Asymmetric Message Franking

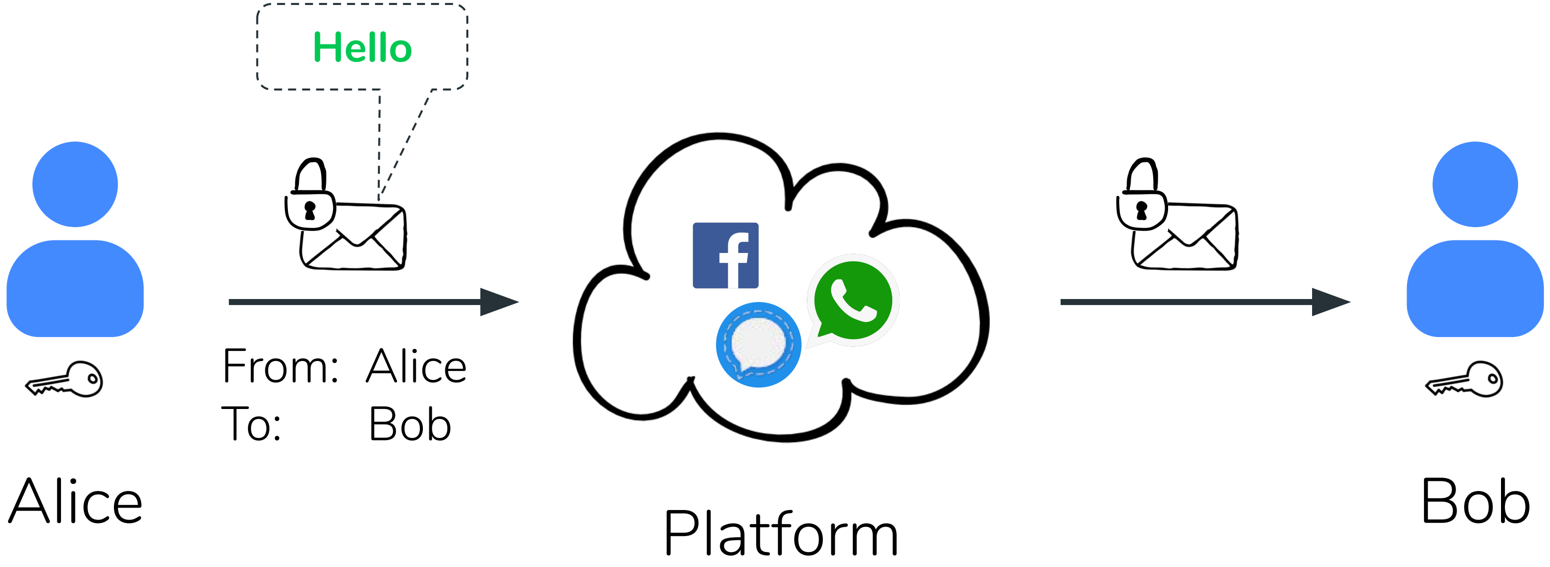
Content Moderation for Metadata-Private End-to-End Encryption

Nirvan Tyagi Paul Grubbs Julia Len

Ian Miers Tom Ristenpart

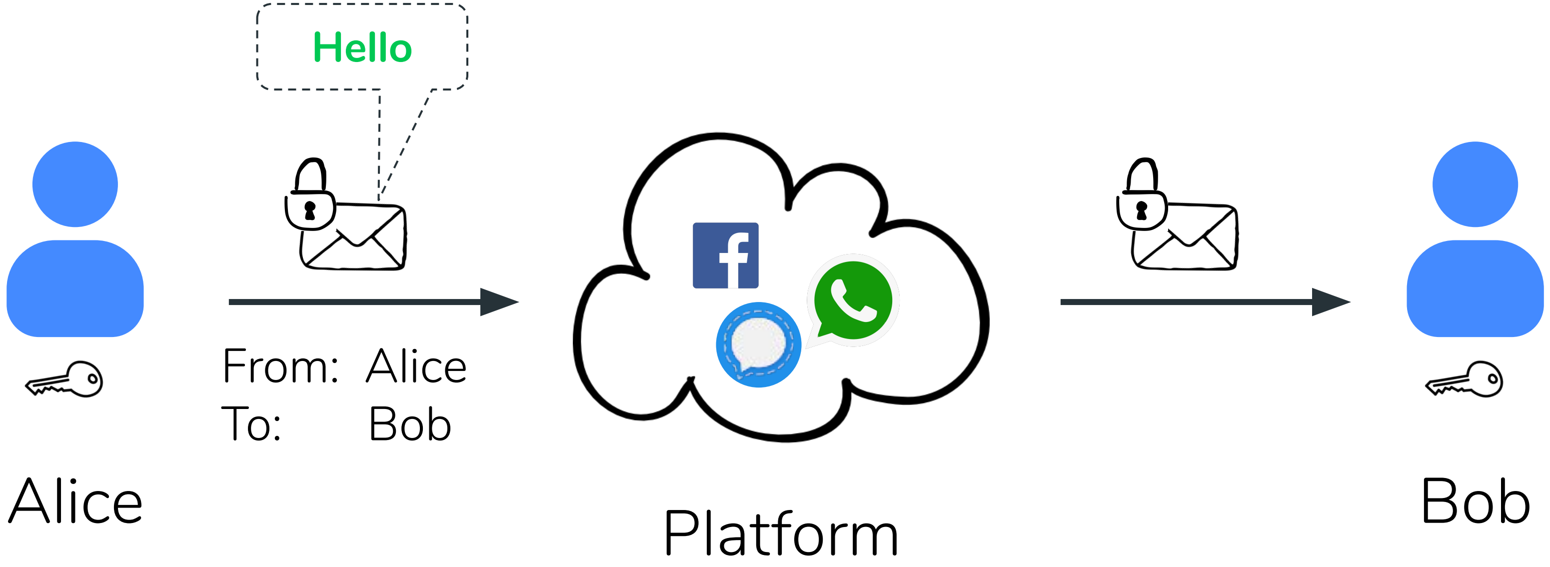
CRYPTO 2019

Setting: End-to-end encrypted messaging



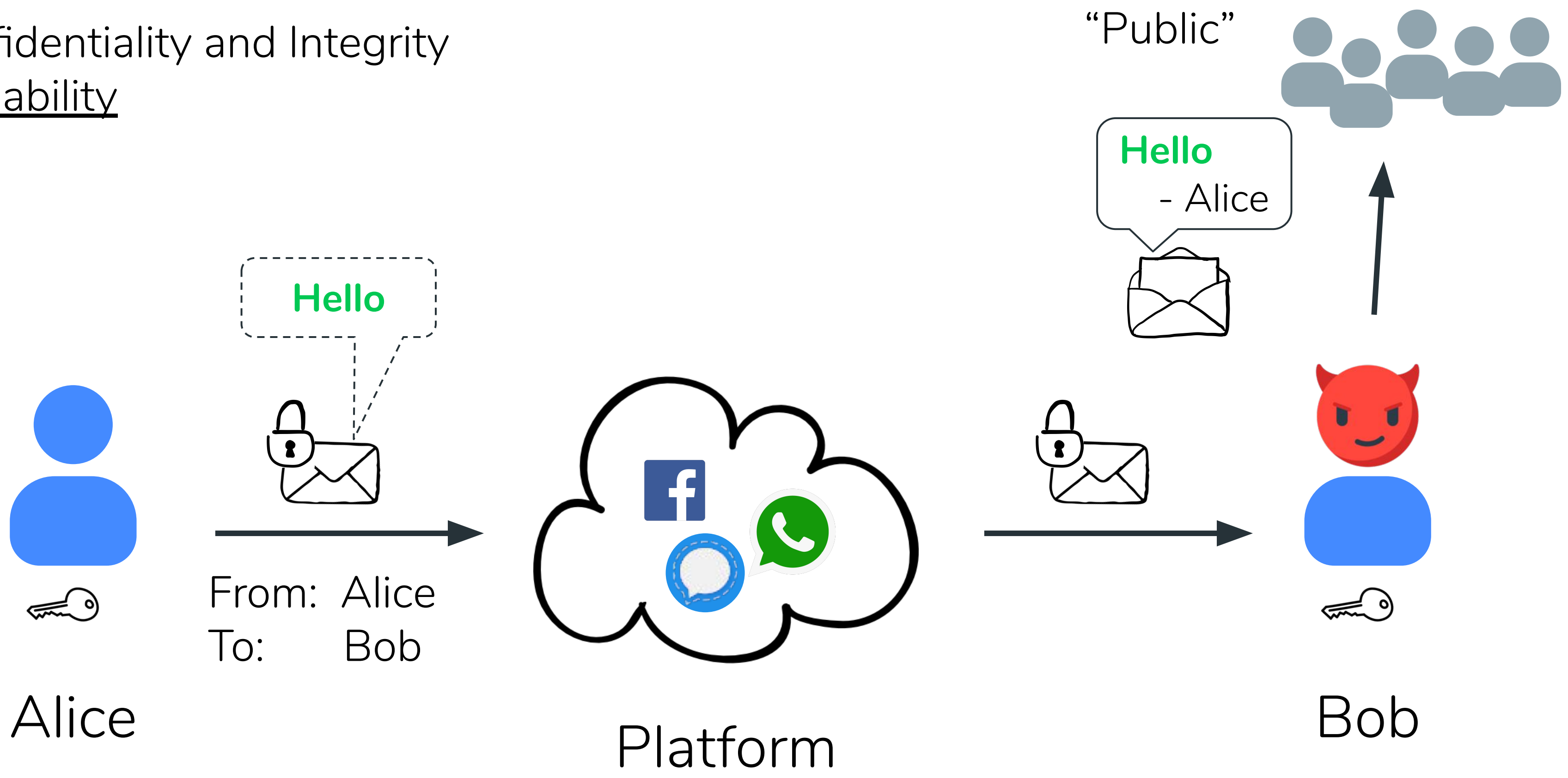
Setting: End-to-end encrypted messaging

- Confidentiality and Integrity



Setting: End-to-end encrypted messaging

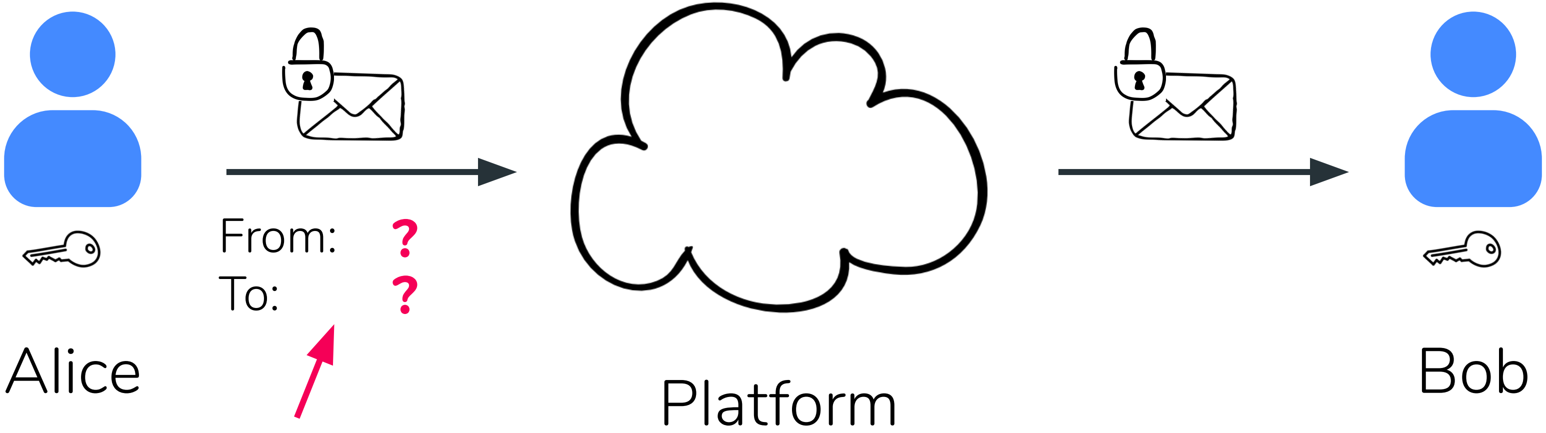
- Confidentiality and Integrity
- <u>Deniability</u>



[OTR BGB '04], [Signal X3DH '16]

Setting: End-to-end encrypted messaging

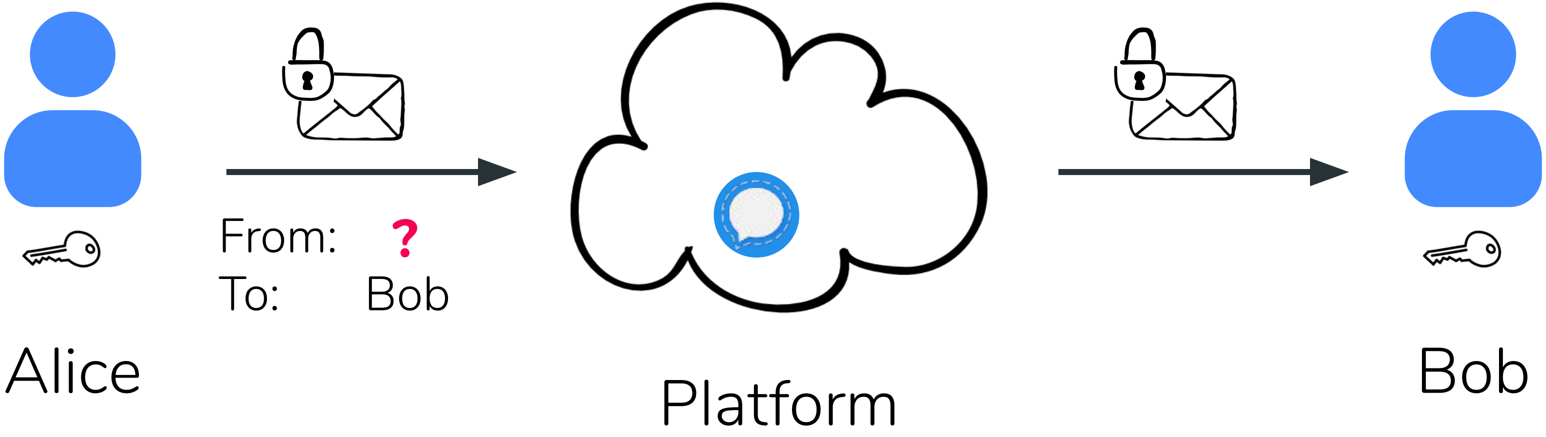
- Confidentiality and Integrity
- Deniability
- Metadata privacy



[Dissent OSDI'12], [Riposte S&P'15], [Vuvuzela SOSP'15], [Pung OSDI'16] . . .

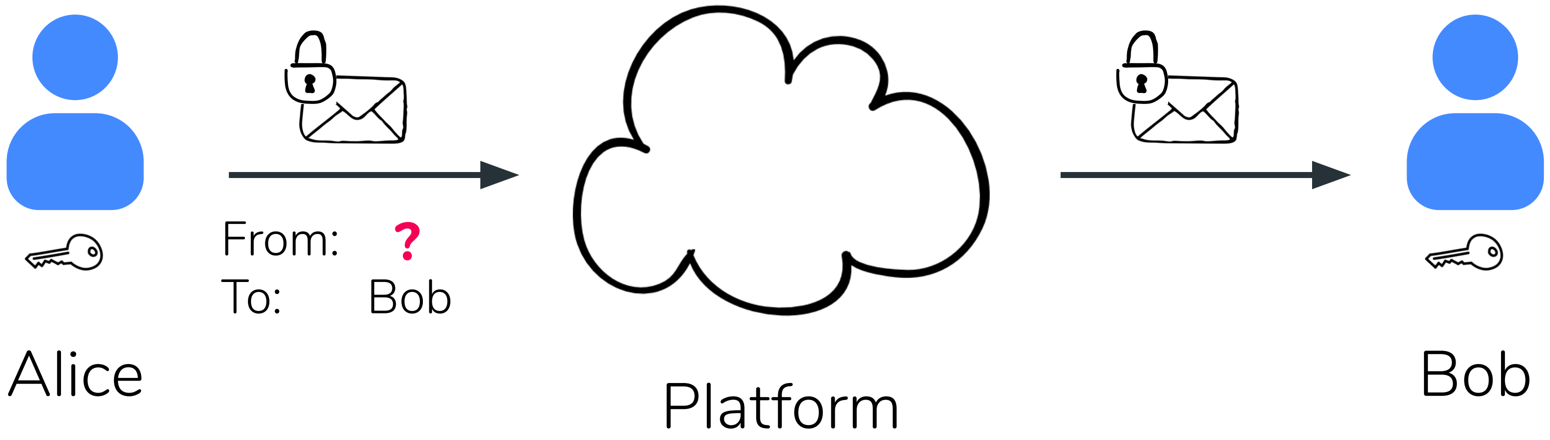
Setting: End-to-end encrypted messaging

- Confidentiality and Integrity
- Deniability
- Metadata privacy

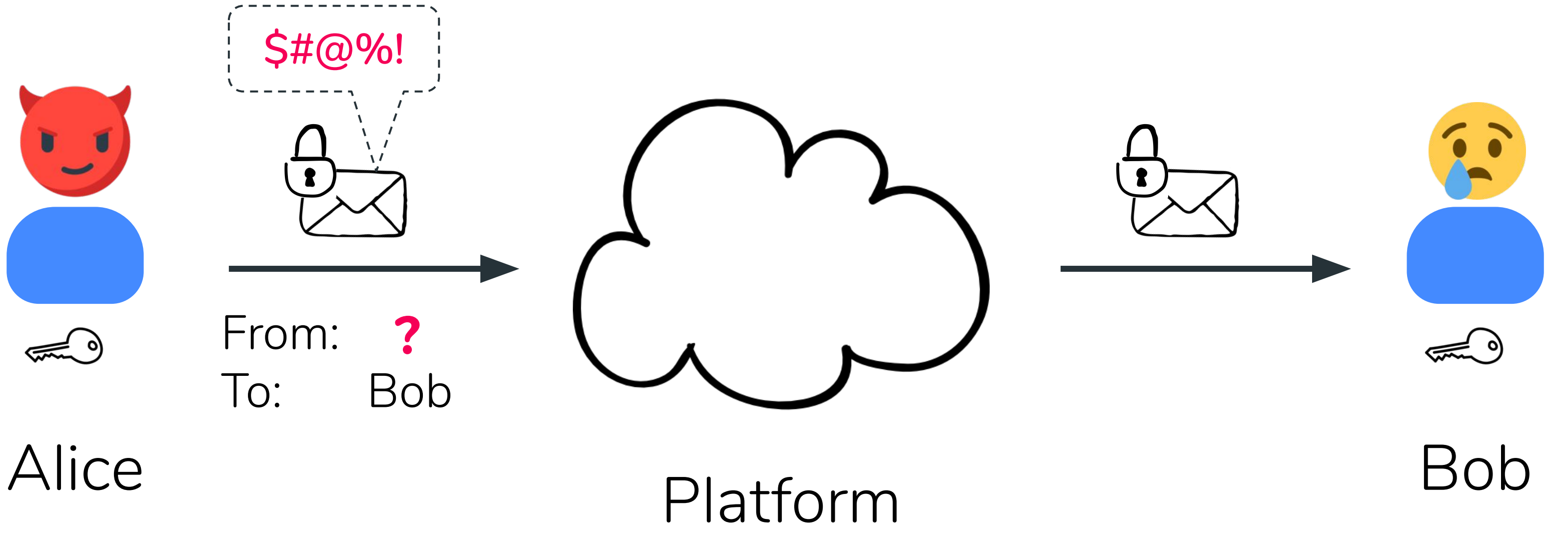


[Dissent OSDI’12], [Riposte S&P’15], [Vuvuzela SOSP’15], [Pung OSDI’16] . . .

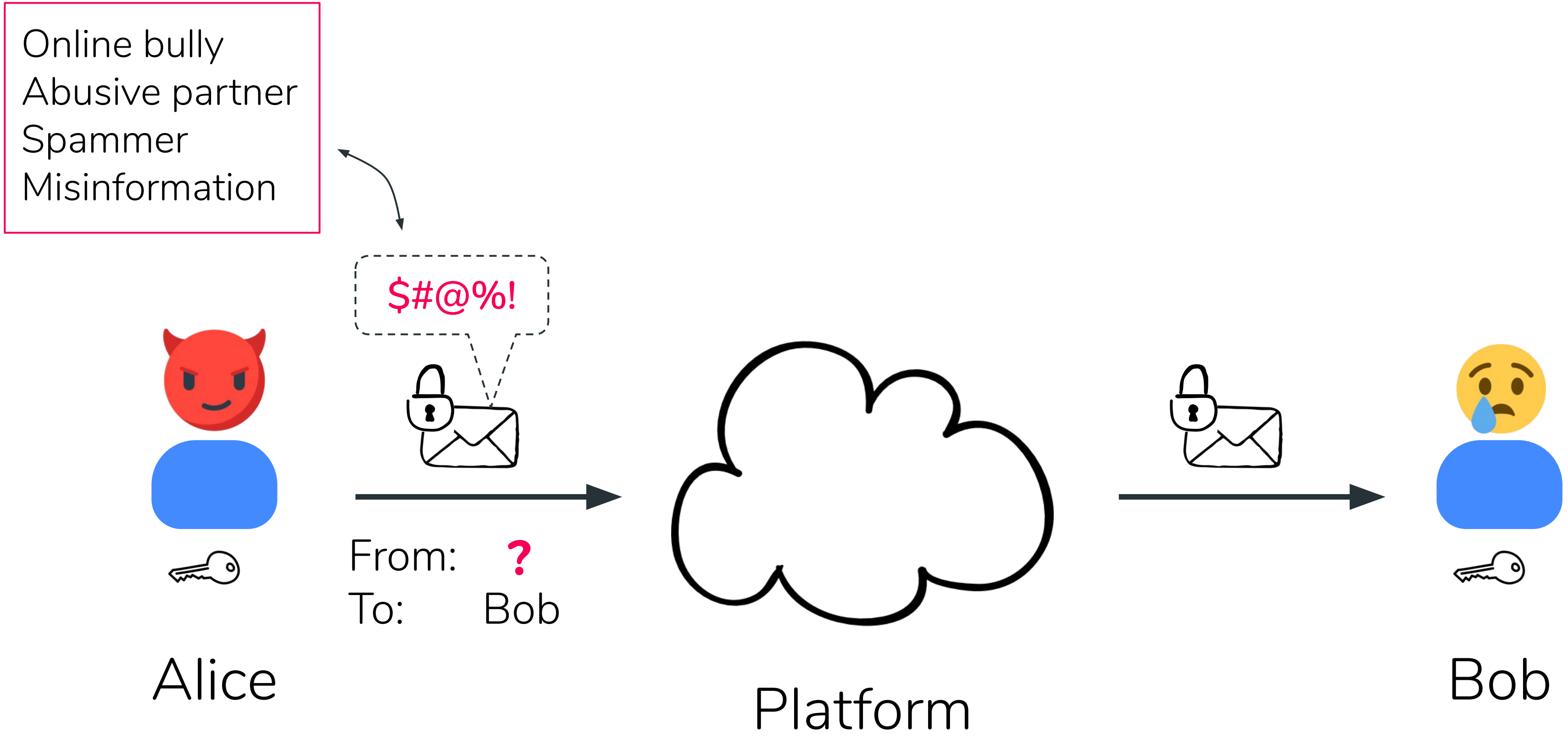
What about abuse?



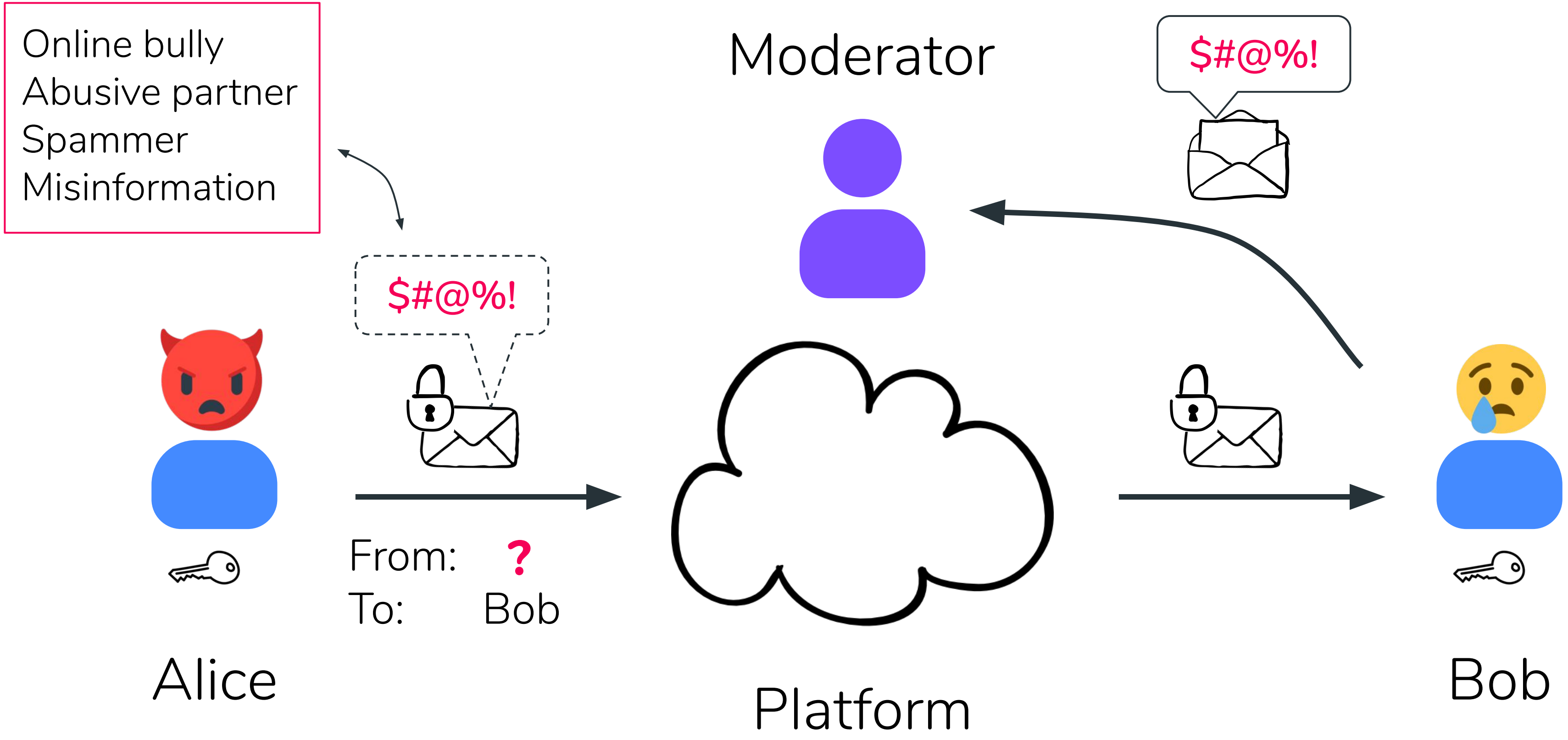
What about abuse?



What about abuse?

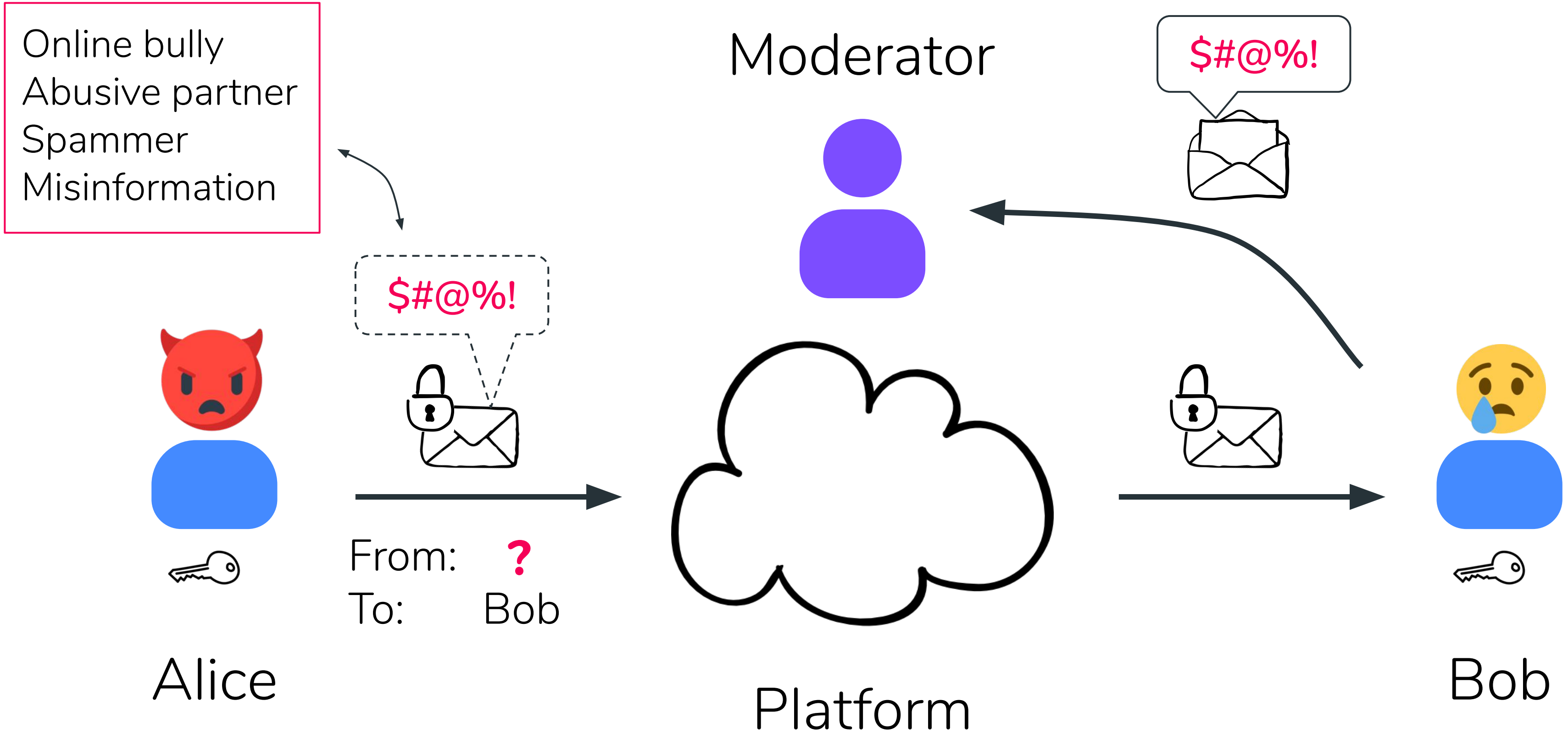


What about abuse?



What about abuse?

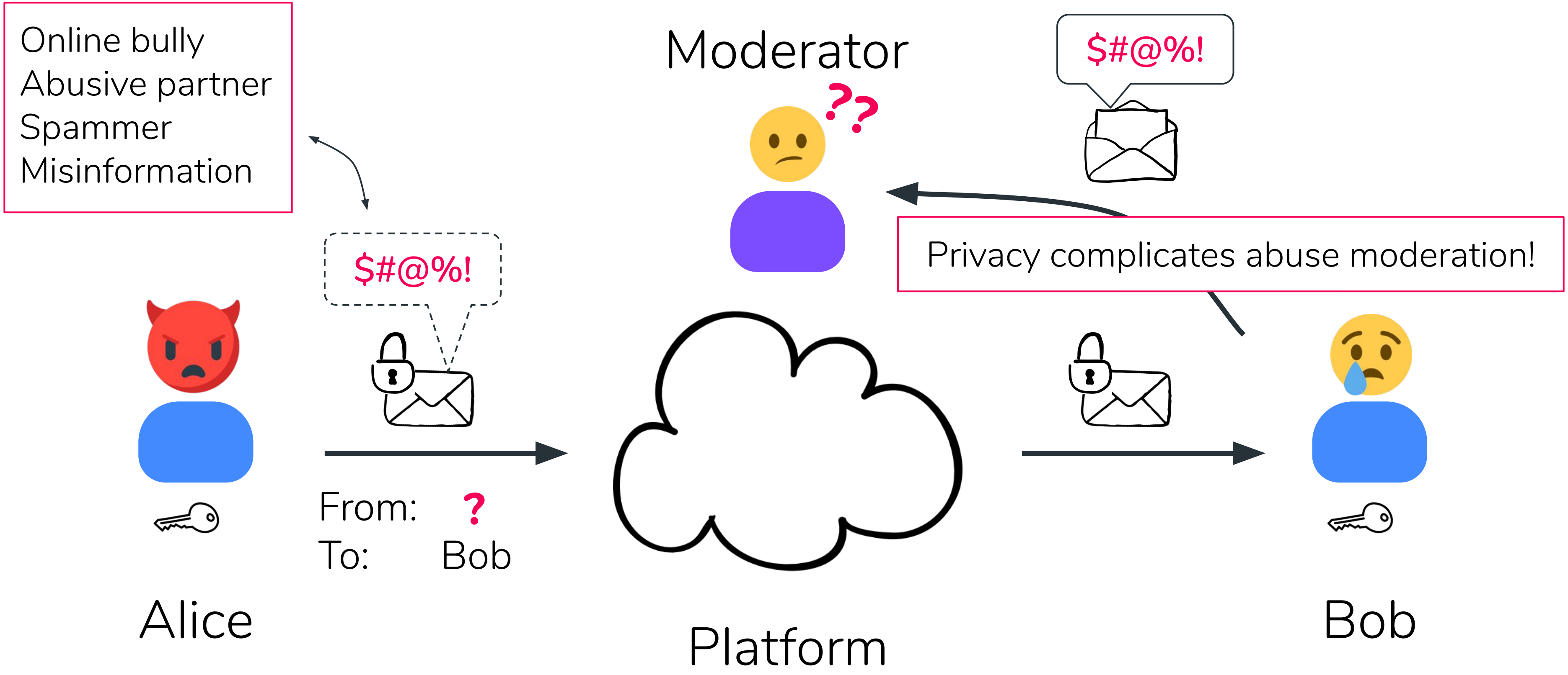
Moderation is a big priority:
Facebook employs ≈15K content moderators*



* “The secret lives of Facebook moderators in America” [The Verge 2019]

What about abuse?

Moderation is a big priority:
Facebook employs ≈15K content moderators*

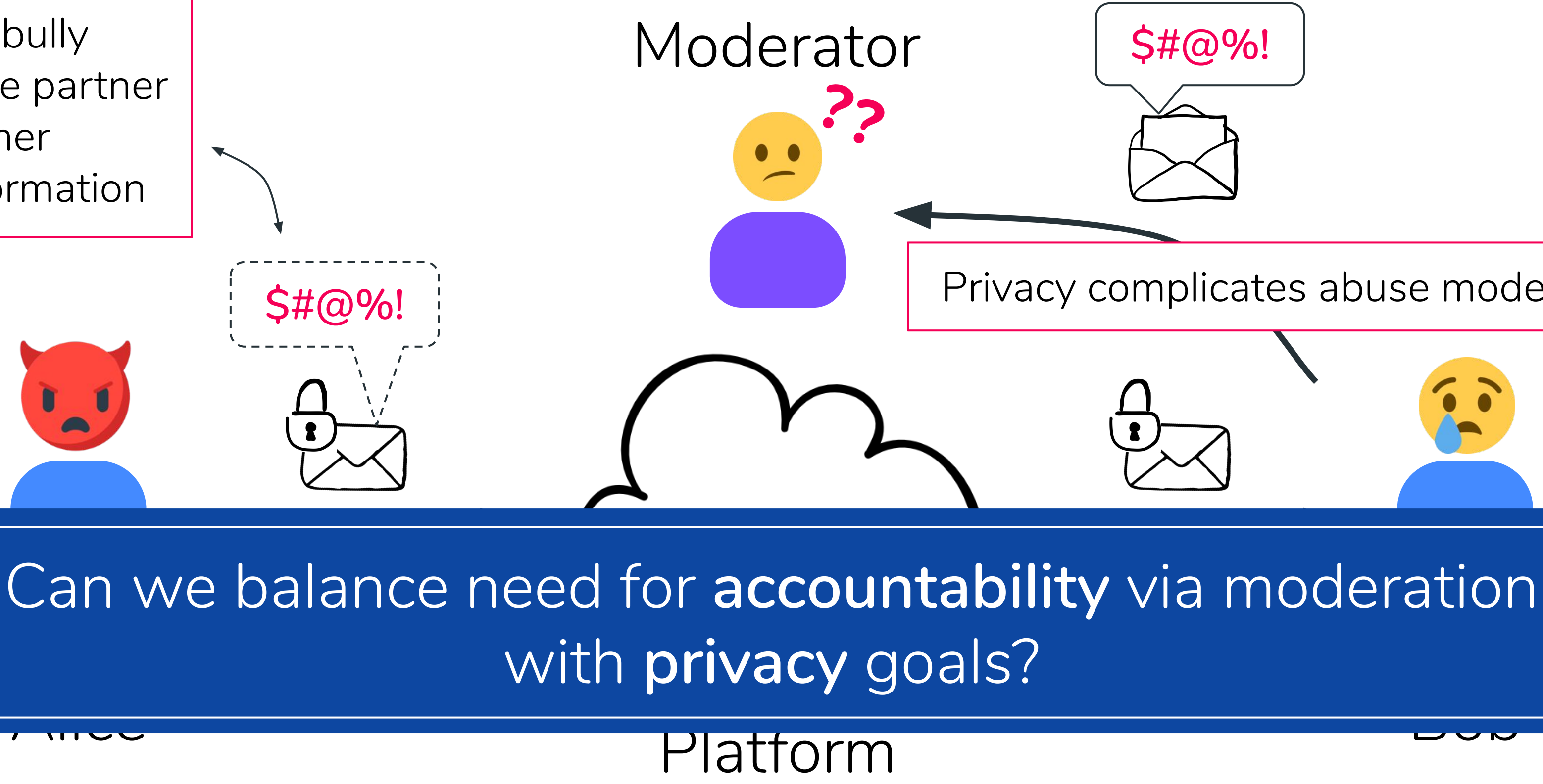


* "The secret lives of Facebook moderators in America" [The Verge 2019]

What about abuse?

Moderation is a big priority:
Facebook employs ≈15K content moderators*

Online bully
Abusive partner
Spammer
Misinformation



Privacy complicates abuse moderation!

Can we balance need for accountability via moderation with privacy goals?

Can we balance need for **accountability** via moderation with **privacy** goals?

* "The secret lives of Facebook moderators in America" [The Verge 2019]

Our contributions

- **Asymmetric Message Franking (AMF)**: a new cryptographic primitive for content moderation
    - **Metadata-privacy**: message sender and/or recipient identities hidden
    - **Third-party moderation**: moderator decoupled from message-delivery platform
- Formal accountability and deniability security notions for content moderation
- Construction inspired by “designated-verifier” signatures
- Implementation and proof-of-concept deployment

[TGLMR CRYPTO’19]

Prior work on moderation in E2E encryption

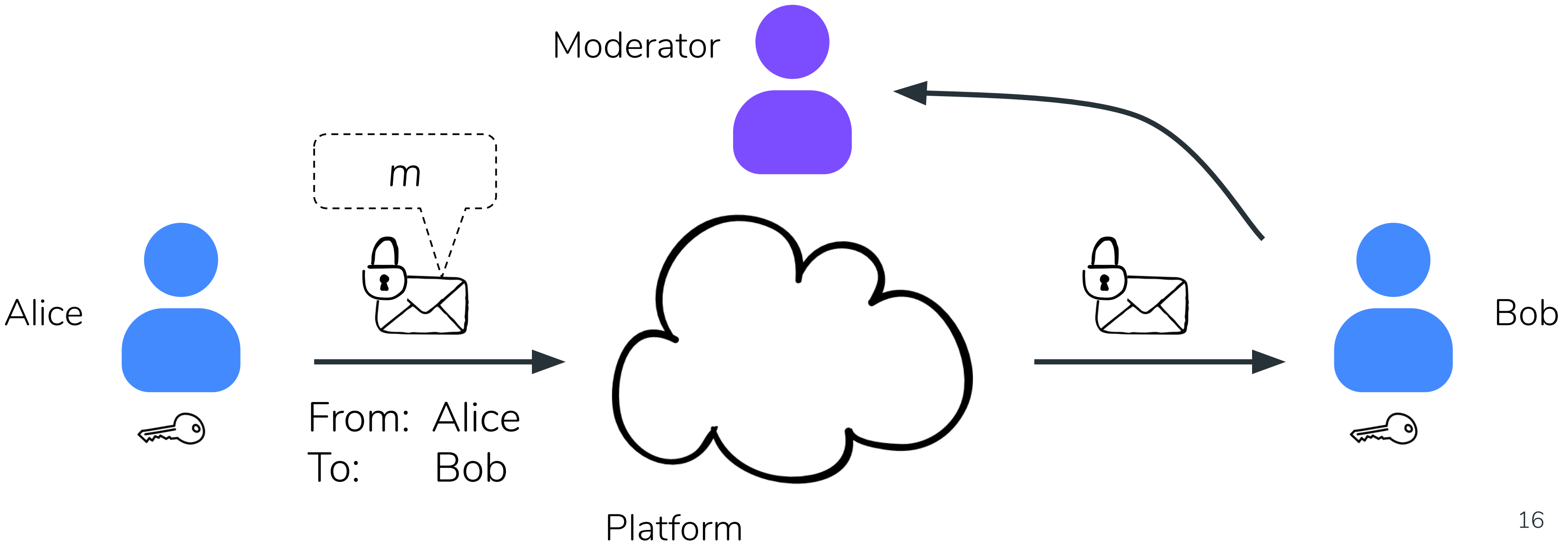
Message franking [FB white paper '17], [GLR CRYPTO'17], [DGRW CRYPTO'18]

- Content-based moderation of encryption that is NOT metadata-private
- Compactly-committing authenticated encryption

Prior work on moderation in E2E encryption

Message franking [FB white paper '17], [GLR CRYPTO'17], [DGRW CRYPTO'18]

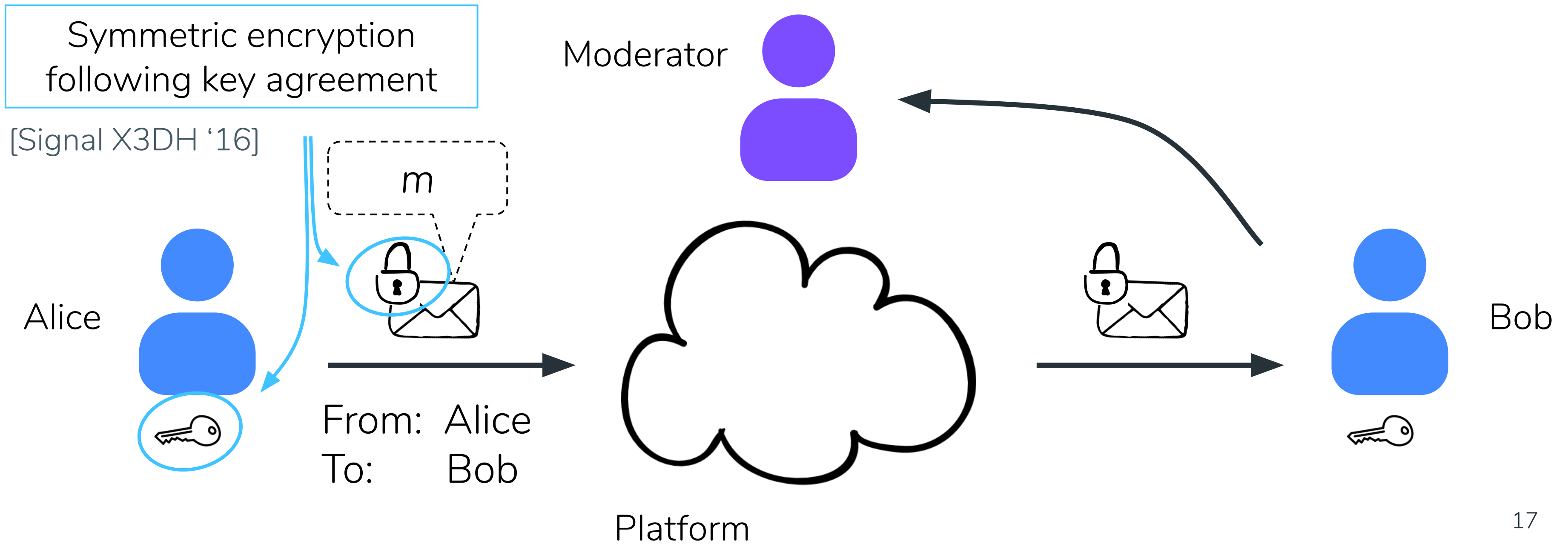
- Content-based moderation of encryption that is NOT metadata-private
- Compactly-committing authenticated encryption



Prior work on moderation in E2E encryption

Message franking [FB white paper '17], [GLR CRYPTO'17], [DGRW CRYPTO'18]

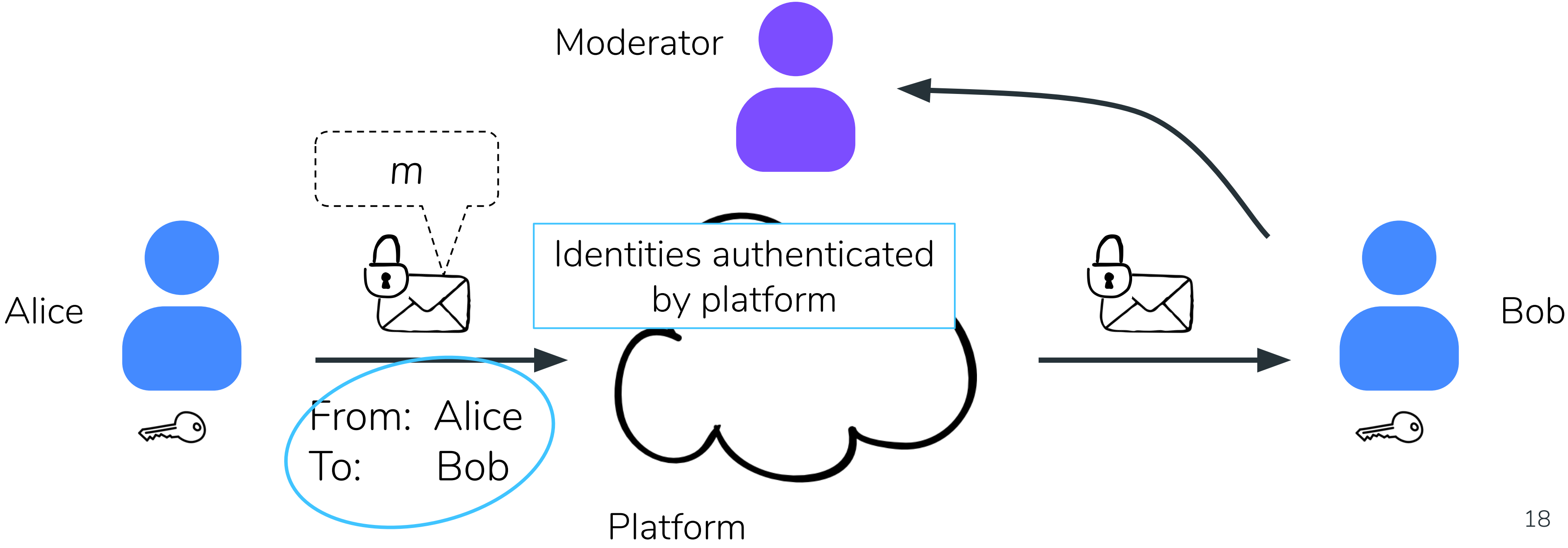
- Content-based moderation of encryption that is NOT metadata-private
- Compactly-committing authenticated encryption



Prior work on moderation in E2E encryption

Message franking [FB white paper ‘17], [GLR CRYPTO‘17], [DGRW CRYPTO‘18]

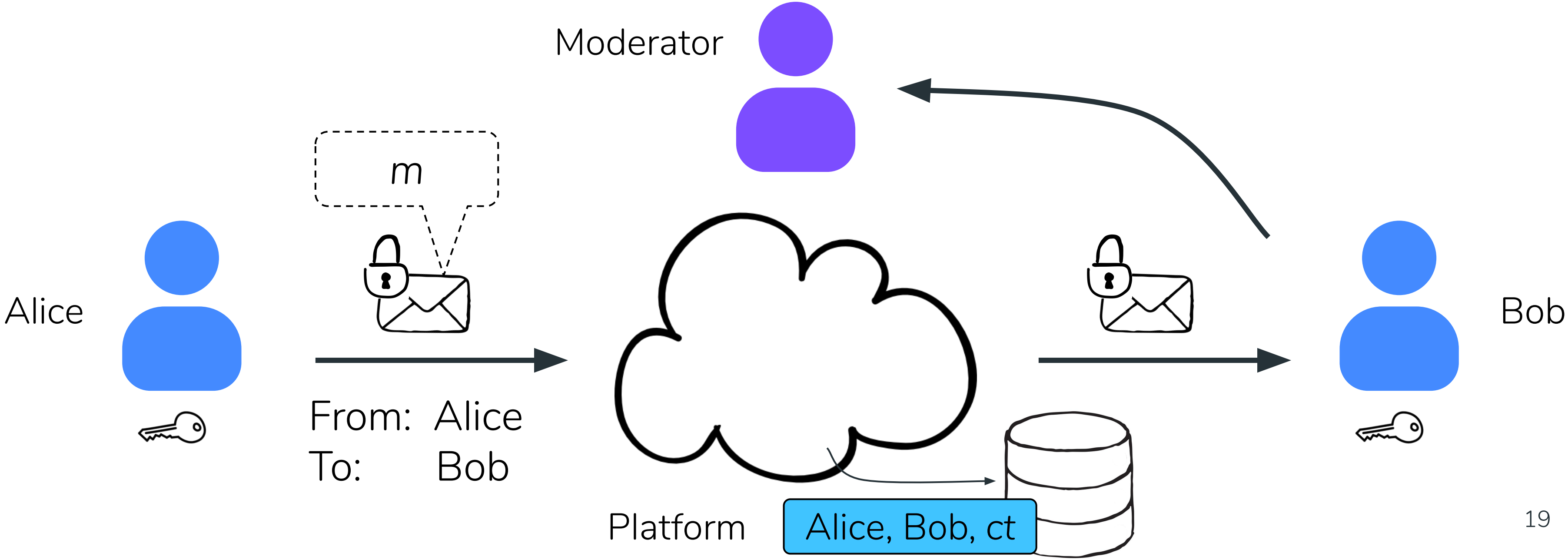
- Content-based moderation of encryption that is NOT metadata-private
- Compactly-committing authenticated encryption



Prior work on moderation in E2E encryption

Message franking [FB white paper '17], [GLR CRYPTO'17], [DGRW CRYPTO'18]

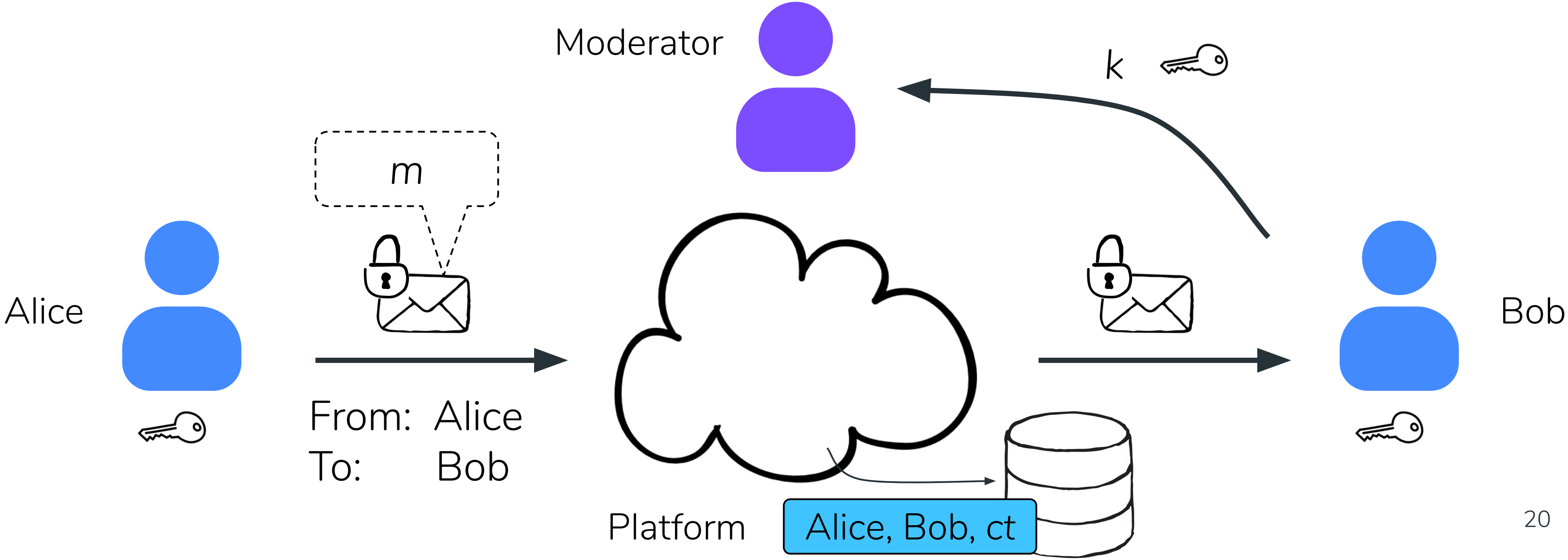
- Content-based moderation of encryption that is NOT metadata-private
- Compactly-committing authenticated encryption



Prior work on moderation in E2E encryption

Message franking [FB white paper '17], [GLR CRYPTO'17], [DGRW CRYPTO'18]

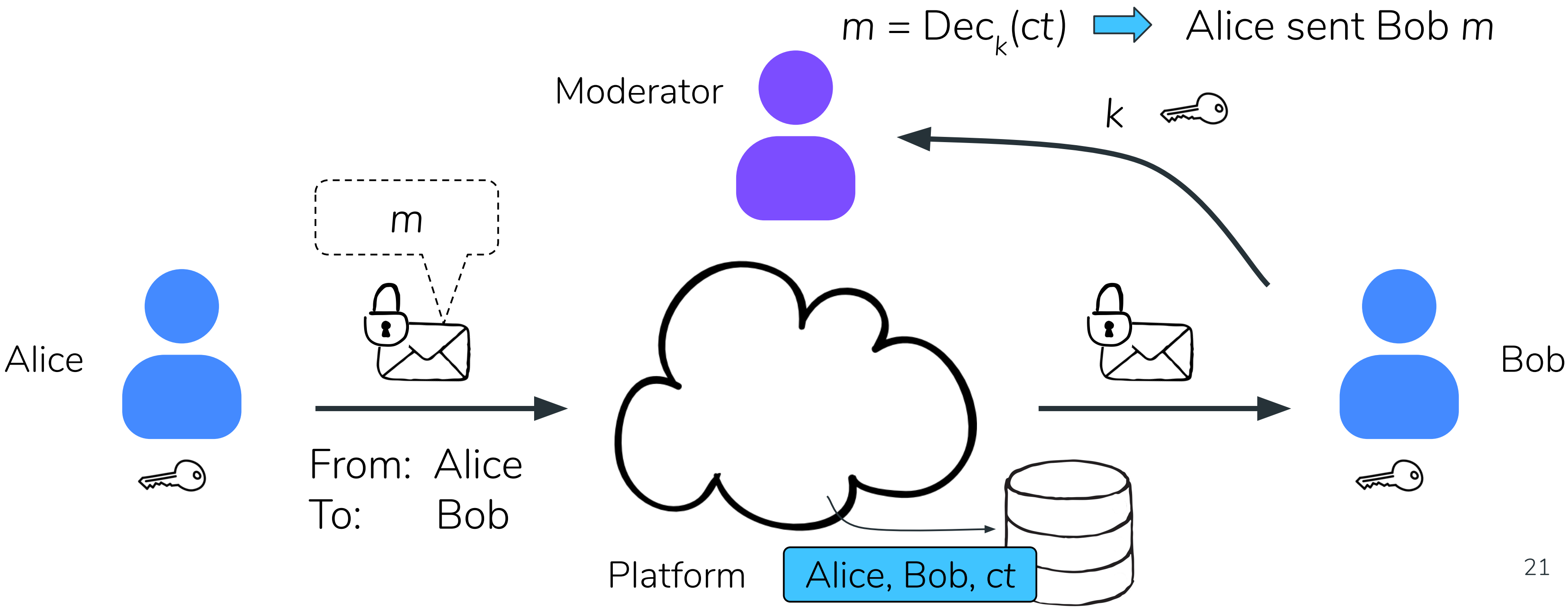
- Content-based moderation of encryption that is NOT metadata-private
- Compactly-committing authenticated encryption



Prior work on moderation in E2E encryption

Message franking [FB white paper ‘17], [GLR CRYPTO‘17], [DGRW CRYPTO‘18]

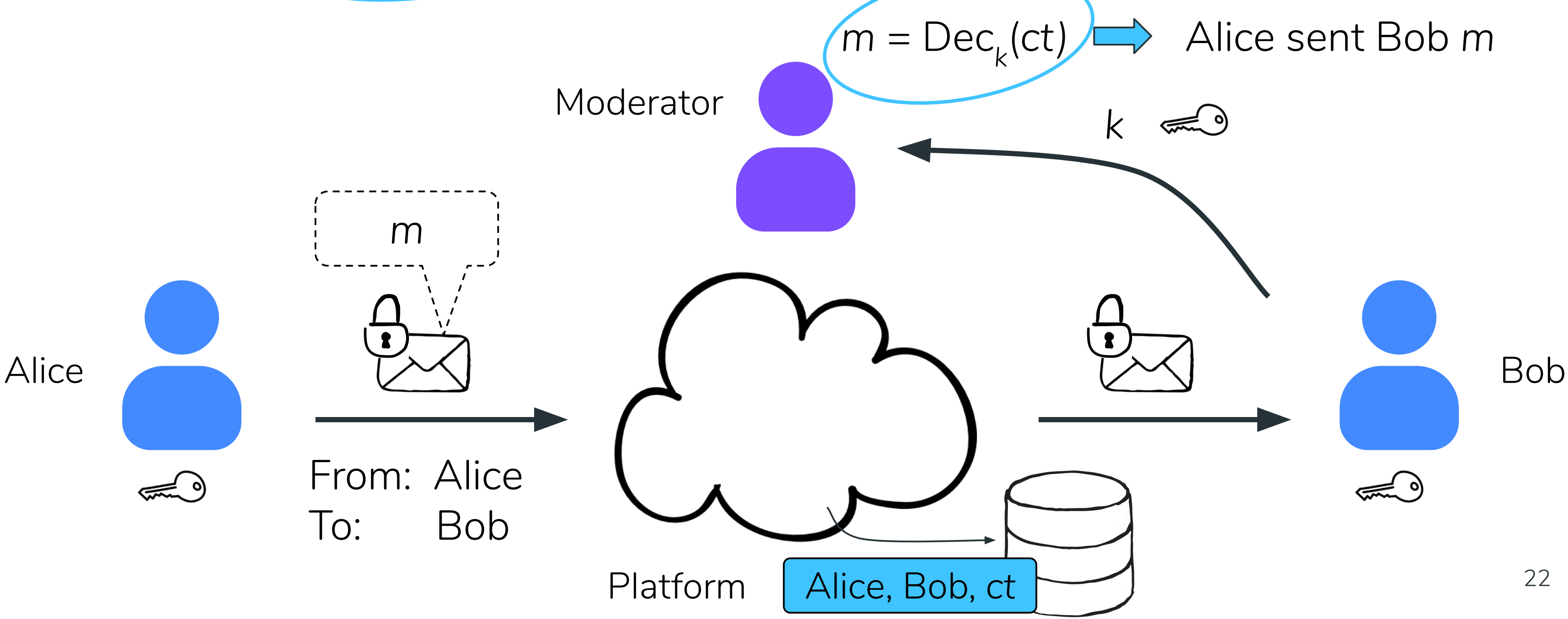
- Content-based moderation of encryption that is NOT metadata-private
- Compactly-committing authenticated encryption



Prior work on moderation in E2E encryption

Message franking [FB white paper ‘17], [GLR CRYPTO‘17], [DGRW CRYPTO‘18]

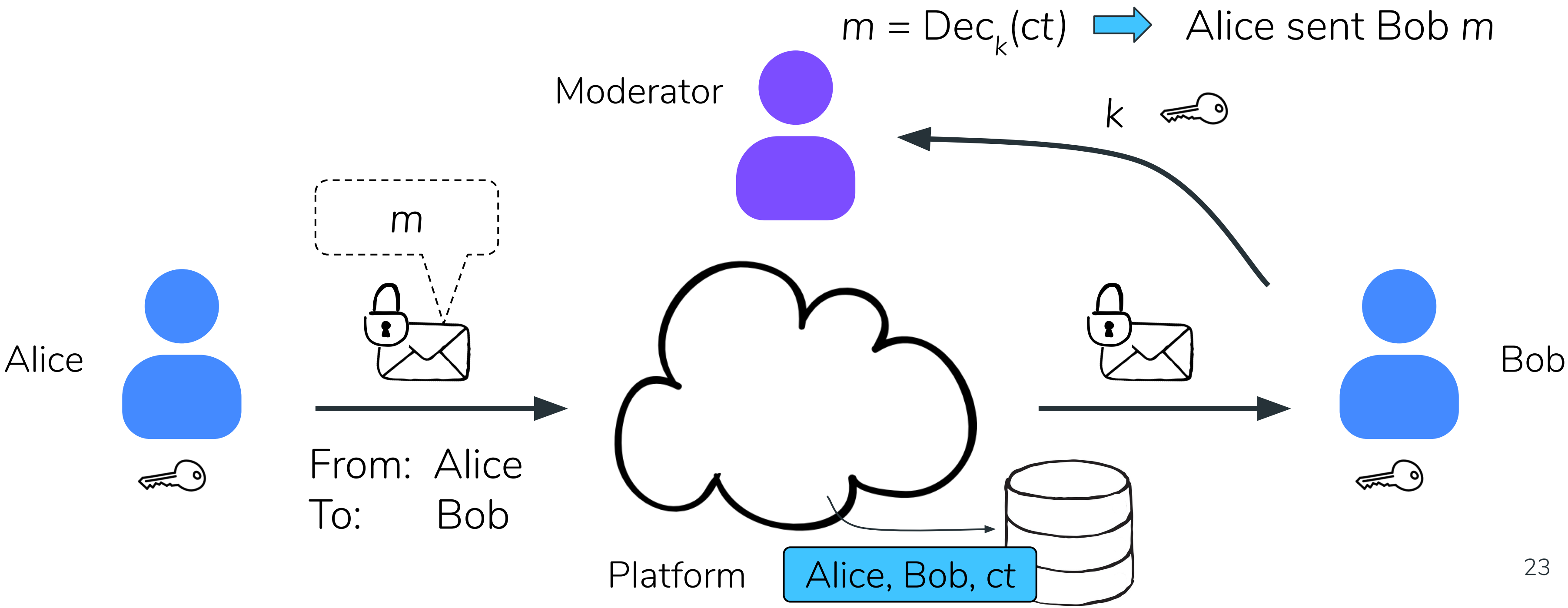
- Content-based moderation of encryption that is NOT metadata-private
- Compactly-committing authenticated encryption



Prior work on moderation in E2E encryption

Message franking [FB white paper '17], [GLR CRYPTO'17], [DGRW CRYPTO'18]

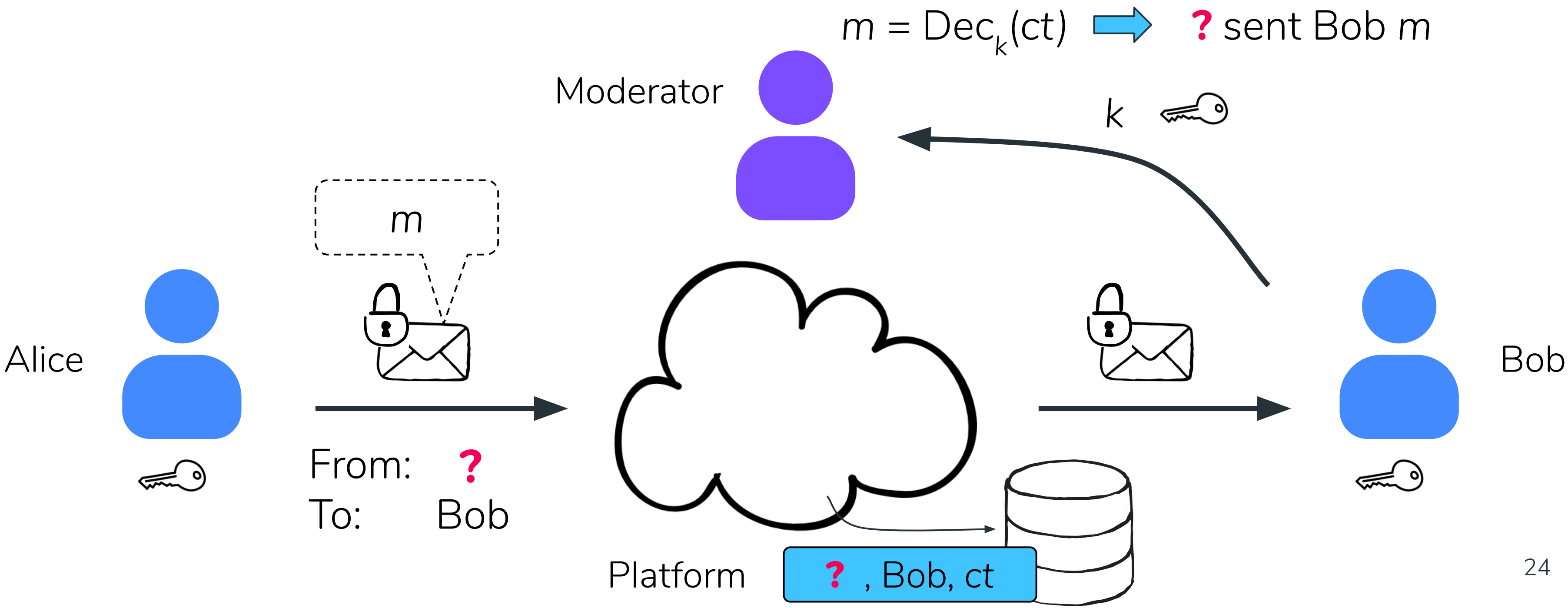
- Content-based moderation of encryption that is **NOT metadata-private**
- Compactly-committing authenticated encryption



Message franking for metadata-private setting?

Message franking [FB white paper ‘17], [GLR CRYPTO‘17], [DGRW CRYPTO‘18]

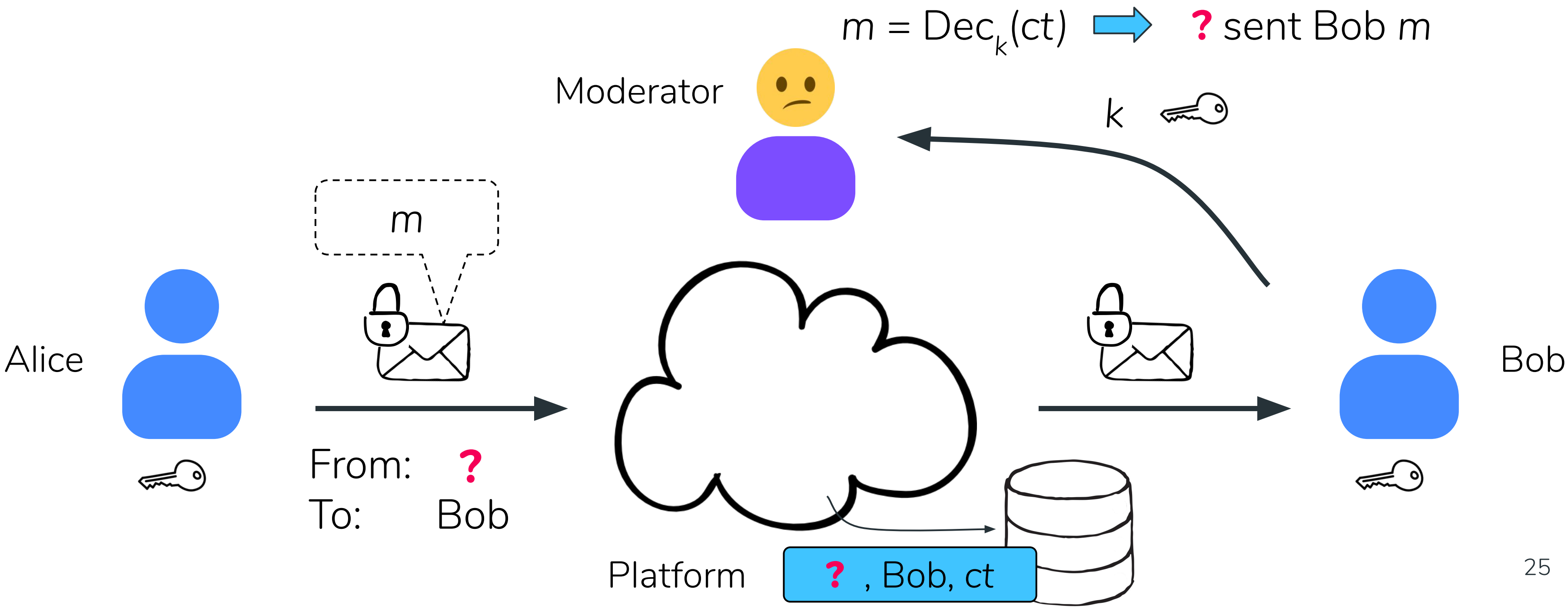
- Content-based moderation of encryption that is **NOT metadata-private**
- Compactly-committing authenticated encryption



Message franking for metadata-private setting?

Message franking [FB white paper '17], [GLR CRYPTO'17], [DGRW CRYPTO'18]

- Content-based moderation of encryption that is **NOT metadata-private**
- Compactly-committing authenticated encryption

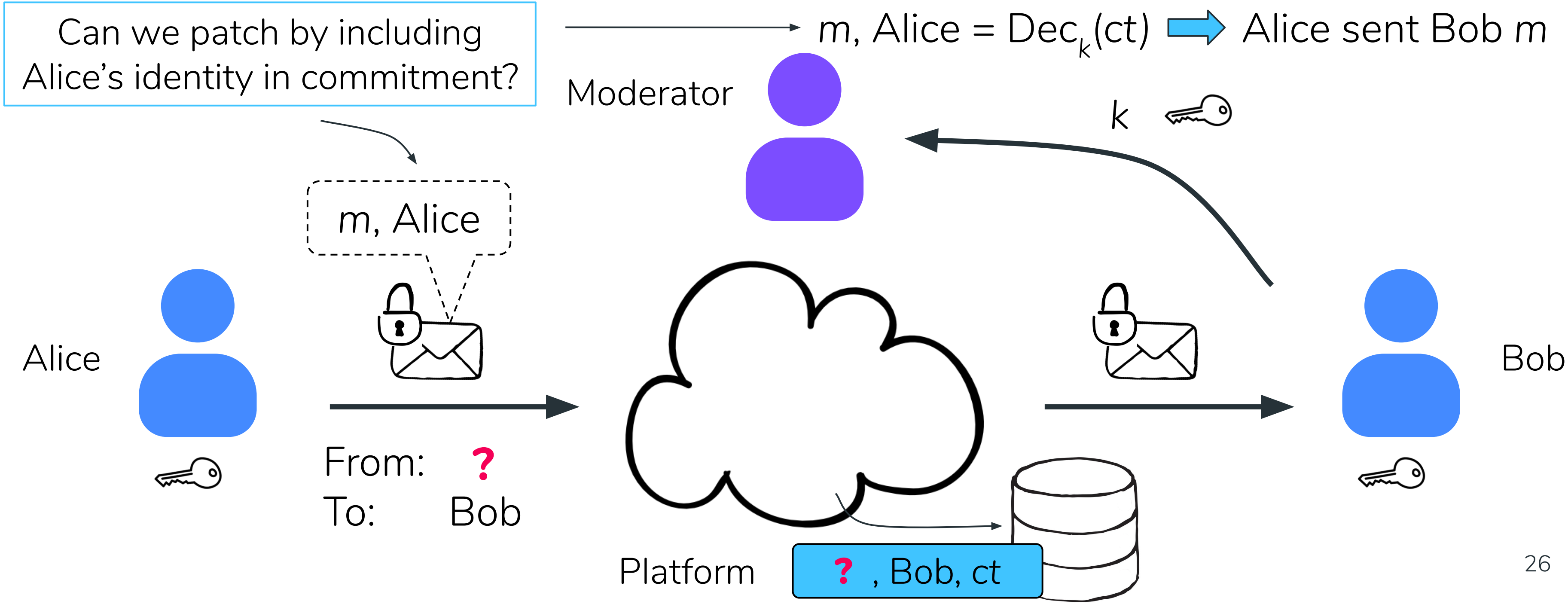


Message franking for metadata-private setting?

Message franking [FB white paper '17], [GLR CRYPTO'17], [DGRW CRYPTO'18]

- Content-based moderation of encryption that is **NOT metadata-private**
- Compactly-committing authenticated encryption

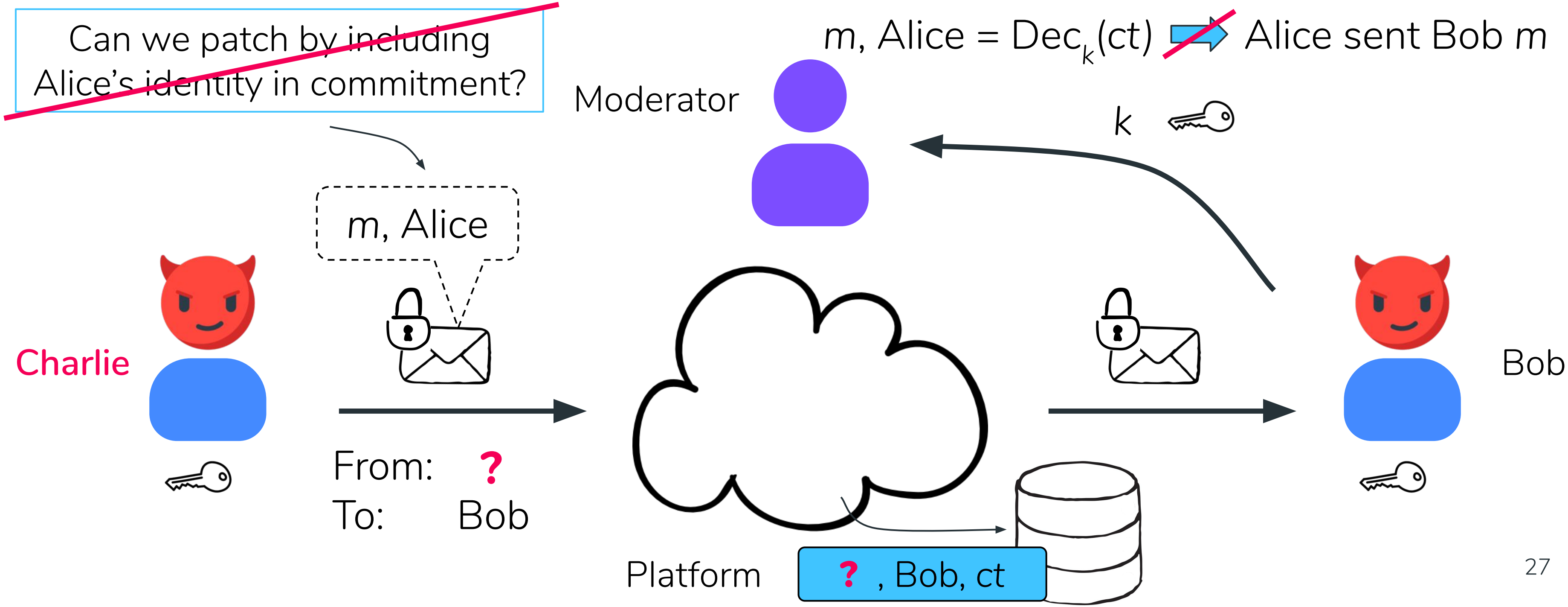
Can we patch by including Alice's identity in commitment?



Message franking for metadata-private setting?

Message franking [FB white paper ‘17], [GLR CRYPTO‘17], [DGRW CRYPTO‘18]

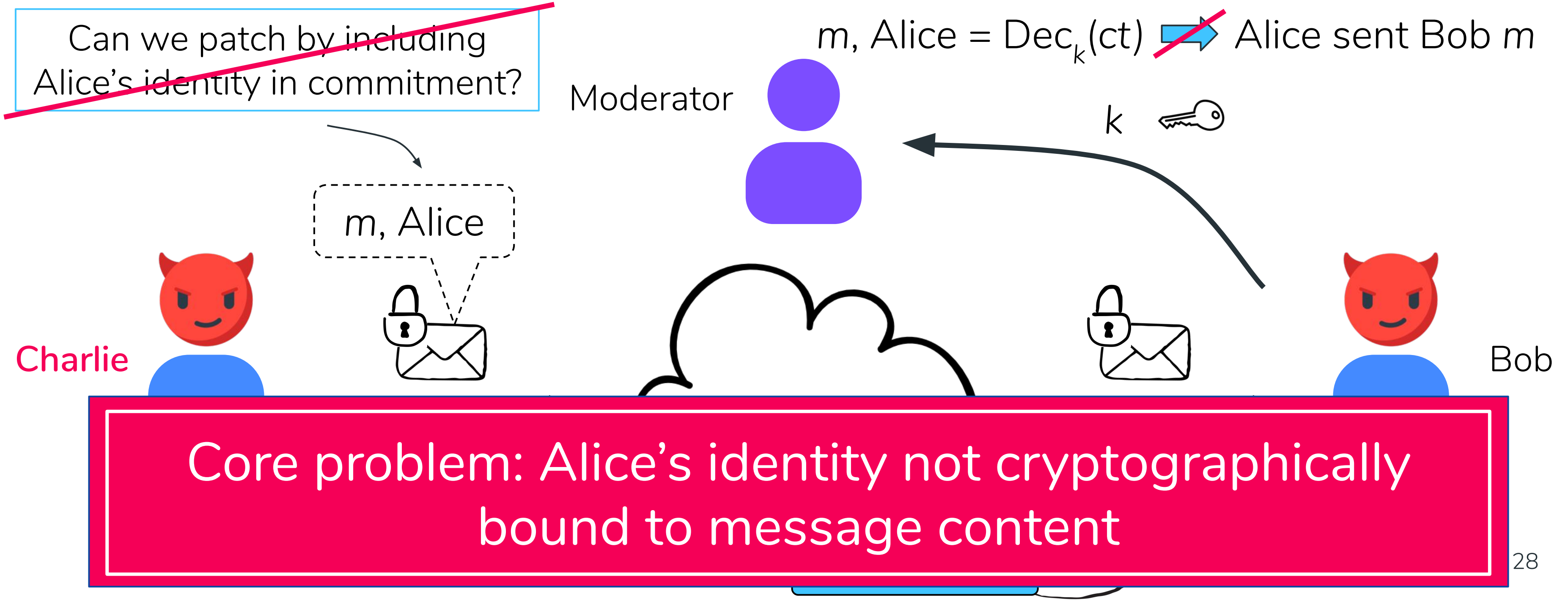
- Content-based moderation of encryption that is **NOT metadata-private**
- Compactly-committing authenticated encryption



Message franking for metadata-private setting?

Message franking [FB white paper ‘17], [GLR CRYPTO‘17], [DGRW CRYPTO‘18]

- Content-based moderation of encryption that is **NOT metadata-private**
- Compactly-committing authenticated encryption



AMFs: High level idea

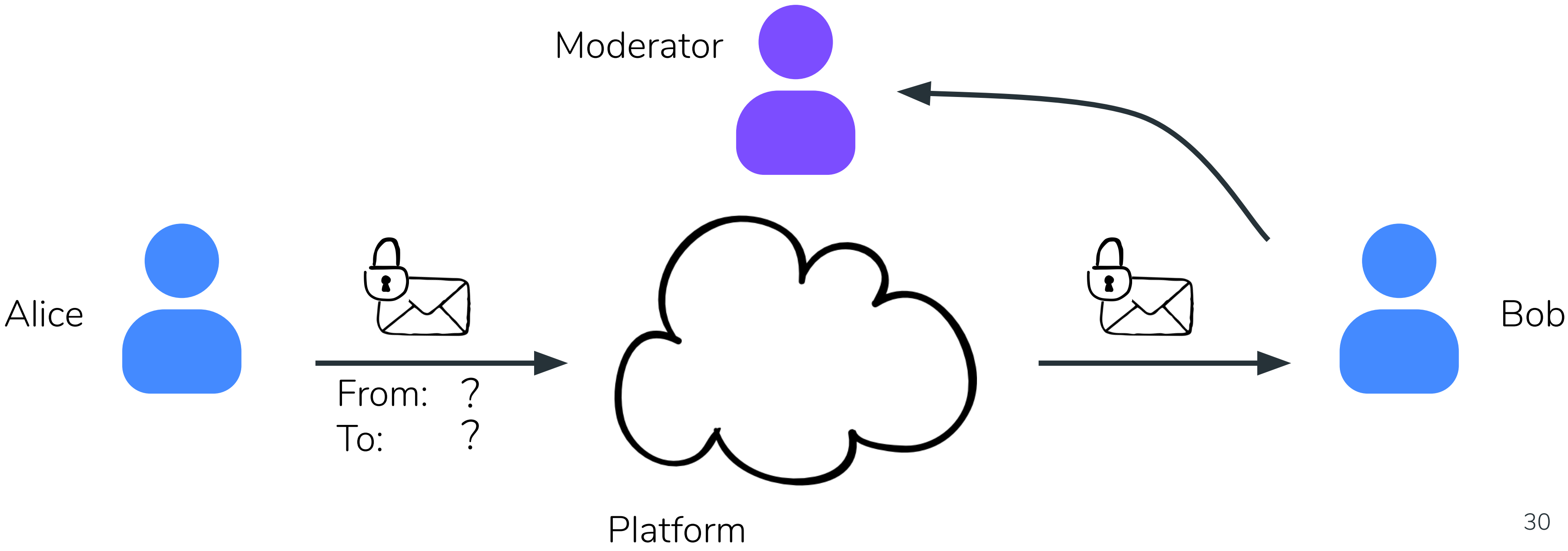
Specialized digital signature scheme that provides:

- Accountability
- Deniability

AMFs: High level idea

Specialized digital signature scheme that provides:

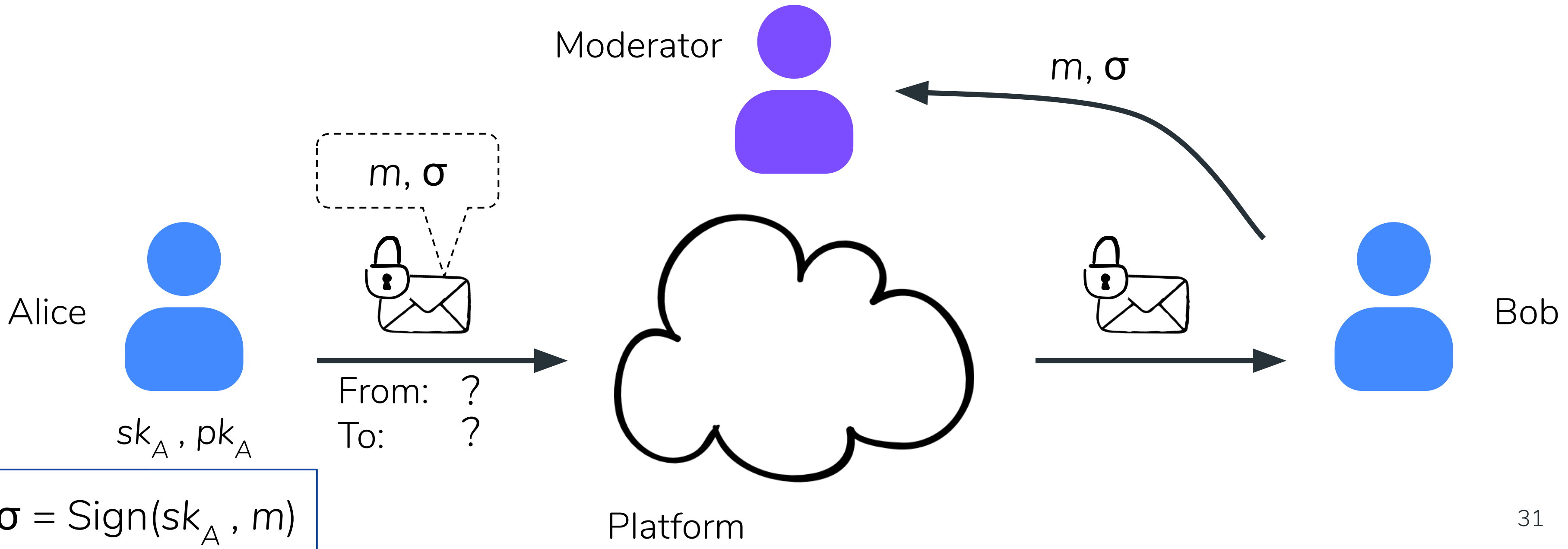
- Accountability
- Deniability



AMFs: High level idea

Specialized digital signature scheme that provides:

- Accountability
- Deniability



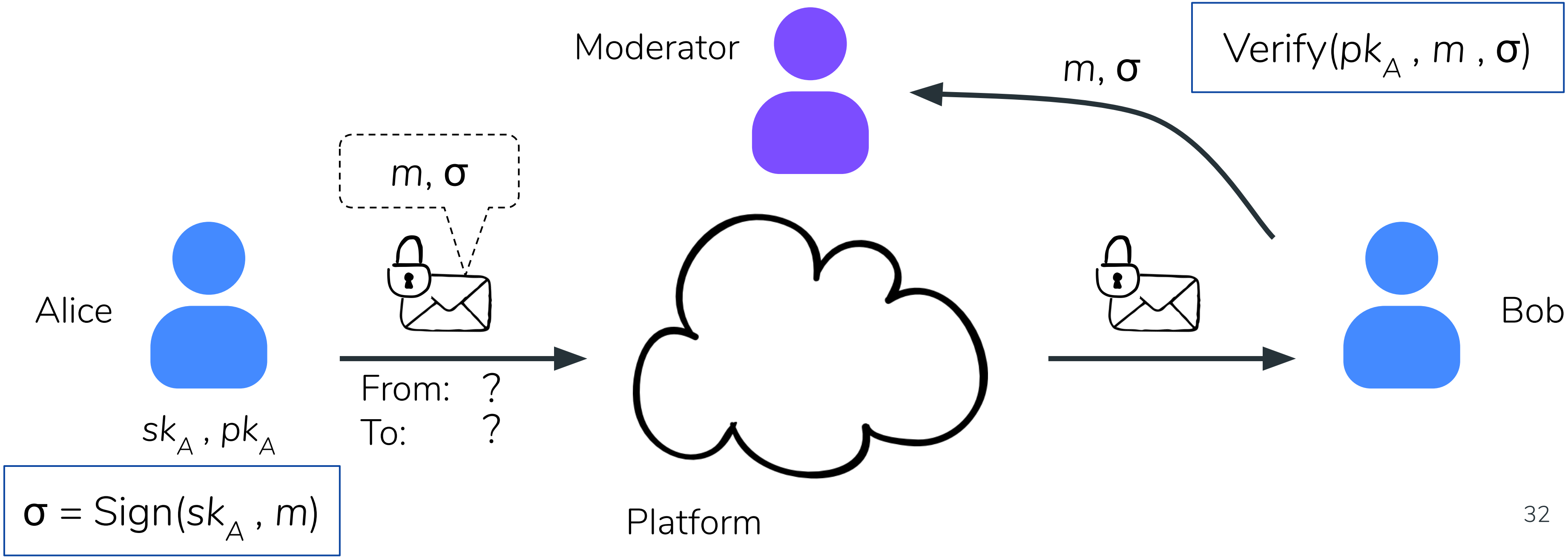
$\sigma = \mathrm{Sign}(sk_A, m)$

AMFs: High level idea

Specialized digital signature scheme that provides:

- Accountability
- Deniability

Standard digital signatures provide accountability



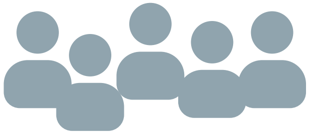
AMFs: High level idea

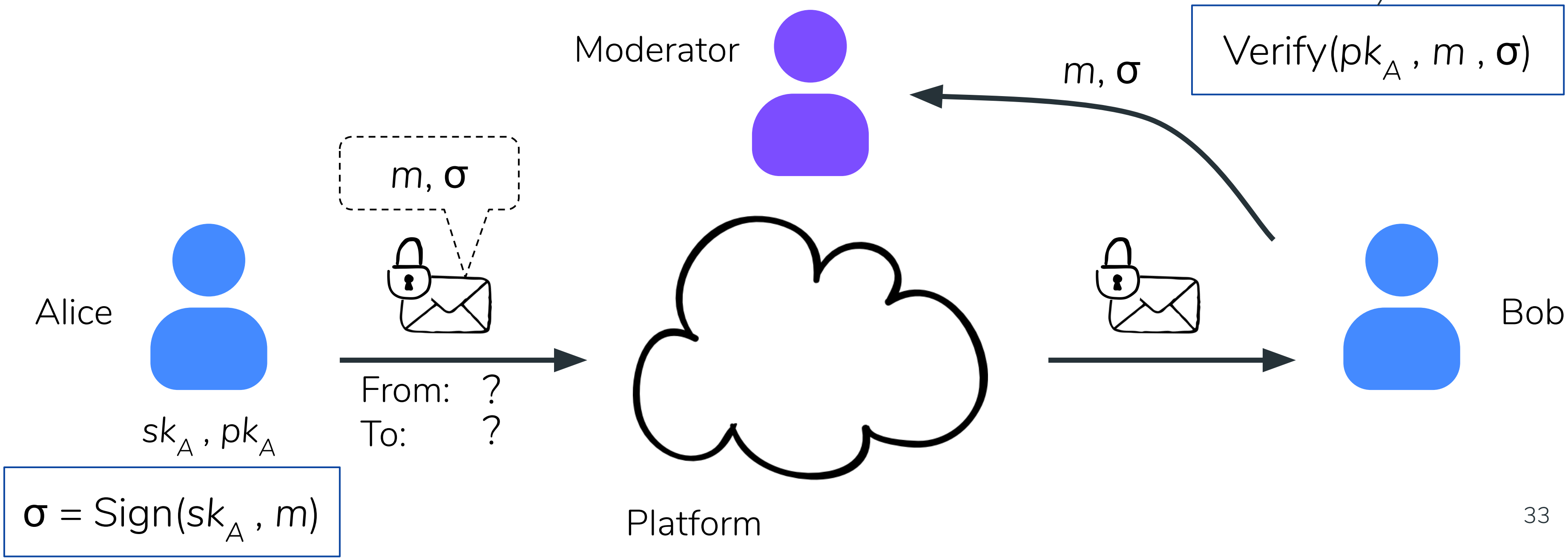
Specialized digital signature scheme that provides:

- Accountability
- Deniability

Standard digital signatures provide accountability ...but not deniability

"Public"





Starting point: Designated-verifier signatures

Digital signatures where only one party can verify [JSI EUROCRYPT ‘96]

Starting point: Designated-verifier signatures

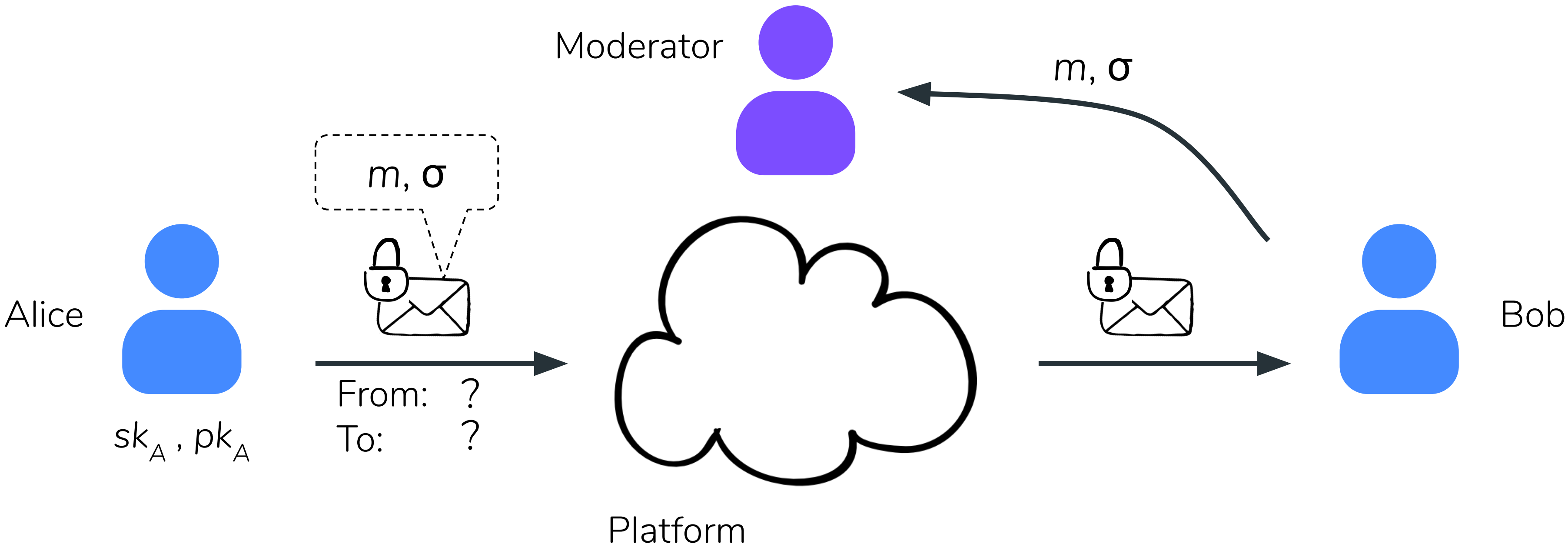
Digital signatures where only one party can verify [JSI EUROCRYPT '96]

- Accountability
  Designated verifier can't be fooled by forgery

- Deniability
  There exists forgery algorithm that fools everyone else

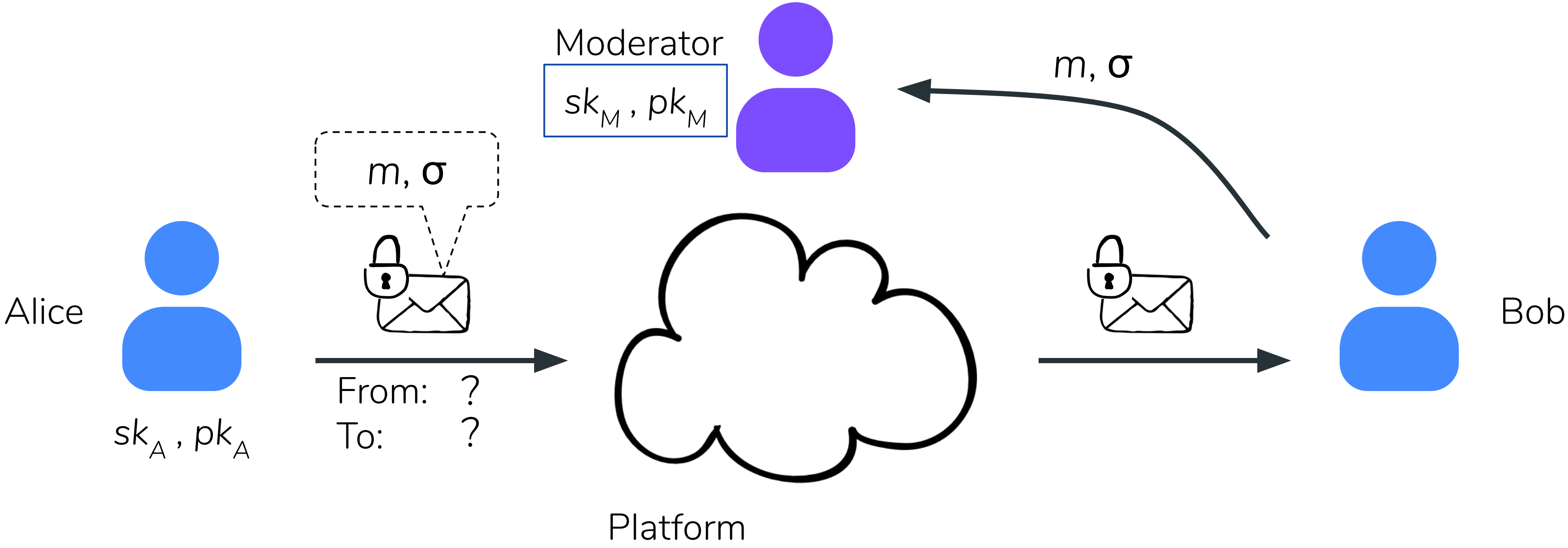
Starting point: Designated-verifier signatures

Idea: Designating the moderator as a verifier?



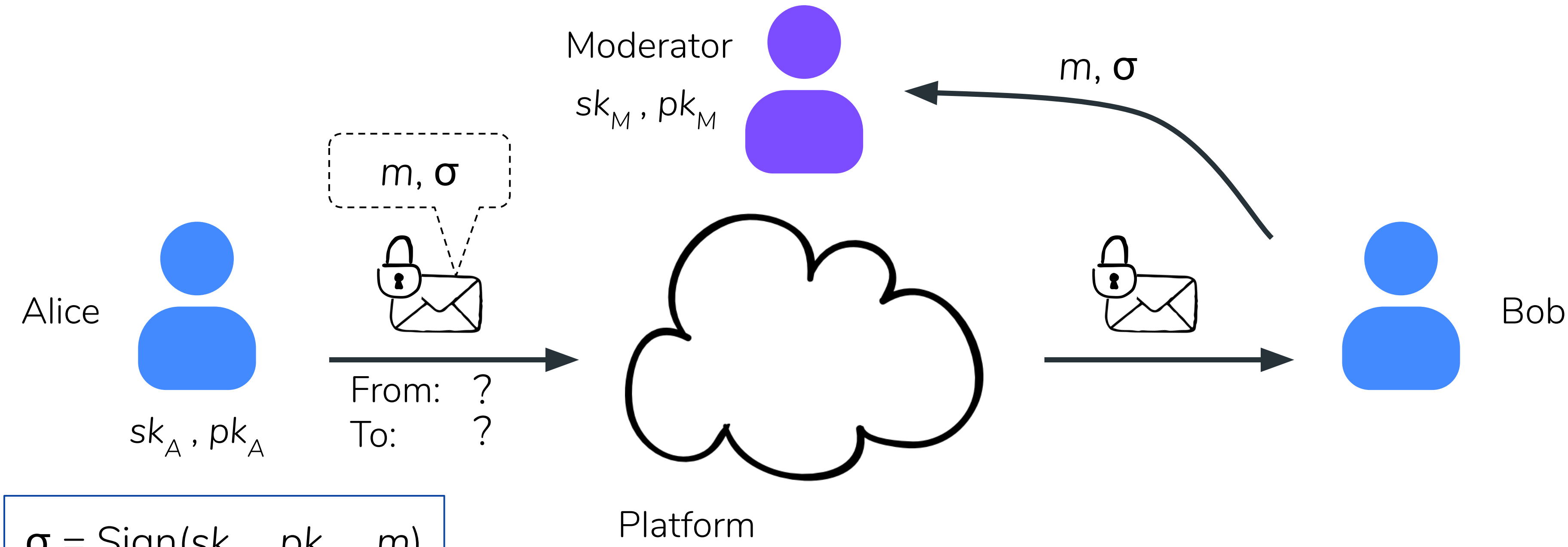
Starting point: Designated-verifier signatures

Idea: Designating the moderator as a verifier?



Starting point: Designated-verifier signatures

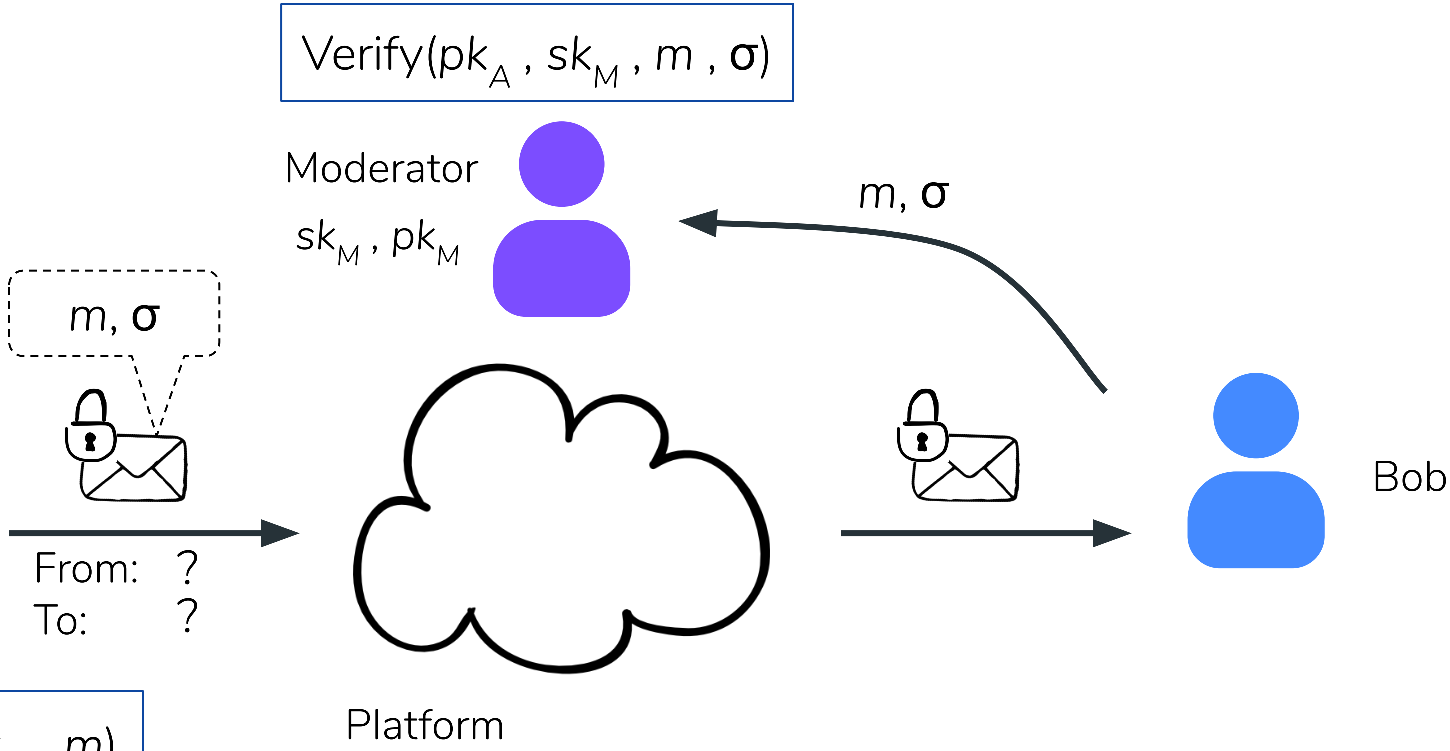
Idea: Designating the moderator as a verifier?



$$\sigma = \text{Sign}(sk_A , pk_M , m)$$

Starting point: Designated-verifier signatures

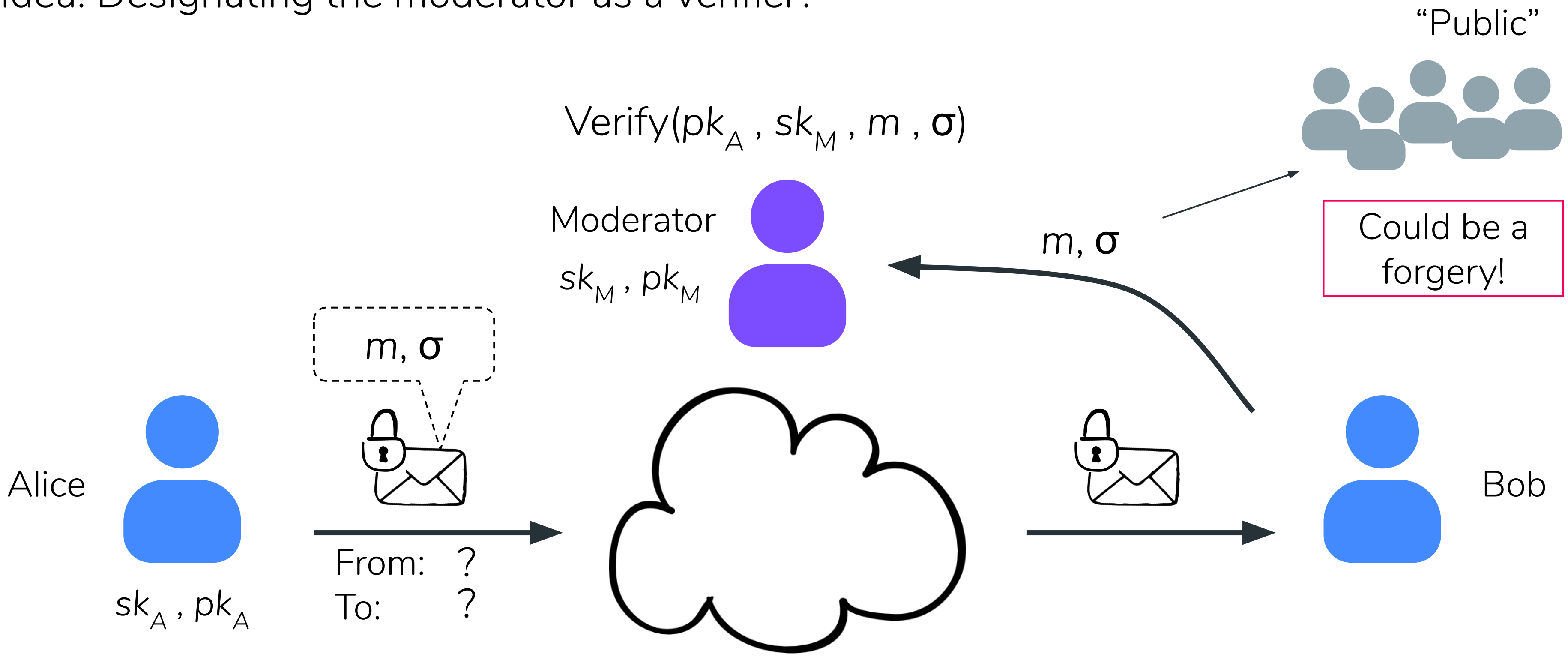
Idea: Designating the moderator as a verifier?



σ = Sign(sk_A , pk_M , m)

Starting point: Designated-verifier signatures

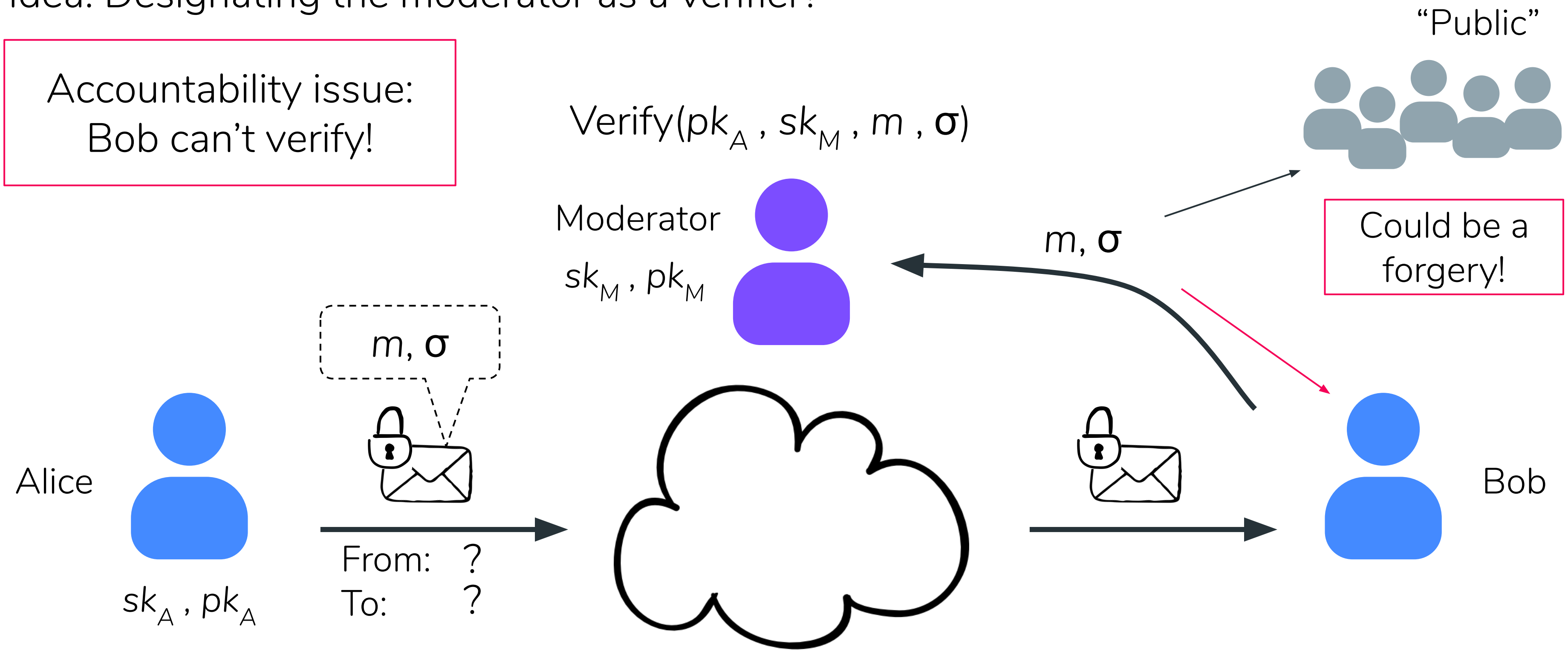
Idea: Designating the moderator as a verifier?



σ = Sign(sk_A , pk_M , m)

Starting point: Designated-verifier signatures

Idea: Designating the moderator as a verifier?



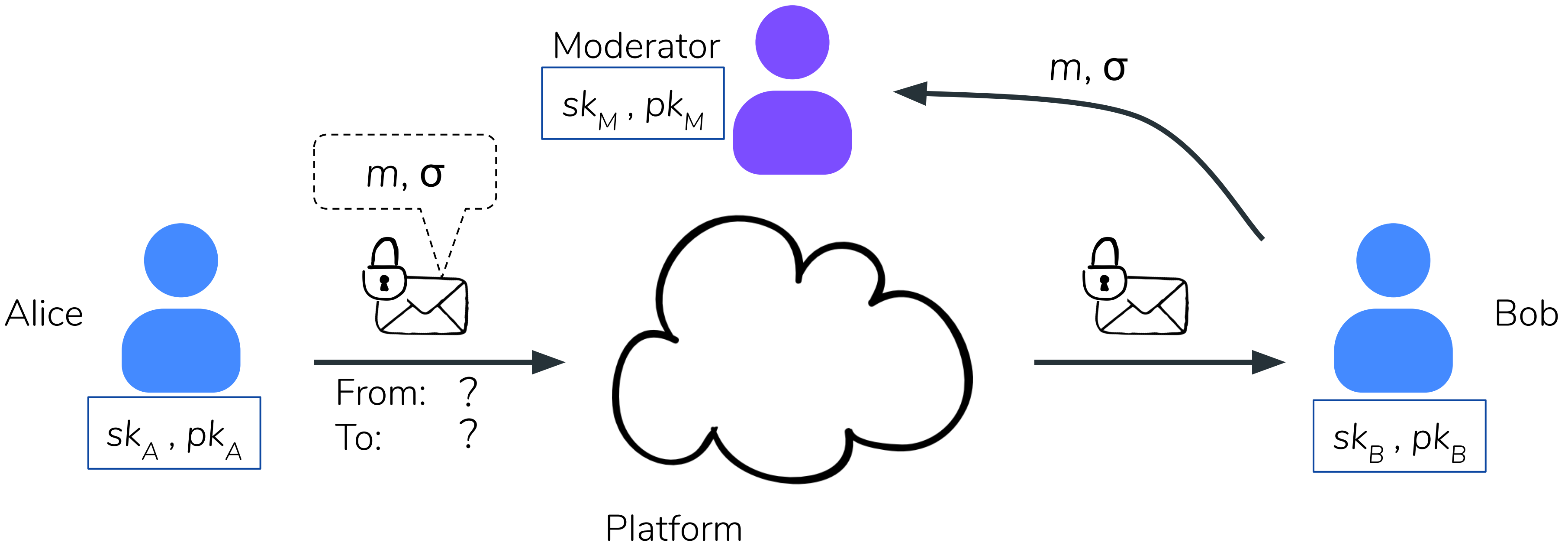
σ = Sign(sk_A , pk_M , m)

AMFs: Include recipient as verifying party

Solution: Designate Bob as verifier of proof that signature to moderator will succeed

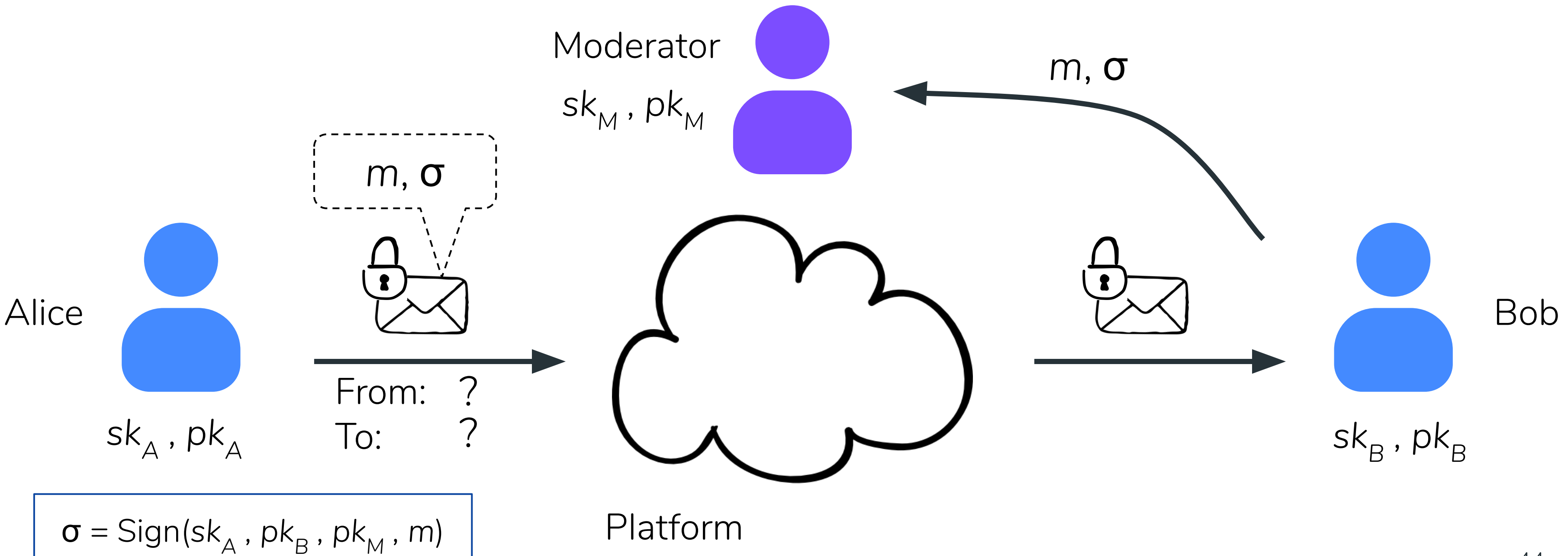
AMFs: Include recipient as verifying party

Solution: Designate Bob as verifier of proof that signature to moderator will succeed



AMFs: Include recipient as verifying party

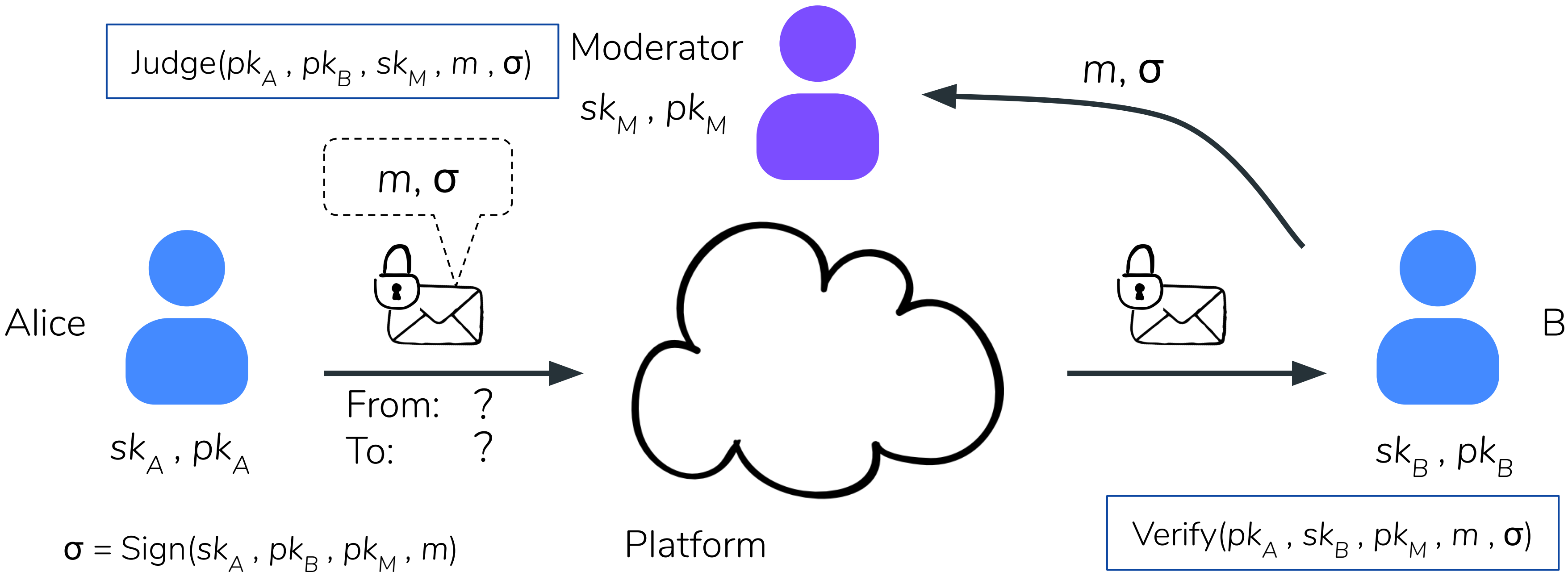
Solution: Designate Bob as verifier of proof that signature to moderator will succeed



σ = Sign(sk_A , pk_B , pk_M , m)

AMFs: Include recipient as verifying party

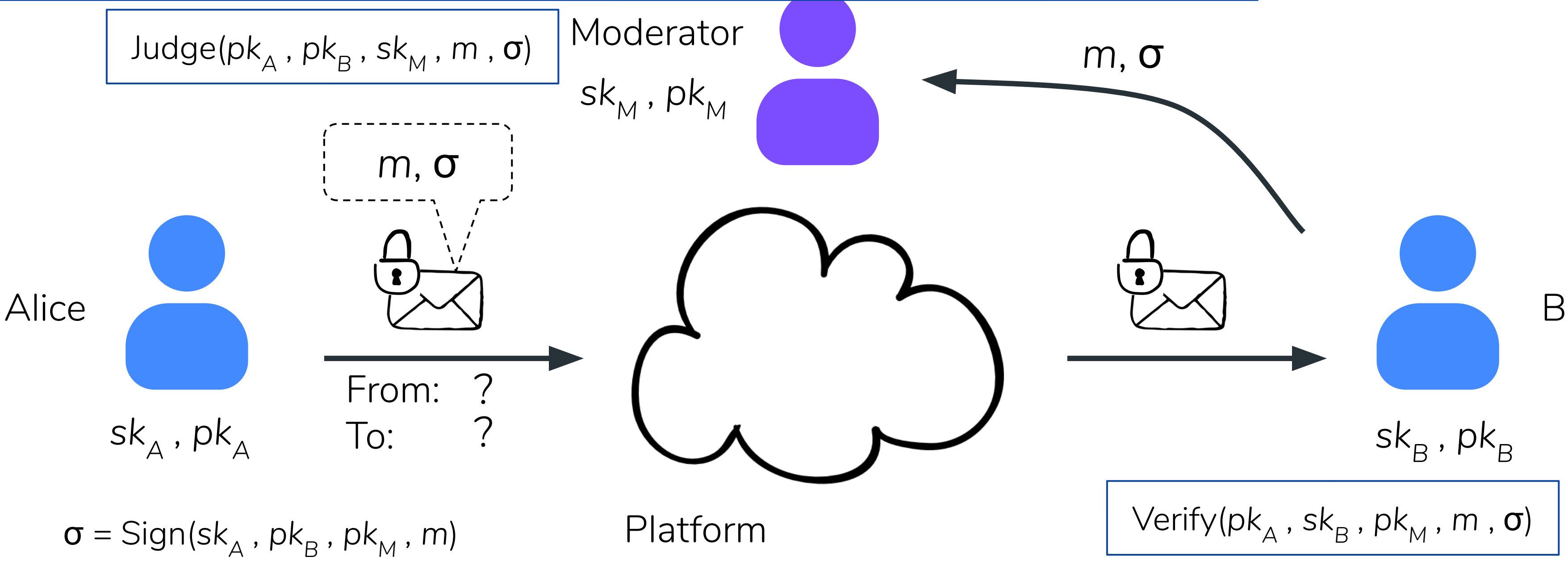
Solution: Designate Bob as verifier of proof that signature to moderator will succeed



AMFs: Include recipient as verifying party

Solution: Designate Bob as verifier of proof that signature to moderator will succeed

Accountability notions
- **Receiver binding**: Bob can't frame Alice for a message she did not send
- **Sender binding**: Alice can't send Bob a message that evades moderation



Deniability landscape: “Who can trick whom?”

Forger		**Distinguisher** D
$\boldsymbol{\sigma'} = \text{Forge}(pk_A, \boldsymbol{sk_B}, pk_M, m)$	$\sigma \approx_D \boldsymbol{\sigma'}$ →	pk_A, pk_B, pk_M

Deniability landscape: "Who can trick whom?"

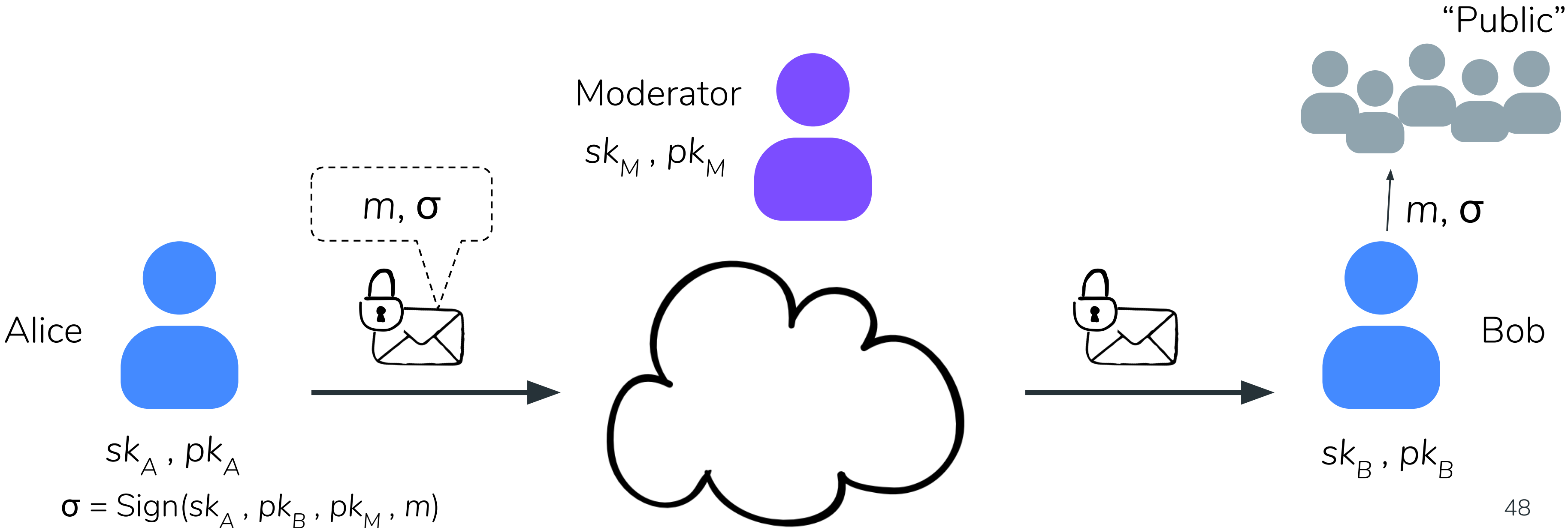
Forger

$\boldsymbol{\sigma'} = \text{Forge}(pk_A, \boldsymbol{sk_B}, pk_M, m)$

$\sigma \approx_D \boldsymbol{\sigma'}$

Distinguisher D

pk_A, pk_B, pk_M



Deniability landscape: "Who can trick whom?"

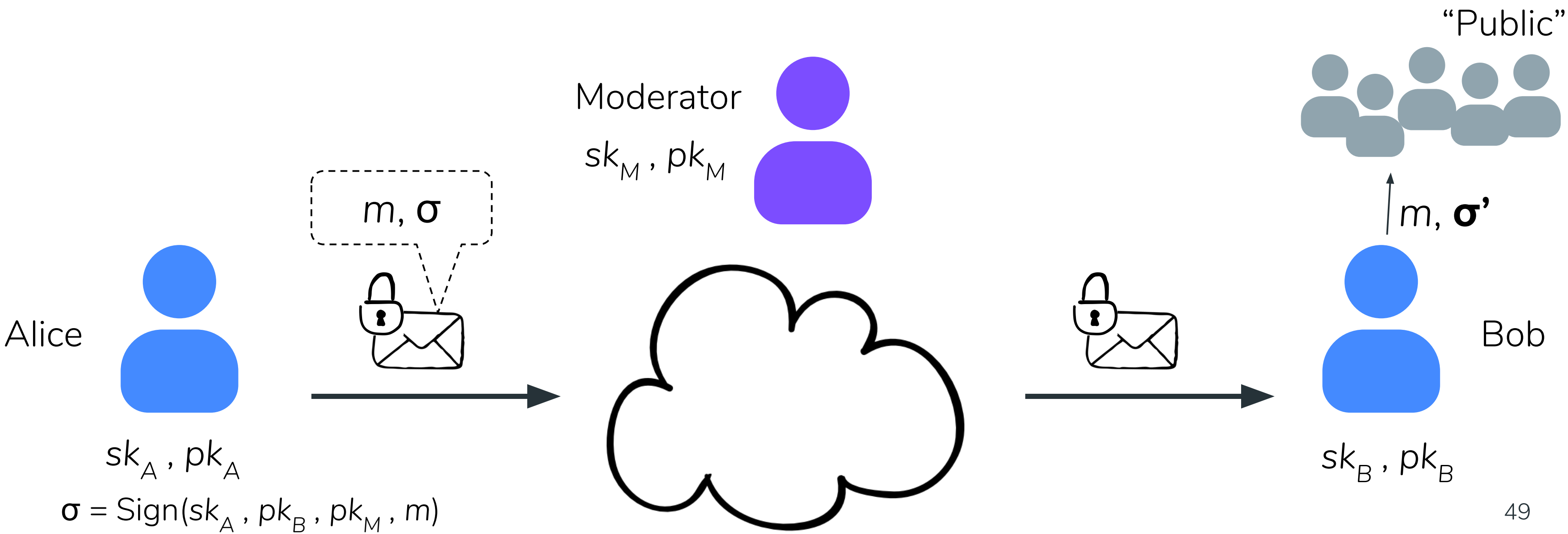
Forger

$\boldsymbol{\sigma'} = \text{Forge}(pk_A, \boldsymbol{sk_B}, pk_M, m)$

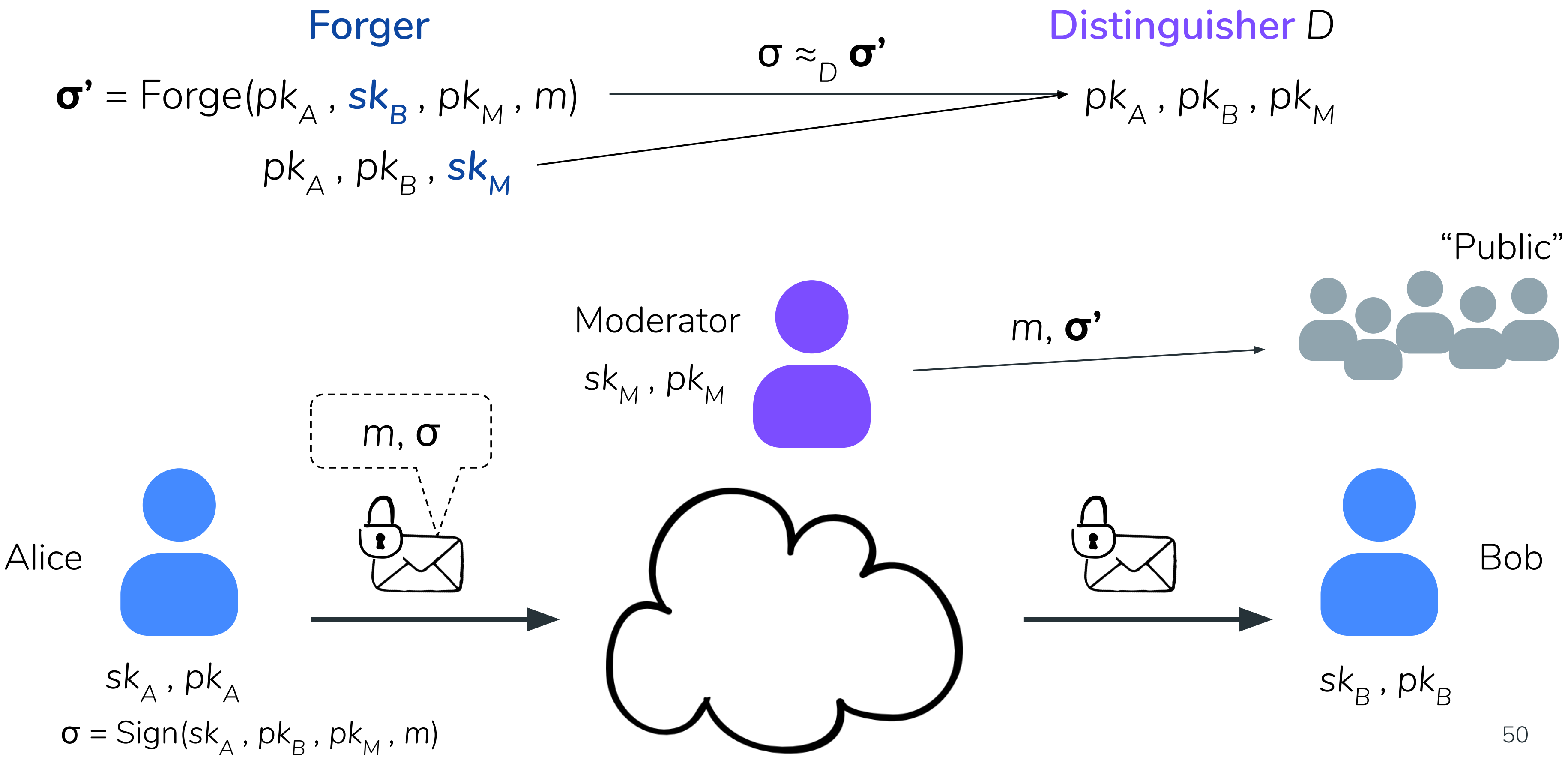
$\sigma \approx_D \boldsymbol{\sigma'}$

Distinguisher D

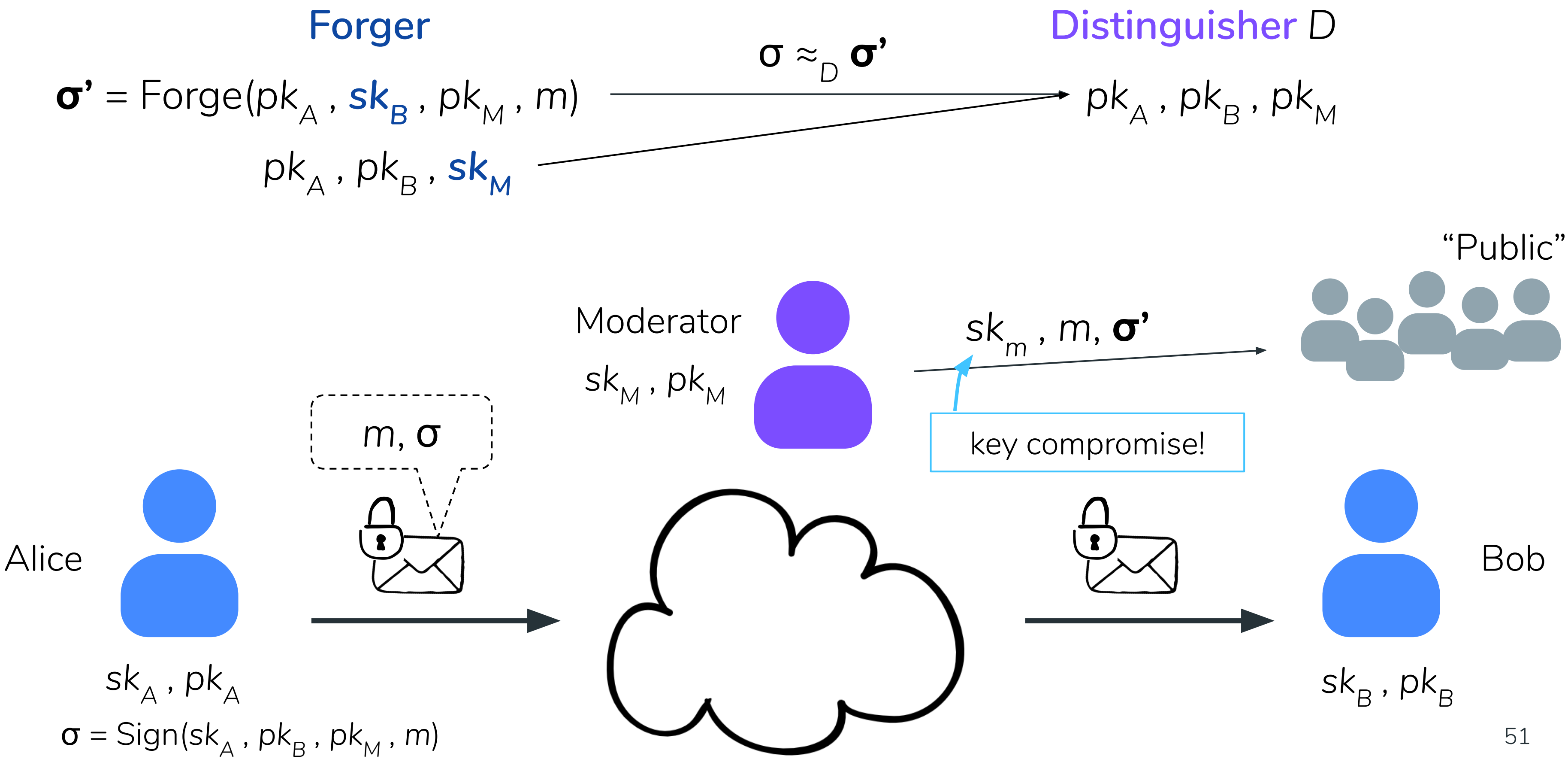
pk_A, pk_B, pk_M



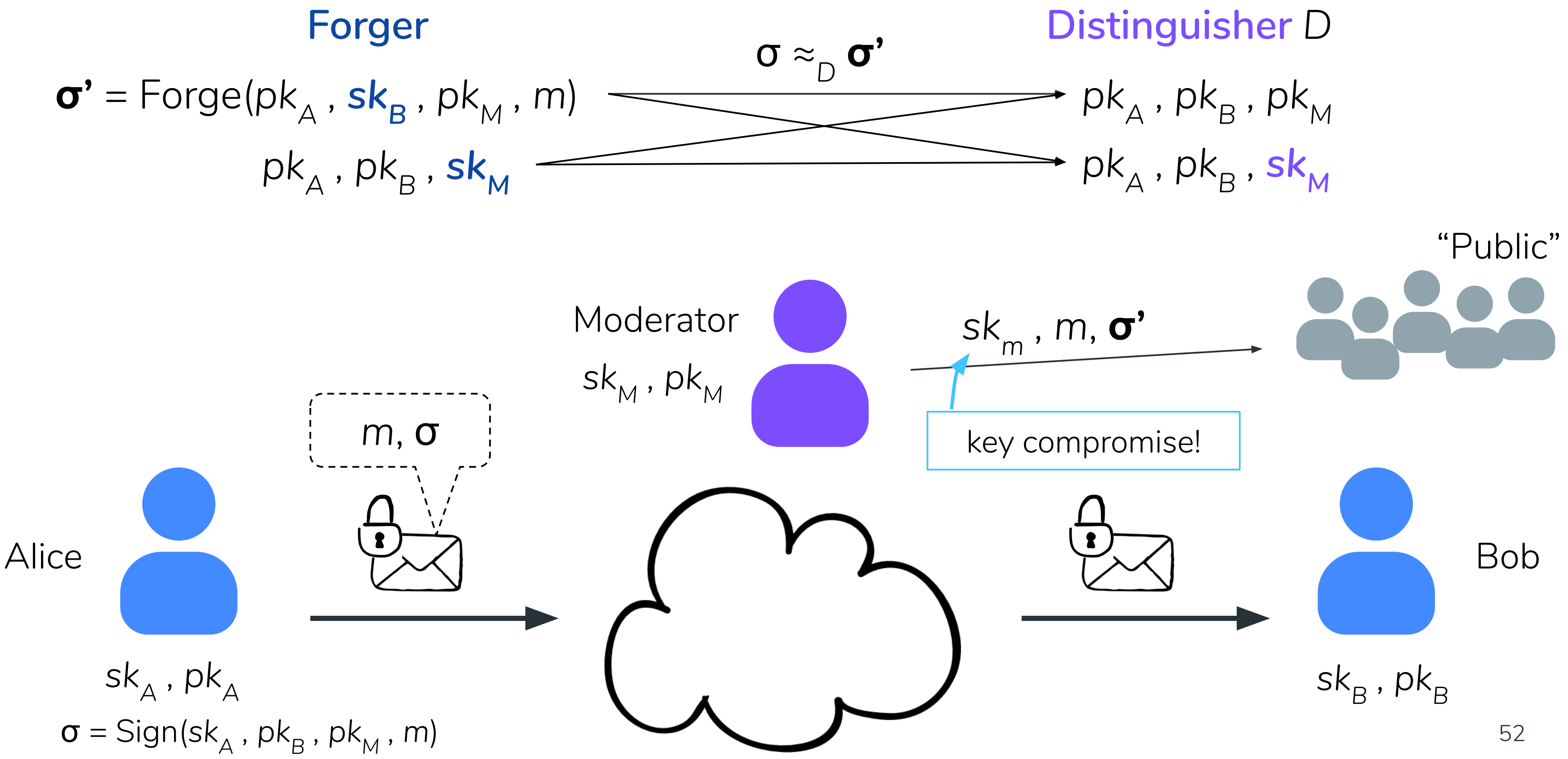
Deniability landscape: “Who can trick whom?”



Deniability landscape: "Who can trick whom?"



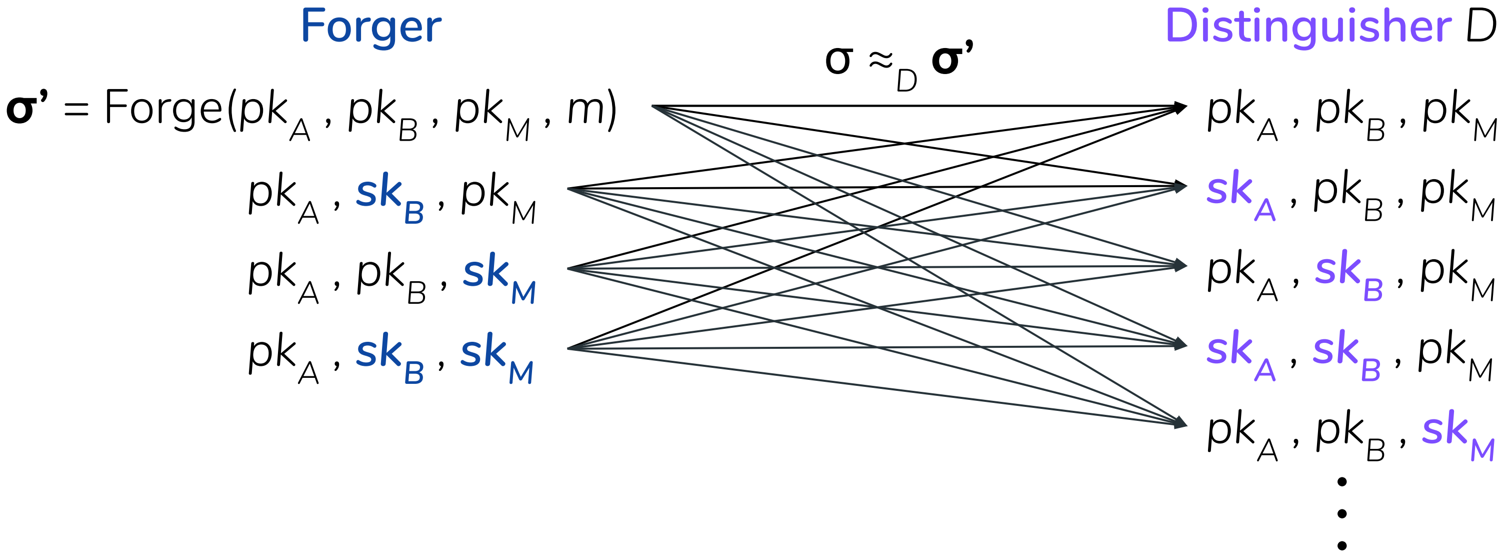
Deniability landscape: "Who can trick whom?"



Deniability landscape: “Who can trick whom?”

Forger		**Distinguisher** *D*
$\boldsymbol{\sigma'}$ = Forge(pk_A , pk_B , pk_M , m)	$\sigma \approx_D \boldsymbol{\sigma'}$ →	pk_A , pk_B , pk_M
pk_A , $\boldsymbol{sk_B}$, pk_M		$\boldsymbol{sk_A}$, pk_B , pk_M
pk_A , pk_B , $\boldsymbol{sk_M}$		pk_A , $\boldsymbol{sk_B}$, pk_M
pk_A , $\boldsymbol{sk_B}$, $\boldsymbol{sk_M}$		$\boldsymbol{sk_A}$, $\boldsymbol{sk_B}$, pk_M
		pk_A , pk_B , $\boldsymbol{sk_M}$
		⋮

Deniability landscape: "Who can trick whom?"



Deniability landscape: "Who can trick whom?"

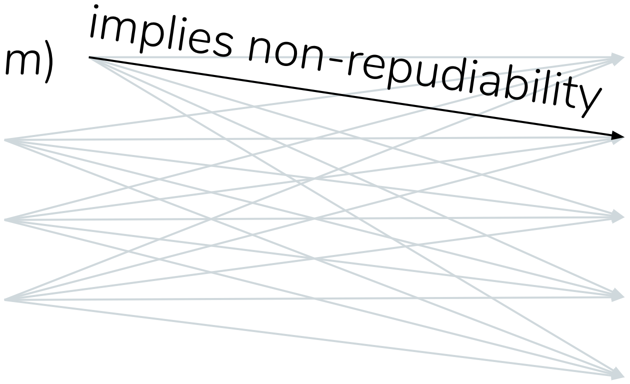
Forger

$\boldsymbol{\sigma}'$ = Forge(pk_A , pk_B , pk_M , m)

pk_A , $\mathbf{sk_B}$, pk_M

pk_A , pk_B , $\mathbf{sk_M}$

pk_A , $\mathbf{sk_B}$, $\mathbf{sk_M}$



Distinguisher *D*

pk_A , pk_B , pk_M

$\mathbf{sk_A}$, pk_B , pk_M

pk_A , $\mathbf{sk_B}$, pk_M

$\mathbf{sk_A}$, $\mathbf{sk_B}$, pk_M

pk_A , pk_B , $\mathbf{sk_M}$

⋮

Some deniability relationships are desirable

Deniability landscape: “Who can trick whom?”

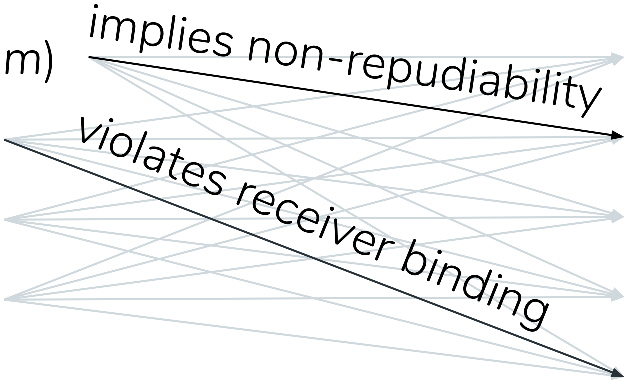
Forger

$\boldsymbol{\sigma}' = \text{Forge}(pk_A, pk_B, pk_M, m)$

$pk_A, \mathbf{sk_B}, pk_M$

$pk_A, pk_B, \mathbf{sk_M}$

$pk_A, \mathbf{sk_B}, \mathbf{sk_M}$



Distinguisher *D*

pk_A, pk_B, pk_M

$\mathbf{sk_A}, pk_B, pk_M$

$pk_A, \mathbf{sk_B}, pk_M$

$\mathbf{sk_A}, \mathbf{sk_B}, pk_M$

$pk_A, pk_B, \mathbf{sk_M}$

⋮

Some deniability relationships are desirable

Others contradict directly with accountability

Deniability landscape: “Who can trick whom?”

Distinguisher

Forger

sk_A			0	1	0	1	0	1	0	1
	sk_B		0	0	1	1	0	0	1	1
		sk_M	0	0	0	0	1	1	1	1
0	0	0			●	●	◆	◆	●◆	●◆
0	1	0					◆	◆	◆	◆
0	0	1								
0	1	1								

● : Incompatible with unforgeability

◆ : Incompatible with receiver binding

Deniability landscape: “Who can trick whom?”

Distinguisher

Forger sk_A	sk_B	sk_M								
sk_A			0	1	0	1	0	1	0	1
	sk_B		0	0	1	1	0	0	1	1
		sk_M	0	0	0	0	1	1	1	1
0	0	0		U	●	●	◆	◆	●◆	●◆
0	1	0				R	◆	◆	◆	◆
0	0	1								J
0	1	1								

U : Universal deniability
R : Receiver compromise deniability
J : Judge compromise deniability

● : Incompatible with unforgeability
◆ : Incompatible with receiver binding

Deniability landscape: “Who can trick whom?”

Distinguisher

Forger		sk_A	0	1	0	1	0	1	0	1
	sk_B		0	0	1	1	0	0	1	1
		sk_M	0	0	0	0	1	1	1	1
0	0	0		U	●	●	◆	◆	●◆	●◆
0	1	0				R	◆	◆	◆	◆
0	0	1								J
0	1	1								

U : Universal deniability
R : Receiver compromise deniability
J : Judge compromise deniability

● : Incompatible with unforgeability
◆ : Incompatible with receiver binding

This represents only one possible set of tradeoffs!

Summary of AMF goals

Specialized digital signature scheme that provides:

- Accountability
  - Receiver binding
  - Sender binding
- Deniability
  - Universal deniability
  - Receiver compromise deniability
  - Judge compromise deniability

Our Construction

- Proof of knowledge of carefully-crafted expression of discrete log relationships
- Create signature by adding message via Fiat-Shamir transform

Our Construction

- Proof of knowledge of carefully-crafted expression of discrete log relationships
- Create signature by adding message via Fiat-Shamir transform

Example of signature proof of knowledge (SPK) notation:

Standard digital signature (Schnorr)

$$\sigma \leftarrow_{\$} SPK\left\{t \,:\, pk_A = g^t\right\}$$

Our Construction

- Proof of knowledge of carefully-crafted expression of discrete log relationships
- Create signature by adding message via Fiat-Shamir transform

Example of signature proof of knowledge (SPK) notation:

Standard digital signature (Schnorr)

$$\sigma \leftarrow_{\$} SPK\left\{t \,:\, pk_A = g^t\right\}$$

∑-Protocol Proof of Knowledge

Prover → com → **Verifier**

Prover ← *chal* ← **Verifier**

Prover → *resp* → **Verifier**

SPK via Fiat-Shamir

Prover → com → **Verifier**

chal = H(com, *m*)

Prover → *resp* → **Verifier**

Our Construction

$$\alpha \leftarrow_{\$} \mathbb{Z}_p;\ E_J \leftarrow g^{\alpha};\ J \leftarrow \mathrm{pk}_M^{\alpha}$$
$$\beta \leftarrow_{\$} \mathbb{Z}_p;\ E_R \leftarrow g^{\beta};\ R \leftarrow \mathrm{pk}_B^{\beta}$$
$$\sigma \leftarrow_{\$} SPK\left\{t,u,v,w\ :\ \underbrace{\left(\mathrm{pk}_A = g^t \vee J = g^u\right)}_{\text{DV signature to moderator}} \wedge \left(\underbrace{(J = (\mathrm{pk}_M)^v \wedge E_J = g^v) \vee R = g^w}_{\text{DV proof to Bob}}\right)\right\}, E_J, E_R$$

Our Construction

$\alpha \leftarrow_\$ \mathbb{Z}_p;\ E_J \leftarrow g^{\alpha};\ J \leftarrow pk_M^{\alpha}$

$\beta \leftarrow_\$ \mathbb{Z}_p;\ E_R \leftarrow g^{\beta};\ R \leftarrow pk_B^{\beta}$

$\sigma \leftarrow_\$ SPK\left\{t, u, v, w\ :\ \underbrace{\left(pk_A = g^t \vee J = g^u\right)}_{\text{DV signature to moderator}} \wedge \underbrace{\left(\left(J = (pk_M)^v \wedge E_J = g^v\right) \vee R = g^w\right)}_{\text{DV proof to Bob}}\right\}, E_J, E_R$

Our Construction

$\alpha \leftarrow_\$ \mathbb{Z}_p;\ E_J \leftarrow g^\alpha;\ J \leftarrow pk_M^\alpha$

$\beta \leftarrow_\$ \mathbb{Z}_p;\ E_R \leftarrow g^\beta;\ R \leftarrow pk_B^\beta$

$\sigma \leftarrow_\$ SPK\left\{t, u, v, w\ :\ \left(pk_A = g^t \lor J = g^u\right) \land \left(\left(J = (pk_M)^v \land E_J = g^v\right) \lor R = g^w\right)\right\}, E_J, E_R$

“What Alice is proving to the moderator”

DV proof to Bob

DV signature to moderator

Our Construction

$$\alpha \leftarrow_{\$} \mathbb{Z}_p;\ E_J \leftarrow g^{\alpha};\ J \leftarrow pk_M^{\alpha}$$

$$\beta \leftarrow_{\$} \mathbb{Z}_p;\ E_R \leftarrow g^{\beta};\ R \leftarrow pk_B^{\beta}$$

$$\sigma \leftarrow_{\$} SPK\left\{t, u, v, w\ :\ \left(pk_A = g^t \vee J = g^u\right) \wedge \left(\left(J = (pk_M)^v \wedge E_J = g^v\right) \vee R = g^w\right)\right\}, E_J, E_R$$

“What Alice is proving to the moderator”

“What allows other parties to forge”

DV proof to Bob

DV signature to moderator

Our Construction

Moderator accepts if pk_M, E_J, J form a Diffie-Hellman triple

$$\alpha \leftarrow_\$ \mathbb{Z}_p;\ E_J \leftarrow g^\alpha;\ J \leftarrow pk_M^\alpha$$

$$\beta \leftarrow_\$ \mathbb{Z}_p;\ E_R \leftarrow g^\beta;\ R \leftarrow pk_B^\beta$$

$$\sigma \leftarrow_\$ SPK\left\{t,u,v,w\ :\ \left(pk_A = g^t \vee J = g^u\right) \wedge \left(\left(J = (pk_M)^v \wedge E_J = g^v\right) \vee R = g^w\right)\right\}, E_J, E_R$$

"What Alice is proving to the moderator"

"What allows other parties to forge"

DV proof to Bob

DV signature to moderator

Our Construction

Moderator accepts if pk_M, E_J, J form a Diffie-Hellman triple

$$\alpha \leftarrow_{\$} \mathbb{Z}_p;\ E_J \leftarrow g^{\alpha};\ J \leftarrow \mathrm{pk}_M^{\alpha}$$
$$\beta \leftarrow_{\$} \mathbb{Z}_p;\ E_R \leftarrow g^{\beta};\ R \leftarrow \mathrm{pk}_B^{\beta}$$
$$\sigma \leftarrow_{\$} SPK\left\{t,u,v,w\ :\ \left(\mathrm{pk}_A = g^t \vee J = g^u\right) \wedge \left((J = (\mathrm{pk}_M)^v \wedge E_J = g^v) \vee R = g^w\right)\right\}, E_J, E_R$$

DV signature to moderator

DV proof to Bob

Our Construction

Moderator accepts if pk_M, E_J, J form a Diffie-Hellman triple

$$\alpha \leftarrow_{\$} \mathbb{Z}_p;\ E_J \leftarrow g^{\alpha};\ J \leftarrow \mathrm{pk}_M^{\alpha}$$
$$\beta \leftarrow_{\$} \mathbb{Z}_p;\ E_R \leftarrow g^{\beta};\ R \leftarrow \mathrm{pk}_B^{\beta}$$
$$\sigma \leftarrow_{\$} SPK\left\{t, u, v, w\ :\ \left(\mathrm{pk}_A = g^t \vee J = g^u\right) \wedge \left(\left(J = (\mathrm{pk}_M)^v \wedge E_J = g^v\right) \vee R = g^w\right)\right\}, E_J, E_R$$

DV signature to moderator

"What Alice is proving to the recipient"

"What allows other parties to forge"

Our Construction

$$\alpha \leftarrow_\$ \mathbb{Z}_p;\ E_J \leftarrow g^\alpha;\ J \leftarrow pk_M^\alpha$$

$$\beta \leftarrow_\$ \mathbb{Z}_p;\ E_R \leftarrow g^\beta;\ R \leftarrow pk_B^\beta$$

$$\sigma \leftarrow_\$ SPK\left\{t, u, v, w\ :\ \left(pk_A = g^t \vee J = g^u\right) \wedge \left(\left(J = (pk_M)^v \wedge E_J = g^v\right) \vee R = g^w\right)\right\},\ E_J, E_R$$

Moderator accepts if pk_M, E_J, J form a Diffie-Hellman triple

DV signature to moderator

Alice is proving Diffie-Hellman relationship to Bob!

DV proof to Bob

Our Construction

Moderator accepts if pk_M, E_J, J form a Diffie-Hellman triple

$$\alpha \leftarrow_{\$} \mathbb{Z}_p;\; E_J \leftarrow g^{\alpha};\; J \leftarrow pk_M^{\alpha}$$
$$\beta \leftarrow_{\$} \mathbb{Z}_p;\; E_R \leftarrow g^{\beta};\; R \leftarrow pk_B^{\beta}$$
$$\sigma \leftarrow_{\$} SPK\left\{t,u,v,w \;:\; \left(pk_A = g^t \vee J = g^u\right) \wedge \left(\left(J = (pk_M)^v \wedge E_J = g^v\right) \vee R = g^w\right)\right\}, E_J, E_R$$

DV signature to moderator

Alice is proving Diffie-Hellman relationship to Bob!

DV proof to Bob

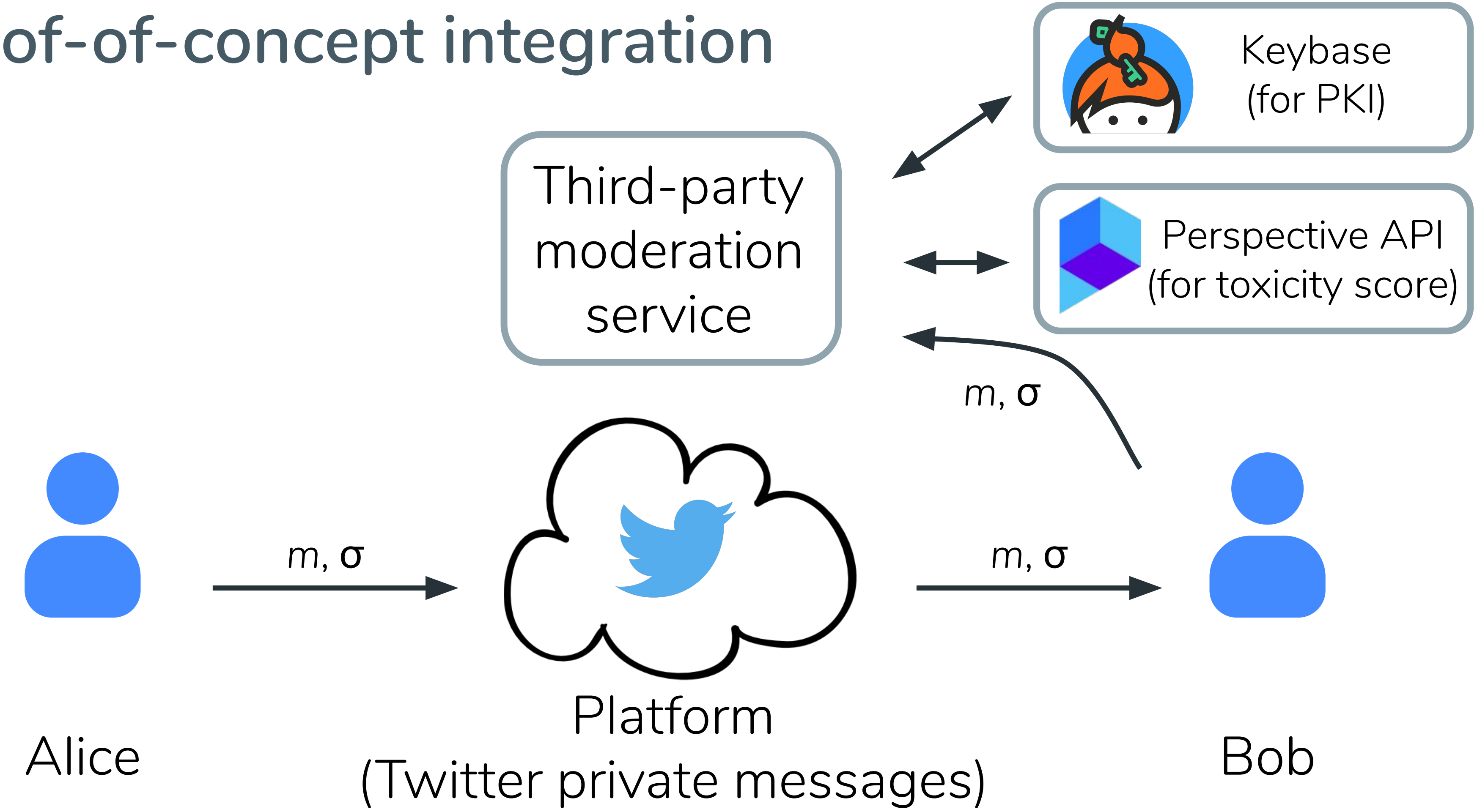
Accountability
- Moderator can attribute signature to sender
- Recipient can verify moderator will accept signature

Deniability
- Signature supports multiple forgery algorithms for various key compromise scenarios

Implementation

- Implemented in Python 3 using petlib (OpenSSL bindings)
- Fast and efficient
    - < 500 bytes for P-256 (9 group elements + 6 scalars)
    - < 10 ms for P-256
- Available at github.com/julialen/asymmetric-message-franking

Proof-of-concept integration



Available at github.com/julialen/asymmetric-message-franking

Our contributions

- Asymmetric Message Franking (AMF)
  - new cryptographic primitive for content moderation of metadata-private messaging
  - formal accountability and deniability security notions for content moderation
- Construction based on “designated-verifier” signatures
- Implementation and proof-of-concept integration
  - Available at github.com/julialen/asymmetric-message-franking