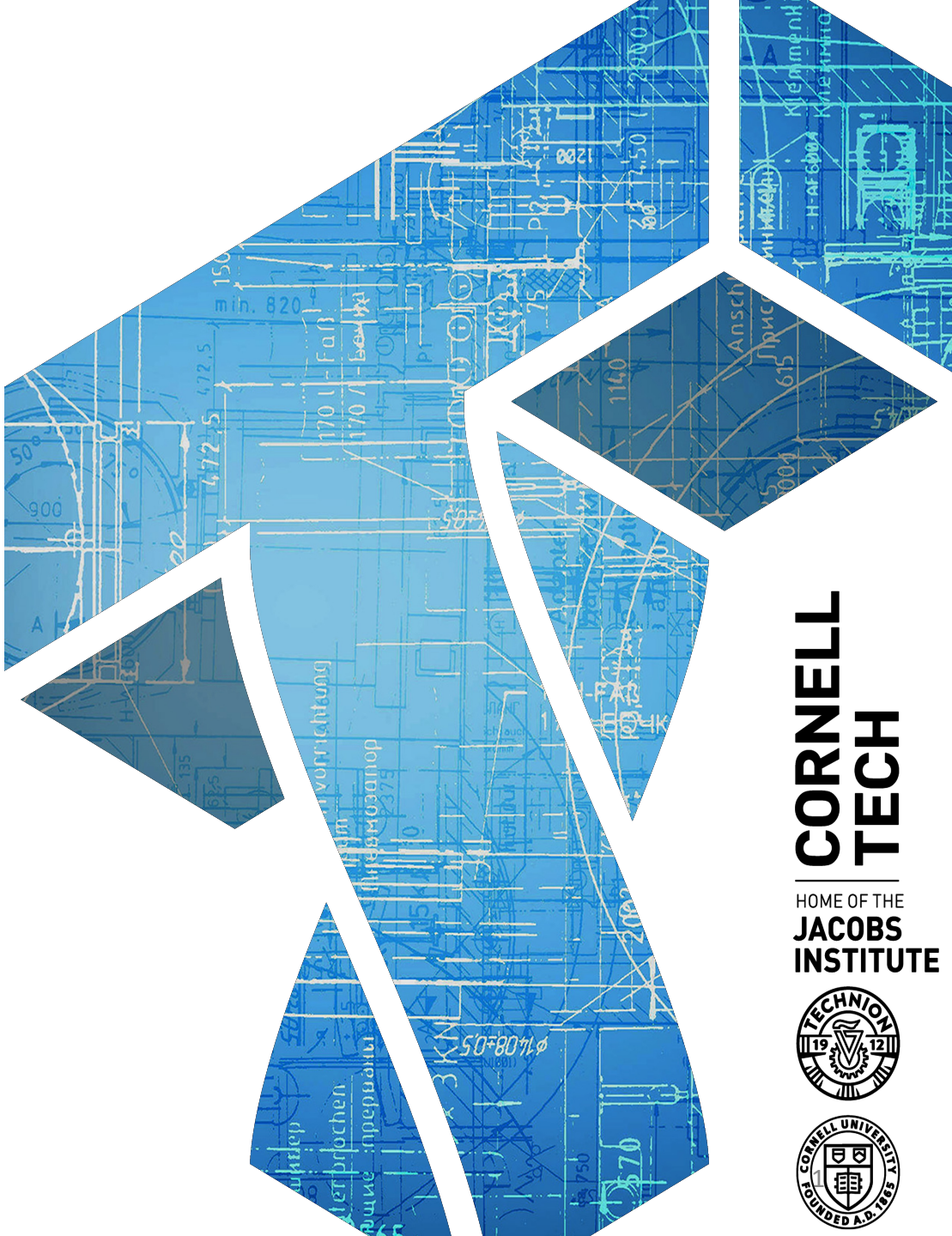
CS 5830
Cryptography



CORNELL TECH

HOME OF THE JACOBS INSTITUTE

End-to-end (E2E) Encrypted messaging

- TextSecure by Open Whisper Systems (2013)
  - Trevor Perrin and Moxie Marlinspike (Levchin award winners)
  - Later became Signal
  - Developed Signal Protocol, which was adopted elsewhere (most notably: WhatsApp)
- Protocol provides:
  - "… confidentiality, integrity, authentication, participant consistency, destination validation, forward secrecy, post-compromise security (aka future secrecy), causality preservation, message unlinkability, message repudiation, participation repudiation, and asynchronicity."

Signal Protocol (X3DH)

https://signal.org/docs/specifications/x3dh/

Identity key

Signed pre-key

Signature of SPK_B using secret key associated to IK_B

One-time pre-keys

Service provider

IK_B, SPK_B, σ , $OPK1_B$, … , $OPKn_B$

IK_B, SPK_B, $OPKi_B$ all public DH keys, recipient retains secret keys

IK_B, SPK_B, σ , $OPKi_B$

Verify σ

DH(PK1,PK2) is means output $g^{sk1*sk2}$

$r \leftarrow \$ \mathbf{Z}_p$; $EK_A \leftarrow g^r$

$DH1 = DH(IK_A, SPK_B)$

$DH2 = DH(EK_A, IK_B)$

$DH3 = DH(EK_A, SPK_B)$

~~$DH4 = DH(EK_A, OPK_B)$~~

Mutual authentication

Forward secrecy

$SK = KDF(DH1 \,||\, DH2 \,||\, DH3 \,||\, \sout{DH4})$

$C \leftarrow AEAD(SK, message)$

IK_A , EK_A, key identifiers, C

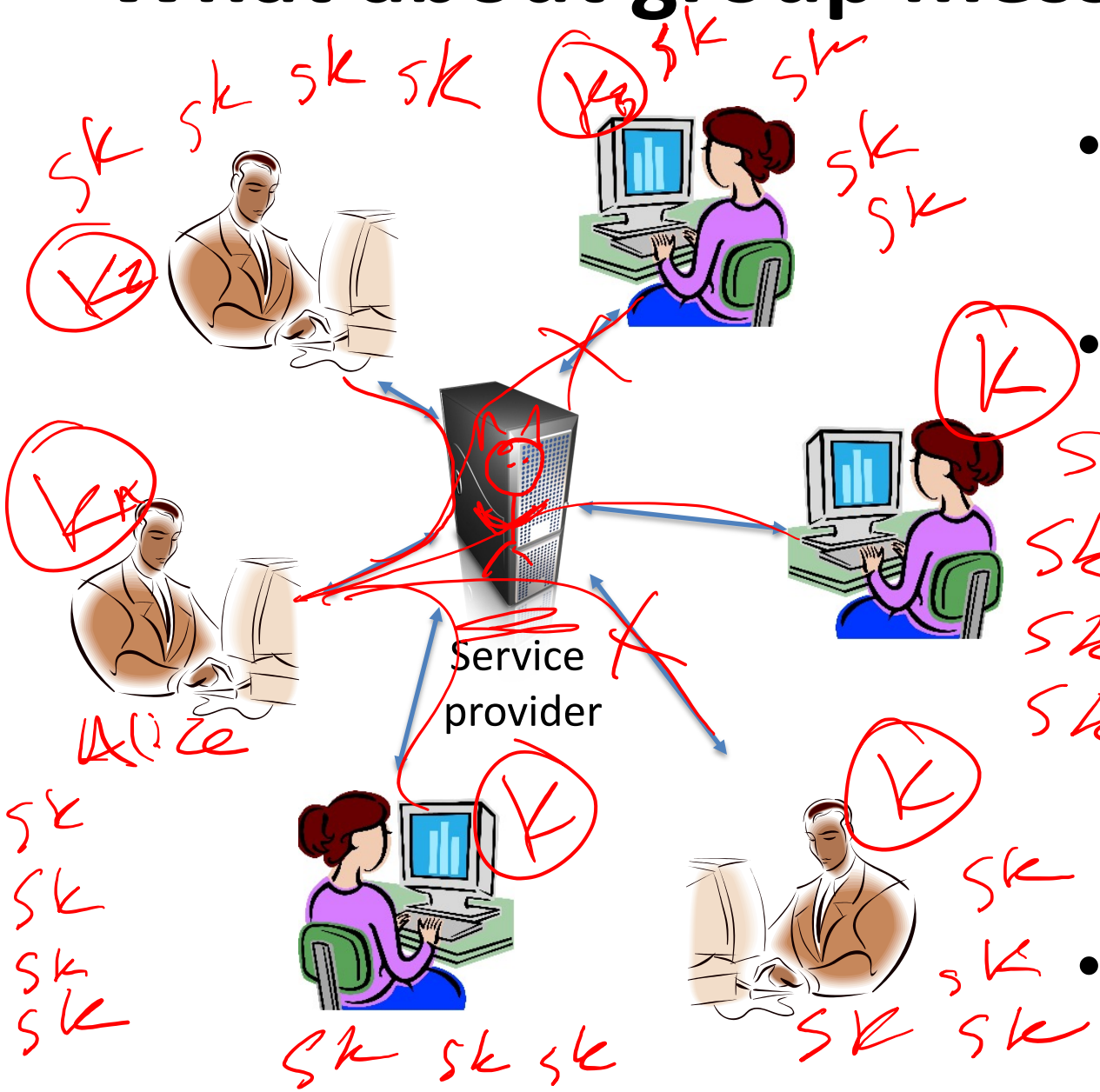
IK_A , EK_A, key identifiers, C

Signal Double Ratchet

https://signal.org/docs/specifications/doubleratchet/

- Use X3DH to establish shared secret SK
- To improve forward secrecy, shouldn’t use SK forever
- Ratcheting refers to deriving new key material to allow deleting old key material
- Asymmetric ratchet:
  - New ephemeral DH values sent along with messages
- Symmetric ratchet:
  - Use KDFs to generate new symmetric keys
- Key complexity: asynchronous communications

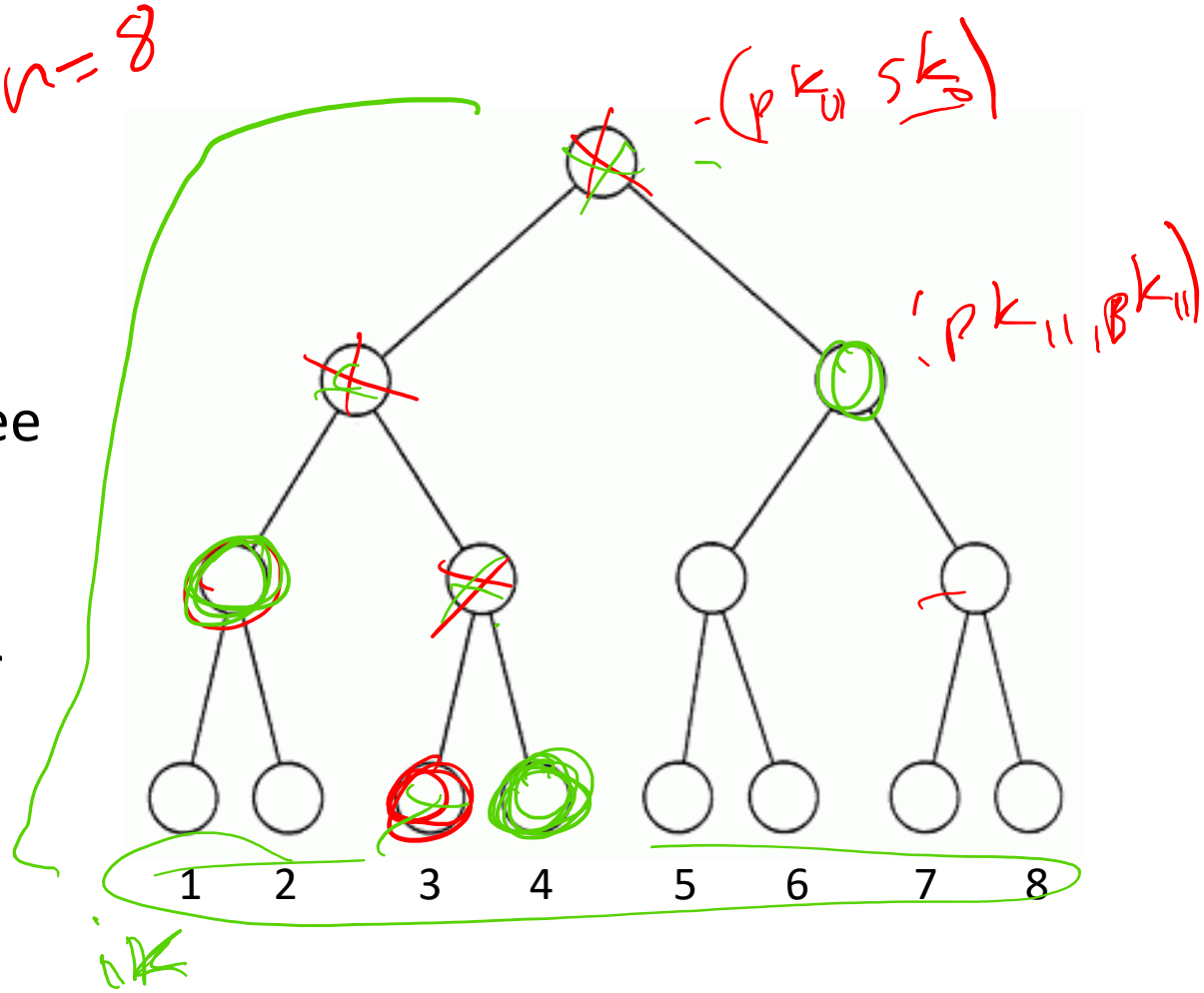
What about group messaging?



- Pairwise broadcast
  - Better: n KEMs and 1 DEM
- Sender keys
  - Each user generates symmetric sender key, sends it to all parties
  - Use sender key when sending
  - Bad post-compromise security: need to do O(n) work to change group membership
- Can we do better?

KEM trees

- Binary tree where each node has (pk,sk) associated to it for KEM
  - Leaves correspond to group members
- ***Invariant***: The private key for a node in the tree is known to a member of the group only if the node's subtree contains that member's leaf.
- ***Updating your own key***: Choose new keys for your direct path. Send update message that encrypts node secrets to co-path
  - log(n) ciphertexts
- ***Removing member***: "Blank out" (invalidate) keys along removed member's direct path. Can encrypt to remaining group using at most log(n) encryptions



Direct path: leaf ancestors

Co-path: children of ancestors not on direct path

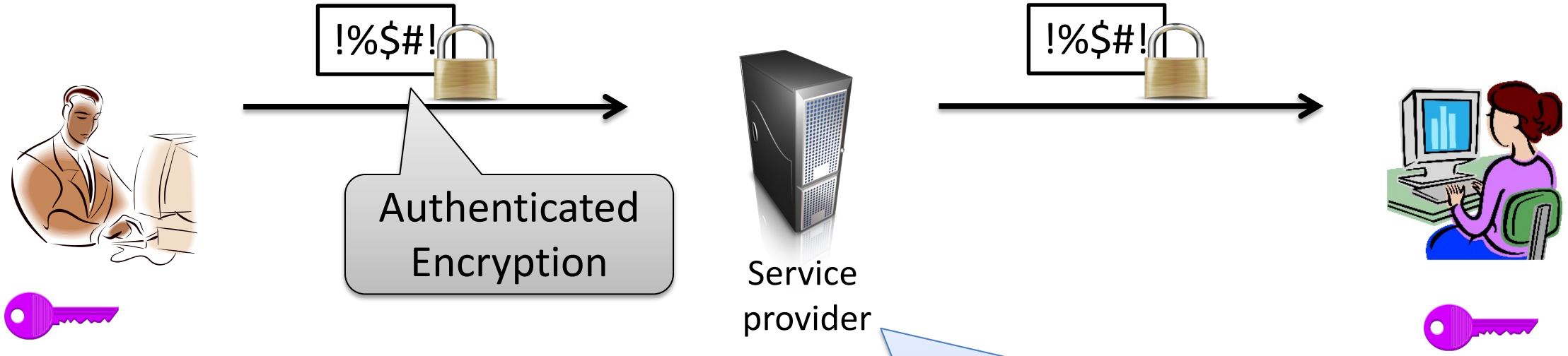
Message Layer Security (MLS)

- New IETF standard (https://datatracker.ietf.org/wg/mls/about/)
- Unified protocol for >=2 client E2E encrypted messaging
- Uses KEM trees, symmetric ratcheting, 2-stage commit protocol for consistency of group state, etc.
- OpenMLS library implements it (https://openmls.tech/)

Encrypted messaging big topic

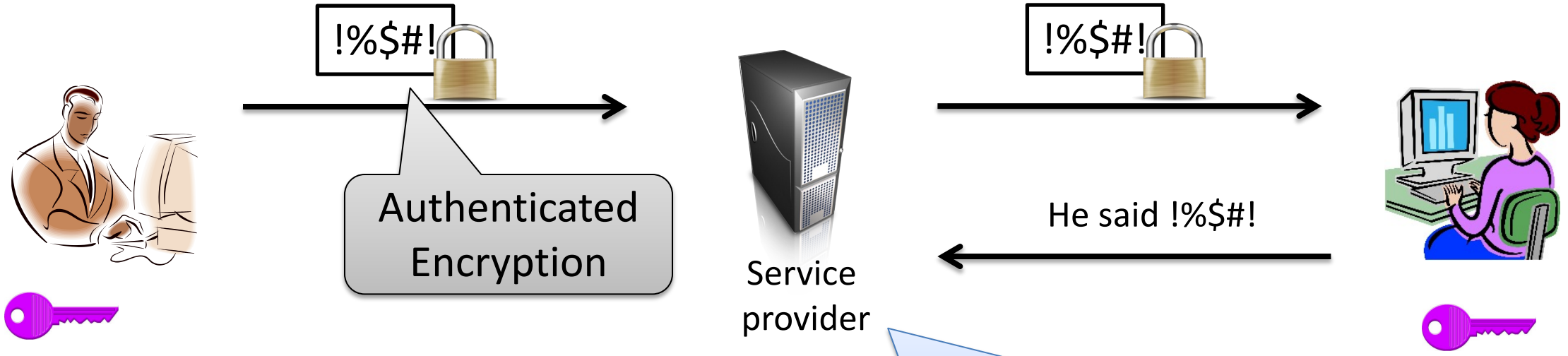
- Google, Facebook (Messenger & WhatsApp), Apple, Signal, Keybase, Skype, ... have implemented some E2E encryption
- Adds complexity to trust and safety features (e.g., abuse reporting)

End-to-end encrypted messaging and abuse



End-to-end security:
Provider cannot read or modify messages

End-to-end encrypted messaging and abuse

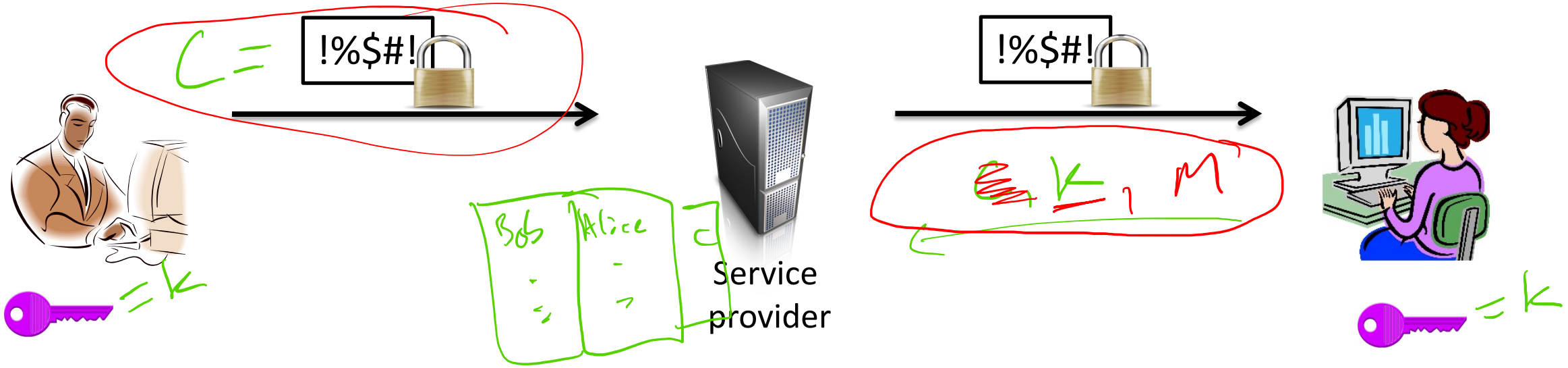


End-to-end security means cannot verify “ !%$#! ” was message sent

[Facebook 2016]:

- Provide cryptographic proof of message contents when reporting abuse
- Called technique ***message franking***

Shouldn't AEAD suffice?

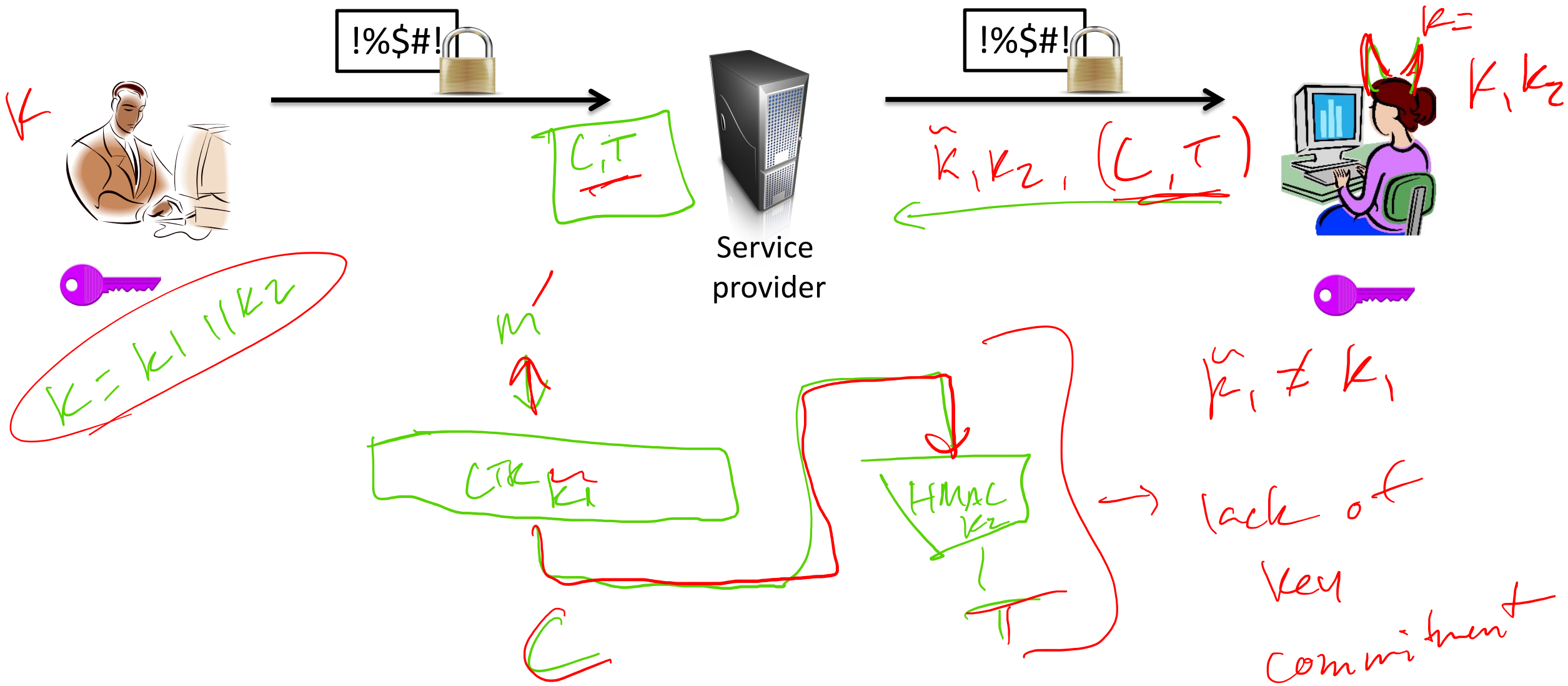


m ← Dec(k, c); IF m = ⊥ then do nothing

If m is bad then do something

Otherwise nothing

Shouldn't AEAD suffice?



Exploration of committing AEAD schemes

Key committing AEAD:

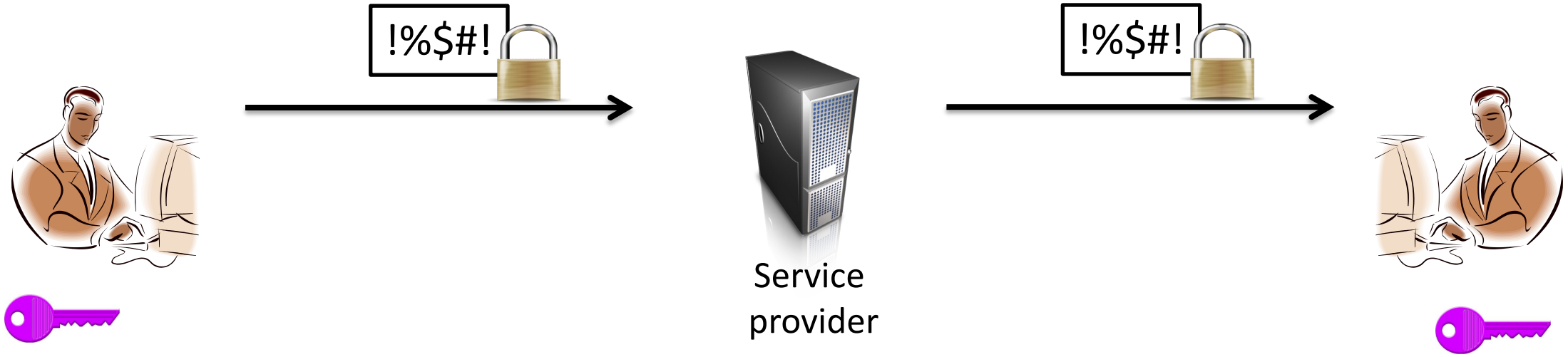
Computationally limited adversary can't find K1, K2 and ciphertext C such that Dec(K1,C) != ⊥ and Dec(K2,C) != ⊥

Are standard AE schemes committing? (treating auth tag as binding tag)

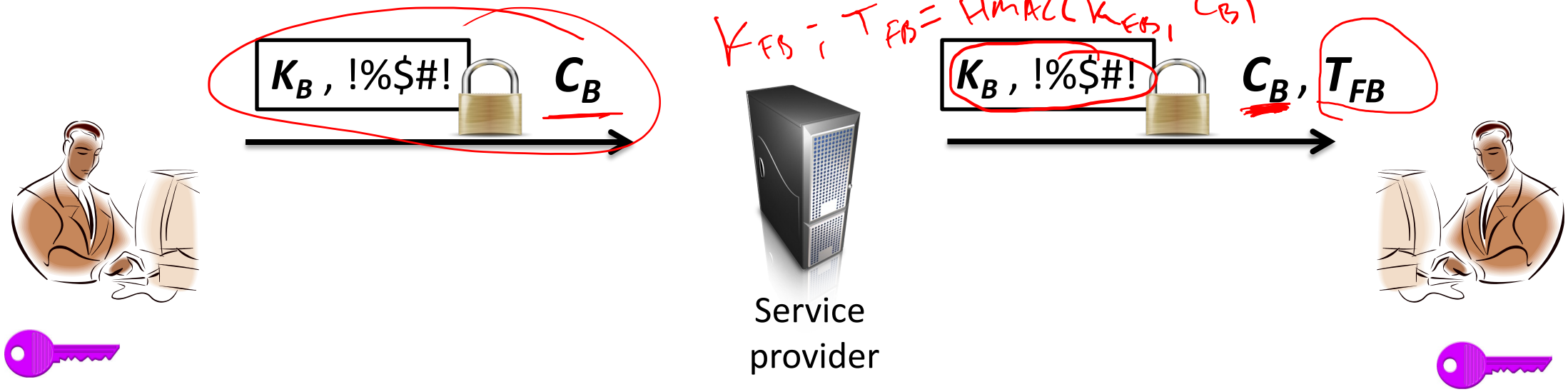
Two new schemes purpose-built for ccAE applications

Scheme	Receiver binding?	# of cryptographic passes
AES-GCM	No	1
OCB	No	1
Encrypt-then-HMAC (distinct keys)	No	2
Encrypt-then-HMAC (one key)	Yes	2
Facebook HMAC-Encrypt-HMAC	Yes	3
Duplex	Yes	1
Committing Encrypt-and-PRF	**Yes**	**2**
Hash Function Chaining	**Yes**	**1**

Facebook's message franking protocol



Facebook's message franking protocol



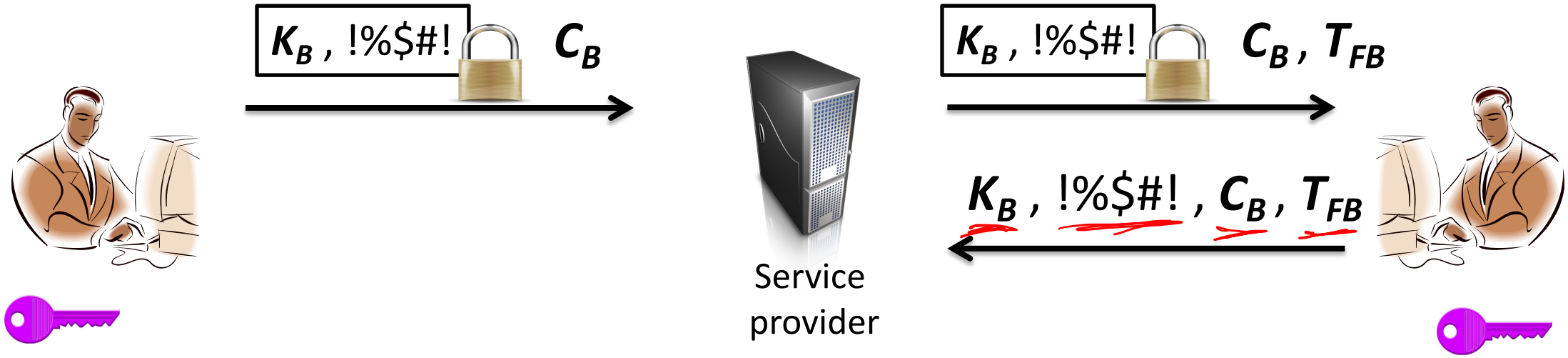
Sender ***cryptographically commits*** to message: $C_B = HMAC(K_B, M)$

Encrypt-then-HMAC (AE) message along with K_B (called the opening)

Provider signs C_B (& sender/receiver IDs) using HMAC to generate tag T_{FB}

Receiver decrypts, retrieves K_B, and verifies C_B

Facebook's message franking protocol



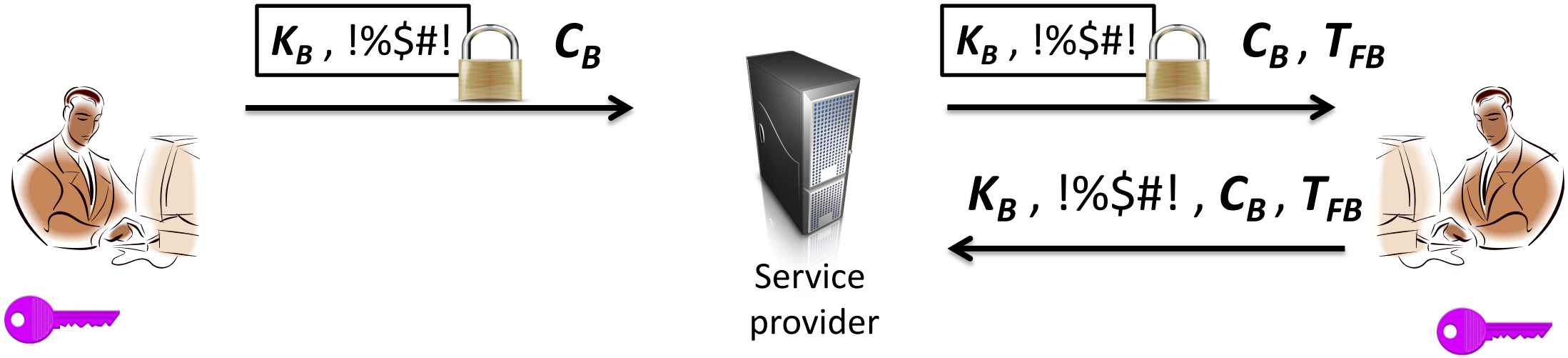
To report abuse, send message as well as K_B , C_B , T_{FB}

Provider can verify C_B , T_{FB}

Provider convinced that sender sent " !%$#! "

Attachments (images, videos) handled differently (stay tuned)

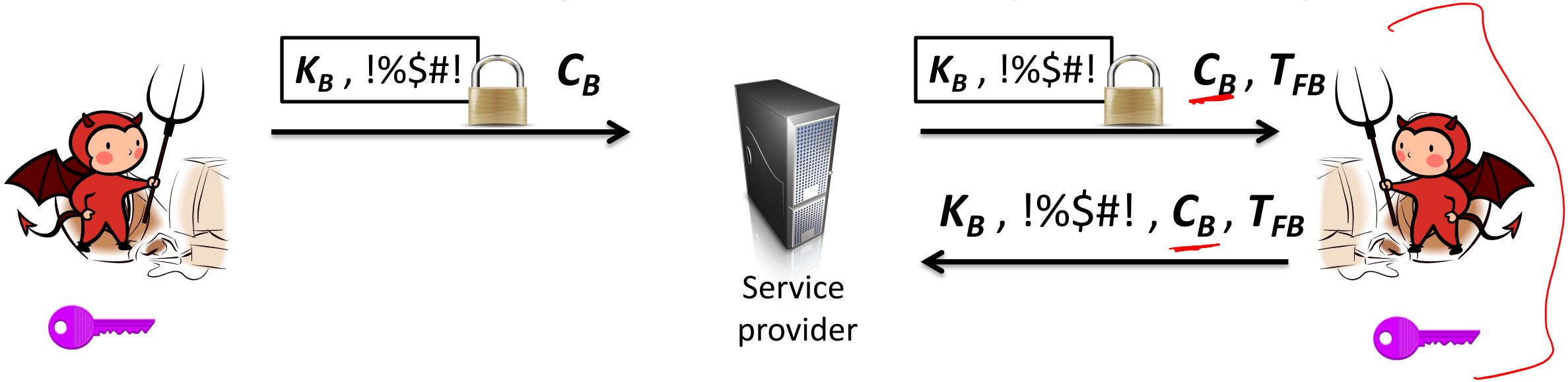
Facebook's message franking protocol



Lots of open questions raised:

- Is Facebook's approach secure? Not for attachments
- Is there useful formal abstraction of encryption plus commitment? Yes!
- Are there faster/better approaches? Yes!

Security goals for message franking

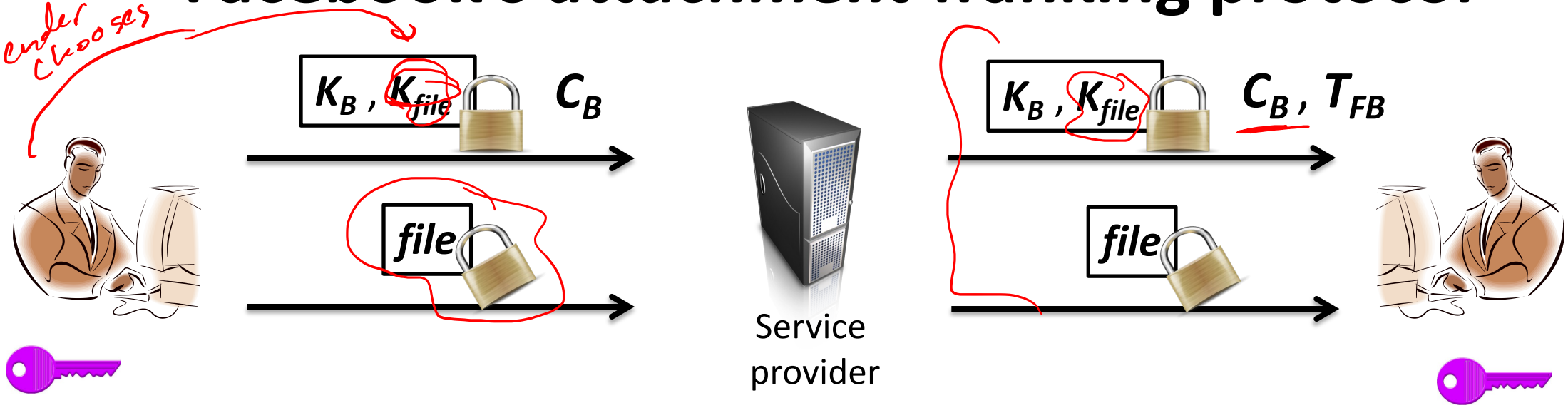


1) ***Receiver binding***: receiver can't report a message not sent
2) ***Sender binding***: can't send a message that can't be reported
3) End-to-end confidentiality/authenticity for messages not reported
4) Deniability: can't convince anyone but provider that message was sent

Our analysis:

Facebook's scheme achieves goals, but not for attachments

Facebook's *attachment* franking protocol



Sender ***cryptographically commits*** to attachment encryption key:

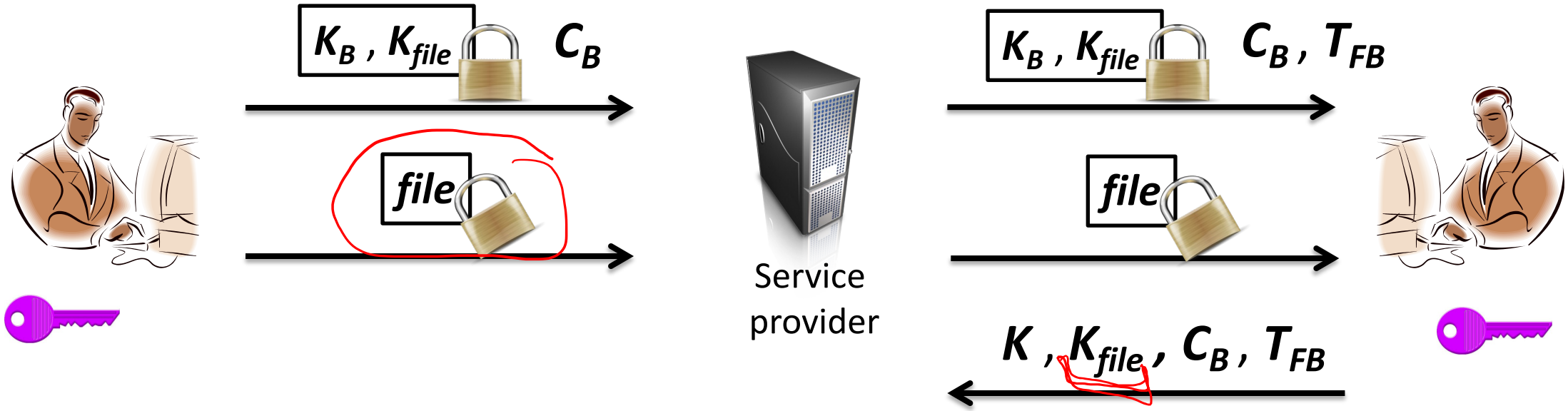
$$C_B = HMAC(K, K_{file})$$

Encrypt-then-HMAC file encryption key K_{file} along with K_B

AES-GCM encrypt attachment: ***AES-GCM(K_{file} , file)***

Receiver decrypts as before to get K_{file} and then decrypts attachment

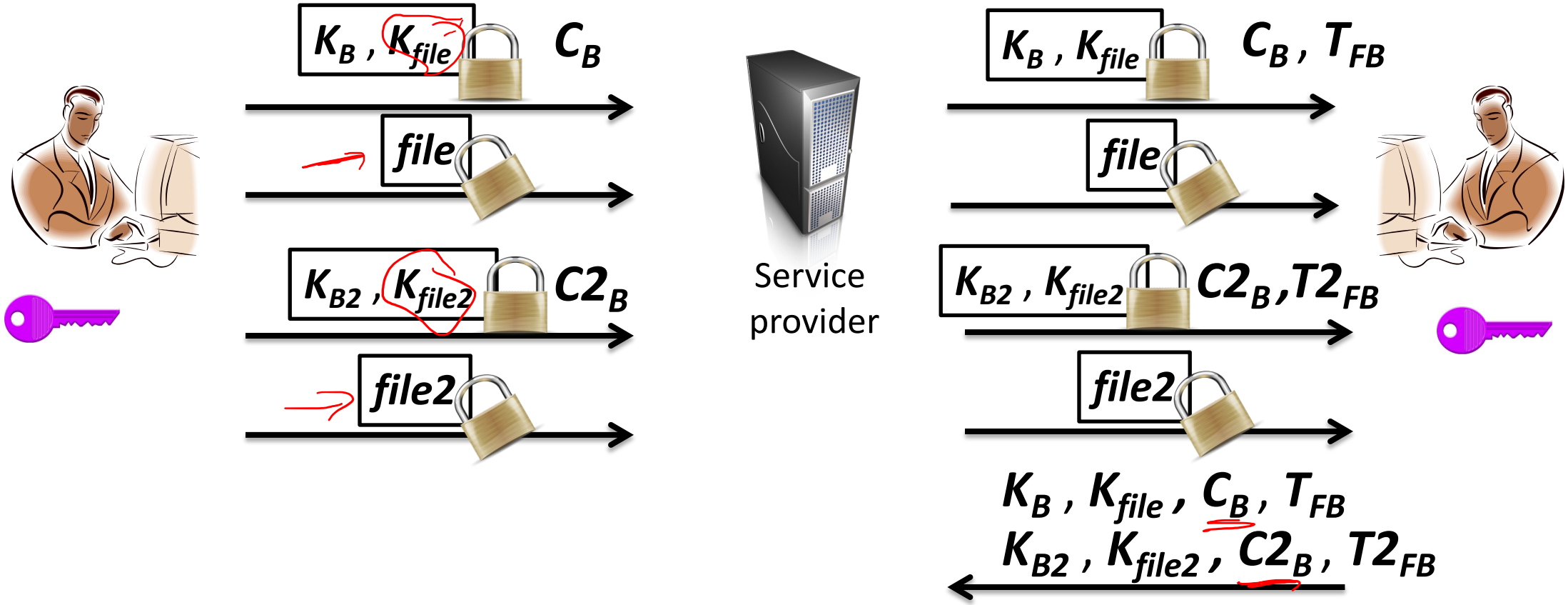
Facebook's *attachment* franking protocol



To report abuse, receiver opens K_{file} and other recent messages

Facebook checks openings & decrypts all unique AES-GCM ciphertexts to add them to abuse report

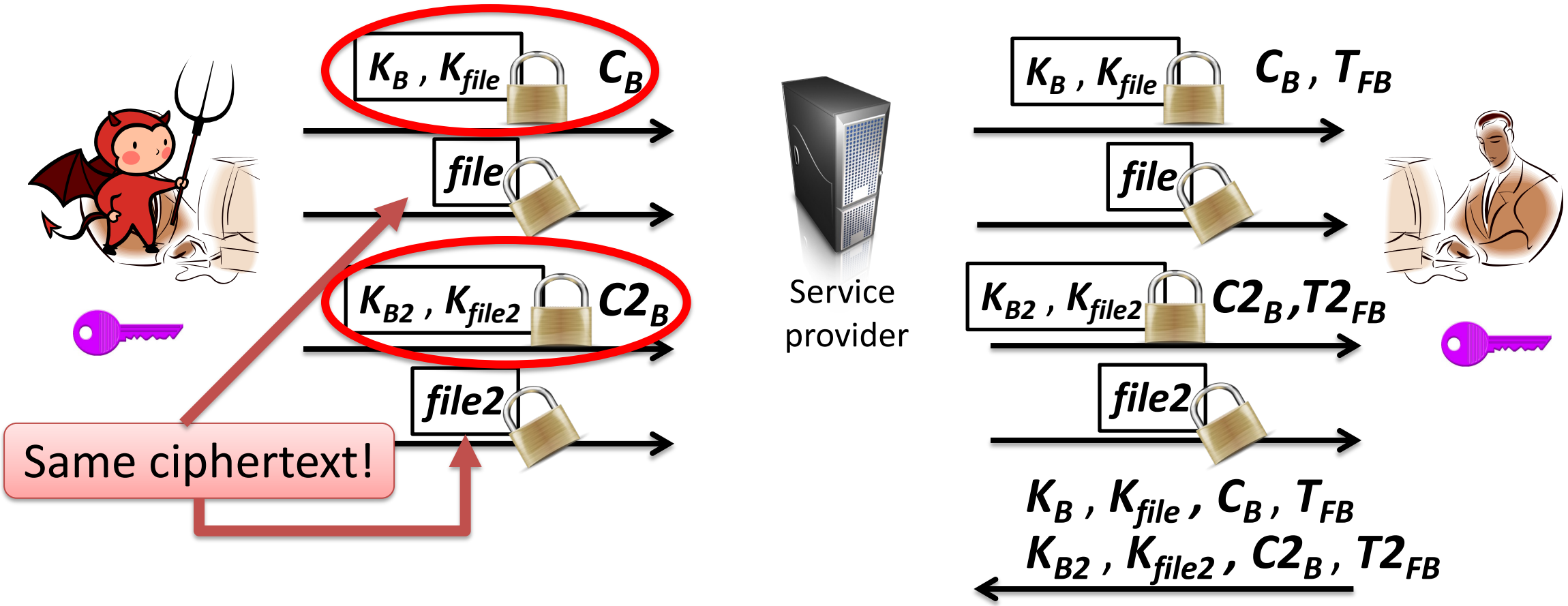
Facebook's *attachment* franking protocol



To report abuse, receiver opens K_{file} and other recent messages

Facebook checks openings & decrypts all unique AES-GCM ciphertexts to add them to abuse report

Our attack exploits AES-GCM



Craft special ***AES-GCM*** ciphertext:

1) Decrypts under K_{file} to innocuous image
2) Decrypts under K_{file2} to abuse image

Only the innocuous image appears in abuse report! (Violates sender binding)

Our attack exploits AES-GCM

Craft special ***AES-GCM*** ciphertext:

1) Decrypts under K_{file} to innocuous image
2) Decrypts under K_{file2} to abuse image

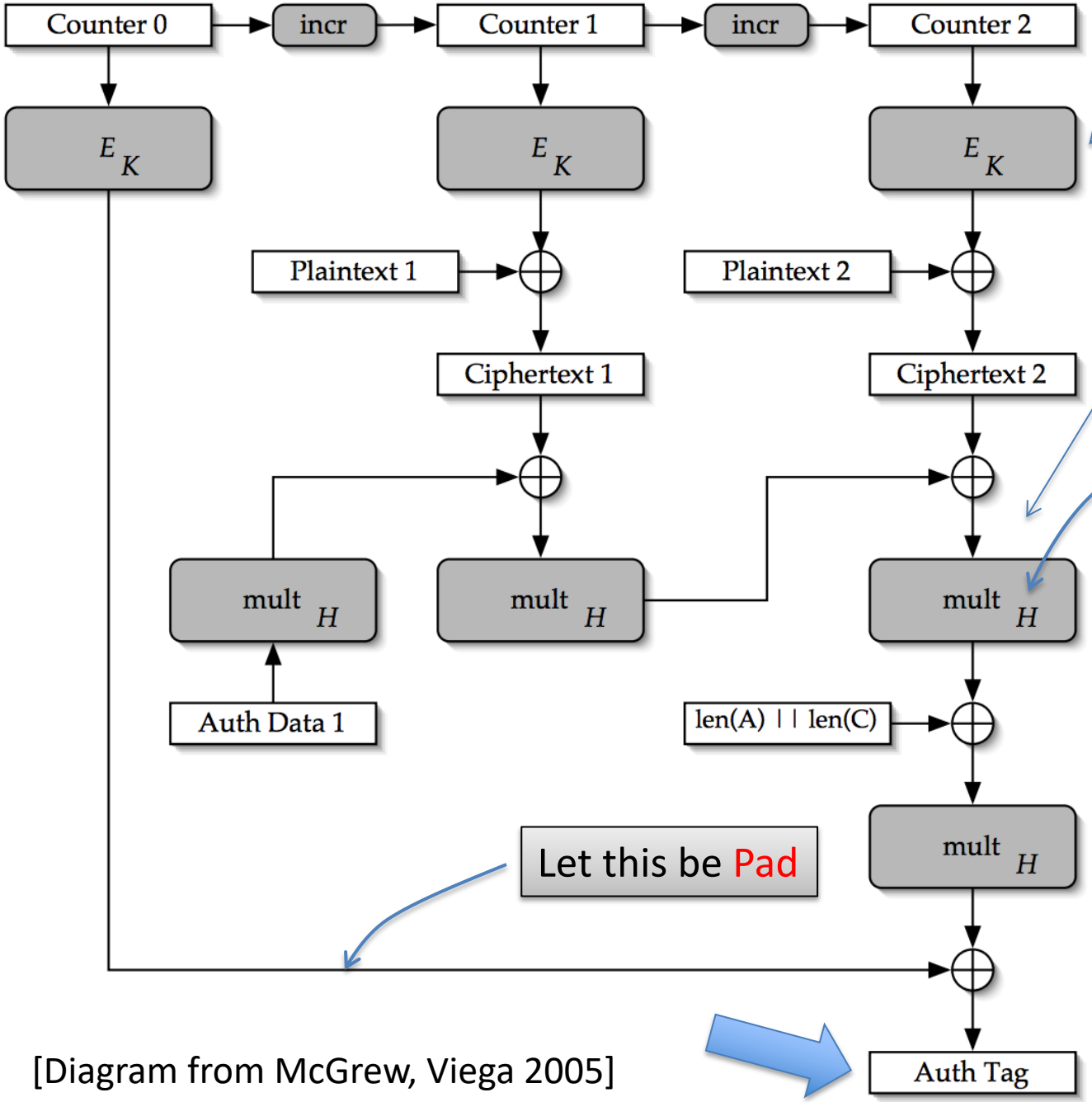
But isn't ***AES-GCM*** a secure authenticated encryption scheme?

Yes, but ... this type of attack is not standard

attacker gets to choose K_{file} and K_{file2}

Our attack exploits that ***AES-GCM*** is **not committing**:
efficient adversary can find ciphertext that decrypts under two keys

[Abdallah, Bellare, Neven 2010] [Farshim et al. 2013] [Farshim et al. 2017]



[Diagram from McGrew, Viega 2005]

CTR mode encryption with AES blockcipher E

Universal hash-based message authentication (called GMAC)

$H = E_K(0^{128})$

Can rewrite GMAC as:

$$\textbf{\textit{Tag}} = \boldsymbol{C_1} * H^3 + \boldsymbol{C_2} * H^2 + \text{len} * H + \text{Pad}$$

1) Pick key $\boldsymbol{K_{file}}$, derive H1, Pad1
2) Pick block of plaintext
3) Let $\boldsymbol{C_1}$ be ciphertext block using $\boldsymbol{K_{file}}$
4) Pick key $\boldsymbol{K_{file2}}$, derive H2, Pad2
5) Solve ***Tag*** equation for $\boldsymbol{C_2}$:

$$\textbf{\textit{Tag}} = \boldsymbol{C_1} * H1^3 + \boldsymbol{C_2} * H1^2 + \text{len} * H1 + \text{Pad1}$$
$$= \boldsymbol{C_1} * H2^3 + \boldsymbol{C_2} * H2^2 + \text{len} * H2 + \text{Pad2}$$

6) Output $\boldsymbol{K_{file}}, \boldsymbol{K_{file2}}, \boldsymbol{C_1}, \boldsymbol{C_2}, \textbf{\textit{Tag}}$

Abusive JPEG seen by receiver,
but not in abuse report

Innocuous BMP
in abuse report

Axolotl: endangered salamander species

- Name of Diffie-Hellman ratcheting used in Signal
- Facebook Messenger source code refers to encrypted messages as salamanders

Worked with Facebook to understand effect of message franking vulnerability & responsibly disclose

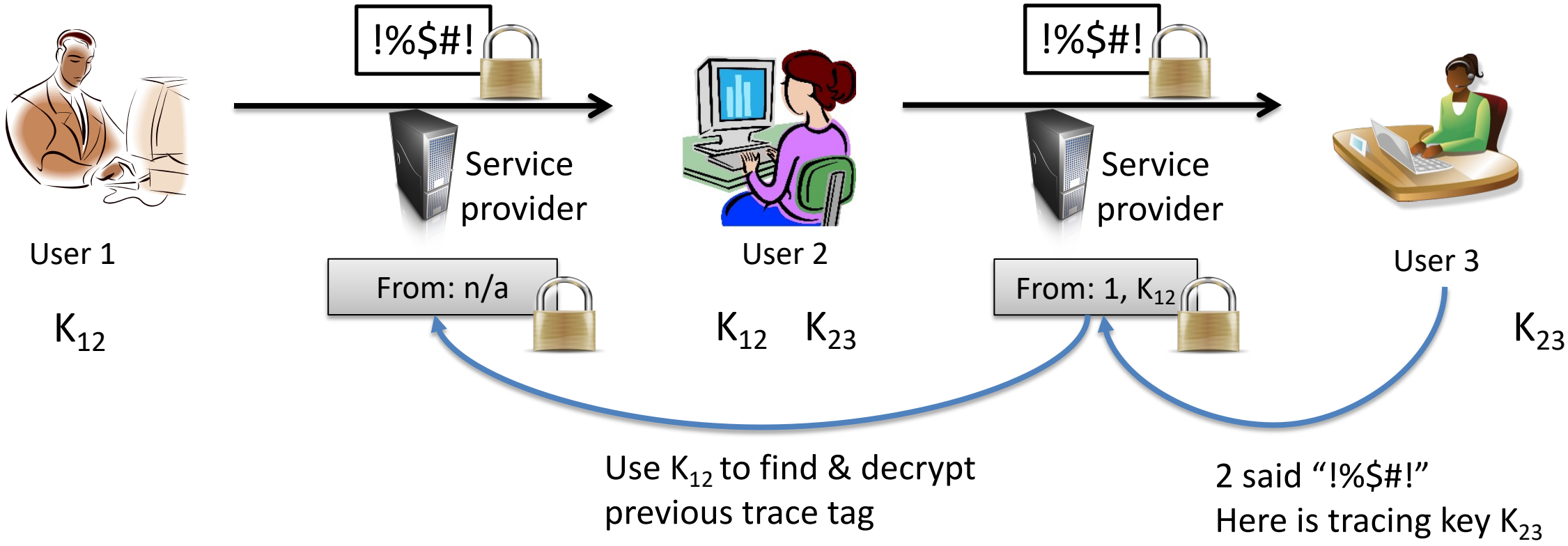
Viral misinformation and cryptographic traceback

APAC OCTOBER 27, 2020 / 6:06 AM / UPDATED 19 DAYS AGO

Fake news spread on WhatsApp to Indian Americans plays stealth role in U.S. election

By Paresh Dave 5 MIN READ

Viral misinformation and cryptographic traceback



Traceback for encrypted messaging:

- Cryptographic abuse report that allows inferring path of message through user network
- Nothing revealed about forwarding behavior & message until report made
- Fast schemes that use only symmetric cryptography
- Lots of interesting open questions

[Tyagi et al. CCS 2019]

Summary

- End-to-end messaging for IM (chat) hot topic, now widely deployed
- Tensions with anti-abuse features, new cryptography needed

Signal Protocol

Signed pre-key

Signature of SPK_B using secret key associated to IK_B

Identity key

One-time pre-keys

Service provider

$IK_B, SPK_B, \sigma , OPK1_B , \dots , OPKn_B$

$IK_B, SPK_B, OPKi_B$ all public DH keys, recipient retains secret keys

$IK_B, SPK_B, \sigma , OPKi_B$

Verify σ

DH(PK1,PK2) is means output $g^{sk1*sk2}$

$r \leftarrow \$ \mathbf{Z}_p$; $EK_A \leftarrow g^r$

$DH1 = DH(IK_A, SPK_B)$

$DH2 = DH(EK_A, IK_B)$

Mutual authentication

$DH3 = DH(EK_A, SPK_B)$

$DH4 = DH(EK_A, OPK_B)$

Forward secrecy

$SK = KDF(DH1 || DH2 || DH3 || DH4)$

$C \leftarrow AEAD(SK,message)$

IK_A , EK_A, key identifiers, C

IK_A , EK_A, key identifiers, C

Summary

- Hybrid encryption uses combination of asymmetric and symmetric cryptography
  - Key encapsulation mechanisms (KEM) based on secure PKE, (elliptic curve) Diffie-Hellman
  - Use an authenticated encryption scheme for data encapsulation mechanism (DEM)
- PGP is historical example (and still somewhat widely used)
- End-to-end messaging for IM, chat hotter topic, now widely deployed

Summary

- Elliptic curves are specially constructed groups where DLP is conjectured to be hard
- These are faster than RSA or DLP over $\mathbf{Z}_p^*$
- Being used widely in practice
  - EC-DSA (bitcoin)
  - TLS EC-DHE (elliptic curve ephemeral DH)
- Post-quantum crypto studies new asymmetric primitives conjectured to resist quantum computers