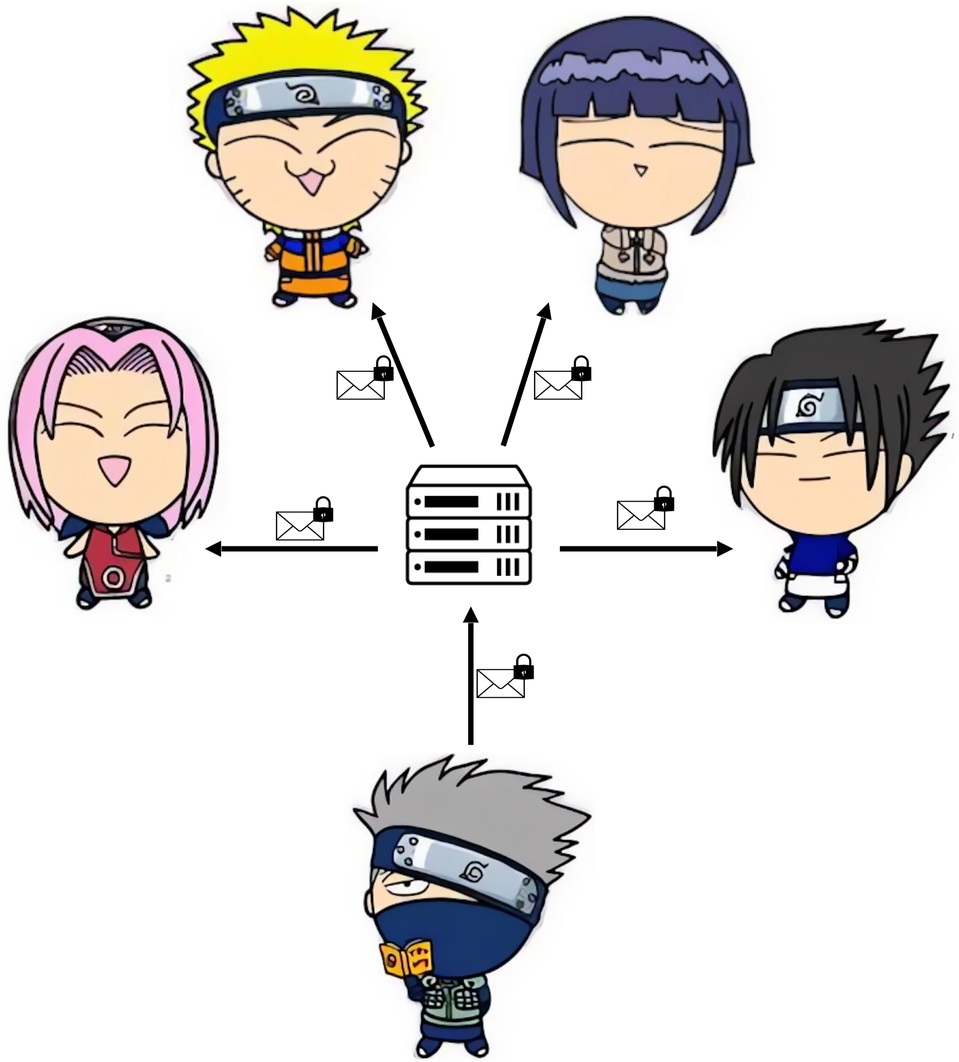
Insider Attacks in MLS

Joseph Jaeger and Akshaya Kumar

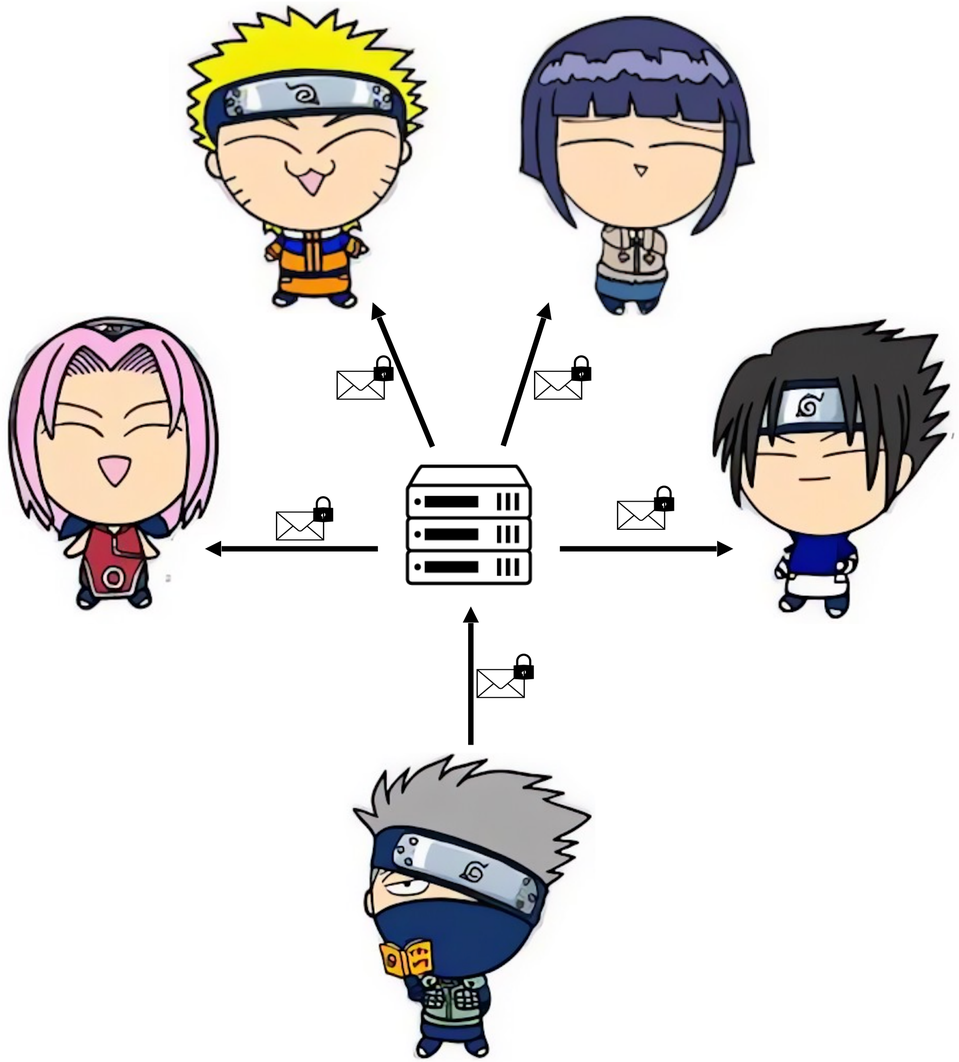
IETF122, March 2025

Georgia Tech

Group Messaging

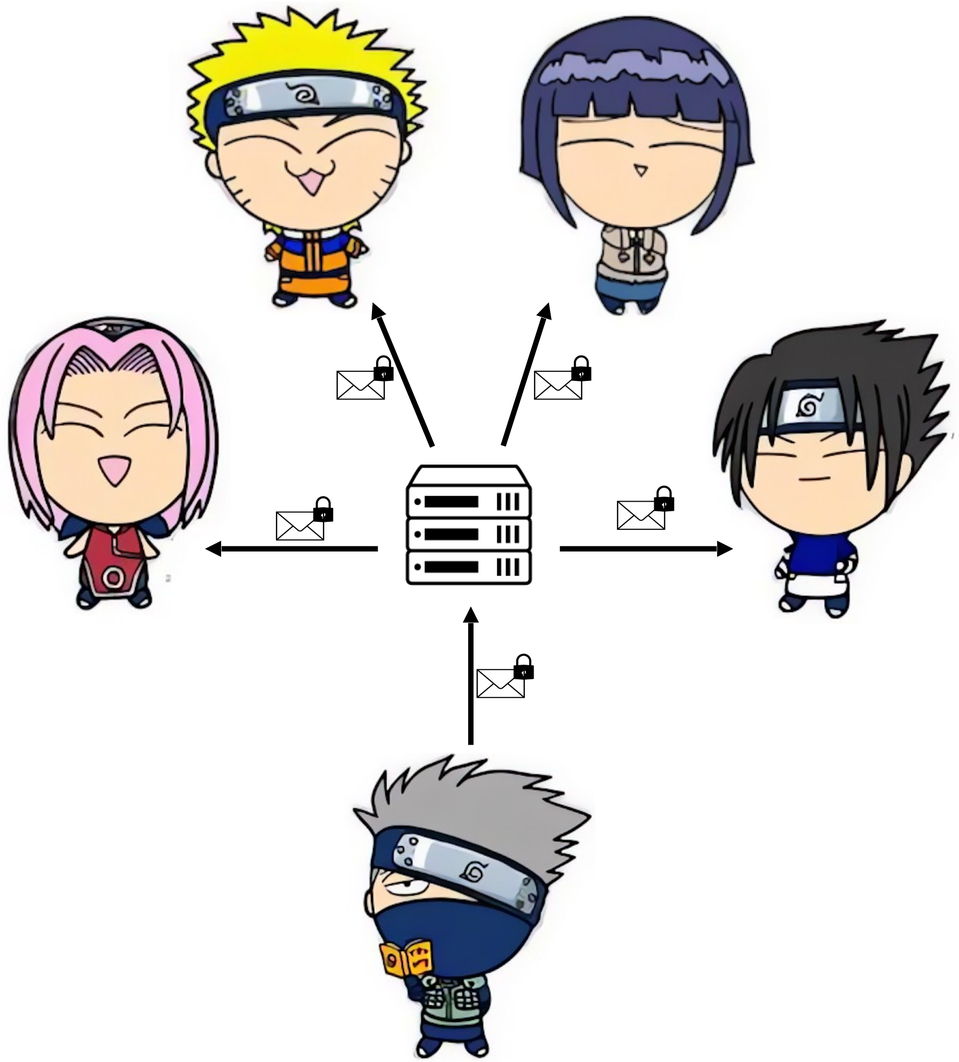


Group Messaging



Secure Group Messaging

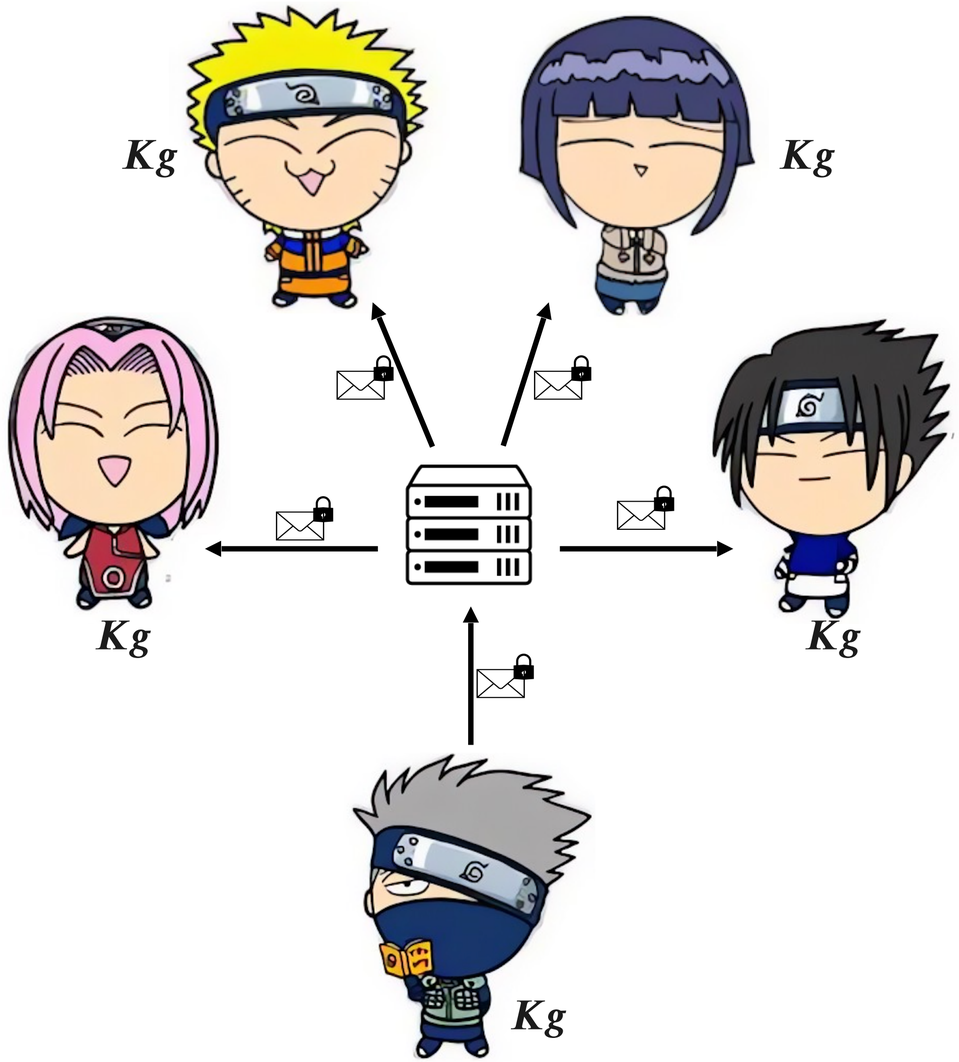
Group Messaging



Secure Group Messaging

Key Agreement

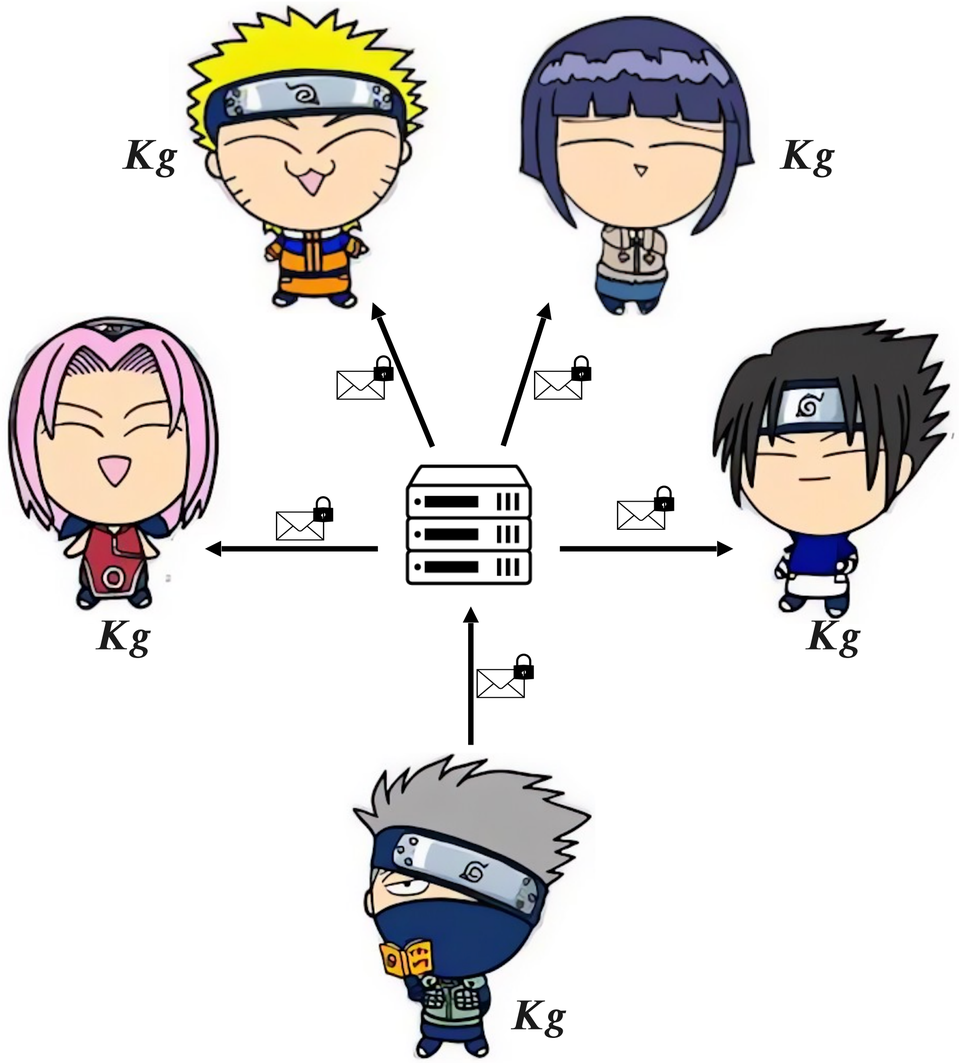
Group Messaging

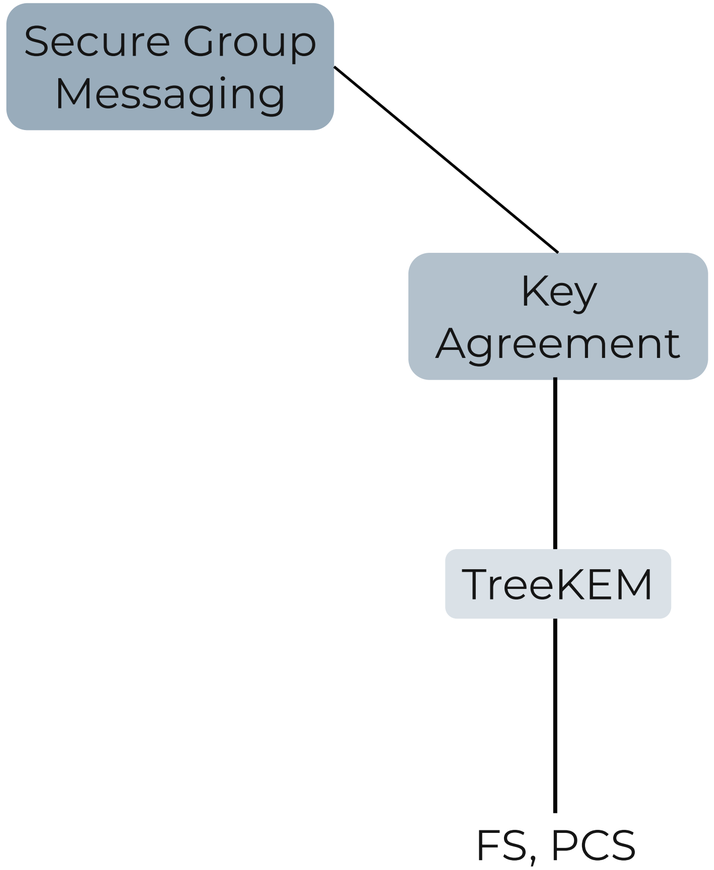


Secure Group Messaging

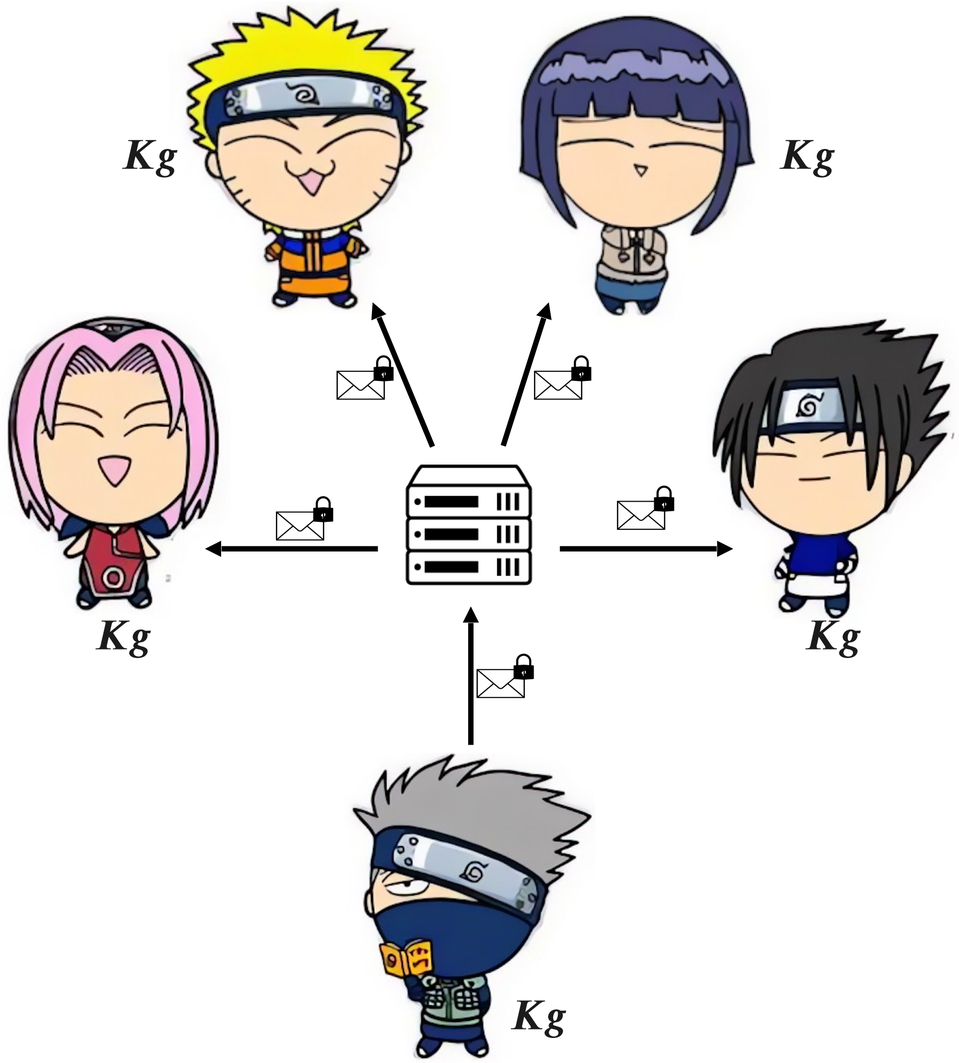
Key Agreement

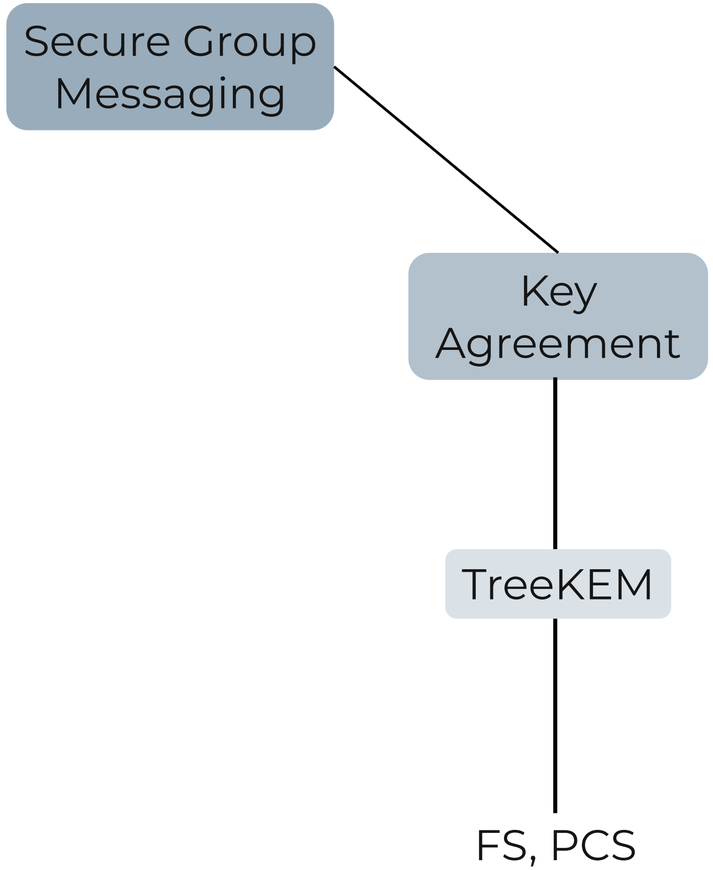
Group Messaging





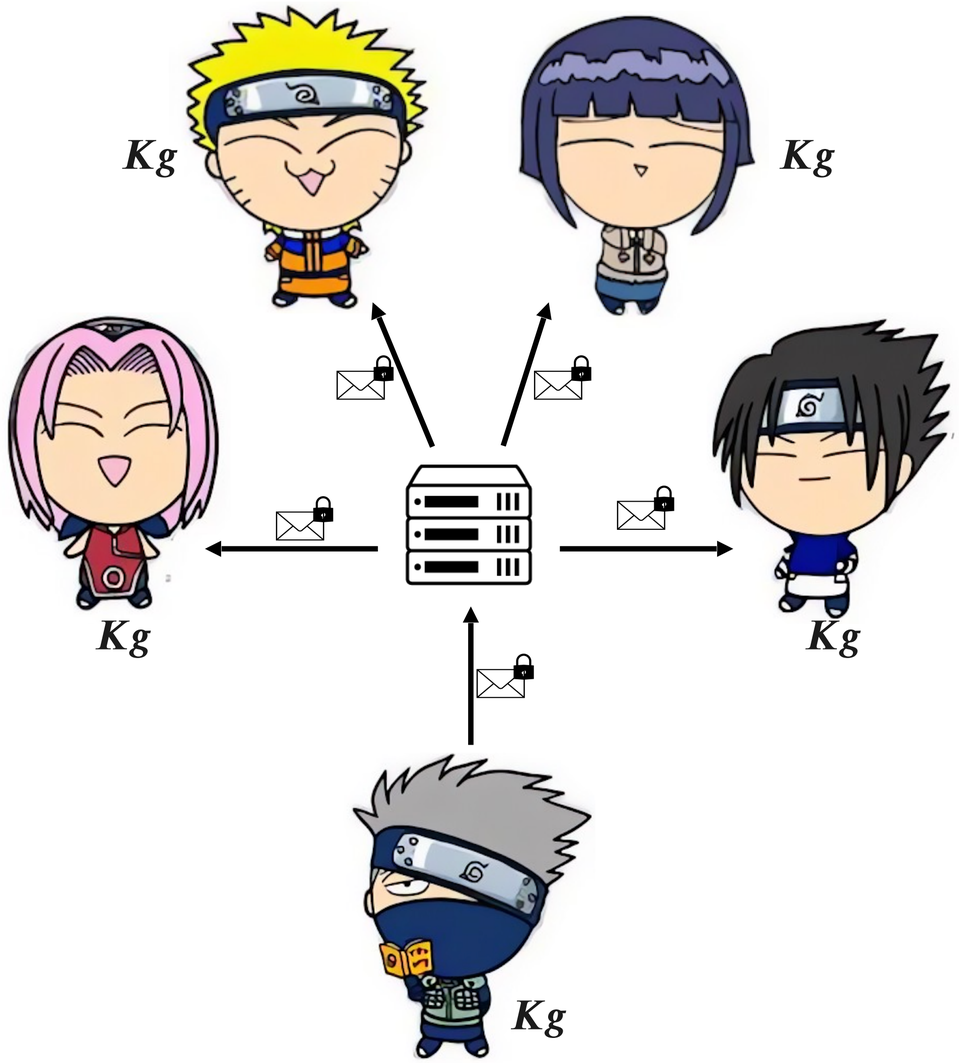
Group Messaging





[BBR18, ACDT20, ACJM20, KPPW+21, ACDT21, HKPPW21, AHKM22, ACPP23, AWT23, BCG23, CLM23, ...]

Group Messaging



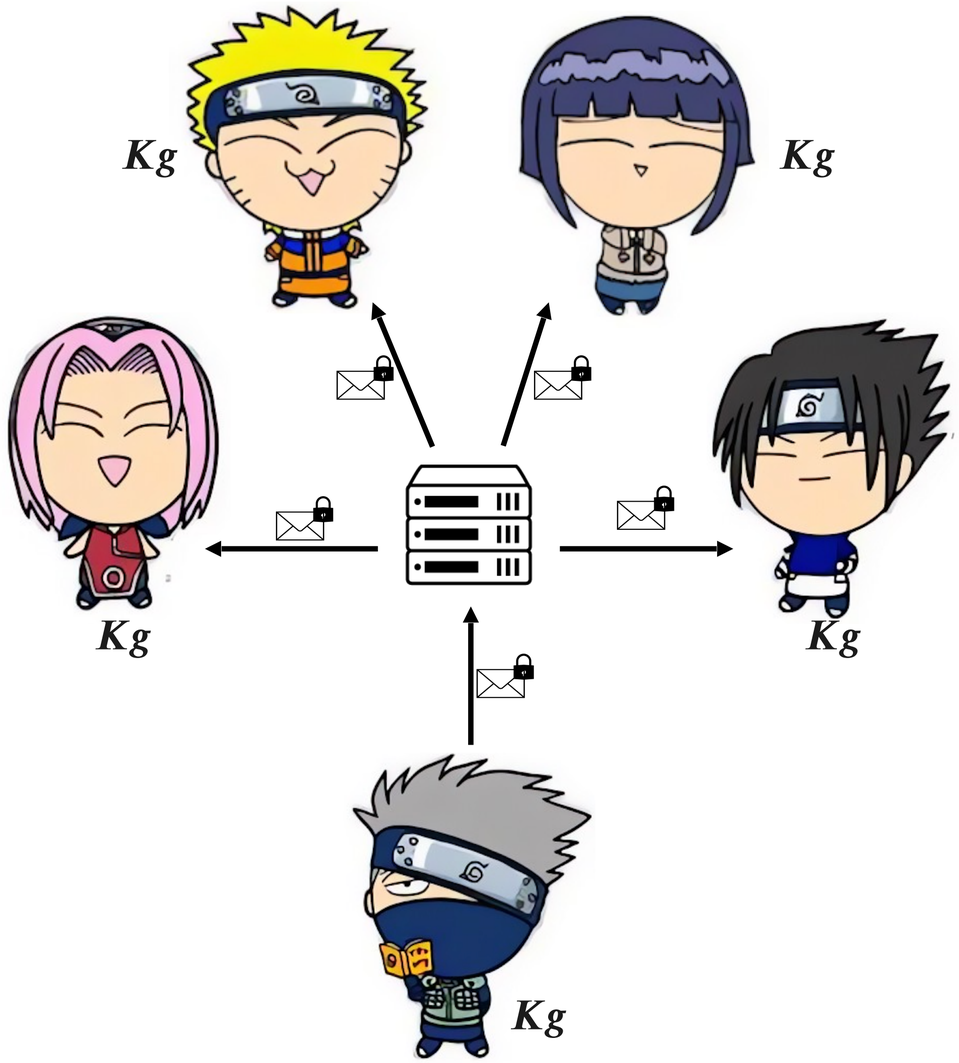
Secure Group Messaging

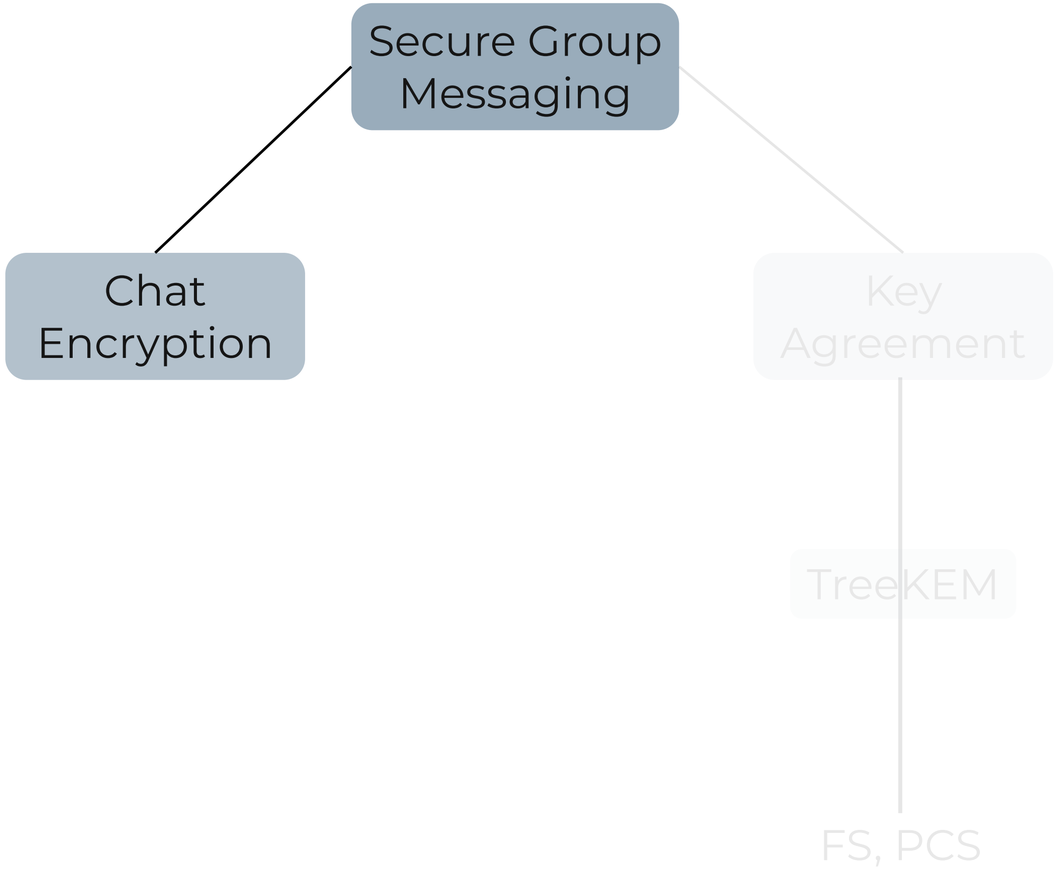
Key Agreement

TreeKEM

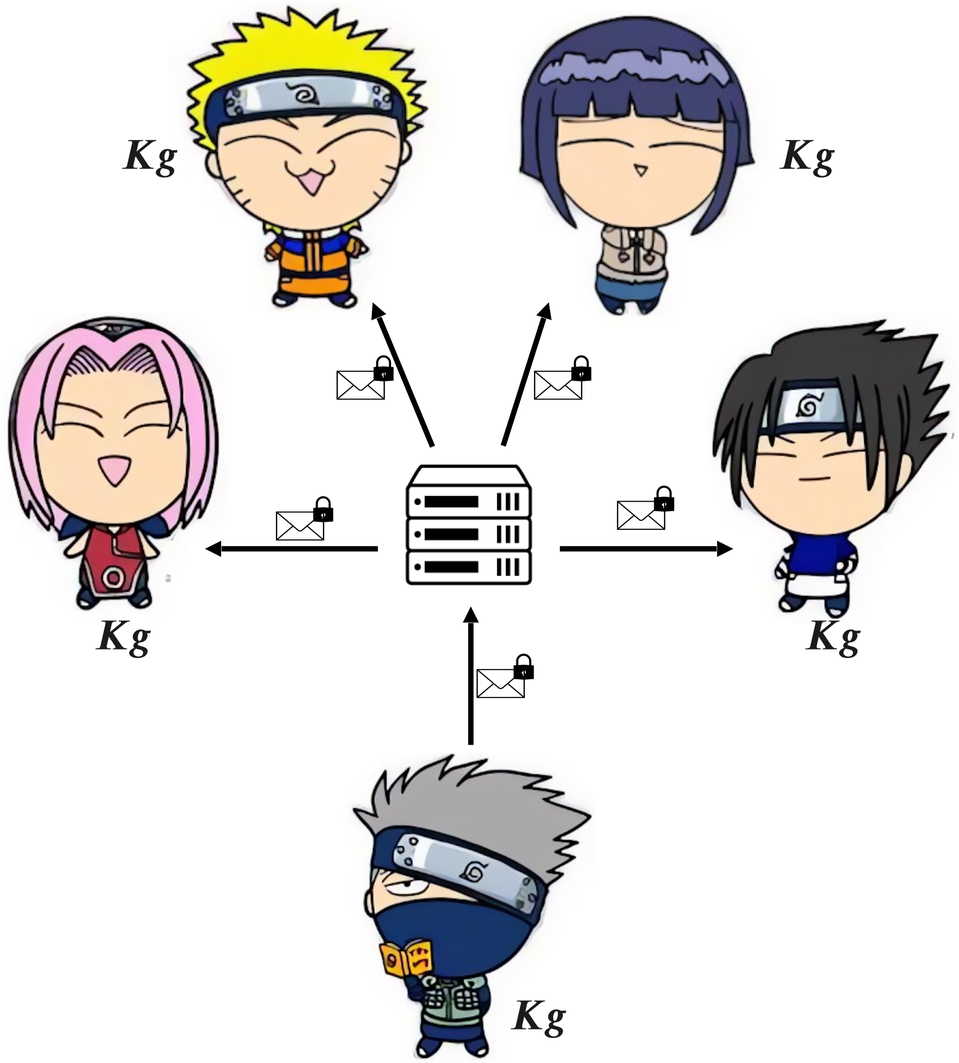
FS, PCS

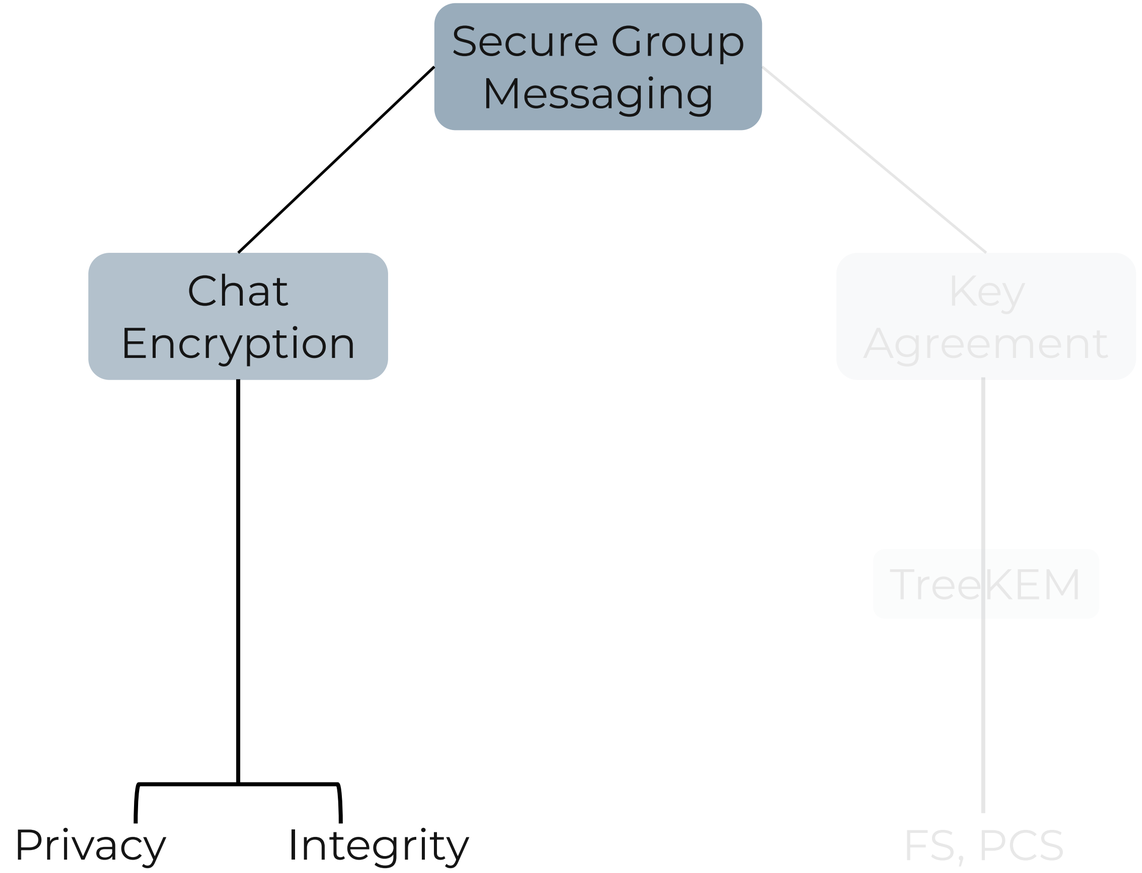
Group Messaging



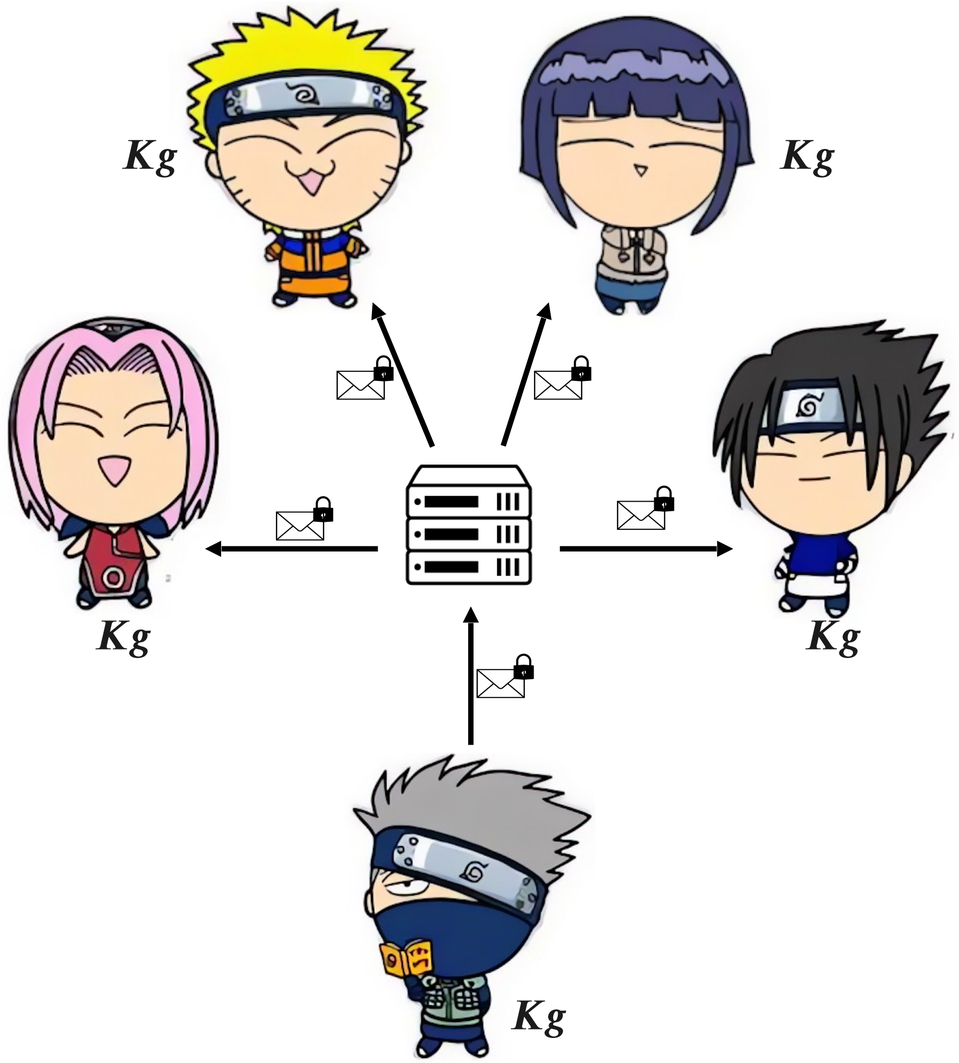


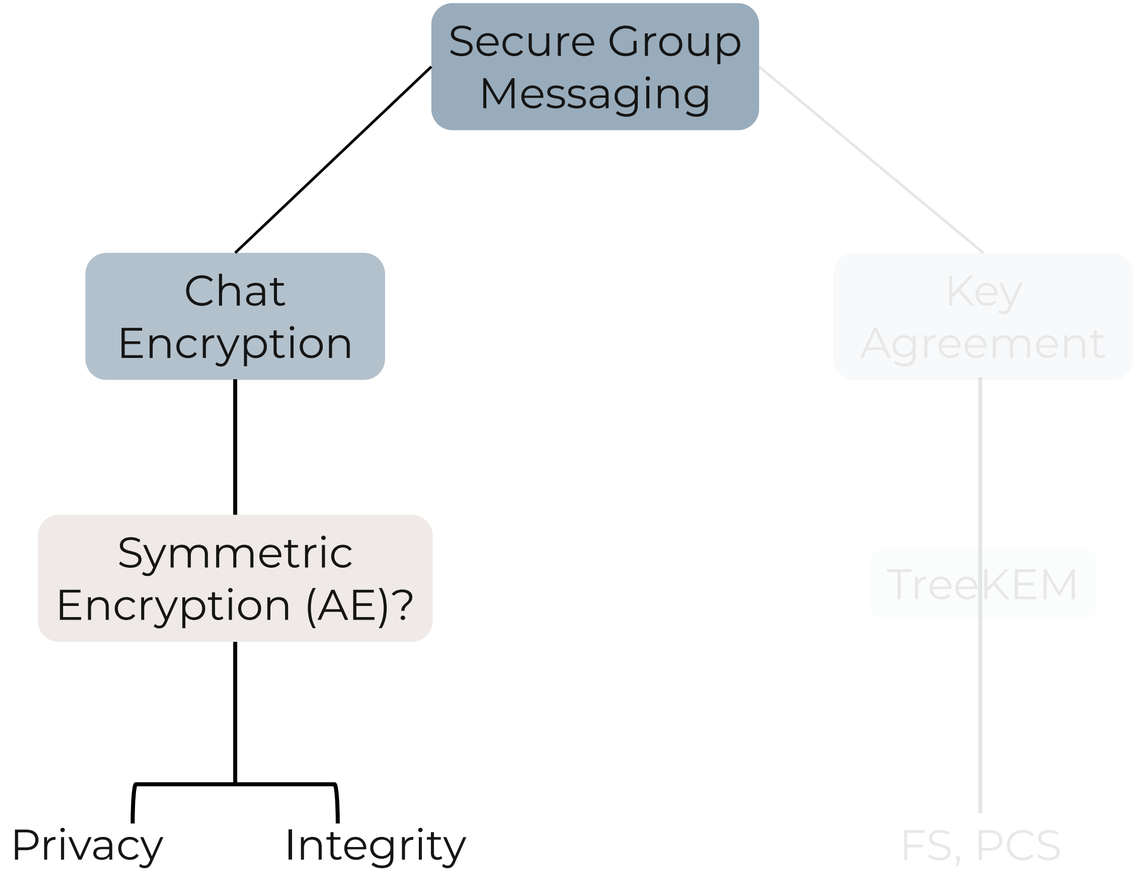
Group Messaging



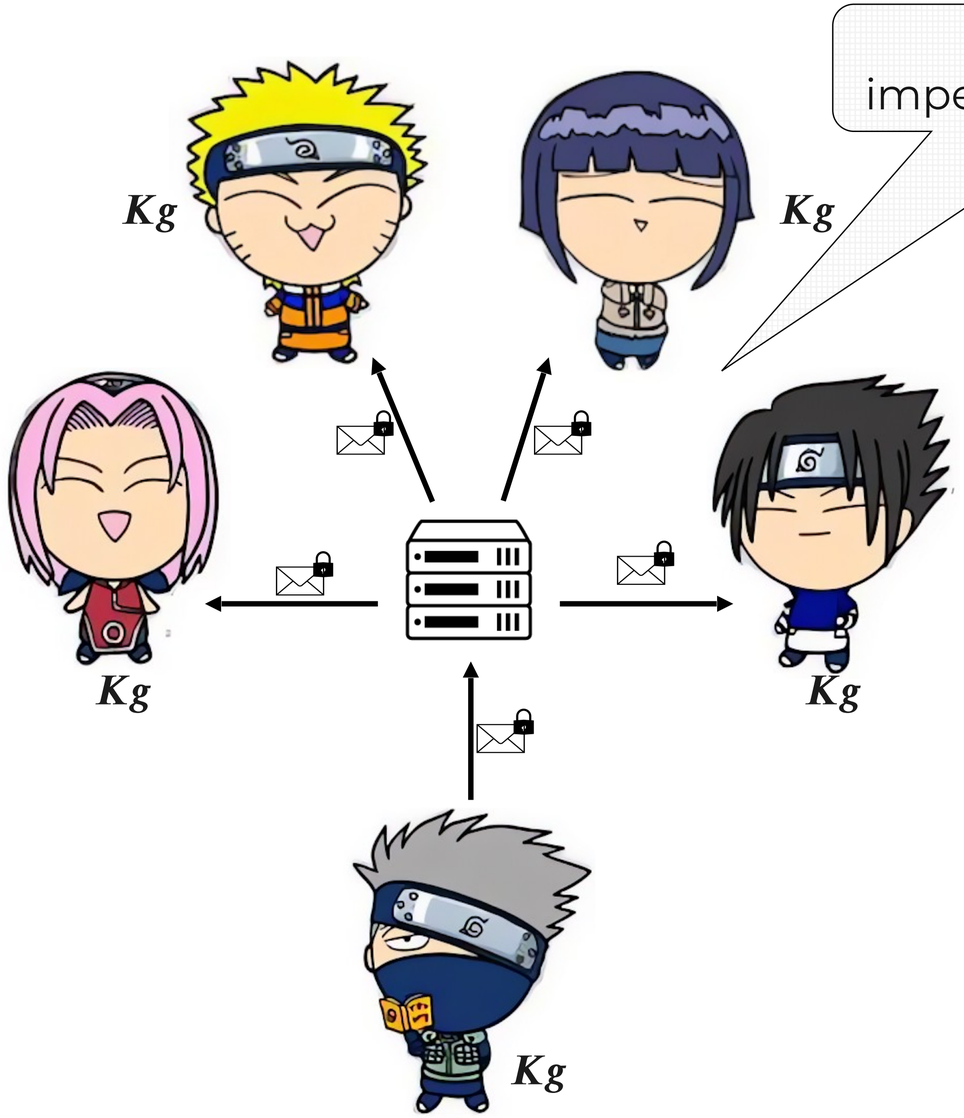


Group Messaging





Group Messaging



Members can impersonate each other!

Secure Group Messaging

Chat Encryption

Symmetric Encryption (AE)?

Privacy

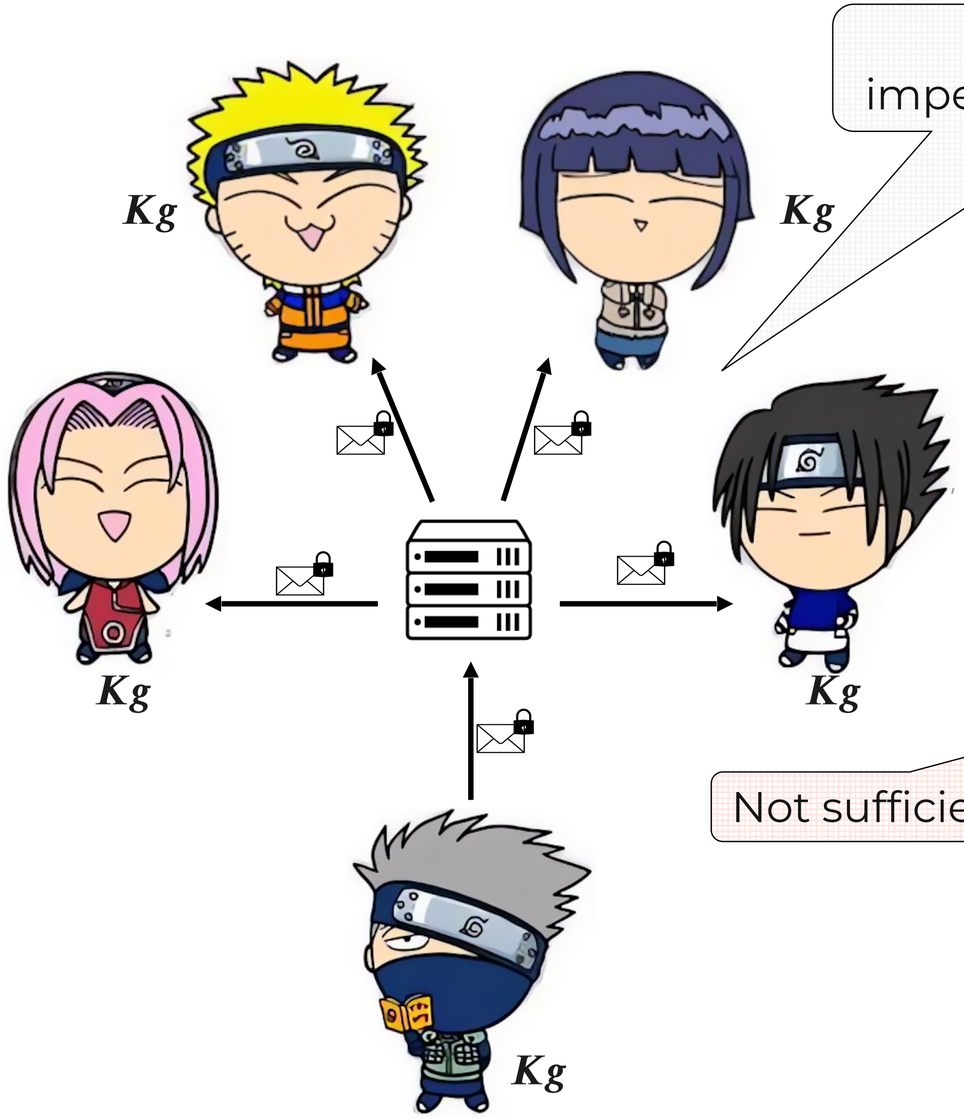
Integrity

Key Agreement

TreeKEM

FS, PCS

Group Messaging



Members can impersonate each other!

Secure Group Messaging

Chat Encryption

Key Agreement

Symmetric Encryption (AE)?

TreeKEM

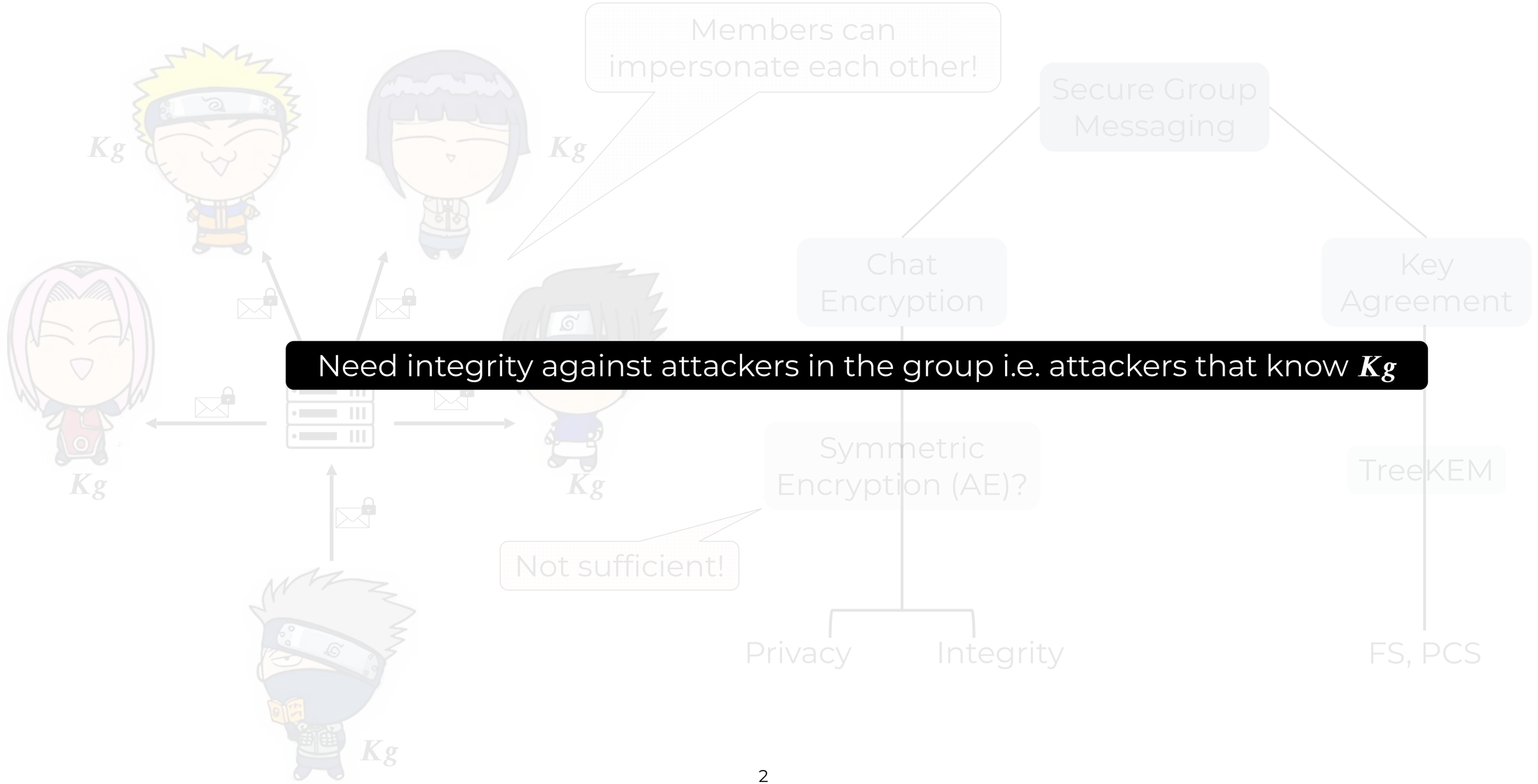
Not sufficient!

Privacy

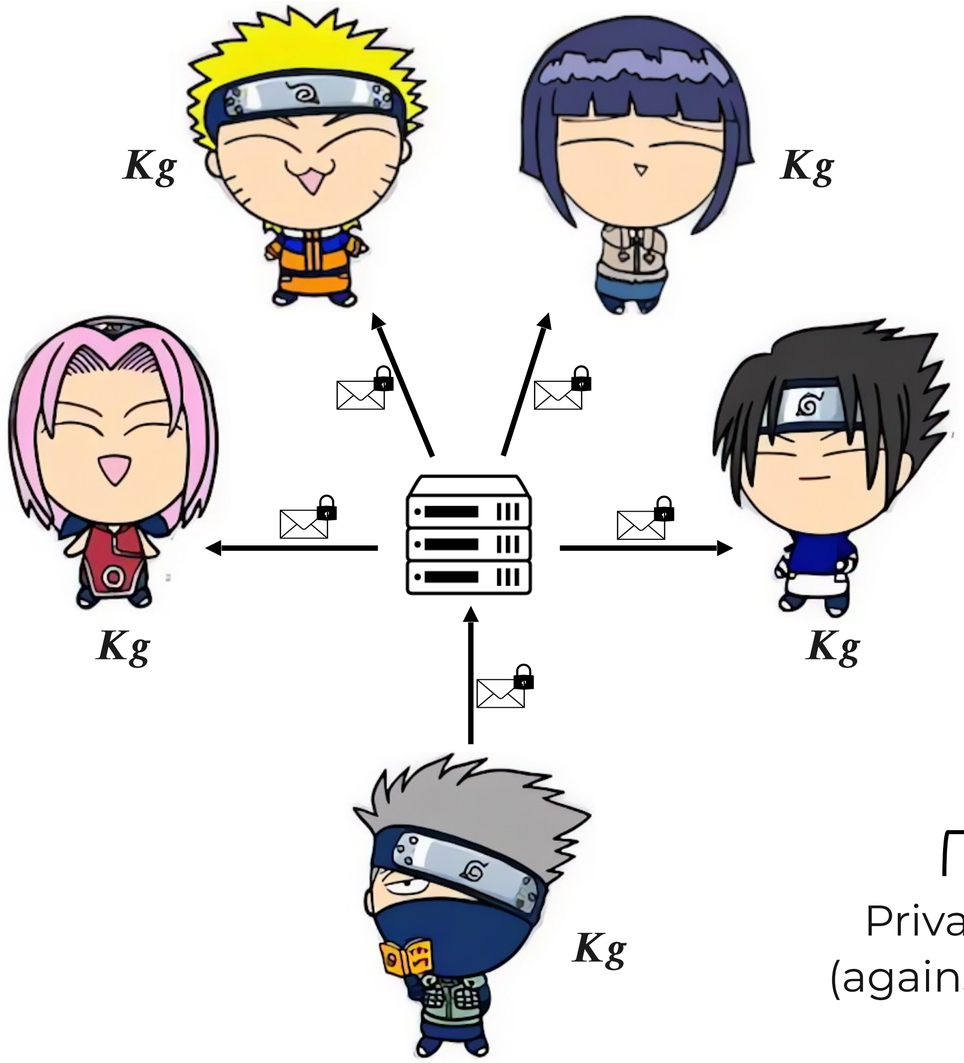
Integrity

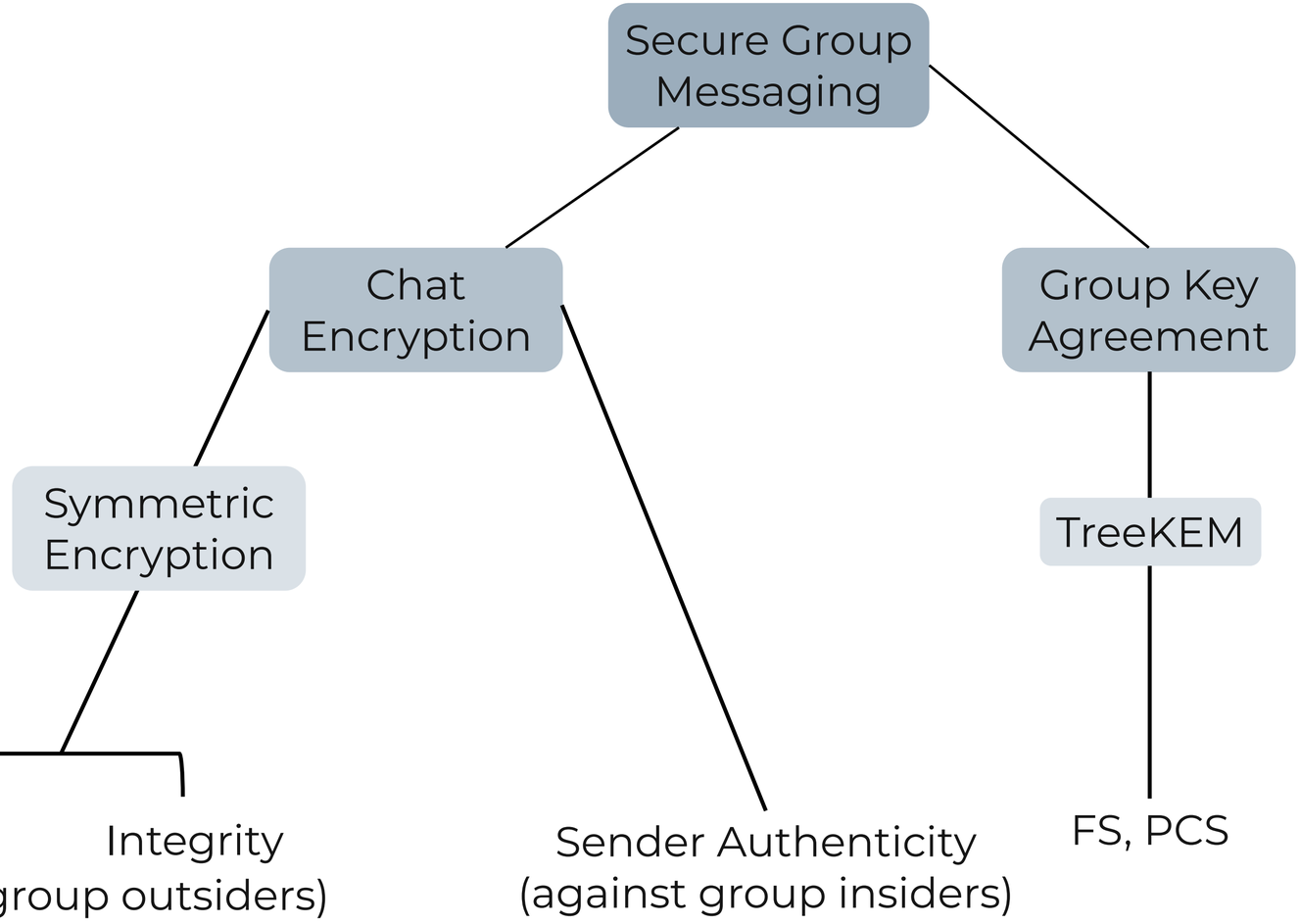
FS, PCS

Group Messaging

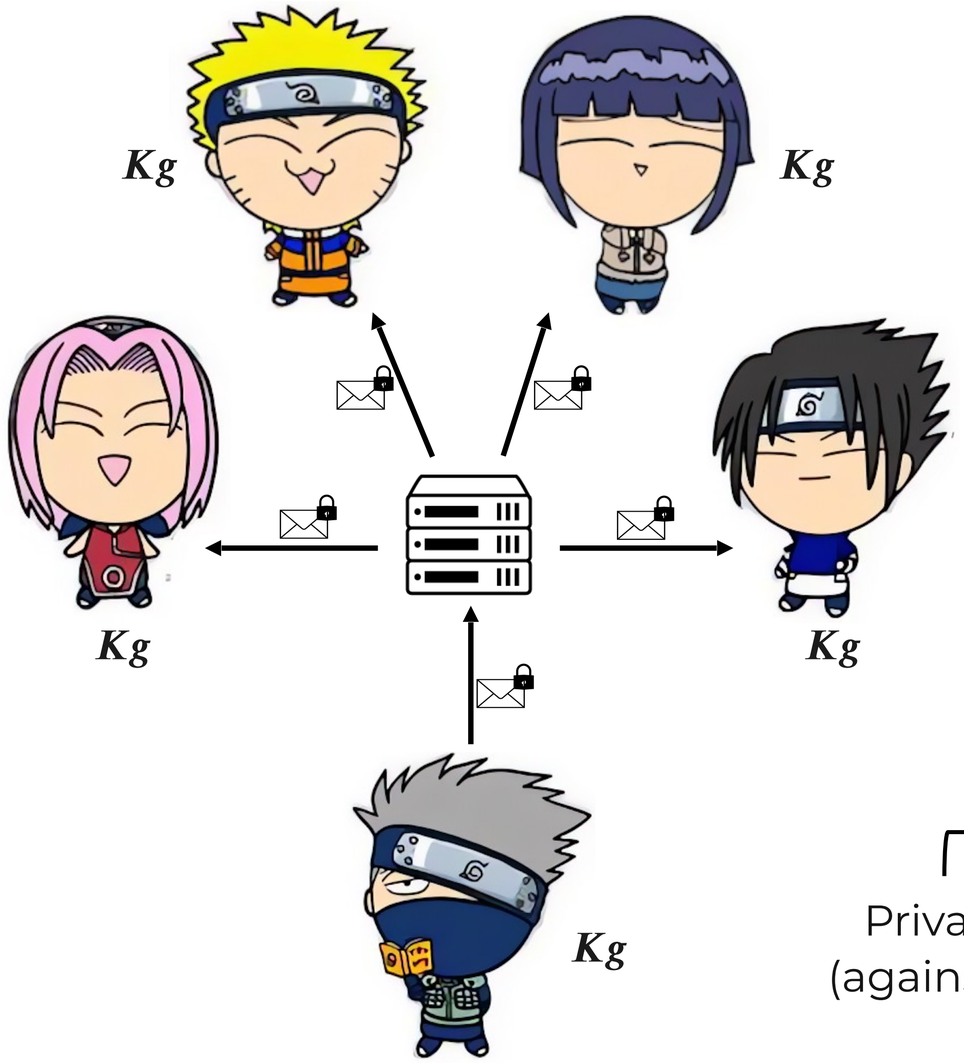


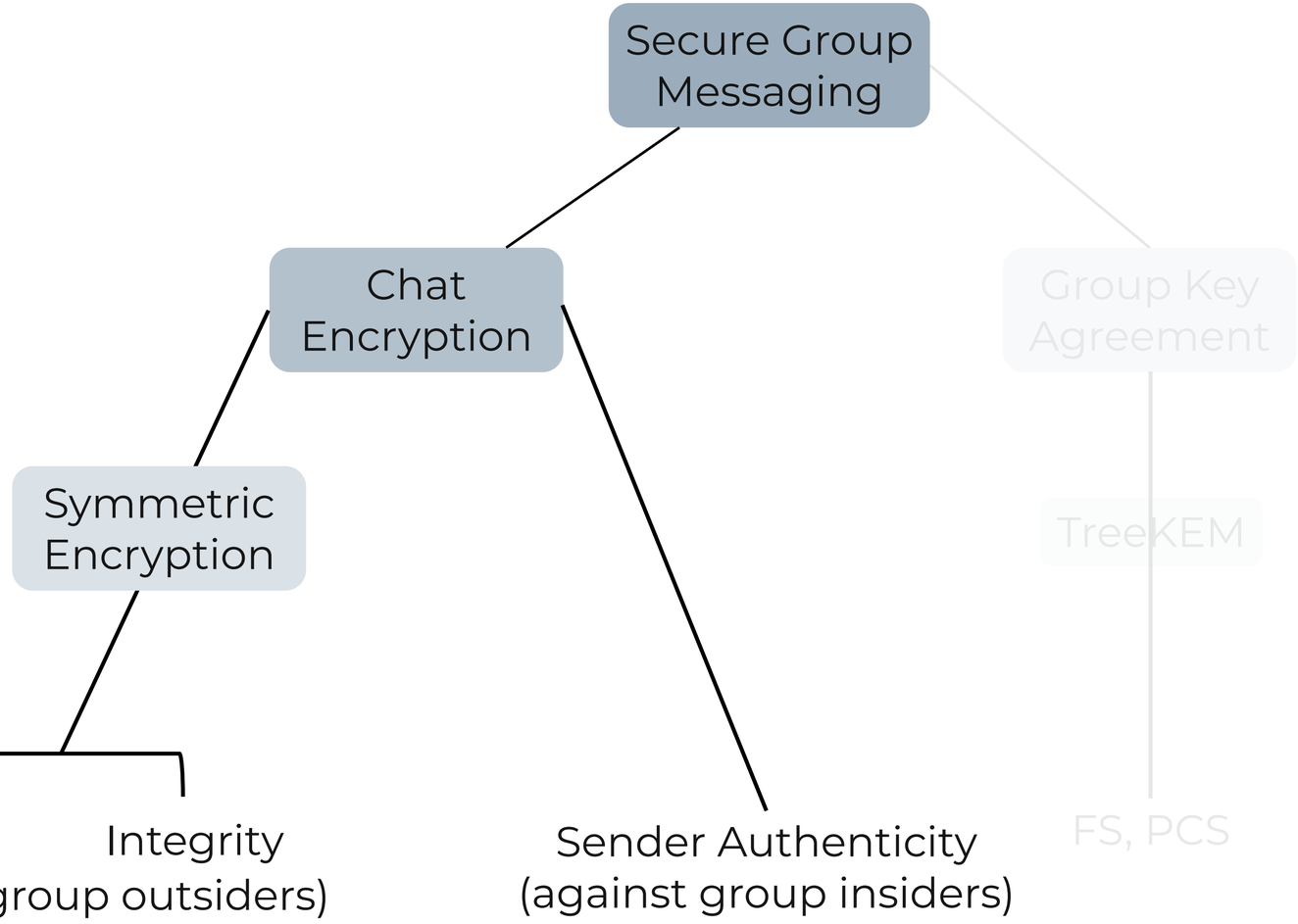
Secure Group Messaging



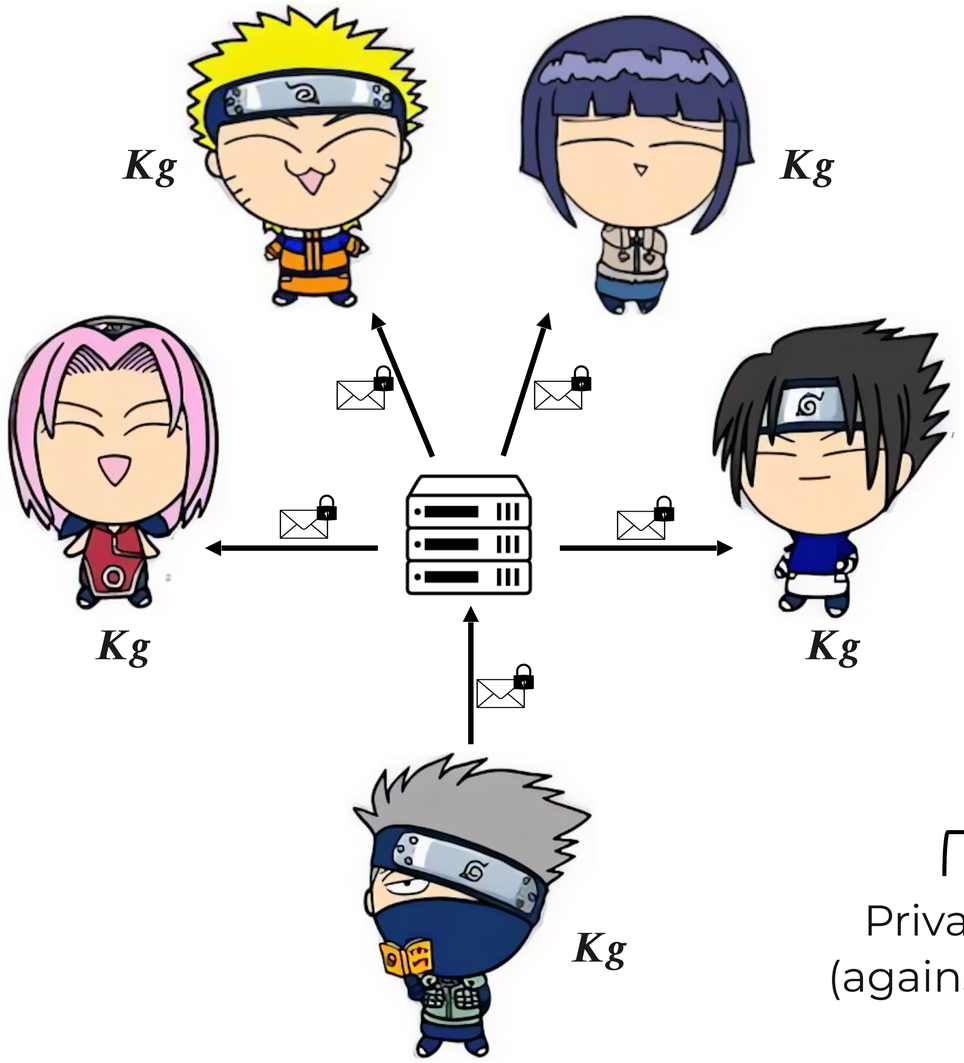


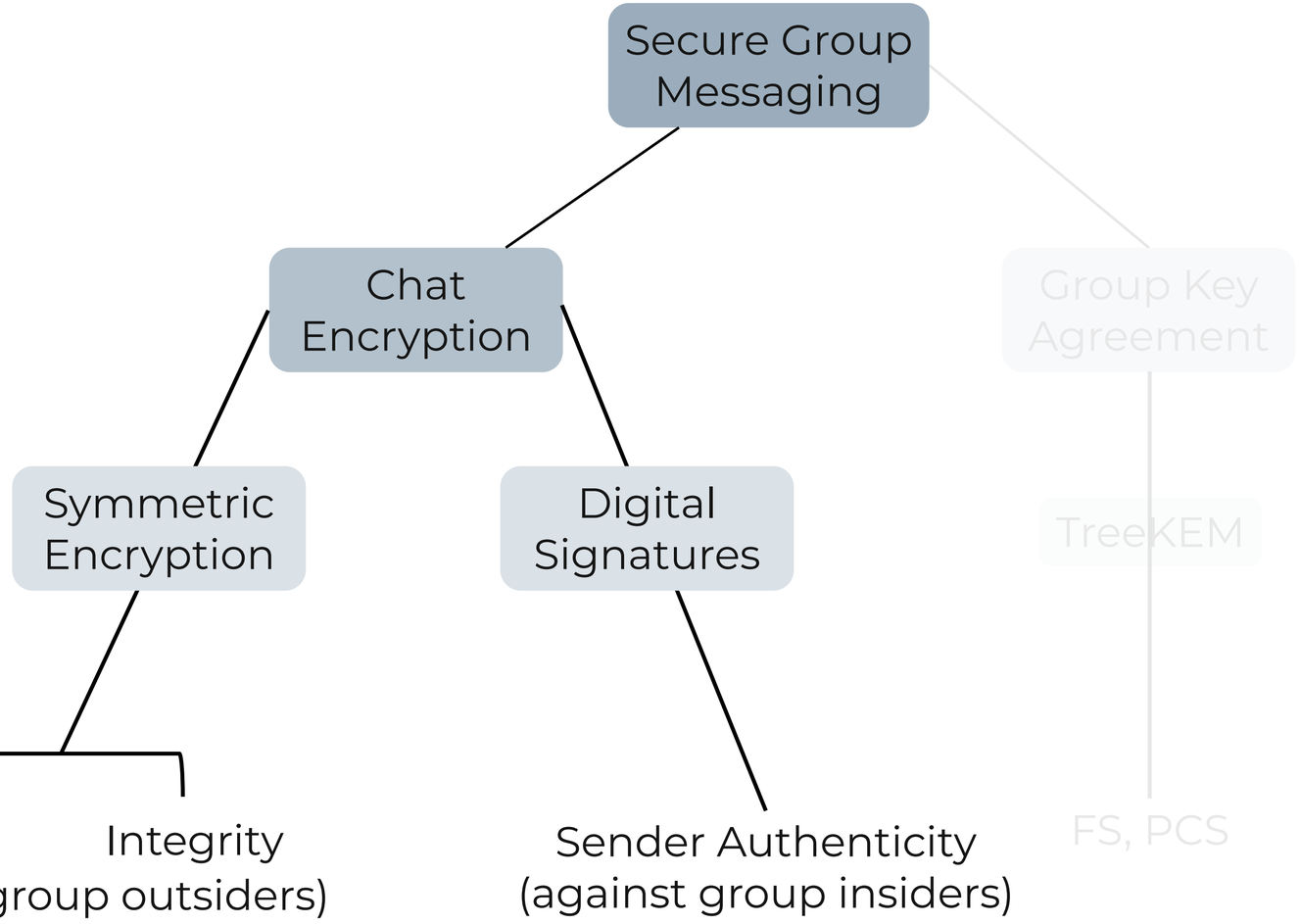
Secure Group Messaging



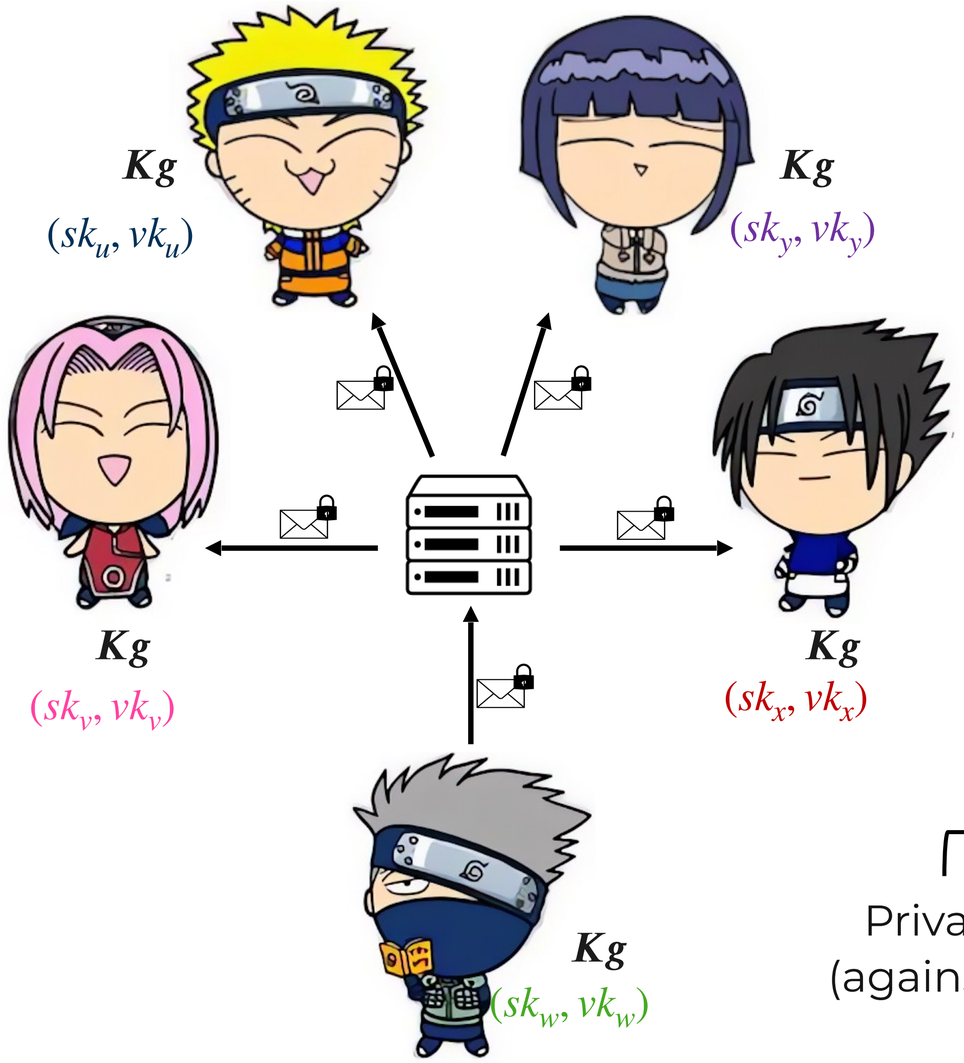


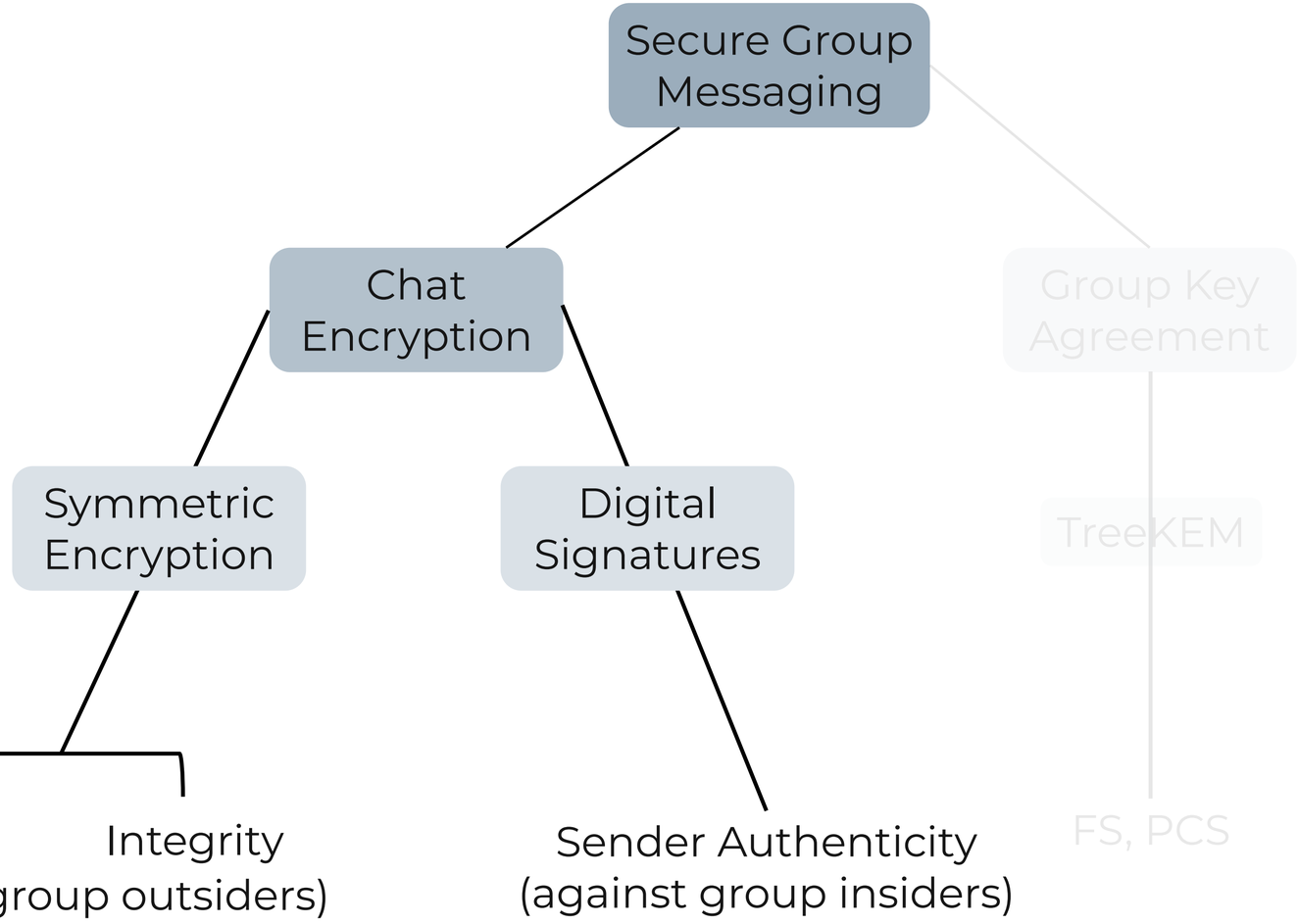
Secure Group Messaging



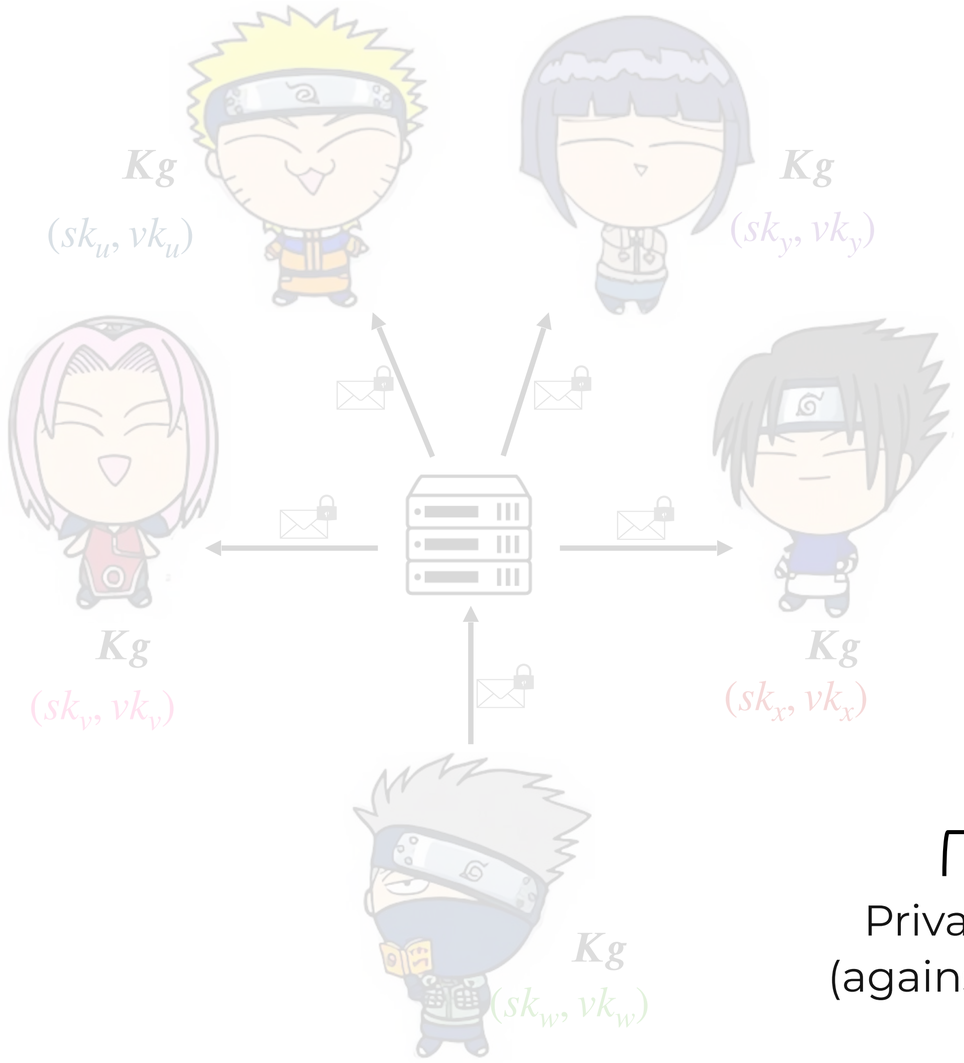


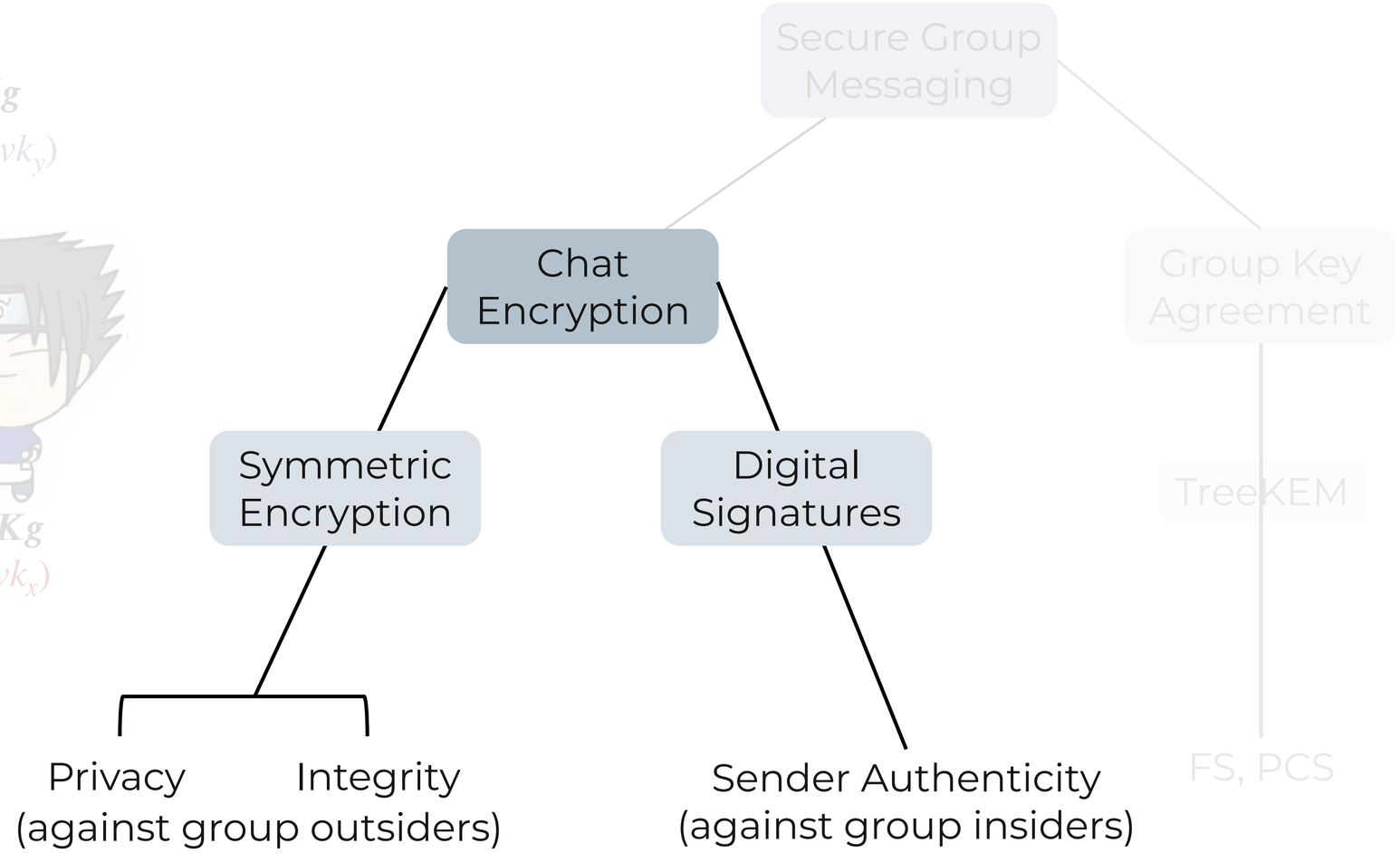
Secure Group Messaging



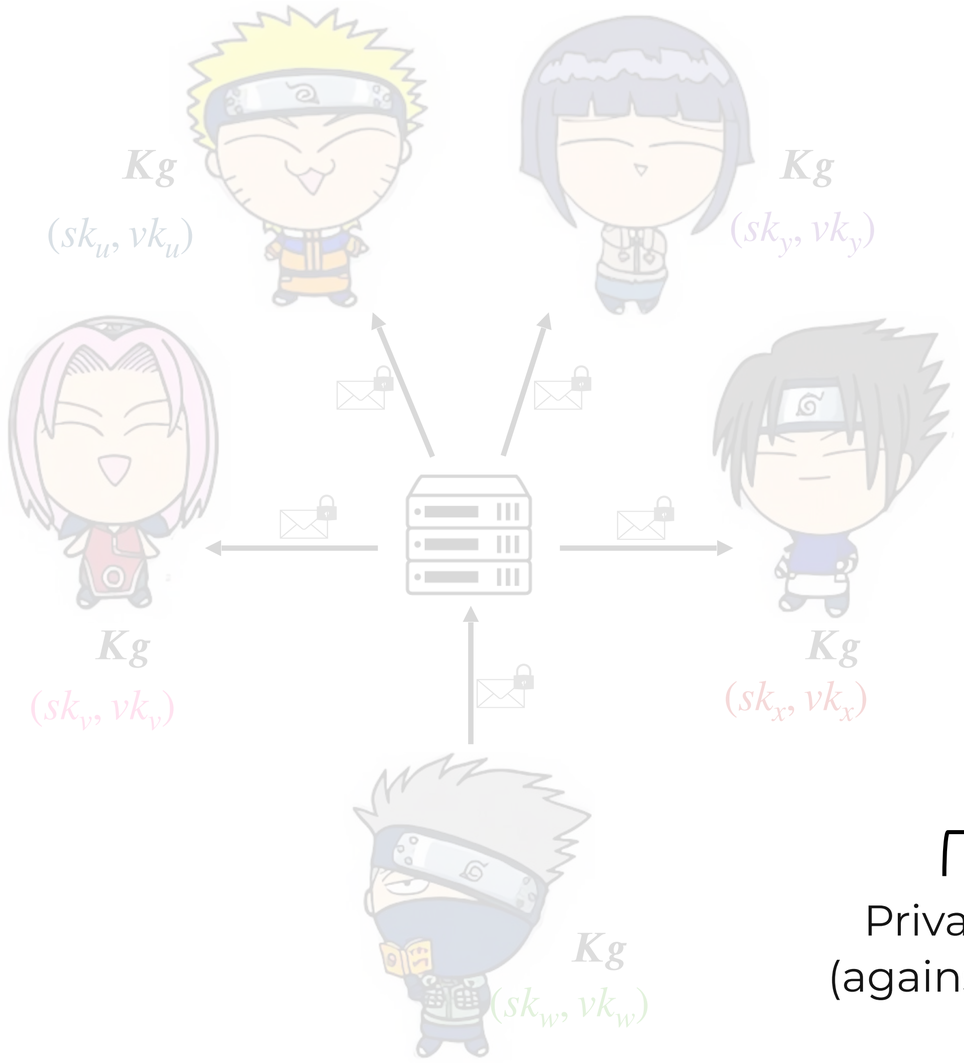


Secure Group Messaging





Secure Group Messaging



Symmetric Signcryption

Secure Group Messaging

Chat Encryption

Group Key Agreement

Symmetric Encryption

Digital Signatures

TreeKEM

Privacy

Integrity

(against group outsiders)

Sender Authenticity

(against group insiders)

FS, PCS

Methodology

Methodology

Symmetric Signcryption and E2EE Group Messaging in Keybase

Joseph Jaeger[1], Akshaya Kumar[1], and Igors Stepanovs[2]

Analyzing Group Chat Encryption in MLS, Session, Signal, and Matrix

Joseph Jaeger and Akshaya Kumar

Methodology

Symmetric Signcryption and E2EE Group Messaging in Keybase

Joseph Jaeger[1], Akshaya Kumar[1], and Igors Stepanovs[2]

Analyzing Group Chat Encryption in MLS, Session, Signal, and Matrix

Joseph Jaeger and Akshaya Kumar

Symmetric Signcryption Model

Methodology

Symmetric Signcryption and E2EE Group Messaging in Keybase

Joseph Jaeger[1], Akshaya Kumar[1], and Igors Stepanovs[2]

Analyzing Group Chat Encryption in MLS, Session, Signal, and Matrix

Joseph Jaeger and Akshaya Kumar

Symmetric Signcryption Model

Application

MLS

matrix

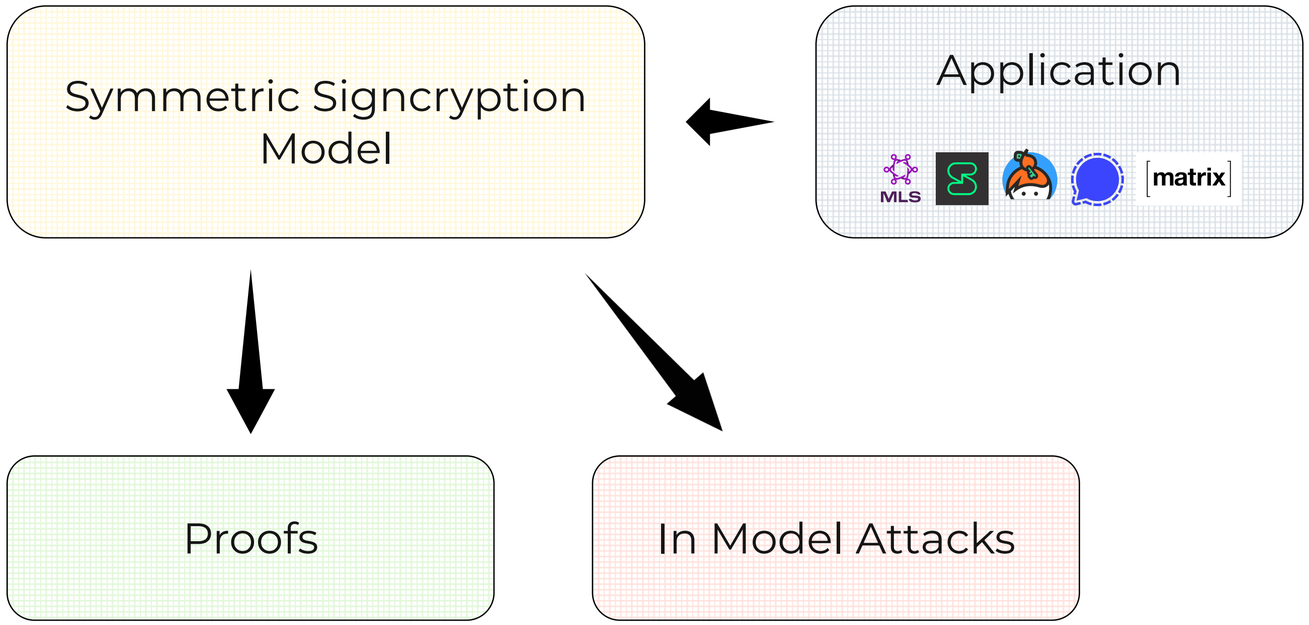
Methodology

Symmetric Signcryption and E2EE Group Messaging in Keybase

Joseph Jaeger[1], Akshaya Kumar[1], and Igors Stepanovs[2]

Analyzing Group Chat Encryption in MLS, Session, Signal, and Matrix

Joseph Jaeger and Akshaya Kumar



Methodology

Symmetric Signcryption and E2EE Group Messaging in Keybase

Joseph Jaeger[1], Akshaya Kumar[1], and Igors Stepanovs[2]

Analyzing Group Chat Encryption in MLS, Session, Signal, and Matrix

Joseph Jaeger and Akshaya Kumar

Application

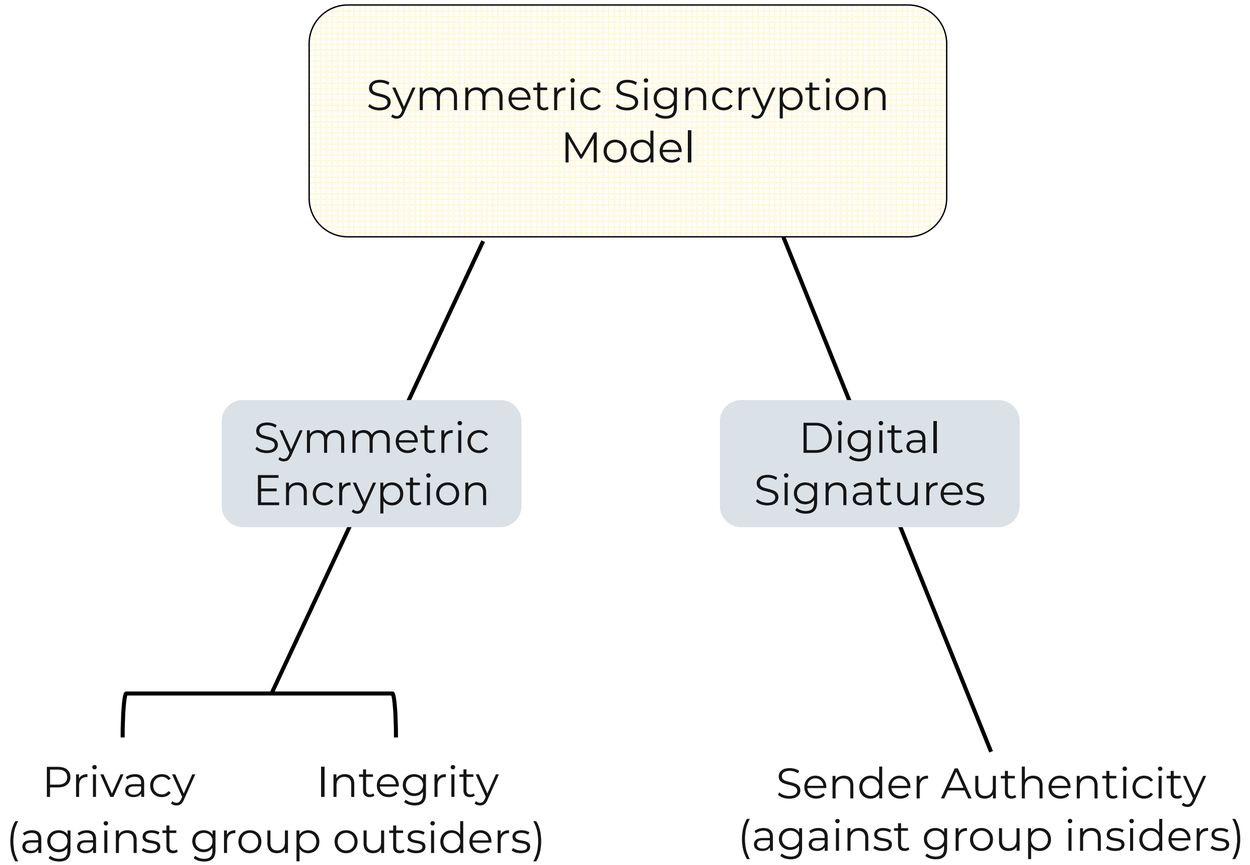
MLS matrix

Symmetric Signcryption Model

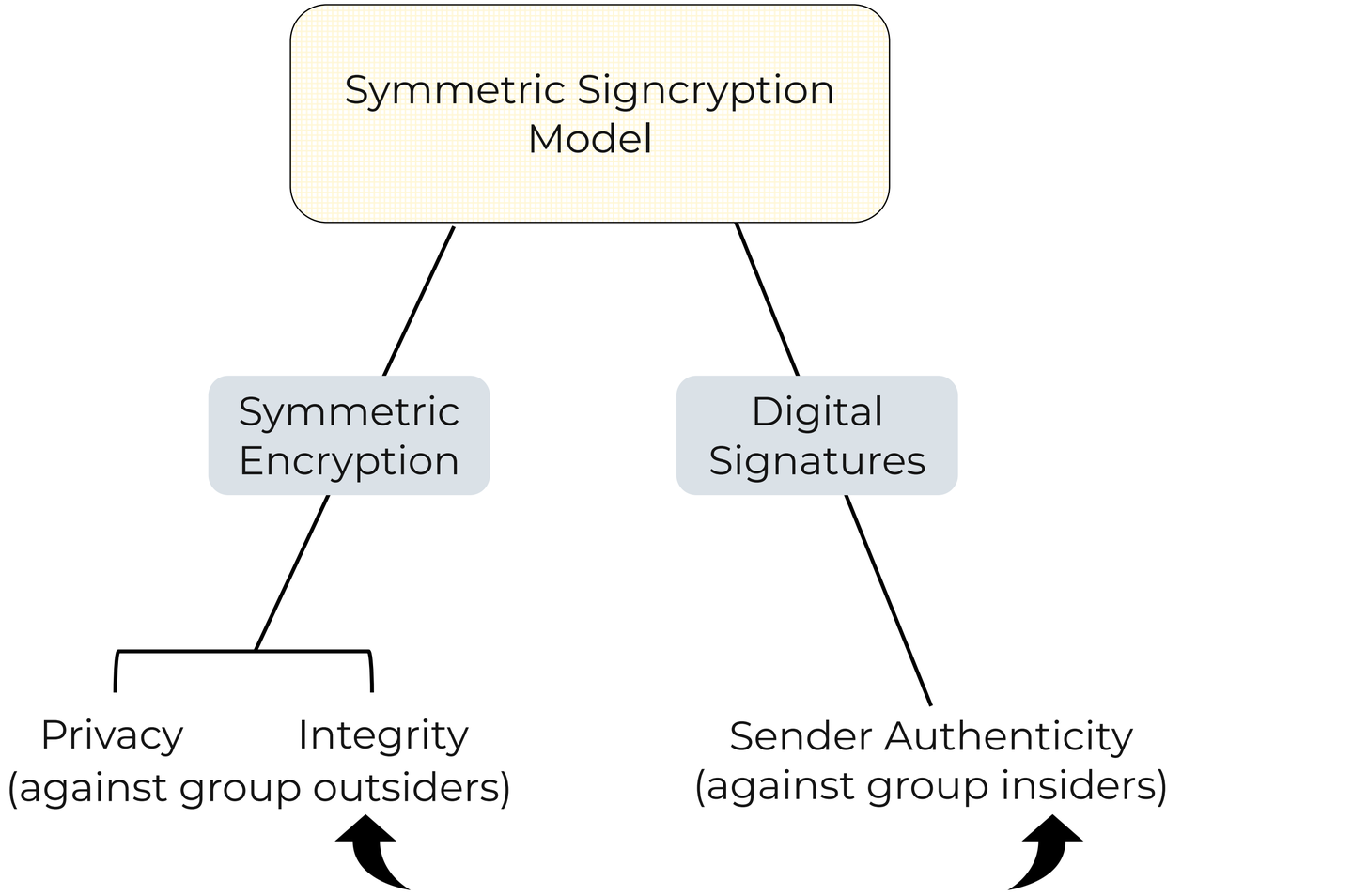
Proofs

In Model Attacks

Insider Attacks and MLS

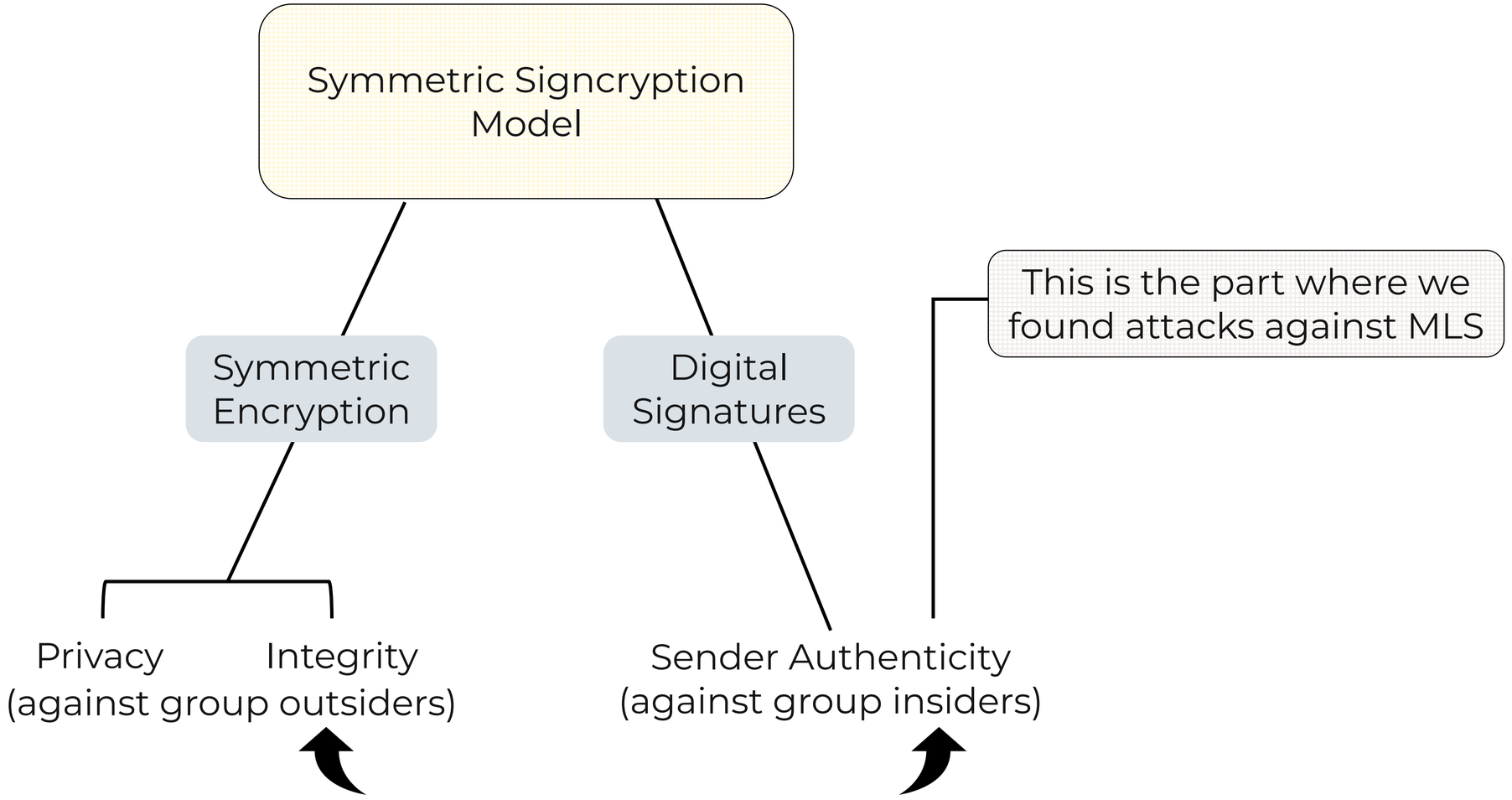


Insider Attacks and MLS



Natural to think about security against insiders and outsiders

Insider Attacks and MLS



Insider Attacks and MLS

Symmetric Signcryption Model

MLS aims to protect against forgeries by group members (aka insiders)

“[Knowledge] of the AEAD keys allows the attacker to send an encrypted message using that key, but cannot send a message to a group which appears to be from any valid client since they cannot forge the signature.”

art where we
s against MLS

Privacy (against group outsiders)

Integrity

Sender Authenticity (against group insiders)

Natural to think about security against insiders and outsiders

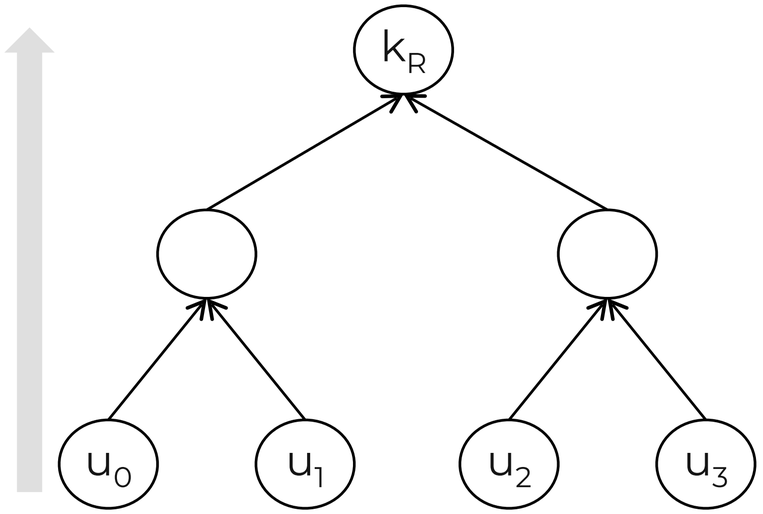
Encryption Key Derivation in MLS

Encryption Key Derivation in MLS

Ratchet tree

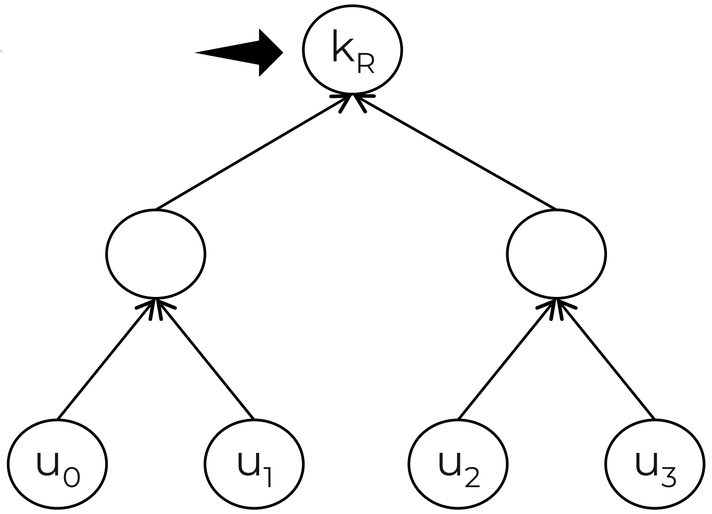
Encryption Key Derivation in MLS

Ratchet tree



Encryption Key Derivation in MLS

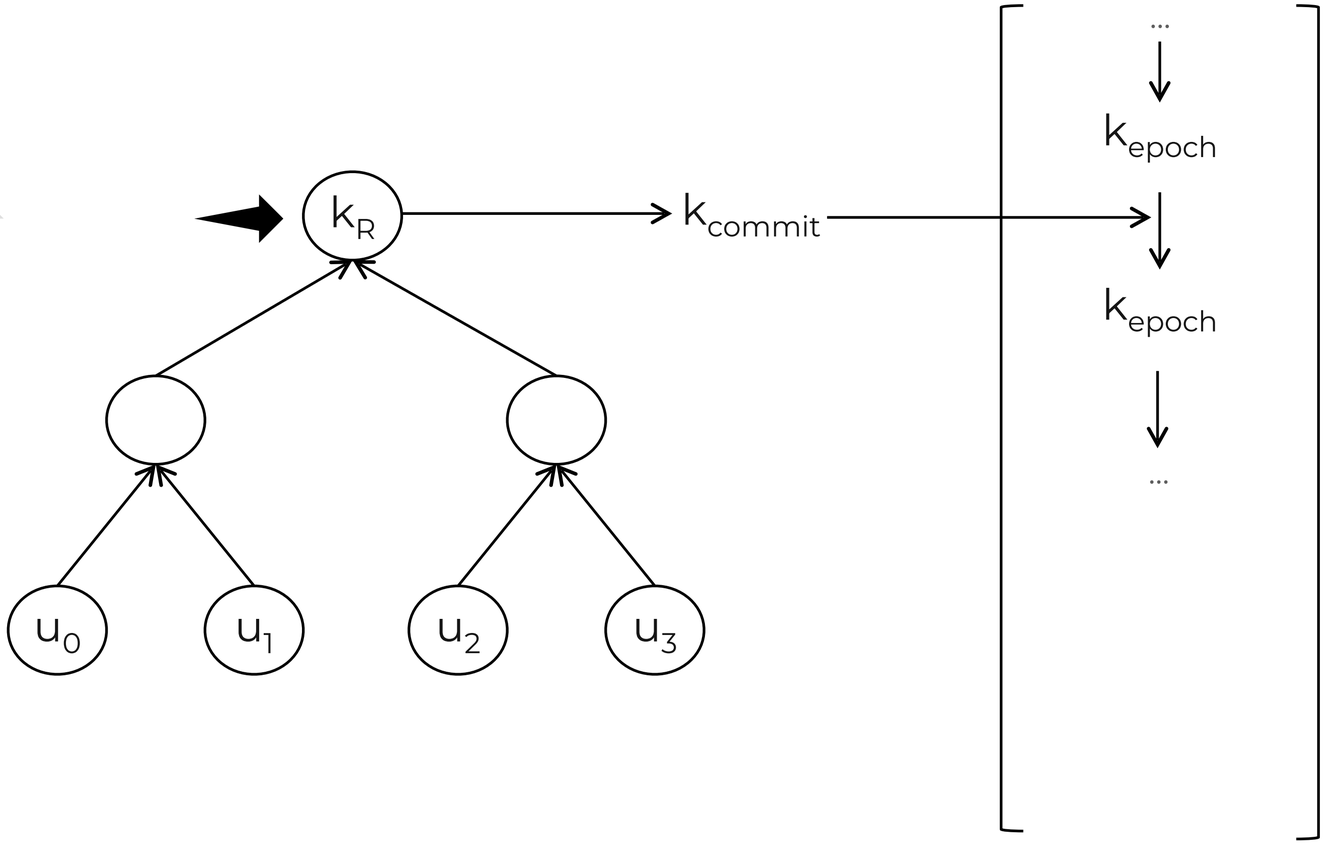
Ratchet tree



Encryption Key Derivation in MLS

Ratchet tree

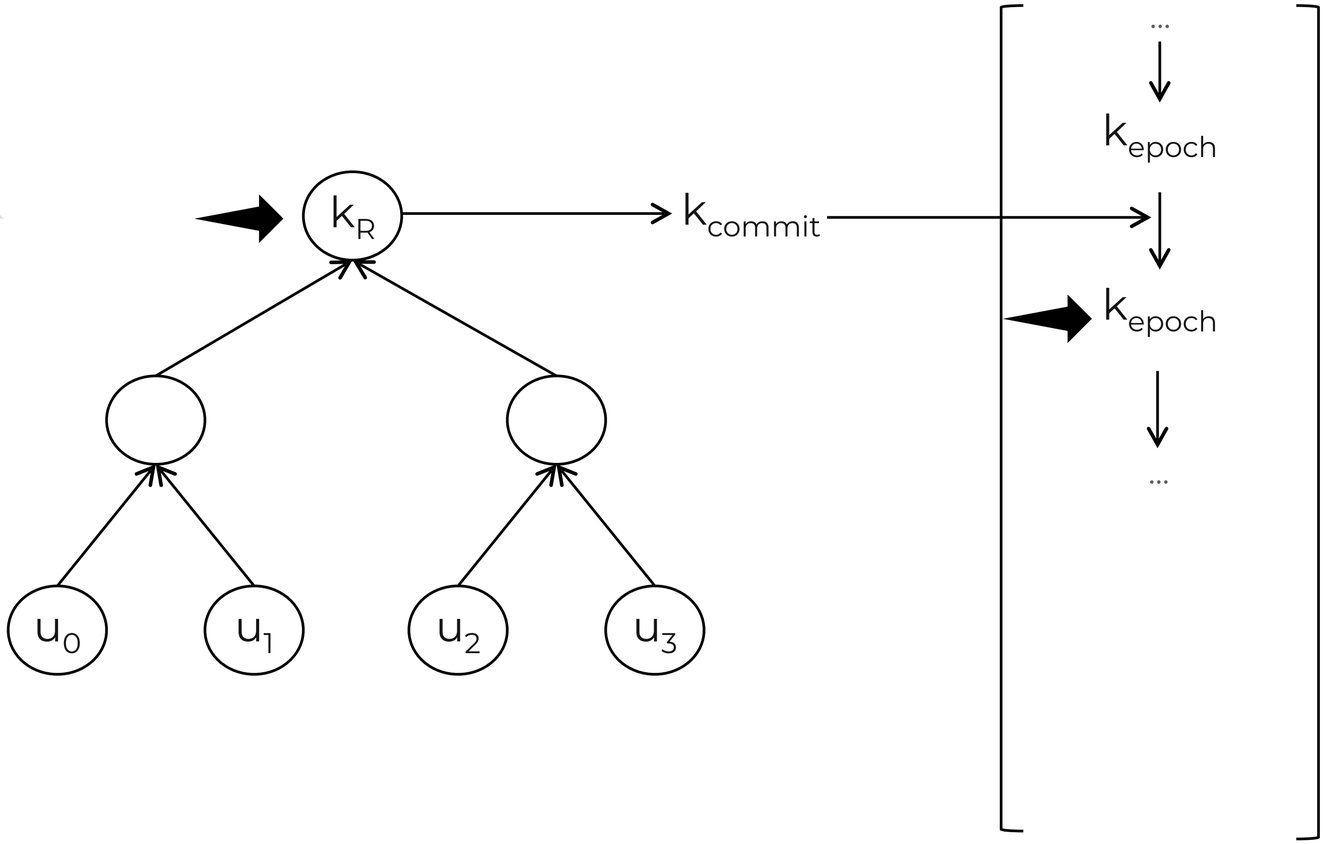
Key schedule



Encryption Key Derivation in MLS

Ratchet tree

Key schedule

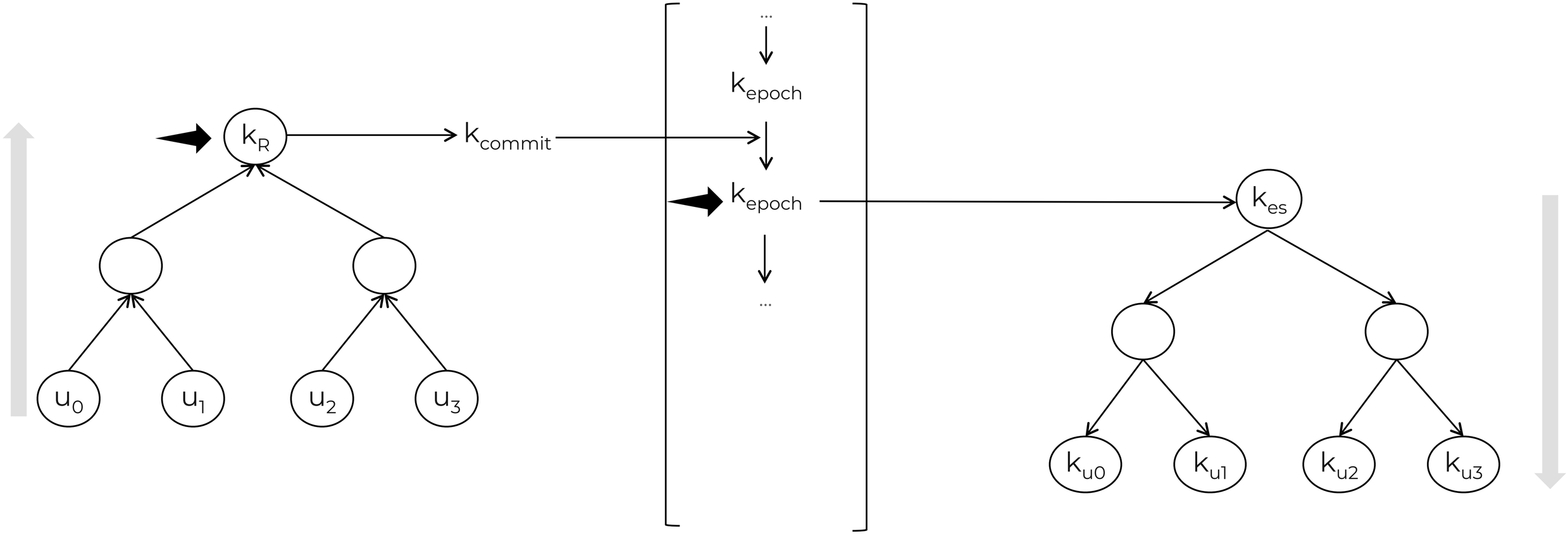


Encryption Key Derivation in MLS

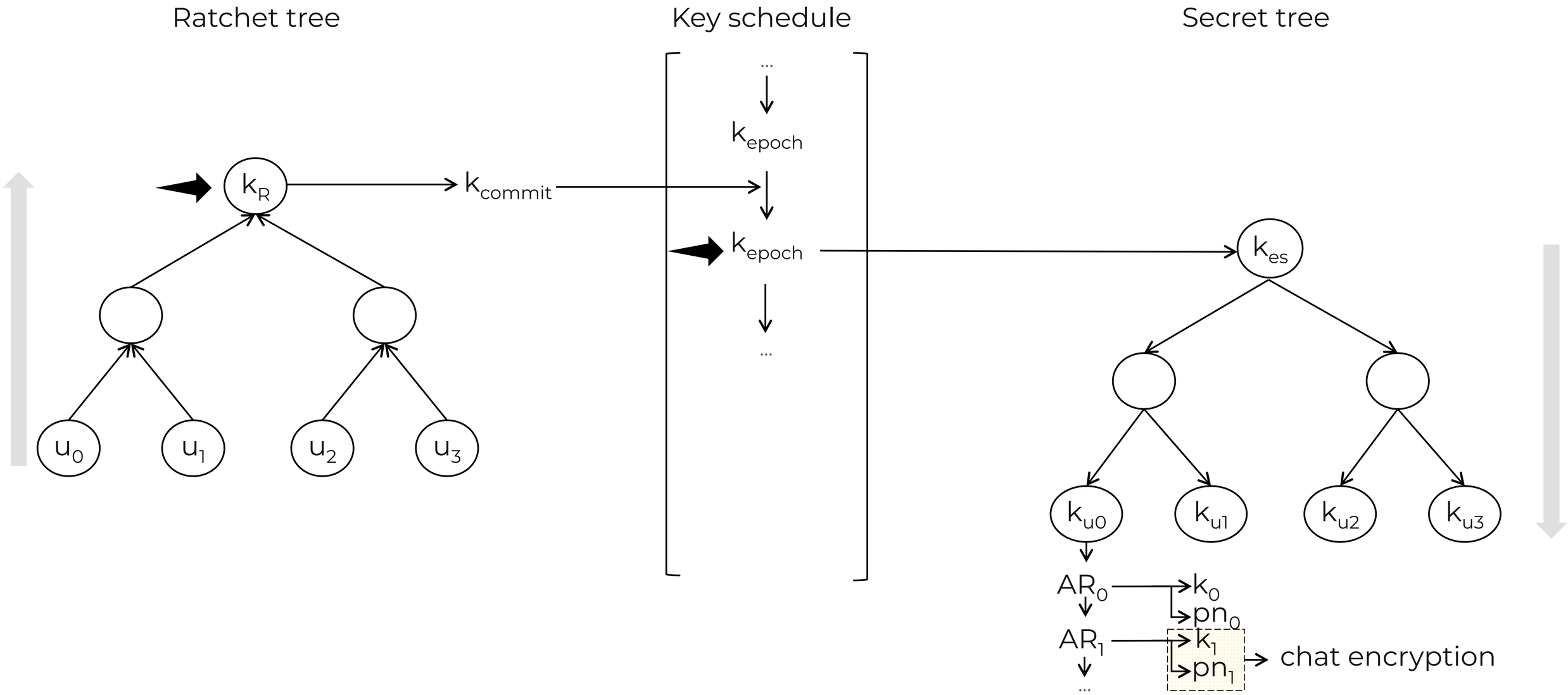
Ratchet tree

Key schedule

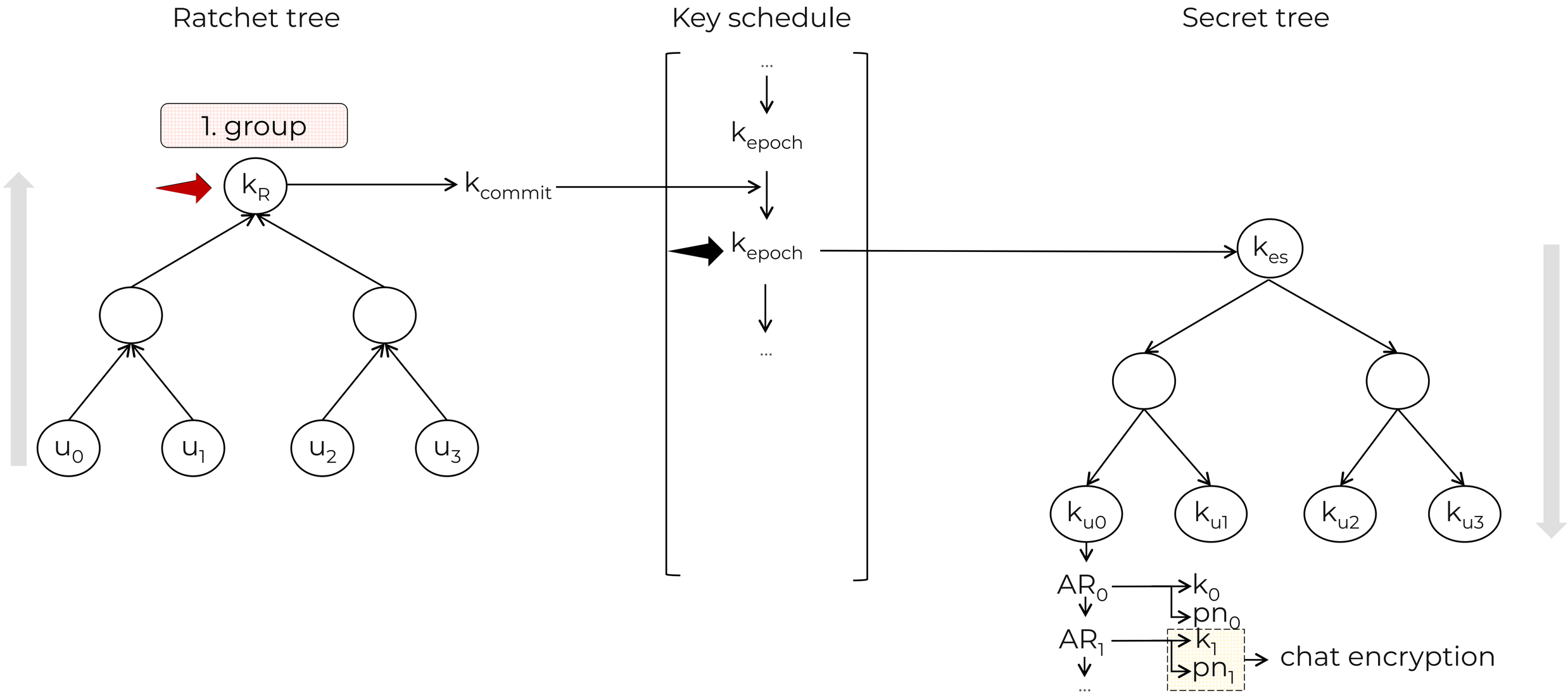
Secret tree



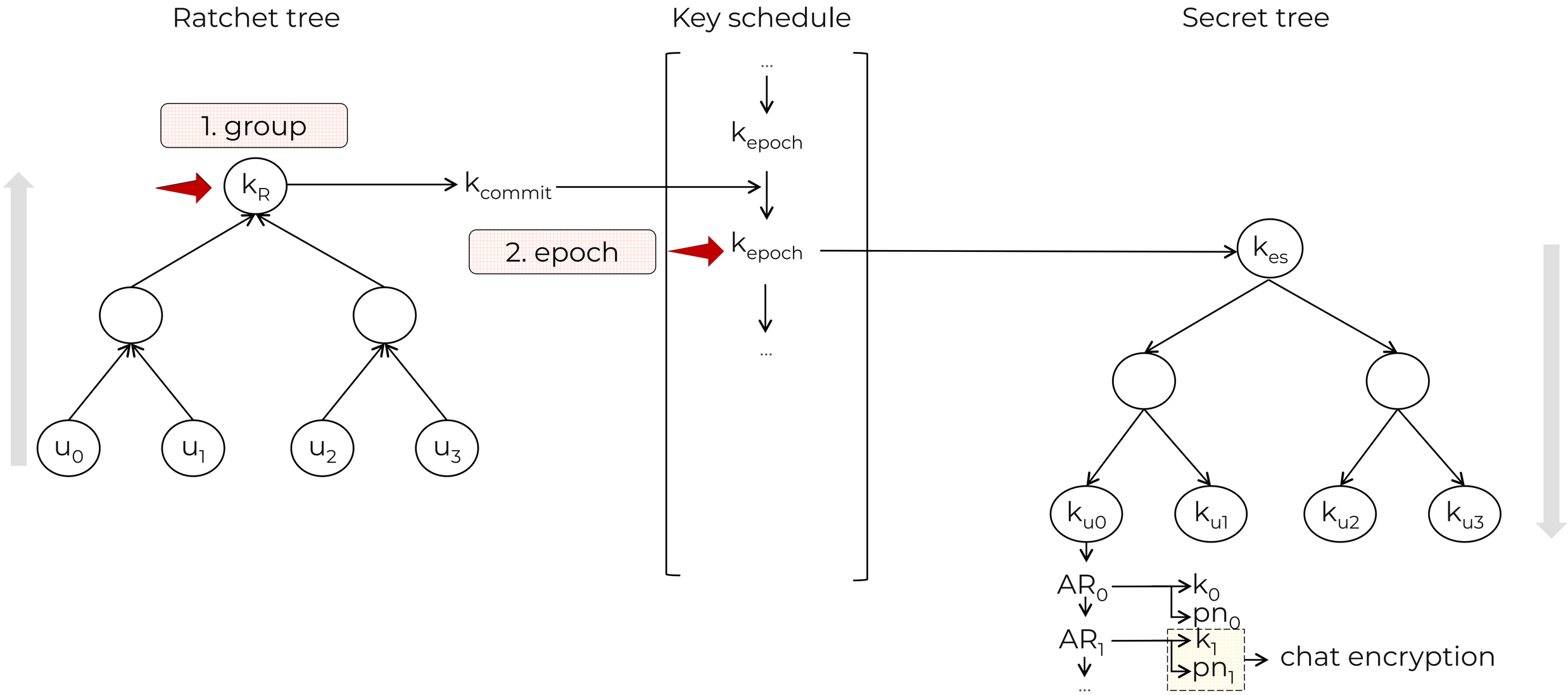
Encryption Key Derivation in MLS



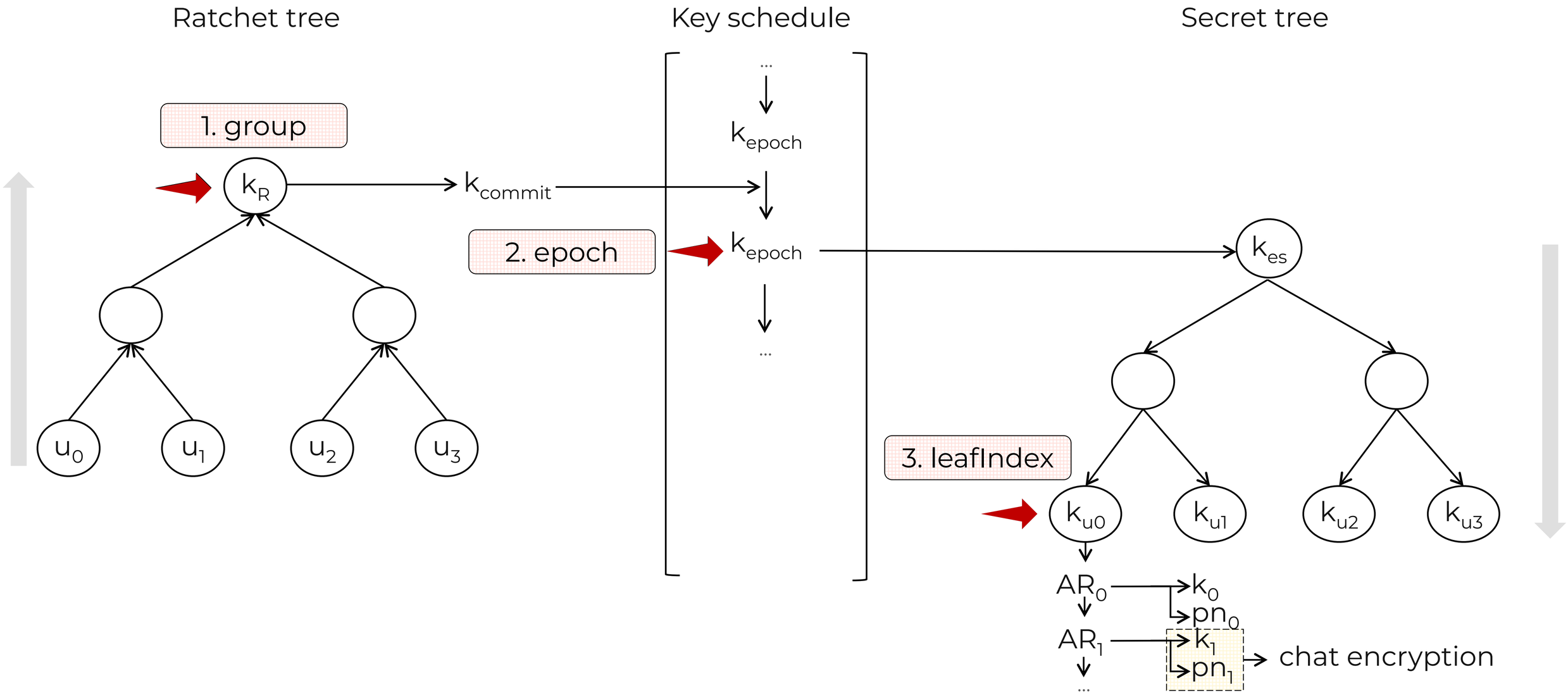
Encryption Key Derivation in MLS



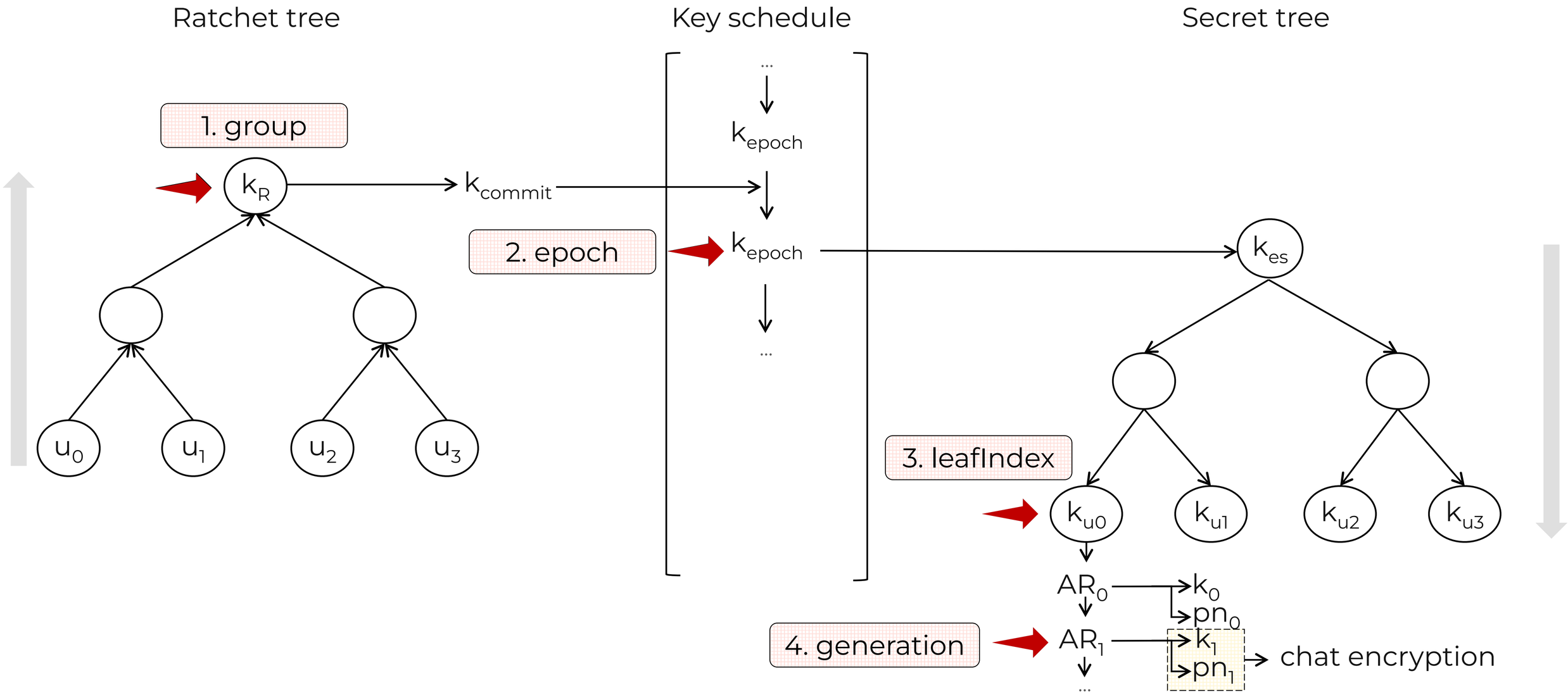
Encryption Key Derivation in MLS



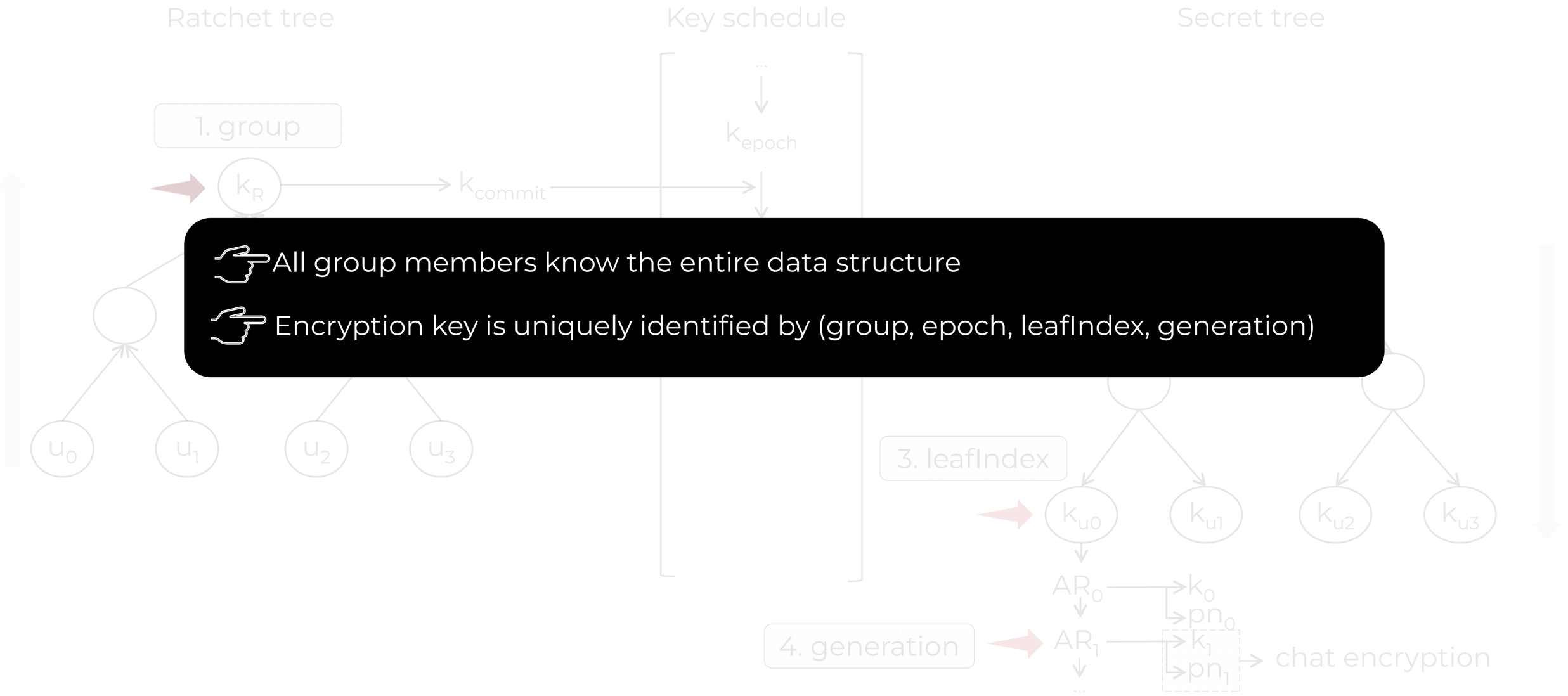
Encryption Key Derivation in MLS



Encryption Key Derivation in MLS



Encryption Key Derivation in MLS



- All group members know the entire data structure
- Encryption key is uniquely identified by (group, epoch, leafIndex, generation)

Chat Encryption in MLS

Chat Encryption in MLS

Chat encryption in MLS composes a digital signature scheme and a nonce-based encryption scheme in a Sign-then-Encrypt fashion

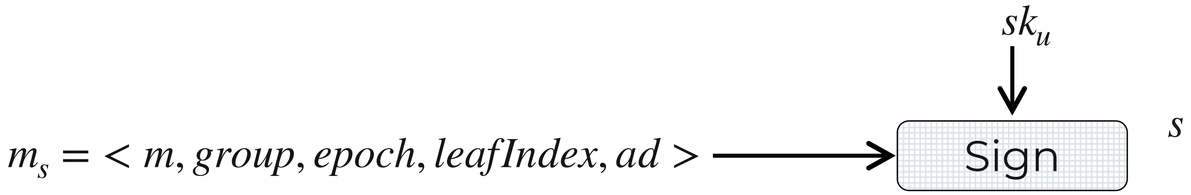
MLS-Sign-then-Encrypt

u: username

Chat Encryption in MLS

Chat encryption in MLS composes a digital signature scheme and a nonce-based encryption scheme in a Sign-then-Encrypt fashion

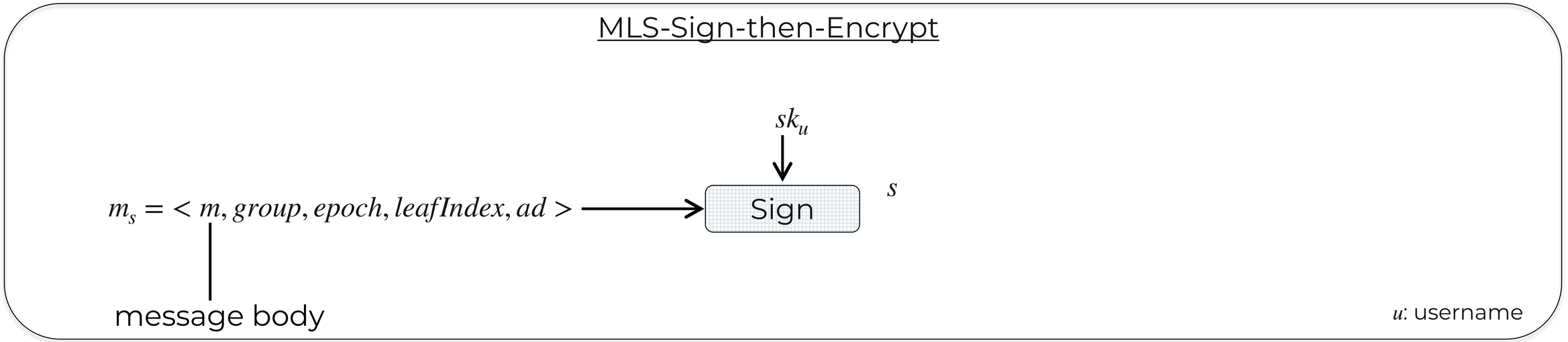
MLS-Sign-then-Encrypt



u: username

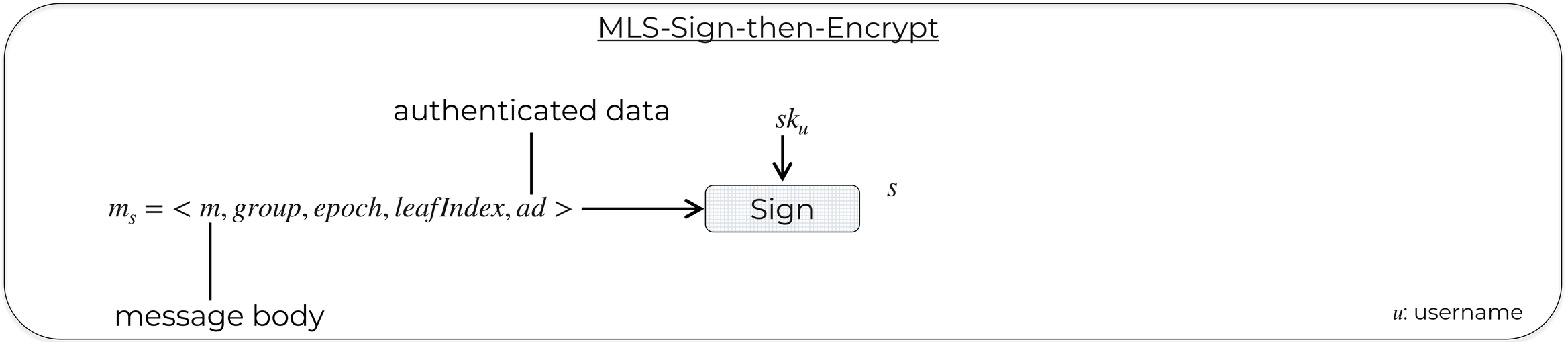
Chat Encryption in MLS

Chat encryption in MLS composes a digital signature scheme and a nonce-based encryption scheme in a Sign-then-Encrypt fashion



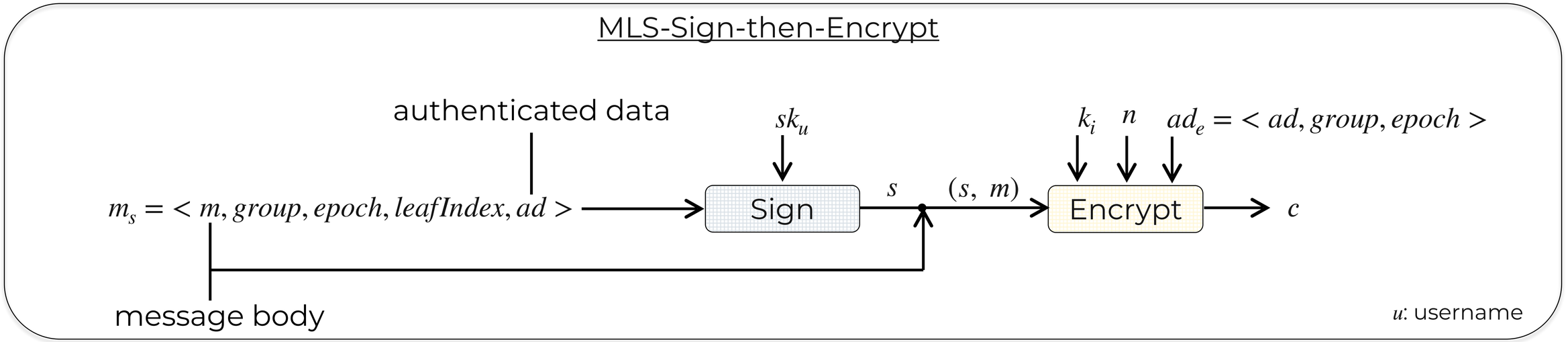
Chat Encryption in MLS

Chat encryption in MLS composes a digital signature scheme and a nonce-based encryption scheme in a Sign-then-Encrypt fashion



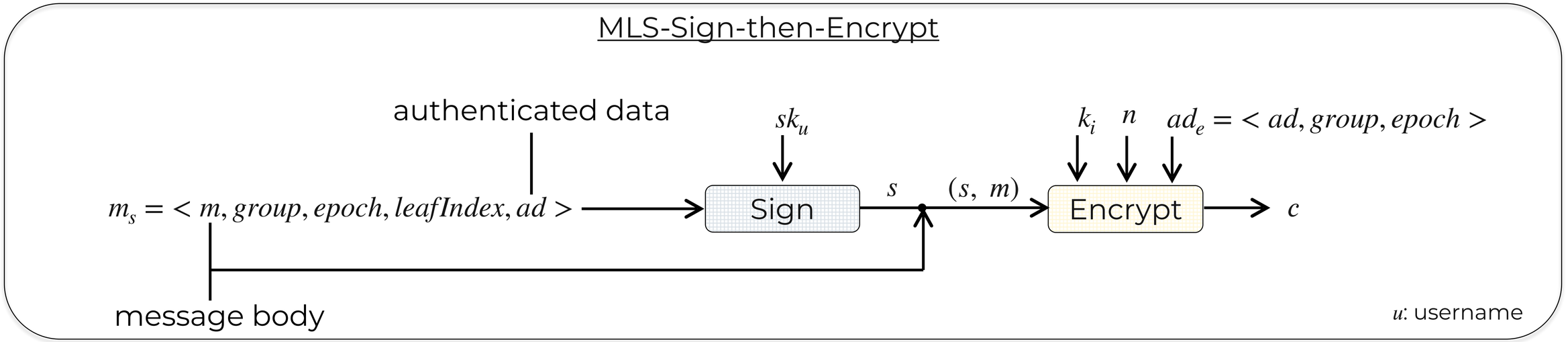
Chat Encryption in MLS

Chat encryption in MLS composes a digital signature scheme and a nonce-based encryption scheme in a Sign-then-Encrypt fashion



Chat Encryption in MLS

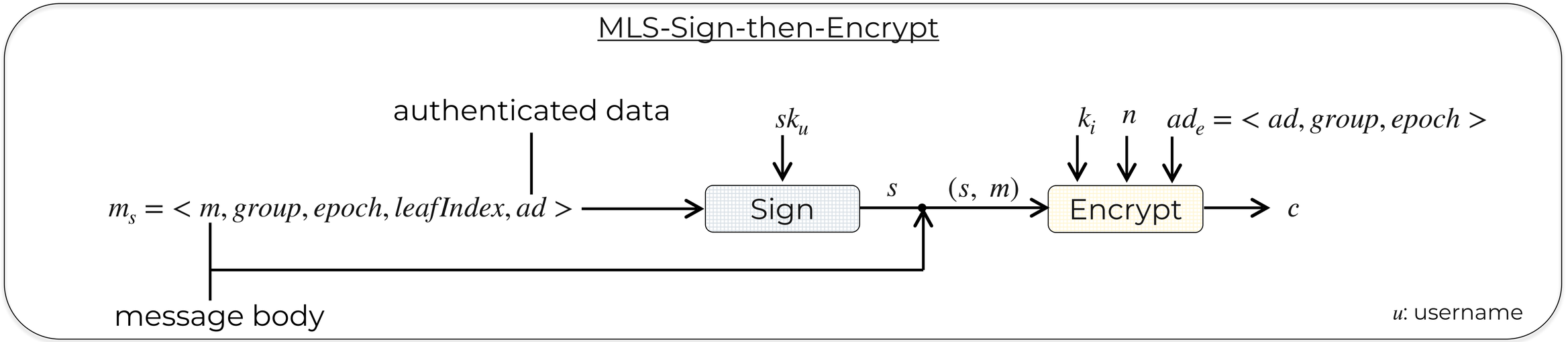
Chat encryption in MLS composes a digital signature scheme and a nonce-based encryption scheme in a Sign-then-Encrypt fashion



Intuition: s should authenticate the key identifier so that group insider cannot re-encrypt $(s,\ m)$ using a different k and replay message to group

Chat Encryption in MLS

Chat encryption in MLS composes a digital signature scheme and a nonce-based encryption scheme in a Sign-then-Encrypt fashion

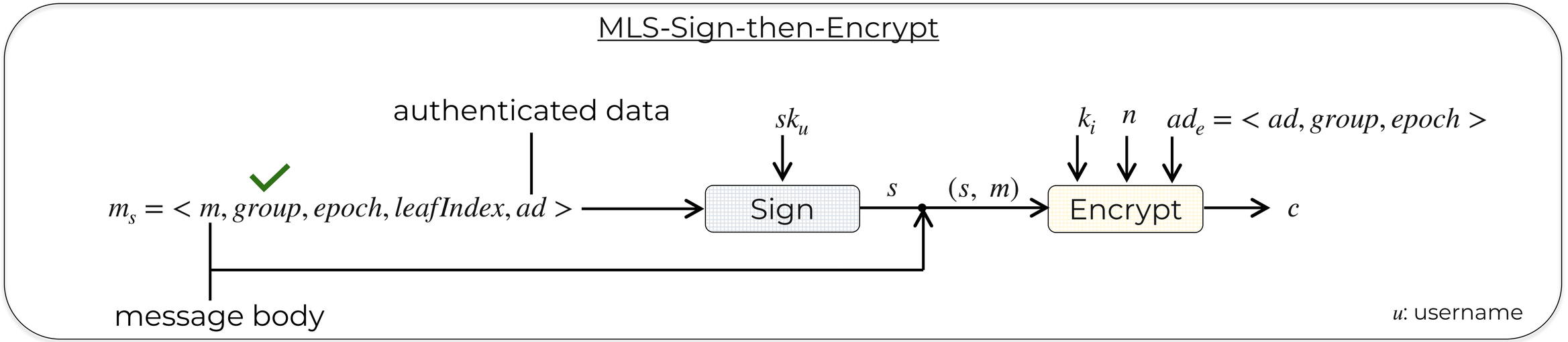


Intuition: s should authenticate the key identifier so that group insider cannot re-encrypt $(s,\ m)$ using a different k and replay message to group

Recall: key identifier $g = (group, epoch, leafIndex, generation)$

Chat Encryption in MLS

Chat encryption in MLS composes a digital signature scheme and a nonce-based encryption scheme in a Sign-then-Encrypt fashion

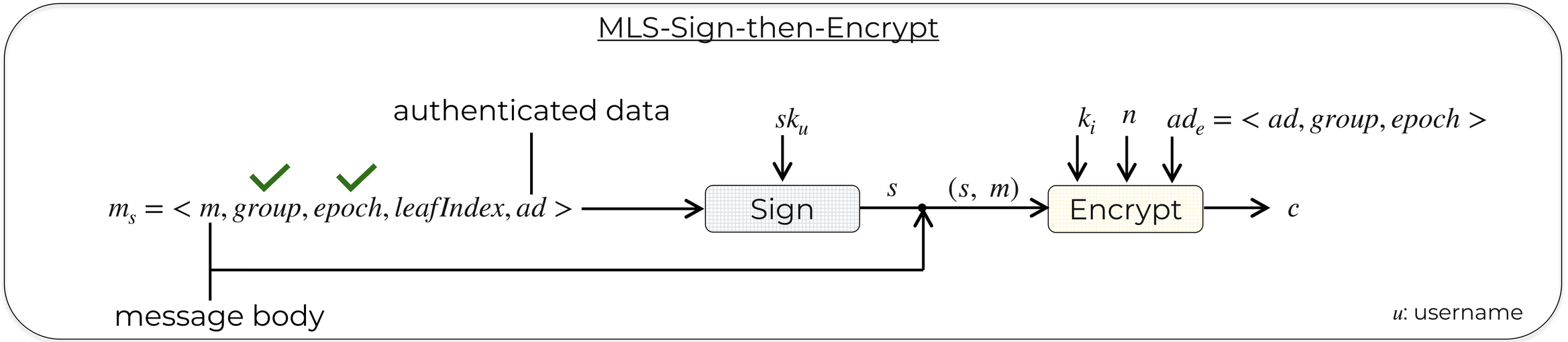


Intuition: s should authenticate the key identifier so that group insider cannot re-encrypt $(s,\ m)$ using a different k and replay message to group

Recall: key identifier $g = (group, epoch, leafIndex, generation)$

Chat Encryption in MLS

Chat encryption in MLS composes a digital signature scheme and a nonce-based encryption scheme in a Sign-then-Encrypt fashion

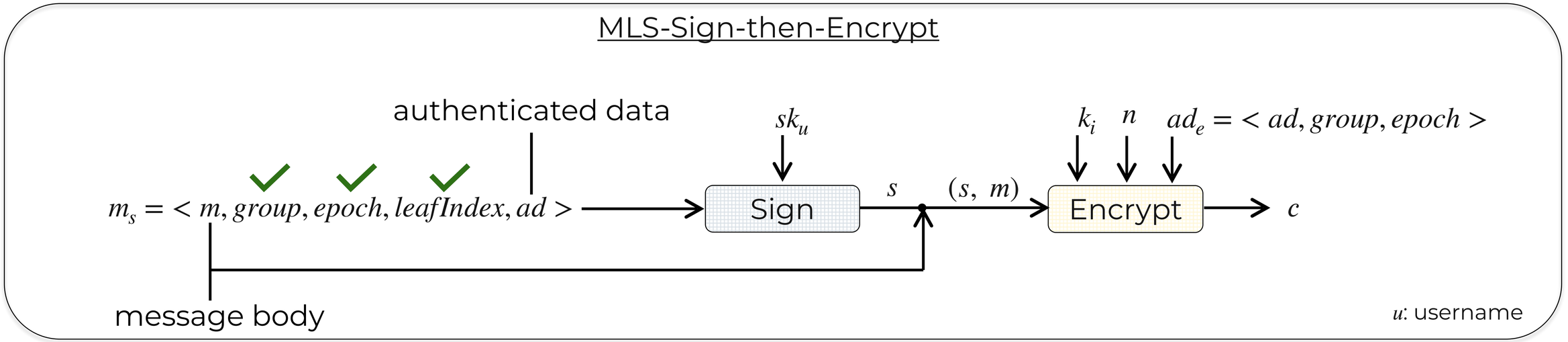


Intuition: s should authenticate the key identifier so that group insider cannot re-encrypt $(s,\ m)$ using a different k and replay message to group

Recall: key identifier $g = (group, epoch, leafIndex, generation)$

Chat Encryption in MLS

Chat encryption in MLS composes a digital signature scheme and a nonce-based encryption scheme in a Sign-then-Encrypt fashion

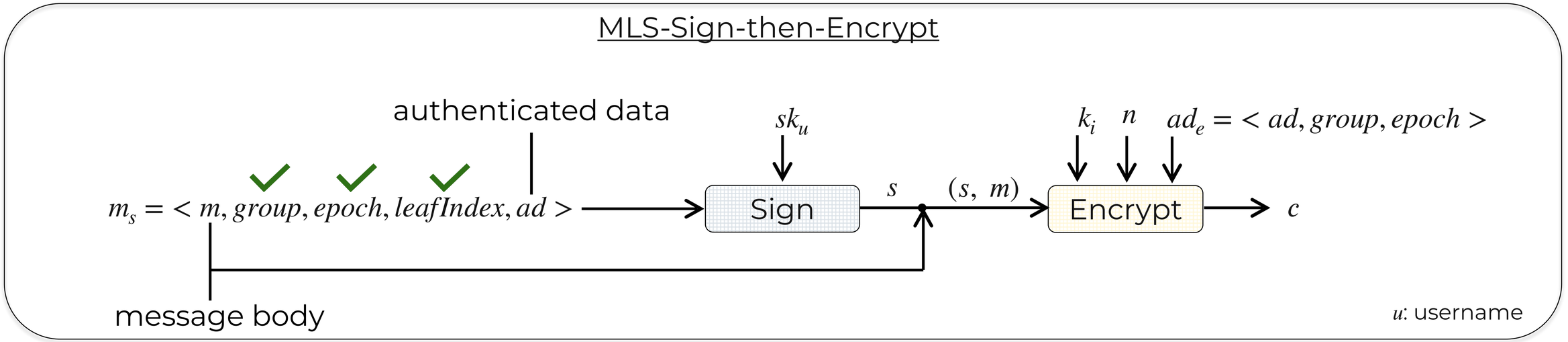


Intuition: s should authenticate the key identifier so that group insider cannot re-encrypt $(s,\ m)$ using a different k and replay message to group

Recall: key identifier $g = (group, epoch, leafIndex, generation)$

Chat Encryption in MLS

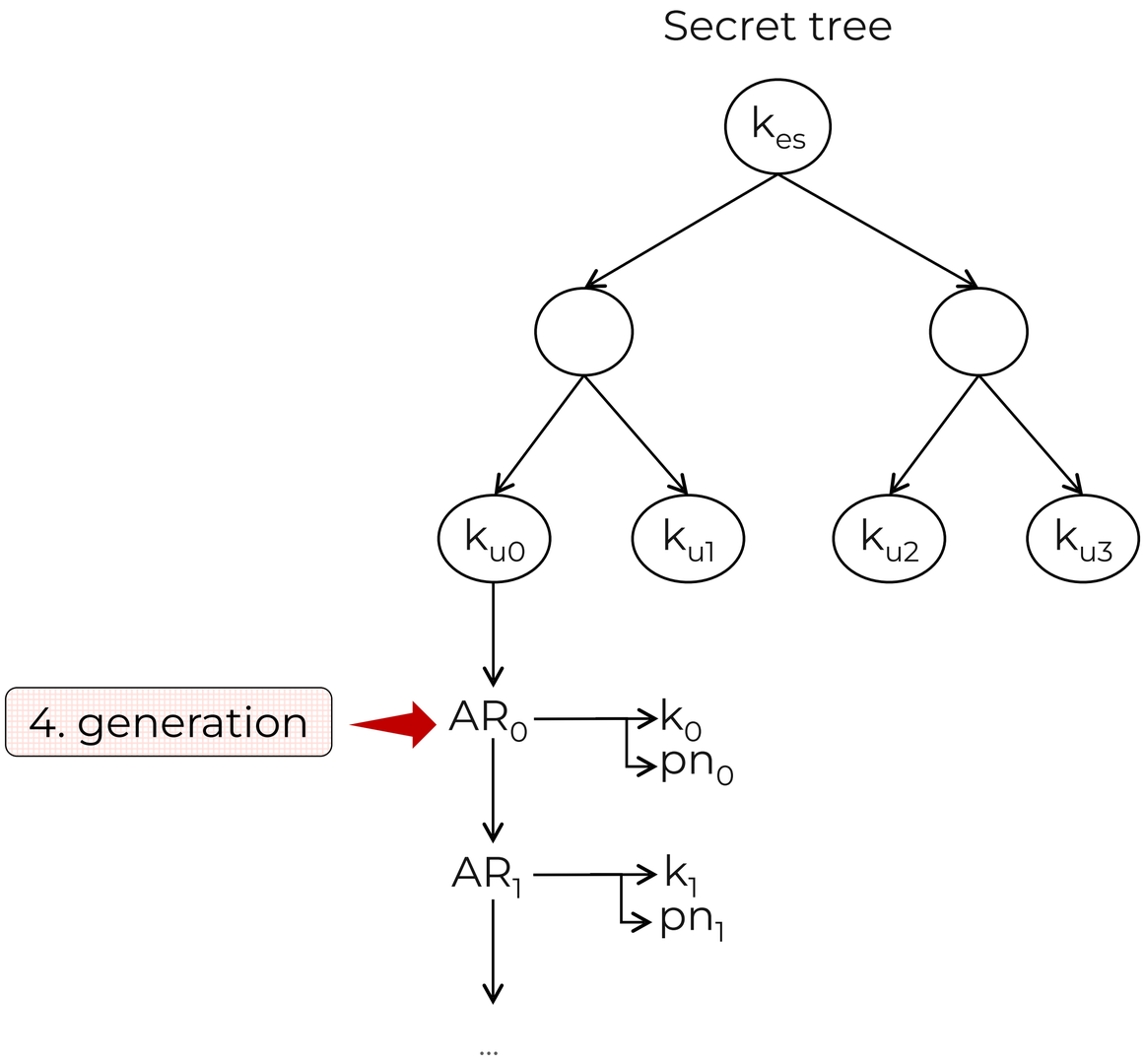
Chat encryption in MLS composes a digital signature scheme and a nonce-based encryption scheme in a Sign-then-Encrypt fashion



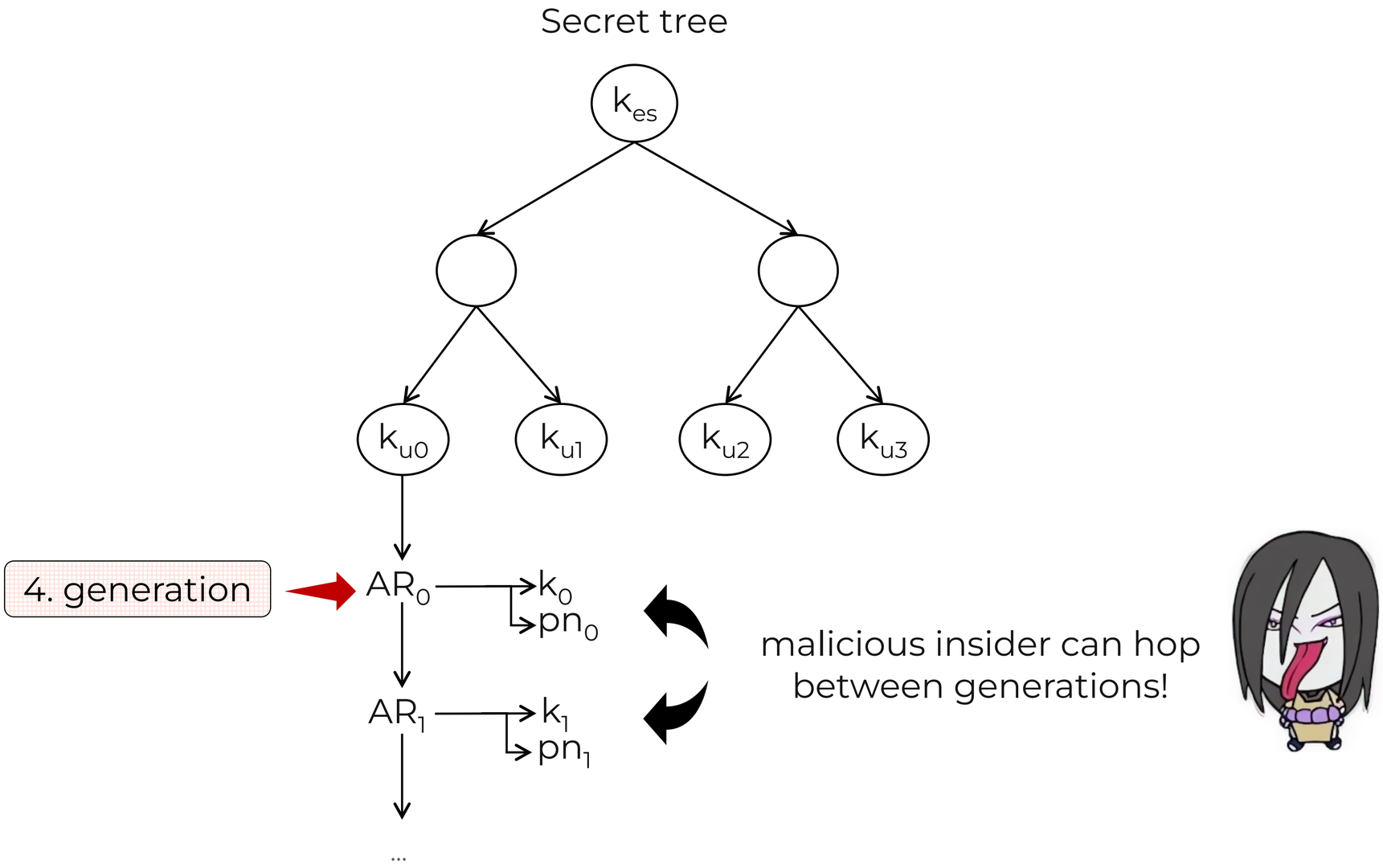
Intuition: s should authenticate the key identifier so that group insider cannot re-encrypt $(s,\ m)$ using a different k and replay message to group

Recall: key identifier $g = (group, epoch, leafIndex, generation)$

Encryption Key Derivation in MLS



Encryption Key Derivation in MLS



1. Insider Replay Attack

Insider Replay Attack

Insider Replay Attack

Insider Replay Attack





Insider Replay Attack





m ← Yes!

Insider Replay Attack





m ← Yes!

MLS-Sign-then-Encrypt

$s \leftarrow \text{Sign}(sk_u,\ m_s)$

$c \leftarrow \text{Enc}(k_{g_0},\ n,\ s\,||\,m,\ ad_e)$

c

c

u : user_id
g_0 : key_id

Insider Replay Attack





m ← Yes!

c

MLS-Sign-then-Encrypt

$s \leftarrow \text{Sign}(sk_u,\ m_s)$

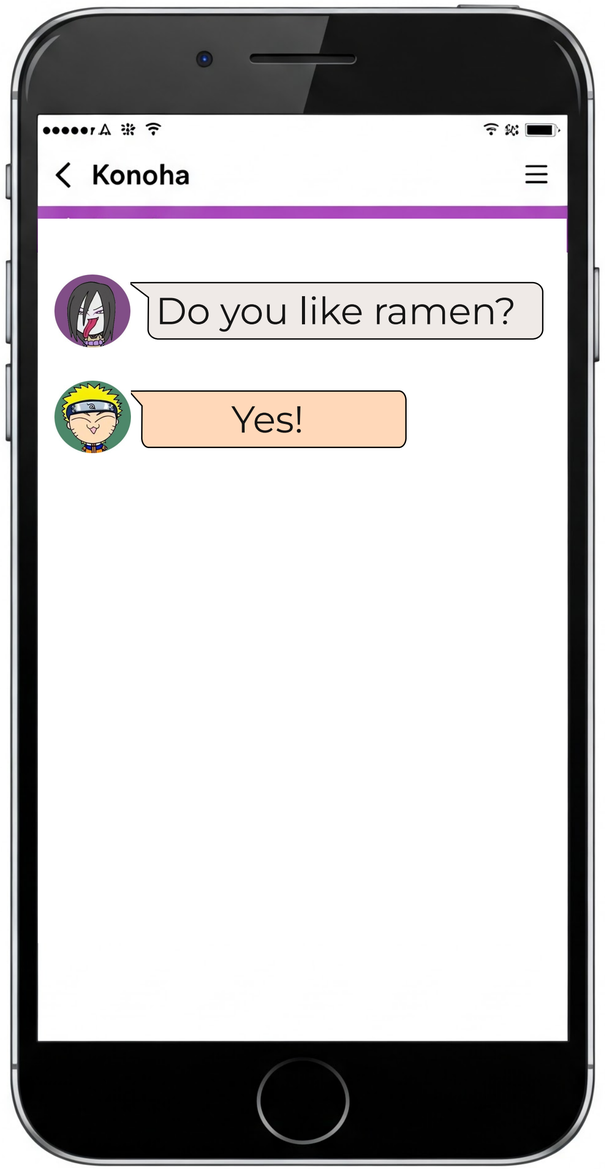
$c \leftarrow \text{Enc}(k_{g_0},\ n,\ s\,||\,m,\ ad_e)$

c

Let key identifier $g_0 = (group, epoch, leafIndex, generation_0)$

u : user_id
g_0 : key_id

Insider Replay Attack



Insider Replay Attack





g_0, g_1 : key_id

Insider Replay Attack

Recall: signature s does not authenticate the $generation$





g_0, g_1 : key_id

Insider Replay Attack

Recall: signature s does not authenticate the $generation$





Decrypt-then-Re-Encrypt

$$s||m \leftarrow \mathsf{Enc}(k_{g_0},\ n,\ c,\ ad_e)$$

$$c' \leftarrow \mathsf{Dec}(k_{g_1},\ n,\ s||m,\ ad_e)$$

g_0, g_1 : key_id

Insider Replay Attack

Recall: signature s does not authenticate the $generation$





Decrypt-then-Re-Encrypt

$$s||m \leftarrow \mathsf{Enc}(k_{g_0},\ n,\ c,\ ad_e)$$

$$c' \leftarrow \mathsf{Dec}(k_{g_1},\ n,\ s||m,\ ad_e)$$

$$g_0 = (group,\ epoch,\ leafIndex,\ generation_0)$$

$$g_1 = (group,\ epoch,\ leafIndex,\ generation_1)$$

$$(generation_1 > generation_0)$$

g_0, g_1 : key_id

Insider Replay Attack

Recall: signature s does not authenticate the $generation$





Save for later

Decrypt-then-Re-Encrypt

$$s||m \leftarrow \mathsf{Enc}(k_{g_0},\ n,\ c,\ ad_e)$$

$$c' \leftarrow \mathsf{Dec}(k_{g_1},\ n,\ s||m,\ ad_e)$$

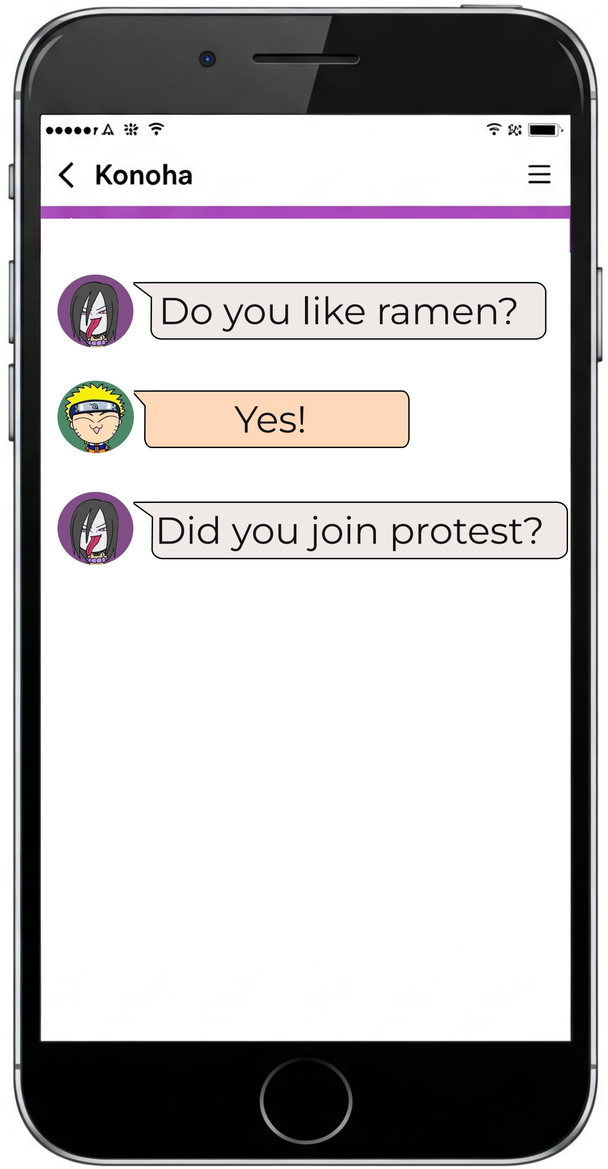
$$g_0 = (group,\ epoch,\ leafIndex,\ generation_0)$$

$$g_1 = (group,\ epoch,\ leafIndex,\ generation_1)$$

$$(generation_1 > generation_0)$$

g_0, g_1 : key_id

Insider Replay Attack



Insider Replay Attack



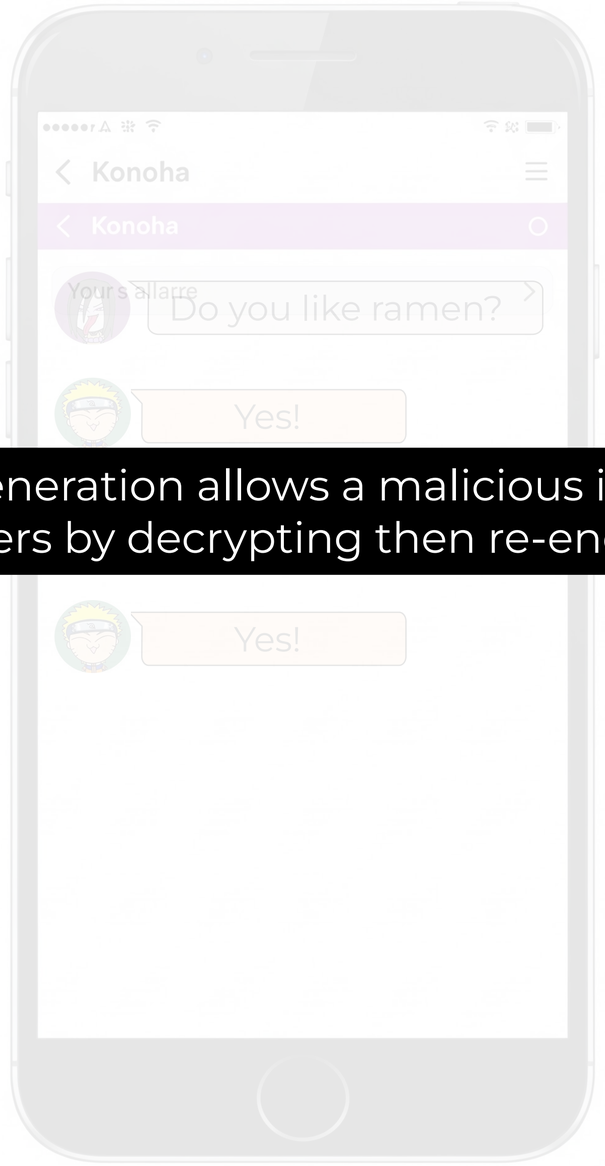


Insider Replay Attack

c' Previously saved ctxt c' c'

Insider Replay Attack

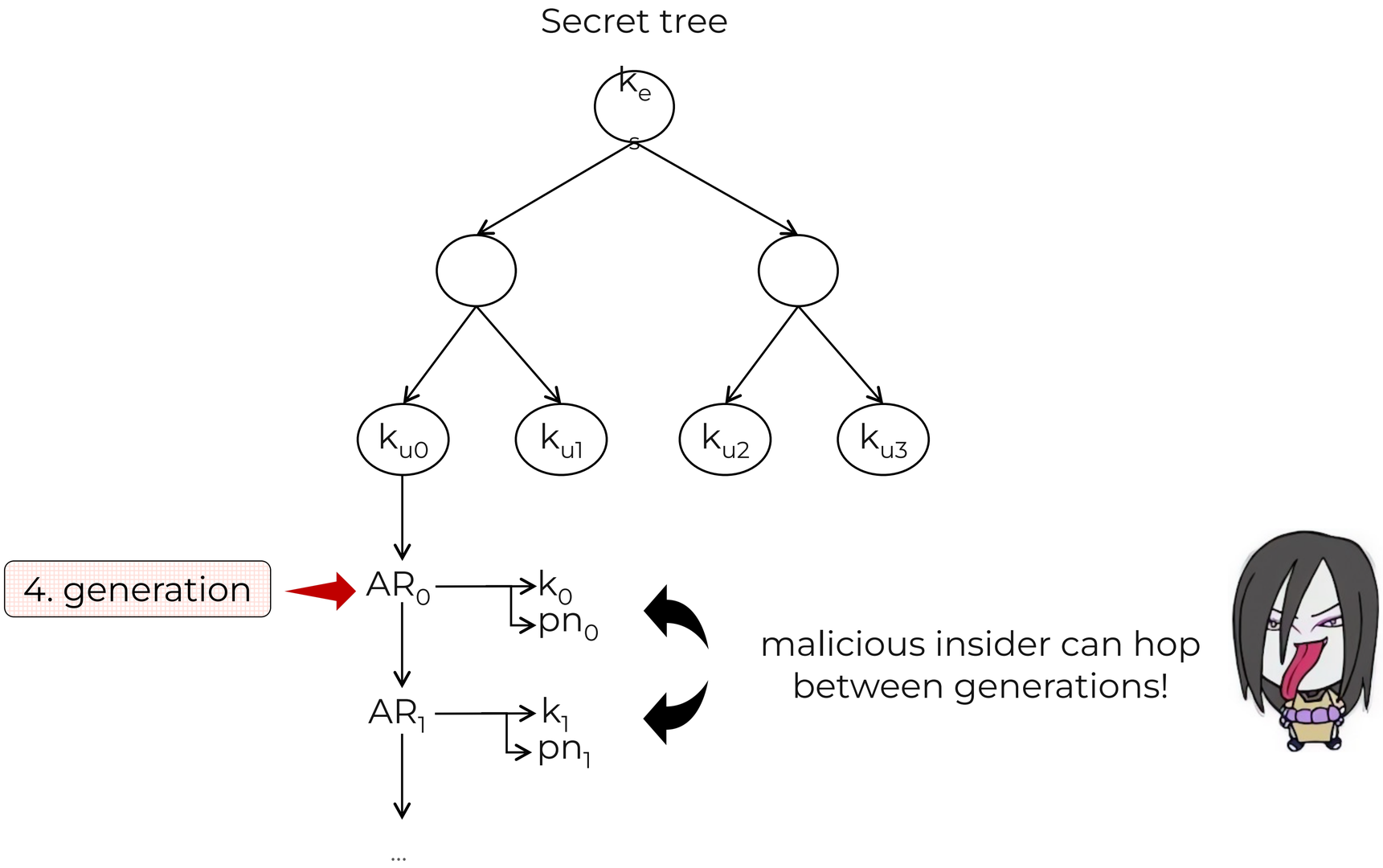
Insider Replay Attack



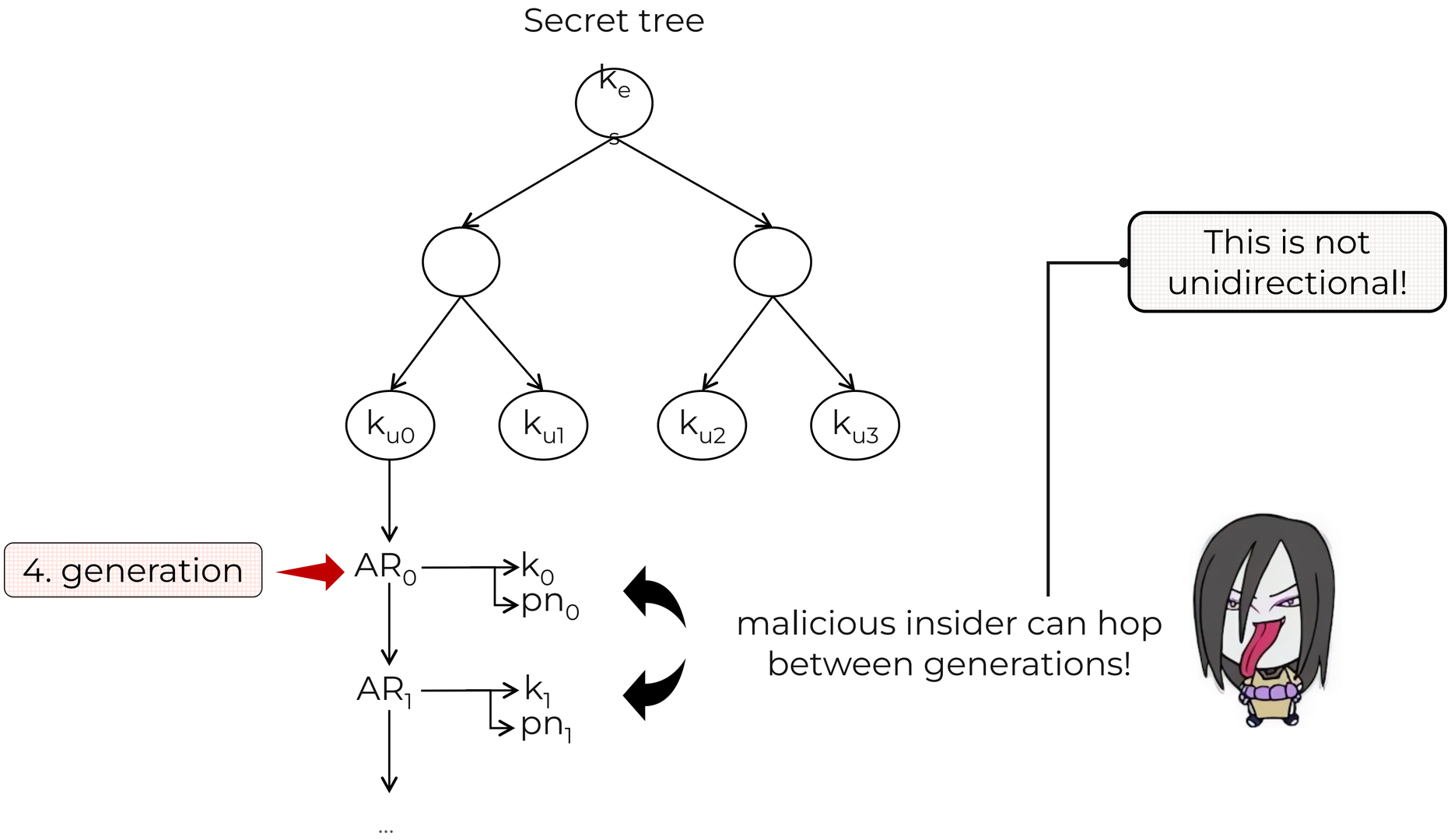
Not authenticating the generation allows a malicious insider to replay messages sent by honest members by decrypting then re-encrypting previous ctxts

2. Insider Re-ordering Attack

Insider Re-ordering Attack



Insider Re-ordering Attack



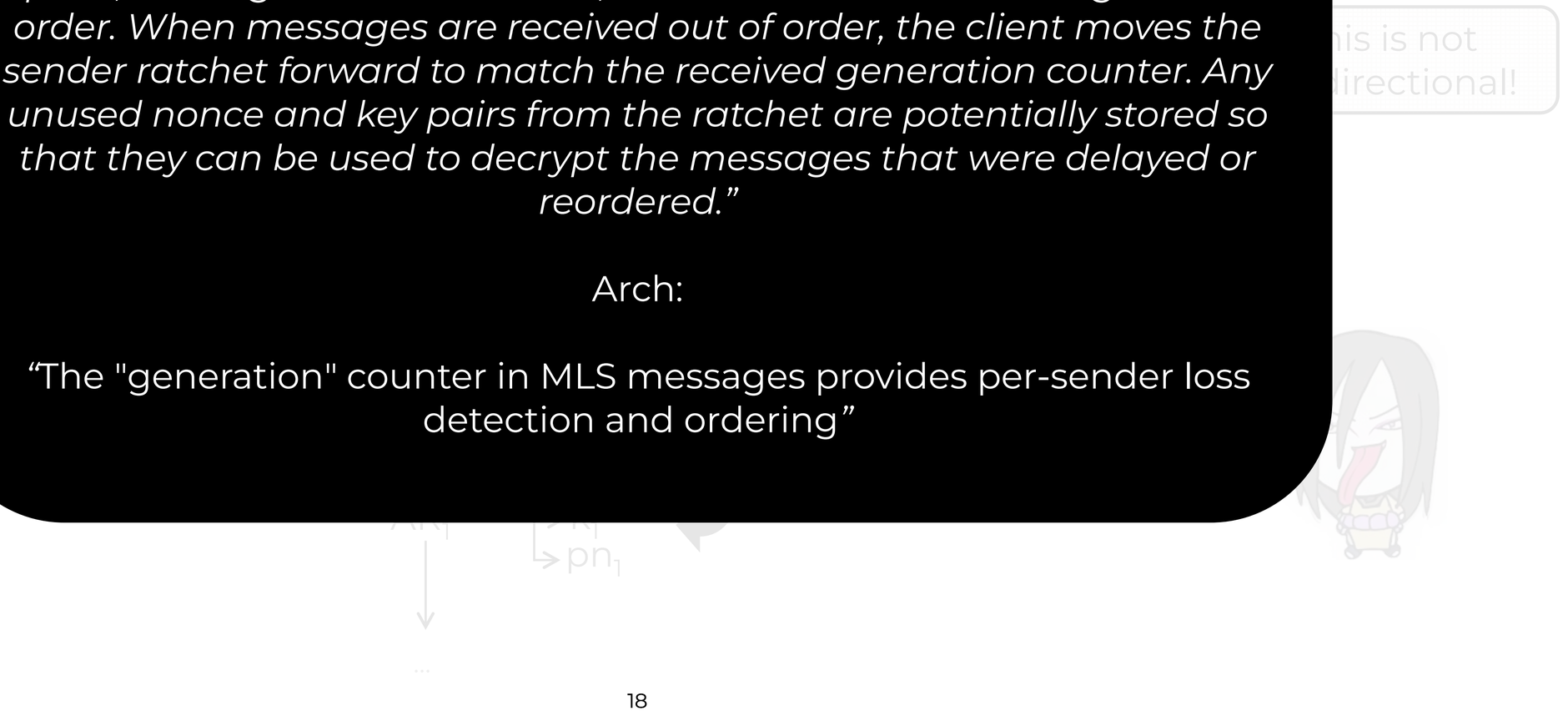
Insider Re-ordering Attack

MLS relies on the *generation* for ordering information:

“Since each application message contains the group identifier, the epoch, and a generation counter, a client can receive messages out of order. When messages are received out of order, the client moves the sender ratchet forward to match the received generation counter. Any unused nonce and key pairs from the ratchet are potentially stored so that they can be used to decrypt the messages that were delayed or reordered.”

Arch:

“The "generation" counter in MLS messages provides per-sender loss detection and ordering”



Insider Re-ordering Attack

Insider Re-ordering Attack






$m_0 \leftarrow$ Not good


$m_1 \leftarrow$ My dog ate an entire birthday cake!


$m_2 \leftarrow$ He's going to be eating that dry dog food for a while now.


$m_3 \leftarrow$ And


$m_4 \leftarrow$ My son forgot to get half the groceries on the list!

Insider Re-ordering Attack





Insider Re-ordering Attack





c_i

MLS-Sign-then-Encrypt

$s_i \leftarrow \text{Sign}(sk_u,\ m_s^i)$

$c_i \leftarrow \text{Enc}(k_{g_i},\ n_i,\ s_i \,||\, m_i,\ ad_e^i)$

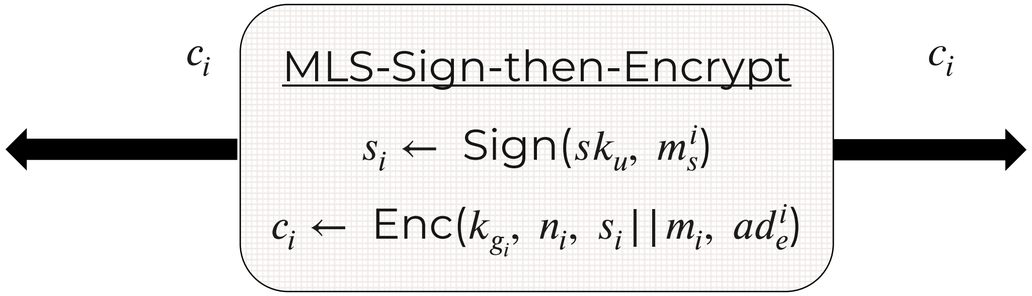
c_i

u : user_id
g_0 : key_id

Insider Re-ordering Attack







$g_i = (group, epoch, leafIndex, i)$

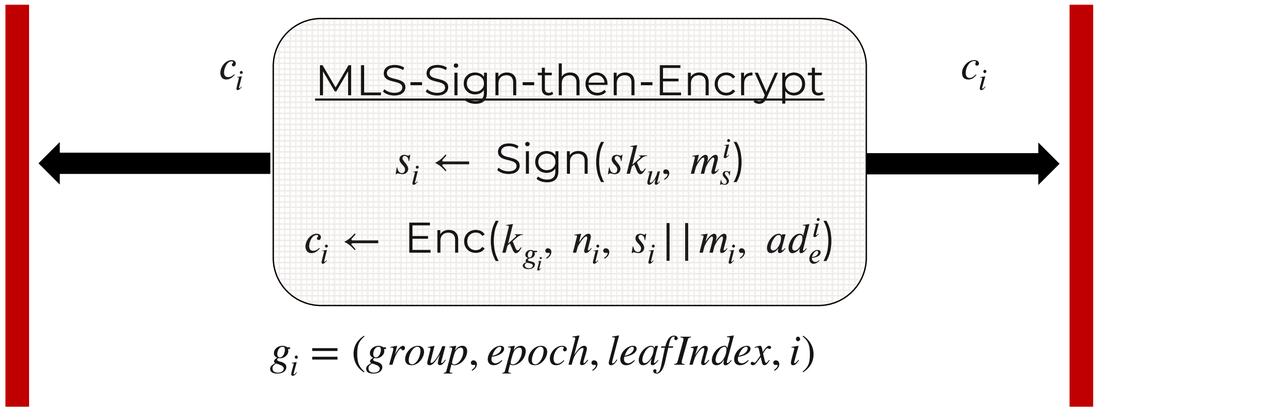
u : user_id
g_0 : key_id

Insider Re-ordering Attack





Block messages!



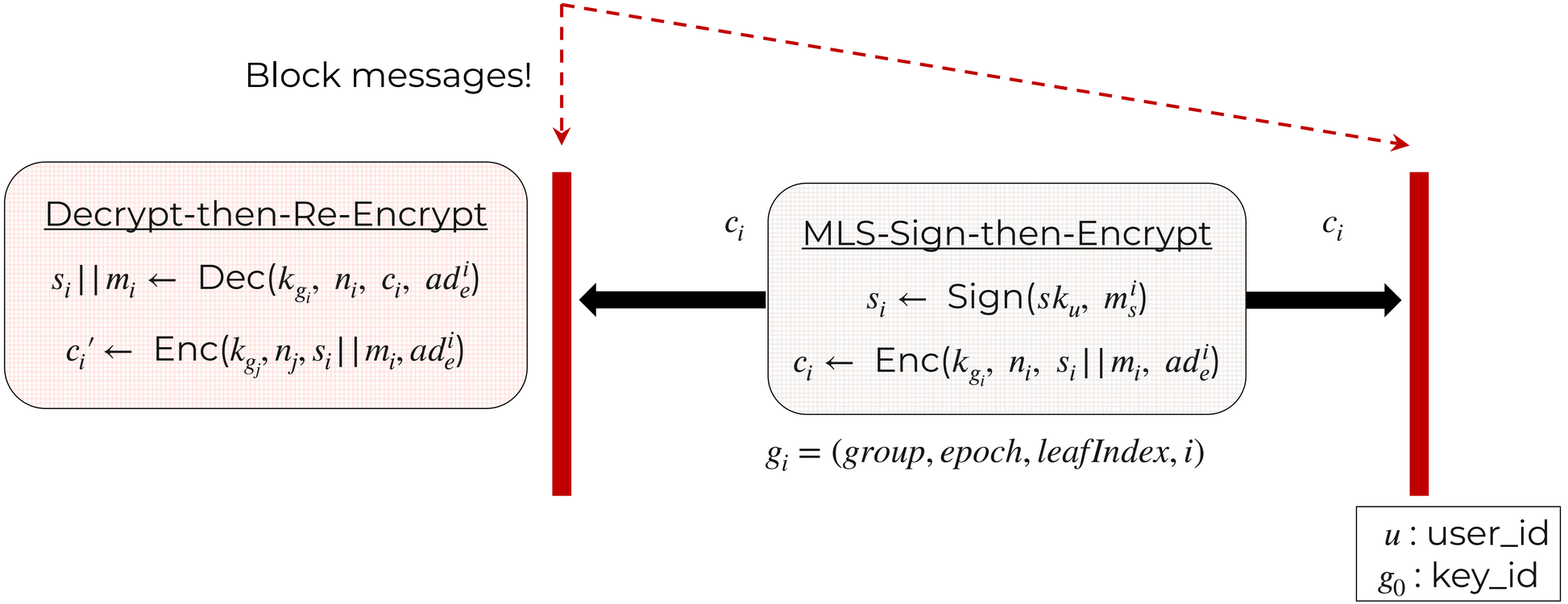
u : user_id
g_0 : key_id

Insider Re-ordering Attack





Block messages!



Insider Re-ordering Attack





Block messages!

Decrypt-then-Re-Encrypt

$s_i || m_i \leftarrow \text{Dec}(k_{g_i}, n_i, c_i, ad_e^i)$

$c_i' \leftarrow \text{Enc}(k_{g_j}, n_j, s_i || m_i, ad_e^i)$

Interchange generations

c_i

MLS-Sign-then-Encrypt

$s_i \leftarrow \text{Sign}(sk_u, m_s^i)$

$c_i \leftarrow \text{Enc}(k_{g_i}, n_i, s_i || m_i, ad_e^i)$

c_i

$g_i = (group, epoch, leafIndex, i)$

u : user_id
g_0 : key_id

Insider Replay Attack





Insider Replay Attack



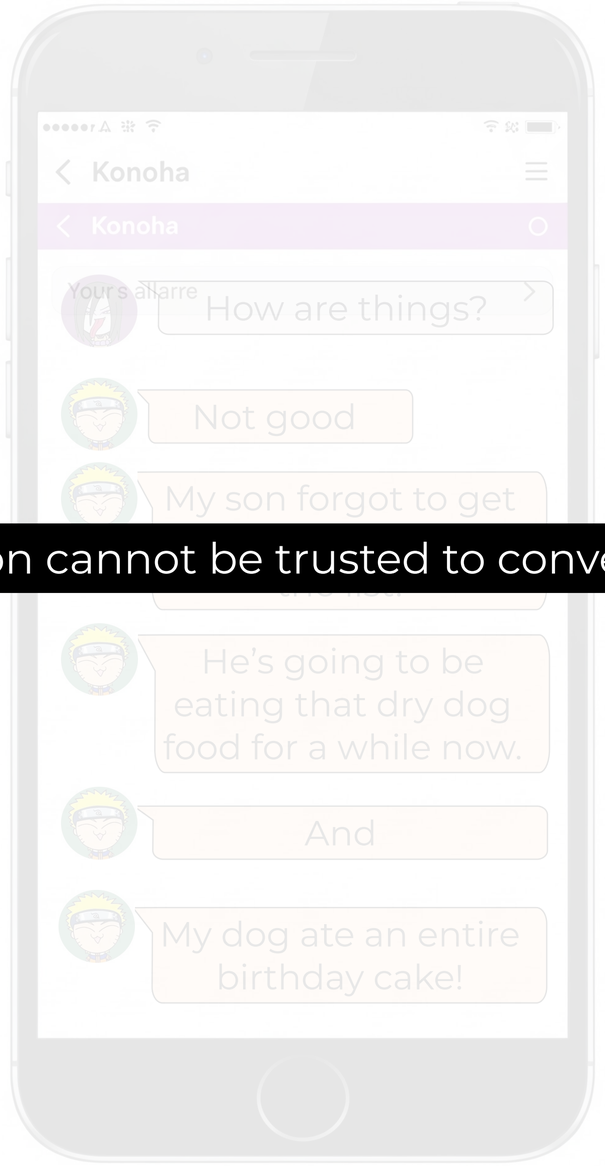


c_i' c_i'

Re-ordered ciphertexts

Insider Re-ordering Attack

Insider Re-ordering Attack

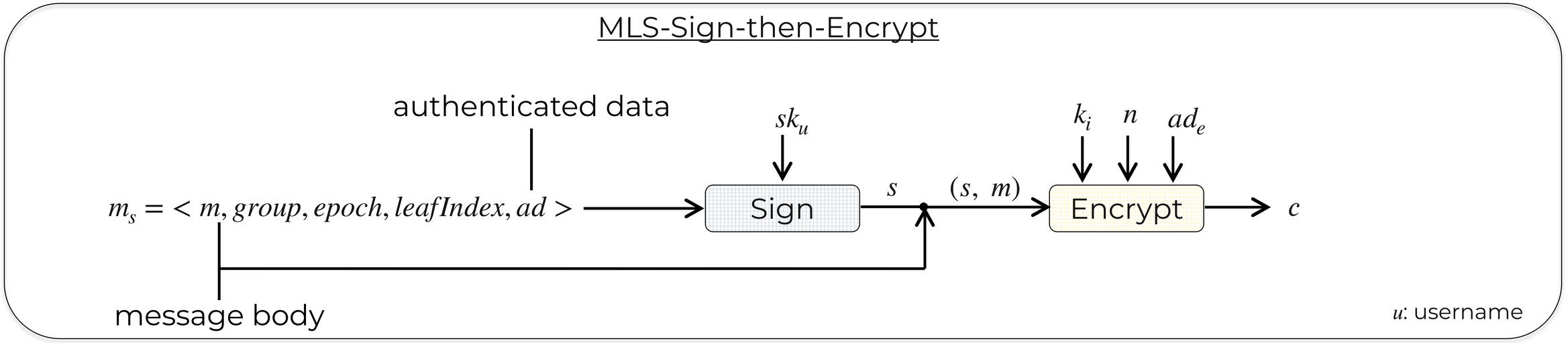


Message generation cannot be trusted to convey order information!

3. Insider INT-CTXT (but not INT-PTXT) Style Attack

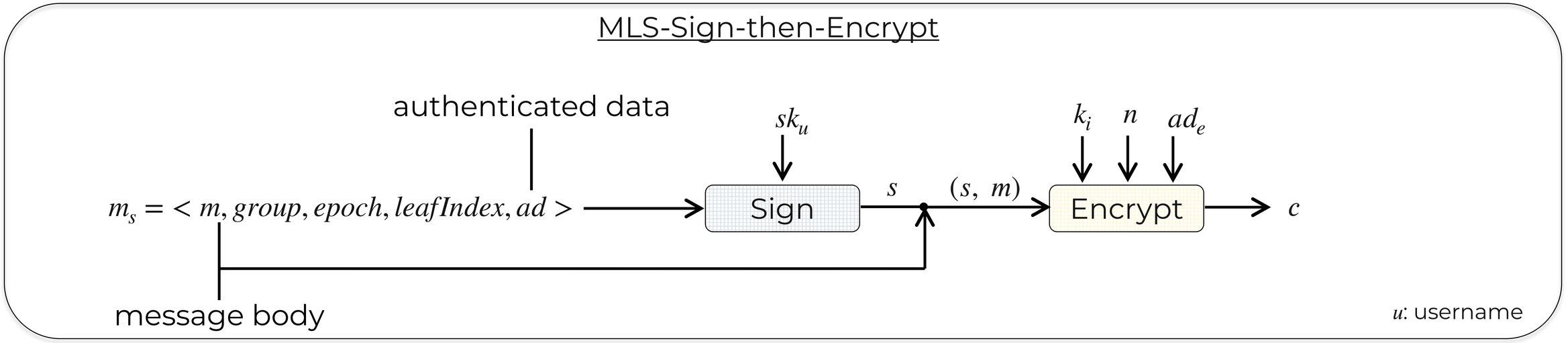
INT-CTXT Style Attack

Recall: Chat encryption in MLS



INT-CTXT Style Attack

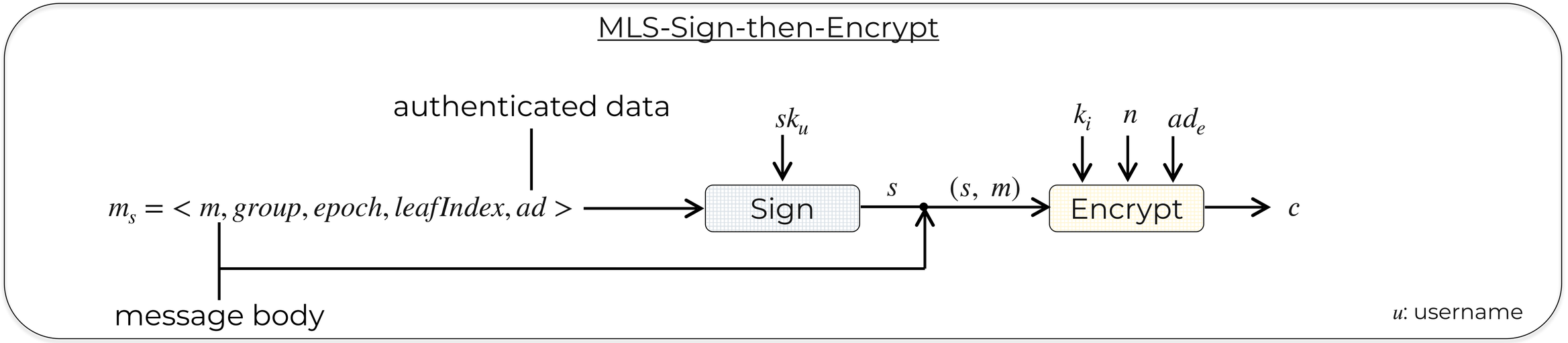
Recall: Chat encryption in MLS



Intuition: s should authenticate the encryption nonce n so that group insiders cannot re-encrypt $(s,\ m)$ using a different nonce and replay message to group

INT-CTXT Style Attack

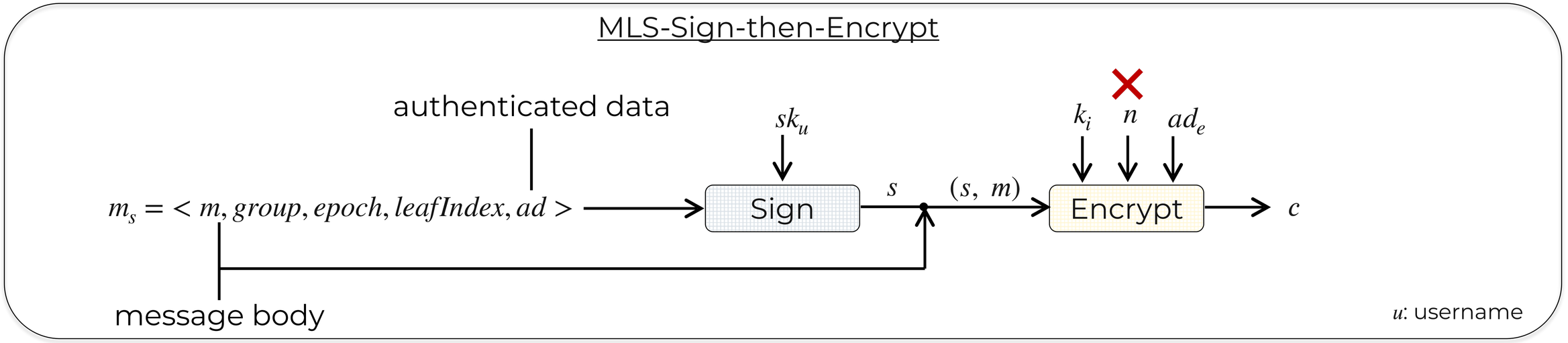
Recall: Chat encryption in MLS



Intuition: s should authenticate the encryption nonce n so that group insiders cannot re-encrypt $(s,\ m)$ using a different nonce and replay message to group

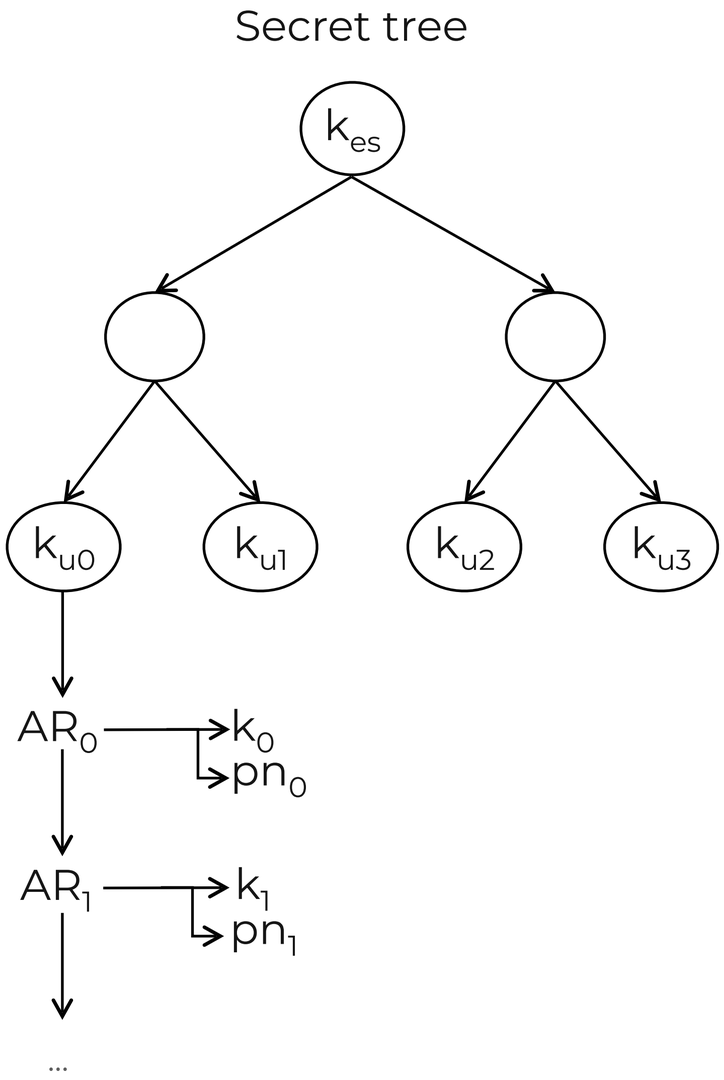
INT-CTXT Style Attack

Recall: Chat encryption in MLS

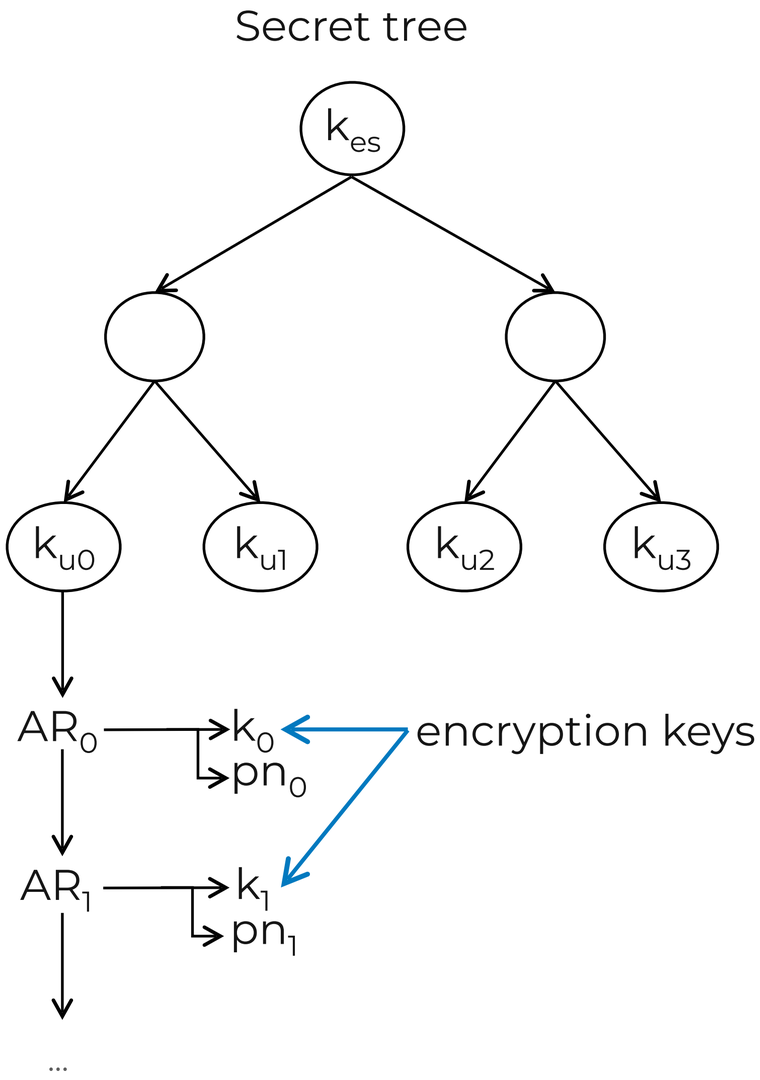


Intuition: s should authenticate the encryption nonce n so that group insiders cannot re-encrypt $(s,\ m)$ using a different nonce and replay message to group

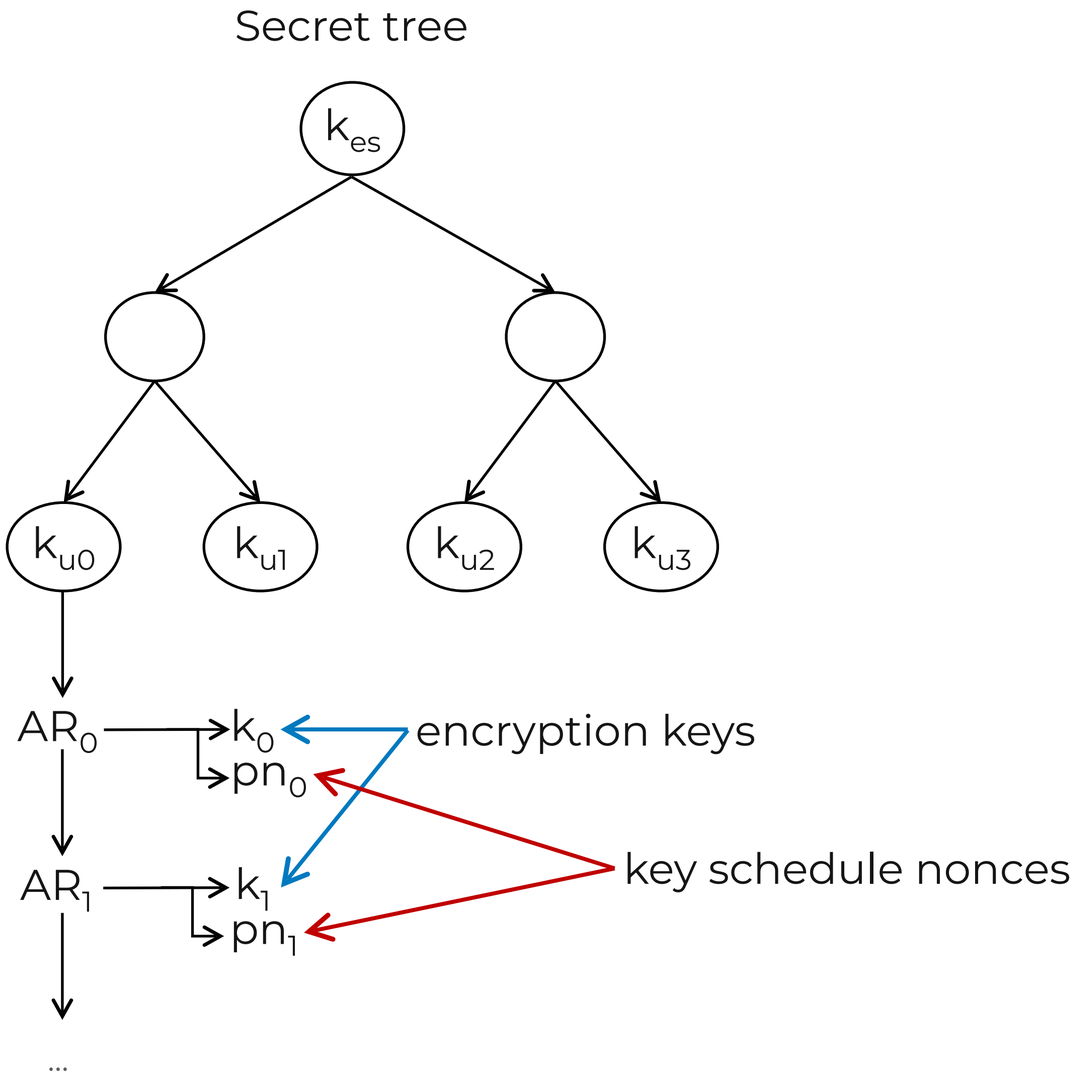
Key Schedule Nonce & Encryption Nonce



Key Schedule Nonce & Encryption Nonce

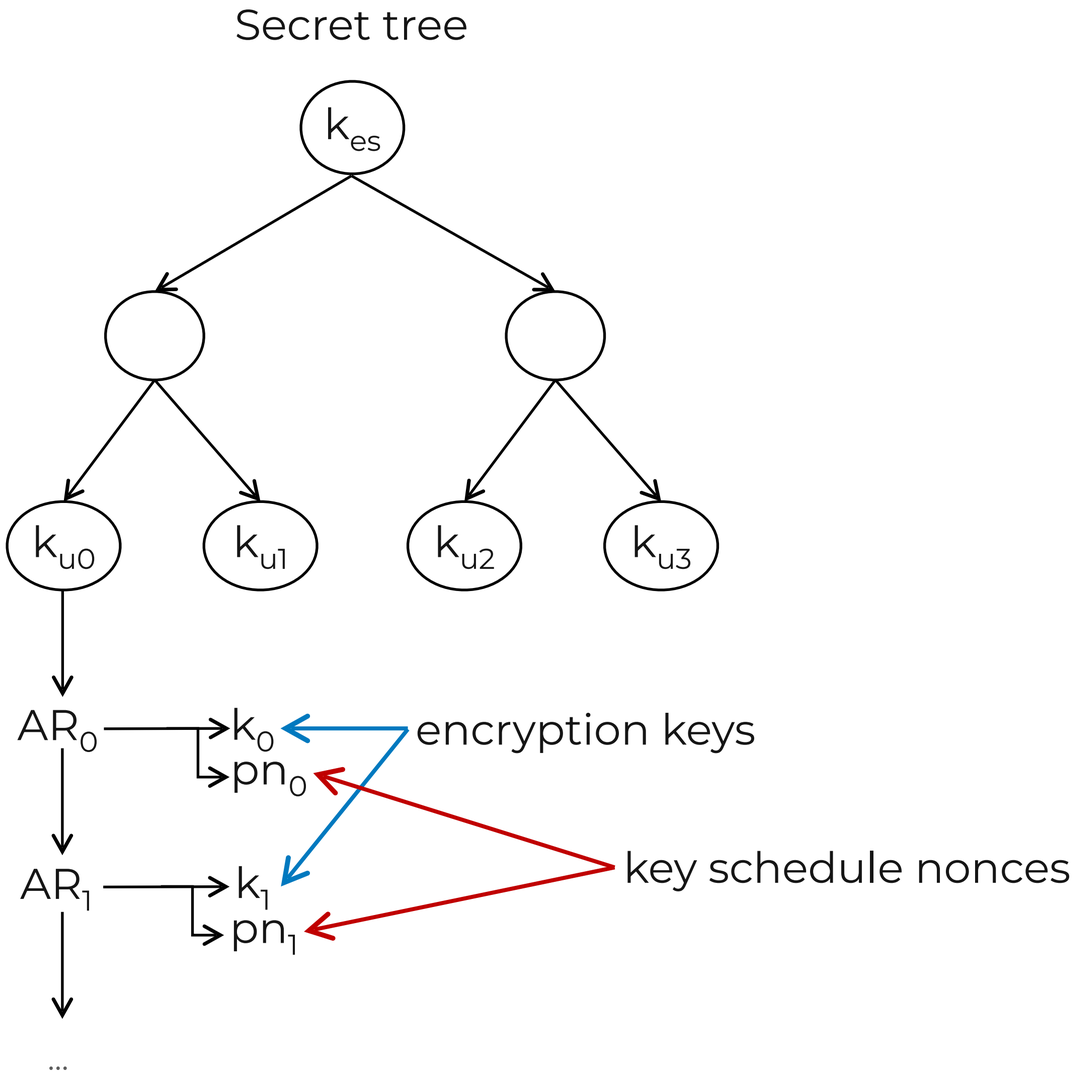


Key Schedule Nonce & Encryption Nonce



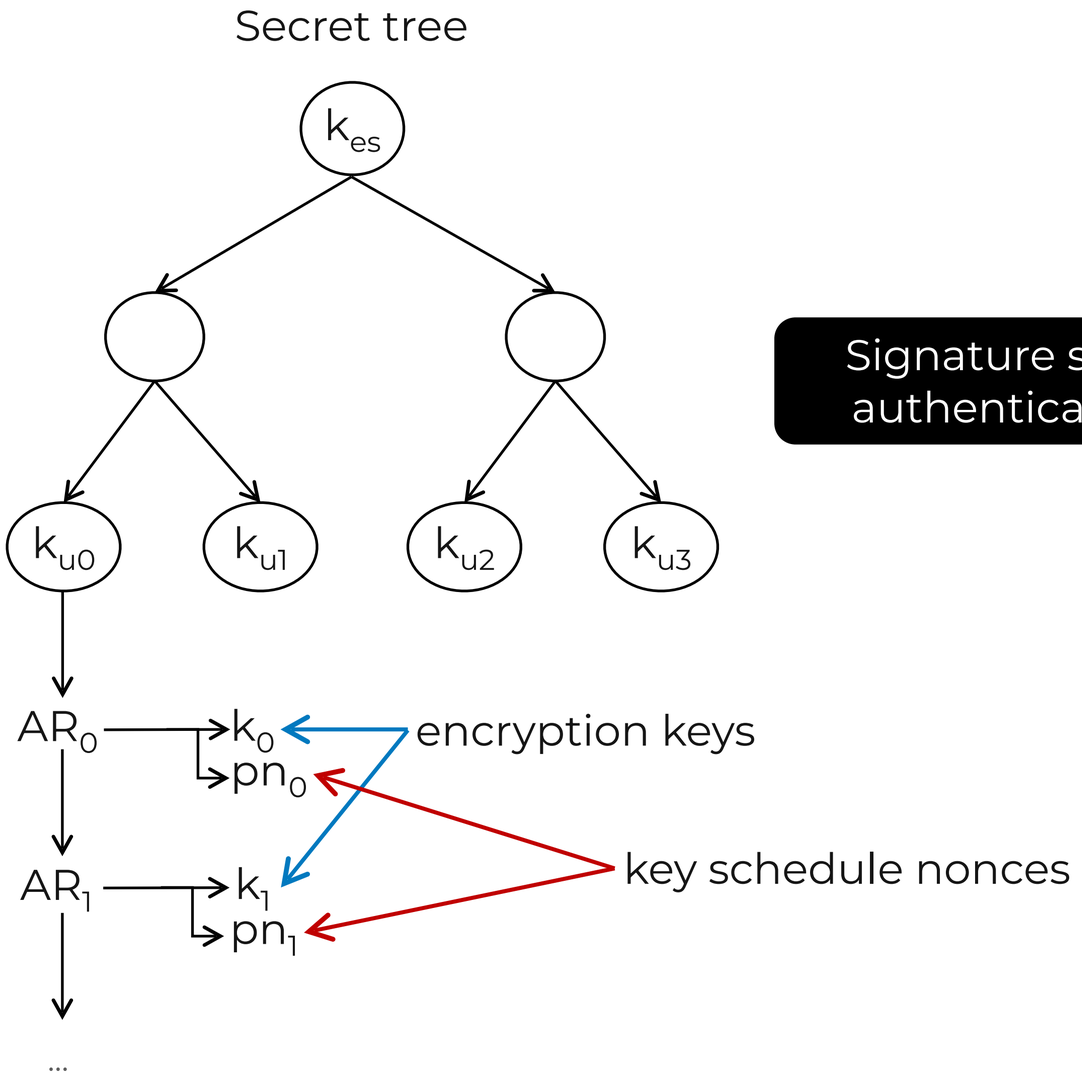
Key Schedule Nonce & Encryption Nonce

Encryption nonce $n = pn \oplus r$
where $r \leftarrow^{\$} \{0,1\}^{32}$



Key Schedule Nonce & Encryption Nonce

Encryption nonce $n = pn \oplus r$ where $r \leftarrow^{\$} \{0,1\}^{32}$



Signature should either authenticate n or $(pn,\ r)$

INT-CTXT Style Attack

INT-CTXT Style Attack





INT-CTXT Style Attack






m ← Hi!

INT-CTXT Style Attack





m ← Hi!

c

MLS-Sign-then-Encrypt

$s \leftarrow \text{Sign}(sk_u,\ m_s)$

$c \leftarrow \text{Enc}(k_g,\ n,\ s||m,\ ad_e)$

c

u : user_id
g : key_id

INT-CTXT Style Attack





m ← Hi!

c

MLS-Sign-then-Encrypt

$s \leftarrow \text{Sign}(sk_u,\ m_s)$

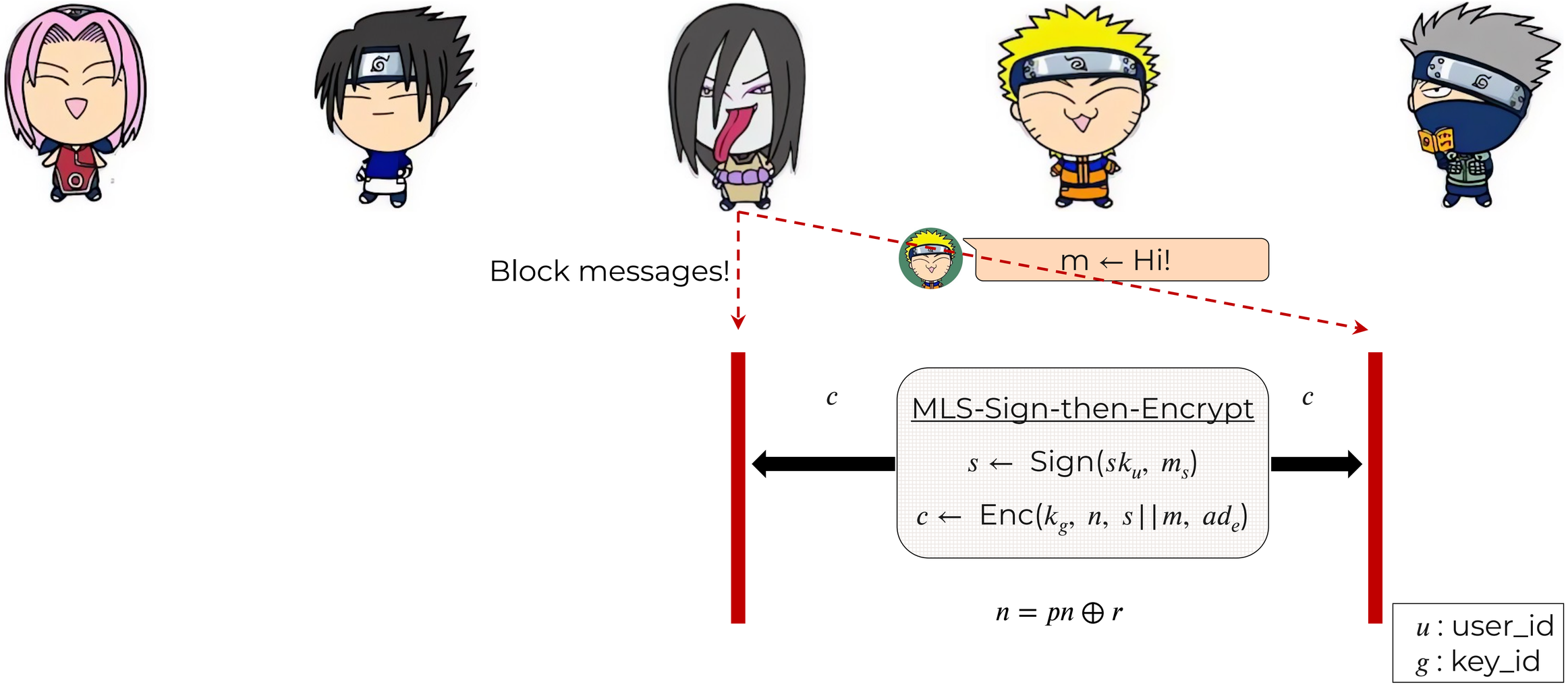
$c \leftarrow \text{Enc}(k_g,\ n,\ s||m,\ ad_e)$

c

$n = pn \oplus r$

u : user_id
g : key_id

INT-CTXT Style Attack



INT-CTXT Style Attack





g : key_id

INT-CTXT Style Attack





Decrypt-then-Re-Encrypt

$s||m \leftarrow \mathsf{Dec}(k_g,\ n,\ c,\ ad_e)$

$c' \leftarrow \mathsf{Enc}(k_g,\ n',\ s||m,\ ad_e)$

g : key_id

INT-CTXT Style Attack





Decrypt-then-Re-Encrypt

$s||m \leftarrow \text{Dec}(k_g,\ n,\ c,\ ad_e)$

$c' \leftarrow \text{Enc}(k_g,\ n',\ s||m,\ ad_e)$

$n = pn \oplus r$

$n' = pn \oplus r'$

g : key_id

INT-CTXT Style Attack





c'

Decrypt-then-Re-Encrypt

$s||m \leftarrow \text{Dec}(k_g,\ n,\ c,\ ad_e)$

$c' \leftarrow \text{Enc}(k_g,\ n',\ s||m,\ ad_e)$

c'

$n = pn \oplus r$

$n' = pn \oplus r'$

g : key_id

INT-CTXT Style Attack





Undetected by group members!

c'

c'

Decrypt-then-Re-Encrypt

$s||m \leftarrow \text{Dec}(k_g,\ n,\ c,\ ad_e)$

$c' \leftarrow \text{Enc}(k_g,\ n',\ s||m,\ ad_e)$

$n = pn \oplus r$

$n' = pn \oplus r'$

g : key_id

INT-CTXT Style Attack

Undetected by group members!

Value of this attack is quite uncertain since attacker can compromise ciphertext integrity but not the integrity of the underlying plaintext

$c' \leftarrow \text{Enc}(k_g,\ n',\ s||m,\ ad_e)$

$n = pn \oplus r$

$n' = pn \oplus r'$

g : key_id

Mitigations

∀ attack ∈ {insider replay, insider reordering, INT-CTXT style} there are two kinds of fixes

Mitigations

∀ attack ∈ {insider replay, insider reordering, INT-CTXT style} there are two kinds of fixes

1. MLS-level fix

Mitigations

∀ attack ∈ {insider replay, insider reordering, INT-CTXT style} there are two kinds of fixes

1. MLS-level fix

2. Application-level fix

Mitigations

∀ attack ∈ {insider replay, insider reordering, INT-CTXT style} there are two kinds of fixes

1. MLS-level fix
2. Application-level fix

	Insider Replay	**Insider Re-ordering**	**INT-CTXT Style**
1. MLS-level fix	Sign generation (include in ad)	Sign generation (include in ad)	Sign (pn and r) and or n
2. Application-level fix	Use increasing ctr, include that as part of message (which will be signed)	Use increasing ctr, include that as part of message (which will be signed)	Application shouldn't use ctxt bits for anything meaningful

Mitigations

∀ attack ∈ {insider replay, insider reordering, INT-CTXT style} there are two kinds of fixes

1. MLS-level fix
2. Application-level fix

	Insider Replay	**Insider Re-ordering**	**INT-CTXT Style**
1. MLS-level fix	Sign generation (include in ad)	Sign generation (include in ad)	Sign (pn and r) and or n
2. Application-level fix	Use increasing ctr, include that as part of message (which will be signed)	Use increasing ctr, include that as part of message (which will be signed)	Application shouldn't use ctxt bits for anything meaningful

Mitigations

∀ attack ∈ {insider replay, insider reordering, INT-CTXT style} there are two kinds of fixes

1. MLS-level fix
2. Application-level fix

	Insider Replay	Insider Re-ordering	INT-CTXT Style
1. MLS-level fix	Sign generation (include in ad)	Sign generation (include in ad)	Sign (pn and r) and or n
2. Application-level fix	Use increasing ctr, include that as part of message (which will be signed)	Use increasing ctr, include that as part of message (which will be signed)	Application shouldn't use ctxt bits for anything meaningful

Mitigations

∀ attack ∈ {insider replay, insider reordering, INT-CTXT style} there are two kinds of fixes

1. MLS-level fix
2. Application-level fix

	Insider Replay	**Insider Re-ordering**	**INT-CTXT Style**
1. MLS-level fix	Sign generation (include in ad)	Sign generation (include in ad)	Sign (pn and r) and or n
2. Application-level fix	Use increasing ctr, include that as part of message (which will be signed)	Use increasing ctr, include that as part of message (which will be signed)	Application shouldn't use ctxt bits for anything meaningful

Mitigations

∀ attack ∈ {insider replay, insider reordering, INT-CTXT style} there are two kinds of fixes

1. MLS-level fix
2. Application-level fix

	Insider Replay	Insider Re-ordering	INT-CTXT Style
1. MLS-level fix	Sign generation (include in ad)	Sign generation (include in ad)	Sign (pn and r) and or n
2. Application-level fix	Use increasing ctr, include that as part of message (which will be signed)	Use increasing ctr, include that as part of message (which will be signed)	Application shouldn't use ctxt bits for anything meaningful

Mitigations

∀ attack ∈ {insider replay, insider reordering, INT-CTXT style} there are two kinds of fixes

1. MLS-level fix
2. Application-level fix

	Insider Replay	Insider Re-ordering	INT-CTXT Style
1. MLS-level fix	Sign generation (include in ad)	Sign generation (include in ad)	Sign (pn and r) and or n
2. Application-level fix	Use increasing ctr, include that as part of message (which will be signed)	Use increasing ctr, include that as part of message (which will be signed)	Application shouldn't use ctxt bits for anything meaningful

Assuming app performs appropriate checks

Mitigations

∀ attack ∈ {insider replay, insider reordering, INT-CTXT style} there are two kinds of fixes

1. MLS-level fix
2. Application-level fix

	Insider Replay	Insider Re-ordering	INT-CTXT Style
1. MLS-level fix	Sign generation (include in ad)	Sign generation (include in ad)	Sign (pn and r) and or n
2. Application-level fix	Use increasing ctr, include that as part of message (which will be signed)	Use increasing ctr, include that as part of message (which will be signed)	Application shouldn't use ctxt bits for anything meaningful

Assuming app performs appropriate checks

Mitigations

∀ attack ∈ {insider replay, insider reordering, INT-CTXT style} there are two kinds of fixes

1. MLS-level fix
2. Application-level fix

	Insider Replay	Insider Re-ordering	INT-CTXT Style
1. MLS-level fix	Sign generation (include in ad)	Sign generation (include in ad)	Sign (pn and r) and or n
2. Application-level fix	Use increasing ctr, include that as part of message (which will be signed)	Use increasing ctr, include that as part of message (which will be signed)	Application shouldn't use ctxt bits for anything meaningful

Assuming app performs appropriate checks

Mitigations

∀ attack ∈ {insider replay, insider reordering, INT-CTXT style} there are two kinds of fixes

1. MLS-level fix
2. Application-level fix

	Insider Replay	Insider Re-ordering	INT-CTXT Style
1. MLS-level fix	Sign generation (include in ad)	Sign generation (include in ad)	Sign (pn and r) and or n
2. Application-level fix	Use increasing ctr, include that as part of message (which will be signed)	Use increasing ctr, include that as part of message (which will be signed)	Application shouldn't use ctxt bits for anything meaningful

Assuming app performs appropriate checks