Impact Tracing: Identifying the Culprit of Misinformation in Encrypted Messaging Systems

Zhongming Wang, Tao Xiang, Xiaoguo Li, Biwen Chen, Guomin Yang, Chuan Ma, and Robert H. Deng

NDSS Symposium 2025

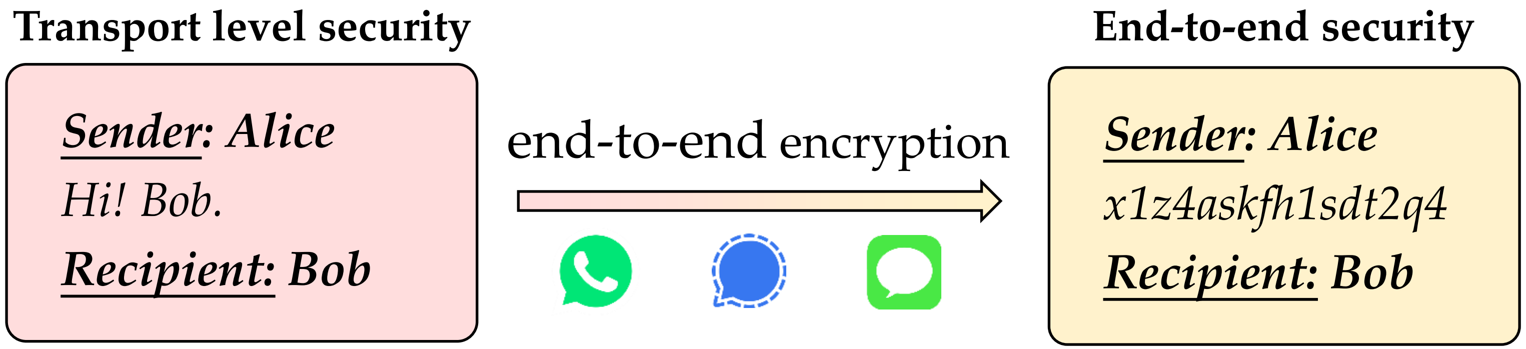
重庆大学
CHONGQING UNIVERSITY

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Content Moderation for EEMSs

End-to-end encrypted messaging systems (EEMS):

Only the END users can read the messages.



Transport level security

Sender*: *Alice

Hi! Bob.

Recipient*: *Bob

end-to-end encryption

End-to-end security

Sender*: *Alice

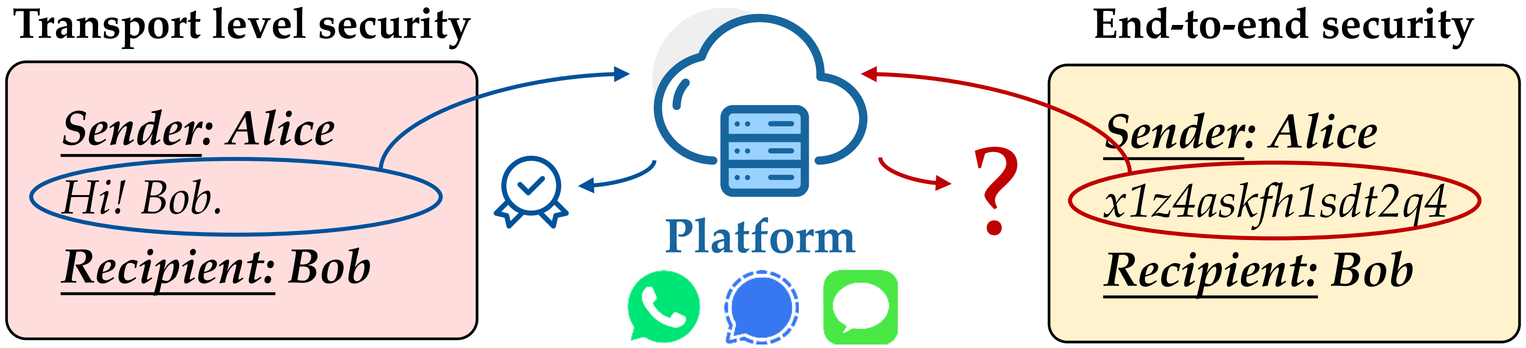
x1z4askfh1sdt2q4

Recipient*: *Bob

Content Moderation for EEMSs

End-to-end encrypted messaging systems (EEMS):

Only the END users can read the messages.



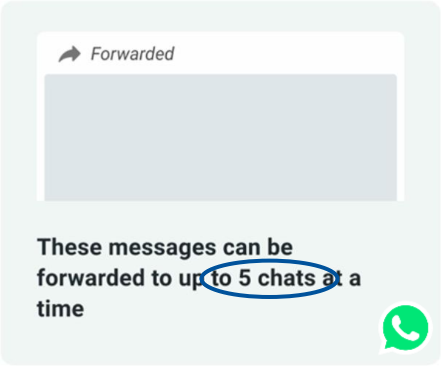
Problematic messages proliferates in EEMSs.

End-to-end encryption obstructs content moderation :(

[TAB+21,S&P] SoK: Hate, Harassment, and the Changing Landscape of Online Abuse

Message Forwarding & Tracing

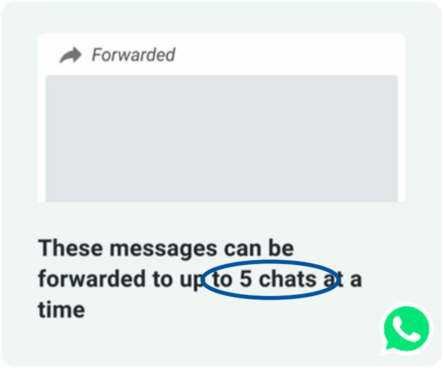
Misinformation propagate rapidly through forwarding.



These messages can be forwarded to up to 5 chats at a time

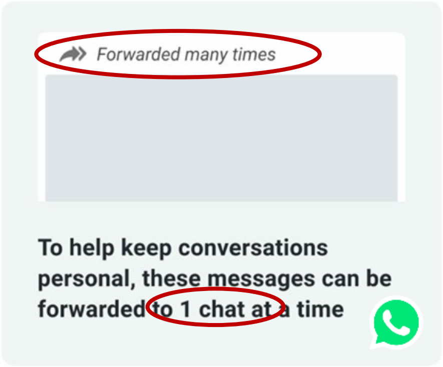
Message Forwarding & Tracing

Misinformation propagate rapidly through forwarding.



These messages can be forwarded to up to 5 chats at a time

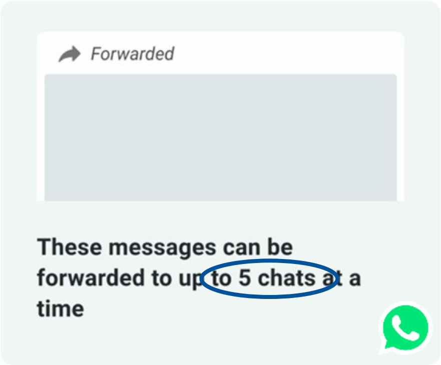
Forward limit



To help keep conversations personal, these messages can be forwarded to 1 chat at a time

Message Forwarding & Tracing

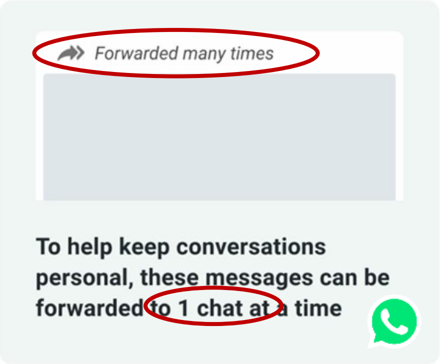
Misinformation propagate rapidly through forwarding.



Forwarded

These messages can be forwarded to up to 5 chats at a time

Forward limit

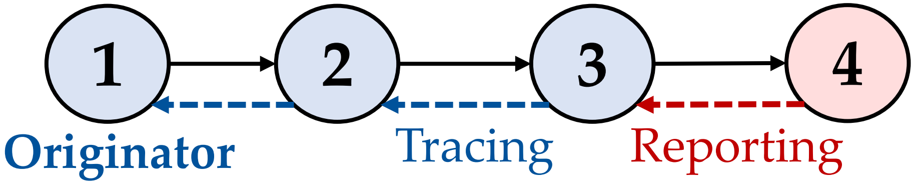


Forwarded many times

To help keep conversations personal, these messages can be forwarded to 1 chat at a time

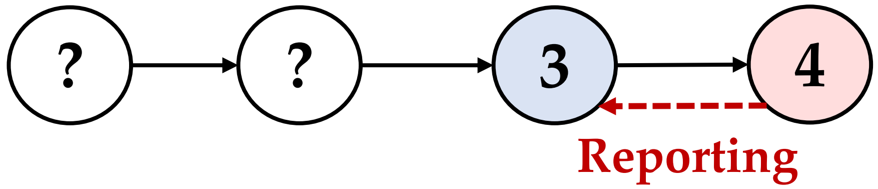
Traceability enables tracing after user reporting.

- The platform can disclose the dissemination path.



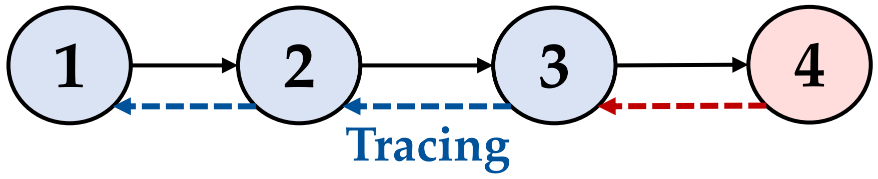
https://faq.whatsapp.com/1053543185312573/?helpref=uf_share

Tracing Policies



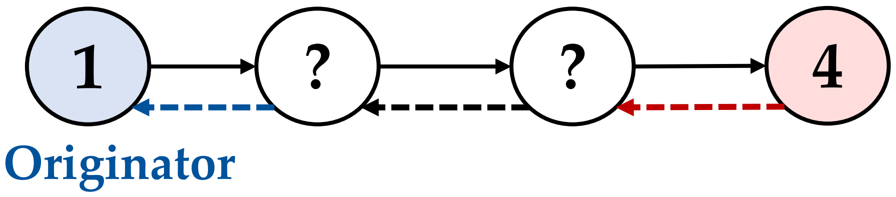
Message Franking

[Facebook17,RWC], [GLR17,CRYPTO], [TGL+19,CRYPTO], [GPE25,NDSS], etc.



Message Traceback

[TMR19,CCS], [KTW22,ESORICS]



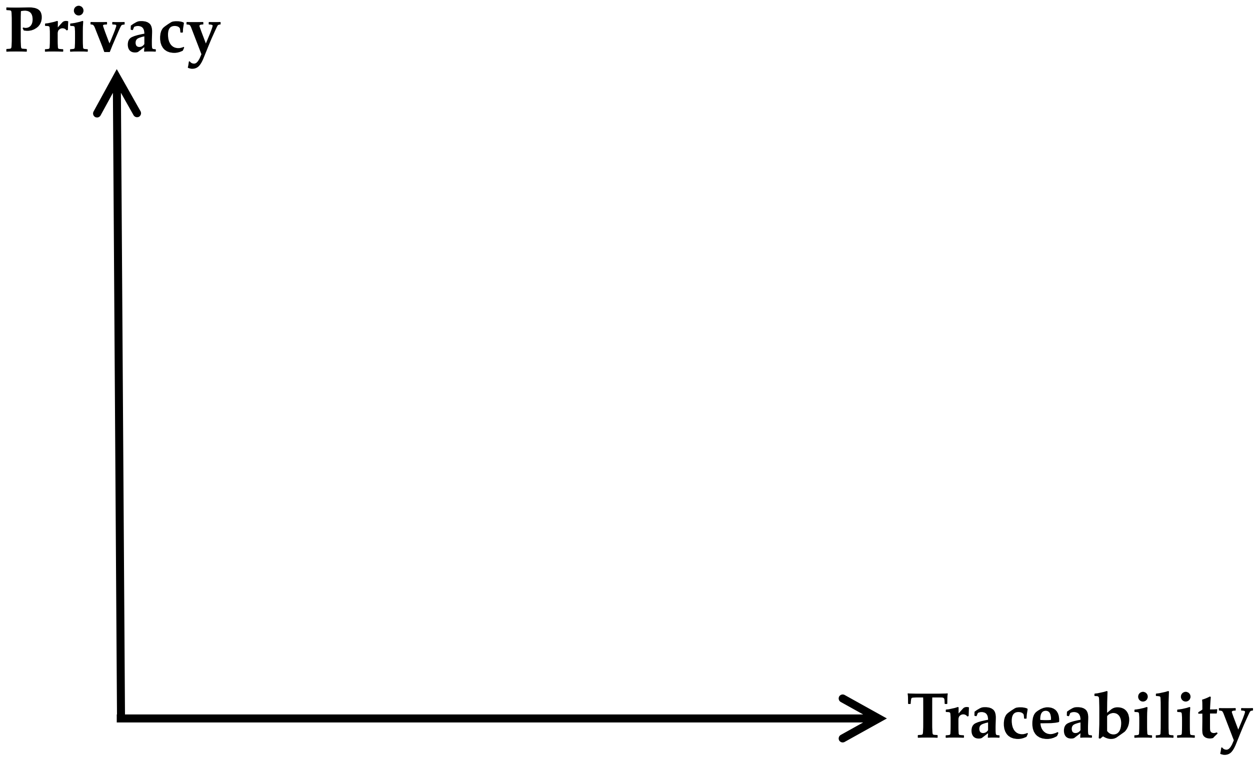
Source Tracing

[PEB21,CCS], [IAV22,Usenix Sec.], [LRTY22,NDSS], [BGJP23,EUROCRYPT], [BE24,PETS], etc.

Which part of the dissemination path is tracked during tracing?

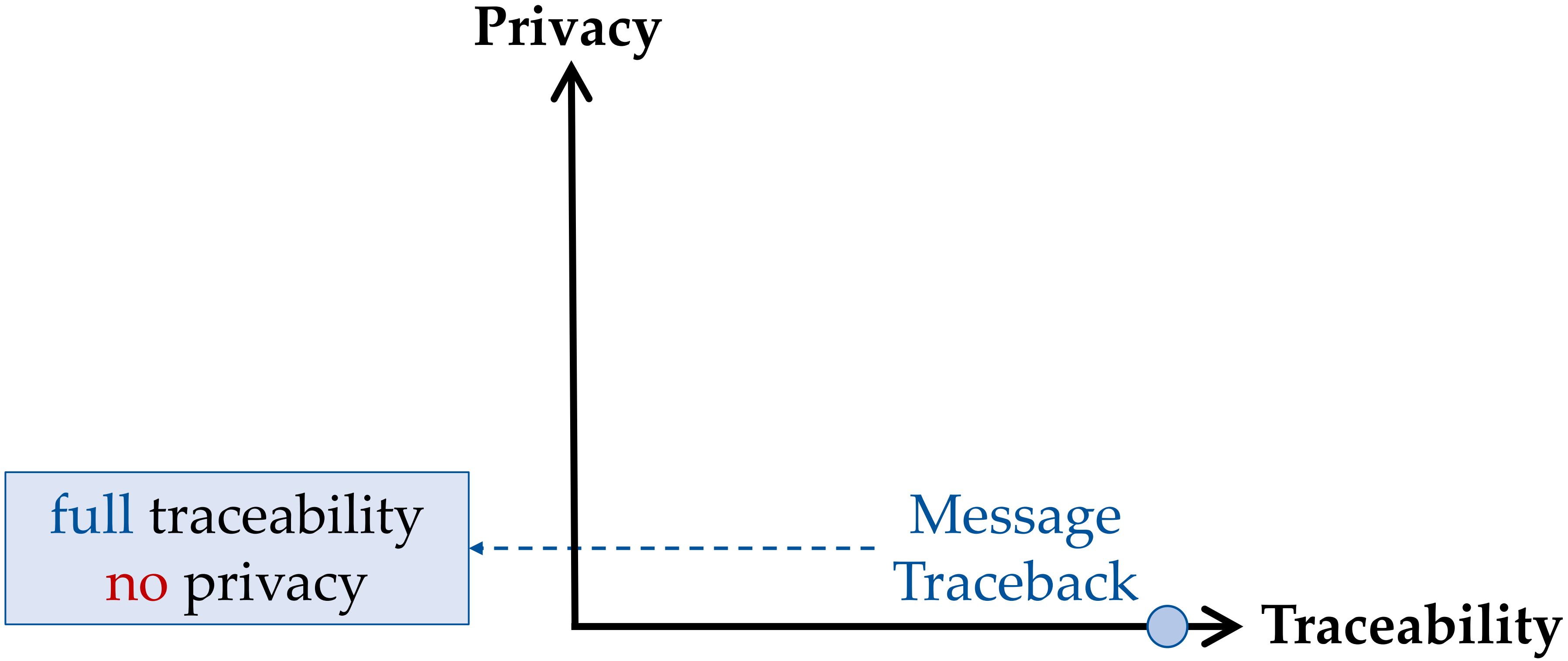
Traceability vs. Privacy

- **Traceability:** Reveal the culprits of spreading misinformation.
- **Privacy:** Reveal nothing about forwarding path.



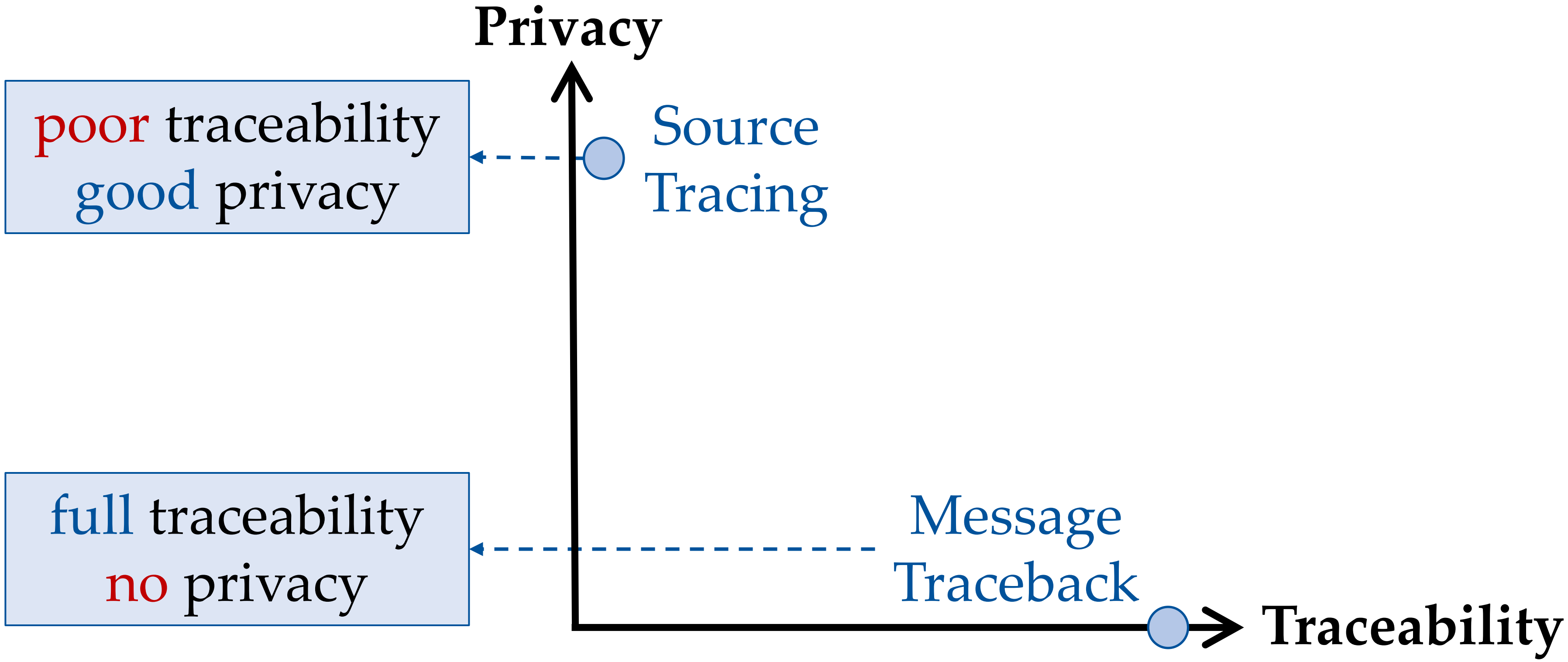
Traceability vs. Privacy

- **Traceability:** Reveal the culprits of spreading misinformation.
- **Privacy:** Reveal nothing about forwarding path.



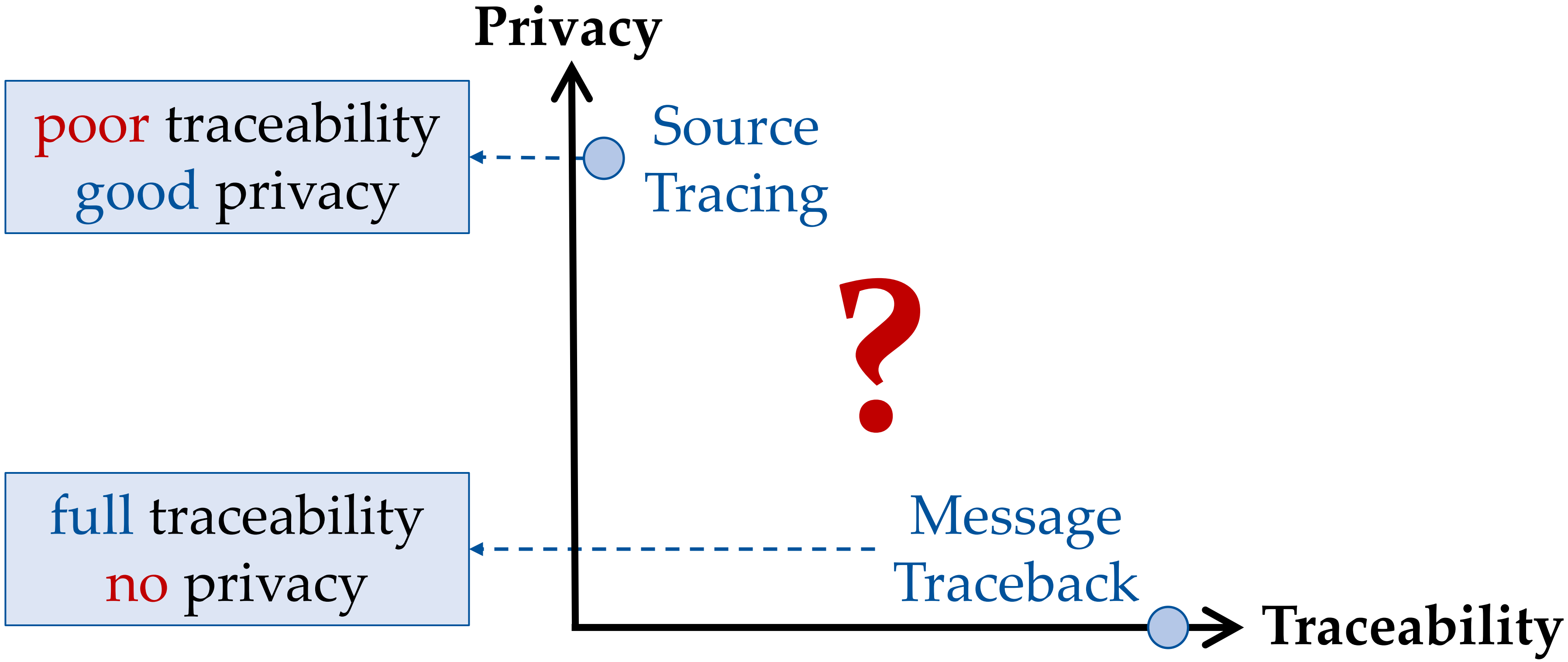
Traceability vs. Privacy

- **Traceability:** Reveal the culprits of spreading misinformation.
- **Privacy:** Reveal nothing about forwarding path.



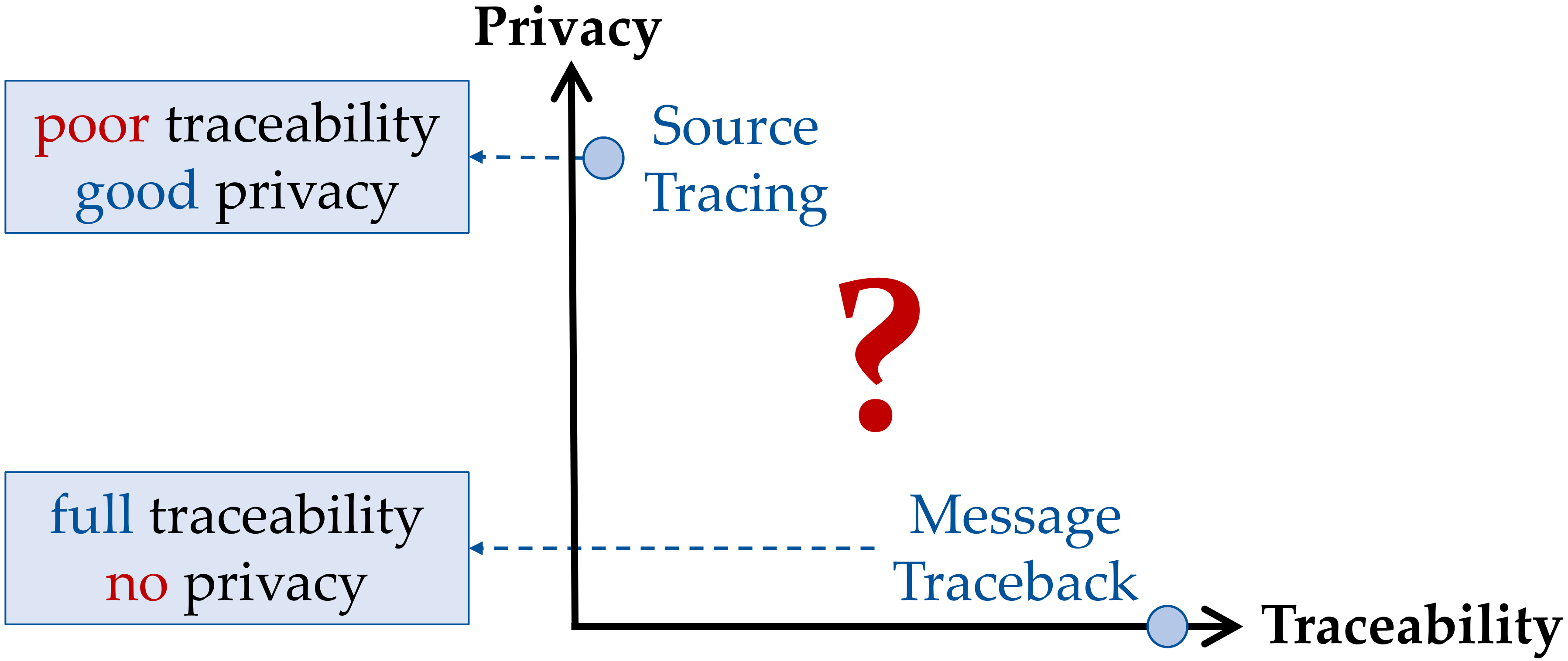
Traceability vs. Privacy

- **Traceability:** Reveal the culprits of spreading misinformation.
- **Privacy:** Reveal nothing about forwarding path.



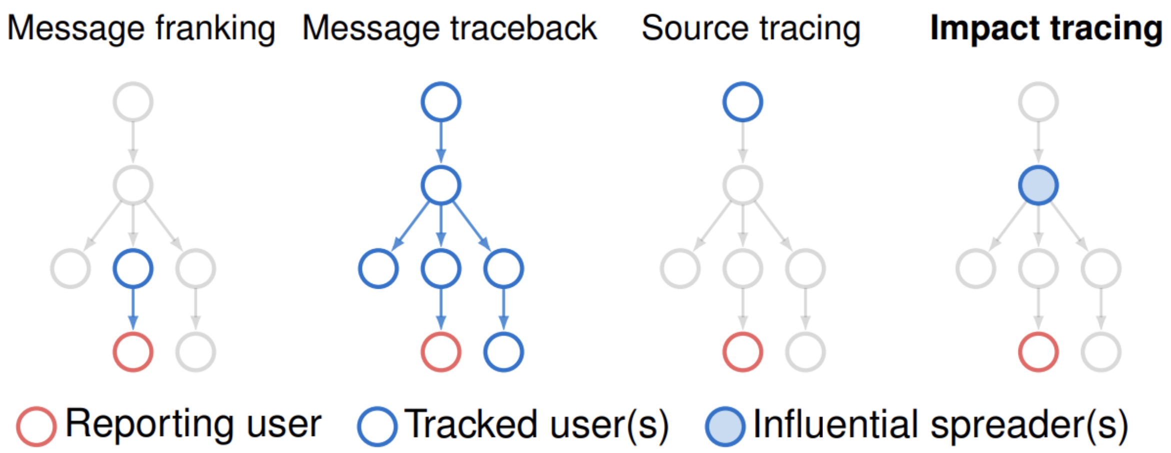
Traceability vs. Privacy

Is there a tracing policy that balances traceability and privacy, but also provides practical values to EEMSs?



Our Solution: Impact Tracing!

A small group of users (called influential spreaders) *significantly* contribute to spreading misinformation.



Traceability: The platform *can* identify influential spreaders.

Privacy: The platform *cannot* uncover non-influential users.

[CCDT21,Report] The Disinformation Dozen: Why Platforms must Act on Twelve Leading Online Anti-Vaxxers

The Design: Enabling Reporting

Sender

Platform

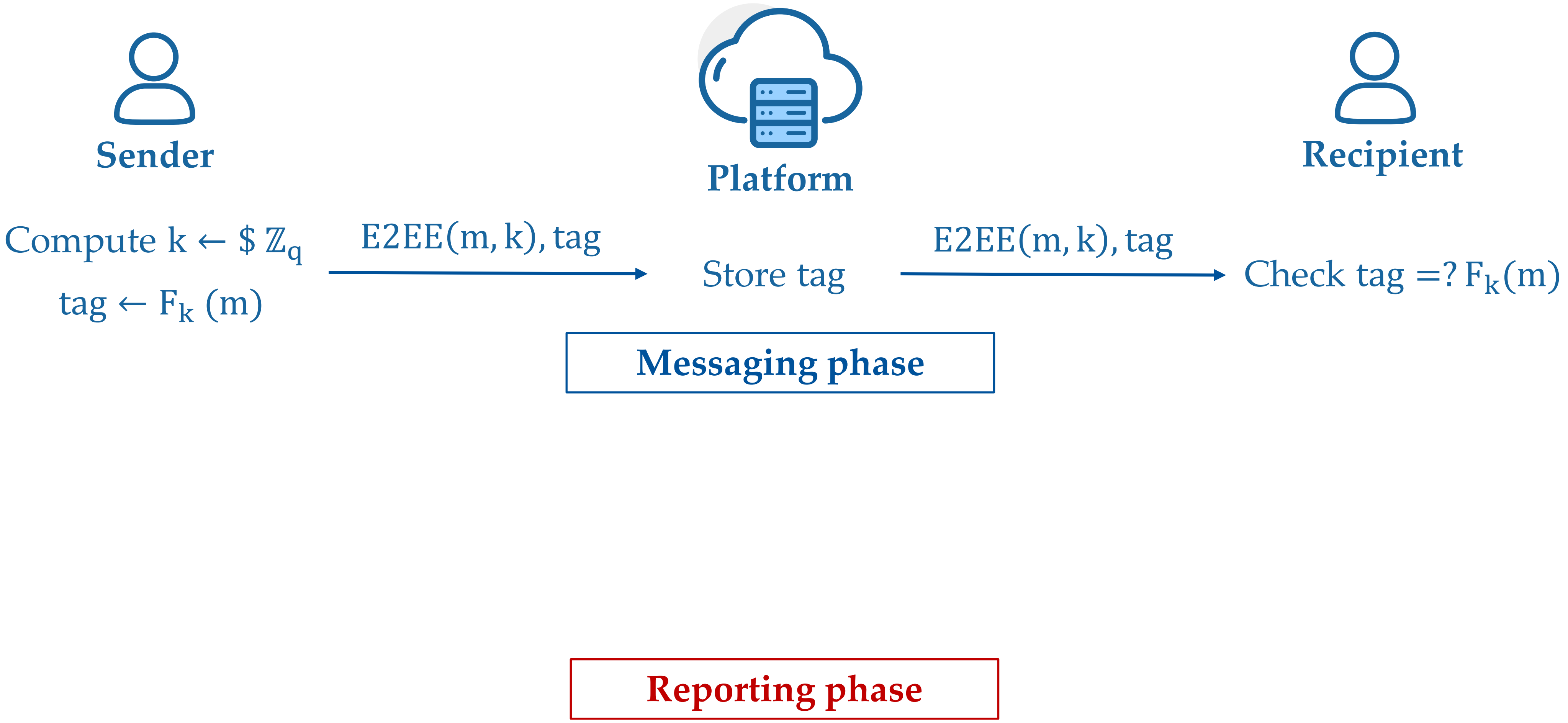
Recipient

Messaging phase

Reporting phase

[Facebook17,RWC] Challenges of E2E Encryption in Facebook Messenger

The Design: Enabling Reporting



Sender

Platform

Recipient

Compute $k \leftarrow \$\ \mathbb{Z}_q$

tag $\leftarrow \mathrm{F}_k\ (m)$

$\xrightarrow{\mathrm{E2EE}(m, k), \mathrm{tag}}$

Store tag

$\xrightarrow{\mathrm{E2EE}(m, k), \mathrm{tag}}$

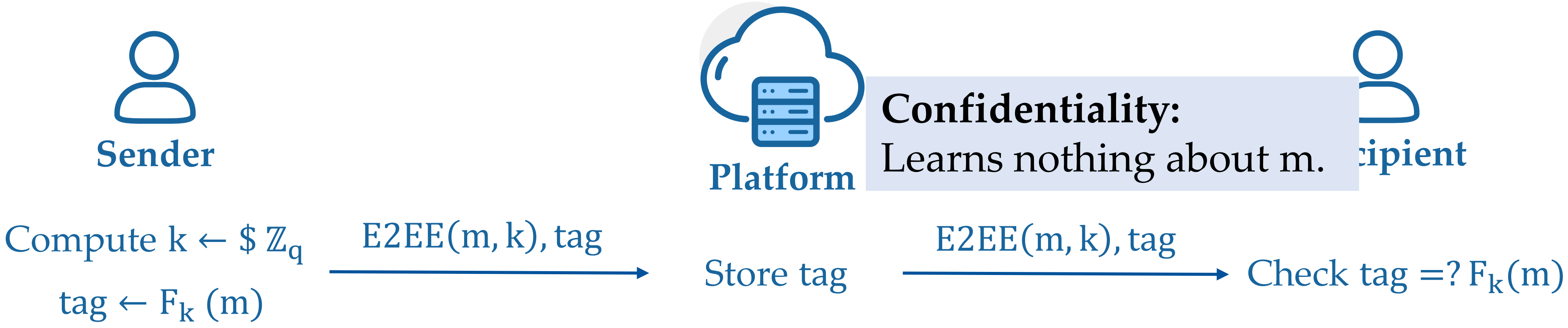
Check tag $=?\ \mathrm{F}_k(m)$

Messaging phase

Reporting phase

[Facebook17,RWC] Challenges of E2E Encryption in Facebook Messenger

The Design: Enabling Reporting



Confidentiality:
Learns nothing about m.

Compute $k \leftarrow \$ \, \mathbb{Z}_q$

$\text{tag} \leftarrow F_k \, (m)$

$\xrightarrow{\text{E2EE}(m, k), \text{tag}}$

Store tag

$\xrightarrow{\text{E2EE}(m, k), \text{tag}}$

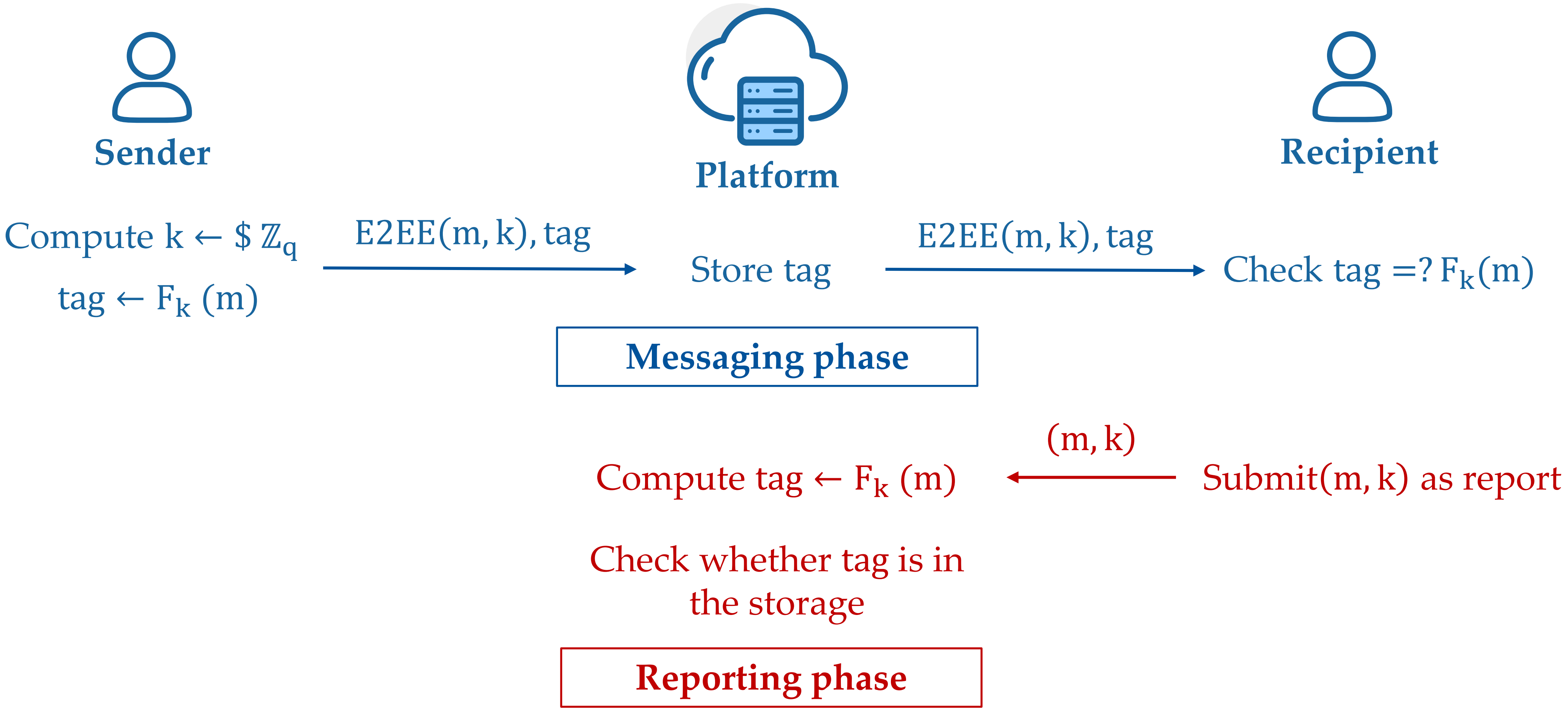
Check $\text{tag} =? F_k(m)$

Messaging phase

Reporting phase

[Facebook17,RWC] Challenges of E2E Encryption in Facebook Messenger

The Design: Enabling Reporting



Sender

Platform

Recipient

Compute $k \leftarrow \$ \, \mathbb{Z}_q$

$\text{tag} \leftarrow F_k\,(m)$

$\text{E2EE}(m, k), \text{tag}$

Store tag

$\text{E2EE}(m, k), \text{tag}$

Check tag $=? \, F_k(m)$

Messaging phase

(m, k)

Compute $\text{tag} \leftarrow F_k\,(m)$

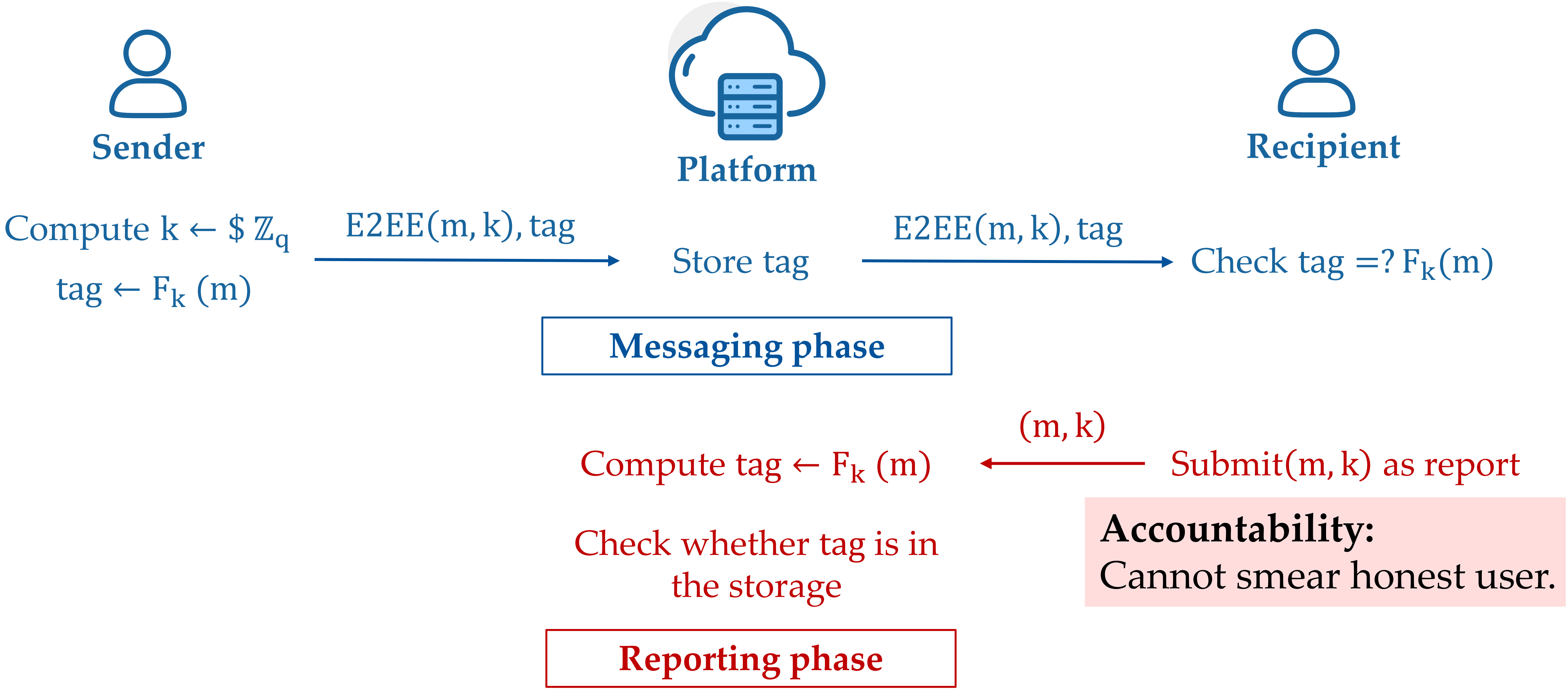
Submit(m, k) as report

Check whether tag is in the storage

Reporting phase

[Facebook17,RWC] Challenges of E2E Encryption in Facebook Messenger

The Design: Enabling Reporting



Sender

Platform

Recipient

Compute $k \leftarrow \$\ \mathbb{Z}_q$

$\text{tag} \leftarrow F_k\ (m)$

$\text{E2EE}(m, k), \text{tag}$

Store tag

$\text{E2EE}(m, k), \text{tag}$

Check tag $=?\ F_k(m)$

Messaging phase

(m, k)

Compute tag $\leftarrow F_k\ (m)$

Submit(m, k) as report

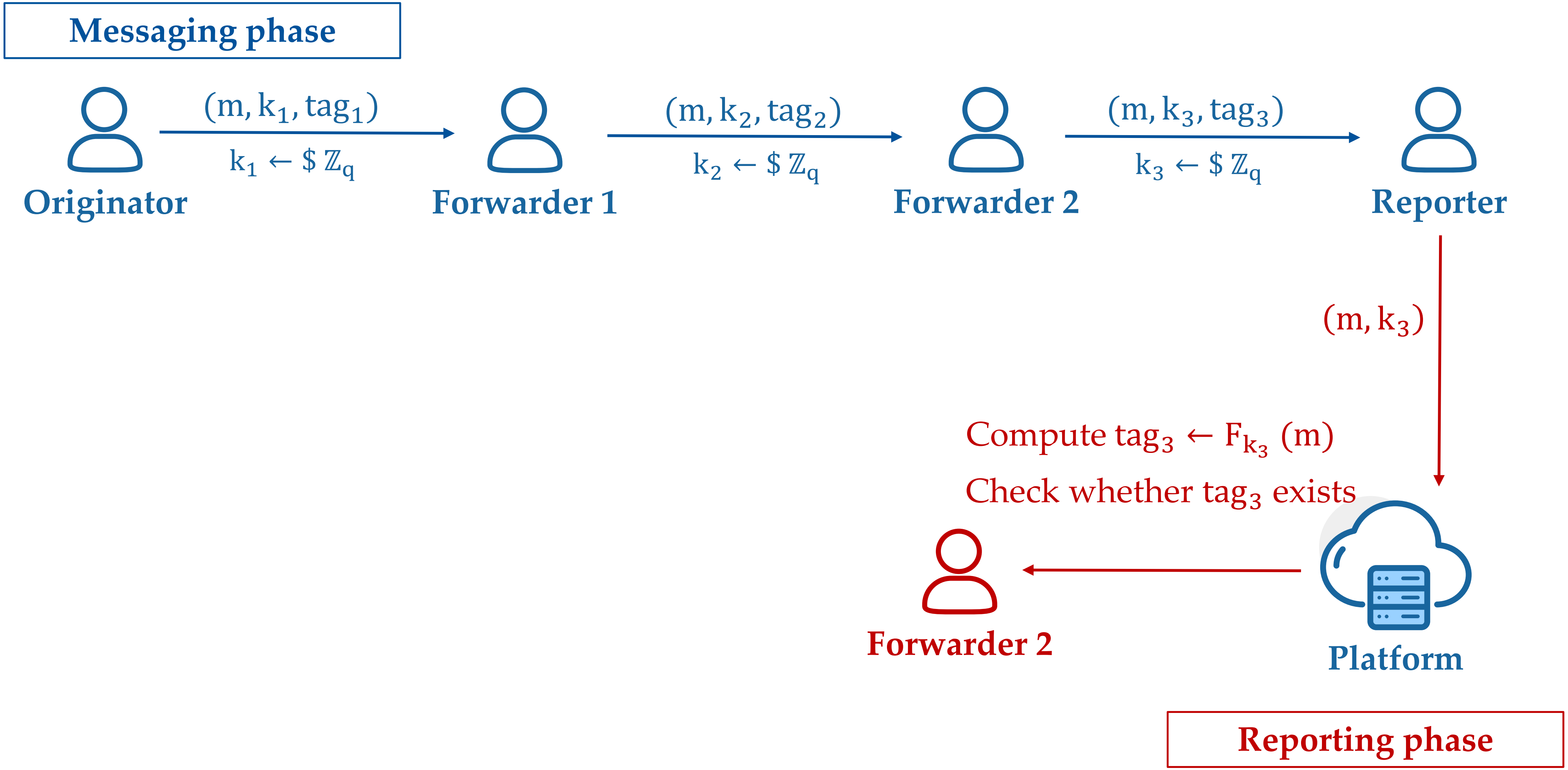
Check whether tag is in the storage

Accountability:
Cannot smear honest user.

Reporting phase

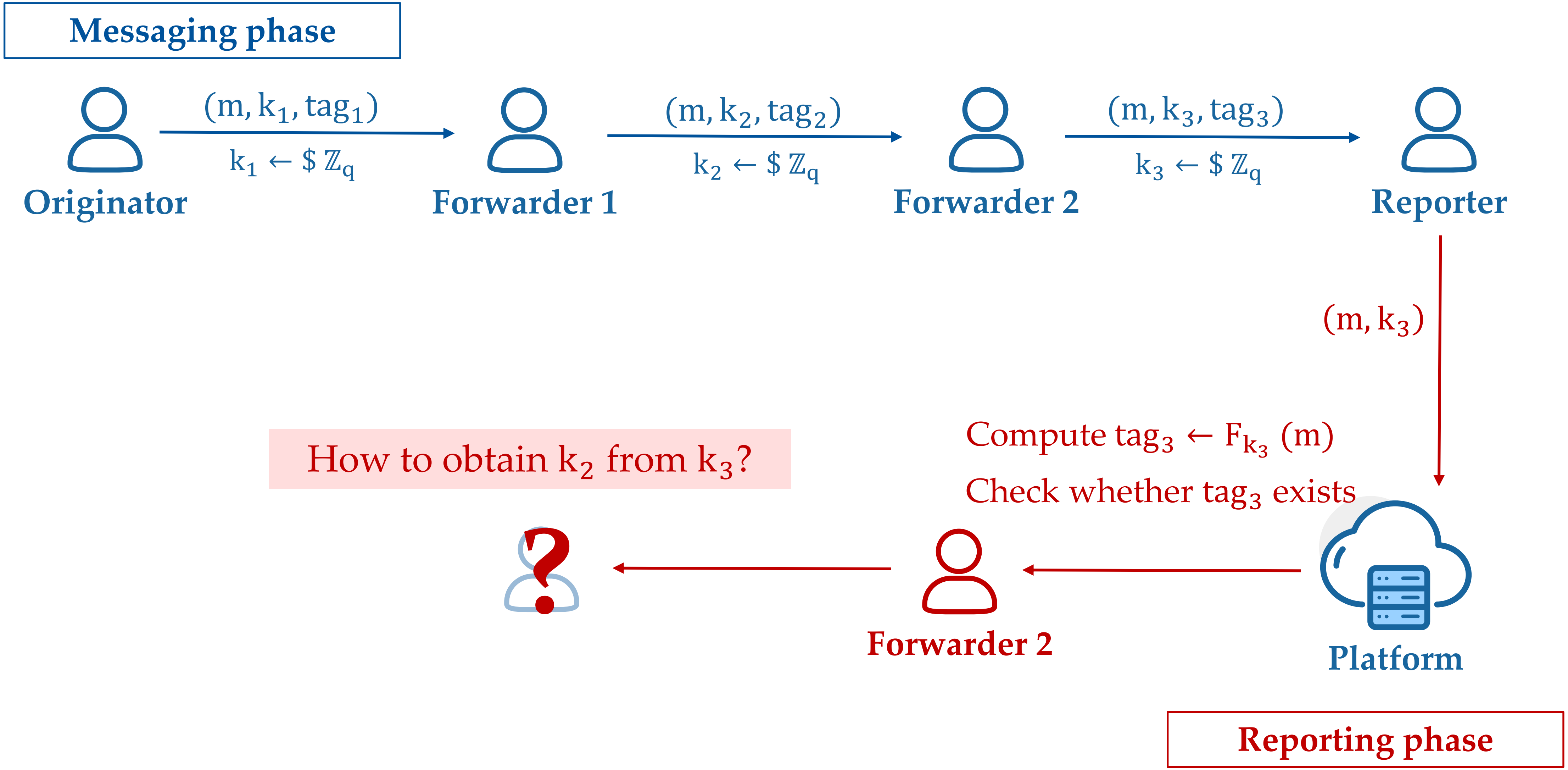
[Facebook17,RWC] Challenges of E2E Encryption in Facebook Messenger

The Design: Enabling Reporting



Messaging phase
(m, k_1, tag_1)
$k_1 \leftarrow \$ \mathbb{Z}_q$
Originator
(m, k_2, tag_2)
$k_2 \leftarrow \$ \mathbb{Z}_q$
Forwarder 1
(m, k_3, tag_3)
$k_3 \leftarrow \$ \mathbb{Z}_q$
Forwarder 2
Reporter
(m, k_3)
Compute $tag_3 \leftarrow F_{k_3}(m)$
Check whether tag_3 exists
Forwarder 2
Platform
Reporting phase

The Design: Enabling Reporting



Messaging phase

Originator

(m, k_1, tag_1)

$k_1 \leftarrow \$ \, \mathbb{Z}_q$

Forwarder 1

(m, k_2, tag_2)

$k_2 \leftarrow \$ \, \mathbb{Z}_q$

Forwarder 2

(m, k_3, tag_3)

$k_3 \leftarrow \$ \, \mathbb{Z}_q$

Reporter

(m, k_3)

Compute $tag_3 \leftarrow F_{k_3}(m)$

Check whether tag_3 exists

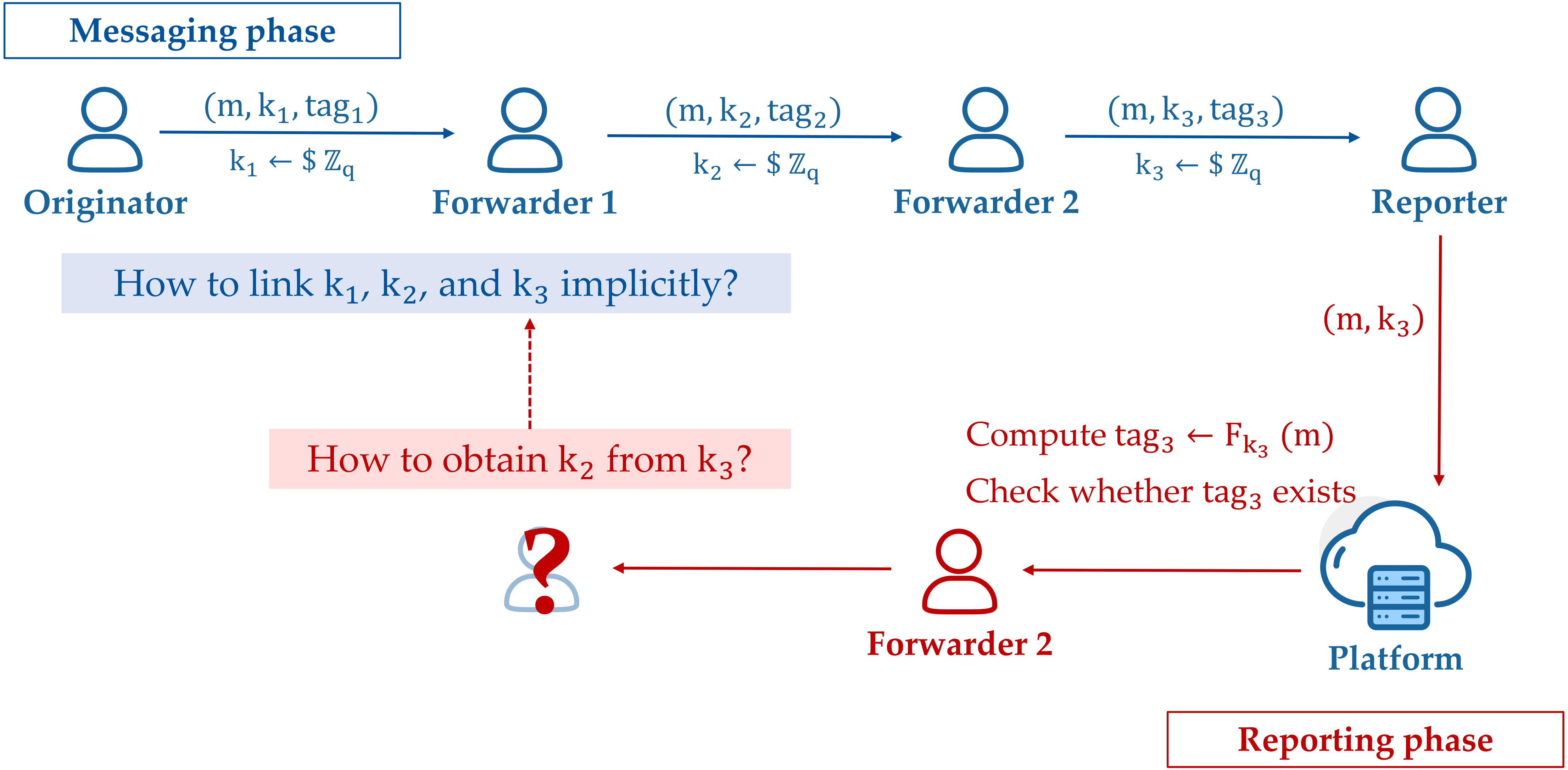
Platform

Forwarder 2

How to obtain k_2 from k_3?

Reporting phase

The Design: Enabling Reporting



Messaging phase

How to link k_1, k_2, and k_3 implicitly?

How to obtain k_2 from k_3?

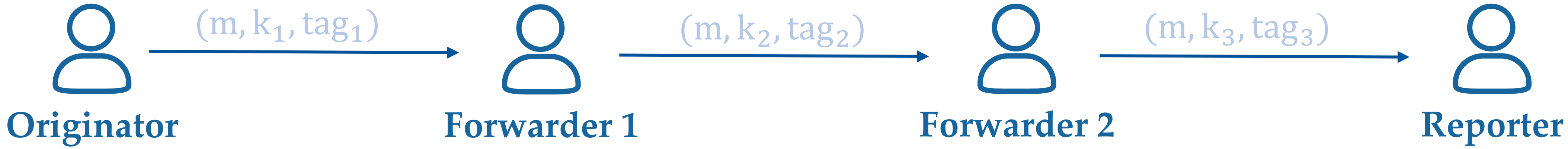
Compute $tag_3 \leftarrow F_{k_3}(m)$

Check whether tag_3 exists

Reporting phase

The Design: Reporting then Tracing

Messaging phase



(m, k_1, tag_1)

(m, k_2, tag_2)

(m, k_3, tag_3)

Originator

Forwarder 1

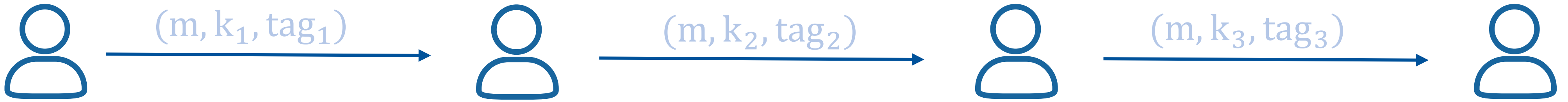
Forwarder 2

Reporter

The Design: Reporting then Tracing

Messaging phase

ik: Users' secret identity key; **U:** Users' public identity; **Enc:** Block cipher



(m, k_1, tag_1) (m, k_2, tag_2) (m, k_3, tag_3)

Originator

$tk_{01} \leftarrow H(ik_0 || U_1)$

$k_1 \leftarrow Enc_{tk_{01}}(k_0)$

$k_0 \leftarrow \$ \, \mathbb{Z}_q$

Forwarder 1

$tk_{12} \leftarrow H(ik_1 || U_2)$

$k_2 \leftarrow Enc_{tk_{12}}(k_1)$

Forwarder 2

$tk_{23} \leftarrow H(ik_2 || U_3)$

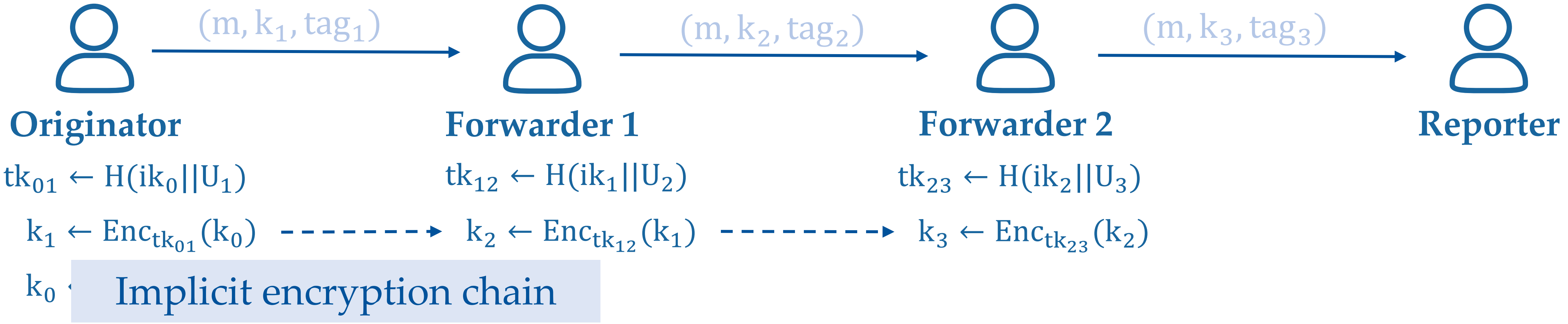
$k_3 \leftarrow Enc_{tk_{23}}(k_2)$

Reporter

The Design: Reporting then Tracing

Messaging phase

ik: Users' secret identity key; **U:** Users' public identity; **Enc:** Block cipher



$(\mathrm{m}, \mathrm{k}_1, \mathrm{tag}_1)$

$(\mathrm{m}, \mathrm{k}_2, \mathrm{tag}_2)$

$(\mathrm{m}, \mathrm{k}_3, \mathrm{tag}_3)$

Originator

Forwarder 1

Forwarder 2

Reporter

$\mathrm{tk}_{01} \leftarrow \mathrm{H}(\mathrm{ik}_0 || \mathrm{U}_1)$

$\mathrm{k}_1 \leftarrow \mathrm{Enc}_{\mathrm{tk}_{01}}(\mathrm{k}_0)$

k_0

$\mathrm{tk}_{12} \leftarrow \mathrm{H}(\mathrm{ik}_1 || \mathrm{U}_2)$

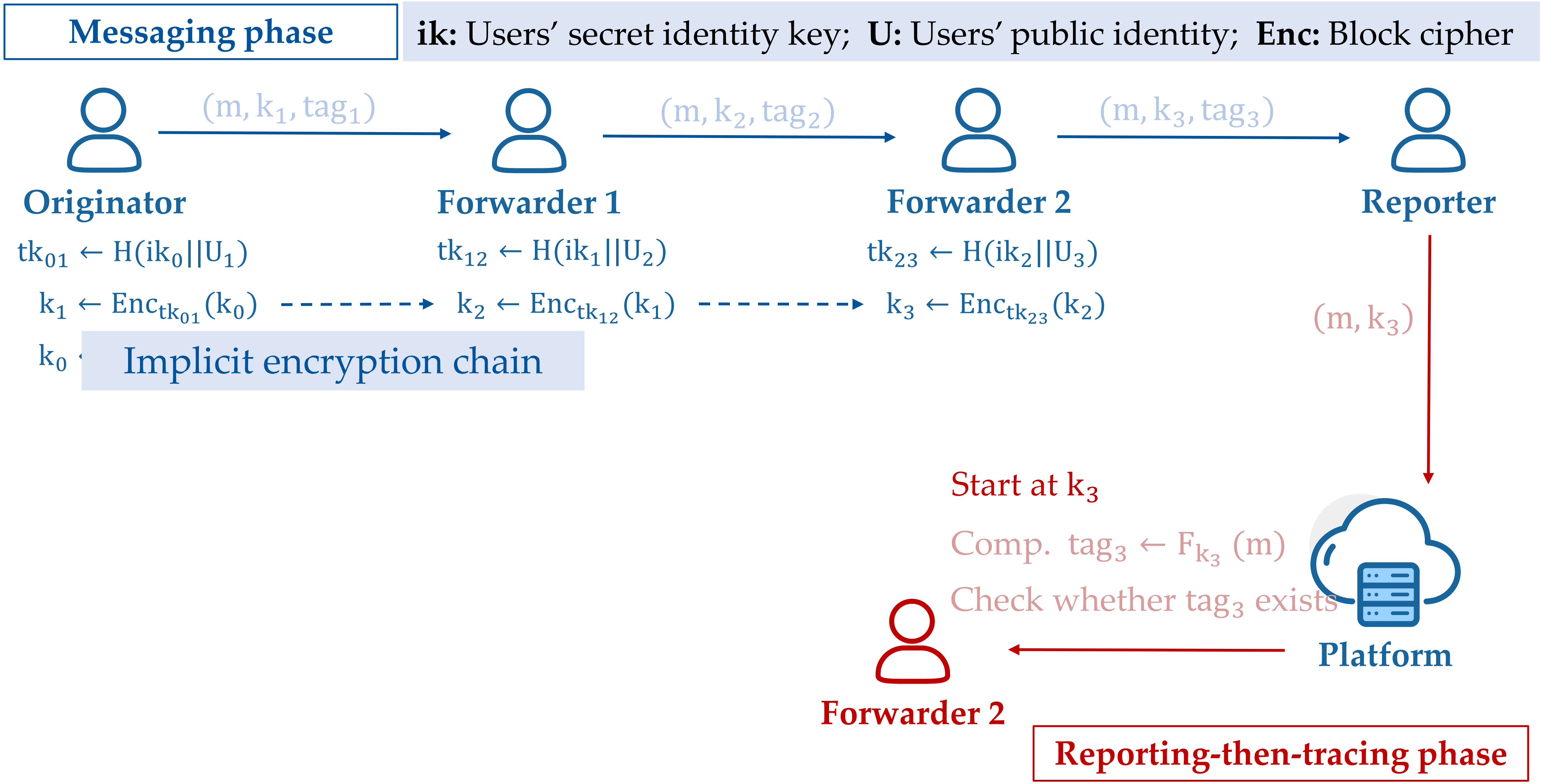
$\mathrm{k}_2 \leftarrow \mathrm{Enc}_{\mathrm{tk}_{12}}(\mathrm{k}_1)$

$\mathrm{tk}_{23} \leftarrow \mathrm{H}(\mathrm{ik}_2 || \mathrm{U}_3)$

$\mathrm{k}_3 \leftarrow \mathrm{Enc}_{\mathrm{tk}_{23}}(\mathrm{k}_2)$

Implicit encryption chain

The Design: Reporting then Tracing



Messaging phase

ik: Users' secret identity key; **U:** Users' public identity; **Enc:** Block cipher

(m, k_1, tag_1)

(m, k_2, tag_2)

(m, k_3, tag_3)

Originator

$tk_{01} \leftarrow H(ik_0 || U_1)$

$k_1 \leftarrow Enc_{tk_{01}}(k_0)$

k_0

Implicit encryption chain

Forwarder 1

$tk_{12} \leftarrow H(ik_1 || U_2)$

$k_2 \leftarrow Enc_{tk_{12}}(k_1)$

Forwarder 2

$tk_{23} \leftarrow H(ik_2 || U_3)$

$k_3 \leftarrow Enc_{tk_{23}}(k_2)$

Reporter

(m, k_3)

Start at k_3

Comp. $tag_3 \leftarrow F_{k_3}(m)$

Check whether tag_3 exists

Platform

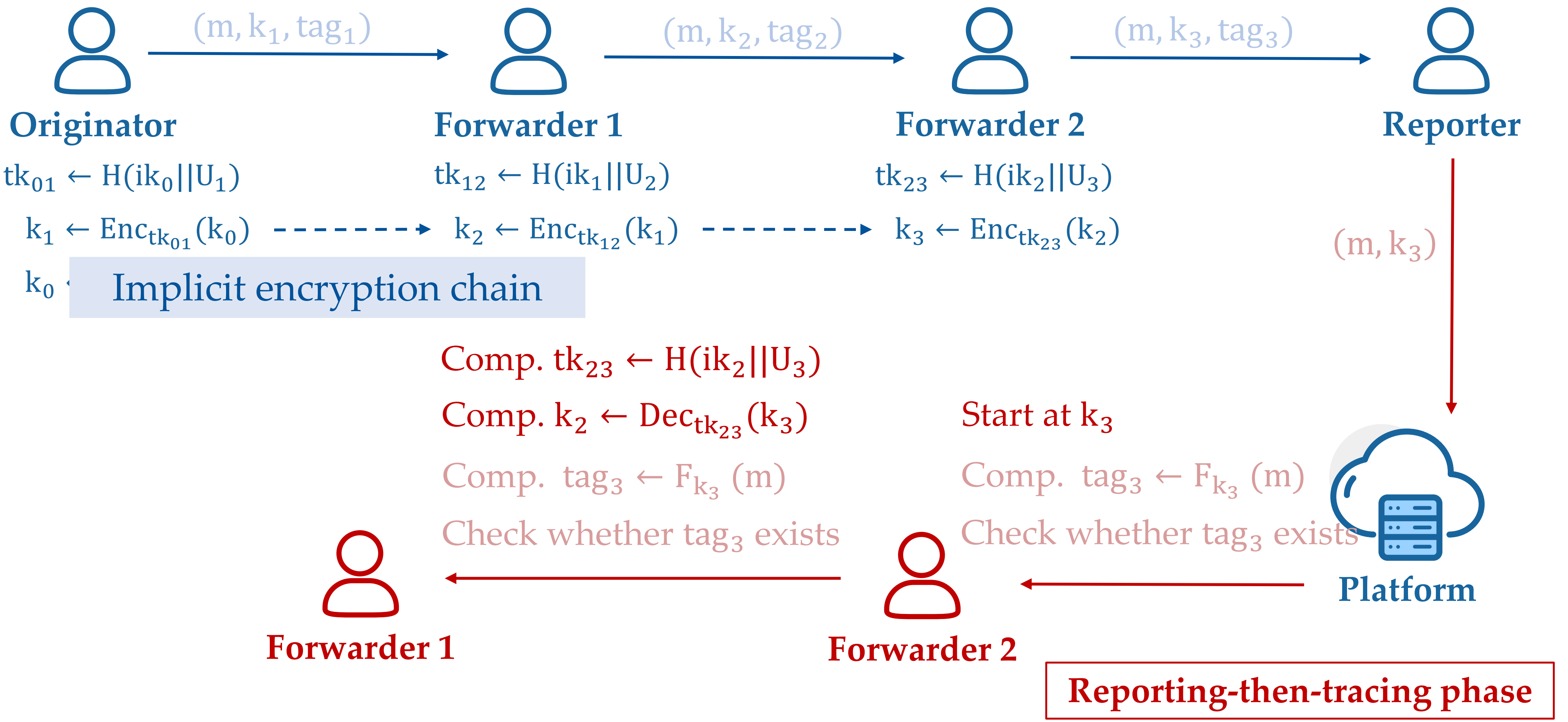
Forwarder 2

Reporting-then-tracing phase

The Design: Reporting then Tracing

Messaging phase

ik: Users' secret identity key; **U:** Users' public identity; **Enc:** Block cipher



(m, k_1, tag_1)

(m, k_2, tag_2)

(m, k_3, tag_3)

Originator

$tk_{01} \leftarrow H(ik_0||U_1)$

$k_1 \leftarrow Enc_{tk_{01}}(k_0)$

k_0

Forwarder 1

$tk_{12} \leftarrow H(ik_1||U_2)$

$k_2 \leftarrow Enc_{tk_{12}}(k_1)$

Forwarder 2

$tk_{23} \leftarrow H(ik_2||U_3)$

$k_3 \leftarrow Enc_{tk_{23}}(k_2)$

Reporter

Implicit encryption chain

(m, k_3)

Start at k_3

Comp. $tag_3 \leftarrow F_{k_3}(m)$

Check whether tag_3 exists

Platform

Comp. $tk_{23} \leftarrow H(ik_2||U_3)$

Comp. $k_2 \leftarrow Dec_{tk_{23}}(k_3)$

Comp. $tag_3 \leftarrow F_{k_3}(m)$

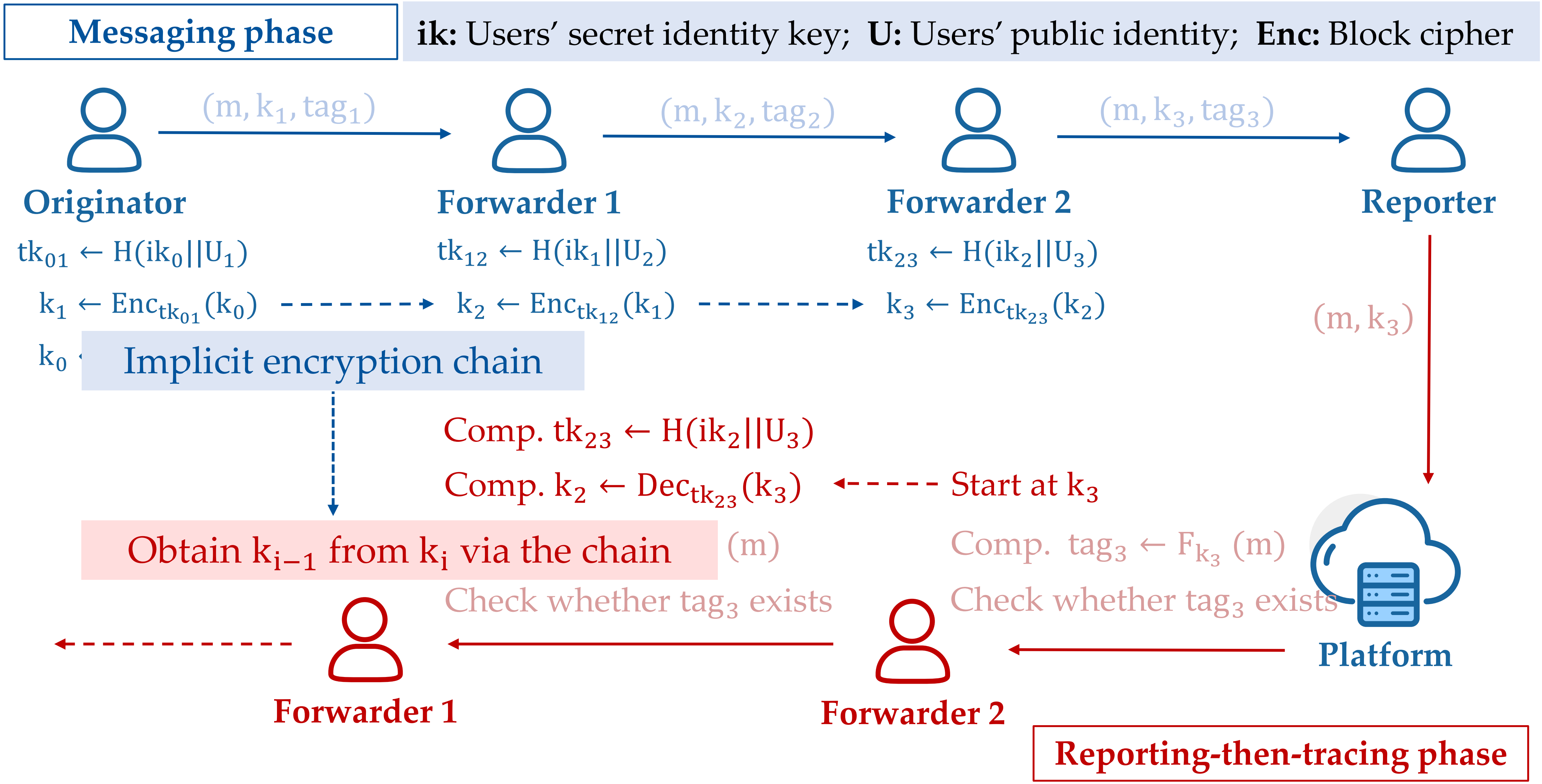
Check whether tag_3 exists

Forwarder 2

Forwarder 1

Reporting-then-tracing phase

The Design: Reporting then Tracing



Messaging phase

ik: Users' secret identity key; **U:** Users' public identity; **Enc:** Block cipher

Originator $\xrightarrow{(m, k_1, tag_1)}$ **Forwarder 1** $\xrightarrow{(m, k_2, tag_2)}$ **Forwarder 2** $\xrightarrow{(m, k_3, tag_3)}$ **Reporter**

Originator:

$tk_{01} \leftarrow H(ik_0 || U_1)$

$k_1 \leftarrow Enc_{tk_{01}}(k_0)$

$k_0 \leftarrow$

Forwarder 1:

$tk_{12} \leftarrow H(ik_1 || U_2)$

$k_2 \leftarrow Enc_{tk_{12}}(k_1)$

Forwarder 2:

$tk_{23} \leftarrow H(ik_2 || U_3)$

$k_3 \leftarrow Enc_{tk_{23}}(k_2)$

Implicit encryption chain

Reporter → Platform: (m, k_3)

Start at k_3

Comp. $tk_{23} \leftarrow H(ik_2 || U_3)$

Comp. $k_2 \leftarrow Dec_{tk_{23}}(k_3)$

Obtain k_{i-1} from k_i via the chain

Platform → Forwarder 2: (m)

Comp. $tag_3 \leftarrow F_{k_3}(m)$

Check whether tag_3 exists

Forwarder 2 → Forwarder 1: (m)

Check whether tag_3 exists

Platform

Forwarder 2

Forwarder 1

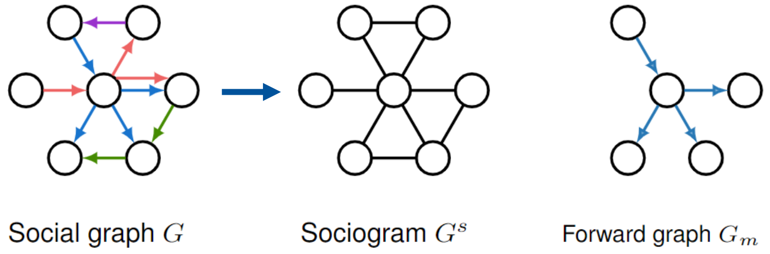
Reporting-then-tracing phase

The Design: Reporting then Tracing

Messaging phase:

- Users forward messages.

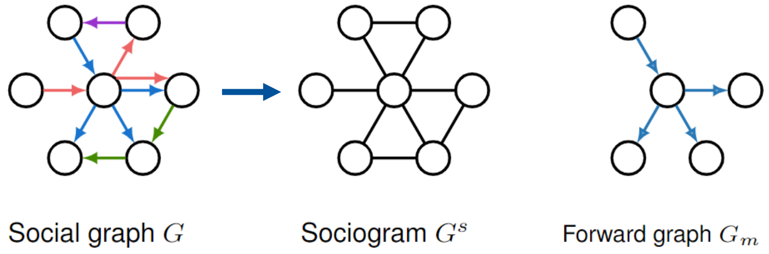
- Platform collects sociogram & stores tags.



The Design: Reporting then Tracing

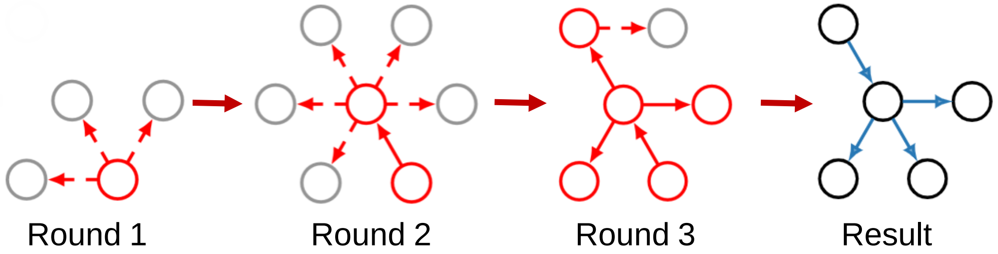
Messaging phase:

- Users forward messages.
- Platform collects sociogram & stores tags.



Reporting-then-Tracing phase:

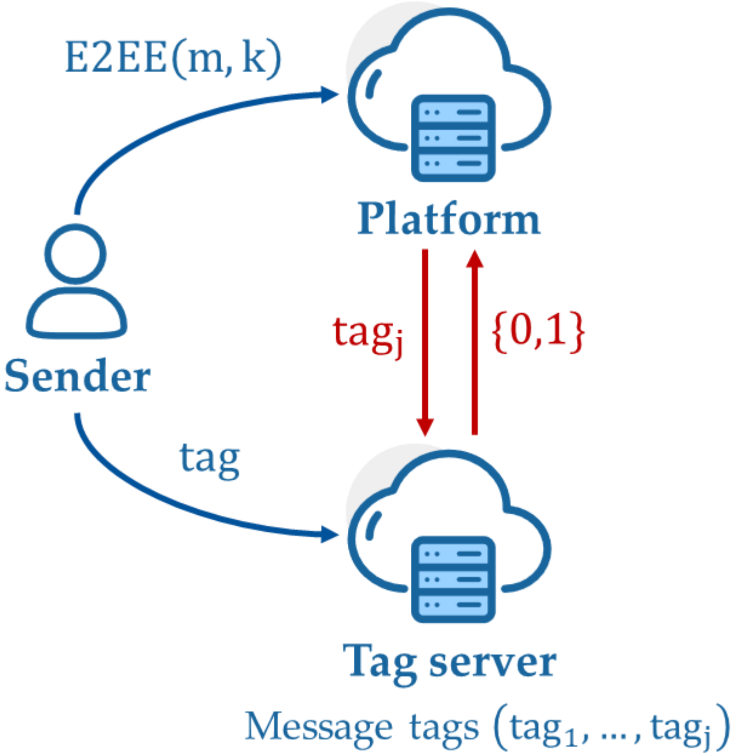
1) Compute tag key and tag.
2) Check the existence of the tag.
3) Start next round at existed vertices.



The Design: Noising the Traceback

Introduce random response to add noise.

- Two servers: Platform & Tag server

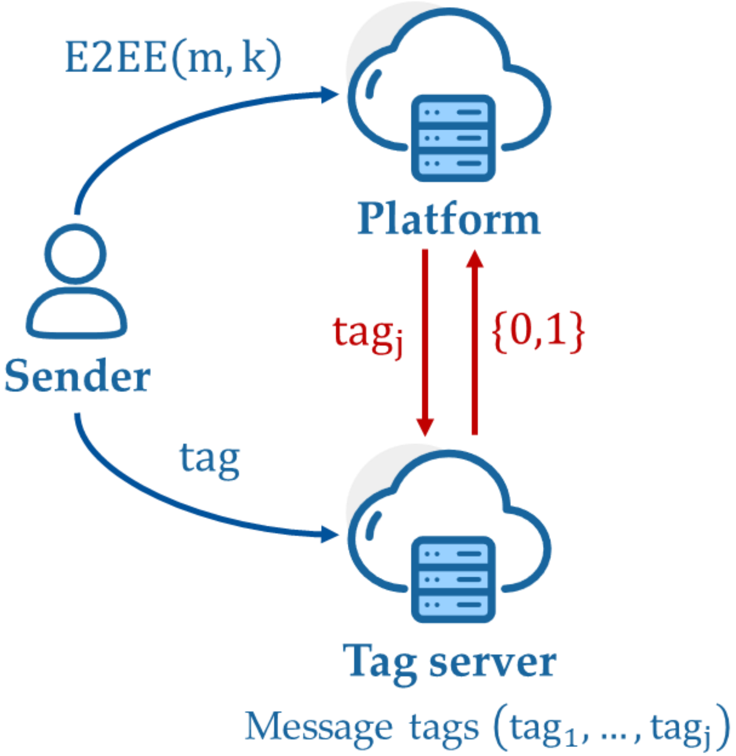


The Design: Noising the Traceback

Introduce random response to add noise.

- Two servers: Platform & Tag server
- Tag server responses a query as:

```
if $tag_j$ exists, return 1
else, return 0 w.p. $1 - \psi$
      return 1 w.p. $\psi$
```

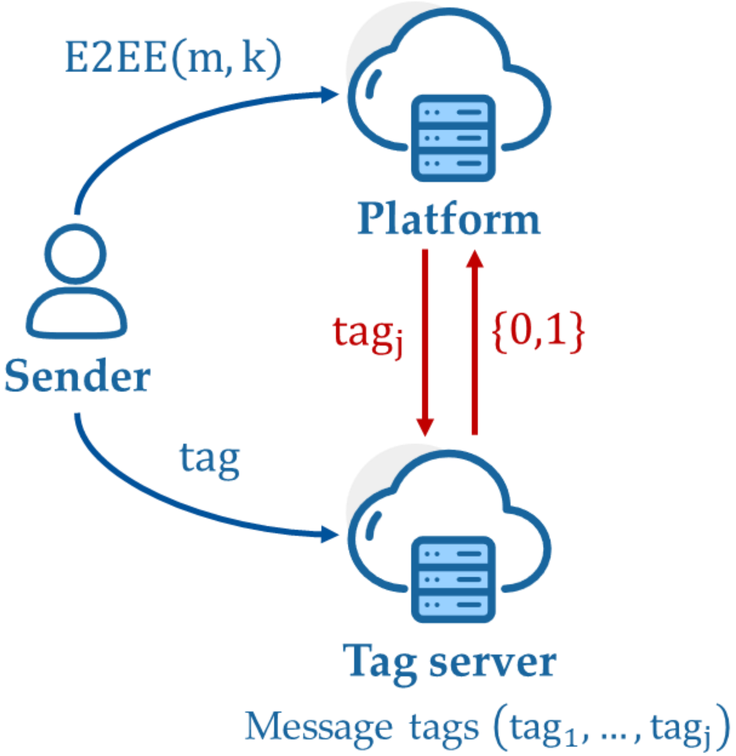


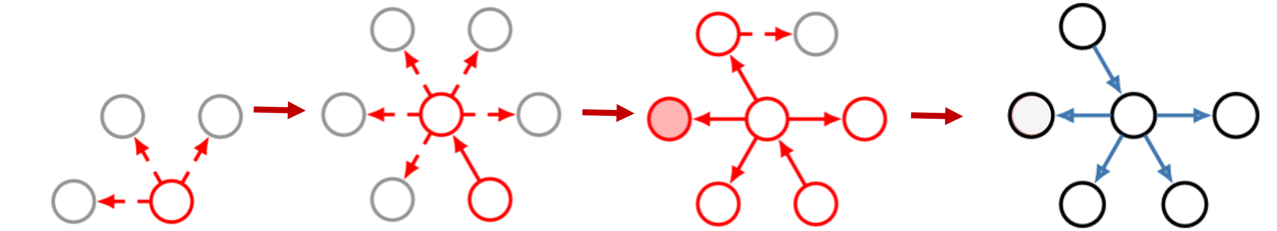
The Design: Noising the Traceback

Introduce random response to add noise.

- Two servers: Platform & Tag server
- Tag server responses a query as:

> if $\mathrm{tag_j}$ exists, return 1
>
> else, return 0 w.p. $1 - \psi$
>
> return 1 w.p. ψ





Round 1

Round 2

Round 3

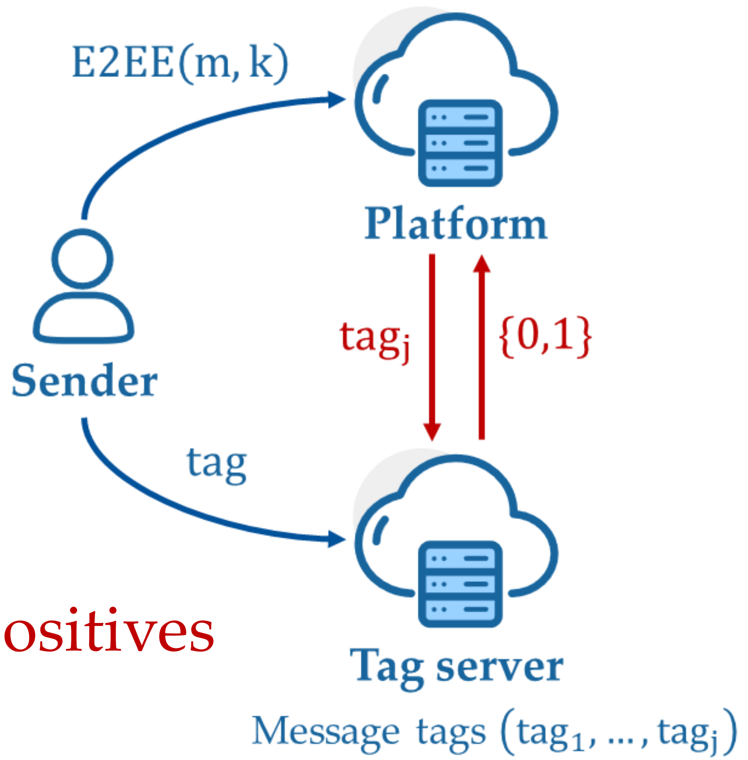
Noised graph G^*_m

The Design: Noising the Traceback

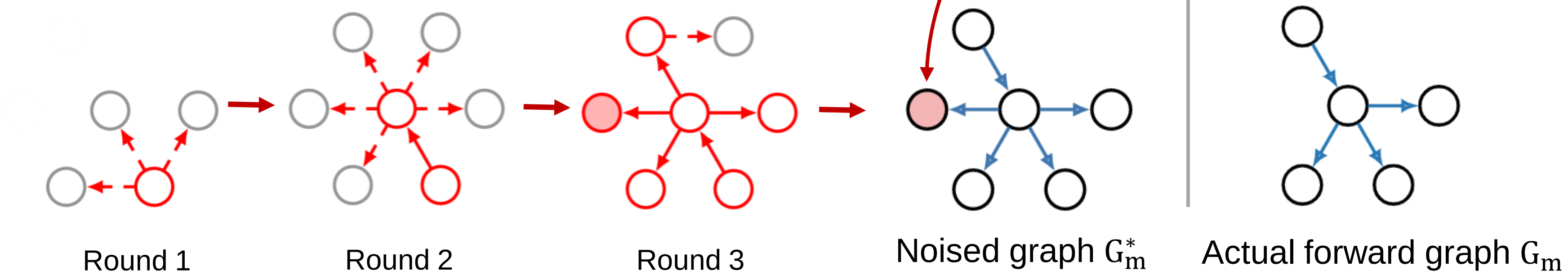
Introduce random response to add noise.

- Two servers: Platform & Tag server
- Tag server responses a query as:

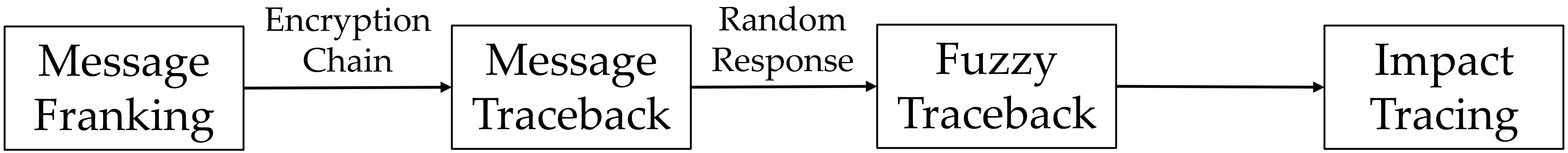
> if tag_j exists, return 1
>
> else, return 0 w.p. $1-\psi$
>
> return 1 w.p. ψ



False positives



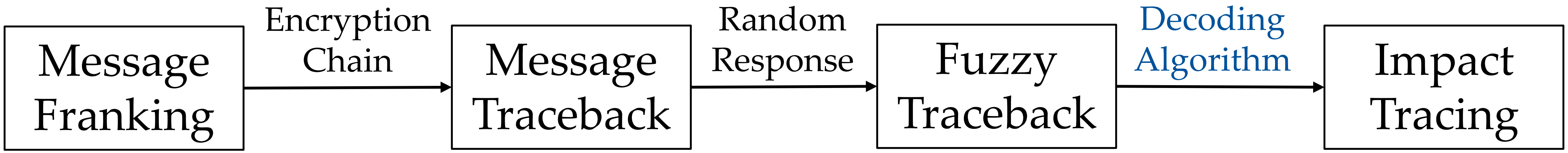
Remaining Concerns



Remaining Concerns

How to reveal influential spreaders from noised graph?

- Design a decoding algorithm to identify influential spreaders.



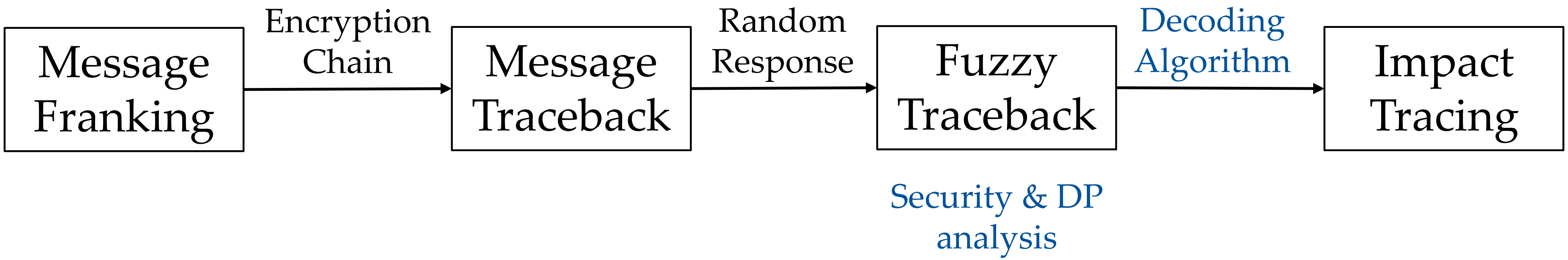
Remaining Concerns

How to reveal influential spreaders from noised graph?

- Design a decoding algorithm to identify influential spreaders.

Does the decoding algorithm violate privacy?

- Prove the noised graph satisfies differential privacy.



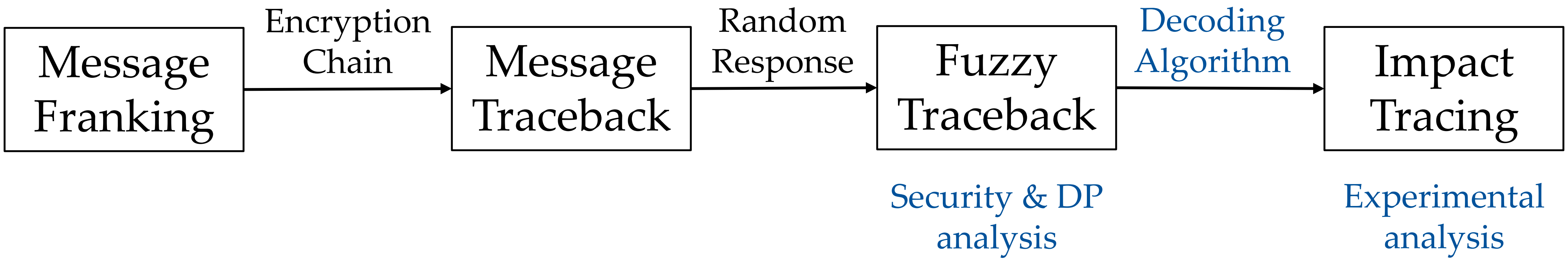
Remaining Concerns

How to reveal influential spreaders from noised graph?

- Design a decoding algorithm to identify influential spreaders.

Does the decoding algorithm violate privacy?

- Prove the noised graph satisfies differential privacy.



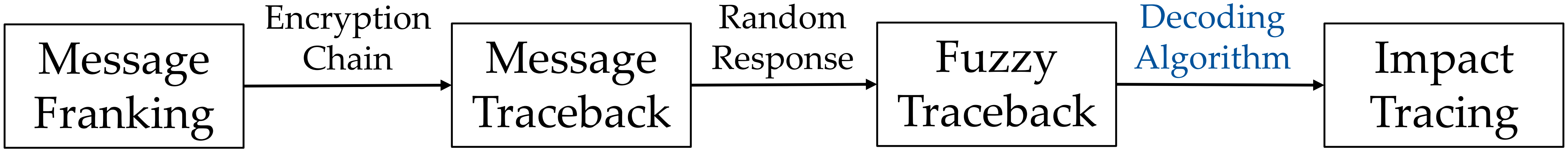
Remaining Concerns

How to reveal influential spreaders from noised graph?

- Design a decoding algorithm to identify influential spreaders.

Does the decoding algorithm violate privacy?

- Prove the noised graph satisfies differential privacy.



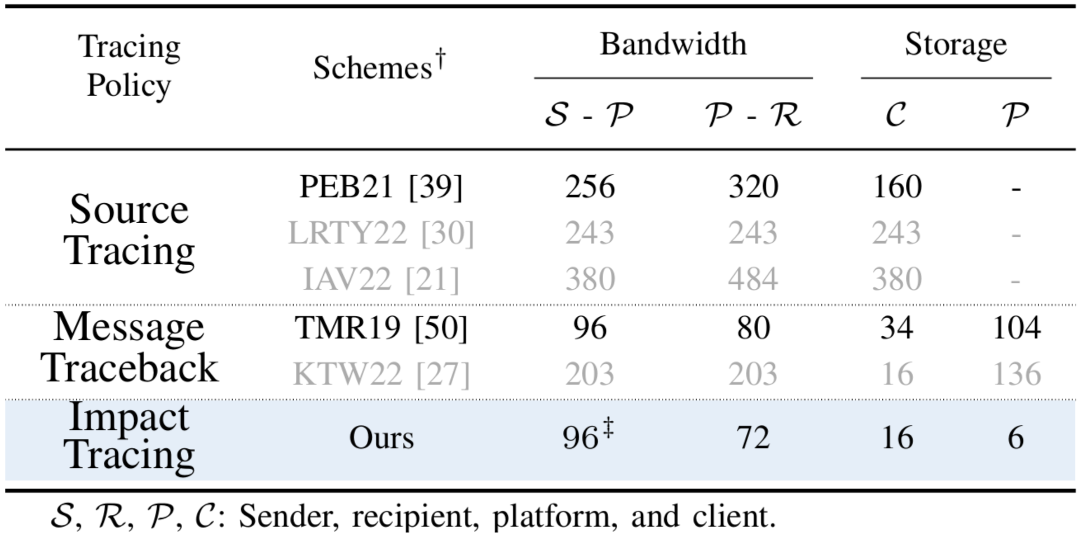
See paper for details.

Security & DP analysis

Experimental analysis

Performance

Bandwidth & Storage (Byte)



Tracing Policy	Schemes†	Bandwidth		Storage	
		$\mathcal{S}$ - $\mathcal{P}$	$\mathcal{P}$ - $\mathcal{R}$	$\mathcal{C}$	$\mathcal{P}$
Source Tracing	PEB21 [39]	256	320	160	-
	LRTY22 [30]	243	243	243	-
	IAV22 [21]	380	484	380	-
Message Traceback	TMR19 [50]	96	80	34	104
	KTW22 [27]	203	203	16	136
Impact Tracing	Ours	$96^{‡}$	72	16	6

$\mathcal{S}$, $\mathcal{R}$, $\mathcal{P}$, $\mathcal{C}$: Sender, recipient, platform, and client.

Store 1 billion messages: 5.6 GB

Performance

Bandwidth & Storage (Byte)

Tracing Policy	Schemes†	Bandwidth		Storage	
		$\mathcal{S}$ - $\mathcal{P}$	$\mathcal{P}$ - $\mathcal{R}$	$\mathcal{C}$	$\mathcal{P}$
Source Tracing	PEB21 [39]	256	320	160	-
	LRTY22 [30]	243	243	243	-
	IAV22 [21]	380	[illegible]	380	-
Message Traceback	TMR19 [50]	96	80	34	104
	KTW22 [27]	203	203	16	136
Impact Tracing	Ours	$96^{‡}$	72	16	6

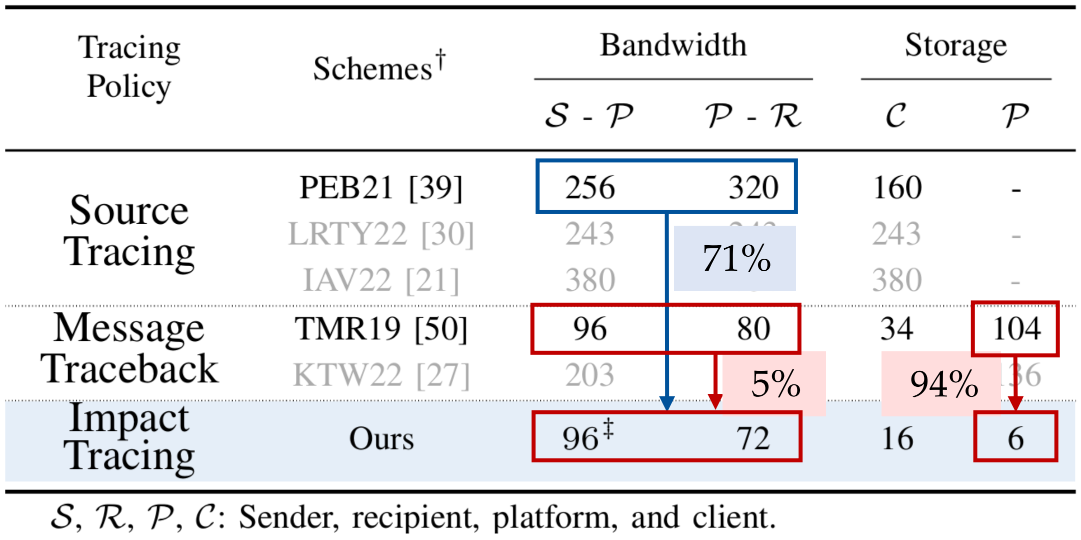
71%

$\mathcal{S}$, $\mathcal{R}$, $\mathcal{P}$, $\mathcal{C}$: Sender, recipient, platform, and client.

Store 1 billion messages: 5.6 GB

Performance

Bandwidth & Storage (Byte)



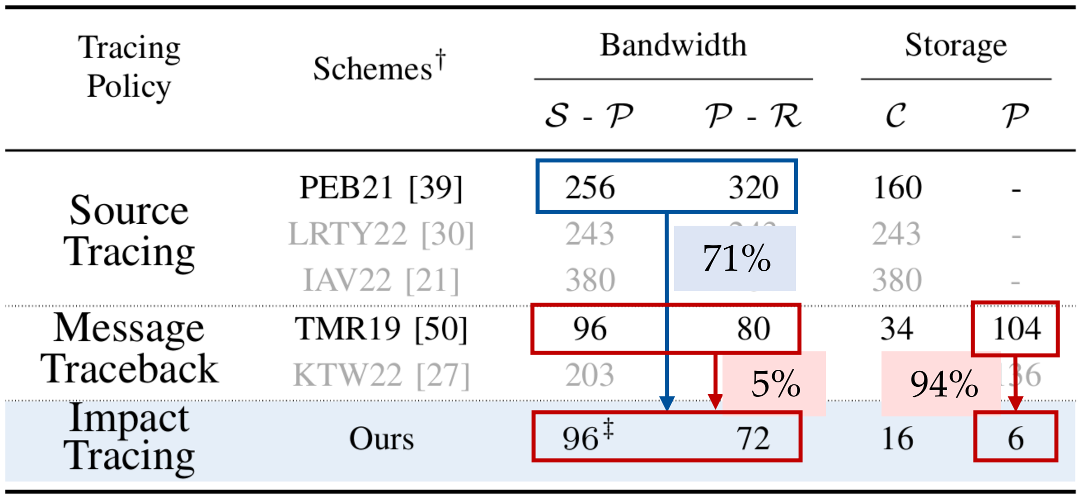
Tracing Policy	Schemes[†]	Bandwidth		Storage	
		$\mathcal{S}$ - $\mathcal{P}$	$\mathcal{P}$ - $\mathcal{R}$	$\mathcal{C}$	$\mathcal{P}$
Source Tracing	PEB21 [39]	256	320	160	-
	LRTY22 [30]	243	[illegible]	243	-
	IAV22 [21]	380	[illegible]	380	-
Message Traceback	TMR19 [50]	96	80	34	104
	KTW22 [27]	203	[illegible]	[illegible]	136
Impact Tracing	Ours	96[‡]	72	16	6

$\mathcal{S}$, $\mathcal{R}$, $\mathcal{P}$, $\mathcal{C}$: Sender, recipient, platform, and client.

Store 1 billion messages: 5.6 GB

Performance

Bandwidth & Storage (Byte)



Tracing Policy	Schemes†	Bandwidth		Storage	
		$\mathcal{S}$ - $\mathcal{P}$	$\mathcal{P}$ - $\mathcal{R}$	$\mathcal{C}$	$\mathcal{P}$
Source Tracing	PEB21 [39]	256	320	160	-
	LRTY22 [30]	243	243	243	-
	IAV22 [21]	380		380	-
Message Traceback	TMR19 [50]	96	80	34	104
	KTW22 [27]	203			136
Impact Tracing	Ours	96‡	72	16	6

71%

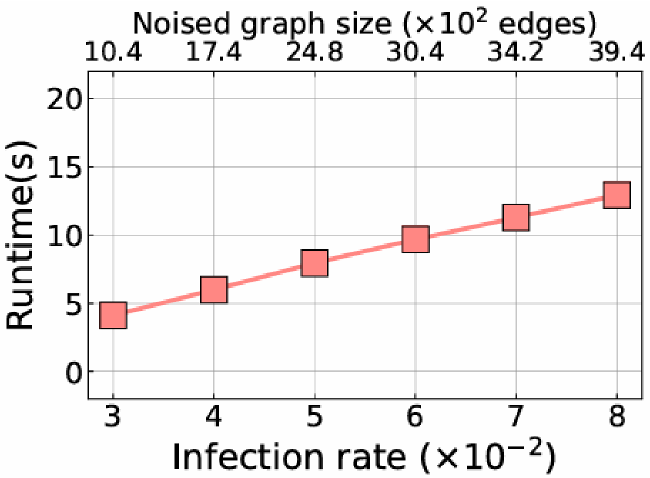
5%

94%

$\mathcal{S}$, $\mathcal{R}$, $\mathcal{P}$, $\mathcal{C}$: Sender, recipient, platform, and client.

Store 1 billion messages: 5.6 GB

Runtime of tracing (s)



Trace graph with 4,000 edges: 15 s

Transmit 1-KB message: 0.3 ms

Performance

Bandwidth & Storage (Byte)

Tracing Policy	Schemes†	Bandwidth		Storage	
		$\mathcal{S}$ - $\mathcal{P}$	$\mathcal{P}$ - $\mathcal{R}$	$\mathcal{C}$	$\mathcal{P}$
Source Tracing	PEB21 [39]	256	320	160	-
	LRTY22 [30]	243	243	243	-
	IAV22 [21]	380	[illegible]	380	-
Message Traceback	TMR19 [50]	96	80	34	104
	KTW22 [27]	203	[illegible]	[illegible]	136
Impact Tracing	Ours	96‡	72	16	6

71%

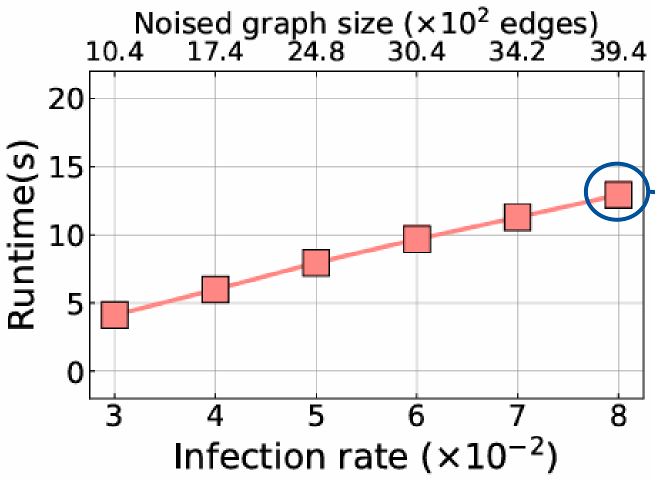
5%

94%

$\mathcal{S}$, $\mathcal{R}$, $\mathcal{P}$, $\mathcal{C}$: Sender, recipient, platform, and client.

Store 1 billion messages: 5.6 GB

Runtime of tracing (s)



Trace graph with 4,000 edges: 15 s

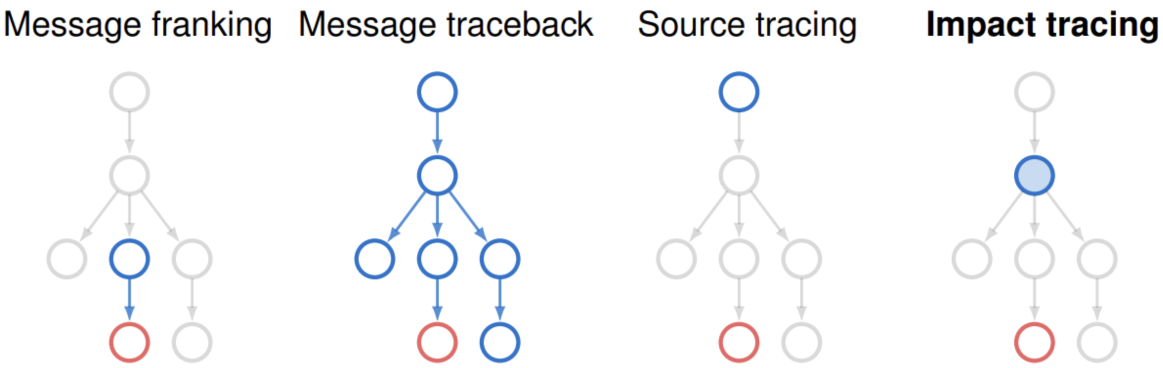
Transmit 1-KB message: 0.3 ms

Summary

Introduce impact tracing: balances traceability and privacy.

Design fuzzy message traceback and decoding algorithm.

Analyze security, privacy, and utility formally.

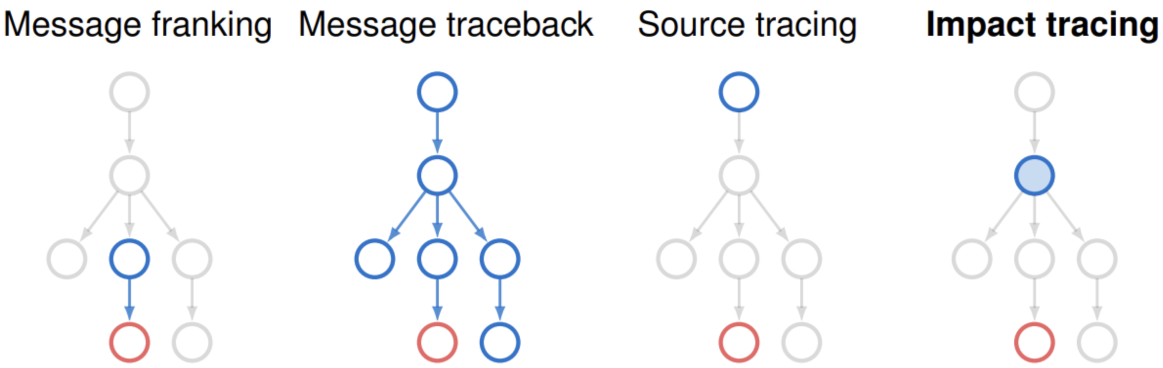


Summary

Introduce impact tracing: balances traceability and privacy.

Design fuzzy message traceback and decoding algorithm.

Analyze security, privacy, and utility formally.



Scan for code.

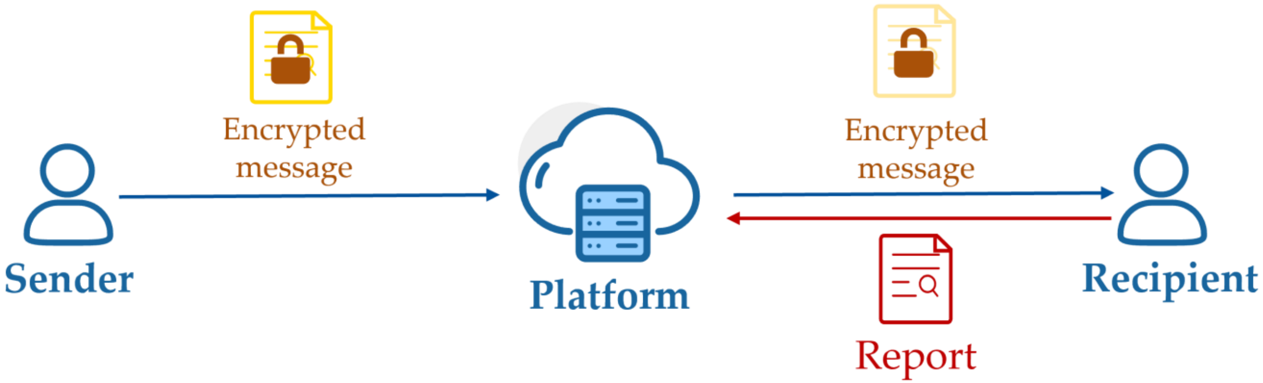
Thank for listening :) Any questions?

Bonus Slides

Message Franking

Message franking enables a recipient to <u>report a message</u>.

And, the platform can <u>authenticate</u> that the sender actually sent it.



Accountability: 1) Recipients cannot smear honest senders.

2) Senders cannot evade reporting.

Confidentiality: The platform learn nothing about unreported messages.

[Millican17,RWC] Challenges of E2E Encryption in Facebook Messenger

Simulation on Real-World Datasets

Influence evaluation

1. Simulate forwarding using SIR model.

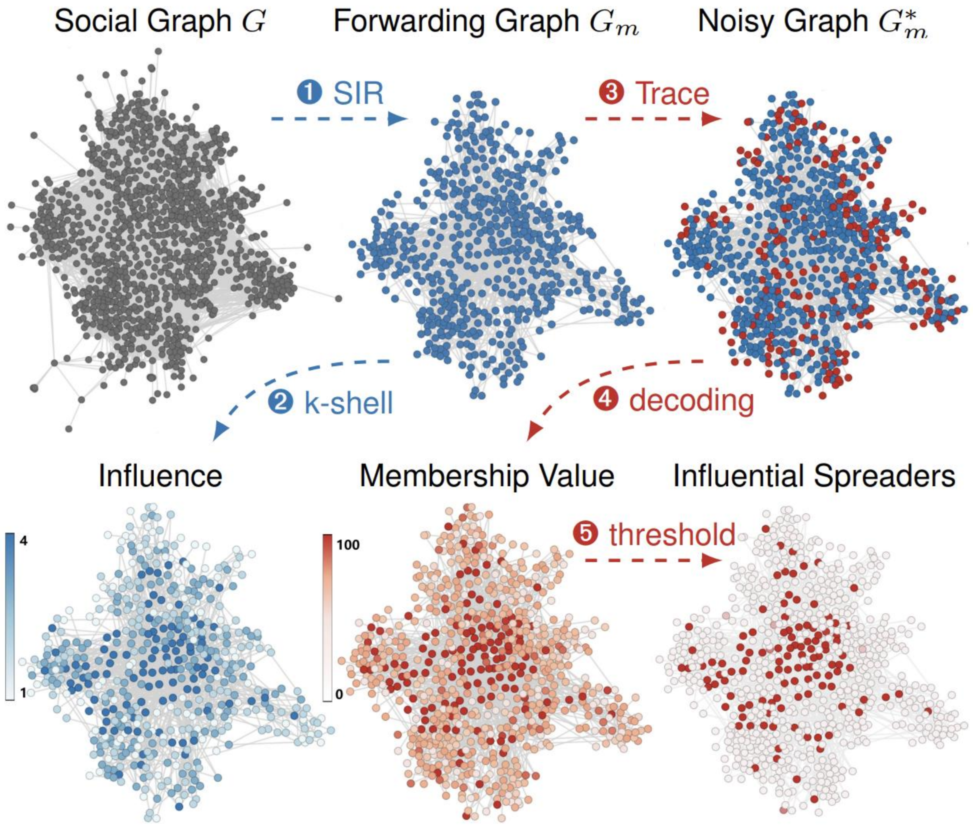
2. Evaluate vertices' impact using k-shell.

Impact tracing

3. Trace with fuzzy message traceback.

4. Decode the traced noisy graph.

5. Output a set of influential spreaders.



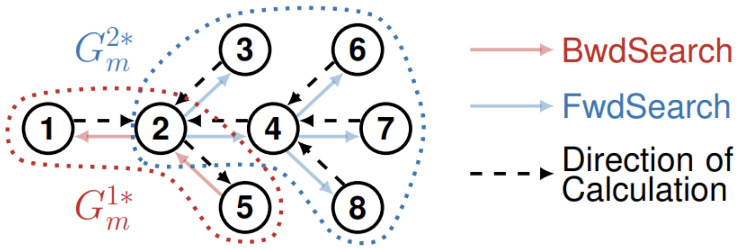
The Design: Decoding the Result

Decoding algorithm: Computes a fuzzy value for each vertex.

- The false positives of a vertex satisfy binomial distribution.

- A vertex has *only* one true precursor.

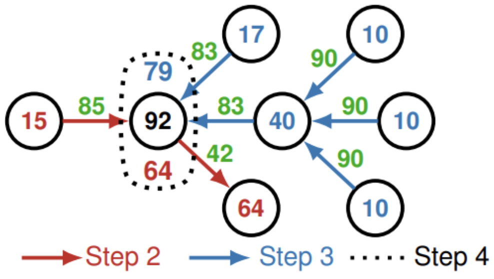
- A vertex is true positive if one of its descendants is true.



(a) llustration of G_m^{1*} and G_m^{2*}

Graph	Vertex	$n(v)$	$n^*(v)$	$a_v(\%)$
G_m^{1*}	v(2)	50	1	85
G_m^{2*}	v(2)	50	2	83
	v(4)	100	3	90
	v(5)	10	1	42

(b) Step 1: caculate α_v



(c) Step 2-4: caculate β_v^1, β_v^2, and μ_v

IAS-DP: Defining Privacy

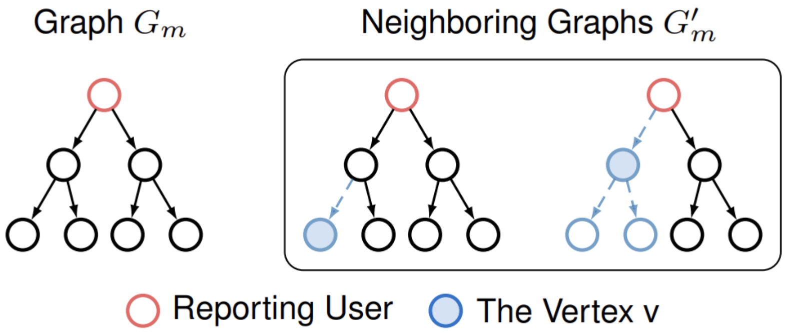
Individualized Asymmetric Subtree Differential Privacy

- **Individualized**: Privacy budgets vary per user.
- **Asymmetric**: Traceability adopts one-side noise.
- **Subtree**: All impacts caused by one user on one forwarding graph.

Definition 2 (ε_v-IAS-DP). A randomized algorithm $\mathcal{M}$ is ε_v-IAS-DP if given a graph G the following equation holds for any $S \in Range(\mathcal{M})$ and neighboring subgraph pair (G_m, G'_m), where G'_m is obtained by removing a subtree $tree(v)$ in G_m.

$$\Pr\left[\mathcal{M}(G_m) = S\right] \leq e^{\varepsilon_v} \cdot \Pr[\mathcal{M}(G'_m) = S],$$

where ε_v is the privacy budget for vertex v.



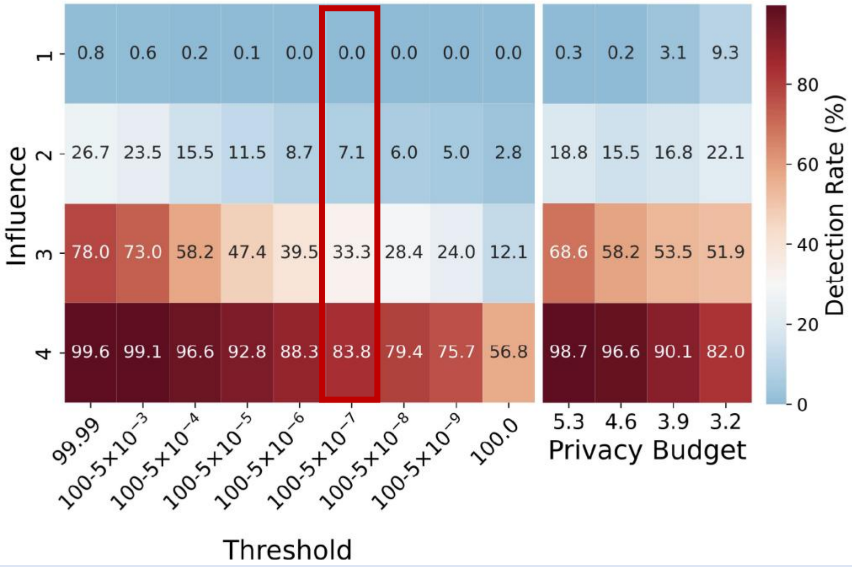
Theorem 1. The fuzzy message traceback scheme satisfies ε_v-IAS-DP, where $\boxed{\varepsilon_v = \ln(1/\psi^n)}$, n is the number of edges in $tree(v)$, and ψ is the FPR of random response.

Evaluating Utility & Privacy

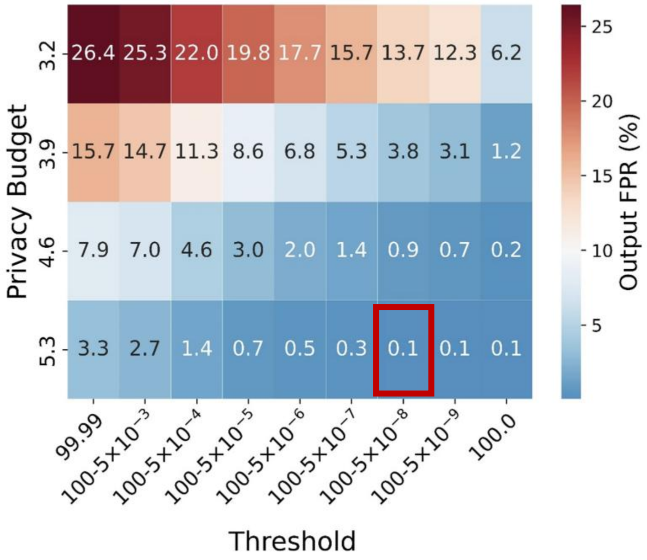
Detection rate: The output contains as most influential users as possible.

Output FPR: The output should contain as few false positives as possible.

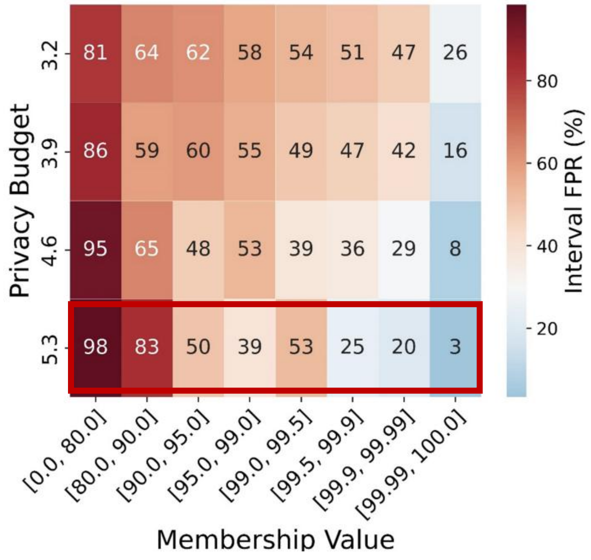
Interval FPR: Non-influential users should be hidden by sufficient noise.



Influence	99.99	$100-5\times10^{-3}$	$100-5\times10^{-4}$	$100-5\times10^{-5}$	$100-5\times10^{-6}$	$100-5\times10^{-7}$	$100-5\times10^{-8}$	$100-5\times10^{-9}$	100.0	5.3	4.6	3.9	3.2
1	0.8	0.6	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.3	0.2	3.1	9.3
2	26.7	23.5	15.5	11.5	8.7	7.1	6.0	5.0	2.8	18.8	15.5	16.8	22.1
3	78.0	73.0	58.2	47.4	39.5	33.3	28.4	24.0	12.1	68.6	58.2	53.5	51.9
4	99.6	99.1	96.6	92.8	88.3	83.8	79.4	75.7	56.8	98.7	96.6	90.1	82.0



Privacy Budget	99.99	$100-5\times10^{-3}$	$100-5\times10^{-4}$	$100-5\times10^{-5}$	$100-5\times10^{-6}$	$100-5\times10^{-7}$	$100-5\times10^{-8}$	$100-5\times10^{-9}$	100.0
3.2	26.4	25.3	22.0	19.8	17.7	15.7	13.7	12.3	6.2
3.9	15.7	14.7	11.3	8.6	6.8	5.3	3.8	3.1	1.2
4.6	7.9	7.0	4.6	3.0	2.0	1.4	0.9	0.7	0.2
5.3	3.3	2.7	1.4	0.7	0.5	0.3	0.1	0.1	0.1



Privacy Budget	[0.0, 80.0]	[80.0, 90.0]	[90.0, 95.0]	[95.0, 99.0]	[99.0, 99.5]	[99.5, 99.9]	[99.9, 99.99]	[99.99, 100.0]
3.2	81	64	62	58	54	51	47	26
3.9	86	59	60	55	49	47	42	16
4.6	95	65	48	53	39	36	29	8
5.3	98	83	50	39	53	25	20	3

Identifies 84% of the most influential spreaders and no the least influential users with 99.9% correctness (i.e., less than 0.25 false positives on average).