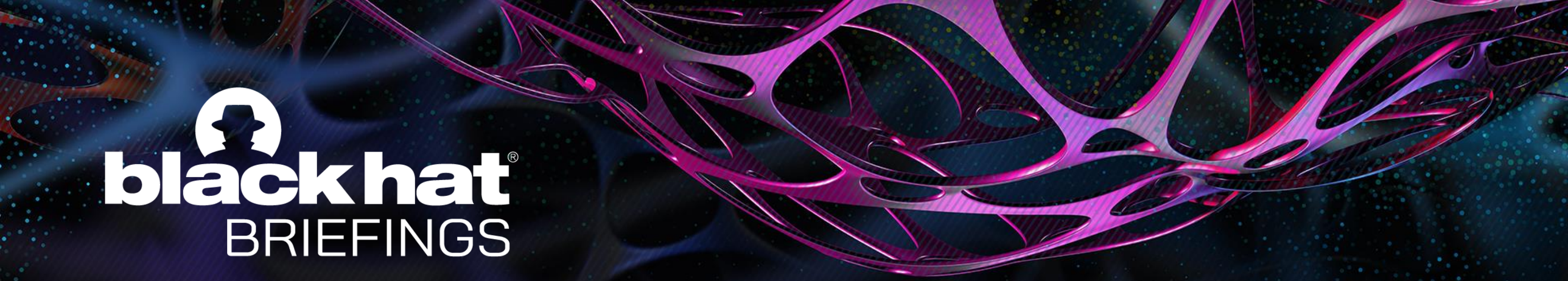


black hat BRIEFINGS
DECEMBER 10-11, 2025
EXCEL LONDON / UNITED KINGDOM

LINE-Break:

Cryptanalysis and Reverse Engineering of Letter Sealing

Diego F. Aranha, Thomas Kingo T. Mogensen, Adam Blatchley Hansen

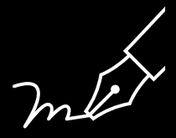
Computer Science @ Aarhus University, DK

linebreak.info

Sending a message

Letter → Envelope → Signature / Signet Ring → Postal Service






Message → Encryption → Digital Signature




What is LINE?

- *Super App*
- LY Corp. HQ in Japan *(Line-Yahoo)*
- 200M monthly users (1B, yearly):
  - Japan: 89M = 85% of pop.
  - Thailand: 51M
  - Taiwan: 21M = 90% of pop.
  - Indonesia: 13M

"*Letter Sealing implies all **end-to-end encrypted (E2EE)** protocols integrated in LINE's messaging [...] Although the first version of Letter Sealing encountered message data integrity issues, Letter Sealing v2 guarantees stronger protection over messages, **with all the issues resolved***"

Letter Sealing V1 x V2

	Version 1	Version 2
Key exchange algorithm	ECDH over Curve25519[5]	
Message encryption algorithm	AES256-CBC	AES256-GCM[6]
Message hash function	SHA-256	N/A
Data authentication	AES-ECB with SHA-256 MAC	AES256-GCM
Message data	Encryption and integrity	
Message metadata	Not protected	Integrity

Source: LINE Encryption Overview – Technical Whitepaper

Where is Letter Sealing applied?

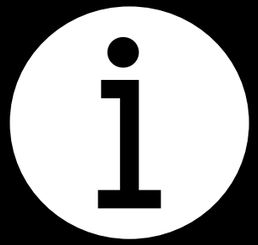
Message/media type	2015	2016	2017/9	2018/4	2019/10	2020/9	2021/9	2022/8	2024/12
Text	○	○ → ◎	◎	◎	◎	◎	◎	◎	◎
Location	○	○ → ◎	◎	◎	◎	◎	◎	◎	◎
Stickers *2	△	△	○	○	○	○	○	○	○
Image files *3	△	△	○	○	○	○	○	○	◎*5
Voice messages *4	x	x	○	○	○	○	○	○	◎
Video files *4	x	x	○	○	○	○	○	○	◎*5
Other files *3	△	△	○	○	○	○	○	○	◎
Message reaction	n/a	n/a	n/a	n/a	n/a	n/a	○	○	○

Legend: ◎ Letter Sealing (end-to-end encryption) / ○ Transport-level encryption / △ Partially protected / x Not encrypted or not sufficiently encrypted

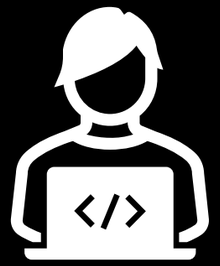
Source: LINE Encryption Report 2024

Presentation Overview

End-to-End
Encryption



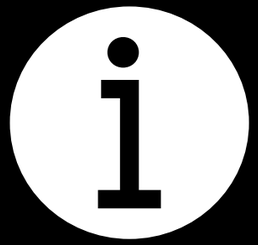
Protocol +
Implementation



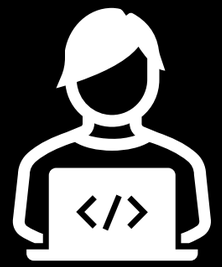
The fun!
+ demos

Presentation Overview

End-to-End Encryption



Protocol + Implementation



The fun! + demos

What is E2EE?

No clear **consensus** definition.

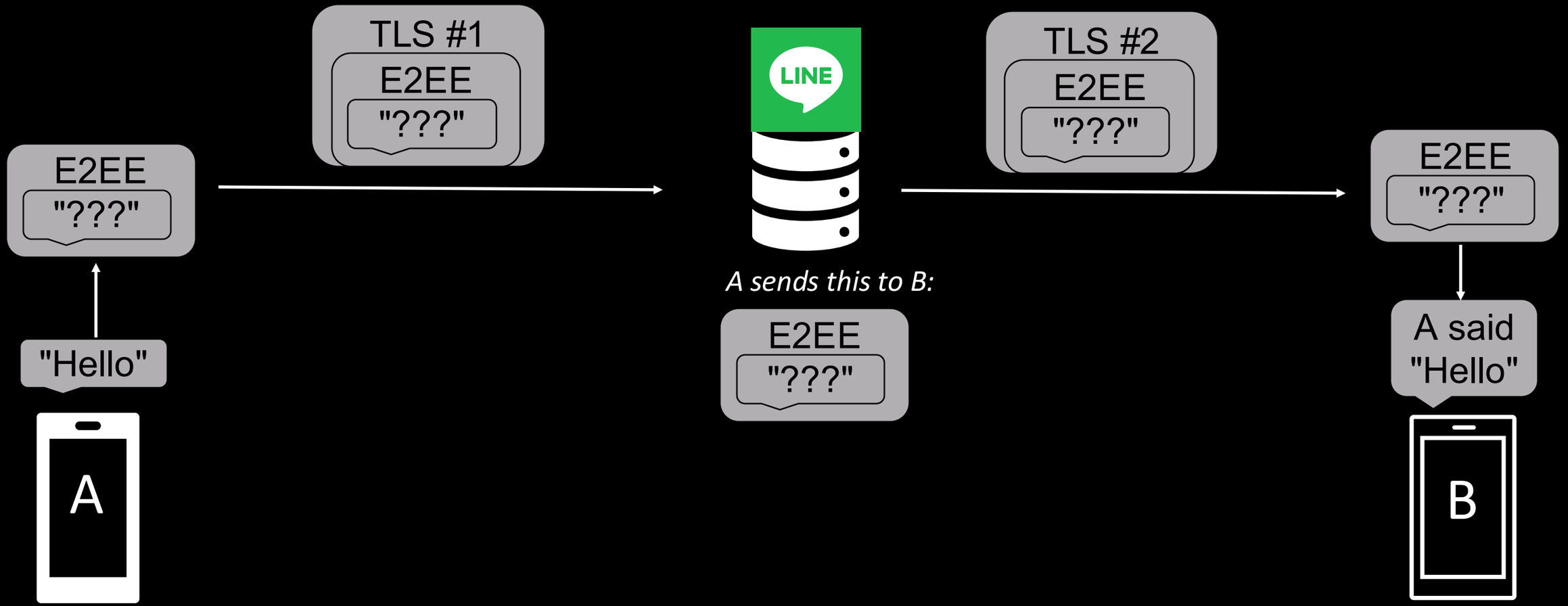
Confidentiality

Integrity

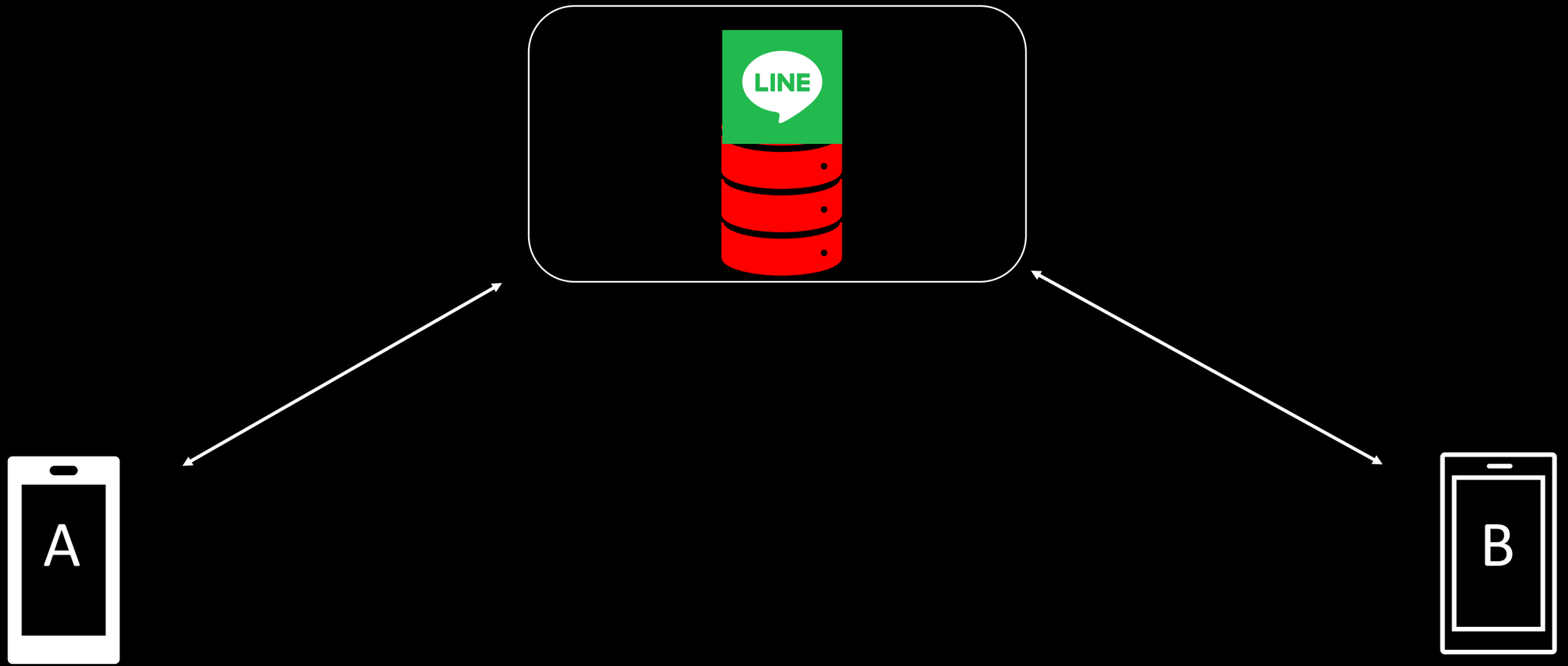
Authentication

Forward Secrecy

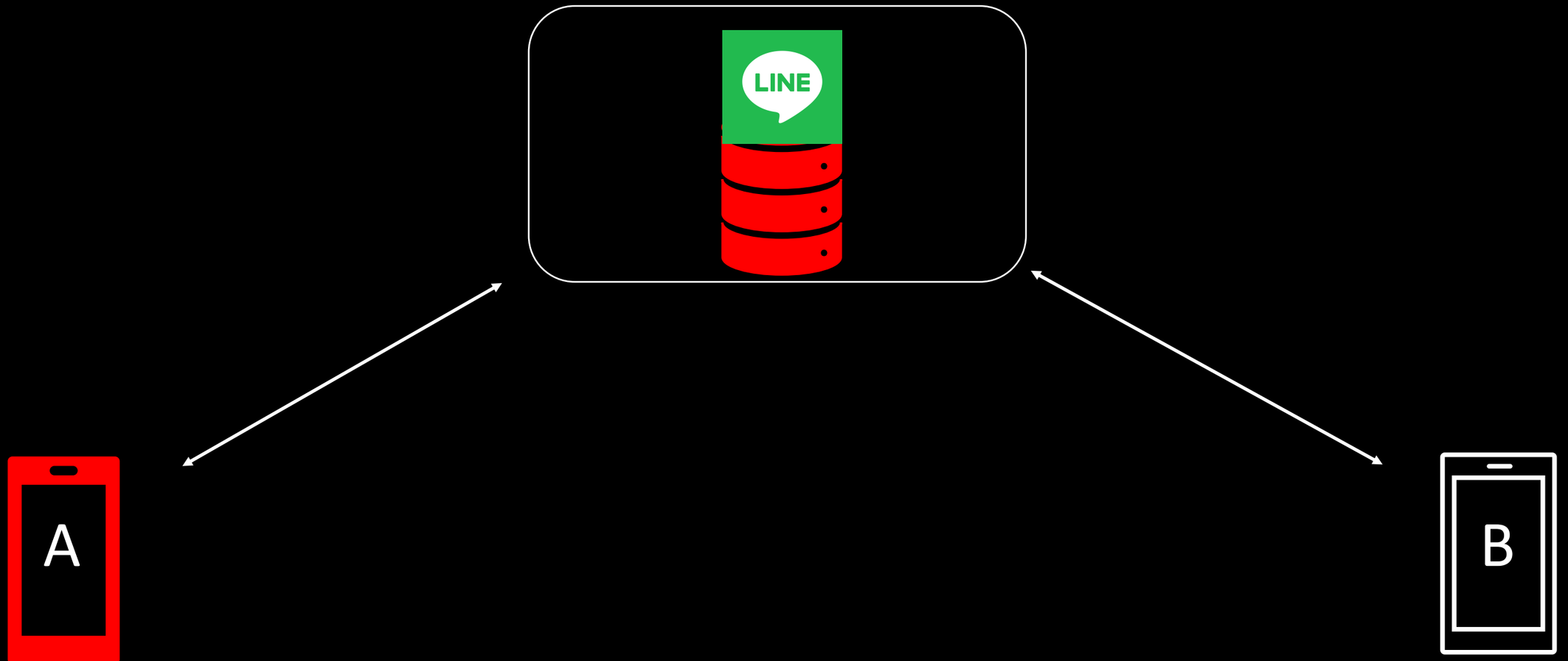
TLS + E2EE


TLS #1
E2EE
"???"
TLS #2
E2EE
"???"
LINE
E2EE
"???"
E2EE
"???"
A said
"Hello"
"Hello"
A
B
A sends this to B:
E2EE
"???"

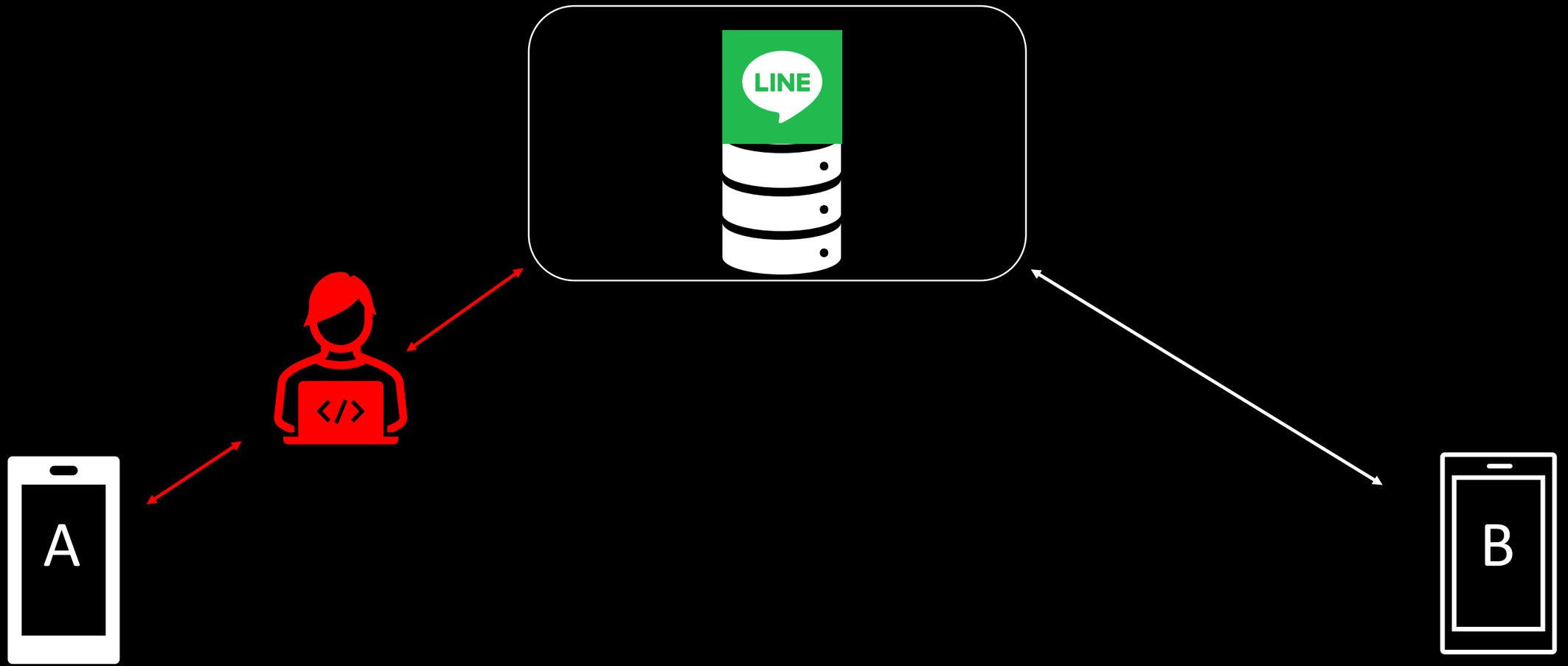
E2E Adversary / Insider



Malicious User + E2E adversary

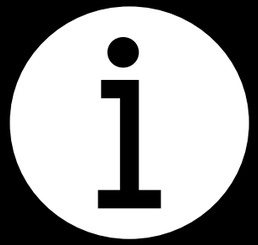


Man-in-the-Middle (MitM)

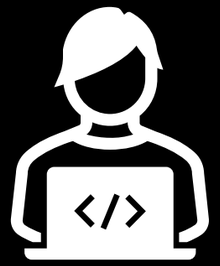


Presentation Overview

End-to-End Encryption

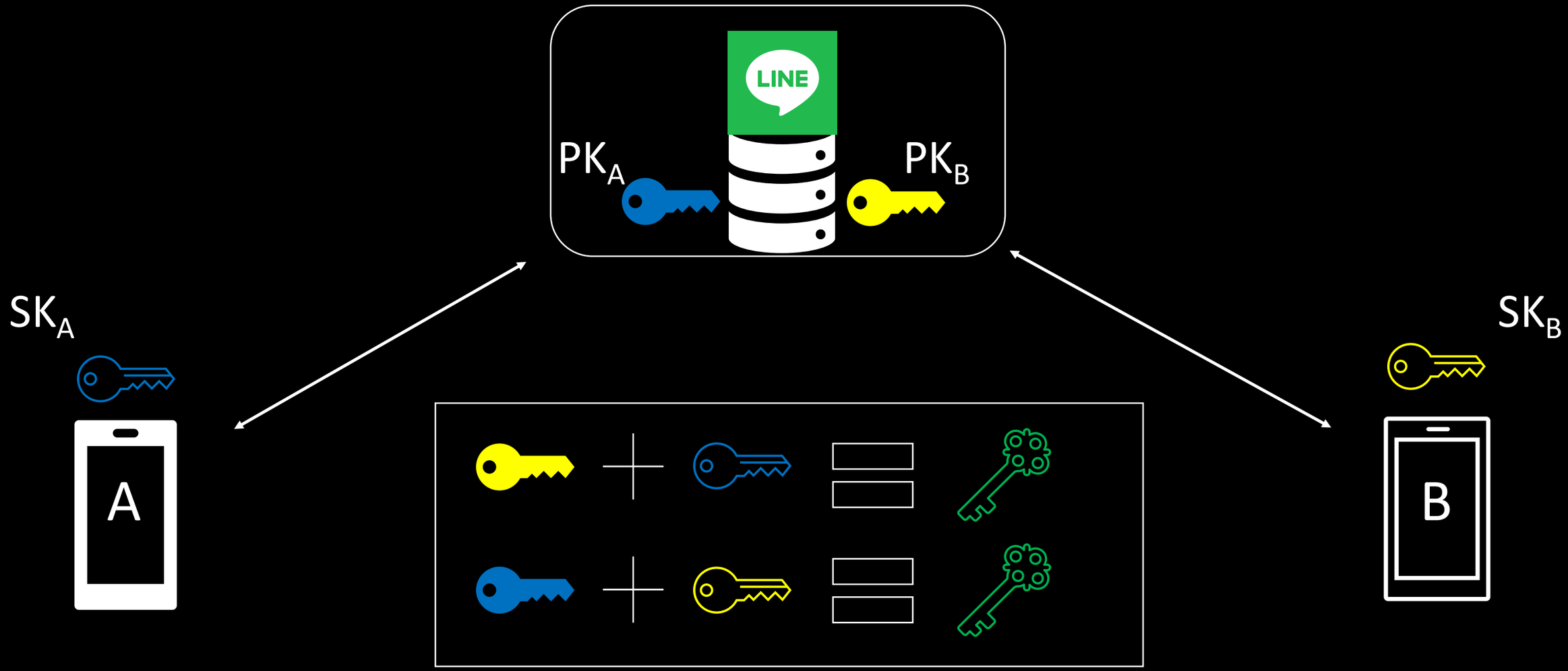


Protocol + Implementation

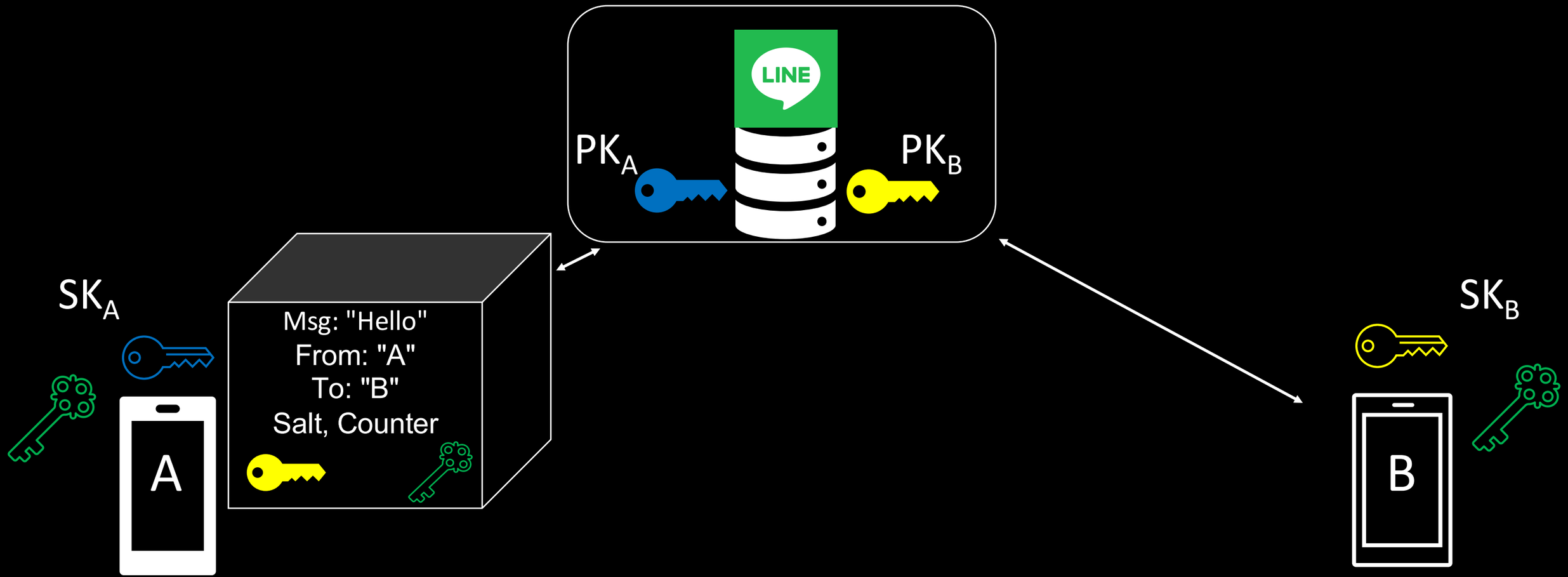


The fun! + demos

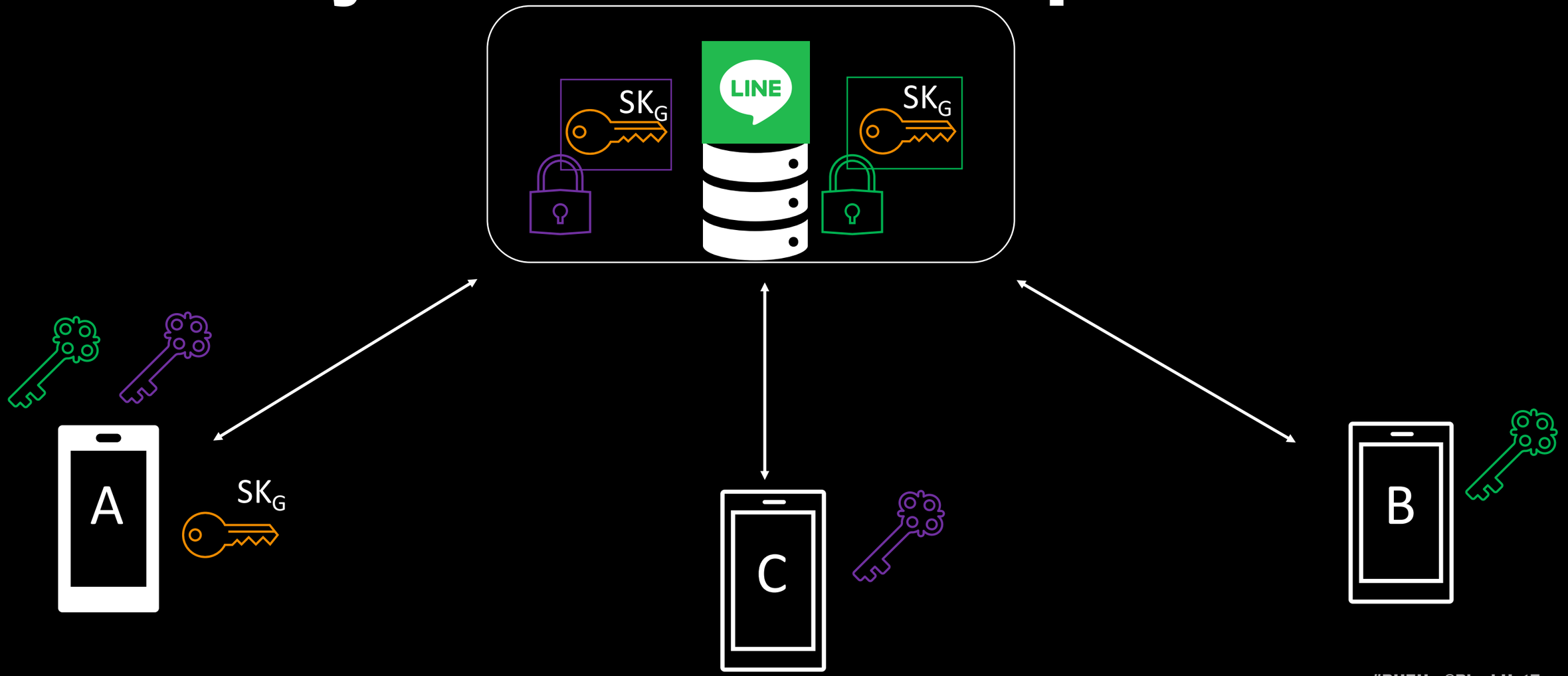
Key Generation – One-to-One chats



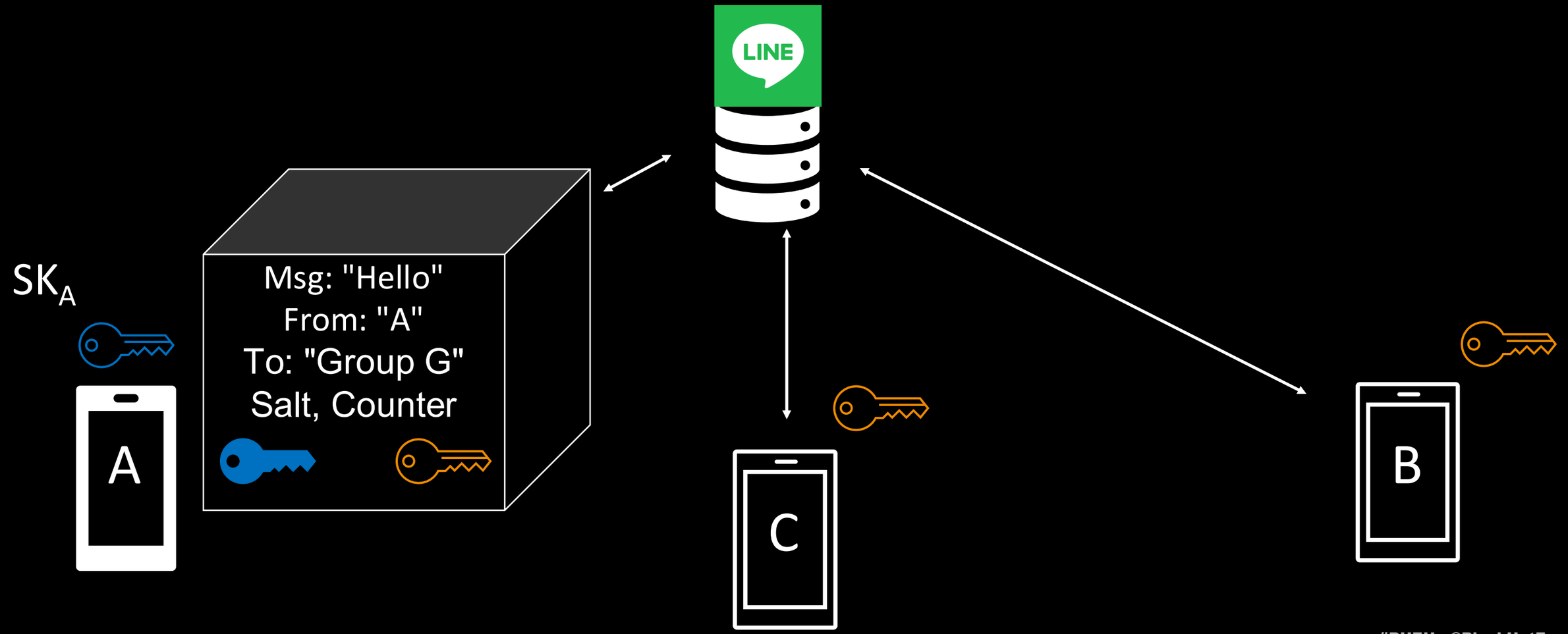
Encryption – One-to-One chats



Key Generation – Group chats



Encryption – Group chats



Encryption in LSv2

- One-to-one chats:

$$Secret = ECDH(pk_{U_j}, sk_{U_i}) = ECDH(pk_{U_i}, sk_{U_j})$$
$$salt = randombytes(16)$$
$$key = SHA256(Secret||salt||"Key")$$
$$nonce(12) = counter[8]||randombytes(4)$$
$$AAD = recipID||senderID||senderKeyID||recipKeyID||version||type$$
$$(C, tag) = AES_{GCM}(key, nonce, M, AAD)$$
$$Msg = version||type||salt||C||tag||nonce||senderKeyID||recipKeyID$$

- *Key Material generated **ONCE***

- Group chats:

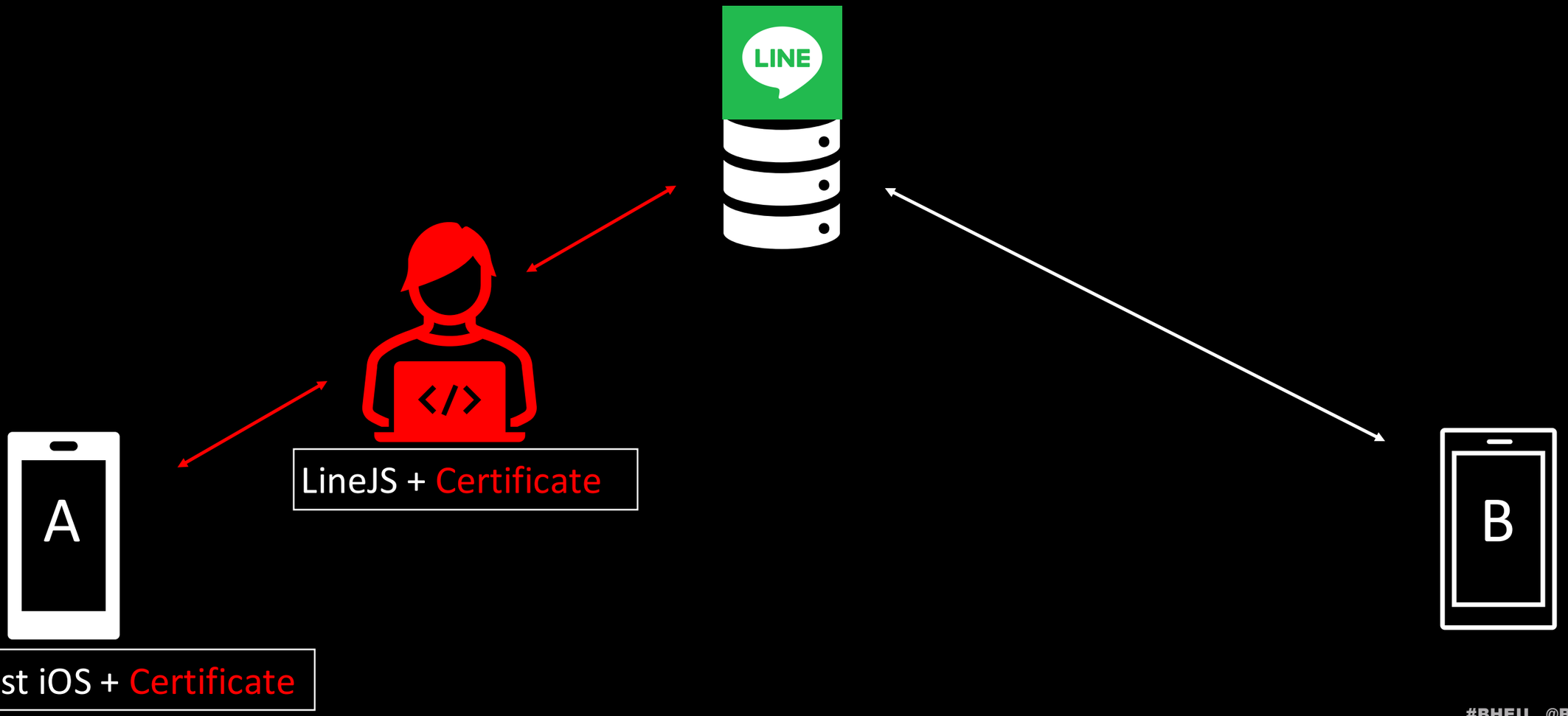
$$Secret = ECDH(sk_g, pk_{U_i})$$
$$salt = randombytes(16)$$
$$key = SHA256(Secret||salt||''Key'')$$
$$nonce(12) = counter[8]||randombytes(4)$$
$$AAD = groupID||senderID||senderKeyID||groupKeyID||version||type$$
$$(C, tag) = AES_{GCM}(key, nonce, M, AAD)$$
$$Msg = version||type||salt||C||tag||nonce||senderKeyID||groupKeyID$$

- *Group Key regenerated at group update*

Overview of findings

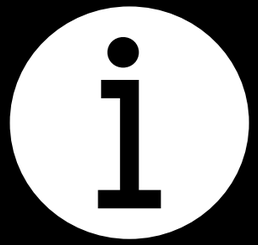
Target	Attack	Threat Model
All chats	Replay, reorder, blocking	E2E or MitM attacker
	Impersonation	Malicious user with E2E/MitM
	Plaintext leakage (stickers + URLs)	E2E or MitM attacker

Experimental Setup

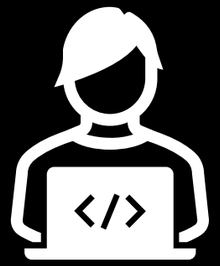


Presentation Overview

End-to-End Encryption

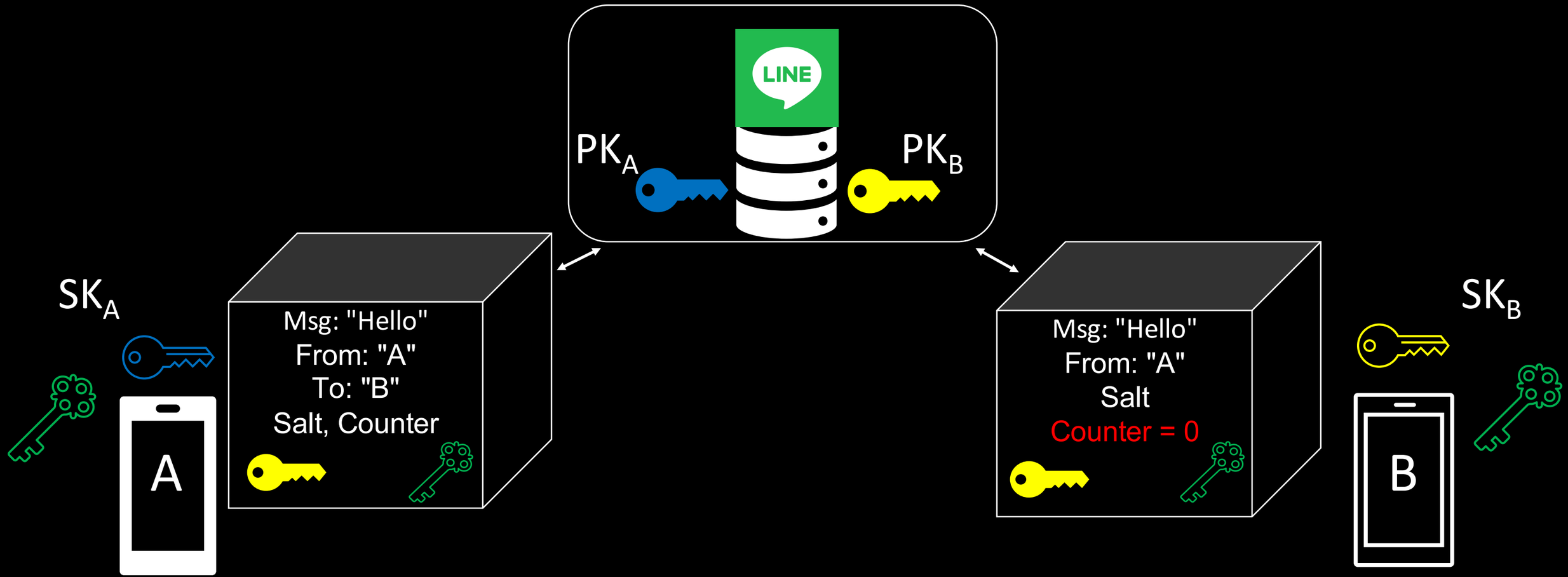


Protocol + Implementation

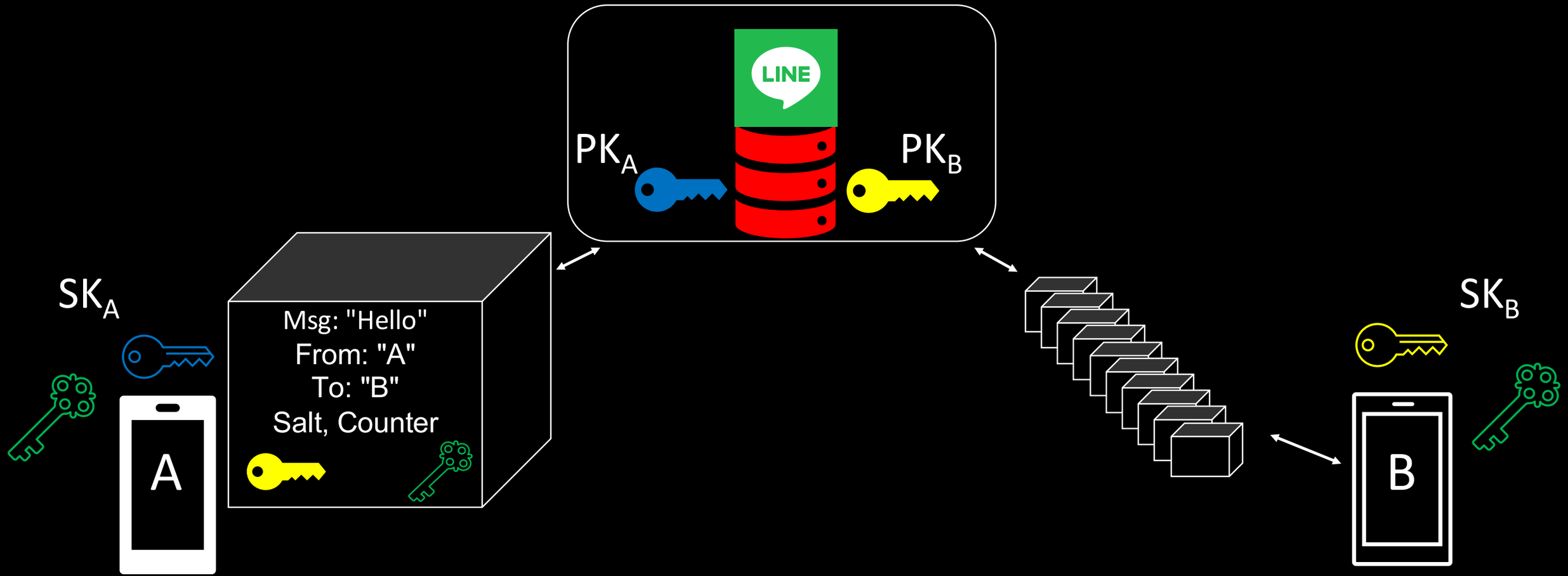


The fun!
+ demos

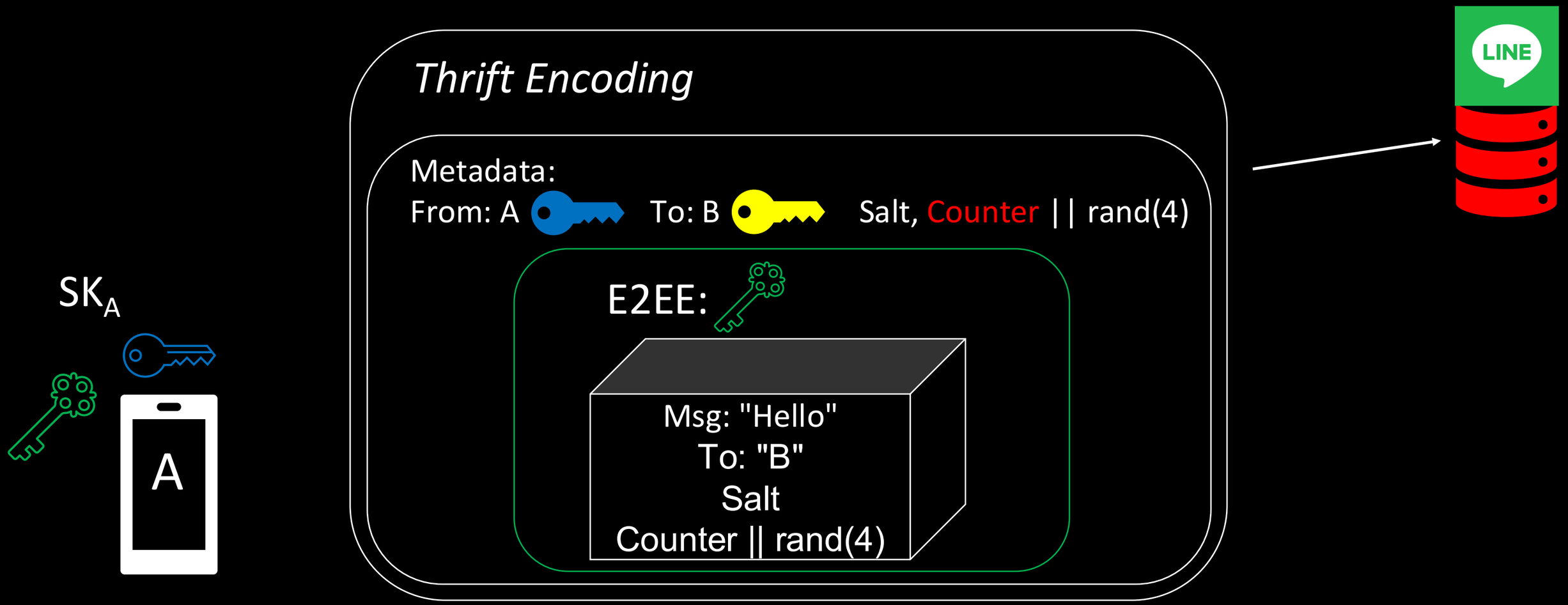
Replay attack - overview



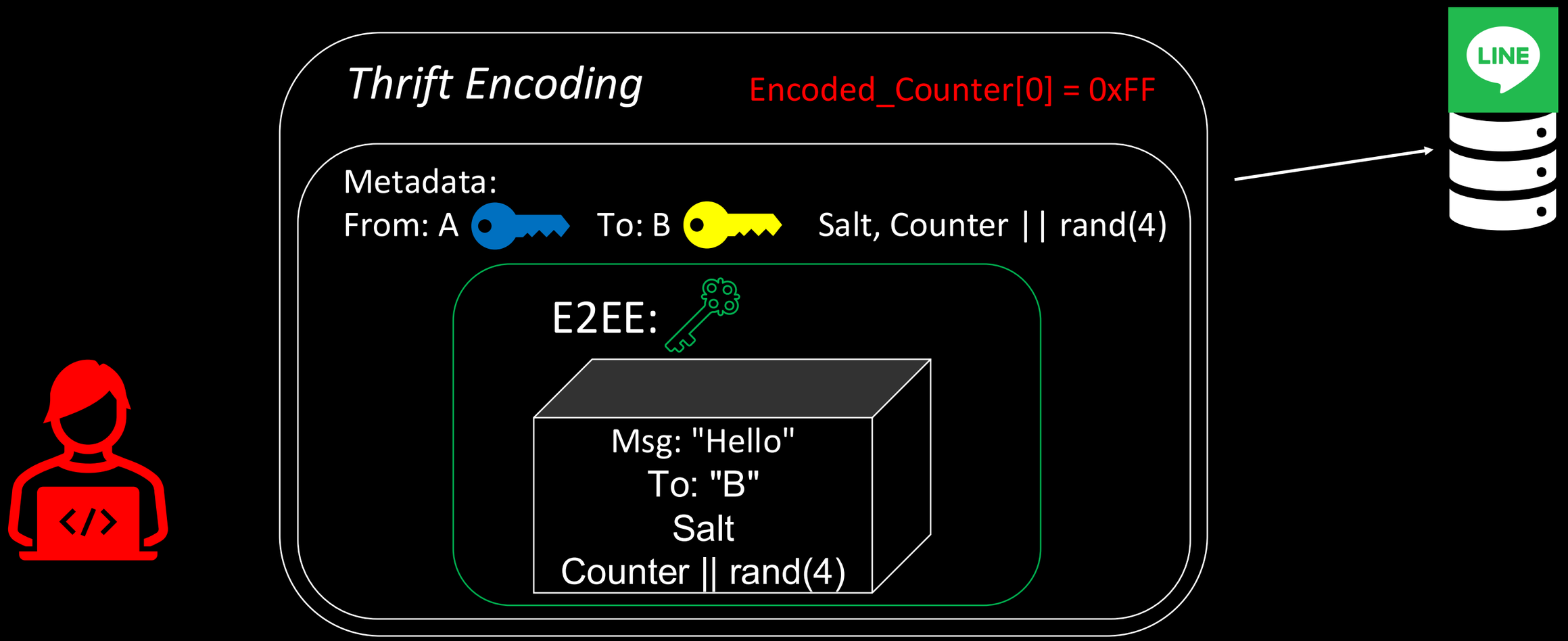
Replay attacks - overview



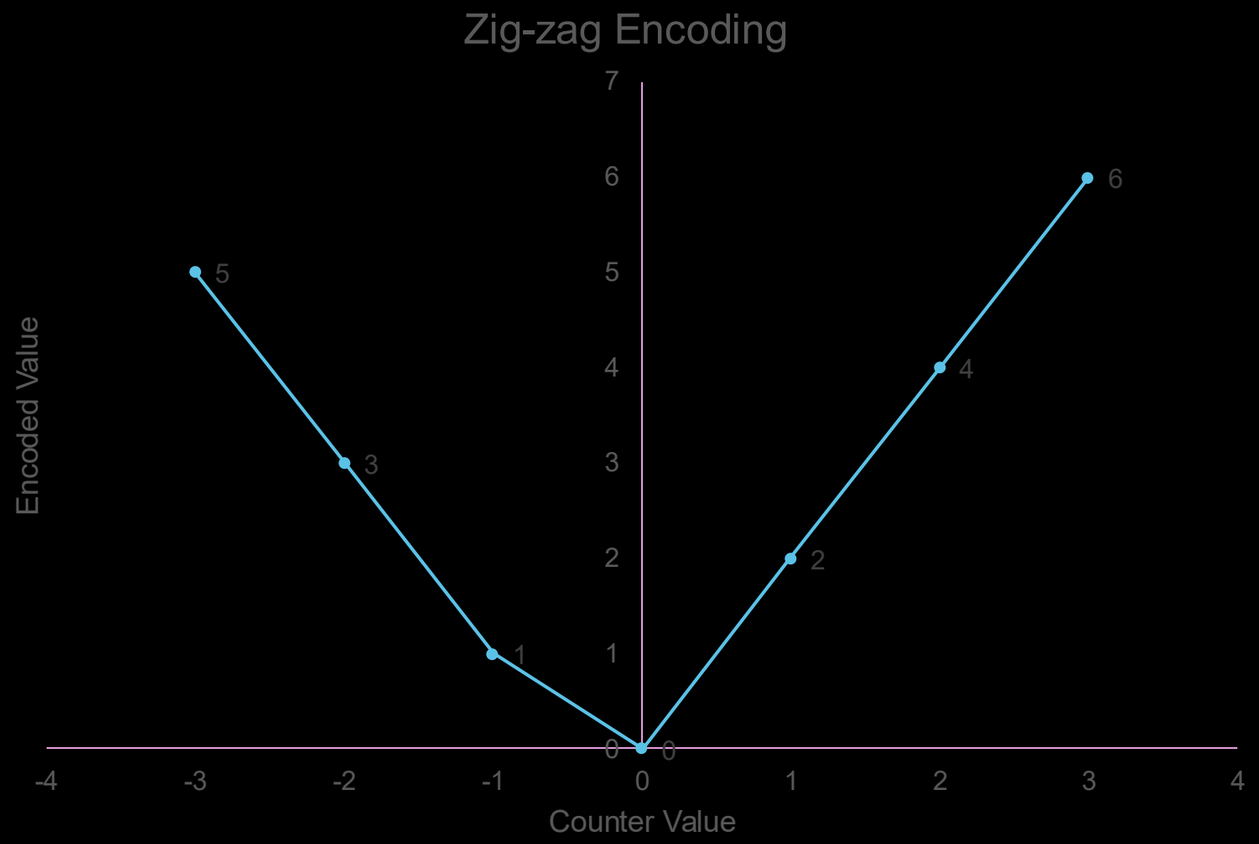
Replay attack – technical details



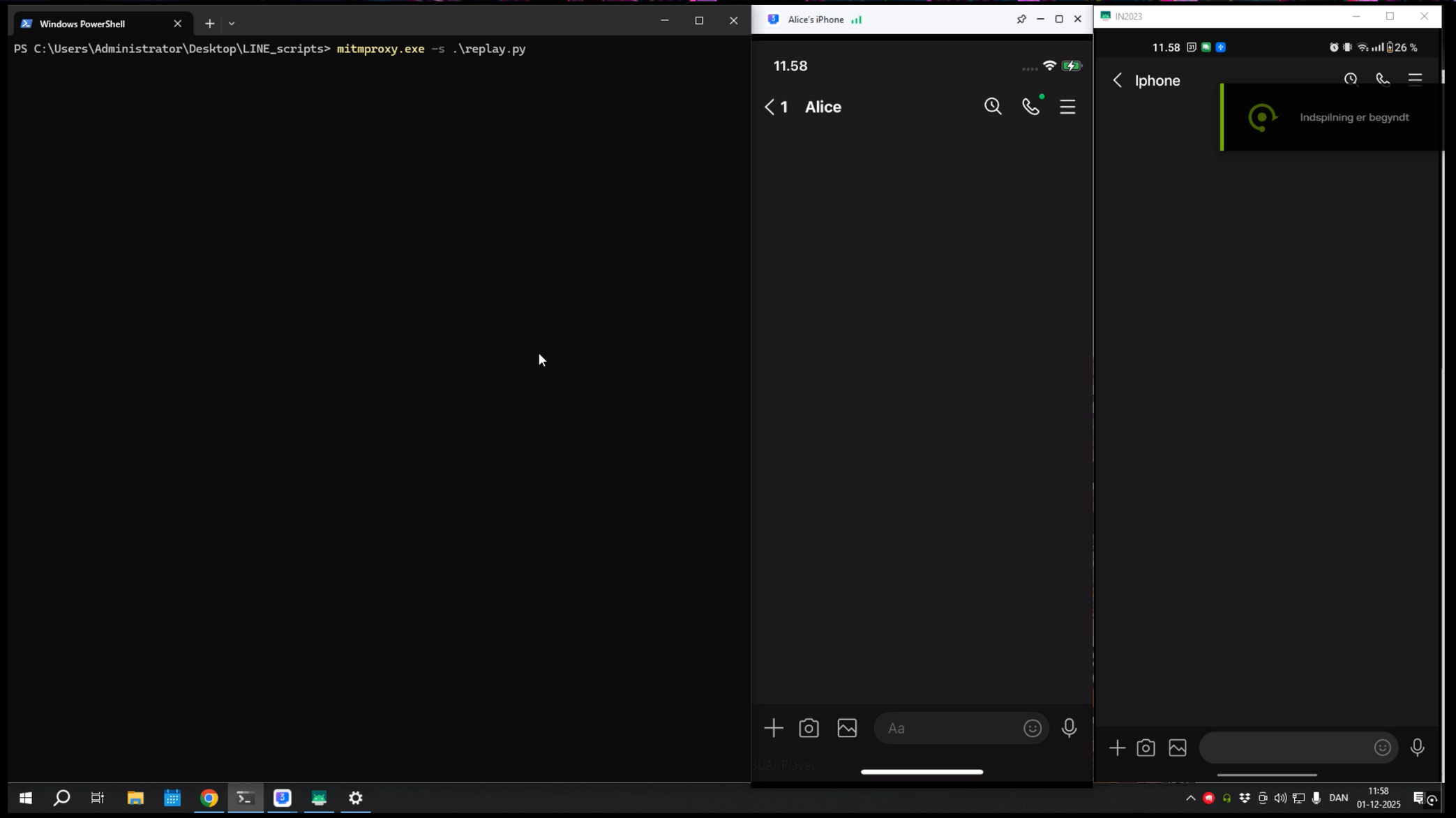
Replay attack – subverting mitigations



Thrift encodings

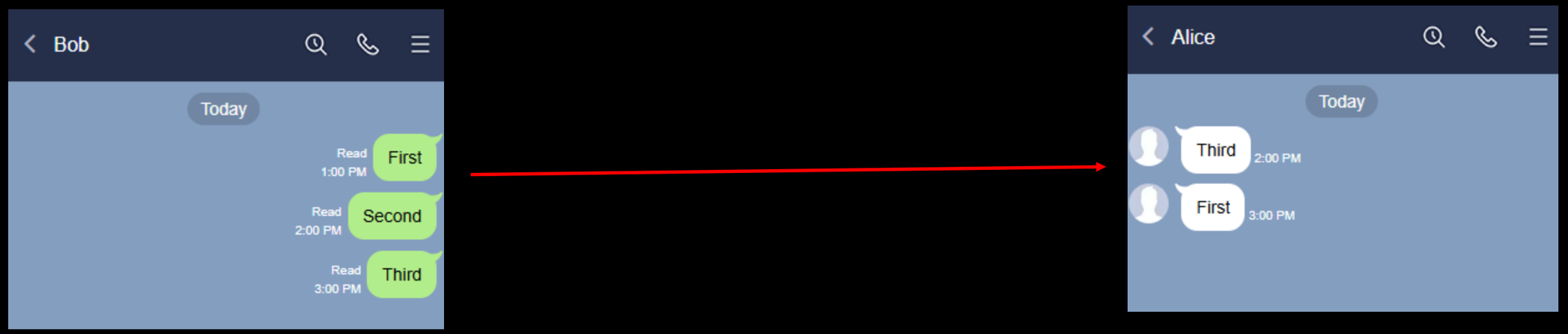


Input	Encoded
Positive	Even
0	0
Negative	Odd


Windows PowerShell
PS C:\Users\Administrator\Desktop\LINE_scripts> mitmproxy.exe -s .\replay.py
Alice's iPhone
11.58
1 Alice
Aa
IN2023
11.58
26 %
Iphone
Indspilning er begyndt
DAN
11:58
01-12-2025

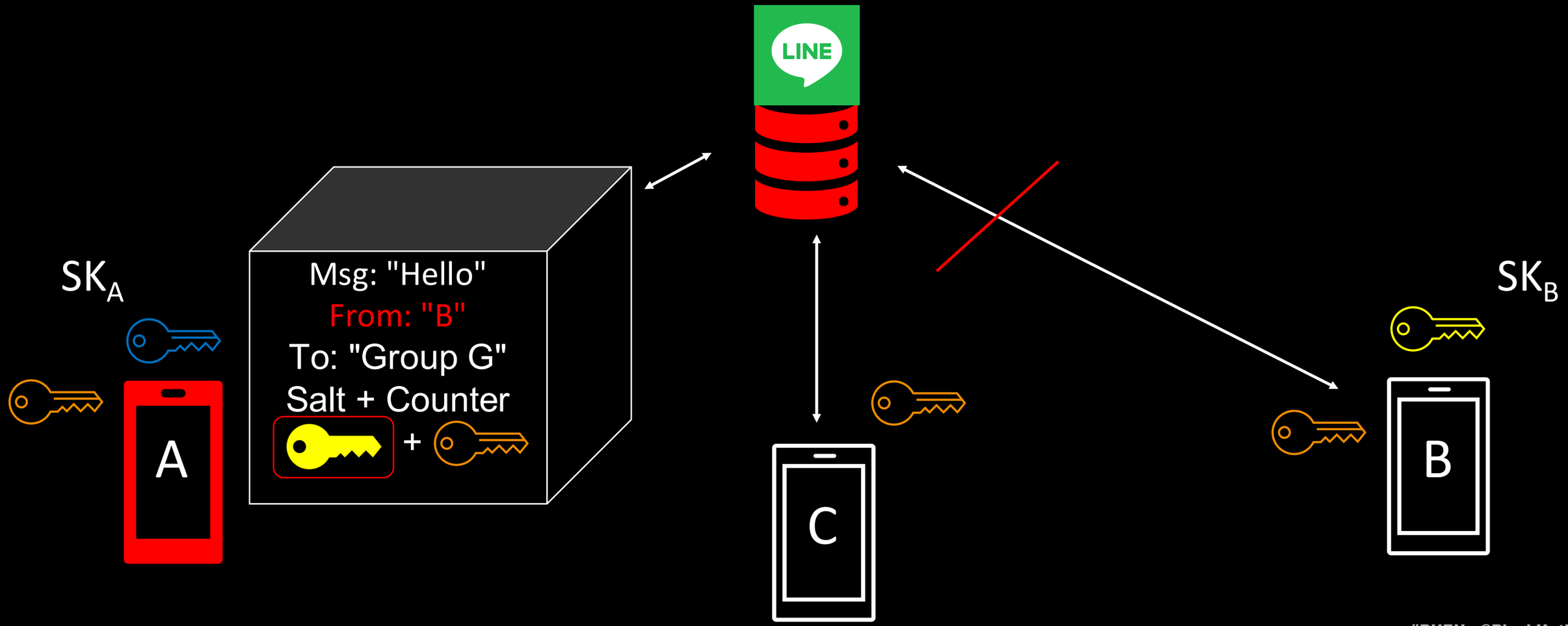
Reorder and blocking attacks

- There is no way to detect **reordered** messages.
- There is no way to detect **dropped** messages.

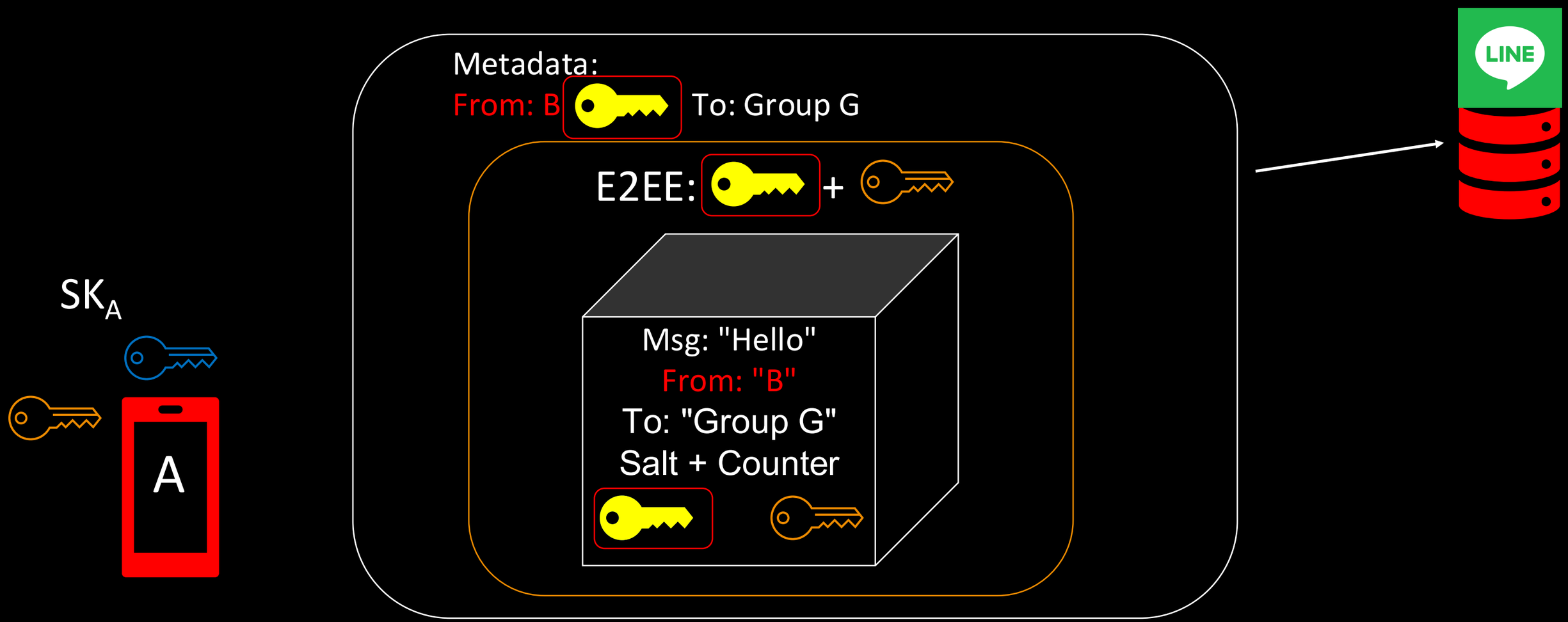


- *Counters could be used to ensure consistent state!*

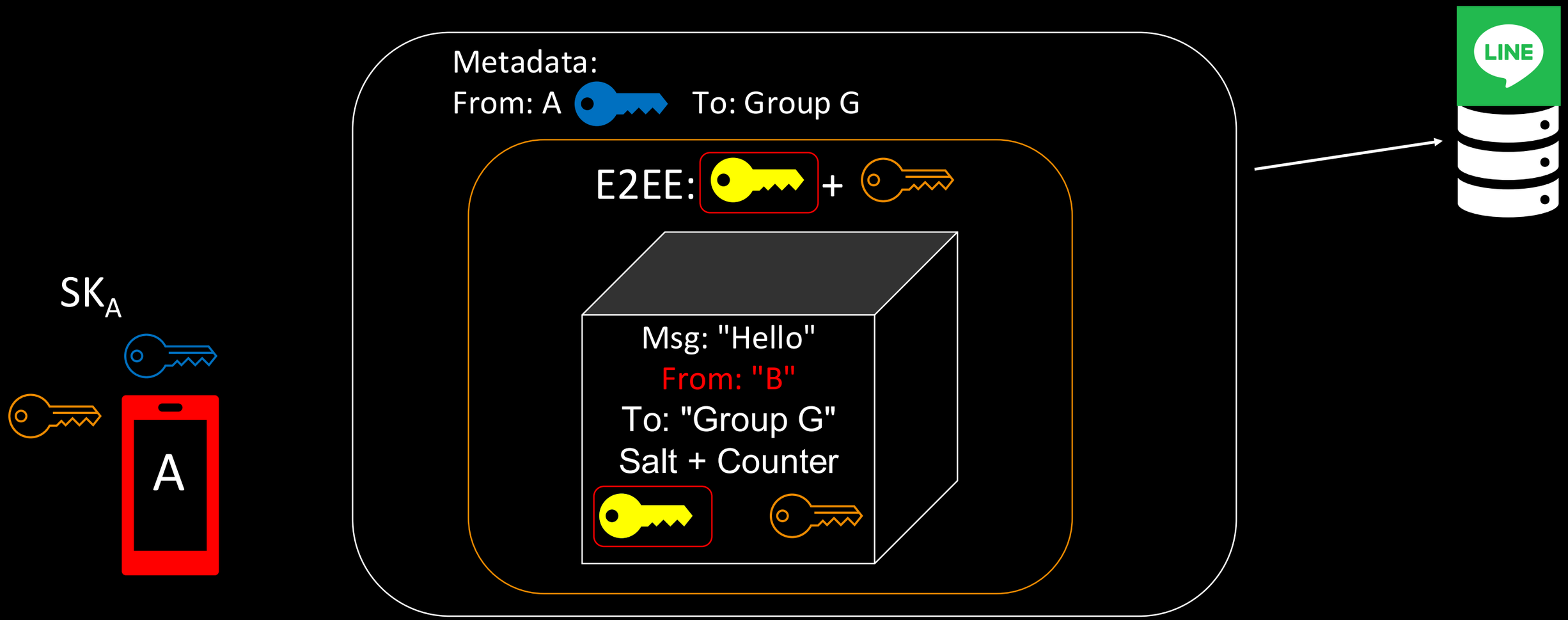
Impersonation attack - overview



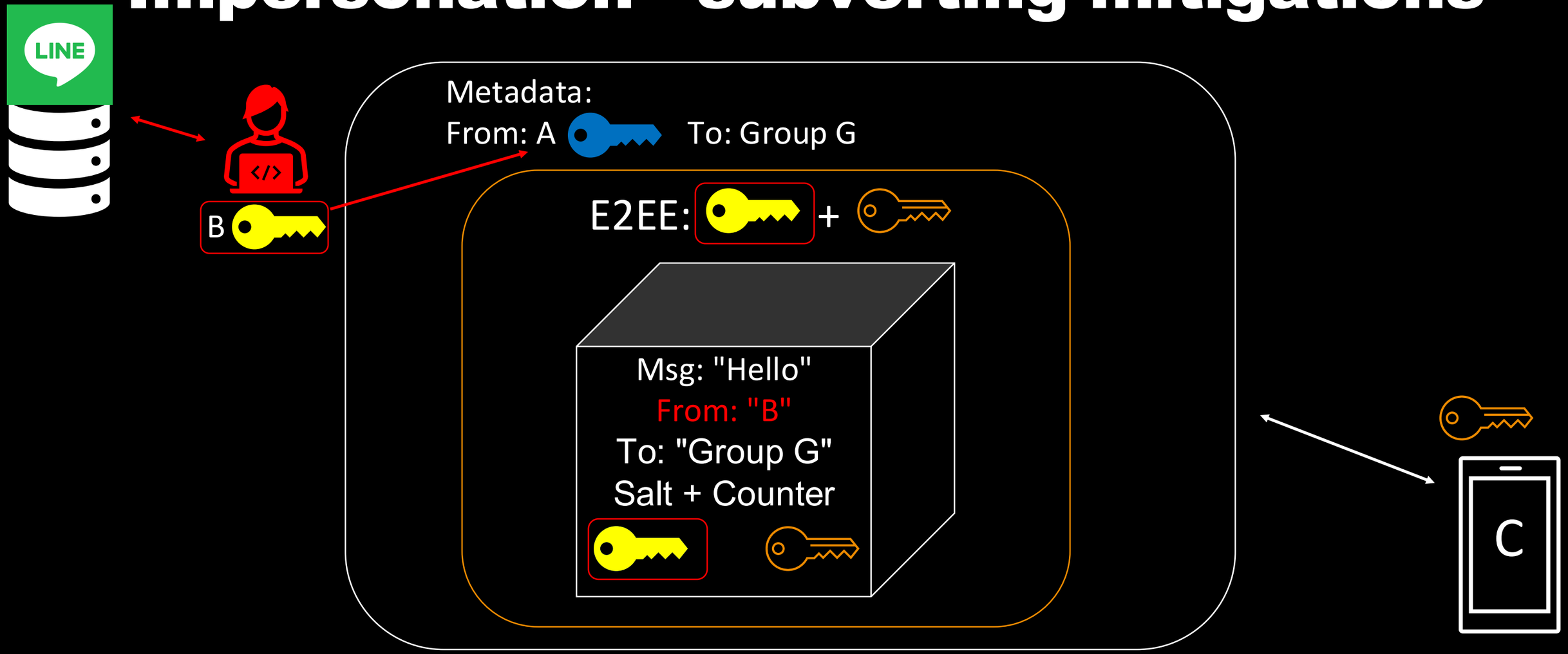
Impersonation – E2E



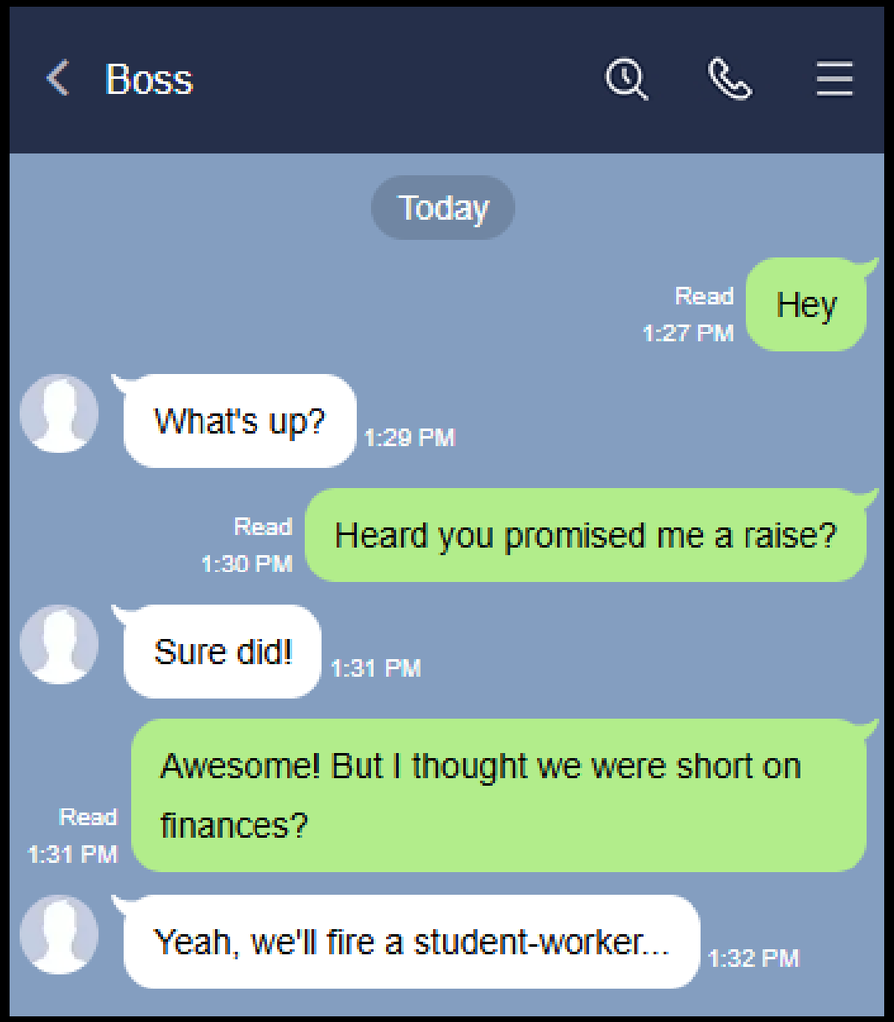
Impersonation - subverting mitigations



Impersonation - subverting mitigations



Impersonation result



Plaintext leakage

Sticker leakage

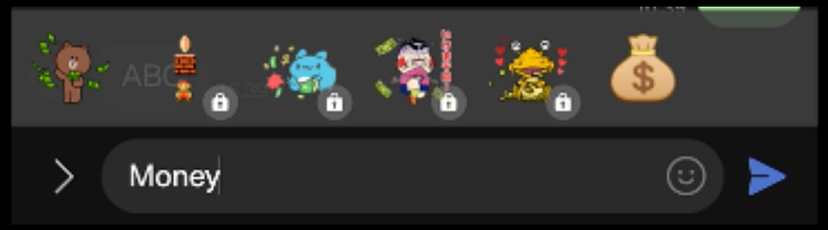
- *Stickers aren't sensitive, right?*

- Guess the word matching the stickers:



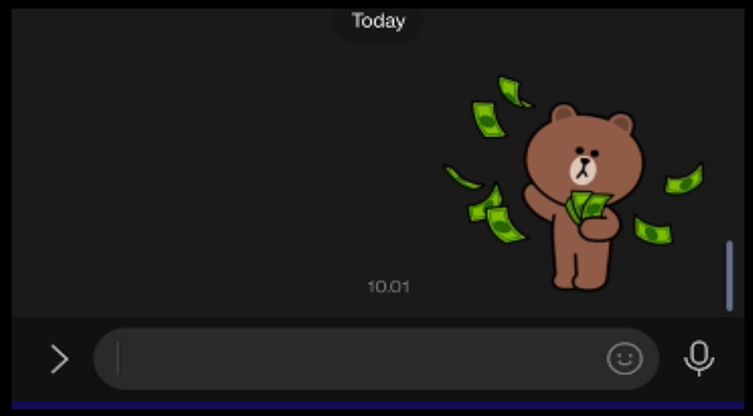

Sticker leakage

Typing:

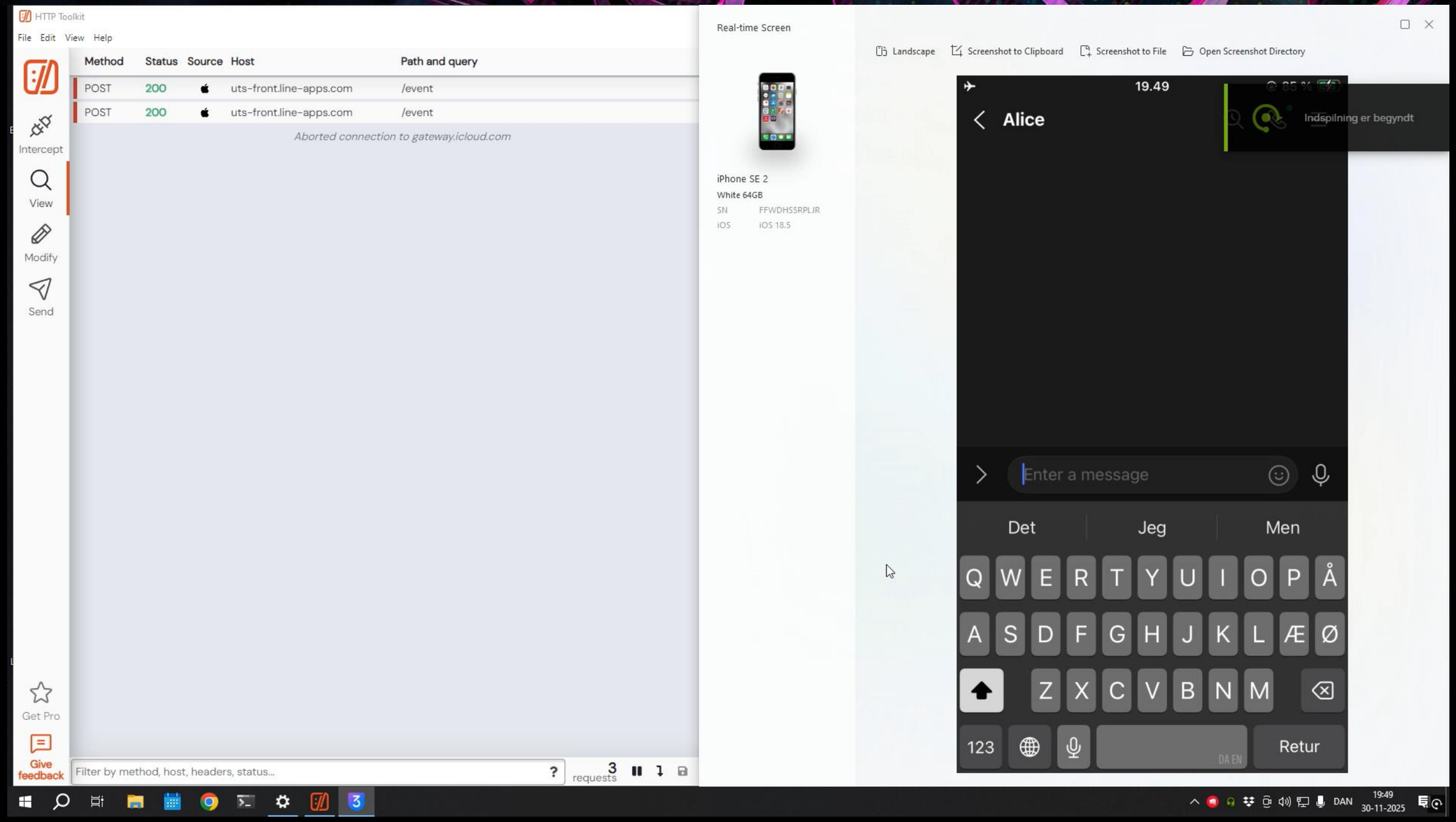


Previewing:

Sending:

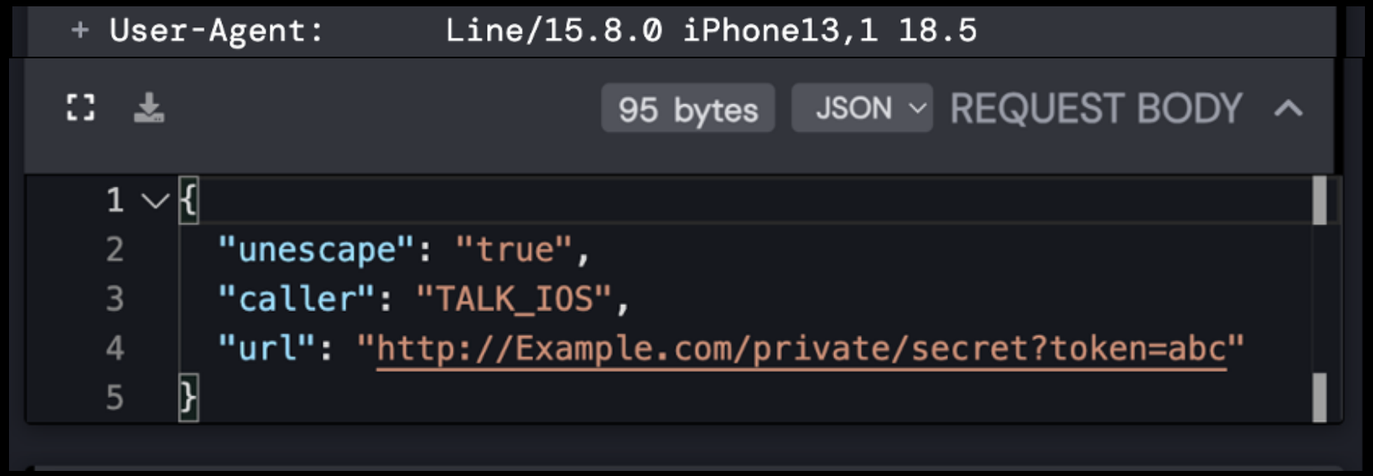


- *"Anonymous lookup"* - leaks User-Agent
- *Malicious Stickers => Spyware?*


HTTP Toolkit
File Edit View Help
Method Status Source Host Path and query
POST 200 uts-front.line-apps.com /event
POST 200 uts-front.line-apps.com /event
Aborted connection to gateway.icloud.com
Intercept
View
Modify
Send
Get Pro
Give feedback
Filter by method, host, headers, status...
3 requests
Real-time Screen
Landscape
Screenshot to Clipboard
Screenshot to File
Open Screenshot Directory
iPhone SE 2
White 64GB
SN FFWDHS5RPLJR
iOS iOS 18.5
19.49
Alice
Indspilning er begyndt
Enter a message
Det Jeg Men
Q W E R T Y U I O P Å
A S D F G H J K L Æ Ø
Z X C V B N M
123
Retur
DA EN
DAN
19:49
30-11-2025

URL leakage

Note: In order to generate URL previews, links shared in chats are also sent to LINE's servers.



- *URL censorship?*

Violated security properties

Attack	Violated Security Property
Replay, reorder, blocking	Integrity
Impersonation	Integrity, Authentication
Plaintext leakage (stickers + URLs)	Confidentiality

No continuous key rotation = no Forward Secrecy by design

Real-world impact

Replay, reorder, blocking:

- (Blind) Conversation Manipulation

Impersonation:

- Fully forged conversations
- *Deniability*

Leakage*:*

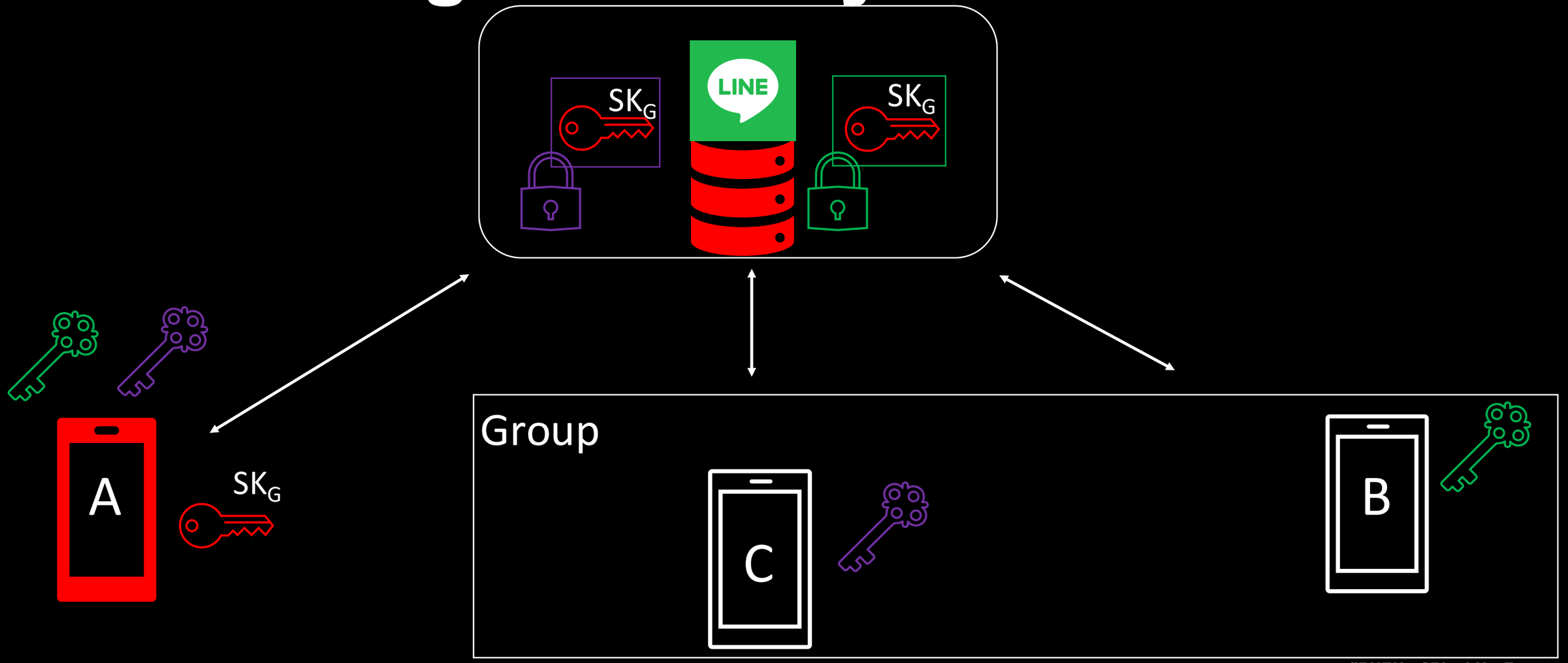
- Spyware
- URL Censorship
- Surveillance

Potentially feasible attacks on groups

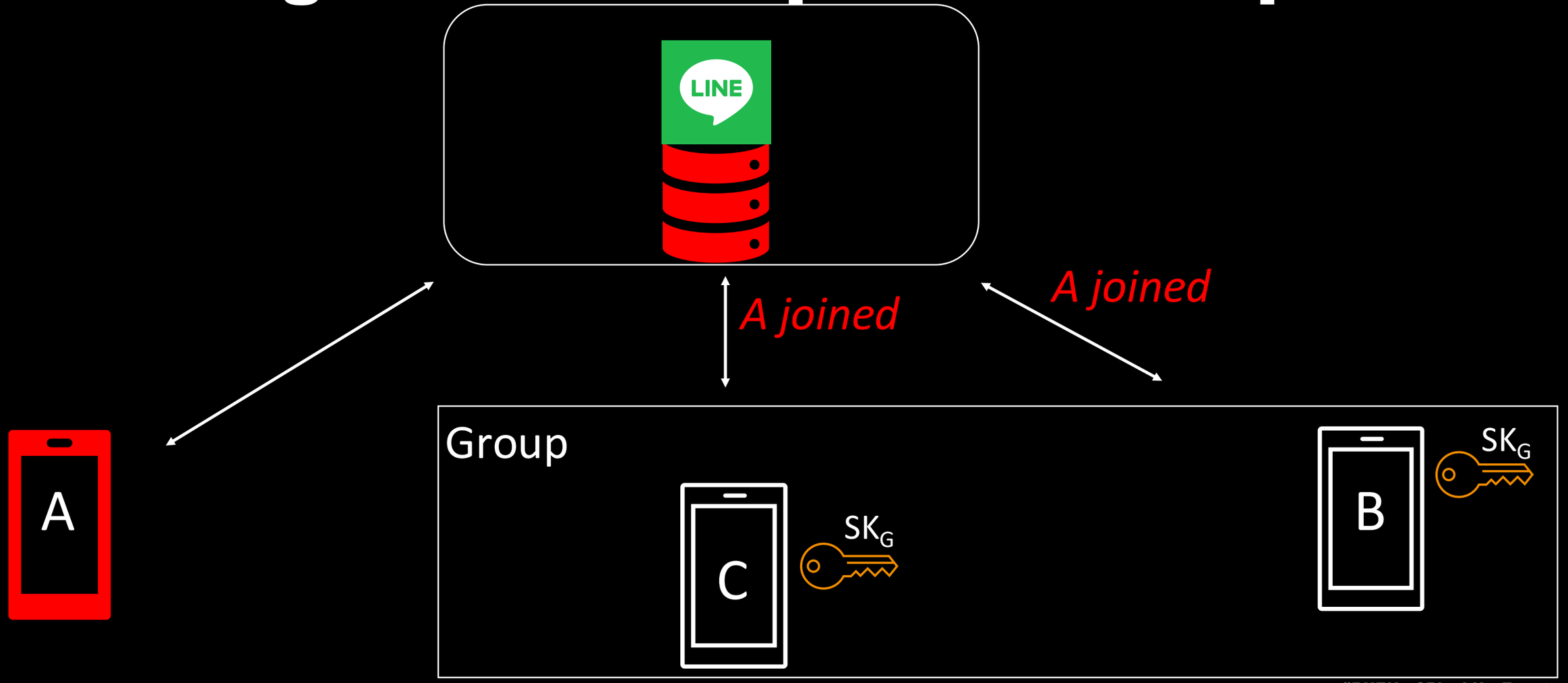
Fully depend on server/app mitigations

1. Illegitimate Key Rotation
2. Illegitimate Group Membership

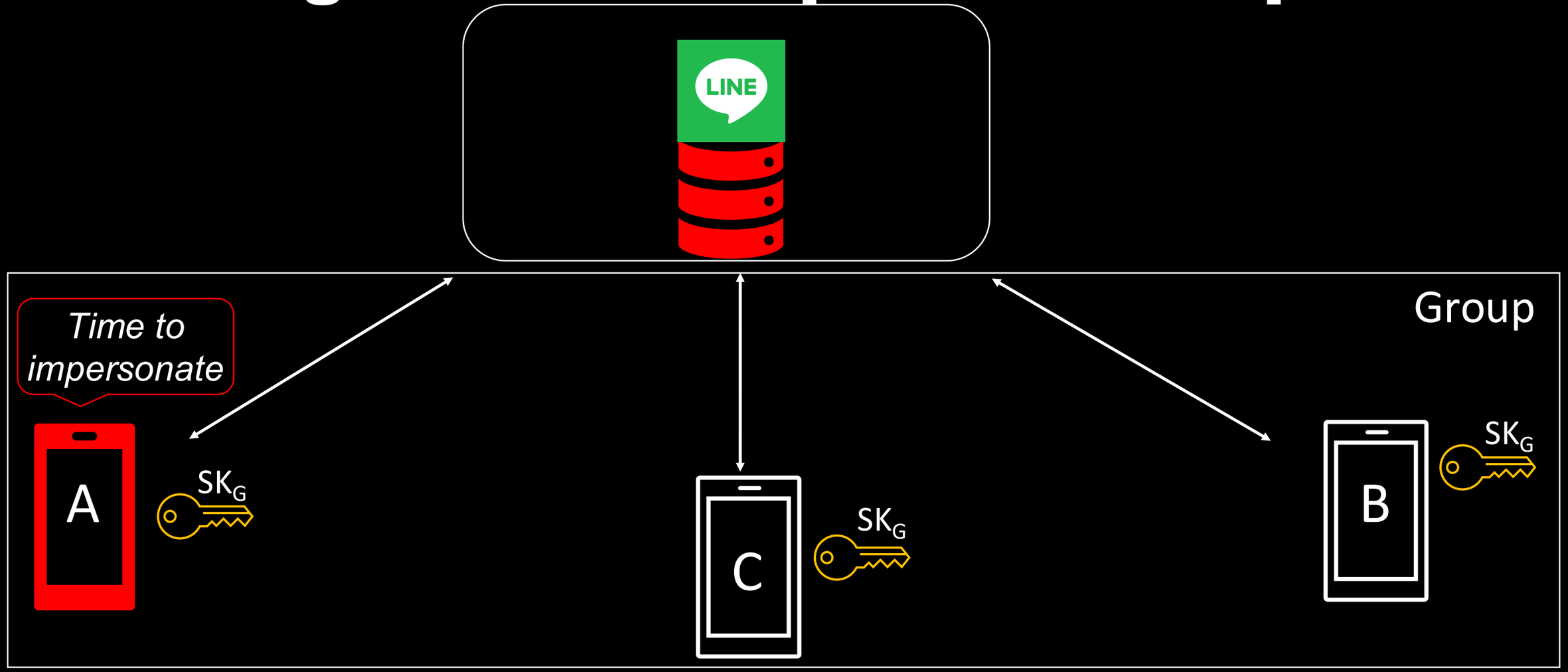
Illegitimate Key Rotation



Illegitimate Group Membership



Illegitimate Group Membership



Coordinated disclosure

LINE recognizes that replay/block/reorder "*are all limitations inherent in our protocol design, of which we were aware.*"

They "*chose to prioritize user experience, which is why we use the server reception timestamp instead, leading to the cryptographic protocol issues*".

Metadata has "*historically been treated as non-sensitive"*, but LINE is "*updating both our threat model and our encryption report accordingly."*

Coordinated disclosure

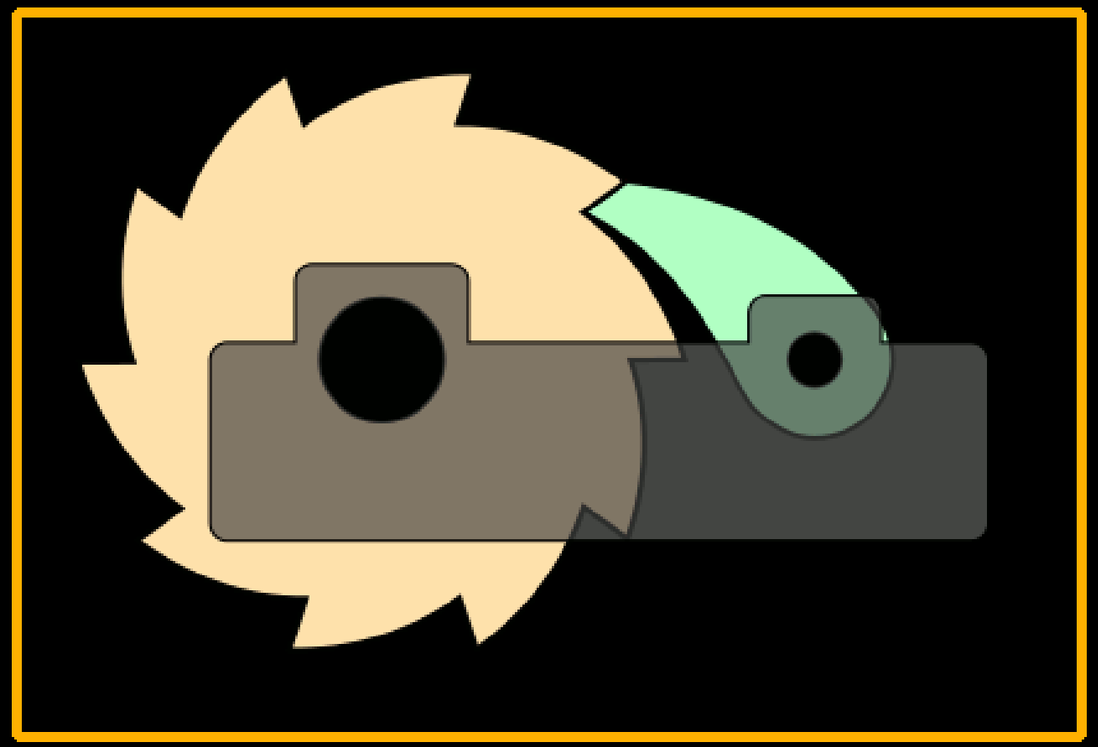
With regards to impersonation, "*there are checks at the server level preventing impersonated messages from being sent through our infrastructure, but we understand this is not a sufficient countermeasure in the E2EE threat model.*"

Plaintext leakage *"is a result of how we chose to balance privacy with usability.*":

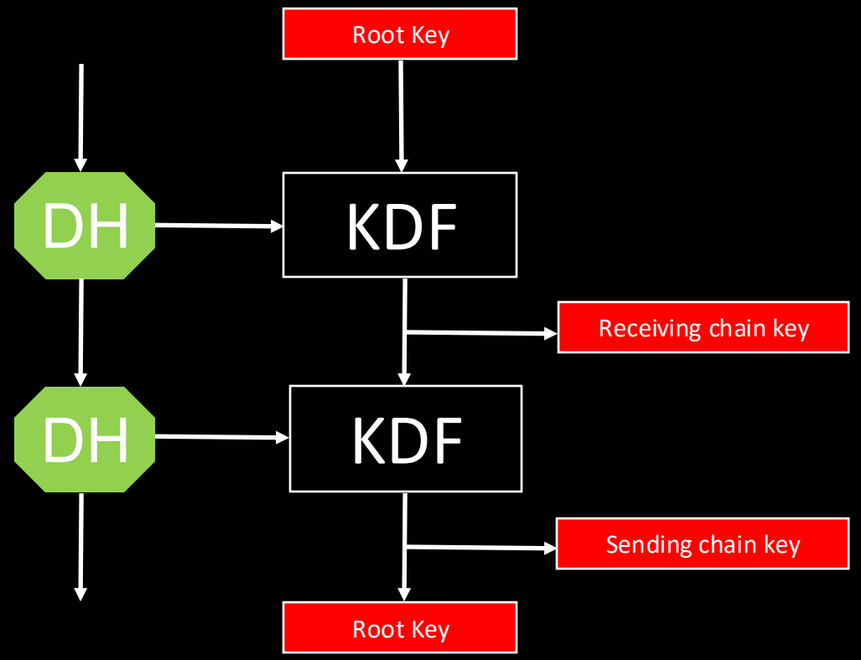
- Private lookups for stickers has been considered, but inefficient at scale
- URL and sticker previews are opt-out features

LINE is committed to *"improving our security and are looking into updating our cryptographic protocol. We will consider how to better address some of the issues you reported, as well as other limitations mentioned in our public encryption report."*

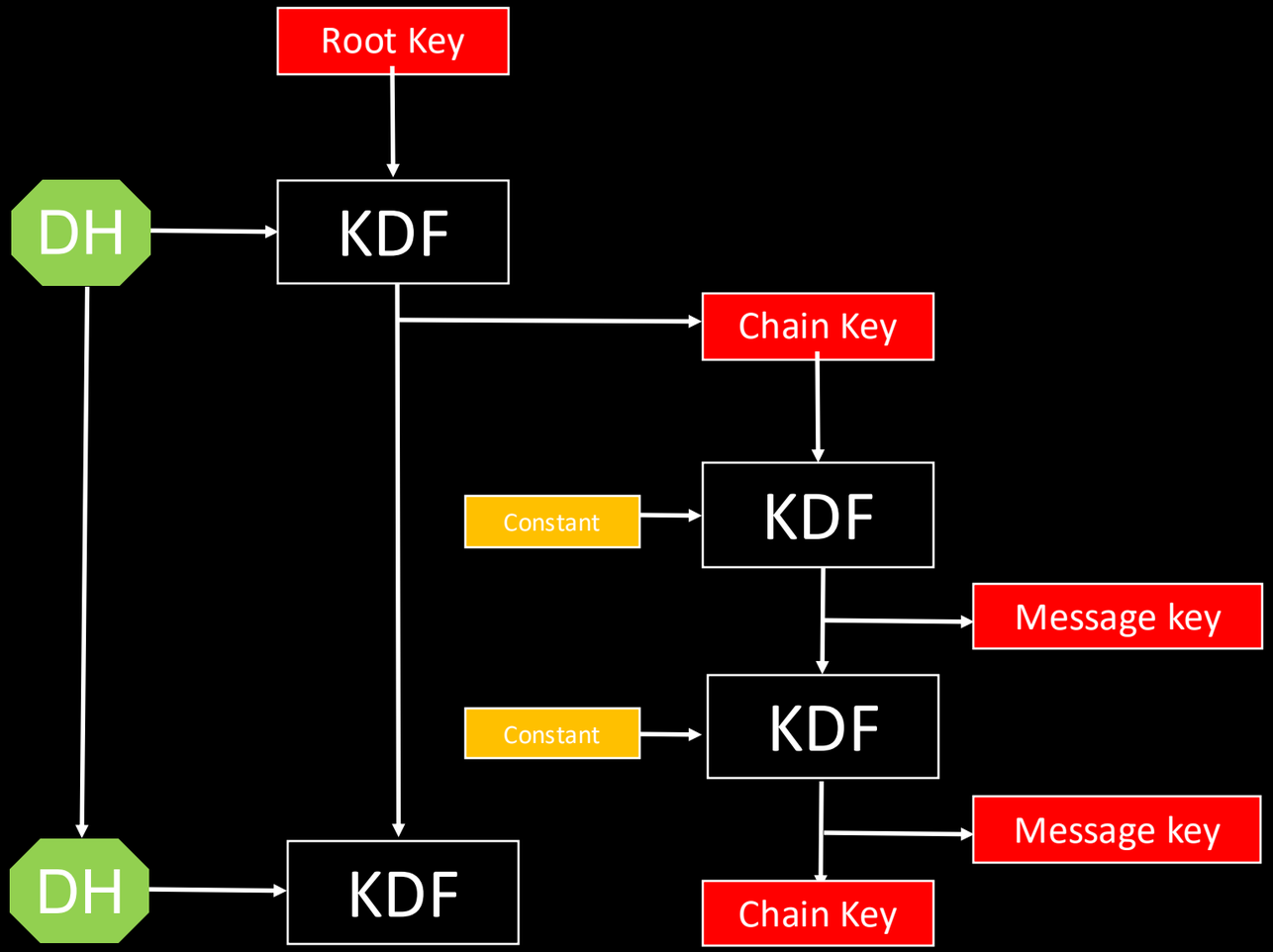
Signal: the Double Ratchet



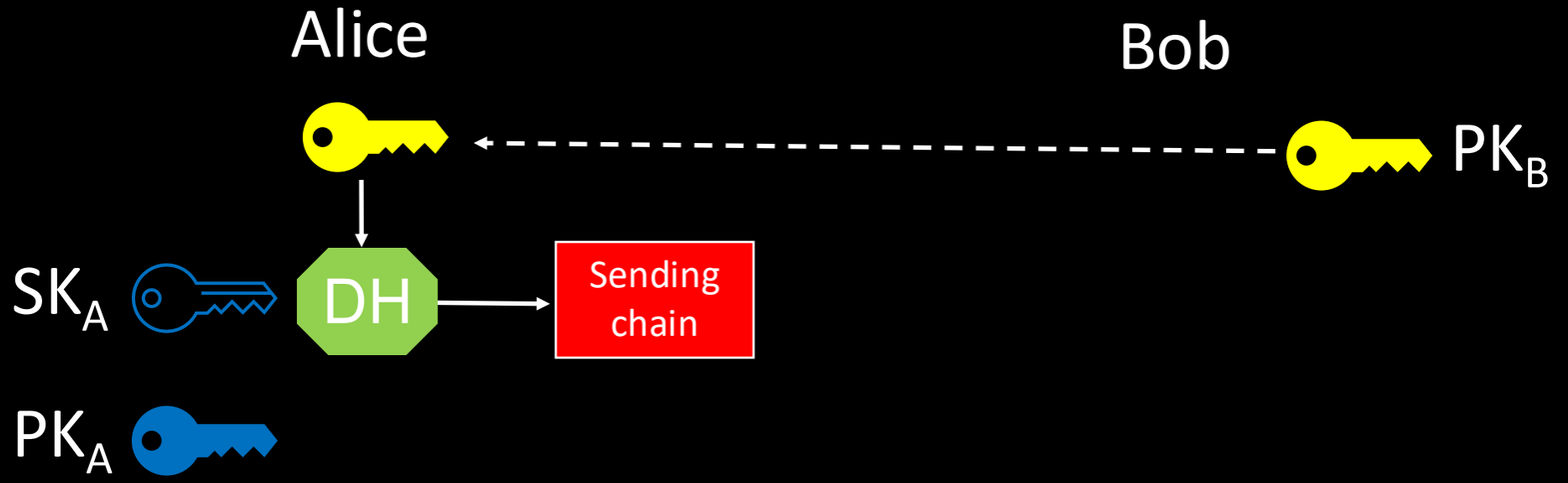
Source: https://commons.wikimedia.org/wiki/File:Ratchet_Drawing.svg
License: CC BY-SA 3.0



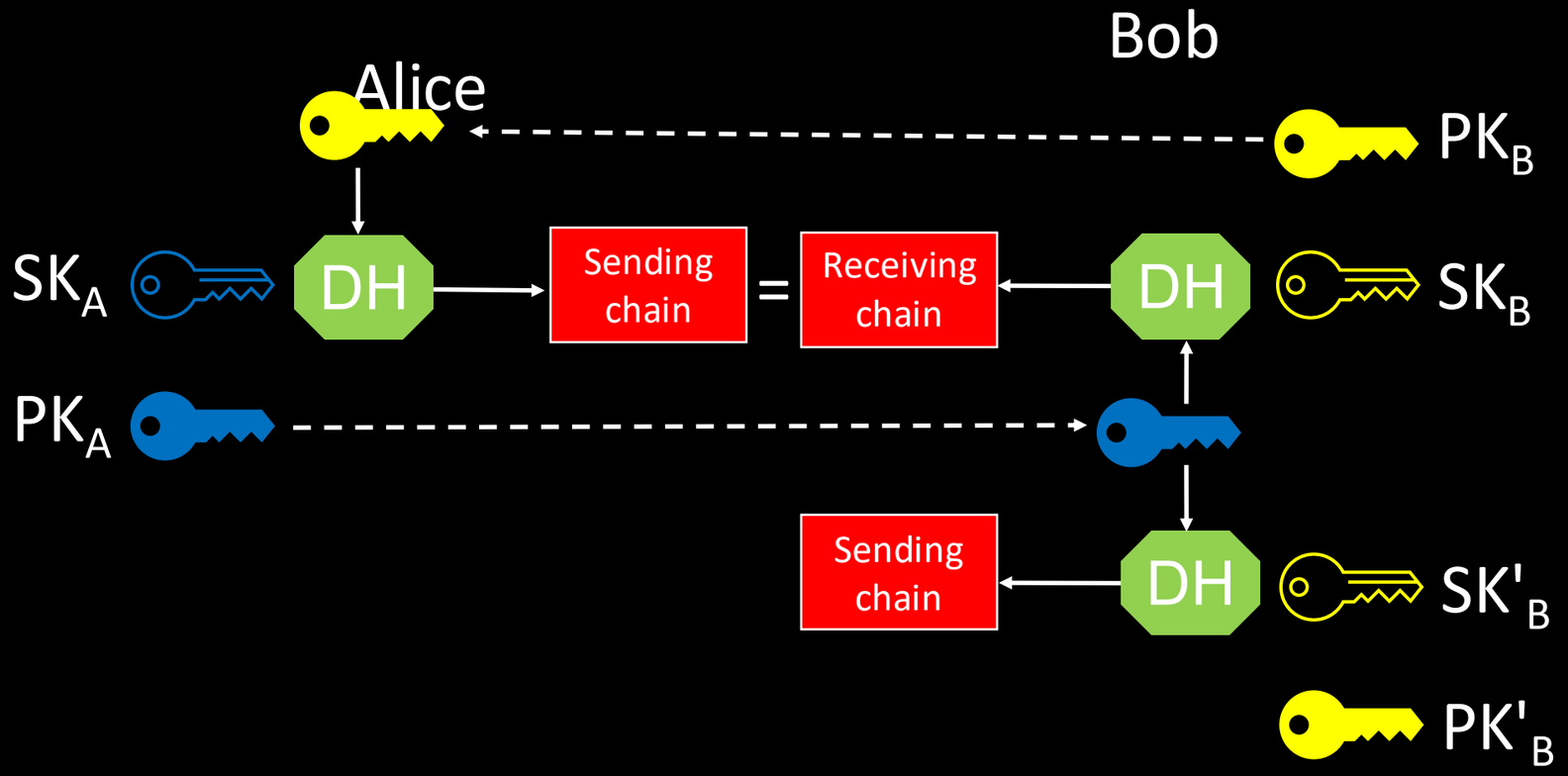
The Symmetric Ratchet



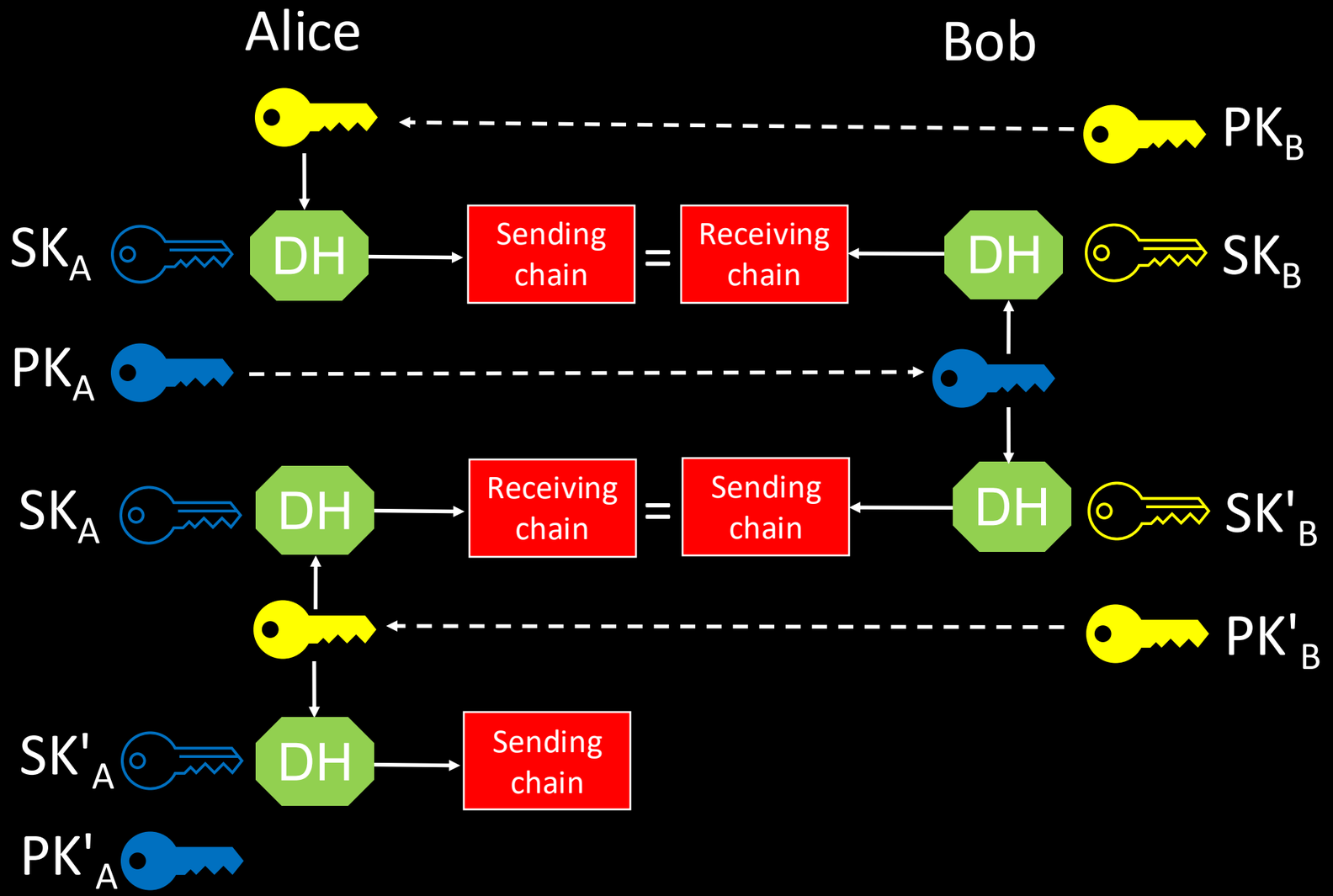
The DH Ratchet



The DH Ratchet



The DH Ratchet



Related Resources

These E2EE messengers all use the Signal protocol: 

You can read more at:

- *More is Less: On the End-to-End Security of Group Chats in Signal, WhatsApp, and Threema*
  `https://eprint.iacr.org/2017/713`
- *The Double Ratchet Algorithm (Signal):*
  `https://signal.org/docs/specifications/doubleratchet/`
- *Three Lessons From Threema: Analysis of a Secure Messenger*
  `https://breakingthe3ma.app/`
- *Breaking Message Integrity of an End-to-End Encryption Scheme of LINE*
  `https://eprint.iacr.org/2018/668`

Takeaways

As a LINE user, you implicitly **trust** the server to:

- Deliver all messages in-order
- Not snoop into what you are typing (stickers + URL)
- Not collude in impersonation attacks

*This is **not** how E2EE is supposed to work.*

In general, do not deploy your own cryptography!

- Read the standards/literature/specifications/best practices
- Get independent review before you deploy

Questions?

linebreak.info