Abuse Reporting and Enforcement for Third-Party Moderators in Private Messaging

Matthew Gregoire
UNC Chapel Hill
Chapel Hill, NC
mattyg@cs.unc.edu

Jade Keegan
UNC Chapel Hill
Chapel Hill, NC
kjade@cs.unc.edu

Saba Eskandarian
UNC Chapel Hill
Chapel Hill, NC
saba@cs.unc.edu

***Abstract*—We introduce new techniques for verifiable reporting of abusive messages in private messaging platforms. Our techniques are compatible with both metadata-hiding messaging systems, where the platform does not know who speaks to whom, and with third-party moderation, where the platform is not involved in the process of verifying or judging reported content.**

While prior work in this space considers the question of how a moderator would verify reports, prior works do not address the question of how a moderator and platform would collaborate to enforce moderation decisions. In a setting where the platform does not wish to be involved in or responsible for enforcing moderation decisions, or in federated settings where it is not clear who would be responsible for enforcement, this presents an additional challenge.

Our work solves this problem with a lightweight credentialing and revocation mechanism that does not involve the platform in moderation enforcement at all. In order to support this added functionality, we build on Asymmetric Message Franking (Crypto '19) and improve performance over the original scheme, reducing moderator computation and communication costs to verify reports by $6\times$ and $7\times$, respectively.

1. Introduction

Widespread deployment of end-to-end encrypted (E2EE) messaging and efforts to protect messaging metadata – both in deployed systems and numerous academic research projects – promise a more private future for our communications. However, lack of appropriate accountability measures and the potential for rampant abuse pose major barriers to adoption. Users expect to be able to report abusive messages they receive (e.g., spam, misinformation, inappropriate content, etc.), but at first glance, hiding message contents or metadata from platforms seems to make this reporting impossible. While users can always simply share the messages they receive with platform moderators, moderators also need a way to ensure that the reports they receive correspond to authentic messages sent through the platform, not forgeries. A rich series of works have addressed this problem using variants of a *message franking* technique first introduced by Meta for use in Messenger and WhatsApp [6], [18], [25], [38], [41].

While this deployed message franking solution fits Meta's use case, it does not support situations where message metadata is hidden from the platform, nor does it support situations where the moderator shares no secret key material with the platform. These concerns are especially relevant for federated or interoperable messaging use cases, where a single platform does not control the entire messaging ecosystem [33], or for cases where messaging apps wish to allow for multiple independent moderators on their platform. Various academic works have since studied how to extend message franking to function in these more challenging settings [19], [23], [28], [43].

Unfortunately, while these prior works have shown how to handle verification of abuse reports, they have not considered how such a process would result in consequences for abusive users.

When the platform is also the moderator, enforcement of moderation decisions is a non-issue, because the platform has complete authority to take action on its own decisions. However, when the moderator is a third party, a platform must either fully trust this moderator's decisions or require proof that a message is abusive before it can take enforcement actions. In the latter case, the deniability properties of abuse reporting schemes would make such a proof impossible. Moreover, even if this technical problem were resolved, such an arrangement would put some degree of moderation responsibility back on the platform, by either deciding whether the moderator has made the right decision, or even just offering tacit approval of the moderator's decisions. This puts platforms that outsource moderation, either as a policy decision or as an architectural requirement of a federated system, in a difficult position.

Prior works focus on moderator verification of abuse reports, but they do not provide answers to these important practical enforcement questions. We will show that by addressing these enforcement challenges in the design of an abuse reporting system, opportunities arise for new, more efficient verification techniques. As a result, we can simultaneously attain both functionality and performance improvements over prior, work while also broadening the kinds of real-world E2EE messaging deployments that can

benefit from abuse report verification.

1.1. Our Contributions

This paper introduces *Enforceable Message Franking* (EMF), a new message franking primitive that makes three contributions toward improved abuse reporting and verification schemes.

Connecting reports with ramifications. Whereas previous abuse reporting schemes focus only on the mechanism for verifying abuse reports, we show how to integrate the report verification process with revocation of access. We do this without requiring the platform to become directly involved in endorsing moderation decisions or taking enforcement actions. In fact, because the platform is used in a completely black-box manner, the platform need not be aware of what moderation decisions are made, or even the fact that moderation is happening at all. Broadening the scope of moderation processes in this way makes our schemes usable for federated or interoperable messaging systems, as well as cases where the platform does not want to be held responsible for moderation decisions made by third parties. It also allows for platforms to support multiple different moderators without modification to the scheme.

Faster moderator verification procedures. A consequence of tying together report verification with user ramifications is a more direct connection between a user and its moderator(s), including issuance of an access credential or key material to participate in good standing as a user (see technical overview for details). We show that this stronger connection can be used to significantly improve the performance of abuse report verification schemes, especially for the moderator.

Prior work requires both a message recipient and moderator to verify a zero-knowledge proof of the well-formedness of an abuse report [43]. We replace proof verification by the moderator with the more lightweight task of verifying a MAC. While in principle any MAC can be used, we show that a choice of well-structured algebraic MAC allows for the message recipient to do work comparable to the proof verification required in prior schemes, while offering the moderator a significantly more lightweight report verification process.

More nuanced deniability definitions. Message franking requires a strong notion of deniability: reports should be verifiable to the moderator, but they should be deniable to all other parties, including under a variety of key compromise scenarios. Prior works either require a full E2EE messaging abstraction as part of their deniability definitions [28], or they define deniability by implicitly making simplifying assumptions that rule out schemes like ours [43]. In particular, Asymmetric Message Franking [43] assumes that the message franking material sent to the message recipient is identical to the report sent to the moderator.

Our schemes involve computing a MAC tag intended for the moderator, in addition to a proof intended for the message recipient, when sending messages. The proof can be discarded after verification, reducing abuse report size, but poses a problem for deniability definitions that do not distinguish between deniability of messages and reports. We adapt the framework used to define deniability in prior works to handle this new situation without needing to incorporate a full E2EE messaging abstraction into our definitions.

Empirical results. We implement and evaluate the performance of our scheme in comparison to prior work, showing that our scheme matches or outperforms prior work in the same setting for all steps of the message franking process. The moderator, whose performance benefits the most from our scheme, has a $6.1\times$ reduction in computation costs and $7.6\times$ reduction in communication costs when verifying reports.

We also introduce a more lightweight version of our scheme, EMF-lite, that makes reasonable deniability compromises in exchange for increased performance. EMF-lite has a $21.8\times$ reduction in computation costs and a $15.3\times$ reduction in communication costs to verify a report compared to prior work. It also cuts time to send a message by 35% and reduces time to verify a received message by 26%.

2. Background and Related Work

Message franking was first introduced by Meta as a way to verify abuse reports in Messenger [38] and was first formally studied by Grubbs et al. [25]. The original message franking design has the user produce an additional commitment c_2 to their message plaintext, with the commitment randomness included separately in the message ciphertext c_1. The platform then computes a MAC tag t on c_2 and delivers c_1, c_2, t to the recipient. To report a message, the recipient sends t, c_2, and the commitment randomness from c_1 to the platform, alongside the message. The MAC on the commitment allows the platform to verify that this was a message that really was sent through the platform, and the binding property of the commitment ensures that the message being reported matches the one that was sent. A number of subsequent works have improved abstractions for message franking, identified and corrected aspects of its implementation, or broadened the scope of its applicability [6], [18], [39], [41].

Unlike other approaches to dealing with abuse in private messaging which may raise questions about infringement on users' private messaging expectations, e.g., client-side scanning [1], [2], [4], [32] or message traceback [5], [10], [35], [40], [44], [45], [46], message franking does not require any caveats to the privacy expectations of E2EE messaging. While the message franking scheme deployed by Meta requires the platform to have access to users' messaging metadata, other message franking schemes have done away with this requirement, either by requiring stronger cryptographic tools or restricting to particular classes of metadata-hiding communication systems [19], [23], [27], [28], [43]. A partially overlapping set of works support *third-party moderation*, where the platform does not need

to share key material with the moderator [24], [27], [28], [43].

The most directly related prior work to ours is Asymmetric Message Franking (AMF) [43], which is the first work to support message franking for metadata-hiding messaging and third-party reporting. Hecate [28] offers similar security properties to AMF, but it requires a per-message preprocessing phase and, during message delivery, a platform-computed signature for each message sent through the system. AMF is also the only message franking scheme that does not make any assumptions about the messaging platform or require any kind of involvement by the platform in the message franking process. It operates entirely independently of the platform, allowing for any underlying mechanism to deliver messages.

AMF has recently been extended to support additional features [27], but the extension does not have an implementation and uses more heavy-weight techniques. Our work directly improves upon AMF, simultaneously increasing its functionality and improving its performance.

3. Technical Overview

This section briefly sketches our technical ideas, which we describe in greater detail in subsequent sections.

3.1. Reporting with Enforcement of Moderation Decisions

At a high level, this paper is concerned with the problem of handling abuse in private messaging, and our solutions are applicable both to situations where message metadata is hidden from the platform and where the platform only provides E2EE. We present a scheme that allows users who receive a message they find objectionable (spam, inappropriate content, etc.) to report that message to a third-party moderator, who can then cryptographically verify that the report represents the authentic contents and metadata of a message sent through the platform.

Our scheme goes beyond prior work by further considering what a moderator should do after deciding that a verified report has indeed violated some platform policy. In this case, the moderator can revoke the message sender's access to the platform, preventing them from sending future messages. We illustrate the high-level design of our solution in Figure 1.

To allow for the moderator to unilaterally remove users, we have users register with the moderator prior to sending messages. During this process, the moderator gives each user a short-lived *sender certificate*. The sender certificate consists of a user identifier id, an expiration time t_{exp}, and some cryptographic key material to be described later that is signed under the moderator's signing key sk_M. This certificate is transmitted by the sender to a new conversation partner and updated regularly as evidence of permission to send messages through the platform. If revocation is necessary, the moderator can simply stop issuing new certificates for the sender. This adds minimal additional amortized cost over many conversations but allows the moderator to enforce messaging permissions without the platform being involved, or even aware, of its choices.

Limitations. While the choice of short-lived certificates means there may be a slight delay between moderation decision and enforcement, we believe this is a reasonable tradeoff, given that manual decision-making about abuse reports may itself introduce some delay into the process. An affected user can use a local block list to protect against repeat offenses until a system-wide block goes into effect.

In addition, certificates in our scheme are implemented as signatures on session information under a moderator's signing key. This provides globally verifiable proof that a user has agreed to the use of a particular moderator. However, since blocking only consists of not issuing a new certificate when an old one expires, there is never proof that a user has been banned by a moderator. While we don't further explore schemes that hides users' choices of moderators, we briefly present a mechanism to achieve this in Section 5.1.

An inherent limitation of our design is that detecting and dropping messages from revoked users must happen on the client side, as the platform is fully unaware of enforcement decisions made by the moderator. Although it comes with additional client-side costs, this is an inherent consequence of the policy decision of decoupling the platform from making or enforcing moderation decisions, not a technical limitation unique to our solution.

Supporting multiple moderators. Our design naturally extends to a setting with multiple independent and unrelated moderators that work with the same platform. In this setting, users can select their moderators, or they may use a combination of moderators of their own choosing and those they are required to use, e.g., by local laws. Each moderator operates independently, issuing sender certificates and making its own moderation/revocation decisions.

Users only need to exchange the sender certificates for the chosen moderators they have in common, which can be determined either by sending the user's list of moderators prior to initiating a new conversation or by using a private set intersection (PSI) protocol [21]. If a user starts a conversation with sender certificates for some set of moderators and is later revoked by any one of them, the message recipient will see this and can refuse future messages from the same sender.

3.2. Cryptographic Core: Enforceable Message Franking

We now show how the setting described in Section 3.1 allows us to construct a new message franking scheme, called *Enforceable Message Franking* (EMF), that supports third-party moderation while hiding message metadata and outperforming asymmetric message franking (AMF) [43], the only prior work which could previously be used in such a setting.

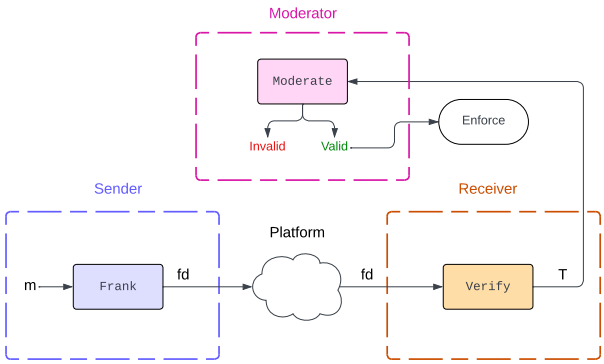


Figure 1: Our Enforceable Message Franking scheme produces franking data fd that is sent alongside an encrypted message. Message recipients can use fd to verify that messages they get can be reported. In the event a message is reported, the moderator can check the validity of the report and then take concrete enforcement action to prevent the sender from sending future messages. Throughout this process, the platform only plays the role of a delivery system oblivious to moderation decisions, simply routing messages from senders to receivers.

The basic idea. Observe that since a moderator in our setting needs to issue a sender certificate to each user, this occasion can also be used to communicate a shared secret. Alongside the sender certificate, the moderator selects and sends a MAC key k_f, which is unique to the sender. To tie the MAC key to the certificate, the moderator includes a value k_{pub}, which is a binding and hiding commitment to k_f, in the sender certificate.

Once a message sender and the moderator share a key, so long as a sender is assumed to be honest, message franking is very simple. The sender produces a MAC tag $T \leftarrow \mathsf{MAC.Tag}(k_f, m)$ on any message m it sends and appends the tag T to m before encrypting it to the message recipient. If the recipient wishes to report the message, it sends T to the moderator, which then verifies the MAC tag T to ensure the message was actually sent by the alleged sender. This approach is very similar to that of deployed message franking systems, except the role of the platform is replaced by the sender, which computes the MAC tag T itself.

Dealing with malicious senders. Unfortunately, this simple approach fails to capture an important property known as *accountability*. A malicious sender can simply compute an incorrect tag T, ensuring that whenever a recipient reports one of its messages, it will be rejected by the moderator since it is malformed.

In order to provide accountability, we require the sender to additionally produce a zero-knowledge proof π that the tag T is well-formed, i.e., it is a MAC formed using the same key k_f that is committed to by k_{pub} in the corresponding sender certificate. The message recipient verifies this proof to be convinced that the tag T would be accepted if it were to report the message. A malformed tag would be caught and rejected at this stage before the message is shown to the human user.

Improving performance. The high-level design sketched above dramatically reduces costs to the moderator, which only needs to verify a MAC tag to verify reports, but it may increase costs for the message recipient, which needs to verify a proof of the well-formedness of a MAC tag with respect to a committed key in the sender certificate. We show that, by selecting appropriate MAC and commitment schemes, we can balance the costs between recipient and moderator such that moderator costs still drop significantly compared to prior work, while maintaining or even reducing receiver costs. Specifically, we evaluate our scheme using the algebraic MAC $\mathsf{MAC}_{\mathsf{GGM}}$, where the key $k_f = (x_0, x_1)$ and tags on a message m are computed as $T \leftarrow g^{x_0 + x_1 H(m)}$. This MAC scheme is proven secure in the generic group model [11], [42] and used as part of the Signal private group system [12]. We could also instantiate our scheme in the standard model using its slightly less performant sibling $\mathsf{MAC}_{\mathsf{DDH}}$.

Beyond the choice of concrete MAC instantiation, we further improve performance by separating out infrequent session initiation costs, e.g., verifying sender certificates, from per-message proof verification costs.

In prior work, AMF addresses the problem of malicious senders by replacing the MAC with a carefully constructed zero knowledge proof that acts as a designated-verifier signature [29], [30], [31], [34], verifiable only by the moderator and the message recipient. Our scheme avoids using a single proof that must convince two different parties of the reportability of a message, reducing both communication and computation costs. We report on the empirical evaluation of our scheme in Section 6.

3.3. Handling Deniability

The designs of modern E2EE messaging protocols make a particular effort to provide their users with *deniability*. This is the property that if a message recipient R publicly announces that the sender S sent a particular message, S can deny that claim. Formally, for any real message that R could claim S sent, including any cryptographic evidence presented as proof (e.g., ciphertexts, keys, proofs, etc.), S should be able to show that R could have forged the message and evidence with the keys available to it.

When it comes to abuse reporting, we aim to selectively break deniability, allowing moderators to verify reports, but maintaining deniability of report contents for all other parties. This means that while a moderator vetted by a platform can be confident in the authenticity of reports it receives, a malicious receiver which leaks messages to the press or to government agencies (either willingly or under coercion) is unable to provide proof of their veracity.

The key technical challenge to achieving deniability is the presence of a zero-knowledge proof that could potentially convince anyone that the MAC tag T is truly a MAC on m, even if they don't have the key k_f. Following a similar

approach to AMF, we introduce additional components to the proof, so that the proof actually claims that either the MAC is well-formed, *or* the creator of the proof holds a secret known only to the message recipient. This means that the receiver of the message can always forge a valid-looking proof. The accountability of the scheme only relies on the proof convincing the verifier, so this change ensures accountability without compromising other security properties. Compared to AMF, we can prove a simpler statement, as their proofs need to convince both the message recipient and the moderator, whereas ours can be discarded after the recipient verifies them.

More flexible definitions. Hecate [28] and AMF [43] each give excellent definitional frameworks for reasoning about deniability in abuse reporting for private messaging. Unfortunately, these two frameworks are not a perfect fit for our scenario.

Hecate models the underlying E2EE messaging system as part of its deniability definitions. This is necessary in their case because their scheme requires platform involvement in the delivery process and is therefore intertwined with the messaging system itself. On the other hand, AMF, like our scheme, does not need to do this. AMF operates completely independently of the encryption mechanism, simply presenting a set of algorithms that produce and verify tags that can be appended to messages before encryption and processed after decryption.

Unfortunately, the AMF deniability definitions focus only on the deniability of abuse reports, since the underlying messaging protocol ensures the deniability of the messages themselves. This is fine for the AMF setting where their scheme syntax ensures that abuse reports are exactly the same as the tags that are appended to messages by the sender.

Although our schemes enjoy a similar independence from the underlying messaging infrastructure as AMF, we improve upon AMF's performance by introducing different tag contents for the message recipient and moderator. The recipient portion is discarded after message delivery. This allows for a more efficient construction but is not supported by AMF syntax and definitions, which require identical receiver and moderator tags. Thus, we adapt AMF's definitional framework to add the flexibility needed to express and reason about our performance improvements.

New deniability tradeoffs. We consider two variants of our EMF scheme that achieve two slightly different degrees of deniability. Our main EMF construction offers a degree of deniability identical to that of AMF. This means we satisfy three deniability definitions that always assume that a message sender's keys are compromised, in addition to any other information that is available to someone trying to verify an alleged message, e.g., receiver keys or moderator keys.

We also consider a weaker form of deniability that assumes a message sender's keys are *not* compromised. This is motivated by the observation that many sender key compromise scenarios would be accompanied by more concrete real-world evidence that a message was sent, rendering cryptographic deniability less relevant. We feel this setting is well-motivated in light of recent work on how deniability properties are or are not applied in courts [13]. In this setting, a lighter-weight version of our scheme, EMF-lite, offers even better performance than EMF (although both outperform AMF). The key difference is that the weaker deniability attacker does not have access to the MAC key k_f. This means the MAC tag T itself appears pseudorandom (assuming an appropriately selected MAC scheme), so an additional step of encrypting the MAC tag in the full EMF scheme, a detail elided in our high-level description above, is not needed.

We give full descriptions of our two schemes in Sections 5.1 and 5.2. We begin with EMF-lite because it is slightly simpler conceptually, and EMF can be viewed as EMF-lite augmented with an additional step to provide stronger deniability.

4. Formalizing Enforceable Message Franking

This section formalizes our EMF primitive, describing the formal syntax and its usage as well as presenting our security definitions.

Notation. We denote assignment of the value x to y using $y \leftarrow x$, and assignment to y uniformly at random from $\mathcal{X}$ as $y \xleftarrow{R} \mathcal{X}$. Tables, initialized as $T \leftarrow \{\}$, store multiple values, and can be added to via $T.\mathsf{add}(v)$. A function $\mathsf{negl}(x)$ is *negligible* if, for all $c > 0$, there exists an x_0 such that, for all $x > x_0$, $\mathsf{negl}(x) < \frac{1}{x^c}$. We use $\perp$ as a special character indicating protocol failure.

4.1. Syntax and Usage

We formally define an *enforceable message franking scheme* (EMF) as a seven-tuple of algorithms (KeyGenUser, KeyGenMod, IssueCert, AcceptSession, Frank, Verify, Moderate). The algorithms KeyGenUser and KeyGenMod capture setup for users and the moderator, and IssueCert captures the operations run by the moderator when registering a user or renewing its credentials. The Frank, Verify, and Moderate functions are called when sending, receiving, or moderating a message, respectively. AcceptSession includes the one-time operations that a message recipient runs when receiving a message from someone for the first time or after a credential is rotated. We separate this function for the purposes of distinguishing between one-time and per-message operations, but this algorithm could also be considered a part of Verify with no ramifications for security.

An EMF sheme uses the following syntax:

- $\mathsf{KeyGenUser}(1^\lambda) \mapsto (\mathsf{sk}_U, \mathsf{pk}_U)$: this function takes as input a security parameter, and outputs a user's secret and public keys $(\mathsf{sk}_U, \mathsf{pk}_U)$.
- $\mathsf{KeyGenMod}(1^\lambda) \mapsto (\mathsf{sk}_M, \mathsf{pk}_M)$: this function takes as input a security parameter, and outputs a moderator's secret and public keys $(\mathsf{sk}_M, \mathsf{pk}_M)$.

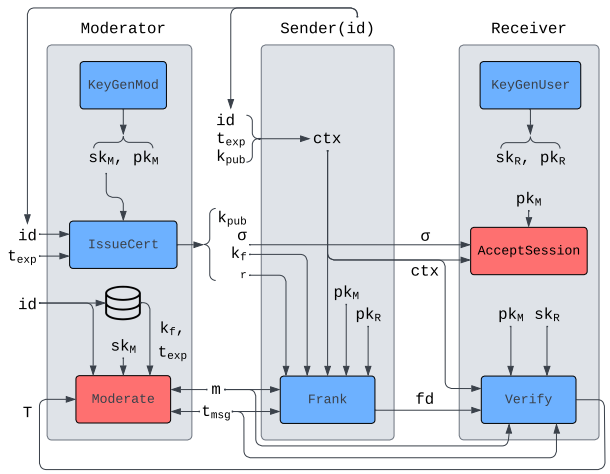


Figure 2: Illustration of how to use an EMF scheme. Boxes in blue produce outputs that are used elsewhere in the protocol. Boxes colored in red check assertions and output 0 or 1.

- $\mathsf{IssueCert}(\mathsf{sk}_M, \mathsf{id}, t_{\mathsf{exp}}) \mapsto (k_f, r, k_{\mathsf{pub}}, \sigma)$: this function takes as input a moderator's secret key sk_M, a user id, and an expiry time t_{exp}. It returns a key k_f to be shared between the moderator and the user, a commitment k_{pub} to the key and associated randomness r, and a certificate σ.
- $\mathsf{AcceptSession}(\mathsf{pk}_M, \mathsf{ctx}, \sigma) \mapsto 0/1$: this function takes as input the moderator's public key pk_M, a messaging session context ctx, and a certificate σ. It checks that the signature is well-formed.
- $\mathsf{Frank}(\mathsf{pk}_M, \mathsf{pk}_R, k_f, r, m, t_{\mathsf{msg}}, \mathsf{ctx}) \mapsto \mathsf{fd}$: this function takes in the moderator's and receiver's public keys pk_M and pk_R, a key k_f, associated randomness r, a message m, a message timestamp t_{msg}, and a session context ctx. It returns associated franking data fd to be sent to the message receiver.
- $\mathsf{Verify}(\mathsf{pk}_M, \mathsf{sk}_R, m, t_{\mathsf{msg}}, \mathsf{fd}, \mathsf{ctx}) \mapsto T/\perp$: this function takes as input the moderator's public key pk_M, a receiver's secret key sk_R, a message m and its timestamp t_{msg}, franking data fd, and a session context ctx. It checks that the franking data is valid and returns either a franking tag T or $\perp$.
- $\mathsf{Moderate}(\mathsf{sk}_M, k_f, \mathsf{id}, t_{\mathsf{exp}}, m, t_{\mathsf{msg}}, T) \mapsto 0/1$: this function takes as input the moderator's secret key sk_M, a key k_f, a sender id, an expiration time t_{exp}, message m, a message timestamp t_{msg}, and a franking tag T. It returns a bit b indicating whether or not the reported message is valid.

We define correctness for an EMF scheme in Appendix C, where we also include additional technical preliminaries.

Figure 2 demonstrates how the usage of an EMF scheme, showing which algorithms are run by which parties and when. When used as part of a messaging scheme, the output of Frank is included alongside the message in the end-to-end encrypted ciphertext sent to the recipient, i.e., it is appended to the end of the plaintext message before encryption. This means that from a confidentiality perspective, EMF benefits from the privacy properties of the underlying messaging scheme. As such, our security definitions in Section 4.2 focus on ensuring that attackers cannot abuse or avoid the abuse reporting mechanism, not on protecting confidentiality.

4.2. Security Definitions

We require the following security properties of an EMF scheme:

- *Accountability*: a malicious sender cannot evade the abuse reporting mechanism by sending a message that is accepted by the recipient who runs Verify, but is then rejected as an invalid report by a moderator who runs Moderate. This property is analogous to the *sender binding* property in AMF.
- *Unforgeability*: a malicious receiver cannot trick a moderator into thinking someone sent a message that they did not send. This property is analogous to the *receiver binding* property in AMF.
- *Deniability*: only the moderator is able to verify reports. No third party can use the message franking information to verify the authenticity of messages, even under a variety of key compromise scenarios.

This section discusses the security definition for each property in detail.

Accountability. Our accountability security experiment models an adversary who attempts to send a message that evades the EMF report verification mechanism. The adversary is given a valid messaging certificate and is allowed access to SendMsg and ForgeMsg oracles that model a message recipient and platform, respectively, and return the the outputs of Verify and Moderate on the adversary's chosen messages or reports.

After interacting with the oracles, the adversary outputs a maliciously designed message of its choice (inclusive of changing the certificate if it wishes). The experiment outputs 1 if this message is accepted by Verify, i.e., Verify outputs a tag T that is not $\perp$, but Moderate outputs 0.

Definition 4.1 (Accountability)**.** We define the franking accountability experiment $\mathsf{ACCT}[\mathcal{A}, \mathcal{F}, Q_S, \lambda]$ in Figure 3 with respect to an efficient adversary $\mathcal{A}$ which makes at most Q_S queries to the $\mathcal{O}_{\mathsf{SendMsg}}$ oracle, an enforceable franking scheme $\mathcal{F}$, and a security parameter λ.

We define the *accountability advantage* of $\mathcal{A}$ as

$$\mathsf{ACCTAdv}[\mathcal{A}, \mathcal{F}, Q_S, \lambda] = \Pr\big[\mathsf{ACCT}[\mathcal{A}, \mathcal{F}, Q_S, \lambda] = 1\big].$$

We say that a scheme $\mathcal{F}$ has *accountability* if, for all efficient adversaries $\mathcal{A}$,

$$\mathsf{ACCTAdv}[\mathcal{A}, \mathcal{F}, Q_S, \lambda] \leq \mathsf{negl}(\lambda).$$

Unforgeability. Our unforgeability security experiment models an adversary who attempts to frame another user

$\mathsf{ACCT}[\mathcal{A}, \mathcal{F}, Q_S, \lambda]$

$\mathsf{id}, t_{\mathsf{exp}} \leftarrow \mathcal{A}(1^\lambda)$
$\mathsf{sk}_R, \mathsf{pk}_R \leftarrow \mathsf{KeyGenUser}(1^\lambda)$
$\mathsf{sk}_M, \mathsf{pk}_M \leftarrow \mathsf{KeyGenMod}(1^\lambda)$
$k_f, r, k_{\mathsf{pub}}, \sigma \leftarrow \mathsf{IssueCert}(\mathsf{sk}_M, \mathsf{id}, t_{\mathsf{exp}})$
$\mathsf{ctx} \leftarrow (\mathsf{id}, k_{\mathsf{pub}}, t_{\mathsf{exp}})$
$(m^*, t^*_{\mathsf{msg}}, \mathsf{fd}^*, \mathsf{ctx}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{SendMsg}}, \mathcal{O}_{\mathsf{ForgeMsg}}}(\mathsf{pk}_R, \mathsf{pk}_M, k_f, r, \mathsf{ctx}, \sigma, 1^\lambda)$
$b_0 \leftarrow \mathsf{AcceptSession}(\mathsf{pk}_M, \mathsf{ctx}^*, \sigma^*)$
$T^* \leftarrow \mathsf{Verify}(\mathsf{pk}_M, \mathsf{sk}_R, m^*, t^*_{\mathsf{msg}}, \mathsf{fd}^*, \mathsf{ctx}^*)$
$b_1 \leftarrow T^* \neq \perp$
$b_2 \leftarrow \mathsf{Moderate}(\mathsf{sk}_M, k_f, \mathsf{id}, t_{\mathsf{exp}}, m^*, t^*_{\mathsf{msg}}, T^*)$
output $b_0 \wedge b_1 \wedge (\neg b_2)$

$\mathcal{O}_{\mathsf{SendMsg}}(m, t_{\mathsf{msg}}, \mathsf{fd}, \mathsf{ctx}^*, \sigma^*)$

$b_0 \leftarrow \mathsf{AcceptSession}(\mathsf{pk}_M, \mathsf{ctx}^*, \sigma^*)$
$T \leftarrow \mathsf{Verify}(\mathsf{pk}_M, \mathsf{sk}_R, m, t_{\mathsf{msg}}, \mathsf{fd}, \mathsf{ctx}^*)$
if $b_0 = 1$: **return** T
return $\perp$

$\mathcal{O}_{\mathsf{ForgeMsg}}(m^*, t^*_{\mathsf{msg}}, T^*)$

return $\mathsf{Moderate}(\mathsf{sk}_M, k_f, \mathsf{id}, t_{\mathsf{exp}}, m^*, t^*_{\mathsf{msg}}, T^*)$

Figure 3: Accountability

$\mathsf{FORG}[\mathcal{A}, \mathcal{F}, Q_S, Q_F, \lambda]$

$\mathsf{pk}_R, \mathsf{id}, t_{\mathsf{exp}} \leftarrow \mathcal{A}(1^\lambda)$
$\mathsf{sk}_M, \mathsf{pk}_M \leftarrow \mathsf{KeyGenMod}(1^\lambda)$
$k_f, r, k_{\mathsf{pub}}, \sigma \leftarrow \mathsf{IssueCert}(\mathsf{sk}_M, \mathsf{id}, t_{\mathsf{exp}})$
$\mathsf{ctx} \leftarrow (\mathsf{id}, k_{\mathsf{pub}}, t_{\mathsf{exp}})$
$T_{\mathsf{seen}} \leftarrow \{\}$
$\mathsf{win} \leftarrow 0$
$\mathcal{A}^{\mathcal{O}_{\mathsf{SendMsg}}, \mathcal{O}_{\mathsf{ForgeMsg}}}(\mathsf{pk}_M, \mathsf{ctx}, \sigma, 1^\lambda)$
output win

$\mathcal{O}_{\mathsf{SendMsg}}(m, t_{\mathsf{msg}}, \mathsf{ctx}^*)$

$\mathsf{fd} \leftarrow \mathsf{Frank}(\mathsf{pk}_M, \mathsf{pk}_R, k_f, r, m, t_{\mathsf{msg}}, \mathsf{ctx}^*)$
$T_{\mathsf{seen}} \leftarrow T_{\mathsf{seen}} \cup \{(m, t_{\mathsf{msg}})\}$
return fd

$\mathcal{O}_{\mathsf{ForgeMsg}}(m^*, t^*_{\mathsf{msg}}, T^*)$

if $\mathsf{Moderate}(\mathsf{sk}_M, k_f, \mathsf{id}, t_{\mathsf{exp}}, m^*, t^*_{\mathsf{msg}}, T^*)$:
  if $(m^*, t^*_{\mathsf{msg}}) \notin T_{\mathsf{seen}}$:
    $\mathsf{win} \leftarrow 1$
return win

Figure 4: Unforgeability

as having sent a message that they did not in fact send. The adversary is given an honest user's certificate and is allowed to request arbitrary messages from that honest user, including arbitrary timestamps, through a SendMsg oracle. It can also send any report of its choosing to be moderated through a ForgeMsg oracle. The adversary wins if it produces a valid report on a message, timestamp pair for which it did not previously get franking data fd from the SendMsg oracle.

Definition 4.2 (Unforgeability)**.** We define the franking unforgeability experiment $\mathsf{FORG}[\mathcal{A}, \mathcal{F}, Q_S, Q_F, \lambda]$ in Figure 4 with respect to an efficient adversary $\mathcal{A}$ who makes at most Q_S queries to the $\mathcal{O}_{\mathsf{SendMsg}}$ oracle and Q_F queries to the $\mathcal{O}_{\mathsf{ForgeMsg}}$ oracle, a enforceable franking scheme $\mathcal{F}$, and a security parameter λ.

We define the *unforgeability advantage* of $\mathcal{A}$ as

$$\mathsf{FORGAdv}[\mathcal{A}, \mathcal{F}, Q_S, Q_F, \lambda] = \mathsf{Pr}\big[\mathsf{FORG}[\mathcal{A}, \mathcal{F}, Q_S, Q_F, \lambda] = 1\big].$$

We say that a scheme $\mathcal{F}$ has *unforgeability* if, for all efficient adversaries $\mathcal{A}$, and all $Q_S, Q_F \in \mathbb{N}$,

$$\mathsf{FORGAdv}[\mathcal{A}, \mathcal{F}, Q_S, Q_F, \lambda] \leq \mathsf{negl}(\lambda).$$

Deniability. Our deniability definitions are each parameterized by a Forge algorithm that produces fake franking data and a fake report without access to the sending user's secrets. The adversary's goal is to distinguish between the forged materials and the real franking data and report generated by Frank and Verify. Both the forger and the distinguishing adversary are given the public keys of the receiver and moderator, as well as the certificate for an honestly generated user. The distinguisher is additionally given access to the honest user's secrets.

We have three different deniability definitions that capture deniability under three different scenarios:

- *Universal deniability*. This models deniability to an arbitrary third party who does not have any additional keys beyond the ones described above.
- *Receiver compromise deniability*. This models deniability in the case that the receiver's secrets are compromised, either by accident, by force, or because the receiver wants to reveal their key to prove the validity of a message. In this case, the Forge algorithm and the adversary are both given the receiver's secret key sk_R as an additional input.
- *Moderator compromise deniability*. This models deniability in the case that the moderator's secrets are compromised. Analogously to the receiver compromise scenario, the Forge algorithm and the adversary are both given the moderator secrets – both the moderator key sk_M and the key k_f shared with the honest user.

An important observation is that our definitions are written with respect to the franking data fd produced by Frank, *and* the report T that is output by Verify and actually sent to the moderator in correct usage of the scheme. We define deniability in this way because requiring only report deniability fails to capture all the ways an adversary might try to convince a third party of a message's veracity. In

prior work where the franking data and report are the same, this distinction is unimportant. However, in our setting, deniability of the report itself is easy to satisfy while still leaving gaps that would allow anyone to verify a message's authenticity.

Remark 4.3. In order to reduce redundancy, we collapse all three deniability definitions into one figure and color-code which keys are given to which party. We also color-code the differences between the forgery algorithms in our deniability proofs. Throughout this paper, universal deniability components are shown in black, moderator compromise components are shown in blue, and receiver compromise components are shown in yellow. In some cases, in the proofs, the same additional component is present for both moderator and receiver compromise deniability. In these cases, that component is shown in green.

Definition 4.4 (Deniability)**.** We define the universal, moderator compromise, and receiver compromise deniability experiments $\{\mathsf{U},\mathsf{MC},\mathsf{RC}\}\text{–}\mathsf{DENY}[\mathcal{A},\mathcal{F},N,\mathsf{Forge},Q_S,b,\lambda]$ in Figure 5 with respect to an efficient adversary $\mathcal{A}$ which makes at most Q_S queries to the $\mathcal{O}_{\mathsf{SendMsg}}$ oracle, a enforceable franking scheme $\mathcal{F}$, a report forgery algorithm Forge, a bit b, and a security parameter λ.

We define the *deniability advantages*, $\{\mathsf{U},\mathsf{MC},\mathsf{RC}\}\text{–}\mathsf{DENYAdv}[\mathcal{A},\mathcal{F},\mathsf{Forge},Q_S,\lambda]$, of $\mathcal{A}$ as

$$\Big|\Pr\big[\{\mathsf{U},\mathsf{MC},\mathsf{RC}\}\text{–}\mathsf{DENY}[\mathcal{A},\mathcal{F},\mathsf{Forge},Q_S,0,\lambda]=1\big]-\Pr\big[\{\mathsf{U},\mathsf{MC},\mathsf{RC}\}\text{–}\mathsf{DENY}[\mathcal{A},\mathcal{F},\mathsf{Forge},Q_S,1,\lambda]=1\big]\Big|.$$

We say that a scheme $\mathcal{F}$ has *deniability* if there exist (potentially distinct) forgery algorithms Forge for all three deniability games such that, for all efficient adversaries $\mathcal{A}$,

$$\{\mathsf{U},\mathsf{MC},\mathsf{RC}\}\text{–}\mathsf{DENYAdv}[\mathcal{A},\mathcal{F},\mathsf{Forge},Q_S,\lambda]\le\mathsf{negl}(\lambda).$$

Weak deniability. Note that our deniability definitions, written to mirror the properties offered by AMF, always give the distinguisher the message sender's secret keys, regardless of the key compromise scenario under consideration. This means that every deniability definition captures the case where the message sender's keys have been compromised by the party attempting to verify a message, but they are not available to the party who is trying to prove that a message is real. Giving the sender's keys to the forger would make deniability trivial, as the forger could just honestly produce the franking materials itself.

The AMF paper [43] provides a detailed analysis showing that the combination of key compromise scenarios captured by these definitions is in some sense the best possible, i.e., all other attainable deniability definitions under their threat model are implied by these. However, we argue that it is possible that these definitions may actually be stronger than what is needed in some use cases, and we define a slightly weakened version of deniability where the distinguisher is not given the sender's secrets.

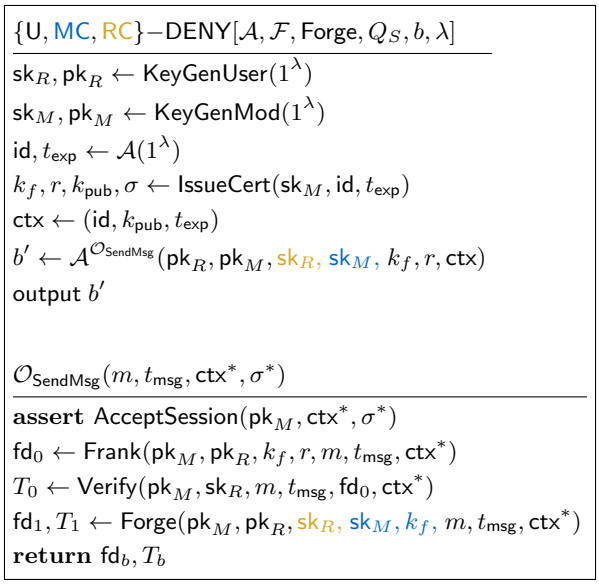


Figure 5: Deniability security experiments. Universal deniability is shown in black. Additions necessary for moderator compromise deniability are shown in blue, and those necessary for receiver compromise deniability are shown in yellow.

In general, if a user's keys are available to a distinguisher only, this means that the distinguisher must have been able to compromise the user's device, cloud backups, or other sensitive storage medium. In light of a recent study on how formal deniability properties are not really considered in courts [13], such a complete compromise may suffice to provide alternative evidence of a message's validity in many cases. To be clear, this is not a formal or absolute statement, as a careful user may delete messages, intentionally insert forged messages on their storage media, etc. Nonetheless it is an observation that for many users, this level of deniability is undercut by other aspects of messaging application design.

In the context of our specific schemes, observe that while universal and receiver compromise deniability are weakened by not giving the distinguisher access to the sender's secrets, moderator compromise deniability is not. This is because the sender's only secrets are k_f and randomness r. The former is shared with the moderator, and the latter is generated deterministically by the moderator, so the weak and standard deniability definitions in this case would be the same. We thus only define weak versions of universal and receiver compromise deniability. In Section 5.1, we show a scheme that achieves weak deniability but has better performance than our scheme that satisfies the stronger deniability definition.

Definition 4.5 (Weak Deniability)**.** We define the universal and receiver compromise weak deniability experiments $\{\mathsf{U},\mathsf{RC}\}\text{–}\mathsf{WDENY}[\mathcal{A},\mathcal{F},N,\mathsf{Forge},Q_S,b,\lambda]$ in Figure 5 with respect to an efficient adversary $\mathcal{A}$ which makes at most Q_S queries to the $\mathcal{O}_{\mathsf{SendMsg}}$ oracle, a enforceable franking scheme $\mathcal{F}$, a report forgery algorithm Forge, a bit

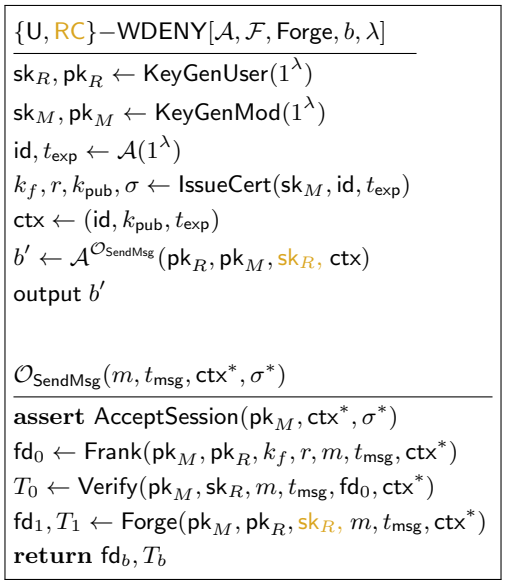


Figure 6: Weak deniability experiments. Universal deniability is shown in black. Additions necessary for receiver-compromise deniability are shown in yellow. Note that we do not include a separate weak version of moderator compromise deniability, as it would be equivalent to the stronger moderator compromise deniability definition in Figure 5.

b, and a security parameter λ.

We define the respective *weak deniability advantages*, $\{\mathsf{U}, \mathsf{RC}\}\text{−}\mathsf{WDENYAdv}[\mathcal{A}, \mathcal{F}, \mathsf{Forge}, Q_S, \lambda]$, of $\mathcal{A}$ as

$$\Big|\Pr\big[\{\mathsf{U}, \mathsf{RC}\}\text{−}\mathsf{WDENY}[\mathcal{A}, \mathcal{F}, \mathsf{Forge}, Q_S, 0, \lambda] = 1\big] - \Pr\big[\{\mathsf{U}, \mathsf{RC}\}\text{−}\mathsf{WDENY}[\mathcal{A}, \mathcal{F}, \mathsf{Forge}, Q_S, 1, \lambda] = 1\big]\Big|.$$

We say that a scheme $\mathcal{F}$ has *weak deniability* if there exist (potentially distinct) forgery algorithms Forge for the two weak deniability games and for moderator compromise deniability (Definition 4.4) such that, for all efficient adversaries $\mathcal{A}$ which make at most Q_S queries to $\mathcal{O}_{\mathsf{SendMsg}}$,

$$\{\mathsf{U}, \mathsf{RC}\}\text{−}\mathsf{WDENYAdv}[\mathcal{A}, \mathcal{F}, \mathsf{Forge}, Q_S, \lambda] \leq \mathsf{negl}(\lambda)$$
$$\text{and } \mathsf{MC}\text{−}\mathsf{DENYAdv}[\mathcal{A}, \mathcal{F}, \mathsf{Forge}, Q_S, \lambda] \leq \mathsf{negl}(\lambda).$$

5. Building Enforceable Message Franking

This section describes our constructions for enforceable message franking: EMF-lite and EMF. We present EMF-lite first as it can be viewed as a stepping-stone toward our full EMF construction, containing most of the same ideas but satisfying the weaker deniability property described in Section 4.2.

5.1. EMF-Lite

Our EMF-lite construction relies on an initial certificate issuance step, where the moderator shares a MAC key k_f with the sender and issues a short-lived certificate that the sender uses to prove its authorization to send messages.

Certificate issuance. Prior to communication with a receiver, the sender must register with a moderator that both parties agree upon. Upon registration, the moderator calls IssueCert for the sender's user ID id and a chosen expiration time t_{exp} in the near future (on the span of days to weeks). It computes a pseudorandom value r by evaluating a PRF on input $(\mathsf{id}, t_{\mathsf{exp}})$, to be used as commitment randomness later. It then generates a MAC key k_f. While in principle we could use any MAC, we instantiate the MAC with $\mathsf{MAC}_{\mathsf{GGM}}$ as shown in Figure 8, so k_f takes the form (x_0, x_1). Next, the moderator computes a Pedersen multi-commitment $k_{pub} = g_0^{x_0} g_1^{x_1} h^r$ to k_f, which will be used later in the sender's zero-knowledge proofs. Finally, the moderator uses its own secret signing key to sign the triple $(\mathsf{id}, k_{\mathsf{pub}}, t_{\mathsf{exp}})$, where id is the sender's identity. We call this triple the *session context*, or ctx. The context and its signature σ act as a short-lived certificate, confirming that the sender is in the moderator's good graces and should be allowed to communicate with the receiver. The moderator sends the user the secret values k_f and r, as well as σ.

Once the moderator has issued the certificate, the sender initiates a communication session by providing the receiver with the certificate values (ctx, σ). The receiver verifies the signature σ under the moderator's verification key. If verification succeeds, the receiver accepts the messaging session for the lifespan of the certificate.

Note that, if hiding the sender's choices of moderators is desireable, the certificate can include a commitment to the sender's ID, rather than the ID itself. This hides the sender's identity in the certificate, and we can use the same designated verifier proof techniques described below to convince the receiver that the certificate is tied to the correct ID. This would incur a performance overhead for session initialization to compute and verify the proof, but because this operation happens rarely (once a day or once a week, say), the overall impact on our scheme is marginal.

Sending messages. To send a message m, the sender must complete a "franking" operation to generate franking data fd that it includes alongside m in the underlying private messaging system. First, it uses k_f to compute a MAC tag T on the message m and its corresponding metadata (the sender's id and the message timestamp t_{msg}). This tag T can later serve as a report for the message m.

In addition to T, the sender must also produce some proof to convince the receiver, who does not know k_f, that T is an honestly computed MAC tag on the values $(\mathsf{id}, m, t_{\mathsf{msg}})$. To do this, the sender generates a zero-knowledge proof that it knows values k_f and r such that $k_{\mathsf{pub}} = \mathsf{Com.Commit}(k_f, r)$, and such that $T = \mathsf{MAC.Tag}(k_f, (\mathsf{id}, m, t_{\mathsf{msg}}))$.

If the sender only produces this proof, we would lose deniability, as the proof could serve as a signature of knowledge on the message. Thus anyone could verify the authenticity of the message by verifying the proof rather than T. We address this issue with techniques similar to

$$\mathcal{R} = \left\{((k_f, r, y), (k_{\mathsf{pub}}, R, T)) : \left(k_{\mathsf{pub}} = \mathsf{Com.Commit}(k_f, r) \wedge T = \mathsf{MAC.Tag}(k_f, (\mathsf{id}, m, t_{\mathsf{msg}}))\right) \vee (R = g^y)\right\}$$

$\underline{\mathsf{KeyGenUser}(1^\lambda)}$
$\mathsf{sk} \xleftarrow{R} \mathbb{Z}_q$
$\mathsf{pk} \leftarrow g^{\mathsf{sk}}$
return sk, pk

$\underline{\mathsf{KeyGenMod}(1^\lambda)}$
$k_{\mathsf{PRF}} \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$
$\mathsf{sk}_{\mathsf{sig}}, \mathsf{pk}_{\mathsf{sig}} \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
$\mathsf{sk} \leftarrow (k_{\mathsf{PRF}}, \mathsf{sk}_{\mathsf{sig}})$
$\mathsf{pk} \leftarrow \mathsf{pk}_{\mathsf{sig}}$
return sk, pk

$\underline{\mathsf{IssueCert}(\mathsf{sk}_M, \mathsf{id}, t_{\mathsf{exp}})}$
$k_{\mathsf{PRF}}, \mathsf{sk}_{\mathsf{sig}} \leftarrow \mathsf{sk}_M$
$k_f \leftarrow \mathsf{MAC.KeyGen}(1^\lambda)$
$r \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, (\mathsf{id}, t_{\mathsf{exp}}))$
$k_{\mathsf{pub}} \leftarrow \mathsf{Com.Commit}(k_f, r)$
$\mathsf{ctx} \leftarrow (\mathsf{id}, k_{\mathsf{pub}}, t_{\mathsf{exp}})$
$\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{sig}}, \mathsf{ctx})$
return $k_f, r, k_{\mathsf{pub}}, \sigma$

$\underline{\mathsf{AcceptSession}(\mathsf{pk}_M, \mathsf{ctx}, \sigma)}$
$b \leftarrow \mathsf{Sig.Verify}(\mathsf{pk}_M, \mathsf{ctx}, \sigma)$
return b

$\underline{\mathsf{Frank}(\mathsf{pk}_M, \mathsf{pk}_R, k_f, r, m, t_{\mathsf{msg}}, \mathsf{ctx})}$
$\mathsf{id}, k_{\mathsf{pub}}, \cdot \leftarrow \mathsf{ctx}$
$\alpha \xleftarrow{R} \mathbb{Z}_q$
$R \leftarrow (\mathsf{pk}_R)^\alpha$
$E_R \leftarrow g^\alpha$
$T \leftarrow \mathsf{MAC.Tag}(k_f, (\mathsf{id}, m, t_{\mathsf{msg}}))$
$\pi \leftarrow \mathsf{ZK.Prove}(\mathcal{R},$
$\quad (k_f, r, \cdot), (k_{\mathsf{pub}}, \cdot, T), m, \mathsf{id}, t_{\mathsf{msg}})$
$\mathsf{fd} \leftarrow (R, E_R, T, \pi)$
return fd

$\underline{\mathsf{Verify}(\mathsf{pk}_M, \mathsf{sk}_R, m, t_{\mathsf{msg}}, \mathsf{fd}, \mathsf{ctx})}$
$R, E_R, T, \pi \leftarrow \mathsf{fd}$
$\mathsf{id}, k_{\mathsf{pub}}, t_{\mathsf{exp}} \leftarrow \mathsf{ctx}$
$b_0 \leftarrow R = (E_R)^{\mathsf{sk}_R}$
$b_1 \leftarrow t_{\mathsf{msg}} < t_{\mathsf{exp}}$
$b_2 \leftarrow \mathsf{ZK.Verify}(\mathcal{R},$
$\quad (k_{\mathsf{pub}}, R, T), m, \mathsf{id}, t_{\mathsf{msg}}, \pi)$
if $\neg(b_0 \wedge b_1 \wedge b_2)$:
$\quad$ **return** $\perp$
return T

$\underline{\mathsf{Moderate}(\mathsf{sk}_M, k_f, \mathsf{id}, t_{\mathsf{exp}}, m, t_{\mathsf{msg}}, T)}$
$b_0 \leftarrow t_{\mathsf{msg}} < t_{\mathsf{exp}}$
$b_1 \leftarrow \mathsf{MAC.Verify}(k_f, (\mathsf{id}, m, t_{\mathsf{msg}}), T)$
return $b_0 \wedge b_1$

Figure 7: EMF-lite construction.

$\underline{\mathsf{MAC}_{\mathsf{GGM}}.\mathsf{Tag}(k_f, m)}$
$x_0, x_1 \leftarrow k_f$
return $g^{x_0 + x_1 H(m)}$

$\underline{\mathsf{Com.Commit}(k_f, r)}$
$x_1, x_2, \ldots, x_n \leftarrow k_f$
$c \leftarrow g_1^{x_1} g_2^{x_2} \cdots g_n^{x_n} h^r$
return c

$\underline{\mathsf{MAC}_{\mathsf{DDH}}.\mathsf{Tag}(k_f, m)}$
$x_0, x_1, y_0, y_1, z \leftarrow k_f$
$r \xleftarrow{R} \mathbb{Z}_q$
$\sigma_w \leftarrow g^r$
$\sigma_x \leftarrow \sigma_w^{r(x_0 + x_1 H(m))}$
$\sigma_y \leftarrow \sigma_w^{r(y_0 + y_1 H(m))}$
$\sigma_z \leftarrow \sigma_w^z$
return $(\sigma_w, \sigma_x, \sigma_y, \sigma_z)$

Figure 8: Two MACs compatible with our EMF schemes [11], and the Pedersen multi-commitment scheme.

AMF: the sender computes a proof that either (1) these two clauses in the original proof are true, or (2) it knows the discrete log of some other value R. To compute R, the sender selects a random $\alpha \in \mathbb{Z}_q$ and computes $E_R \leftarrow g^\alpha$ and $R \leftarrow (\mathsf{pk}_R)^\alpha$, where pk_R is the receiver's public key. The sender can then send its franking data (R, E_R, T, π) to the receiver along with its message m.

Note that the discrete log of R is then $\alpha \cdot \mathsf{sk}_R$, so the sender is unable to prove knowledge of the discrete log of R without knowledge of sk_R. The receiver can therefore check that (1) the proof is valid, and (2) $R = E_R^{\mathsf{sk}_R}$, to be convinced that the MAC tag must be valid with respect to the correct k_{pub} commitment. However, without knowledge of sk_R, check (2) is not possible, and anyone can generate proofs that successfully verify by proving the second clause with an arbitrarily chosen R. This allows our scheme to maintain deniability.

The last thing the receiver checks is that $t_{\mathsf{msg}} < t_{\mathsf{exp}}$. If true, it displays the message to the end user and stores (m, t_{msg}, T) for potential future reporting.

Making reports. To report a message, the receiver sends the values (m, t_{msg}, T) along with the sender id to the moderator. The moderator then looks up the values k_f and t_{exp} for the relevant user's certificate. With these, it simply checks that $t_{\mathsf{msg}} < t_{\mathsf{exp}}$, and that $\mathsf{MAC.Verify}(k_f, (\mathsf{id}, m, t_{\mathsf{msg}})) = 1$. If so, the report is accepted. The moderator can then take action, if necessary, by declining to issue a new certificate to the sender when their current one expires.

Construction 5.1 (EMF-lite)**.** Our EMF-lite construction appears in Figure 7 and makes use of the following primitives:

- A group $\mathbb{G}$ of prime order q with generator g, where DDH is hard and the knowledge of exponent assumption holds.
- A PRF $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^{\lambda'} \to \mathbb{Z}_q$.
- A signature scheme $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.
- A MAC scheme $\mathsf{MAC} = (\mathsf{KeyGen}, \mathsf{Tag}, \mathsf{Verify})$, where $\mathsf{MAC.Tag}$ is a PRF.
- A commitment scheme $\mathsf{Com} = (\mathsf{Commit}, \mathsf{Open})$.
- A zero-knowledge proof system $\mathsf{ZK} = (\mathsf{Prove}, \mathsf{Verify})$.

$$\mathcal{R} = \Big\{((k_f, r, y), (k_{\mathsf{pub}}, R, \mathsf{pk}_{\mathsf{enc}}, T)) : \Big(k_{\mathsf{pub}} = \mathsf{Com.Commit}(k_f, r) \wedge T = \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{enc}}, \mathsf{MAC.Tag}(k_f, (\mathsf{id}, m, t_{\mathsf{msg}})))\Big) \vee (R = g^y)\Big\}$$

$\mathsf{KeyGenUser}(1^\lambda)$

$\mathsf{sk} \xleftarrow{R} \mathbb{Z}_q$
$\mathsf{pk} \leftarrow g^{\mathsf{sk}}$
return sk, pk

$\mathsf{KeyGenMod}(1^\lambda)$

$k_{\mathsf{PRF}} \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$
$\mathsf{sk}_{\mathsf{sig}}, \mathsf{pk}_{\mathsf{sig}} \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
$\mathsf{sk}_{\mathsf{dec}}, \mathsf{pk}_{\mathsf{enc}} \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$
$\mathsf{sk} \leftarrow (k_{\mathsf{PRF}}, \mathsf{sk}_{\mathsf{sig}}, \mathsf{sk}_{\mathsf{dec}})$
$\mathsf{pk} \leftarrow (\mathsf{pk}_{\mathsf{sig}}, \mathsf{pk}_{\mathsf{enc}})$
return sk, pk

$\mathsf{IssueCert}(\mathsf{sk}_M, \mathsf{id}, t_{\mathsf{exp}})$

$k_{\mathsf{PRF}}, \mathsf{sk}_{\mathsf{sig}}, \cdot \leftarrow \mathsf{sk}_M$
$k_f \leftarrow \mathsf{MAC.KeyGen}(1^\lambda)$
$r \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, (\mathsf{id}, t_{\mathsf{exp}}))$
$k_{\mathsf{pub}} \leftarrow \mathsf{Com.Commit}(k_f, r)$
$\mathsf{ctx} \leftarrow (\mathsf{id}, k_{\mathsf{pub}}, t_{\mathsf{exp}})$
$\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{sig}}, \mathsf{ctx})$
return $k_f, r, k_{\mathsf{pub}}, \sigma$

$\mathsf{AcceptSession}(\mathsf{pk}_M, \mathsf{ctx}, \sigma)$

$\mathsf{pk}_{\mathsf{sig}}, \cdot \leftarrow \mathsf{pk}_M$
$b \leftarrow \mathsf{Sig.Verify}(\mathsf{pk}_{\mathsf{sig}}, \mathsf{ctx}, \sigma)$
return b

$\mathsf{Frank}(\mathsf{pk}_M, \mathsf{pk}_R, k_f, r, m, t_{\mathsf{msg}}, \mathsf{ctx})$

$\cdot, \mathsf{pk}_{\mathsf{enc}} \leftarrow \mathsf{pk}_M$
$\mathsf{id}, k_{\mathsf{pub}}, \cdot \leftarrow \mathsf{ctx}$
$\alpha \xleftarrow{R} \mathbb{Z}_q$
$R \leftarrow (\mathsf{pk}_R)^\alpha$
$E_R \leftarrow g^\alpha$
$T_{\mathsf{pt}} \leftarrow \mathsf{MAC.Tag}(k_f, (\mathsf{id}, m, t_{\mathsf{msg}}))$
$T \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{enc}}, T_{\mathsf{pt}})$
$\pi \leftarrow \mathsf{ZK.Prove}(\mathcal{R}, (k_f, r, \cdot), (k_{\mathsf{pub}}, \cdot, \mathsf{pk}_{\mathsf{enc}}, T), m, \mathsf{id}, t_{\mathsf{msg}})$
$\mathsf{fd} \leftarrow (R, E_R, T, \pi)$
return fd

$\mathsf{Verify}(\mathsf{pk}_M, \mathsf{sk}_R, m, t_{\mathsf{msg}}, \mathsf{fd}, \mathsf{ctx})$

$\cdot, \mathsf{pk}_{\mathsf{enc}} \leftarrow \mathsf{pk}_M$
$R, E_R, T, \pi \leftarrow \mathsf{fd}$
$\mathsf{id}, k_{\mathsf{pub}}, t_{\mathsf{exp}} \leftarrow \mathsf{ctx}$
$b_0 \leftarrow R = (E_R)^{\mathsf{sk}_R}$
$b_1 \leftarrow t_{\mathsf{msg}} < t_{\mathsf{exp}}$
$b_2 \leftarrow \mathsf{ZK.Verify}(\mathcal{R}, (k_{\mathsf{pub}}, R, \mathsf{pk}_{\mathsf{enc}}, T), m, \mathsf{id}, t_{\mathsf{msg}}, \pi)$
if $\neg(b_0 \wedge b_1 \wedge b_2)$:
  return $\perp$
return T

$\mathsf{Moderate}(\mathsf{sk}_M, k_f, \mathsf{id}, t_{\mathsf{exp}}, m, t_{\mathsf{msg}}, T)$

$\cdot, \cdot, \mathsf{sk}_{\mathsf{dec}} \leftarrow \mathsf{sk}_M$
$b_0 \leftarrow t < t_{\mathsf{exp}}$
$T_{\mathsf{pt}} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_{\mathsf{dec}}, T)$
$b_1 \leftarrow \mathsf{MAC.Verify}(k_f, (\mathsf{id}, m, t_{\mathsf{msg}}), T_{\mathsf{pt}})$
return $b_0 \wedge b_1$

Figure 9: EMF construction.

Instantiations of MAC and commitment schemes suitable for use with our scheme appear in Figure 8. We formalize and prove the following theorem in Appendix D.

Theorem 5.2 (EMF-lite security, informal)**.** *Assuming that DDH is hard and KEA holds in* $\mathbb{G}$*,* PRF *is a secure PRF,* Sig *is an existentially unforgeable signature scheme,* MAC *is an existentially unforgeable MAC where* MAC.Tag *is a PRF,* Com *satisfies hiding and binding, and* ZK *is a sound and zero-knowledge proof system, our EMF-lite scheme described in Construction 5.1 satisfies accountability, unforgeability, and weak deniability.*

5.2. Strengthening Deniability with EMF

Our second scheme builds upon the first, and provides additional deniability. Recall that our (stronger) deniability definitions always give the adversary access to the message sender's secrets. This poses a problem for deniability because an adversary who holds this key and sees the MAC tag T can check whether it is a valid tag or not. If the forger does not also have access to this key, then it cannot produce a convincing forged tag.

We solve this problem by encrypting the tag T, called T_{pt} in this scheme, under a public key belonging to the moderator. Since the moderator can decrypt and recover T_{pt}, this has no impact on the scheme's correctness, but it means that the message receiver never sees the MAC tag. Thus, an adversary who has k_f and r still cannot verify the MAC because the MAC is no longer visible to anyone except the moderator. This upgrades the universal and receiver compromise deniability of our scheme from their weak versions to the stronger version. Moderator compromise deniability, where the forger already has access to k_f, already satisfied the stronger deniability notion.

The only other change needed to the scheme in order to accommodate the additional encryption lies in the zero-knowledge proof that the sender produces. Instead of proving that T is a MAC produced with k_f, it now needs to prove that T is an encryption of a MAC T_{pt}, which is in turn a MAC produced with k_f on the appropriate values. This proof can be computed efficiently if we use the ElGamal encryption scheme [22].

Construction 5.3 (EMF)**.** Our EMF construction appears in Figure 9 and makes use of the following primitives:

- A group $\mathbb{G}$ of prime order q with generator g, where

DDH is hard and the knowledge of exponent assumption holds.

- A PRF $\mathsf{PRF} : \{0,1\}^{\lambda} \times \{0,1\}^{\lambda'} \to \mathbb{Z}_q$.
- A signature scheme $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.
- A public-key encryption scheme $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.
- A MAC scheme $\mathsf{MAC} = (\mathsf{KeyGen}, \mathsf{Tag}, \mathsf{Verify})$.
- A commitment scheme $\mathsf{Com} = (\mathsf{Commit}, \mathsf{Open})$.
- A zero-knowledge proof system $\mathsf{ZK} = (\mathsf{Prove}, \mathsf{Verify})$.

We formalize and prove the following theorem in Appendix E.

Theorem 5.4 (EMF security, informal)**.** *Assuming that DDH is hard and KEA holds in* $\mathbb{G}$*,* PRF *is a secure PRF,* Sig *is an existentially unforgeable signature scheme,* PKE *is CPA secure,* MAC *is an existentially unforgeable MAC,* Com *satisfies hiding and binding, and* ZK *is a sound and zero-knowledge proof system, our EMF scheme described in Construction 5.3 satisfies accountability, unforgeability, and deniability.*

6. Implementation and Evaluation

This section reports on the implementation and evaluation of EMF and EMF-lite. We discuss the implementation details and evaluate our performance results, comparing our schemes' performance to Asymmetric Message Franking (AMF) [43], the most relevant and comparable prior work.

6.1. Implementation

We implemented both EMF and EMF-lite in Rust. We use the Ristretto abstraction offered by the `curve25519_dalek` library [17], [26] for all group operations. We use 32-byte client IDs, 32-byte `u64` timestamps, and 32-byte scalars. All group elements, including commitments in the ZKPoK, are encoded as Ristretto points, which are compressed to 32 bytes for communication.

We use the algebraic MAC-GGM [11], depicted in Figure 8, to instantiate our MAC scheme, the Pedersen multi-commitment in Figure 8 for franking key commitments, and the `ed25519-dalek` [16] implementation of ed25519 [8] as our signature algorithm. We instantiate our zero knowledge proofs using generic composition of Sigma Protocols [9], [15] made non-interactive via the Fiat-Shamir transform [7], [20]. For our hash functions, we use SHA-256 when we only need collision-resistance or when we are modeling hash functions as a random oracle with fixed-length inputs. When we need a random oracle with variable-length inputs, we use SHA-512_256. These choices are made to ensure that our hash functions are indifferentiable from a random oracle when necessary [14], [36]. We use hash function implementations from the `sha2` crate [3].

Our implementation and raw evaluation data are available at https://github.com/jadekeegan/enforceable-message-franking.

	Frank	Verify	Moderate
EMF-lite	$167.9\mu s$	$191.9\mu s$	$11.9\mu s$
EMF	$244.5\mu s$	$250.0\mu s$	$42.4\mu s$
AMF	$257.6\mu s$	$258.4\mu s$	$258.4\mu s$

TABLE 1: Computation time for 100 byte messages, in microseconds, for EMF-lite, EMF, and AMF (prior work). EMF-lite and EMF reduce report verification costs for the moderator by $21.8\times$ and $6.1\times$, respectively, compared to AMF.

6.2. Performance Evaluation

We measure the runtimes for our schemes and prior work using Criterion, a Rust benchmarking suite. We perform our evaluation on an 11th Gen Intel i7 processor @3.6GHz, where we evaluate our schemes using a message size of 100 bytes.

Computation costs. Table 1 shows the computation time for each operation (Frank, Verify, Moderate) that must be run to send, receive, and report each message in our schemes. For both schemes, Frank and Verify times are dominated by their respective zero knowledge proof of knowledge (ZKPoK) proof and verification steps. EMF incurs higher computation costs than EMF-lite, a result of the extra ElGamal encryption and proof thereof. In addition to the costs incurred to send each message, IssueCert takes $82.6\mu s$ and AcceptSession takes $29.9\mu s$, and this overhead is the same for both schemes. These operations must be run each time a new pair of users starts a conversation or when a user's credential is refreshed.

Table 2 measures the performance impact of increasing the message size from 100 bytes to 100 kilobytes. Despite a $1000\times$ increase in message size, computation times for both schemes across most operations only increase 2-3$\times$. All operations require executing a SHA-256 hash function, and frank and verify require executing an additional SHA-512/256 hash. Both hash functions have linear running times dependent on message size, so the time delta when increasing message size is the same for corresponding methods in each scheme. For most operations, the fixed overhead of group operations, which are independent of message size, dominate computation time. This is also why EMF-lite, which incurs lower fixed costs, has a larger relative growth in running time. Still, overall, message size does not have a significant impact on the computation time for our operations.

Communication costs. Table 3 shows the communication overhead of our schemes over the size of the message and the ciphertext containing it. For Frank, the cost is measured as the size of the franking data the sender transmits, which is 384 bytes for the EMF-lite scheme and 480 bytes for the EMF. For Verify and Moderate, communication overhead is measured as the size of the data the respective party receives. For Verify, this is just the size of the franking data

Scheme	Operation	100 B	100 KB	Δ	Ratio
EMF-Lite	Frank	168	359	+191	2.1×
	Verify	192	381	+189	2.0×
	Moderate	12	58	+46	4.9×
EMF	Frank	245	437	+193	1.8×
	Verify	250	441	+191	1.8×
	Moderate	42	90	+48	2.1×

TABLE 2: Computation time comparison for 100 byte and 100 kilobyte messages. All times shown are in microseconds.

	Frank	Verify	Moderate
EMF-lite	384 B	384 B	32 B
EMF	480 B	480 B	64 B
AMF	489 B	489 B	489 B

TABLE 3: Communication overhead, in bytes, for EMF-lite, EMF, and AMF. EMF-lite and EMF reduce report sizes by 15.3× and 7.6×, respectively, compared to AMF.

the receiver gets from the sender per message. In addition to this per-message communication, for each conversation partner after running AcceptSession, users need to store 72 bytes of context information ctx, consisting of a sender ID (32 bytes), a commitment to the franking key $\mathsf{k}_{\mathsf{pub}}$ (32 bytes), and a session expiration t_{exp} (8 bytes). For Moderate, communication overhead per message, beyond the message itself, is the size of the MAC tag (1 group element, 32 bytes) or encrypted MAC tag (2 group elements, 64 bytes) that the moderator gets from the receiver in EMF-lite and EMF, respectively.

Comparison to prior work. We compare our schemes' performance to that of AMF, the most closely related work to ours [43]. While AMF does not offer the same moderation enforcement opportunities as EMF, it is otherwise the closest in functionality, security properties, and flexibility with respect to message delivery protocols. We do not consider Hecate here: Hecate requires the platform to actively participate in the message franking protocol, making it less suitable for direct comparison.

For AMF, we use its `amaze` Rust implementation [37], which is much faster than the original paper's Python implementation. Note that the `amaze` implementation does not compress Ristretto points and thus avoids the additional overhead incurred from point compression and decompression at the cost of passing significantly larger in-memory representations of elliptic curve points between parties. To better reflect real usage scenarios and match our own implementation, we modified the `amaze` implementation to compress the Ristretto points. The forked repository can be found here: https://github.com/jadekeegan/amaze-compressed.

Our schemes are more performant than or on par with AMF across all operations. In particular, our Moderate algorithm is significantly more performant than AMF, with the EMF-lite being $21.8\times$ faster and the EMF being $6.1\times$ faster. The improvement arises from the fact that our Moderate algorithm only needs to verify a MAC, whereas in AMF, Moderate must verify the same proof that is checked in Verify, which is in turn more involved. The savings in proof generation and verification are especially evident in EMF-lite, where Frank and Verify run in 35% and 26% less time than AMF, respectively.

Both our schemes enjoy significantly smaller reporting communication overheads than AMF, a savings of $15.3\times$ and $7.6\times$, respectively, for EMF-lite and EMF. Again, this is because our moderator only receives a single 32- or 64-byte MAC tag, as opposed to a full zero-knowledge proof as in AMF. Since EMF also reduces the size of the ZKPoK, which is the main contributor to high communication costs for franking, we are also able to reduce the communication overhead for Frank and Verify by around 21% in EMF-lite.

7. Conclusion

We have introduced enforceable message franking, a scheme that allows users to opt in to policy enforcement by third-party moderators in a way that is independent of the messaging platform. This allows EMF to be deployed in a wide variety of settings, from messaging platforms that want to delegate moderation responsibilities to third parties, to decentralized messaging platforms where the platform is unable to globally perform moderation duties. Our two schemes, offering slightly different notions of deniability, both outperform AMF, the only prior work that is applicable in such a setting. Our EMF schemes also improve upon prior work by providing a lightweight enforcement mechanism, offering moderators a way to enact their moderation decisions on the platform.

Acknowledgments

We would like to thank Julia Len for thoughtful comments. This material is based upon work supported by the National Science Foundation under Grant No. 2234408. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Hal Abelson, Ross J. Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Jon Callas, Whitfield Diffie, Susan Landau, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller, Bruce Schneier, Vanessa Teague, and Carmela Troncoso. Bugs in our pockets: The risks of client-side scanning. 2021.

[2] Joël Alwen, Xiaohui Ding, Sanjam Garg, and Yiannis Tselekounis. Policy compliant secure messaging. In *ASIACRYPT*, 2025.

[3] Tony Arcieri and Artyom Pavlov. sha2, 2022.

[4] James Bartusek, Sanjam Garg, Abhishek Jain, and Guru-Vamsi Policharla. End-to-end secure messaging with traceability only for illegal content. In *EUROCRYPT*, 2023.

[5] Connor Bell and Saba Eskandarian. Anonymous complaint aggregation for secure messaging. *Proc. Priv. Enhancing Technol.*, 2024.

[6] Mihir Bellare and Viet Tung Hoang. Efficient schemes for committing authenticated encryption. *EUROCRYPT*, 2022.

[7] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS*, 1993.

[8] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In *CHES*, 2011.

[9] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography (version 0.6)*. 2023. https://cryptobook.us.

[10] Pedro Branco, Matthew Green, Aditya Hegde, Abhishek Jain, and Gabriel Kaptchuk. How to trace viral content in end-to-end encrypted messaging. *IACR Cryptol. ePrint Arch.*, page 1052, 2025.

[11] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic macs and keyed-verification anonymous credentials. In *ACM CCS*, pages 1205–1216, 2014.

[12] Melissa Chase, Trevor Perrin, and Greg Zaverucha. The signal private group system and anonymous credentials supporting efficient verifiable encryption. In *CCS*, pages 1445–1459. ACM, 2020.

[13] Daniel Collins, Simone Colombo, and Loïs Huguenin-Dumittan. Real-world deniability in messaging. *Proceedings on Privacy Enhancing Technologies*, 2025.

[14] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-damgård revisited: How to construct a hash function. In *CRYPTO*, 2005.

[15] Ronald Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, January 1997.

[16] Henry de Valence and Isis Agora Lovecruft. ed25519-dalek (version 2.0.0), 2024.

[17] Henry de Valence and Isis Agora Lovecruft. curve25519-dalek (version 4.1.3), 2025.

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

[19] Saba Eskandarian. Abuse reporting for metadata-hiding communication based on secret sharing. *USENIX Security*, 2024.

[20] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, 1986.

[21] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2004.

[22] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, 1984.

[23] Matthew Gregoire, Margaret Pierce, and Saba Eskandarian. Onion franking: Abuse reports for mix-based private messaging. In *NDSS*, 2025.

[24] Matthew Gregoire, Gabriel Schell, and Saba Eskandarian. Third-party moderation of abuse reports for end-to-end encrypted messaging with multiple moderators. *IACR Cryptol. ePrint Arch.*, page 010, 2026.

[25] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. In *CRYPTO*, 2017.

[26] Mike Hamburg. Decaf: Eliminating cofactors through point compression. In *CRYPTO*, 2015.

[27] Zhengan Huang, Junzuo Lai, Gongxian Zeng, and Jian Weng. Mild asymmetric message franking: Illegal-messages-only and retrospective content moderation. In *ASIACRYPT*, Lecture Notes in Computer Science, 2024.

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

[29] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, 1996.

[30] Caroline Kudla and Kenneth G. Paterson. Non-interactive designated verifier proofs and undeniable signatures. In *Cryptography and Coding, 10th IMA International Conference*, 2005.

[31] Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In *SCN*, 2004.

[32] Diane Leblanc-Albarel and Bart Preneel. Black-box collision attacks on widely deployed perceptual hash functions. *IACR Cryptol. ePrint Arch.*, page 1869, 2024.

[33] Julia Len, Esha Ghosh, Paul Grubbs, and Paul Rösler. Interoperability in end-to-end encrypted messaging. *IACR Cryptol. ePrint Arch.*, page 386, 2023.

[34] Helger Lipmaa, Guilin Wang, and Feng Bao. Designated verifier signature schemes: Attacks, new security notions and a new construction. In *ICALP*, 2005.

[35] Linsheng Liu, Daniel S. Roche, Austin Theriault, and Arkady Yerukhimovich. Fighting fake news in encrypted messaging with the fuzzy anonymous complaint tally system (FACTS). In *NDSS*, 2022.

[36] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *TCC*, 2004.

[37] Sanketh Menda and Michael Rosenberg. amaze. https://github.com/sgmenda/amaze, 2022.

[38] Meta. Messenger end-to-end encryption overview, December 2023.

[39] Armin Namavari and Thomas Ristenpart. Transcript franking for encrypted messaging. In *ASIACRYPT*, 2025.

[40] Charlotte Peale, Saba Eskandarian, and Dan Boneh. Secure complaint-enabled source-tracking for encrypted messaging. In *ACM CCS*, 2021.

[41] Carolina Ortega Pérez, Thomas Ristenpart, and Julia Len. Interoperable symmetric message franking. In *ACM CCS*. ACM, 2025.

[42] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, 1997.

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

[44] Nirvan Tyagi, Ian Miers, and Thomas Ristenpart. Traceback for end-to-end encrypted messaging. In *ACM CCS*, 2019.

[45] Zhongming Wang, Tao Xiang, Xiaoguo Li, Biwen Chen, Guomin Yang, Chuan Ma, and Robert H. Deng. Impact tracing: Identifying the culprit of misinformation in encrypted messaging systems. In *NDSS*. The Internet Society, 2025.

[46] Zhongming Wang, Tao Xiang, Xiaoguo Li, Guomin Yang, Biwen Chen, Ze Jiang, Jiacheng Wang, Chuan Ma, and Robert H. Deng. Abuse resistant traceability with minimal trust for encrypted messaging systems. In *NDSS*. The Internet Society, 2026.

Appendix A. Open Science

Our code artifact implements both of our third-party reporting schemes and its benchmarks. The repository can be accessed at https://github.com/jadekeegan/

enforceable-message-franking. This includes the raw data from this paper, as well as instructions for reproducing our graphs and results. To reproduce all benchmarks, run `./scripts/run.sh` in the cloned repository after making `scripts/run.sh` executable. For more detailed instructions, see the README.md in the artifact repository.

We also include a modified Rust implementation of AMF, which correctly compresses values sent between parties instead of sending the much larger in-memory representations of elliptic curve points. The forked repository can be accessed at https://github.com/jadekeegan/amaze-compressed. To reproduce benchmark data, run `cargo bench --bench compressed_franking` in the cloned repository. For more detailed instructions, see the README.md in the artifact repository.

Appendix B. Ethical Considerations

No existing messaging platforms or real-world users were impacted in the course of this research. We believe that this work broadens the space of potential policy options that are available to messaging platforms to enable abuse reporting. Specifically, by allowing for third-party moderators of users' choosing, and enforcement mechanisms that are totally independent of the platform itself, we increase the control and autonomy that users have over what kinds of content they find permissible, and improve the quality of the messaging platform for all users.

Appendix C. Deferred Definitions and Preliminaries

This section includes definitions deferred from the body of the paper, as well as additional preliminaries that formalize some of our assumptions.

Correctness. Correctness for an EMF scheme requires that, for all messages, given an honestly generated, unexpired sender certificate and honestly generated franking data, that message can be successfully read by the receiver and reported to the moderator. Specifically, for all choices of id and t_{exp}, after running

$$\mathsf{sk}_M, \mathsf{pk}_M \leftarrow \mathsf{KeyGenMod}(1^\lambda), \text{and}$$
$$k_f, r, k_{\mathsf{pub}}, \sigma \leftarrow \mathsf{IssueCert}(\mathsf{sk}_M, \mathsf{id}, t_{\mathsf{exp}}),$$

then, defininng ctx as $(\mathsf{id}, k_{\mathsf{pub}}, t_{\mathsf{exp}})$, we must have

$$\mathsf{AcceptSession}(\mathsf{pk}_M, \mathsf{ctx}, \sigma) = 1.$$

Furthermore, for all choices of m and t_{msg}, where $t_{\mathsf{msg}} < t_{\mathsf{exp}}$, if we compute

$$\begin{aligned}\mathsf{sk}_R, \mathsf{pk}_R &\leftarrow \mathsf{KeyGenUser}(1^\lambda),\\ \mathsf{fd} &\leftarrow \mathsf{Frank}(\mathsf{pk}_M, \mathsf{pk}_R, k_f, r, m, t_{\mathsf{msg}}, \mathsf{ctx}), \text{and}\\ T &\leftarrow \mathsf{Verify}(\mathsf{pk}_M, \mathsf{sk}_R, m, t_{\mathsf{msg}}, \mathsf{fd}, \mathsf{ctx}),\end{aligned}$$

$\mathsf{KEA}[\mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{G}, g]$

$b \xleftarrow{R} \mathbb{Z}_q$
$Y, C \leftarrow \mathcal{A}(g, g^b)$
$c \leftarrow \mathcal{E}_{\mathcal{A}}(g, g^b)$
$\mathsf{win} \leftarrow (Y = C^b) \wedge C \neq g^c$
output win

Figure 10: Knowledge of exponents assumption security experiment.

then it must hold that

$$T \neq \perp, \text{and}$$
$$\mathsf{Moderate}(\mathsf{sk}_M, k_f, \mathsf{id}, t_{\mathsf{exp}}, m, t_{\mathsf{msg}}, T) = 1.$$

Knowledge of exponents assumption. The knowledge of exponents assumption (KEA) is relative to a group $\mathbb{G}$ of prime order q and generator g. It asserts that, if an adversary $\mathcal{A}$ produces a Diffie-Hellman triple of the form (g^a, g^b, g^{ab}) after having been given the values (g, g^b), then $\mathcal{A}$ must "have knowledge" of the exponent a. We state this formally below, using similar notation to that used in AMF [27].

Definition C.1 (Knowledge of Exponents Assumption)**.** We define the knowledge of exponents experiment $\mathsf{KEA}[\mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{G}, g]$ in Figure 10 with respect to an efficient adversary $\mathcal{A}$, an extractor $\mathcal{E}_{\mathcal{A}}$ defined relative to $\mathcal{A}$, with access to the same random coins as $\mathcal{A}$, and a prime order group $\mathbb{G}$ with generator g.

We define the *KEA advantage* of $\mathcal{A}$ and $\mathcal{E}_{\mathcal{A}}$ as

$$\mathsf{KEAAdv}[\mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{G}, g] = \mathsf{Pr}\big[\mathsf{KEA}[\mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{G}, g] = 1\big].$$

We say that the KEA assumption holds in $\mathbb{G}$ if, for all efficient adversaries $\mathcal{A}$ with associated extractor $\mathcal{E}_{\mathcal{A}}$,

$$\mathsf{KEAAdv}[\mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{G}, g] \leq \mathsf{negl}(\lambda).$$

Appendix D. Proofs of Security for EMF-lite

Theorem D.1 (EMF-lite Accountability)**.** *Assuming that* Sig *satisfies existential unforgeability, the knowledge of exponents assumption holds in* $\mathbb{G}$*,* ZK *satisfies knowledge soundness, and hardness of discrete log holds in* $\mathbb{G}$*, our enforceable franking scheme* Π_1*, described in Construction 5.1, satisfies accountability (Definition 4.1).*

Specifically, for all efficient adversaries $\mathcal{A}$*, all* $Q_S \in \mathbb{N}$*, and any security parameter* λ*, there exist adversaries* $\mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}$ *such that our scheme* Π_1 *satisfies*

$$\begin{aligned}\mathsf{ACCTAdv}[\mathcal{A}, \Pi_1, Q_S, \lambda] \leq\ & \mathsf{MACAdv}[\mathcal{B}, \lambda] +\\ & (Q_S + 1) \cdot \mathsf{KEAAdv}[\mathcal{C}, \lambda] +\\ & \mathsf{POKAdv}[\mathcal{D}, \lambda] + \mathsf{DLAdv}[\mathcal{E}, \lambda].\end{aligned}$$

Proof. We proceed with a series of hybrids.

- **H0**: This hybrid is the experiment $\mathsf{ACCT}[\mathcal{A}, \Pi_1, Q_S, \lambda]$.
- **H1**: This hybrid is identical to the previous one, except we add an abort condition. The experiment aborts and outputs 0 if a call to AcceptSession ever outputs 1 on some input $\mathsf{ctx}^* \neq \mathsf{ctx}$. This hybrid is indistinguishable from the previous one by the unforgeability of the signature scheme and Lemma D.2.
- **H2**: This hybrid is identical to the previous one, except for each of the $Q_S + 1$ calls to Verify, the experiment runs the KEA extractor $\mathcal{E}_{\mathcal{A}}$ on input (g, pk_R) to return α'. The experiment aborts and outputs 0 if, in any call to Verify, b_0 is set to 1, but $E_R \neq g^{\alpha'}$. This hybrid is indistinguishable from the previous one by the knowledge of exponents assumption and Lemma D.3.
- **H3**: This hybrid is identical to the previous one, except in the last call to Verify, if b_2 is set to 1, we run the witness extractor on π to return (k_f, r, y). The experiment aborts and outputs zero if the relation $\mathcal{R}$ does not hold with respect to the witness. This hybrid is indistinguishable from the previous one by knowledge soundness of ZK and Lemma D.4.
- **H4**: This hybrid is identical to the previous one, except in each call to Verify, we instead compute b_0 by checking $R = (\mathsf{pk}_R)^{\alpha'}$, where α' is the result of the KEA extractor above. This hybrid is indistinguishable from the previous one, as it changes nothing in the adversary's view.
- **H5**: This hybrid is identical to the previous one, except we add an abort condition. The experiment aborts and outputs 0 if Verify succeeds but, for the witness to the proof π extracted from last call to Verify in **H3**, the first clause is false and $R = g^y$ for the y provided in the witness. This hybrid is indistinguishable from the previous one by the hardness of discrete log in $\mathbb{G}$ and Lemma D.5.

Now note that the experiment in **H5** never outputs 1. If so, the adversary has supplied an $(m^*, t^*_{\mathsf{msg}}, \mathsf{fd}^*, \mathsf{ctx}^*, \sigma^*)$ such that AcceptSession and Verify pass, but Moderate fails. By the abort condition added in **H1**, we must have $\mathsf{ctx}^* = \mathsf{ctx}$, so the adversary is using the same $(\mathsf{id}, k_{\mathsf{pub}}, t_{\mathsf{exp}})$ that the experiment is in its final calls to Verify and Moderate. Then, by the abort condition added in **H5**, the adversary must have produced a T^* that is a valid MAC tag under the key k_f used by the experiment. Therefore Moderate must also succeed.

Since $\mathcal{A}$ has no advantage, the theorem follows from the indistinguishability of each successive pair of hybrids and the triangle inequality.

□

Lemma D.2. *Suppose that the advantage of any signature adversary $\mathcal{B}$ in breaking the existential unforgeability of* Sig *is at most* $\mathsf{SigAdv}[\mathcal{B}, \lambda]$. *Then for all efficient adversaries $\mathcal{A}$ that make at most Q_S calls to the $\mathcal{O}_{\mathsf{SendMsg}}$ oracle,*

$$\left|\mathsf{Pr}[\boldsymbol{H1}(\mathcal{A})] - \mathsf{Pr}[\boldsymbol{H0}(\mathcal{A})]\right| \leq \mathsf{SigAdv}[\mathcal{B}, \lambda].$$

Proof. We will show that an adversary who can distinguish between the two hybrids can break the existential unforgeability of Sig with the same advantage. Suppose that there exists such an adversary $\mathcal{A}$. Then define an adversary $\mathcal{B}$ for the signature unforgeability game as follows. $\mathcal{B}$ simulates the hybrid **H1** to $\mathcal{A}$ exactly, except for one change: it replaces Sig.Sign calls with queries to its challenger in the signature unforgeability experiment. If at any point, the abort condition added in **H1** is triggered on input $(\mathsf{ctx}^*, \sigma^*)$, $\mathcal{B}$ submits these values as its signature forgery.

We now argue that $\mathcal{B}$ breaks the existential unforgeability of the signature scheme with the same probability that $\mathcal{A}$ triggers the abort condition. Since this condition is the only difference between **H0** and **H1**, this completes the lemma.

If $\mathcal{A}$ triggers the abort condition in **H1**, then it produced values ctx^* and σ^* such that (a) $\mathsf{Sig.Verify}(\mathsf{pk}_M, \mathsf{ctx}^*, \sigma^*) = 1$, and (b) $\mathsf{ctx}^* \neq \mathsf{ctx}$. The first condition follows from the fact that
AcceptSession succeeded, and the second by construction of the abort condition. But this means that $(\mathsf{ctx}^*, \sigma^*)$ must be an existential forgery of the signature scheme, because the only value for which a valid signature is computed in the ACCT game is ctx. □

Lemma D.3. *Suppose that the advantage of any KEA adversary $\mathcal{B}$ in breaking the knowledge of exponents assumption in $\mathbb{G}$ is at most* $\mathsf{KEAAdv}[\mathcal{B}, \lambda]$. *Then for all efficient adversaries $\mathcal{A}$ that make at most Q_S calls to the $\mathcal{O}_{\mathsf{SendMsg}}$ oracle,*

$$\left|\mathsf{Pr}[\boldsymbol{H2}(\mathcal{A})] - \mathsf{Pr}[\boldsymbol{H1}(\mathcal{A})]\right| \leq Q_S \cdot \mathsf{KEAAdv}[\mathcal{B}, \lambda].$$

Proof. Define $Q_S + 1$ sub-hybrid distributions $\mathbf{H2}_0, \ldots \mathbf{H2}_{Q_S}$, where in the ith hybrid, the experiment computes α' by running the KEA extractor $\mathcal{E}_{\mathcal{A}}$ on input (g, pk_R) in the first i calls to Verify, aborting if $E_R \neq g^{\alpha'}$, and does nothing different for all remaining calls. Let $\mathbf{H2}_0 = \mathbf{H1}$. We will show that an adversary that can distinguish between the two sub-hybrids can break the KEA assumption in $\mathbb{G}$ with the same advantage. Suppose that there exists such an adversary $\mathcal{A}$. Then define an adversary $\mathcal{B}$ for the KEA experiment as follows. $\mathcal{B}$ begins by sampling $i^* \xleftarrow{\mathrm{R}} \{1, \ldots, Q_S\}$. Then it simulates the **H1** experiment to $\mathcal{A}$ exactly, except that for the first i^* calls to Verify, it computes $\alpha' \leftarrow \mathcal{E}_{\mathcal{A}}(g, \mathsf{pk}_R)$ and aborts if $E_R \neq g^{\alpha'}$. It then submits the (E_R, R) pair from the i^*th call to Verify as its KEA response.

Note that for each hybrid $\mathbf{H2}_i$, conditioned on selecting $i^* = i$, $\mathcal{B}$ provides $\mathcal{A}$ with a perfect simulation of $\mathbf{H2}_i$, and the abort condition is triggered precisely when $\mathcal{E}_{\mathcal{A}}$ fails, which is the win condition for the KEA game. Since the abort condition is the only difference between the two sub-hybrids, $\mathcal{B}$ wins the KEA game with the same advantage that $\mathcal{A}$ distinguishes between hybrids $\mathbf{H2}_{i-1}$ and $\mathbf{H2}_i$.

Let $\mathsf{KEA}[\mathcal{B}]$ be the KEA experiment in $\mathbb{G}$ with security parameter λ. By the reasoning above, we have that:

$$\mathsf{Pr}[\mathsf{KEA}[\mathcal{B}] = 1 | i^* = 1] = \left|\mathbf{H2}_{i-1}(\mathcal{A}) - \mathbf{H2}_i(\mathcal{A})\right|.$$

Using laws of probability, we have that

$$\begin{aligned}
&\mathsf{Pr}[\mathsf{KEA}[\mathcal{B},1]=1] \\
&= \sum_{i=1}^{Q_S} \mathsf{Pr}[\mathsf{KEA}[\mathcal{B}]=1 \wedge (i^*=i)] \\
&= \sum_{i=1}^{Q_S} \mathsf{Pr}[i^*=i] \cdot \mathsf{Pr}[\mathsf{KEA}[\mathcal{B}]=1 | (i^*=i)]
\end{aligned}$$

Therefore, since $\mathsf{Pr}[i^*=i] = 1/Q_S$, we continue

$$\begin{aligned}
&= 1/Q_S \left| \sum_{i=1}^{Q_S} \mathbf{H2}_{i-1}(\mathcal{A}) - \mathbf{H2}_i(\mathcal{A}) \right| \\
&= 1/Q_S \left| \mathbf{H2}_0(\mathcal{A}) - \mathbf{H2}_{Q_S}(\mathcal{A}) \right| \\
&= 1/Q_S \left| \mathbf{H1}(\mathcal{A}) - \mathbf{H2}(\mathcal{A}) \right|
\end{aligned}$$

Since we know by assumption that

$$\mathsf{Pr}[\mathsf{KEA}[\mathcal{B}]=1=1] \leq \mathsf{KEAAdv}[\mathcal{B},\lambda],$$

We have that

$$1/Q_S \left| \mathbf{H1}(\mathcal{A}) - \mathbf{H2}(\mathcal{A}) \right| \leq \mathsf{KEAAdv}[\mathcal{B},\lambda],$$

from which the lemma follows. □

Lemma D.4. *Suppose that the advantage of any adversary $\mathcal{B}$ in breaking knowledge soundness of* ZK *is at most* $\mathsf{POKAdv}[\mathcal{B},\lambda]$. *Then for all efficient adversaries $\mathcal{A}$,*

$$\left| \mathsf{Pr}[\boldsymbol{H3}(\mathcal{A})] - \mathsf{Pr}[\boldsymbol{H2}(\mathcal{A})] \right| \leq \mathsf{POKAdv}[\mathcal{B},\lambda].$$

Proof. We will show that an adversary that can distinguish between the two hybrids can break the knowledge soundness of ZK with the same advantage. Suppose that there exists such an adversary $\mathcal{A}$. Then define an adversary $\mathcal{B}$ for the knowledge soundness game as follows. $\mathcal{B}$ simulates the hybrid **H3** to $\mathcal{A}$ exactly, and in the last call to Verify, it submits the proof π and extracted witness (k_f, r, y) for the relation $\mathcal{R}$ as its knowledge soundness counterexample.

Note that $\mathcal{B}$ breaks knowledge soundness of ZK with the same probability that $\mathcal{A}$ triggers the abort condition. This is because by construction, we abort precisely when the relation $\mathcal{R}$ does not hold with respect to the extracted witness. Since this condition is the only difference between **H2** and **H3**, this completes the lemma. □

Lemma D.5. *Suppose that the advantage of any discrete log adversary $\mathcal{B}$ in $\mathbb{G}$ is at most* $\mathsf{DLAdv}[\mathcal{B},\lambda]$. *Then for all efficient adversaries $\mathcal{A}$,*

$$\left| \mathsf{Pr}[\boldsymbol{H5}(\mathcal{A})] - \mathsf{Pr}[\boldsymbol{H4}(\mathcal{A})] \right| \leq \mathsf{DLAdv}[\mathcal{B},\lambda].$$

Proof. We will show that an adversary that can distinguish between the two hybrids can break hardness of discrete log in $\mathbb{G}$ with the same advantage. Suppose that there exists such an adversary $\mathcal{A}$. Then define an adversary $\mathcal{B}$ for the discrete log game as follows. $\mathcal{B}$ simulates the hybrid **H5** to $\mathcal{A}$ exactly, except for one change: it replaces pk_R with its challenge from the discrete log adversary, and returns $\mathsf{sk}_R = y/\alpha'$, for witness y extracted in **H3** and exponent α' extracted in **H2**. Note that the experiment in **H5** never needs the value sk_R, as the only place it's used in the ACCT game was removed in hybrid **H4**.

We now argue that $\mathcal{B}$ wins the discrete log game with the same probability that $\mathcal{A}$ triggers the abort condition. Since this condition is the only difference between the two hybrids, this completes the lemma.

If $\mathcal{A}$ triggers the abort condition, then it has provided a valid witness to the relation $\mathcal{R}$, where the first clause of the disjunction is false and $R = g^y$ for the extracted witness y. Because Verify succeeded (by assumption), we also know that $R = \mathsf{pk}_R^{\alpha'}$, by the b_0 check added in **H4**. Rearranging, we see that $\mathsf{pk}_R = g^{y/\alpha'}$, so returning $\mathsf{sk}_R = y/\alpha'$ wins the discrete log game for $\mathcal{B}$. □

Theorem D.6 (EMF-lite Unforgeability)**.** *Assuming that* Com *satisfies perfect hiding,* ZK *satisfies zero-knowledge, DDH is hard in $\mathbb{G}$, and* MAC *is an existentially unforgeable MAC scheme, our enforceable franking scheme Π_1, described in Construction 5.1, satisfies unforgeability (Definition 4.2).*

Specifically, for any adversary $\mathcal{A}$ attacking the unforgeability of our scheme Π_1, and for any security parameter λ, there exist adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$, such that

$$\begin{aligned}
\mathsf{FORGAdv}[\mathcal{A},\Pi_1,Q_S,Q_F,\lambda] \leq\ & Q_S \cdot \mathsf{ZKAdv}[\mathcal{B},\lambda] \\
&+ \mathsf{PRFAdv}[\mathcal{C},\lambda] + \mathsf{MACAdv}[\mathcal{D},\lambda].
\end{aligned}$$

Proof. We proceed with a short series of hybrids.

- **H0**: This hybrid is the experiment $\mathsf{FORG}[\mathcal{A},\Pi_1,Q_S,Q_F,\lambda]$.
- **H1**: This hybrid is identical to the preceeding one, except that in each call to $\mathcal{O}_{\mathsf{SendMsg}}$, the experiment computes π as a simulated proof. This hybrid is indistinguishable from the preceeding one by the zero-knowledge of ZK and Lemma D.7.
- **H2**: This hybrid is indentical to the preceeding one, except the experiment replaces the value r selected in the call to IssueCert with a random element of $\mathbb{Z}_q$. This hybrid is indistinguishable from the preceeding one by the pseudorandomness of PRF and Lemma D.8.
- **H3**: This hybrid is identical to the preceeding one, except the experiment replaces k_{pub} with a random group element. This is indistinguishable from the previous hybrid by the perfect hiding of our commitment scheme.
- **H4**: This hybrid is identical to the preceeding one, except that in each call to $\mathcal{O}_{\mathsf{ForgeMsg}}$, the experiment aborts and outputs 0 if the MAC.Verify call within Moderate succeeds on $(\mathsf{id}, m^*, t^*_{\mathsf{msg}})$ for some $(m^*, t^*_{\mathsf{msg}}) \notin T_{\mathsf{seen}}$. This hybrid is indistinguishable from the previous one by the existential unforgeability of MAC and Lemma D.9.

Now note that the experiment in **H4** never outputs 1: if the line setting win $\leftarrow$ 1 is reached, this means that

the adversary has produced a triple $(m^*, t^*_{\mathsf{msg}}, T^*)$ such that $\mathsf{MAC.Verify}(k_f, (\mathsf{id}, m^*, t^*_{\mathsf{msg}}), T^*) = 1$, and $(m^*, t^*_{\mathsf{msg}}) \notin T_{\mathsf{seen}}$. But this contradicts the abort condition added in **H4**. Since $\mathcal{A}$ has no advantage, the theorem follows from the indistinguishability of each successive pair of hybrids and the triangle inequality. □

Lemma D.7. *Suppose that the advantage of any adversary $\mathcal{B}$ in breaking the zero-knowledge of the proof system* ZK *is at most* $\mathsf{ZKAdv}[\mathcal{B}, \lambda]$. *Then for all efficient adversaries $\mathcal{A}$ that make at most Q_S calls to the $\mathcal{O}_{\mathsf{SendMsg}}$ oracle,*

$$\left|\mathsf{Pr}[\boldsymbol{H1}(\mathcal{A})] - \mathsf{Pr}[\boldsymbol{H0}(\mathcal{A})]\right| \leq Q_S \cdot \mathsf{ZKAdv}[\mathcal{B}, \lambda].$$

Proof. Define $Q_S + 1$ sub-hybrid distributions $\mathbf{H1}_0, \ldots \mathbf{H1}_{Q_S}$, where in the ith hybrid, the experiment simulates the proof π in the first i calls to Frank, and produces real proofs for all remaining calls. Let $\mathbf{H1}_0 = \mathbf{H0}$. We will show that an adversary that can distinguish between the two sub-hybrids can distinguish between a real and simulated proof from ZK with the same advantage. Suppose that there exists such an adversary $\mathcal{A}$. Then define an adversary $\mathcal{B}$ for the zero-knowledge security experiment as follows. $\mathcal{B}$ begins by sampling $i^* \xleftarrow{R} \{1, \ldots, Q_S\}$. Then it simulates the **H1** experiment to $\mathcal{A}$ exactly, except that for the first i^* calls to Frank, π is computed using the simulation algorithm Sim for ZK.

Note that for each hybrid $\mathbf{H1}_i$, conditioned on selecting $i^* = i$, $\mathcal{B}$ provides $\mathcal{A}$ with a perfect simulation of $\mathbf{H1}_{i-1}$ if its challenger supplied a real proof, and provides a perfect simulation of $\mathbf{H1}_i$ if its challenger supplied a simulated proof. Therefore $\mathcal{B}$ distinguishes between a real and simulated proof with the same advantage that $\mathcal{A}$ distinguishes between hybrids $\mathbf{H1}_{i-1}$ and $\mathbf{H1}_i$.

Let $\mathsf{ZK}[\mathcal{B}, \mathsf{Real}]$ and $\mathsf{ZK}[\mathcal{B}, \mathsf{Sim}]$ be the real and simulated zero-knowledge experiments, respectively, with security parameter λ. By the reasoning above, we have that:

$$\begin{aligned}\mathsf{Pr}[\mathsf{ZK}[\mathcal{B}, \mathsf{Real}] = 1 | i^* = 1] &= \mathbf{H1}_{i-1}(\mathcal{A}), \text{and} \\ \mathsf{Pr}[\mathsf{ZK}[\mathcal{B}, \mathsf{Sim}] = 1 | i^* = 1] &= \mathbf{H1}_i(\mathcal{A}).\end{aligned}$$

Using laws of probability, we have that

$$\begin{aligned}&|\mathsf{Pr}[\mathsf{ZK}[\mathcal{B}, \mathsf{Real}] = 1] - \mathsf{Pr}[\mathsf{ZK}[\mathcal{B}, \mathsf{Sim}] = 1]| \\ &= \left|\sum_{i=1}^{Q_S} \mathsf{Pr}[\mathsf{ZK}[\mathcal{B}, \mathsf{Real}] = 1 \wedge (i^* = i)]\right. \\ &\quad \left. - \mathsf{Pr}[\mathsf{ZK}[\mathcal{B}, \mathsf{Sim}] = 1 \wedge (i^* = i)]\right| \\ &= \left|\sum_{i=1}^{Q_S} \mathsf{Pr}[i^* = i] \cdot \big(\mathsf{Pr}[\mathsf{ZK}[\mathcal{B}, \mathsf{Real}] = 1 | (i^* = i)]\right. \\ &\quad \left. - \mathsf{Pr}[\mathsf{ZK}[\mathcal{B}, \mathsf{Sim}] = 1 | (i^* = i)]\big)\right|.\end{aligned}$$

Therefore, since $\mathsf{Pr}[i^* = i] = 1/Q_S$, we continue

$$\begin{aligned}&= 1/Q_S \left|\sum_{i=1}^{Q_S} \mathbf{H1}_{i-1}(\mathcal{A}) - \mathbf{H1}_i(\mathcal{A})\right| \\ &= 1/Q_S \left|\mathbf{H1}_0(\mathcal{A}) - \mathbf{H1}_{Q_S}(\mathcal{A})\right| \\ &= 1/Q_S \left|\mathbf{H0}(\mathcal{A}) - \mathbf{H1}(\mathcal{A})\right|\end{aligned}$$

Since we know by assumption that

$$|\mathsf{Pr}[\mathsf{ZK}[\mathcal{B}, \mathsf{Real}] = 1] - \mathsf{Pr}[\mathsf{ZK}[\mathcal{B}, \mathsf{Sim}] = 1]| \leq \mathsf{ZKAdv}[\mathcal{B}, \lambda],$$

We have that

$$1/Q_S \left|\mathbf{H0}(\mathcal{A}) - \mathbf{H1}(\mathcal{A})\right| \leq \mathsf{ZKAdv}[\mathcal{B}, \lambda],$$

from which the lemma follows. □

Lemma D.8. *Suppose that the advantage of any adversary $\mathcal{B}$ in breaking the pseudorandomness of* PRF *is at most* $\mathsf{PRFAdv}[\mathcal{B}, \lambda]$. *Then for all efficient adversaries $\mathcal{A}$ against the* FORG *experiment,*

$$\left|\mathsf{Pr}[\boldsymbol{H2}(\mathcal{A}) - \boldsymbol{H1}(\mathcal{A})]\right| \leq \mathsf{PRFAdv}[\mathcal{B}, \lambda].$$

Proof. We will show that an adversary that can distinguish between the two hybrids can break the pseudorandomness of the PRF scheme with the same advantage. Suppose that there exists such an adversary $\mathcal{A}$. Then define an adversary $\mathcal{B}$ for the PRF pseudorandomness game as follows. $\mathcal{B}$ simulates the hybrid **H2** to $\mathcal{A}$ exactly, except it queries the PRF game to get the value r. $\mathcal{B}$ then outputs whatever $\mathcal{A}$ outputs. Note that, if $\mathcal{B}$ receives evaluations of a random function, $\mathcal{A}$ sees a perfect simulation of **H2**, and if it receives evaluations of the PRF, $\mathcal{A}$ sees a perfect simulation of **H1**. Since $\mathcal{B}$ distinguishes the PRF from a random function with probability bounded above by the probability that $\mathcal{A}$ distinguishes the two hybrids, this completes the lemma. □

Lemma D.9. *Suppose that the advantage of any adversary $\mathcal{B}$ in breaking the existential unforgeability of* MAC *is at most* $\mathsf{MACAdv}[\mathcal{B}, \lambda]$. *Then for all efficient adversaries $\mathcal{A}$ that make at most Q_F calls to the $\mathcal{O}_{\mathsf{ForgeMsg}}$ oracle,*

$$\left|\mathsf{Pr}[\boldsymbol{H4}(\mathcal{A})] - \mathsf{Pr}[\boldsymbol{H3}(\mathcal{A})]\right| \leq \mathsf{MACAdv}[\mathcal{B}, \lambda].$$

Proof. We will show that an adversary that can distinguish between the two hybrids can break the unforgeability of the MAC scheme with the same advantage. Suppose that there exists such an adversary $\mathcal{A}$. Then define an adversary $\mathcal{B}$ for the MAC unforgeability game as follows. $\mathcal{B}$ simulates the hybrid **H4** to $\mathcal{A}$ exactly, except that it replaces calls to MAC.Tag and MAC.Verify with corresponding queries to its challenger in the MAC unforgeability game. At any point, if the abort condition added in **H4** is met in some $\mathcal{O}_{\mathsf{ForgeMsg}}$ query with input $(m^*, t^*_{\mathsf{msg}}, T^*)$, $\mathcal{B}$ submits $(\mathsf{id}, m^*, t^*_{\mathsf{msg}}), T^*$ as its MAC forgery.

We now argue that $\mathcal{B}$ breaks the existential unforgeability of the MAC with the same probability that $\mathcal{A}$ triggers the abort condition. Since the condition is the only difference between **H3** and **H4**, this completes the lemma.

If $\mathcal{A}$ triggers the abort condition in **H4**, then it produced values $(\mathsf{id}, m^*, t^*_{\mathsf{msg}})$ (recall that id is chosen by the adversary) and T^* such that $\mathsf{MAC.Verify}(k_f, (\mathsf{id}, m^*, t^*_{\mathsf{msg}}), T^*) = 1$, and $(m^*, t^*_{\mathsf{msg}}) \notin T_{\mathsf{seen}}$. The first condition follows from the fact that Moderate succeeded, and the second by construction of the abort condition. But this means that $((\mathsf{id}, m^*, t^*_{\mathsf{msg}}), T^*)$ must be an existential forgery for the MAC. If T^* were a tag returned from a $\mathsf{MAC.Tag}$ query to $\mathcal{B}$'s challenger, then there must be some id' such that the challenger was queried on input $(\mathsf{id}', m^*, t^*_{\mathsf{msg}})$, meaning that $(m^*, t^*_{\mathsf{msg}}) \in T_{\mathsf{seen}}$, contradicting the second condition. This is because the only way $\mathcal{B}$ generates such queries is from within $\mathcal{O}_{\mathsf{ForgeMsg}}$. □

Theorem D.10 (EMF-lite Weak Deniability)**.** *Assuming that* Sig *satisfies existential unforgeability,* ZK *satisfies zero-knowledge,* MAC.Tag *is a PRF, and DDH is hard in* $\mathbb{G}$*, our enforceable franking scheme* Π_1*, described in Construction 5.1, satisfies weak deniability (Definition 4.5).*

Proof. Define the forgery algorithms as follows. Forgery algorithms for all three deniability games are shown below. Additions necessary for moderator compromise deniability are shown in blue, those necessary for receiver compromise deniability are shown in yellow, and those necessary for both are shown in green.

$\mathsf{Forge}(\mathsf{pk}_M, \mathsf{pk}_R, \mathsf{sk}_R, \mathsf{sk}_M, k_f, m, t_{\mathsf{msg}}, \mathsf{ctx})$

$k_{\mathsf{PRF}}, \cdot \leftarrow \mathsf{sk}_M$
$\mathsf{id}, k_{\mathsf{pub}}, t_{\mathsf{exp}} \leftarrow \mathsf{ctx}$
$\alpha, \beta \xleftarrow{R} \mathbb{Z}_q$
$R \leftarrow g^{\beta}$
$R \leftarrow (\mathsf{pk}_R)^{\alpha}$
$E_R \leftarrow g^{\alpha}$
$T \xleftarrow{R} \mathbb{G}$
$T \leftarrow \mathsf{MAC.Tag}(k_f, (\mathsf{id}, m, t_{\mathsf{msg}}))$
$\pi \leftarrow \mathsf{ZK.Prove}(\mathcal{R}, (\cdot, \cdot, \beta), (\cdot, R, \cdot), \cdot, \cdot, \cdot)$
$\pi \leftarrow \mathsf{ZK.Prove}(\mathcal{R}, (\cdot, \cdot, \mathsf{sk}_R\, \alpha), (\cdot, R, \cdot), \cdot, \cdot, \cdot)$
$r \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, (\mathsf{id}, t_{\mathsf{exp}}))$
$\pi \leftarrow \mathsf{ZK.Prove}(\mathcal{R}, (k_f, r, \cdot), (k_{\mathsf{pub}}, \cdot, T), \mathsf{id}, m, t_{\mathsf{msg}})$
$\mathsf{fd} \leftarrow (R, E_R, T, \pi)$
return fd

For each deniability definition, the proof proceeds as a short series of hybrids. The first two are identical in all cases.

- **H0**: This is the experiment $\{\mathsf{U}, \mathsf{MC}, \mathsf{RC}\}-\mathsf{WDENY}[\mathcal{A}, \Pi_1, \mathsf{Forge}, Q_S, 0, \lambda]$.
- **H1**: This hybrid is identical to the previous one, except adding an abort condition. The experiment aborts and outputs 0 if, in a call to AcceptSession outputs 1 on some input $\mathsf{ctx}^* \neq \mathsf{ctx}$. This hybrid is indistinguishable from the previous one by the unforgeability of the signature scheme and Lemma D.11.

Now note that the only information the adversary receives in all **H1** hybrids is report_b, so it suffices to show that $\{\mathsf{report}_0\}$ is distributed computationally indistinguishably from $\{\mathsf{report}_1\}$.

Universal Deniability. Note that E_R is distributed identically in both experiments, so we need only prove that the values (R, T, π) are distributed indistinguishably in both experiments. We proceed with a short series of hybrids.

- **H0** and **H1**: These are hybrids **H0** and **H1** described above, with respect to the experiment $\mathsf{U} - \mathsf{WDENY}[\mathcal{A}, \Pi_1, \mathsf{Forge}, Q_S, 0, \lambda]$.
- **H2**: This hybrid is identical to the preceeding one, except in each call to Frank, the experiment samples $\beta \xleftarrow{R} \mathbb{Z}_q$ and instead computes $R \leftarrow g^{\beta}$. This is indistinguishable from the previous hybrid by the hardness of DDH in $\mathbb{G}$ and Lemma D.12.
- **H3**: This hybrid is identical to the preceeding one, except in each respective call to Frank, the proof π is replaced with a simulated proof. This is indistinguishable from the previous hybrid by the zero-knowledge of ZK and Lemma D.13.
- **H4**: This hybrid is identical to the preceeding one, except in each call to Frank, the simulated proof π is replaced with one proving that $R = g^{\beta}$ and simulating the first clause. This is indistinguishable from the previous hybrid by the zero-knowledge of ZK and reasoning identical to Lemma D.13.
- **H5**: This hybrid is identical to the preceeding one, except in each call to Frank, the MAC tag T is replaced with a random group element $T \xleftarrow{R} \mathbb{G}$. This is indistinguishable from the previous hybrid by the pseudorandomness of MAC.Tag and Lemma D.14.

Now note that fd in **H5** is distributed identically as in the experiment $\mathsf{U} - \mathsf{WDENY}[\mathcal{A}, \Pi_1, \mathsf{Forge}, Q_S, 1, \lambda]$. By the indistinguishability of adjacent pairs of hybrids and the triangle inequality, Π_1 satisfies weak universal deniability.

Receiver Compromise. Note that R and E_R are distributed identically in both experiments, so we need only prove that the values (T, π) are distributed indistinguishably in both experiments. We proceed with a short series of hybrids.

- **H0** and **H1**: These are hybrids **H0** and **H1** described above, with respect to the experiment $\mathsf{RC} - \mathsf{WDENY}[\mathcal{A}, \Pi_1, \mathsf{Forge}, Q_S, 0, \lambda]$.
- **H2**: This hybrid is identical to the preceeding one, except in each call to Frank, the proof π is replaced with a simulated proof. This is indistinguishable from the previous hybrid by the zero-knowledge of ZK and reasoning identical to Lemma D.13.
- **H3**: This hybrid is identical to the preceeding one, except in each call to Frank, the simulated proof π is replaced with one proving that $R = g^{\alpha \cdot \mathsf{sk}_R}$ and simulating the first clause. This is indistinguishable from the previous hybrid by the zero-knowledge of ZK and reasoning identical to Lemma D.13.
- **H3**: This hybrid is identical to the preceeding one, except in each call to Frank, the MAC tag T is replaced with a random group element $T \xleftarrow{R} \mathbb{G}$. This is

indistinguishable from the previous hybrid by the pseudorandomness of MAC.Tag and reasoning identical to that in Lemma D.14.

Now note that fd in **H3** is distributed identically as in the experiment $\mathsf{RC} - \mathsf{WDENY}[\mathcal{A}, \Pi_1, \mathsf{Forge}, Q_S, 1, \lambda]$. By the indistinguishability of adjacent pairs of hybrids and the triangle inequality, Π_1 satisfies weak receiver-compromise deniability.

Moderator Compromise. Note that in hybrid **H1**, $\{\mathsf{fd}_0\}$ is distributed identically to $\{\mathsf{fd}_1\}$. Therefore Π_1 satisfies moderator-compromise deniability.

Since in all three experiments, $\{\mathsf{fd}_0\}$ is distributed indistinguishably from $\{\mathsf{fd}_1\}$, our basic scheme satisfies weak deniability. □

Lemma D.11. *Suppose that the advantage of any signature adversary $\mathcal{B}$ in breaking the existential unforgeability of* Sig *is at most* $\mathsf{SigAdv}[\mathcal{B}, \lambda]$. *Then for all efficient adversaries $\mathcal{A}$ that make at most Q_S calls to the $\mathcal{O}_{\mathsf{SendMsg}}$ oracle,*

$$\left|\mathsf{Pr}[\boldsymbol{H1}(\mathcal{A})] - \mathsf{Pr}[\boldsymbol{H0}(\mathcal{A})]\right| \leq \mathsf{SigAdv}[\mathcal{B}, \lambda].$$

Proof. The proof of this lemma is identical to that of Lemma D.2. □

Lemma D.12. *Suppose that the advantage of any DDH adversary $\mathcal{B}$ in the DDH experiment for $\mathbb{G}$ is at most* $\mathsf{DDHAdv}[\mathcal{B}, \lambda]$. *Then for all efficient adversaries $\mathcal{A}$ that make at most Q_S calls to the $\mathcal{O}_{\mathsf{SendMsg}}$ oracle,*

$$\left|\mathsf{Pr}[\boldsymbol{H2}(\mathcal{A})] - \mathsf{Pr}[\boldsymbol{H1}(\mathcal{A})]\right| \leq Q_S \cdot \mathsf{DDHAdv}[\mathcal{B}, \lambda].$$

Proof. Define $Q_S + 1$ sub-hybrid distributions $\mathbf{H2}_0, \ldots \mathbf{H2}_{Q_S}$, where in the ith hybrid, the experiment samples $\beta \xleftarrow{R} \mathbb{Z}_q$ and computes R as $R \leftarrow g^\beta$ in the first i calls to Frank, and computes R as $(\mathsf{pk}_R)^\alpha$ for all remaining calls. Let $\mathbf{H2}_0 = \mathbf{H1}$. We will show that an adversary that can distinguish between the two sub-hybrids can distinguish between a Diffie-Hellman triple and three randomly selected group elements with the same advantage. Suppose that there exists such an adversary $\mathcal{A}$. Then define an adversary $\mathcal{B}$ for the DDH experiment as follows. $\mathcal{B}$ begins by sampling $i^* \xleftarrow{R} \{1, \ldots, Q_S\}$. Then it simulates the **H1** experiment to $\mathcal{A}$ exactly, except that for the first i^* calls to Frank, $\mathcal{B}$ samples $\beta \xleftarrow{R} \mathbb{Z}_q$ and computes $R \leftarrow g^\beta$. $\mathcal{B}$ then outputs whatever $\mathcal{A}$ outputs.

Note that for each hybrid $\mathbf{H2}_i$, conditioned on selecting $i^* = i$, $\mathcal{B}$ provides $\mathcal{A}$ with a perfect simulation of $\mathbf{H2}_{i-1}$ if its challenger supplied a Diffie-Hellman triple, and provides a perfect simulation of $\mathbf{H2}_i$ if its challenger supplied three random group elements. Therefore $\mathcal{B}$ distinguishes between a Diffie-Hellman triple and a random triple with the same advantage that $\mathcal{A}$ distinguishes between hybrids $\mathbf{H2}_{i-1}$ and $\mathbf{H2}_i$.

Let $\mathsf{DDH}[\mathcal{B}, 1]$ and $\mathsf{DDH}[\mathcal{B}, 0]$ be, the DDH experiments providing a Diffie-Hellman triple and a random triple, respectively, with security parameter λ. By the reasoning above, we have that:

$$\mathsf{Pr}[\mathsf{DDH}[\mathcal{B}, 1] = 1 | i^* = 1] = \mathbf{H2}_{i-1}(\mathcal{A}), \text{and}$$
$$\mathsf{Pr}[\mathsf{DDH}[\mathcal{B}, 0] = 1 | i^* = 1] = \mathbf{H2}_i(\mathcal{A}).$$

Using laws of probability, we have that

$$\begin{aligned}
&|\mathsf{Pr}[\mathsf{DDH}[\mathcal{B}, 1] = 1] - \mathsf{Pr}[\mathsf{DDH}[\mathcal{B}, 0] = 1]| \\
&= \Bigg| \sum_{i=1}^{Q_S} \mathsf{Pr}[\mathsf{DDH}[\mathcal{B}, 1] = 1 \wedge (i^* = i)] \\
&\quad - \mathsf{Pr}[\mathsf{DDH}[\mathcal{B}, 0] = 1 \wedge (i^* = i)] \Bigg| \\
&= \Bigg| \sum_{i=1}^{Q_S} \mathsf{Pr}[i^* = i] \cdot \big(\mathsf{Pr}[\mathsf{DDH}[\mathcal{B}, 1] = 1 | (i^* = i)] \\
&\quad - \mathsf{Pr}[\mathsf{DDH}[\mathcal{B}, 0] = 1 | (i^* = i)]\big) \Bigg|.
\end{aligned}$$

Therefore, since $\mathsf{Pr}[i^* = i] = 1/Q_S$, we continue

$$\begin{aligned}
&= 1/Q_S \left| \sum_{i=1}^{Q_S} \mathbf{H2}_{i-1}(\mathcal{A}) - \mathbf{H2}_i(\mathcal{A}) \right| \\
&= 1/Q_S \left| \mathbf{H2}_0(\mathcal{A}) - \mathbf{H2}_{Q_S}(\mathcal{A}) \right| \\
&= 1/Q_S \left| \mathbf{H0}(\mathcal{A}) - \mathbf{H2}(\mathcal{A}) \right|
\end{aligned}$$

Since we know by assumption that

$$|\mathsf{Pr}[\mathsf{DDH}[\mathcal{B}, 1] = 1] - \mathsf{Pr}[\mathsf{DDH}[\mathcal{B}, 0] = 1]| \leq \mathsf{DDHAdv}[\mathcal{B}, \lambda],$$

We have that

$$1/Q_S \left| \mathbf{H0}(\mathcal{A}) - \mathbf{H2}(\mathcal{A}) \right| \leq \mathsf{DDHAdv}[\mathcal{B}, \lambda],$$

from which the lemma follows. □

Lemma D.13. *Suppose that the advantage of any adversary $\mathcal{B}$ in breaking the zero-knowledge of the proof system* ZK *is at most* $\mathsf{ZKAdv}[\mathcal{B}, \lambda]$. *Then for all efficient adversaries $\mathcal{A}$ that make at most Q_S calls to the $\mathcal{O}_{\mathsf{SendMsg}}$ oracle,*

$$\left|\mathsf{Pr}[\boldsymbol{H3}(\mathcal{A})] - \mathsf{Pr}[\boldsymbol{H2}(\mathcal{A})]\right| \leq Q_S \cdot \mathsf{ZKAdv}[\mathcal{B}, \lambda].$$

Proof. The proof of this lemma is identical to that of Lemma D.7. □

Lemma D.14. *Suppose that the advantage of any PRF adversary $\mathcal{B}$ in the PRF experiment for* MAC.Tag *is at most* $\mathsf{PRFAdv}[\mathcal{B}, \lambda]$. *Then for all efficient adversaries $\mathcal{A}$ that make at most Q_S calls to the $\mathcal{O}_{\mathsf{SendMsg}}$ oracle,*

$$\left|\mathsf{Pr}[\boldsymbol{H5}(\mathcal{A})] - \mathsf{Pr}[\boldsymbol{H4}(\mathcal{A})]\right| \leq Q_S \cdot \mathsf{PRFAdv}[\mathcal{B}, \lambda].$$

Proof. Define $Q_S + 1$ sub-hybrid distributions $\mathbf{H5}_0, \ldots \mathbf{H5}_{Q_S}$, where in the ith hybrid, the experiment samples $T \xleftarrow{R} \mathbb{G}$ in the first i calls to Frank, and computes T as $\mathsf{MAC.Tag}(k_f, (\mathsf{id}, m, t_{\mathsf{msg}}))$ for all remaining calls. Let $\mathbf{H5}_0 = \mathbf{H4}$. We will show that an adversary that can distinguish between the two sub-hybrids can distinguish between a MAC tag and a selected group elements with the same advantage. Suppose that there exists such an adversary $\mathcal{A}$. Then define an adversary $\mathcal{B}$ for the PRF experiment as

follows. $\mathcal{B}$ begins by sampling $i^* \xleftarrow{R} \{1, \ldots, Q_S\}$. Then it simulates the **H4** experiment to $\mathcal{A}$ exactly, except that for the first i^* calls to Frank, $\mathcal{B}$ samples $T \xleftarrow{R} \mathbb{G}$.

Note that for each hybrid $\mathbf{H5}_i$, conditioned on selecting $i^* = i$, $\mathcal{B}$ provides $\mathcal{A}$ with a perfect simulation of $\mathbf{H5}_{i-1}$ if its challenger supplied a MAC.Tag evaluation, and provides a perfect simulation of $\mathbf{H5}_i$ if its challenger supplied a random group element. Therefore $\mathcal{B}$ distinguishes between a PRF and a random function with the same advantage that $\mathcal{A}$ distinguishes between hybrids $\mathbf{H5}_{i-1}$ and $\mathbf{H5}_i$.

Let $\mathsf{PRF}[\mathcal{B}, 1]$ and $\mathsf{PRF}[\mathcal{B}, 0]$ be the PRF experiments using evaluations of a PRF and evaluations of a random function, respectively, with security parameter λ. By the reasoning above, we have that:

$$\begin{aligned}&\mathsf{Pr}[\mathsf{PRF}[\mathcal{B}, 1] = 1 | i^* = 1] = \mathbf{H5}_{i-1}(\mathcal{A}), \text{and}\\ &\mathsf{Pr}[\mathsf{PRF}[\mathcal{B}, 0] = 1 | i^* = 1] = \mathbf{H5}_i(\mathcal{A}).\end{aligned}$$

Using laws of probability, we have that

$$\begin{aligned}&|\mathsf{Pr}[\mathsf{PRF}[\mathcal{B}, 1] = 1] - \mathsf{Pr}[\mathsf{PRF}[\mathcal{B}, 0] = 1]|\\ &= \left| \sum_{i=1}^{Q_S} \mathsf{Pr}[\mathsf{PRF}[\mathcal{B}, 1] = 1 \wedge (i^* = i)] - \right.\\ &\left. \mathsf{Pr}[\mathsf{PRF}[\mathcal{B}, 0] = 1 \wedge (i^* = i)] \right|\\ &= \left| \sum_{i=1}^{Q_S} \mathsf{Pr}[i^* = i] \cdot \big(\mathsf{Pr}[\mathsf{PRF}[\mathcal{B}, 1] = 1 | (i^* = i)] \right.\\ &\left. - \mathsf{Pr}[\mathsf{PRF}[\mathcal{B}, 0] = 1 | (i^* = i)] \big) \right|.\end{aligned}$$

Therefore, since $\mathsf{Pr}[i^* = i] = 1/Q_S$, we continue

$$\begin{aligned}&= 1/Q_S \left| \sum_{i=1}^{Q_S} \mathbf{H5}_{i-1}(\mathcal{A}) - \mathbf{H5}_i(\mathcal{A}) \right|\\ &= 1/Q_S \, |\mathbf{H5}_0(\mathcal{A}) - \mathbf{H5}_{Q_S}(\mathcal{A})|\\ &= 1/Q_S \, |\mathbf{H0}(\mathcal{A}) - \mathbf{H5}(\mathcal{A})|\end{aligned}$$

Since we know by assumption that

$$|\mathsf{Pr}[\mathsf{PRF}[\mathcal{B}, 1] = 1] - \mathsf{Pr}[\mathsf{PRF}[\mathcal{B}, 0] = 1]| \leq \mathsf{PRFAdv}[\mathcal{B}, \lambda],$$

We have that

$$1/Q_S \, |\mathbf{H0}(\mathcal{A}) - \mathbf{H5}(\mathcal{A})| \leq \mathsf{PRFAdv}[\mathcal{B}, \lambda],$$

from which the lemma follows. □

Appendix E. Proofs of Security for EMF

Theorem E.1 (EMF Accountability)**.** *Assuming that* Sig *satisfies existential unforgeability, the knowledge of exponents assumption holds in* $\mathbb{G}$*,* ZK *satisfies knowledge soundness, and hardness of discrete log holds in* $\mathbb{G}$*, our enforceable franking scheme* Π_2*, described in Construction 5.3, satisfies accountability (Definition 4.1).*

Specifically, for all efficient adversaries $\mathcal{A}$*, all* $Q_S \in \mathbb{N}$*, and any security parameter* λ*, there exist adversaries* $\mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}$ *such that our scheme* Π_2 *satisfies*

$$\begin{aligned}\mathsf{ACCTAdv}[\mathcal{A}, \Pi_2, Q_S, \lambda] \leq\ &\mathsf{MACAdv}[\mathcal{B}, \lambda] +\\ &(Q_S + 1) \cdot \mathsf{KEAAdv}[\mathcal{C}, \lambda] +\\ &\mathsf{POKAdv}[\mathcal{D}, \lambda] + \mathsf{DLAdv}[\mathcal{E}, \lambda].\end{aligned}$$

Proof. The proof of this theorem is identical to the proof of Theorem D.1. □

Theorem E.2 (EMF Unforgeability)**.** *Assuming that* ZK *satisfies zero-knowledge,* Com *satisfies perfect hiding,* PKE *satisfies CPA security, and* MAC *is existentially unforgeable, our enforceable franking scheme* Π_2*, described in Construction 5.3, satisfies unforgeability (Definition 4.2).*

Specifically, for any adversary $\mathcal{A}$ *attacking the unforgeability of our scheme* Π_1*, and for any security parameter* λ*, there exist adversaries* $\mathcal{B}$*,* $\mathcal{C}$*, and* $\mathcal{D}$*, such that*

$$\begin{aligned}\mathsf{FORGAdv}[\mathcal{A}, \Pi_1, Q_S, Q_F, \lambda] \leq\ &Q_S \cdot \mathsf{ZKAdv}[\mathcal{B}, \lambda]\\ &+ \mathsf{PRFAdv}[\mathcal{C}, \lambda] + \mathsf{MACAdv}[\mathcal{D}, \lambda].\end{aligned}$$

Proof. The proof of this theorem is identical to that of Theorem D.6. □

Theorem E.3 (EMF Deniability)**.** *Assuming that* Sig *satisfies existential unforgeability,* ZK *satisfies zero-knowledge,* PKE *satisfies semantic security, and DDH is hard in* $\mathbb{G}$*, our enforceable franking scheme* Π_2*, described in Construction 5.3, satisfies deniability (Definition 4.4).*

Proof. Define the forgery algorithms as follows. Additions necessary for moderator compromise deniability are shown in blue, those necessary for receiver compromise deniability are shown in yellow, and those necessary for both are shown in green.

$\mathsf{Forge}(\mathsf{pk}_M, \mathsf{pk}_R, \mathsf{sk}_R, \mathsf{sk}_M, k_f, m, t_{\mathsf{msg}}, \mathsf{ctx})$

$k_{\mathsf{PRF}}, \cdot, \cdot \leftarrow \mathsf{sk}_M$
$\cdot, \mathsf{pk}_{\mathsf{enc}} \leftarrow \mathsf{pk}_M$
$\mathsf{id}, k_{\mathsf{pub}}, t_{\mathsf{exp}} \leftarrow \mathsf{ctx}$
$\alpha, \beta \xleftarrow{R} \mathbb{Z}_q^2$
$R \leftarrow g^\beta$
$R \leftarrow (\mathsf{pk}_R)^\alpha$
$E_R \leftarrow g^\alpha$
$T_{\mathsf{pt}} \xleftarrow{R} \mathbb{G}$
$T_{\mathsf{pt}} \leftarrow \mathsf{MAC.Tag}(k_f, (\mathsf{id}, m, t_{\mathsf{msg}}))$
$T \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{enc}}, T_{\mathsf{pt}})$
$\pi \leftarrow \mathsf{ZK.Prove}(\mathcal{R}, (\cdot, \cdot, \beta), (\cdot, R, \cdot, \cdot), \cdot, \cdot, \cdot)$
$\pi \leftarrow \mathsf{ZK.Prove}(\mathcal{R}, (\cdot, \cdot, \mathsf{sk}_R\, \alpha), (\cdot, R, \cdot, \cdot), \cdot, \cdot, \cdot)$
$r \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, (\mathsf{id}, t_{\mathsf{exp}}))$
$\pi \leftarrow \mathsf{ZK.Prove}(\mathcal{R}, (k_f, r, \cdot), (k_{\mathsf{pub}}, \cdot, \mathsf{pk}_{\mathsf{enc}}, T), \mathsf{id}, m, t_{\mathsf{msg}})$
$\mathsf{fd} \leftarrow (R, E_R, T, \pi)$
return fd

For each deniability definition, the proof proceeds as a short series of hybrids. The first two are identical in all cases.

- **H0**: This is the experiment $\{\mathsf{U}, \mathsf{MC}, \mathsf{RC}\}-\mathsf{DENY}[\mathcal{A}, \Pi_2, \mathsf{Forge}, Q_S, 0, \lambda]$.
- **H1**: This hybrid is identical to the previous one, except adding an abort condition. The experiment aborts and outputs 0 if, in a call to AcceptSession outputs 1 on some input $\mathsf{ctx}^* \neq \mathsf{ctx}$. This hybrid is indistinguishable from the previous one by the unforgeability of the signature scheme and Lemma E.4.

Now note that the only information the adversary receives in all **H1** hybrids is report_b, so it suffices to show that $\{\mathsf{report}_0\}$ is distributed computationally indistinguishably from $\{\mathsf{report}_1\}$.

Universal Deniability. Note that E_R is distributed identically in both experiments, so we need only prove that the values (R, T, π) are distributed indistinguishably in both experiments. We proceed with a short series of hybrids.

- **H0** and **H1**: These are hybrids **H0** and **H1** described above, with respect to the experiment $\mathsf{U} - \mathsf{DENY}[\mathcal{A}, \Pi_2, \mathsf{Forge}, Q_S, 0, \lambda]$.
- **H2**: This hybrid is identical to the preceeding one, except in each call to Frank, the experiment samples $\beta \xleftarrow{R} \mathbb{Z}_q$ and instead computes $R \leftarrow g^\beta$. This is indistinguishable from the previous hybrid by the hardness of DDH in $\mathbb{G}$ and reasoning identical to Lemma D.12.
- **H3**: This hybrid is identical to the preceeding one, except in each call to Frank, each proof π is replaced with a simulated proof. This is indistinguishable from the previous hybrid by the zero-knowledge of ZK and reasoning identical to Lemma D.13.
- **H4**: This hybrid is identical to the preceeding one, except in each call to Frank, the simulated proof π is replaced with one proving that $R = g^\beta$ and simulating the first clause. This is indistinguishable from the previous hybrid by the zero-knowledge of ZK and reasoning identical to Lemma D.13.
- **H5**: This hybrid is identical to the preceeding one, except in each call to Frank, the ciphertext T is replaced with an encryption of a random group element $T_{\mathsf{pt}} \xleftarrow{R} \mathbb{G}$. This is indistinguishable from the previous hybrid by the semantic security of PKE and Lemma E.5.

Now note that fd in **H5** is distributed identically as in the experiment $\mathsf{U} - \mathsf{DENY}[\mathcal{A}, \Pi_2, \mathsf{Forge}, Q_S, 1, \lambda]$. By the indistinguishability of adjacent pairs of hybrids and the triangle inequality, Π_2 satisfies universal deniability.

Receiver Compromise. Note that R and E_R are distributed identically in both experiments, so we need only prove that the values (T, π) are distributed indistinguishably in both experiments. We proceed with a short series of hybrids.

- **H0** and **H1**: These are hybrids **H0** and **H1** described above, with respect to the experiment $\mathsf{RC} - \mathsf{DENY}[\mathcal{A}, \Pi_2, \mathsf{Forge}, Q_S, 0, \lambda]$.
- **H2**: This hybrid is identical to the preceeding one, except in each call to Frank, the proof π is replaced with a simulated proof. This is indistinguishable from the previous hybrid by the zero-knowledge of ZK and reasoning identical to Lemma D.13.
- **H3**: This hybrid is identical to the preceeding one, except in each call to Frank, the simulated proof π is replaced with one proving that $R = g^{\alpha \cdot \mathsf{sk}_R}$ and simulating the first clause. This is indistinguishable from the previous hybrid by the zero-knowledge of ZK and reasoning identical to Lemma D.13.
- **H3**: This hybrid is identical to the preceeding one, except in each call to Frank, the ciphertext T is replaced with an encryption of a random group element $T_{\mathsf{pt}} \xleftarrow{R} \mathbb{G}$. This is indistinguishable from the previous hybrid by the semantic security of PKE and reasoning identical to that in Lemma E.5.

Now note that fd in **H3** is distributed identically as in the experiment $\mathsf{RC} - \mathsf{DENY}[\mathcal{A}, \Pi_2, \mathsf{Forge}, Q_S, 1, \lambda]$. By the indistinguishability of adjacent pairs of hybrids and the triangle inequality, Π_2 satisfies receiver-compromise deniability.

Moderator Compromise. Note that in hybrid **H1**, $\{\mathsf{fd}_0\}$ is distributed identically to $\{\mathsf{fd}_1\}$. Therefore Π_2 satisfies moderator-compromise deniability.

Since in all three experiments, $\{\mathsf{fd}_0\}$ is distributed indistinguishably from $\{\mathsf{fd}_1\}$, our basic scheme satisfies deniability. □

Lemma E.4. *Suppose that the advantage of any signature adversary $\mathcal{B}$ in breaking the existential unforgeability of* Sig *is at most* $\mathsf{SigAdv}[\mathcal{B}, \lambda]$. *Then for all efficient adversaries $\mathcal{A}$ that make at most Q_S calls to the $\mathcal{O}_{\mathsf{SendMsg}}$ oracle,*

$$\left|\Pr[\textbf{\textit{H1}}(\mathcal{A})] - \Pr[\textbf{\textit{H0}}(\mathcal{A})]\right| \leq \mathsf{SigAdv}[\mathcal{B}, \lambda].$$

Proof. The proof of this lemma is identical to the proof of Lemma D.2. □

Lemma E.5. *Suppose that the advantage of any adversary $\mathcal{B}$ in breaking the CPA security of* PKE *is at most* $\mathsf{CPAAdv}[\mathcal{B}, \lambda]$. *Then for all efficient adversaries $\mathcal{A}$ that make at most Q_S calls to the $\mathcal{O}_{\mathsf{SendMsg}}$ oracle,*

$$\left|\Pr[\textbf{\textit{H5}}(\mathcal{A})] - \Pr[\textbf{\textit{H4}}(\mathcal{A})]\right| \leq \mathsf{CPAAdv}[\mathcal{B}, \lambda].$$

Proof. We will show that an adversary that can distinguish between the two hybrids can break the CPA security of PKE with the same advantage. Suppose that there exists such an adversary $\mathcal{A}$. Then define an adversary $\mathcal{B}$ for the CPA security game as follows. $\mathcal{B}$ simulates the hybrid **H5** to $\mathcal{A}$ exactly, except that for each call to PKE.Enc, $\mathcal{B}$ selects a random group element T'_{pt} and submits $(T_{\mathsf{pt}}, T'_{\mathsf{pt}})$ as a query to its challenger in the CPA security game.

We now argue that $\mathcal{B}$ breaks the CPA security of PKE with the same probability that $\mathcal{A}$ distinguishes these two hybrids. Note that, if the CPA security experiment returns encryptions of T_{pt}, $\mathcal{A}$ sees exactly hybrid **H4**, and if the CPA security experiment returns encryptions of T'_{pt}, then $\mathcal{A}$ sees exactly hybrid **H5**. Since this is the only difference between the two hybrids, this completes the lemma. □