Asymmetric Group Message Franking: Definitions & Constructions

Authors: Junzuo Lai[1] Gongxian Zeng[2] Zhengan Huang[2]
Siu Ming Yiu[3] Xin Mu[2] Jian Weng[1]

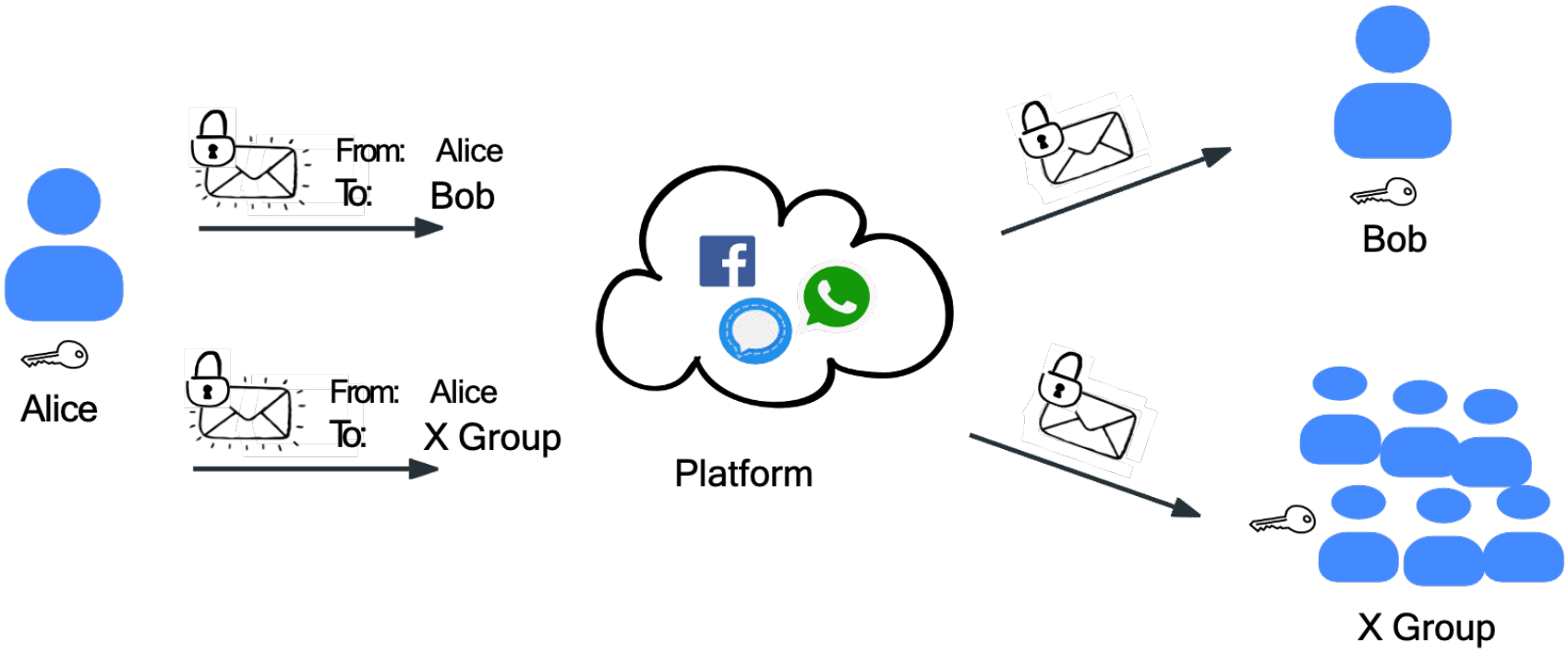
Speaker: Shuai Han

[1]College of Information Science and Technology, Jinan University, Guangzhou, China

[2]Peng Cheng Laboratory, Shenzhen, China

[3]The University of Hong Kong, Hong Kong, China

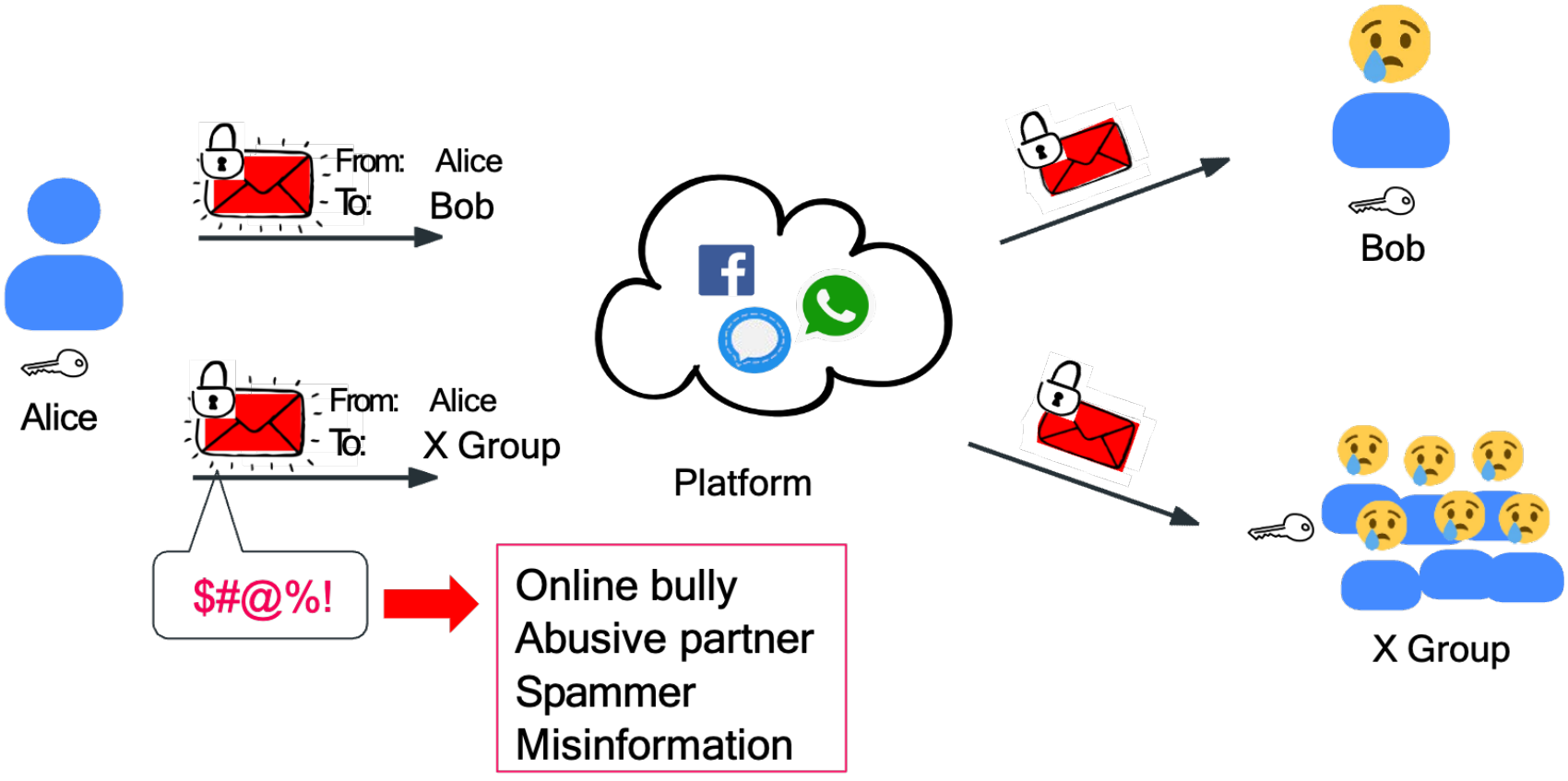
Background: Messaging communication



Security properties:

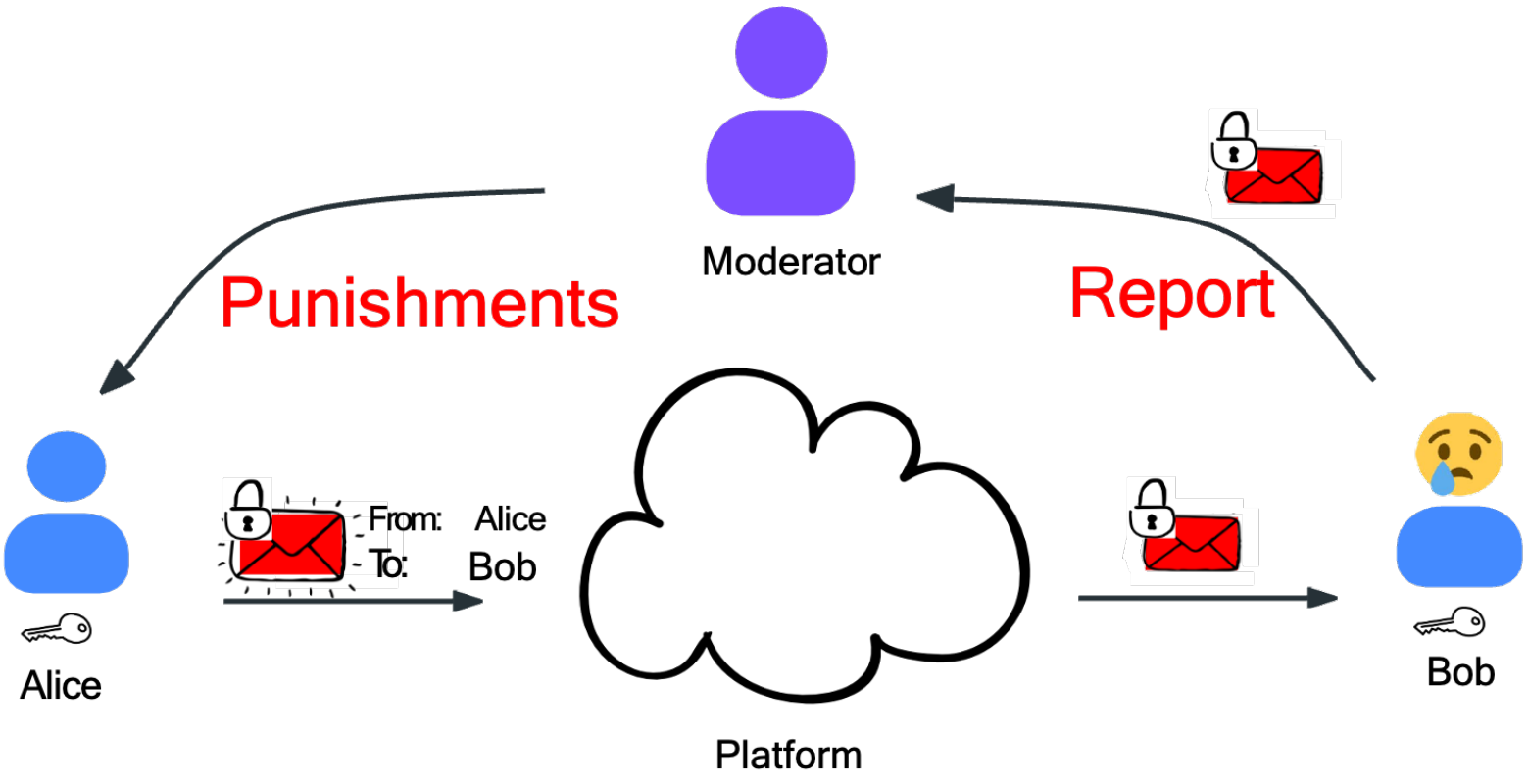
1. Confidentiality
2. Integrity

Background: Abuse of messaging application



However, the messaging applications are abused for the spread of malicious information.

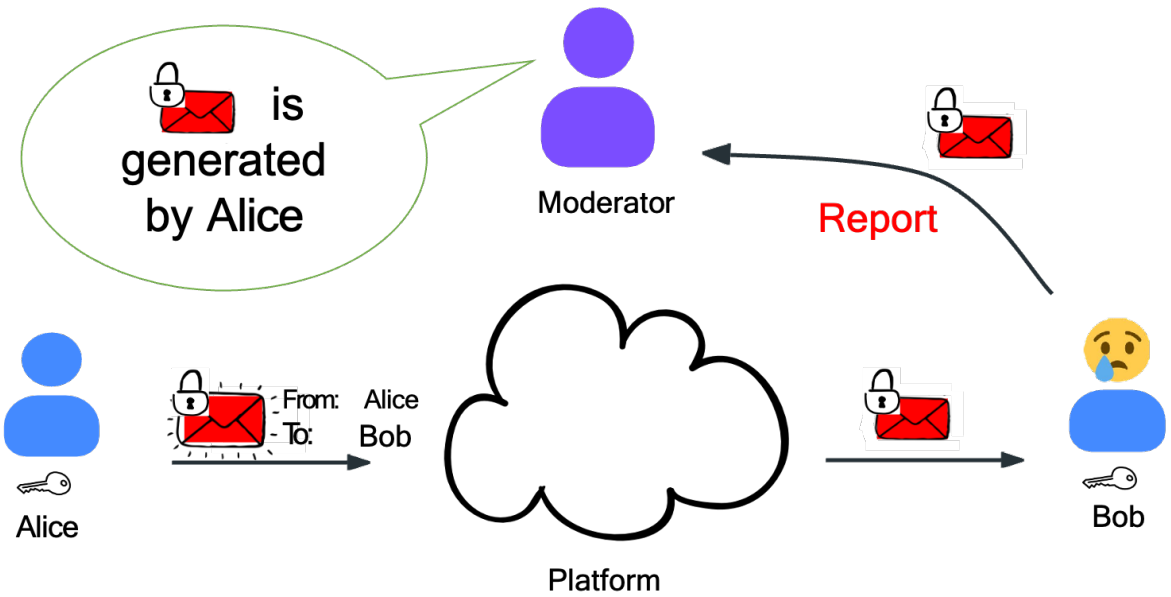
Background: Message franking (MF) [Fac16, GLR17, TGL+19]



Background: Security properties of MF – Accountability I

Here, we recall the security notions defined in [TGL+19].

Accountability: it allows the receiver to report the malicious messages to some moderator (e.g., the platform or some trusted third party), and meanwhile guarantees that no fake reports can be fabricated to frame an honest sender.



Background: Security properties of MF – Accountability II

In [TGL+19], accountability is formalized with two special properties: *sender binding and receiver binding*.

1. **Sender binding** (s-bind) guarantees that any sender should not be able to trick receivers into accepting unreportable messages.
2. **Receiver binding** (r-bind) guarantees that any receivers cannot deceive the judge (to frame the innocent sender).

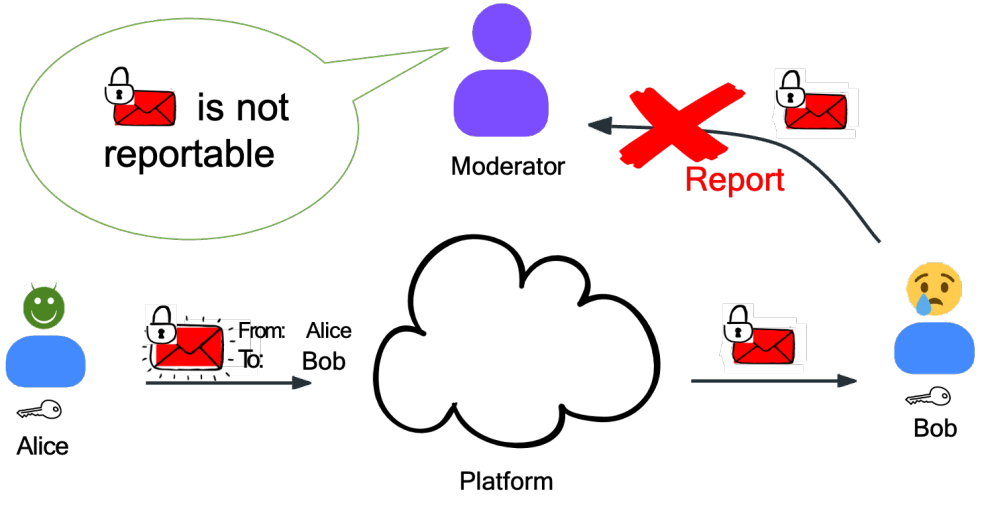


Figure 1: Attack on s-bind

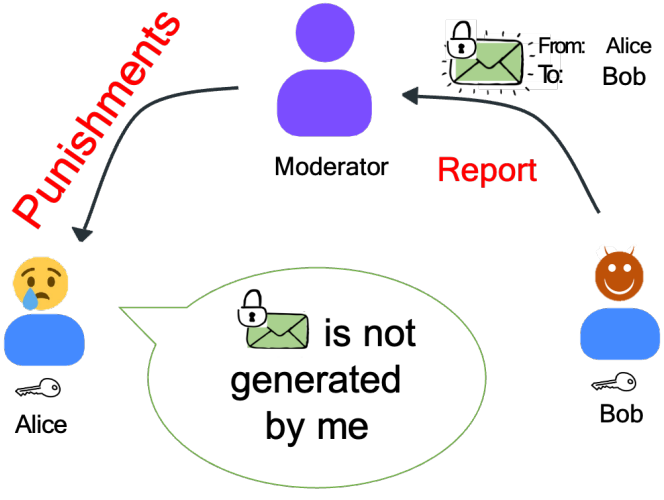
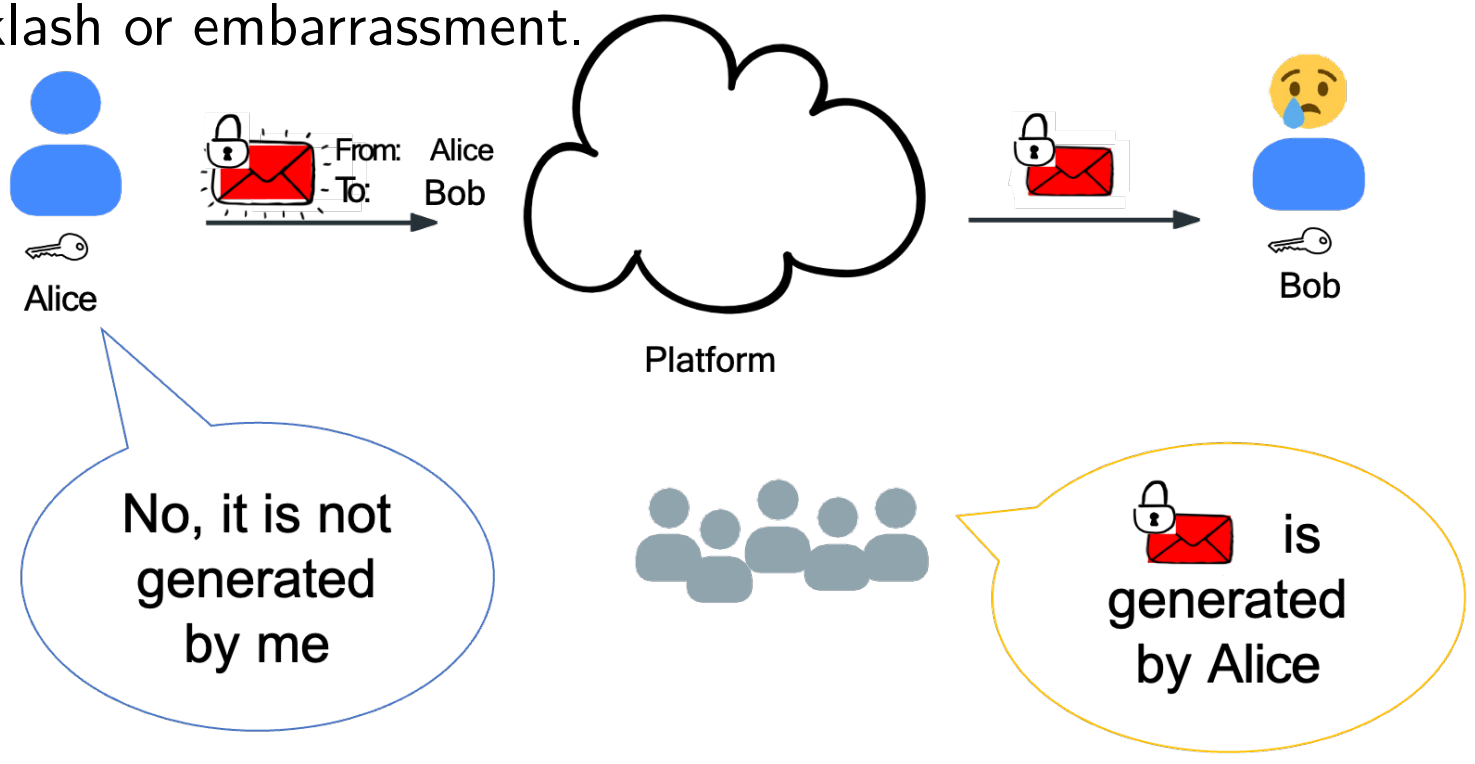


Figure 2: Attack on r-bind

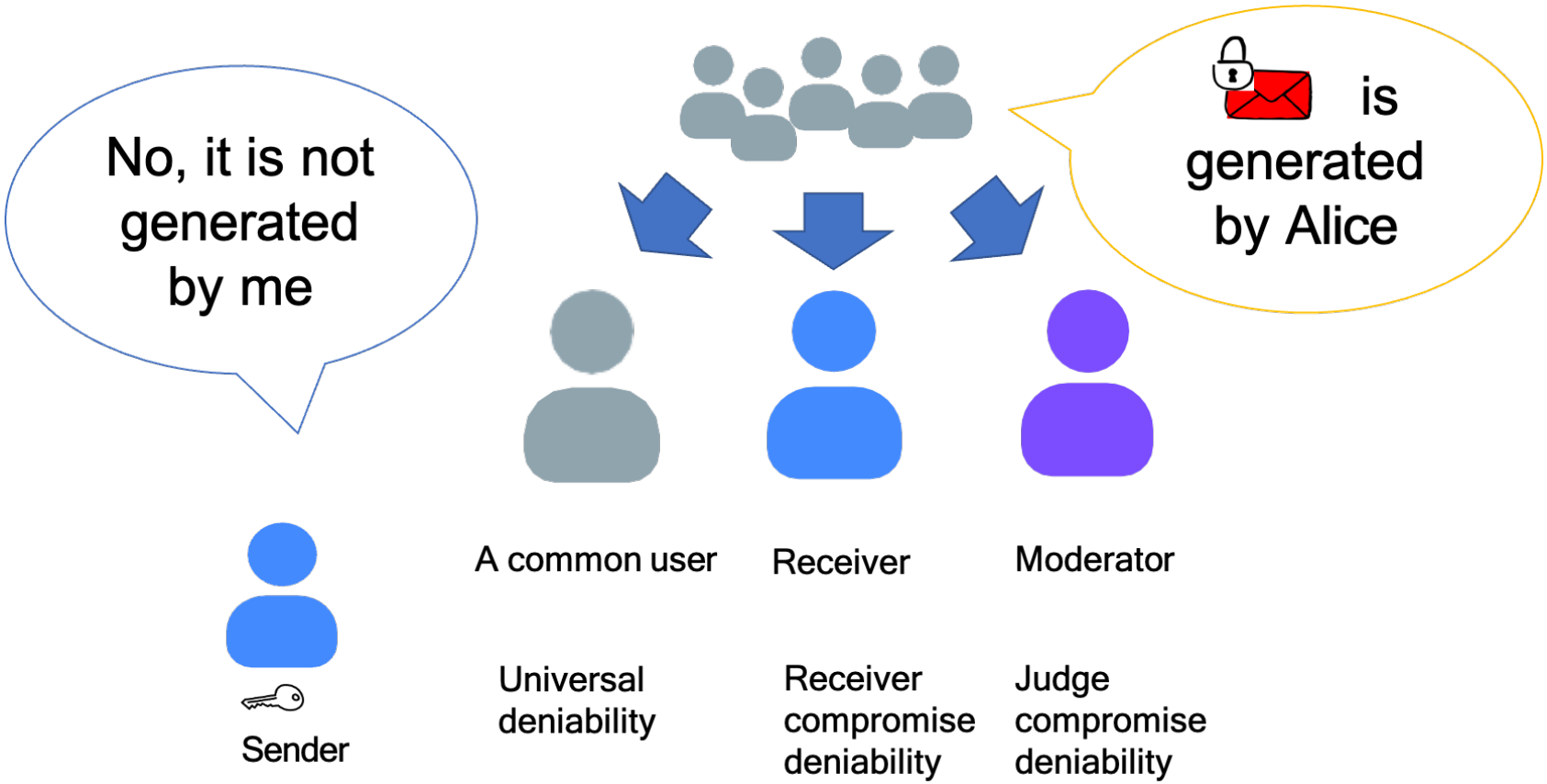
Background: Security properties of MF – Deniability I

Deniability: Informally, deniability ensures that when the receiver reports some malicious messages, only the moderator is able to validate the report. In other words, after **a compromise**, the sender can deny sending the messages technically, in order to avoid backlash or embarrassment.



Background: Security properties of MF – Deniability II

Deniability is formalized with three special properties: universal deniability, receiver compromise deniability and judge compromise deniability.



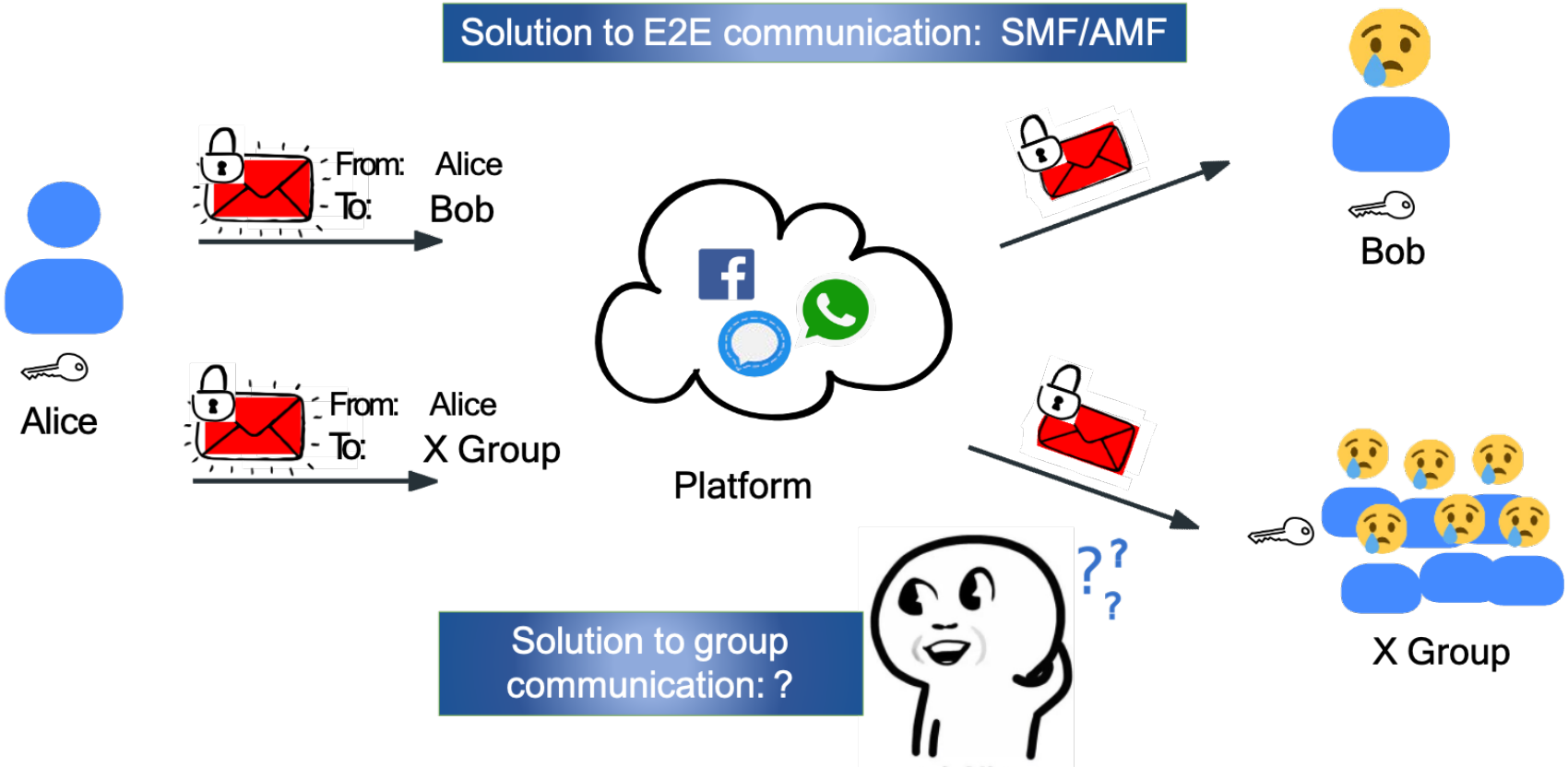
Background: SMF vs. AMF

There are mainly two kinds of message franking:

1. symmetric message franking (SMF), e.g., [Fac16, GLR17]
2. asymmetric message franking (AMF), e.g., [TGL+19]

Differences:
Compared with SMF [Fac16, GLR17], AMF [TGL+19] supports *third-party moderation*, decoupling the platform and the moderator. AMF is advantageous if the platform cannot adequately moderate messages, or if sub-communities want to enforce their own content policies.

Motivation: message franking in group communication scenarios



However, the existing AMF [TGL+19] only considers the case of *1-1 communication*. As for another common scenarios, *group communications*, no works have ever related to this topic.

Contributions

In this paper, we systematically explore message franking in group communication scenarios. The contributions are listed as follows.

- We introduce a new primitive called *asymmetric* group message franking (AGMF), and formalize its security notions.
- We present a variant of key encapsulation mechanism (KEM), called HPS-based KEM supporting Sigma protocols (HPS-KEM$^{\Sigma}$), and provide a construction based on the decisional Diffie-Hellman (DDH) assumption. The construction can be extended to be built based on the k-Linear assumption.
- We provide a framework of constructing AGMF from HPS-KEM$^{\Sigma}$, and show that it achieves the required security properties. Actually, we also obtain a framework of constructing AMF from HPS-KEM$^{\Sigma}$ (i.e., when the size of the receiver set is 1).

Comparison on AMF[TGL$^+$19] and AGMF

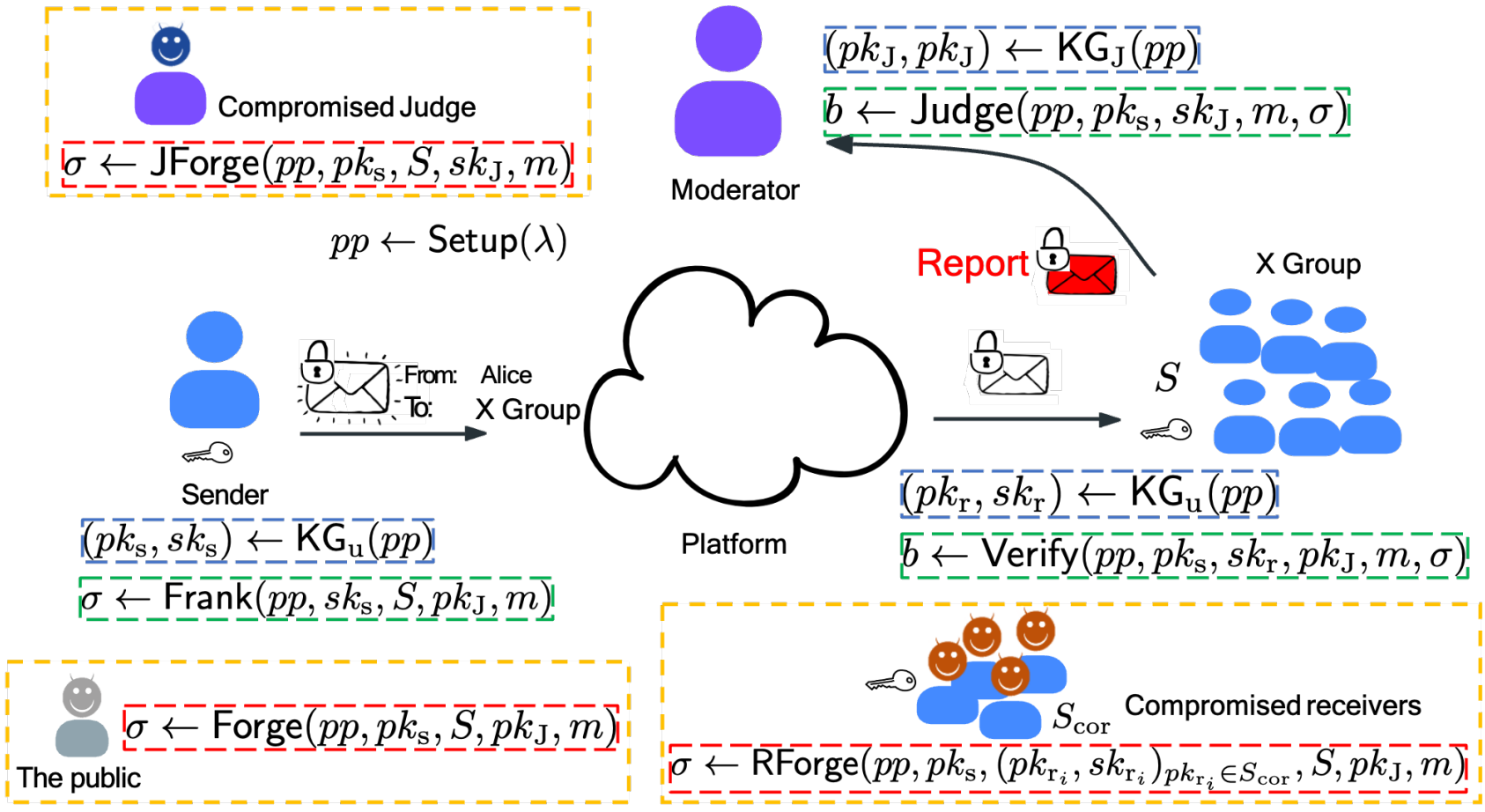
Here, we make a comparison on AMF[TGL$^+$19] and our AGMF.

Differences	AMF[TGL$^+$19]	Our AGMF
Corrupted receivers	$\lvert S_{\mathsf{cor}}\rvert = 1$	$\lvert S_{\mathsf{cor}}\rvert \geq 1$
Receiver anonymity	–	✓
Assumptions	Non-standard: KEA or Gap Diffie-Hellman	Standard: DDH or k-Lin

[0] $\lvert S_{\mathsf{cor}}\rvert$ is the number of corrupted receivers.

Algorithms/definition

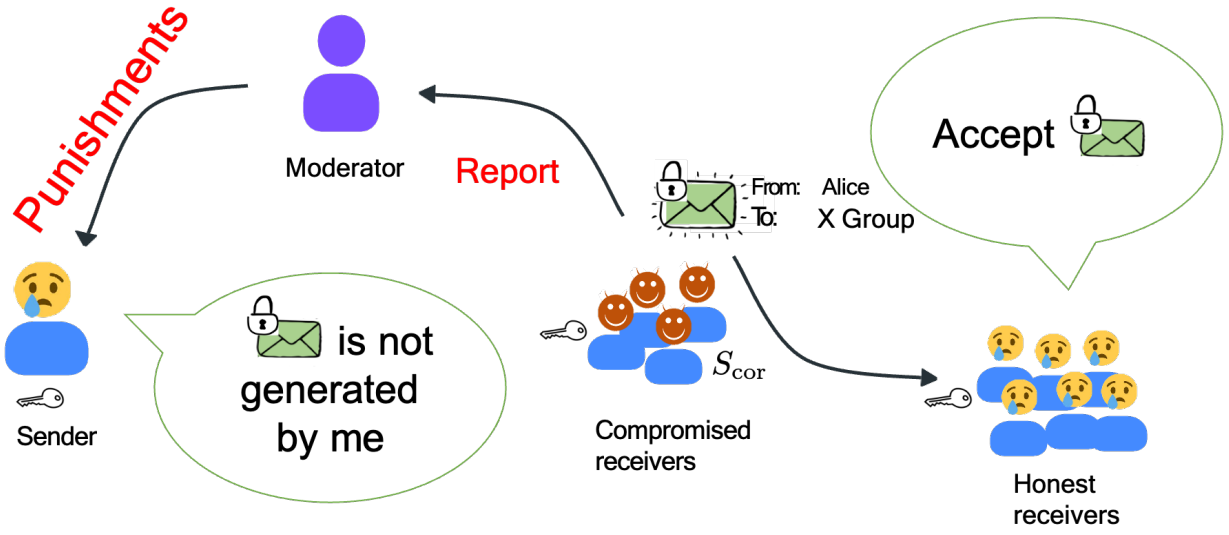
$\mathsf{AGMF} = (\mathsf{Setup}, \mathsf{KG_J}, \mathsf{KG_u}, \mathsf{Frank}, \mathsf{Verify}, \mathsf{Judge}, \mathsf{Forge}, \mathsf{RForge}, \mathsf{JForge})$



Security requirements I

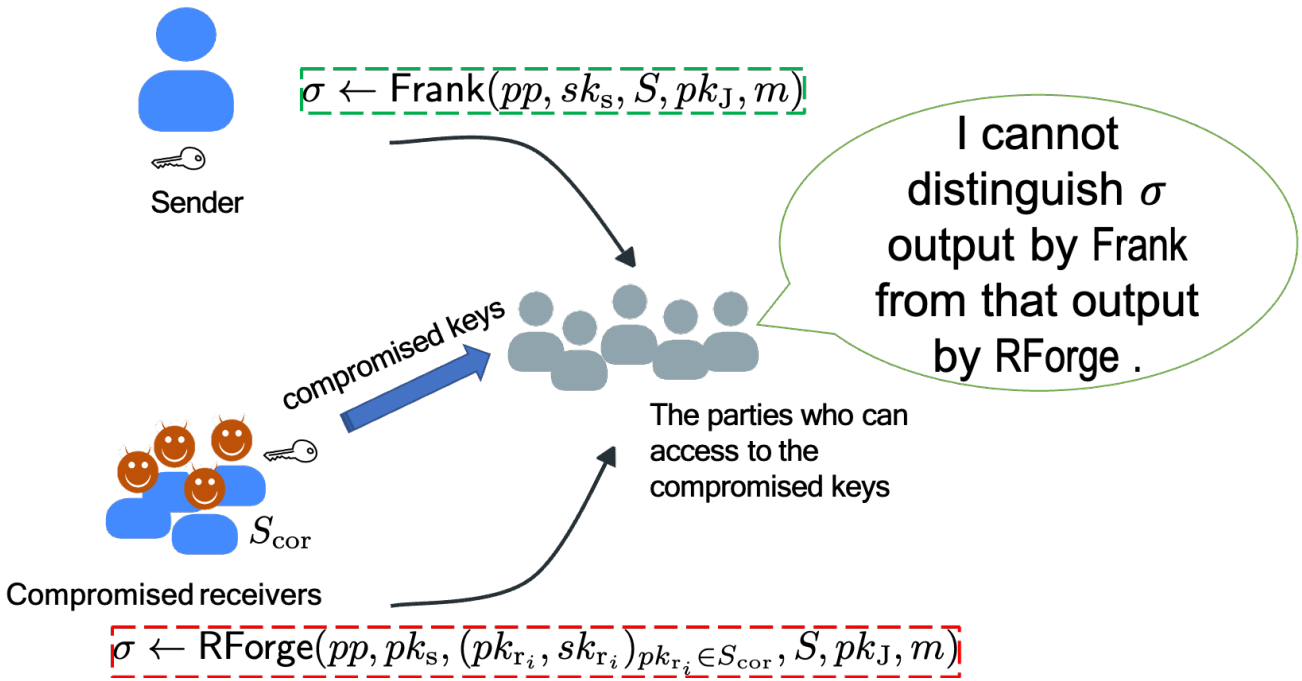
We consider three kinds of security requirements for AGMF: accountability, deniability, and receiver anonymity. Since some of them are similar to those of AMF [TGL+19], here we show the differences.

1. **Corruption of multiple receivers**.
   - *Receiver binding* for AGMF requires that any corrupted receivers cannot deceive the judge or the other honest receivers to frame the innocent sender.



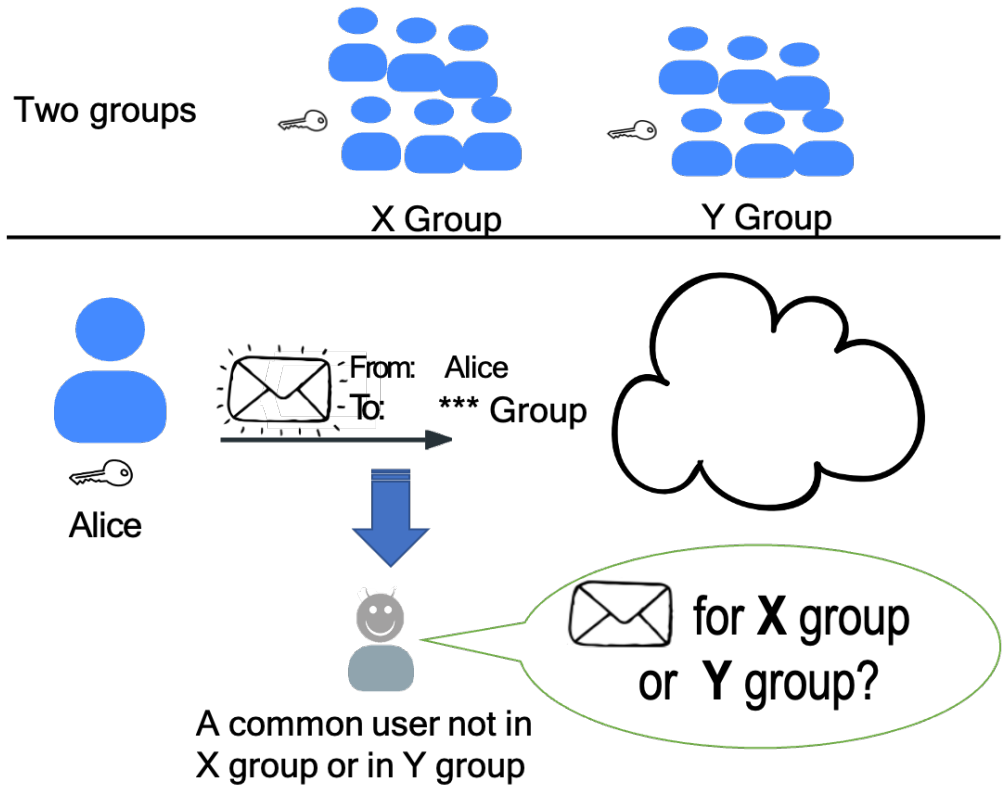
Security requirements II

- *Receiver compromise deniability* for AGMF requires that any corrupted users in the receiver set are able to create a signature, such that for other parties with access to these corrupted users' secret keys, it is indistinguishable from honestly-generated signatures.



Security requirements III

2. **Receiver anonymity** requires that any one (except for the receivers in the receiver set), including the judge, cannot tell which receiver set a signature is generated for.



Security requirements IV

3. Adaptive security. In all our proposed security models, the adversary is allowed to corrupt the receivers *adaptively*. In other words, how many and whose secret keys are compromised is unpredictable in practical scenarios, which is greatly different from that in AMF (i.e., only one receiver's secret key is compromised).

Building block: HPS-KEM$^{\Sigma}$

In order to provide a framework of constructing AGMF, we introduce a new primitive. This primitive is a variant of key encapsulation mechanism (KEM) satisfying that

(i) it can be interpreted from the perspective of hash proof system (HPS) [CS02],

(ii) for some special relations (about the public/secret keys, the encapsulated keys and ciphertexts), there exist corresponding Sigma protocols.

We call this primitive *HPS-based KEM supporting Sigma protocols* (HPS-KEM$^{\Sigma}$).

Definition of HPS-KEM$^{\Sigma}$ I

HPS-KEM$^{\Sigma}$ = (KEMSetup, KG, $\mathsf{encap}_{\mathsf{c}}$, $\mathsf{encap}^{*}_{\mathsf{c}}$, $\mathsf{encap}_{\mathsf{k}}$, decap, CheckKey, CheckCwel)

- $pp \leftarrow$ KEMSetup(1^{λ}): it outputs a public parameter pp.
- $(pk, sk) \leftarrow$ KG(pp): it outputs a key pairs (pk, sk).
- $c \leftarrow \mathsf{encap}_{\mathsf{c}}(pp; r)$: it outputs a well-formed ciphertext c.
- $c \leftarrow \mathsf{encap}^{*}_{\mathsf{c}}(pp; r^{*})$:
- $k \leftarrow \mathsf{encap}_{\mathsf{k}}(pp, pk; r)$: it outputs an encapsulated key $k \in \mathcal{K}$. it outputs a ciphertext c.
- $k' \leftarrow$ decap(pp, sk, c): it outputs an encapsulated key $k' \in \mathcal{K}$.
- $b \leftarrow$ CheckKey(pp, sk, pk): it checks whether the keys are well-formed.
- $b \leftarrow$ CheckCwel(pp, c, r^{*}): it checks whether the ciphertext is well-formed.

Definition of HPS-KEM$^{\Sigma}$ II

For any pp generated by KEMSetup(1^{λ}), we define some relations as follows and we require there exists a Sigma protocol for each relation:

$$\mathcal{R}_{\mathsf{s}} = \{(sk, pk) : \mathsf{CheckKey}(pp, sk, pk) = 1\}$$

$$\mathcal{R}_{\mathsf{c,k}} = \{(r, (c, k, pk)) : c = \mathsf{encap}_{\mathsf{c}}(pp; r) \wedge k = \mathsf{encap}_{\mathsf{k}}(pp, pk; r)\}$$

$$\mathcal{R}^{*}_{\mathsf{c}} = \{(r^{*}, c) : c = \mathsf{encap}^{*}_{\mathsf{c}}(pp; r^{*})\}$$

where

- $\mathcal{R}_{\mathsf{s}}$ is a relation proving that the keys are valid,
- $\mathcal{R}_{\mathsf{c,k}}$ is a relation proving that (c, k) are generated via $\mathsf{encap}_{\mathsf{c}}$ and $\mathsf{encap}_{\mathsf{k}}$,
- $\mathcal{R}^{*}_{\mathsf{c}}$ is a relation proving that c is a ciphertext output by $\mathsf{encap}^{*}_{\mathsf{c}}$.

Definition of HPS-KEM$^{\Sigma}$ III

We also require the properties:

- *universality*,
- *unexplainability*,
- *indistinguishability*,
- *SK-second-preimage resistance(SK-2PR)*,
- *smoothness*.

Due to the limitation of time, please refer to the definitions of these properties and a concrete construction based on DDH in our paper [LZH+23].

AGMF from HPS-KEM$^{\Sigma}$: setup and key generation algorithms

Taking HPS-KEM$^{\Sigma}$ as a building block, we can construct AGMF as follows.

$\underline{\mathsf{Setup}(\lambda)}$: $pp \leftarrow \mathsf{KEMSetup}(1^{\lambda})$; Return pp

$\underline{\mathsf{KG_J}(pp)}$: $(pk_{\mathsf{J}}, sk_{\mathsf{J}}) \leftarrow \mathsf{KG}(pp)$; Return $(pk_{\mathsf{J}}, sk_{\mathsf{J}})$

$\underline{\mathsf{KG_u}(pp)}$: $(pk, sk) \leftarrow \mathsf{KG}(pp)$; Return (pk, sk)

AGMF from HPS-KEM$^{\Sigma}$: Franking and verification algorithms

$\underline{\mathsf{Frank}(pp, sk_{\mathsf{s}}, S, pk_{\mathsf{J}}, m)}$:
$r \leftarrow \mathcal{RS}$; $c \leftarrow \mathsf{encap}_{\mathsf{c}}(pp; r)$; $k_{\mathsf{J}} \leftarrow \mathsf{encap}_{\mathsf{k}}(pp, pk_{\mathsf{J}}; r)$
For $pk_{\mathsf{r}_i} \in S$:
  $k_{\mathsf{r}_i} \leftarrow \mathsf{encap}_{\mathsf{k}}(pp, pk_{\mathsf{r}_i}; r)$
$x \leftarrow (sk_{\mathsf{s}}, r, \perp)$; $y \leftarrow (pp, pk_{\mathsf{s}}, pk_{\mathsf{J}}, c, k_{\mathsf{J}})$
$\overline{m} \leftarrow (m || \{k_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S})$; $\pi \leftarrow \mathsf{NIZK}^{\mathcal{R}}.\mathsf{PoK}(\overline{m}, x, y)$
Return $\sigma \leftarrow (\pi, c, k_{\mathsf{J}}, \{k_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S})$

$\underline{\mathsf{Verify}(pp, pk_{\mathsf{s}}, sk_{\mathsf{r}}, pk_{\mathsf{J}}, m, \sigma)}$:
$(\pi, c, k_{\mathsf{J}}, \{k_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S}) \leftarrow \sigma$; $y \leftarrow (pp, pk_{\mathsf{s}}, pk_{\mathsf{J}}, c, k_{\mathsf{J}})$
$\overline{m} \leftarrow (m || \{k_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S})$
If $\mathsf{NIZK}^{\mathcal{R}}.\mathsf{PoKVer}(\overline{m}, \pi, y) = 0$: Return 0
If $\mathsf{decap}(pp, sk_{\mathsf{r}}, c) \in \{k_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S}$: Return 1
Return 0

$\underline{\mathsf{Judge}(pp, pk_{\mathsf{s}}, sk_{\mathsf{J}}, m, \sigma)}$:
$(\pi, c, k_{\mathsf{J}}, \{k_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S}) \leftarrow \sigma$; $y \leftarrow (pp, pk_{\mathsf{s}}, pk_{\mathsf{J}}, c, k_{\mathsf{J}})$
$\overline{m} \leftarrow (m || \{k_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S})$
If $\mathsf{NIZK}^{\mathcal{R}}.\mathsf{PoKVer}(\overline{m}, \pi, y) = 0$: Return 0
If $\mathsf{decap}(pp, sk_{\mathsf{J}}, c) \neq k_{\mathsf{J}}$: Return 0
Return 1

AGMF from HPS-KEM$^{\Sigma}$: Forging algorithms

$\underline{\mathsf{Forge}(pp, pk_{\mathsf{s}}, S, pk_{\mathsf{J}}, m)}$:
$r^* \leftarrow \mathcal{RS}^*$; $c \leftarrow \mathsf{encap}_{\mathsf{c}}^*(pp; r^*)$; $k_{\mathsf{J}} \leftarrow \mathcal{K}$
For $pk_{\mathsf{r}_i} \in S$: $k_{\mathsf{r}_i} \leftarrow \mathcal{K}$
$x \leftarrow (\perp, \perp, r^*)$; $y \leftarrow (pp, pk_{\mathsf{s}}, pk_{\mathsf{J}}, c, k_{\mathsf{J}})$; $\overline{m} \leftarrow (m || \{k_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S})$
$\pi \leftarrow \mathsf{NIZK}^{\mathcal{R}}.\mathsf{PoK}(\overline{m}, x, y)$
Return $\sigma \leftarrow (\pi, c, k_{\mathsf{J}}, \{k_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S})$

$\underline{\mathsf{RForge}(pp, pk_{\mathsf{s}}, \{pk_{\mathsf{r}_i}, sk_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S_{\mathsf{cor}}}, S, pk_{\mathsf{J}}, m)}$:
// S_{cor} here is the set of corrupted receivers
$r^* \leftarrow \mathcal{RS}^*$; $c \leftarrow \mathsf{encap}_{\mathsf{c}}^*(pp; r^*)$; $k_{\mathsf{J}} \leftarrow \mathcal{K}$
For $pk_{\mathsf{r}_i} \in S \backslash S_{\mathsf{cor}}$: $k_{\mathsf{r}_i} \leftarrow \mathcal{K}$
For $pk_{\mathsf{r}_i} \in S_{\mathsf{cor}}$: $k_{\mathsf{r}_i} \leftarrow \mathsf{decap}(pp, sk_{\mathsf{r}_i}, c)$
$x \leftarrow (\perp, \perp, r^*)$; $y \leftarrow (pp, pk_{\mathsf{s}}, pk_{\mathsf{J}}, c, k_{\mathsf{J}})$; $\overline{m} \leftarrow (m || \{k_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S})$
$\pi \leftarrow \mathsf{NIZK}^{\mathcal{R}}.\mathsf{PoK}(\overline{m}, x, y)$
Return $\sigma \leftarrow (\pi, c, k_{\mathsf{J}}, \{k_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S})$

$\underline{\mathsf{JForge}(pp, pk_{\mathsf{s}}, S, sk_{\mathsf{J}}, m)}$:
$r^* \leftarrow \mathcal{RS}^*$; $c \leftarrow \mathsf{encap}_{\mathsf{c}}^*(pp; r^*)$; $k_{\mathsf{J}} \leftarrow \mathsf{decap}(pp, sk_{\mathsf{J}}, c)$
For $pk_{\mathsf{r}_i} \in S$: $k_{\mathsf{r}_i} \leftarrow \mathcal{K}$
$x \leftarrow (\perp, \perp, r^*)$; $y \leftarrow (pp, pk_{\mathsf{s}}, pk_{\mathsf{J}}, c, k_{\mathsf{J}})$; $\overline{m} \leftarrow (m || \{k_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S})$
$\pi \leftarrow \mathsf{NIZK}^{\mathcal{R}}.\mathsf{PoK}(\overline{m}, x, y)$
Return $\sigma \leftarrow (\pi, c, k_{\mathsf{J}}, \{k_{\mathsf{r}_i}\}_{pk_{\mathsf{r}_i} \in S})$

AGMF from HPS-KEM$^{\Sigma}$: NIZK

Applying Fait-Shamir transform to the Sigma protocols for the following relation, we can get a NIZK scheme NIZK$^{\mathcal{R}}$.

$$\mathcal{R} = \{ \ (\ (sk_{\mathsf{s}}, r, r^*), (pp, pk_{\mathsf{s}}, pk_{\mathsf{J}}, c, k_{\mathsf{J}}) \ ) :$$
$$(\ (sk_{\mathsf{s}}, pk_{\mathsf{s}}) \in \mathcal{R}_{\mathsf{s}} \wedge (r, (c, k_{\mathsf{J}}, pk_{\mathsf{J}})) \in \mathcal{R}_{\mathsf{c,k}} \ )$$
$$/\!/ \text{ For normal case, the clause is true when invoking Frank}$$
$$\vee (\ (r^*, c) \in \mathcal{R}^*_{\mathsf{c}} \ )$$
$$/\!/ \text{ For deniability, the clause is true when invoking the forging}$$
$$/\!/ \text{ algorithms: Forge, RForge and JForge}$$
$$\}$$

- $\mathcal{R}_{\mathsf{s}}$ is a relation proving that the sender's public/secret keys are valid. ($\Rightarrow$ accountability)
- $\mathcal{R}_{\mathsf{c,k}}$ is a relation proving that a with the same randomness r. ($\Rightarrow$ accountability)
- $\mathcal{R}^*_{\mathsf{c}}$ is a relation proving that c is a ciphertext output by encap$^*_{\mathsf{c}}$ with randomness r^*. ($\Rightarrow$ deniability)

Security analysis and lower bound I

For the security of AGMF, we have the following theorem.

Theorem 1

If a HPS-KEM$^{\Sigma}$ scheme HPS-KEM$^{\Sigma}$ *is universal, unexplainable, indistinguishble, SK-second-preimage resistant and smooth, and* NIZK$^{\mathcal{R}} = (\mathsf{PoK}, \mathsf{PoKVer})$ *is a Fiat-Shamir NIZK proof system for $\mathcal{R}$, then our scheme* AGMF *achieves the accountability (receiver binding and sender binding), deniability (universal deniability, receiver compromise deniability, and judge compromise deniability) and receiver anonymity simultaneously.*

Due to the limitation of time, please refer to the proof for Theorem 1 in our paper [LZH+23].

Security analysis and lower bound II

Following we present a lower bound of the size of AGMF signature.

Theorem 2

Any AGMF with receiver binding and receiver compromise deniability must have signature size $\Omega(n)$, where n is the number of the members in S.

Due to the limitation of time, please refer to the proof for Theorem 2 in our paper [LZH+23].

References

[CS02] Ronald Cramer and Victor Shoup.
Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption.
In *EUROCRYPT*, pages 45–64. Springer, 2002.

[Fac16] Facebook.
Messenger secret conversations technical whitepaper.
2016.
https://fbnewsroomus.files.wordpress.com/2016/07/secret_conversations_whitepaper-1.pdf.

[GLR17] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart.
Message franking via committing authenticated encryption.
In *CRYPTO 2017*, pages 66–97. Springer, 2017.

[LZH+23] Junzuo Lai, Gongxian Zeng, Zhengan Huang, Siu Ming Yiu, Xin Mu, and Jian Weng.
Asymmetric group message franking: Definitions and constructions.
In *EUROCRYPT 2023*, pages 67–97. Springer, 2023.

[TGL+19] 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*, pages 222–250. Springer, 2019.


Thank you