver}, \ell) + 2q_j \cdot t(\mathsf{AE}.\mathit{Dec}, \ell) + O(1)$, while $\mathcal{C}$ and $\mathcal{D}$ run in time of $\mathcal{A}$ plus $q_f(t(\mathsf{MAC.Tag}, \ell) + t(\mathsf{MAC}.\mathit{Ver}, \ell)) + 2q_j(t(\mathsf{MAC}.\mathit{Ver}, \ell) + t(\mathsf{AE}.\mathit{Dec}, \ell)) + O(1)$.

Sketch. We consider the query that sets win $\leftarrow 1$. If win $\leftarrow 1$ because one of the first two winning conditions in JudgeRP is true, we bound the advantage of $\mathcal{A}$ by reducing to the UF-CMA security of MAC. In this case, the UF-CMA oracle controls K_{RP}.

Then, we consider the case when $\mathcal{A}$ satisfies the third winning condition in JudgeRP with values $((h, \text{S}, \text{R}, M_1) \neq (h', \text{S}', \text{R}', M_2))$ and divide it into two sub-cases: (a) $(h, \text{S}, \text{R}) = (h', \text{S}', \text{R}')$ and (b) $(h, \text{S}, \text{R}) \neq (h', \text{S}', \text{R}')$. We bound the advantage of $\mathcal{A}$ in sub-case (a) by reducing to the CMT security of AE and bound the advantage of $\mathcal{A}$ in sub-case (b) by reducing to the CR security of MAC.

Theorem 7. *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an (S,RP,R)-UNF adversary for scheme Φ using schemes* MAC *and* AE*, which performs q_f FrankSP queries and q_j JudgeSP queries. Then we give UF-CMA adversary $\mathcal{B}$, CMT adversary $\mathcal{C}$, and CR adversary $\mathcal{D}$ such that*

$$\mathsf{Adv}^{\text{(S,RP,R)-UNF}}_{\Phi}(\mathcal{A}) \leq \mathsf{Adv}^{\text{UF-CMA}}_{\mathsf{MAC}}(\mathcal{B}) + \mathsf{Adv}^{\text{cmt}}_{\mathsf{AE}}(C) + \mathsf{Adv}^{\text{cr}}_{\mathsf{MAC}}(\mathcal{D}) ,$$

where $\mathcal{B}$ performs $q_f + 2q_j$ oracle queries while $\mathcal{C}$ and $\mathcal{D}$ perform one. $\mathcal{B}$ runs in time of $\mathcal{A}$ plus $q_f \cdot t(\mathsf{MAC.Tag}, \ell) + 2q_j \cdot t(\mathsf{AE}.\mathit{Dec}, \ell) + O(1)$, while $\mathcal{C}$ and $\mathcal{D}$ run in time of $\mathcal{A}$ plus $2q_f \cdot t(\mathsf{MAC.Tag}, \ell) + 2q_j \cdot (t(\mathsf{MAC}.\mathit{Ver}, \ell) + t(\mathsf{AE}.\mathit{Dec}, \ell)) + O(1)$.

The proof follows the same strategy as that for Theorem 6.

Theorem 8. *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an (SP,R)-UNF adversary for scheme Φ using schemes* MAC *and* AE*, which performs q_s Send queries, q_f FrankRP queries, and q_j JudgeRP queries. Then we give UF-CMA adversary $\mathcal{B}$, CMT adversary $\mathcal{C}$, and CR adversary $\mathcal{D}$ such that*

$$\mathsf{Adv}^{\text{(SP,R)-UNF}}_{\Phi}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\text{UF-CMA}}_{\mathsf{MAC}}(\mathcal{B}) + \mathsf{Adv}^{\text{cmt}}_{\mathsf{AE}}(C) + \mathsf{Adv}^{\text{cr}}_{\mathsf{MAC}}(\mathcal{D}) ,$$

where $\mathcal{B}$ performs at most $q_s + q_f + 2q_j$ oracle queries while $\mathcal{C}$ and $\mathcal{D}$ perform one. $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ run at most in time of $\mathcal{A}$ plus $q_s(t(\mathsf{AE}.\mathit{Enc}, \ell) + t(\mathsf{MAC.Tag}, \ell)) + q_f(t(\mathsf{MAC.Tag}, \ell) + t(\mathsf{MAC}.\mathit{Ver}, \ell)) + 2q_j \cdot (t(\mathsf{MAC}.\mathit{Ver}, \ell) + t(\mathsf{AE}.\mathit{Dec}, \ell)) + O(1)$.

Sketch. The proof follows a similar strategy to the one for Theorem 6. However, if win $\leftarrow 1$ is set in one of the first two winning conditions, we bound the advantage of $\mathcal{A}$ by reducing to an UF-CMA adversary that forges $T_{\text{S,RP}}$ and another that forges T_{RP}.

At last, note that although we did not formalize deniability, since we are using MAC tags it is straightforward to see that only the respective service provider is able to verify the tag it creates.

6 Integration with Messaging Protocols

In this section, we show how one would use IMF in a messaging setting by describing how to integrate Φ with the Signal protocol. We specifically choose Signal for our example use case because it is a popular target for integration. Towards generality, we also discuss the basic criteria for an IMF-compatible messaging protocol.

Background: Protocol abstractions. Signal is composed of the key exchange protocol X3DH [52] and a secure messaging scheme based on the Double Ratchet approach [51]. We first present an abstraction of these components, taken from [2, 43], to aid in our discussion of how IMF integrates into this setting.

We model an initial key exchange protocol as KE = (Gen, EGen, Send, Recv). It enables the initial key exchange between two parties at the start of a session in a secure messaging protocol. In more detail, algorithm KE.Gen generates a key pair, while KE.EGen generates an ephemeral key pair. KE.Send takes as input the sender private key, recipient public key, and recipient ephemeral public key and outputs a shared symmetric key k and ciphertext C_{KE}. The sender runs this algorithm when initiating a new session with a recipient to derive a shared key. When the recipient needs to derive the shared key, they run KE.Recv, which takes as input the recipient secret key, recipient ephemeral secret key, and ciphertext C_{KE}. It outputs the key k and sender public key. We note that while X3DH utilizes other keys, we simplify the key exchange protocol to contain only the long-term identity key and ephemeral key.

We model a secure messaging scheme SM = (InitS, InitR, Send, Recv) using the syntax from [43], which is based on the formalization and analysis from [2]. The algorithms SM.InitS and SM.InitR are run by the sender and recipient, respectively, to initialize their secret states on input the symmetric key k output by KE.Send and KE.Recv, respectively. Algorithm SM.Send is run by the sender on its state, associated data h, and message M to output its updated state and the ciphertext C. Finally, SM.Recv is run by the recipient on its state and the ciphertext to produce its updated state and the message M. We also assume that associated to scheme SM is an algorithm SM.GetK which takes as input the recipient state and a ciphertext and outputs the key used for decrypting the ciphertext from the state. Looking ahead, we will use this algorithm in our reporting flow.

Diagram. We show our integration in a flow diagram in Figure 7. The flow includes *setup*, in which users register with their respective (different) providers and perform key exchange; *messaging*, in which the sender sends the first message; and *reporting*, in which the recipient generates an abuse report for the sender's message. We assume that all pairwise links—that is, between a user and their provider and between the two providers—use authenticated channels (e.g., TLS channels).

We note that we do not use our protocol Φ as a black-box in the diagram because certain aspects of IMF break the abstraction boundary for existing formalizations of secure messaging. In particular, secure messaging already captures encrypting the plaintext and decrypting the ciphertext, as done in IMF.Send and IMF.Receive, respectively. To avoid breaking the security guaranteed by formalizations from prior work, we instead depict the updates required on top of secure messaging to enable abuse reporting, based on our IMF protocol. These are highlighted in yellow in the diagram.

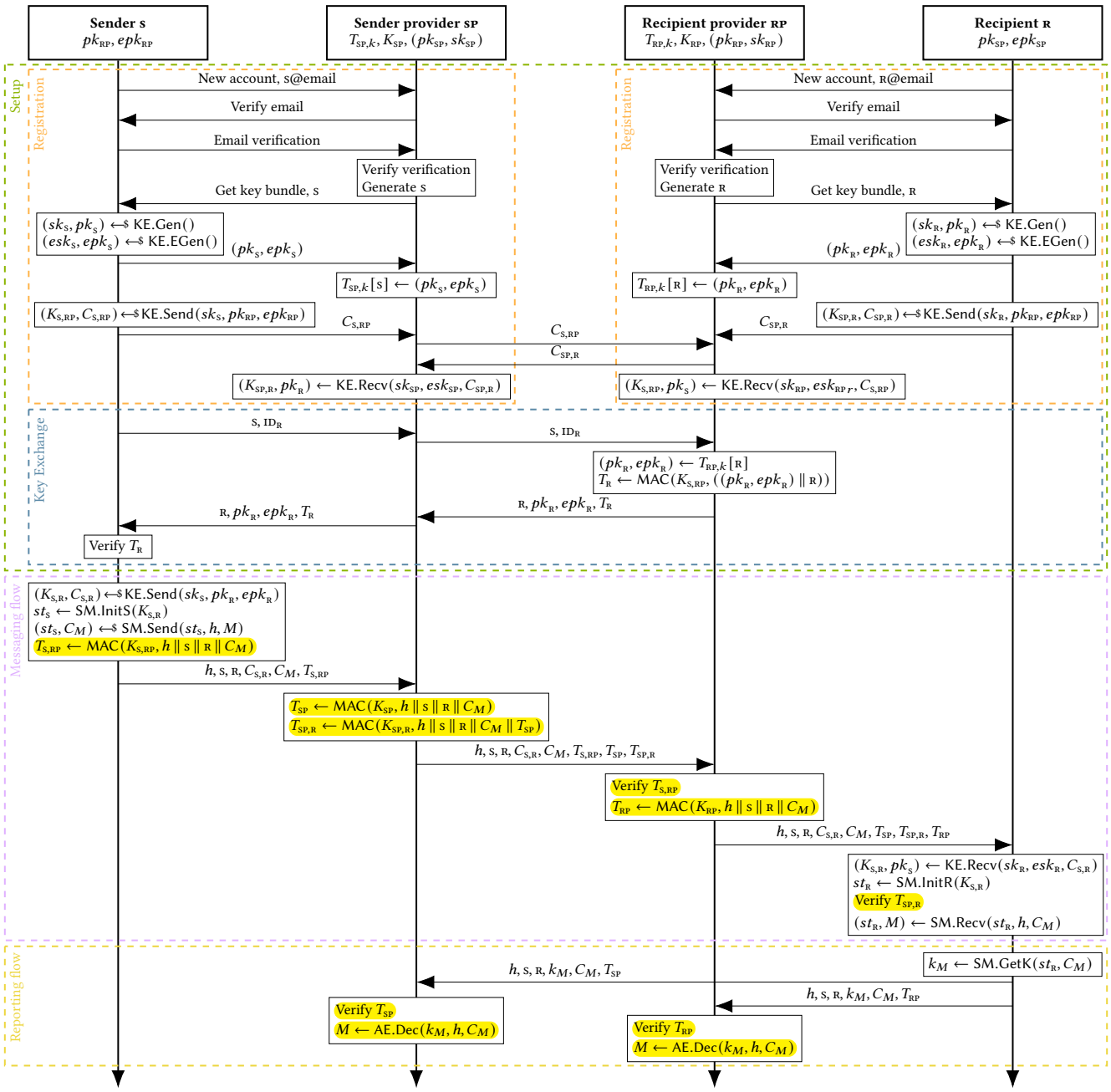


Figure 8: IMF protocol flow for sending the first message using key exchange protocol KE and secure messaging protocol SM. The highlighted sections are parts from our protocol Φ.

We now step through each part of the flow in more detail.

Setup. During setup, each provider generates its own (static) MAC key (K_{SP} and K_{RP}). The providers also have public keys and ephemeral public keys which we assume are known to all parties.

The users register with their respective providers to create accounts bound to some personal identifiers, such as emails. They then generate their keying material and send public keys to their providers for storage and distribution. The sender and RP can afterwards engage in the key exchange protocol to generate their shared key $K_{S,RP}$. Likewise, the recipient and SP perform a similar flow to generate $K_{SP,R}$.

Finally, the sender retrieves the necessary key material to send a message. We leave contact discovery out of scope, and assume that S knows R's personal identifier ID_R and can use it to request R's

key bundle from RP. Here, RP can use shared key $K_{S,RP}$ to compute a MAC over the keys for authenticity. In practice, S could leverage the channel with RP to hide ID_R from SP.

Messaging. The typical flow in messaging, as analyzed in [2], is that the sender generates a symmetric key k via the key exchange protocol and uses this key as input to SM.InitS to generate secure session state. This state is then input to SM.Send with the message to produce the ciphertext. The algorithm SM.Send abstracts out the public-key ratchet in Signal for generating one-time keys and the authenticated encryption with associated data (AEAD) scheme used for producing ciphertext C. Our protocol builds on top of this flow by computing $T_{S,RP} \leftarrow \mathsf{MAC.Tag}(K_{S,RP}, h \| S \| R \| C_M)$ and sending $T_{S,RP}$ along with the ciphertext. The providers then compute their tags by executing what is in Φ.FrankSP and Φ.FrankRP, respectively. Finally, before calling SM.Recv, the recipient verifies $T_{SP,R}$.

Reporting. The reporting stage can be seen as an add-on at the end of the secure messaging flow. To report M, R computes SM.GetK to extract the secret key used to decrypt C_M from its state, and then sends the key, ciphertext, and relevant franking tag to the judge to which it wishes to report. It sends this information through the authenticated channel it has established with the provider. Finally, the selected provider can run the respective judging algorithm as described in Figure 7.

Key exchange properties. While our formalization does not formally capture key exchange security, the guarantees of KE are an important consideration for ensuring secure integration of IMF within a messaging protocol. Here we discuss what properties are needed from KE vis-à-vis our security goals.

To start, KE must satisfy mutual authentication, or else the system would be vulnerable to meddler-in-the-middle (MITM) attacks. The KE must also satisfy key indistinguishability to ensure confidentiality of the system.

Our protocol does not require the identity hiding property [22]. That is, we do not require preventing passive adversaries, specifically service providers, from learning the S and R associated to a session. In our protocol, ciphertexts are always sent in conjunction with the relevant identities for the sender and/or the recipient; Φ is not a metadata hiding scheme [45]. Nevertheless, given that we assume communications between these parties are sent over secure channels, the property would still hold against network adversaries.

Security discussion. To describe how the system ensures security, we identify possible attacks and provide intuition on how our proposed protocol mitigates them. We assume secure KE and SM schemes, where the KE scheme meets the properties described above and the SM scheme satisfies confidentiality, authentication, key-commitment, forward secrecy, and post-compromise security [2]. Because the focus of this work is a secure abuse reporting scheme, we focus on possible attacks during the messaging and reporting stages. We also assume the clients know the public keys of the other service providers. Finally, recall we assume that at most one of the service providers is malicious during messaging and reporting and that there is a TLS connection between a user and its service provider, hence we do not consider network adversaries.

- *Messaging confidentiality attack:* During the messaging stage, a provider could try to learn information about the message sent. We highlight that the only change we make in this stage is to generate MAC tag $T_{S,RP}$ over non-confidential metadata and the ciphertext C_M. Thus, if a provider could break confidentiality of our protocol, then it could break the confidentiality of SM.
- *Messaging payload tampering:* A malicious provider could try to tamper with h, S, R, C_M, $T_{S,RP}$, $T_{SP,R}$, or $C_{S,R}$ in an authenticity attack. First note that either provider successfully tampering with $C_{S,R}$ or C_M such that the recipient accepts the payload as authentic would result in an authenticity attack against schemes KE and SM, respectfully.

  For the other values, we highlight that RP verifies $T_{S,RP}$, which is produced over h, S, R, C_M. Thus, if a malicious SP were to try to tamper with these values, then it would break the unforgeability of the MAC used to produce $T_{S,RP}$. If instead RP is malicious and tries to tamper with h, S, R, C_M, T_{SP}, then it would break the unforgeability of the MAC used to produce $T_{SP,R}$ that is verified by the recipient.
- *Report avoidance:* A malicious sender, who possibly colludes with one of the providers, could try to break reportability by generating a message that can be decrypted by the recipient but cannot be judged successfully by an honest provider. If any of h, S, R, C_M, T_{SP} is tampered with so that reporting fails, e.g., by RP, then this will break the unforgeability of the MAC used to produce tag $T_{SP,R}$ and verified by the recipient. If C_M is tampered with, then this will break the authenticity of SM. Thus, if decryption by the recipient succeeds and they submit a report, then both the verification of T_{SP} or T_{RP} and decryption of C_M by the judge must also succeed.
- *Report forgery:* A malicious recipient, possibly colluding with other parties, could try to report a message not sent by a sender in an effort to frame the sender for malicious behavior. If the adversary is able to generate values h, S, R, C_M and a valid franking tag that is accepted by the honest judge, then it can break the unforgeability of the MAC used to generate the franking tag. Furthermore, if the adversary is able to take honestly generated values h, S, R, C_M and produce a key $k'_M \neq k_M$, where k_M was the key used to produce C_M, such that it decrypts C_M to a different plaintext, then this would result in a key commitment attack against SM.

Out of scope. We now discuss some attacks which are out-of-scope. We do not consider attacks where malicious service providers aim to create fake user accounts, e.g., to burden another provider with messaging traffic or reports. We note that such attacks can be mitigated with other mechanisms, like Sybil attack resistance.

Furthermore, we assume both key distribution and key exchange are honest operations. This means that we do not consider a MITM attack by a malicious service provider, an attack which could be mitigated with a key transparency system [18, 42, 46, 47]. We also do not consider an attack where a user's or provider's keys are compromised by some third-party. We note that compromising a provider's keys could allow an adversary to forge abuse reports or tamper with honest messages such that they are no longer reportable. However, user key compromise can be mitigated by utilizing a scheme SM which meets both forward secrecy and post-compromise security, as is the case with the Signal protocol.

7 Discussion

In this section, we discuss desired properties for the user identifiers, along with some trade-offs, performance, and deployment considerations for our scheme Φ.

User identifiers. Thus far we have treated the user identities S, R opaquely. One approach is to simply make these the account usernames, like a phone number or email, so that both providers always learn the identities of external users communicating via their platform. This is the approach taken by WhatsApp and Messenger. However, this ignores a notable advantage of the server-to-server setting in its potential to provide better user privacy from other providers, as noted by Len et al. [43]. In particular, if a client connects to RP through its provider SP, then RP only needs to learn the recipient's identity for delivery and, moreover, could not learn the sender's identity by other means such as traffic analysis. This approach to more privacy is also promoted by the DMA, which specifies that providers should only share personal data from users when strictly necessary for functioning [26].

Here we briefly sketch the properties that would be desirable for integrating privacy-preserving identifiers for IMF. The main goal for pseudonyms is that other platforms should not learn personally identifiable information about its non-users. In other words, SP knows the sender's identity, while RP knows the recipient's identity; however, SP should not learn the recipient's identity and RP should not learn the sender's identity. Assigning a single static pseudonym per user is enough to satisfy this, e.g., by having each provider compute a keyed PRF over each user identifier.

Because RP may not trust SP to generate such identities, RP should also be able to verify something about pseudonyms to aid in mitigating abuse on its platform. Ideally, a server would be able to (a) verify obliviously that the sender owns a resource that is hard to generate, like a phone number and (b) that the pseudonym(s) assigned to the sender are associated to a single resource, in order to allow blocking the user if necessary. Goal (a) ensures that it is hard for a single user to generate many identities on a platform, hence it is challenging for a blocked user to create a new account and continue abuse of the platform. Goal (b) ensures that each pseudonym uniquely identifies an account of that provider's platform.

However, goal (a) is difficult to satisfy. In particular, verifying a phone number requires either trusting SP about the sender's registration or requiring the sender to authenticate with RP (as WhatsApp proposes [48]), which would break the sender's privacy. Due to the lack of trust between providers, solutions based on anonymous credentials, where the issuer is a trusted party, do not solve this problem [4, 17, 19, 53].

We further note that our solution above using PRFs for computing pseudonyms meets neither verifiability goal. We leave a solution for generating privacy-preserving identifiers that meets all goals we outline above to future work.

Authenticated channels between all parties. We modeled symmetric keys between the sender and the recipient provider ($K_{S,RP}$) and between the sender provider and the recipient ($K_{SP,R}$). They allow parties to verify that the service provider between them did not tamper with the information being sent. This is important to ensure Receive only receives messages that can be reported to an honest provider, since R can detect if RP tampered with T_{SP}.

While key agreements can be an expensive operation, it only needs to occur once per external platform user. As a further optimization, these key exchanges could be integrated with the sender retrieving the key material from a new contact and sending a message. This would remove the extra round-trips. However, these keys do present some additional challenges since they require service providers to manage a secure session with each user. In the case of small messaging platforms interoperating with platforms with billions of users, this could require them to grow their key management infrastructure much more than their user base.

Cryptographic deployment. For our scheme, we require a mutual authentication key agreement scheme that provides secrecy and a key rotation algorithm that support asynchronous messaging environments. This is satisfied for instance by the widely deployed X3DH and Double Ratchet. We also require MACing and symmetric encryption schemes. For the encryption, we do require the scheme to be key-committing. While there is no standard for key-committing encryption, there is growing interest in the community on upgrading the current standards to support commitment properties as well [8, 15] and multiple efficient schemes have been proposed [1, 9, 10, 33].

On the other hand, for the MACs used in Φ, we require only T_{RP} and T_{SP} to be the output of collision resistant MACs (see Theorems 6, 7, and 8). Therefore, the remaining MACs can be faster polynomial-based MACs [12, 24]. However, note that we are targeting messaging applications, and Signal-based apps [51] generally rely on HMAC [6], which is collision resistant.

Performance. Compared to the message franking scheme used by Meta [27, 31], our scheme requires two additional key agreements. However, these are executed only once, so the cost can be amortized throughout the span of a user's conversations. Then, we also require the encryption scheme to be key-committing. Prior work has shown optimized key-committing AEAD constructions [1, 9, 10, 33]. Finally, our scheme requires three additional MACs. Assuming these are HMAC with SHA-512, these would impose $3 \times 512 = 1536$ bits or 192 extra bytes to the payload.

8 Related Work

This section discusses previous work related to symmetric (SMF) and asymmetric message franking (AMF), as well as prior work related to interoperable messaging.

SMF. Starting with Facebook's message franking protocol [27] and formal analysis [23, 31], multiple works have followed [20, 21, 25, 30, 36, 44, 58], enhancing SMFs with additional security properties, such as only partially revealing a message during reporting [20, 44]. Most of these works focus on a setting with a single, trusted server.

Some works, however, do consider a malicious server, although in different settings than ours. Chen and Fischlin [21] consider malicious servers in their threat model, but their work focuses on causality preservation across multiple messages. Eskandarian [25] proposes SMFs for metadata-hiding platforms, using secret sharing across multiple servers. Moreover, Gregoire et al. [30] propose SMFs for encrypted messaging systems based on onion encryption, such as mixnets. The first scheme does assume the moderator is honest, and all assume the moderator is known in advance. In contrast, IMF allows the recipient to select the moderator during reporting.

AMF. Asymmetric message franking [55] was introduced to address abuse reporting for metadata-hiding settings or communication settings with a third-party moderator. They make use of asymmetric cryptographic primitives. Subsequent work has introduced AMFs with source tracing [38], AMFs for group settings [40], and AMFs for illegal content only [35]. Due to the cost of asymmetric primitives, AMFs have not been widely deployed like SMFs. Hence, for performance reasons, we focus on constructions relying only on symmetric primitives.

Interoperable E2EE messaging. Since the release of the DMA, several works analyzing the challenges of these new settings have been released [13, 14]. While these works do not focus on abuse prevention, they all include a discussion of the challenges of message franking, blocklisting, and spam prevention, and point out the challenges with moderation due to users being able to create multiple identities across services. Additional recent work proposes a key transparency protocol for interoperable E2EE messaging [28].

In this work, we focused on an architecture model based on the ones proposed by Len et al. [43]. However, in their work, the recipient provider acts as the judge of any reported message, while here our architecture allows the recipient to select to which provider to report a message. It can be the case that the sender provider has a better infrastructure for verifying reports, which is why we opt to give the recipient the possibility of choosing. Moreover, in [43], the authors suggest using AMFs [55] (see above) and assume that providers behave honestly when it comes to abuse prevention. In contrast, here we consider scenarios where one of the providers is malicious and formalize new security definitions for message franking under a stronger threat model.

Meta has released documentation for their interoperability approach [48]. WhatsApp uses by default a client-to-server architecture, which might allow WhatsApp to learn the IP address of users. It requires users from other providers to register an identity with WhatsApp. In more detail, users authenticate to WhatsApp via the standard OpenID protocol [54] and JSON Web Tokens [39]. The other providers must generate signed tokens for their users, which WhatsApp verifies. Given the client-to-server architecture, WhatsApp can use standard SMFs and simply have WhatsApp always act as the moderator. Additionally, given that users register with WhatsApp, there are no user privacy guarantees for external users interoperating with WhatsApp. Facebook Messenger's approach works in the same way as WhatsApp. The Matrix.org Foundation has also proposed an architecture where the Matrix protocol is used to bridge between service providers [34]. Here we base our model on a different architecture.

Finally, the IETF established the More Instant Messaging Interoperability (MIMI) [37] working group. The current protocol draft [5] proposes using a server-to-server architecture and relying on MLS for the communication among clients. The service provider of the user who initiates the chat (room) serves as the hub for that chat, which means it is in charge of maintaining the state of that chat, distributing messages, among others.

The draft proposes a message franking scheme very similar to the one used by Meta [31], where the hub serves as the central server in charge of franking the messages. Our approach, instead, lets the recipient choose the moderator. Additionally, we consider malicious servers in our threat model, while the MIMI message franking requires trusting the hub. We further note that MIMI's solution is insecure in our threat model, since it does not prevent misbehavior by other providers.

9 Conclusion

We introduced interoperable symmetric message franking (IMF) focused on a server-to-server architecture for messaging. We introduced new security definitions adapted to an interoperable setting, where none of the servers is guaranteed to be honest. We detailed an IMF construction satisfying these security definitions and discussed the goals and challenges of user identities in the IMF setting. Finally, we discussed the overhead and deployment challenges.

Acknowledgments

This work was funded in part by NSF grant CNS-2120651.

References

[1] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg. 2022. How to abuse and fix authenticated encryption without key commitment. In *USENIX Security '22*.
[2] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. 2019. The double ratchet: security notions, proofs, and modularization for the signal protocol. In *EUROCRYPT*.
[3] Apple. 2025. Apple iMessage. https://support.apple.com/messages.
[4] Foteini Baldimtsi and Anna Lysyanskaya. 2013. Anonymous credentials light. In *ACM CCS, 2013*.
[5] R. L. Barnes, M. Hodgson, K. Kohbrok, R. Mahy, T. Ralston, and R. Robert. 2024. More Instant Messaging Interoperability (MIMI) using HTTPS and MLS. IETF working group. https://www.ietf.org/archive/id/draft-ietf-mimi-protocol-02.html
[6] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. 1996. Keying hash functions for message authentication. In *Advances in Cryptology — CRYPTO*.
[7] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. 1997. A concrete security treatment of symmetric encryption. In *FOCS*. IEEE.
[8] Mihir Bellare, Shay Gueron, Viet Tung Hoang, Julia Len, Sanketh Menda, and Thomas Ristenpart. 2024. Building the Next Generation of AEAD. Presented at RWC 2024.
[9] Mihir Bellare and Viet Tung Hoang. 2022. Efficient schemes for committing authenticated encryption. In *Advances in Cryptology – EUROCRYPT*.
[10] Mihir Bellare and Viet Tung Hoang. 2024. Succinctly-committing authenticated encryption. In *Advances in Cryptology — CRYPTO*.
[11] Mihir Bellare and Chanathip Namprempre. 2000. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology – ASIACRYPT*.
[12] Daniel J. Bernstein. 2005. The Poly1305-AES Message-Authentication Code. In *Fast Software Encryption*.
[13] Jenny Blessing and Ross Anderson. 2023. One Protocol to Rule Them All? On Securing Interoperable Messaging. In *Security Protocols Workshop*.
[14] Marc Bourreau. 2022. DMA Horizontal and Vertical Interoperability Obligations. *Centre on Regulation in Europe (CERRE)* (2022).
[15] A.A. Bozhko. 2023. Properties of AEAD algorithms. IETF working group. https://www.ietf.org/archive/id/draft-irtf-cfrg-aead-properties-02.html
[16] Dick Brouwer. 2024. Making messaging interoperability with third parties safe for users in Europe. Engineering at Meta.
[17] Jan Camenisch and Anna Lysyanskaya. 2001. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology —- EUROCRYPT*.
[18] Melissa Chase, Apoorvaa Deshpande, Esha Ghosh, and Harjasleen Malvai. 2019. SEEMless: Secure End-to-End Encrypted Messaging with less Trust. In *ACM SIGSAC Conference on Computer and Communications Security CCS*.
[19] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. 2014. Algebraic MACs and keyed-verification anonymous credentials. In *ACM CCS, 2014*.
[20] Long Chen and Qiang Tang. 2018. People who live in glass houses should not throw stones: targeted opening message franking schemes. *Cryptology ePrint Archive* (2018).
[21] Shan Chen and Marc Fischlin. 2024. Integrating Causality in Messaging Channels. In *Advances in Cryptology – EUROCRYPT*.
[22] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. 2017. A formal security analysis of the signal messaging protocol. In *IEEE European Symposium on Security and Privacy*.

[23] Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. 2018. Fast message franking: From invisible salamanders to encryptment. In *CRYPTO*.

[24] Morris J Dworkin. 2007. Sp 800-38d. recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac.

[25] Saba Eskandarian. 2024. Abuse Reporting for Metadata-Hiding Communication Based on Secret Sharing. In *USENIX Security '24*.

[26] European Parliment and Council of European Union. 2022. Digital Markets Act. https://eur-lex.europa.eu/eli/reg/2022/1925

[27] Facebook. 2017. Messenger Secret Conversations Technical Whitepaper. Meta.

[28] Neenu Garg and Mohammad Tariq Elahi. 2024. (P)KT-IEE: Secure key transparency protocols for interoperable end-to-end encrypted message systems. In *FOCI 2024*. Privacy Enhancing Technologies Board.

[29] Google. 2025. Google Messages. https://m.essages.google.com.

[30] Matthew Gregoire, Margaret Pierce, and Saba Eskandarian. 2025. Onion Franking: Abuse Reports for Mix-Based Private Messaging. In *NDSS '25*.

[31] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. 2017. Message franking via committing authenticated encryption. In *Advances in Cryptology – CRYPTO*.

[32] GSM Association. 2024. RCS Now in iOS: A New Chapter for Mobile Messaging. https://www.gsma.com/newsroom/article/rcs-nowin-ios-a-new-chapter-for-mobile-messaging/ Accessed: 2024-09-30.

[33] Viet Tung Hoang and Sanketh Menda. 2024. Robust AE With Committing Security. In *Advances in Cryptology – ASIACRYPT*.

[34] Matthew Hodgson. 2024. Opening up communication silos with Matrix 2.0 and the EU Digital Markets Act. Presented at FOSDEM 2024.

[35] Zhengan Huang, Junzuo Lai, Gongxian Zeng, and Jian Weng. 2024. Mild Asymmetric Message Franking: Illegal-Messages-Only and Retrospective Content Moderation. In *Advances in Cryptology – ASIACRYPT*.

[36] Loïs Huguenin-Dumittan and Iraklis Leontiadis. 2021. A message franking channel. In *ICISC*.

[37] IETF. 2022. More Instant Messaging Interoperability (MIMI). IETF working group.

[38] Rawane Issa, Nicolas Alhaddad, and Mayank Varia. 2022. Hecate: Abuse reporting in secure messengers with sealed sender. In *USENIX Security '22*.

[39] M. Jones, J. Bradley, and N. Sakimura. 2015. JSON Web Token (JWT). IETF working group. https://datatracker.ietf.org/doc/html/rfc7519

[40] Junzuo Lai, Gongxian Zeng, Zhengan Huang, Siu Ming Yiu, Xin Mu, and Jian Weng. 2023. Asymmetric Group Message Franking: Definitions & Constructions. In *Advances in Cryptology – EUROCRYPT*.

[41] Ravie Lakshmanan. 2024. GSMA Plans End-to-End Encryption for Cross-Platform RCS Messaging. *The Hacker News* (2024). https://thehackernews.com/2024/09/gsma-plans-end-to-end-encryption-for.html Accessed: 2024-09-30.

[42] Julia Len, Melissa Chase, Esha Ghosh, Kim Laine, and Radames Cruz Moreno. 2024. OPTIKS: An Optimized Key Transparency System. In *USENIX Security '24*.

[43] Julia Len, Esha Ghosh, Paul Grubbs, and Paul Rösler. 2023. Interoperability in end-to-end encrypted messaging. *Cryptology ePrint Archive* (2023).

[44] Iraklis Leontiadis and Serge Vaudenay. 2023. Private Message Franking with After Opening Privacy. In *Information and Communications Security*.

[45] J Lund. 2018. Technology preview: Sealed sender for Signal. Signal blog. https://signal.org/blog/sealed-sender/

[46] Harjasleen Malvai, Lefteris Kokoris-Kogias, Alberto Sonnino, Esha Ghosh, Ercan Oztürk, Kevin Lewi, and Sean F. Lawlor. 2023. Parakeet: Practical Key Transparency for End-to-End Encrypted Messaging. In *NDSS 2023*.

[47] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. 2015. CONIKS: Bringing Key Transparency to End Users. In *USENIX Security '15*.

[48] Meta. 2024. Messaging Interoperability. https://developers.facebook.com/m/messaging-interoperability/ Accessed: 2024-10-01.

[49] Meta. 2025. Messenger. https://www.messenger.com/.

[50] Casey Newton. 2023. Three ways the European Union might ruin WhatsApp. *The Verge* (2023). https://www.theverge.com/23001152/whatsapp-eu-digital-markets-act-messaging-interoperable Accessed: 2024-09-30.

[51] Trevor Perrin and Moxie Marlinspike. 2016. The Double Ratchet Algorithm. Signal documentation. https://signal.org/docs/specifications/doubleratchet/

[52] Trevor Perrin and Moxie Marlinspike. 2016. The X3DH Key Agreement Protocol. Signal documentation. https://signal.org/docs/specifications/x3dh/

[53] Michael Rosenberg, Jacob White, Christina Garman, and Ian Miers. 2023. zk-creds: Flexible anonymous credentials from zksnarks and existing identity infrastructure. In *IEEE Symposium on Security and Privacy*.

[54] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore. 2023. OpenID Connect Core. https://openid.net/specs/openid-connect-core-1_0.html

[55] Nirvan Tyagi, Paul Grubbs, Julia Len, Ian Miers, and Thomas Ristenpart. 2019. Asymmetric message franking: Content moderation for metadata-private end-to-end encryption. In *Advances in Cryptology–CRYPTO*.

[56] WhatsApp. 2019. Stopping Abuse: How WhatsApp Fights Bulk Messaging and Automated Behavior. WhatsApp. https://faq.whatsapp.com/5957850900902049

[57] WhatsApp. 2025. WhatsApp. https://www.whatsapp.com/.

[58] Hiroki Yamamuro, Keisuke Hara, Masayuki Tezuka, Yusuke Yoshida, and Keisuke Tanaka. 2021. Forward Secure Message Franking. In *ICISC*.