Key-committing AEAD

Julia Len

Cornell Tech

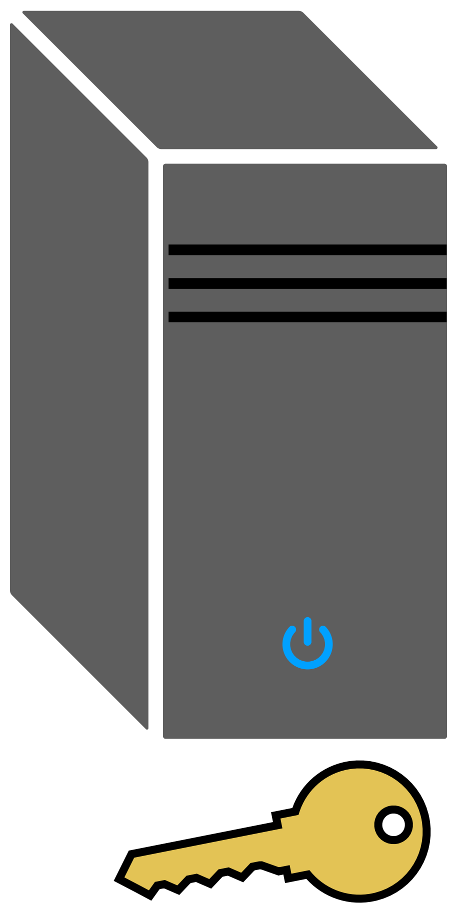
CFRG, IETF 110

For simplicity, we ignore nonces and associated data in this presentation

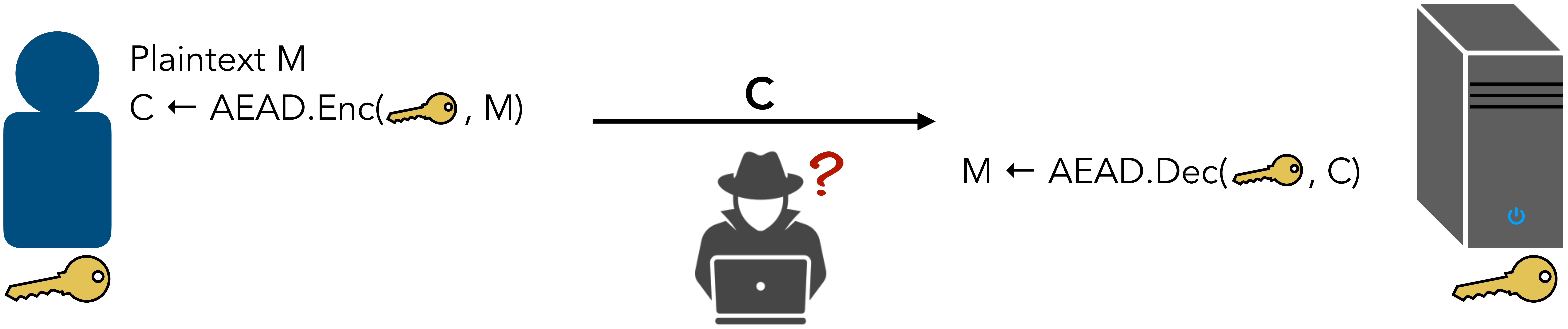
Authenticated Encryption

Plaintext M

C ← AEAD.Enc(, M)

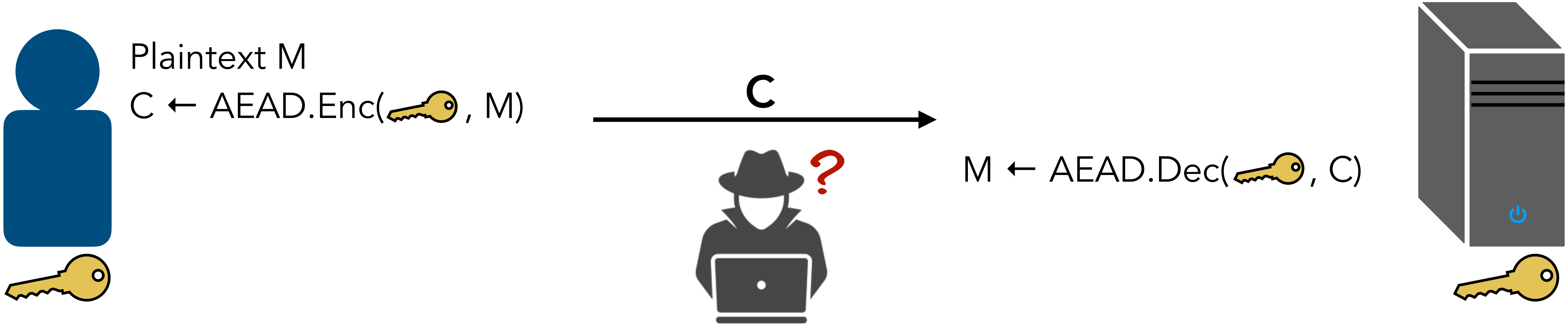


Authenticated Encryption



For simplicity, we ignore nonces and associated data in this presentation

Authenticated Encryption



Popular

- AES-GCM
- XSalsa20/Poly1305
- ChaCha20/Poly1305
- AES-GCM-SIV
- OCB

Easy to use

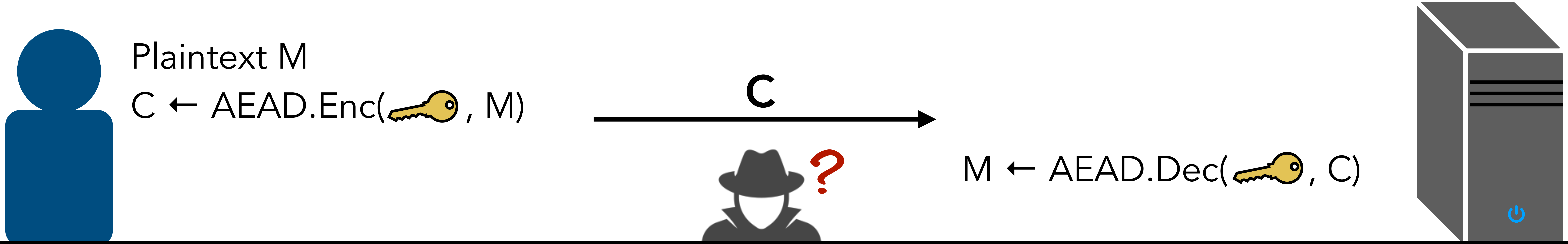
- Efficient
- Standardized
- Widely supported

Secure

- Proven CCA-secure
- Confidentiality
- Integrity

Authenticated Encryption

For simplicity, we ignore nonces and associated data in this presentation



But don’t target robustness, also called **committing AEAD**, as a security goal

[ABN TCC’10], [FLPQ PKC’13] for PKE, [FOR FSE’17] for AEAD

- AES-GCM
- XSalsa20/Poly1305
- ChaCha20/Poly1305
- AES-GCM-SIV
- OCB

- Efficient
- Standardized
- Widely supported

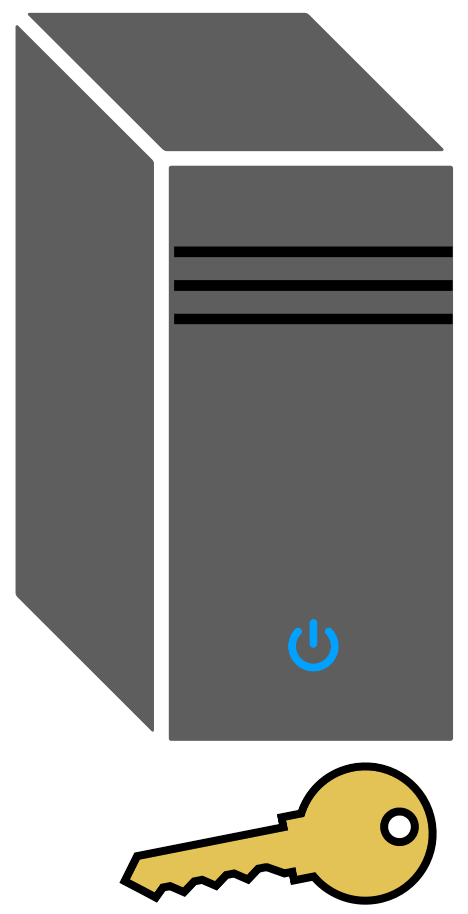
- Proven CCA-secure
- Confidentiality
- Integrity

(Non-) Committing AEAD

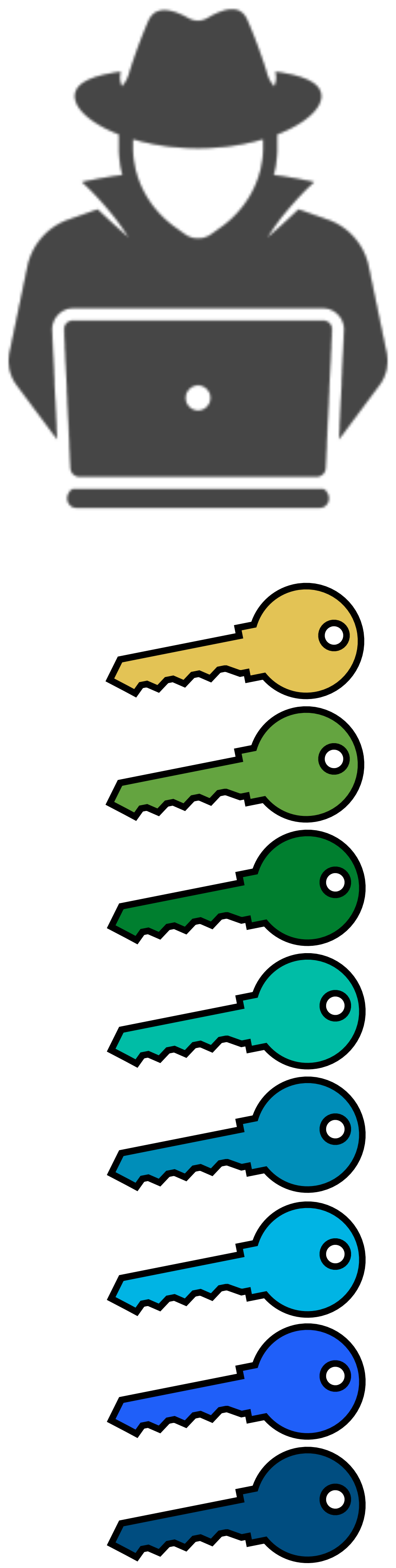
Ciphertext C′

$M_1 \leftarrow \text{AEAD.Dec}(\ ,\ C')$

$M_2 \leftarrow \text{AEAD.Dec}(\ ,\ C')$



Partitioning Oracle Attacks [LGR USENIX'21]



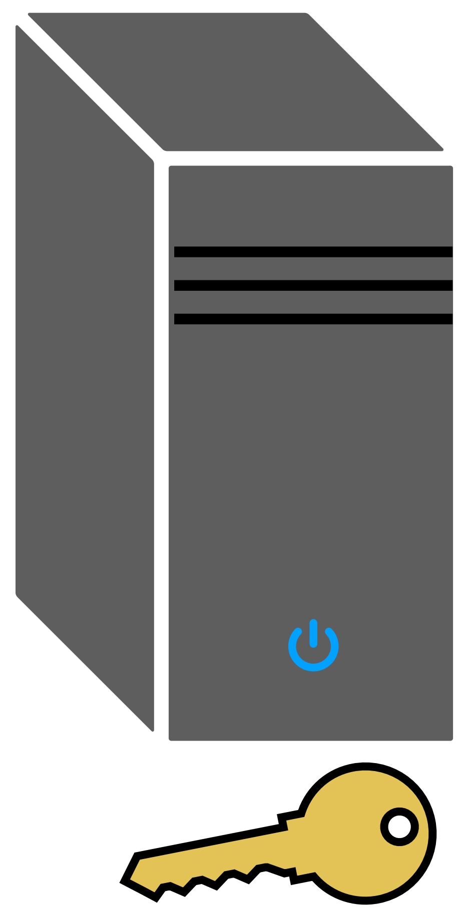
Ciphertext C′

$M_1 \leftarrow$ AEAD.Dec(, C′)

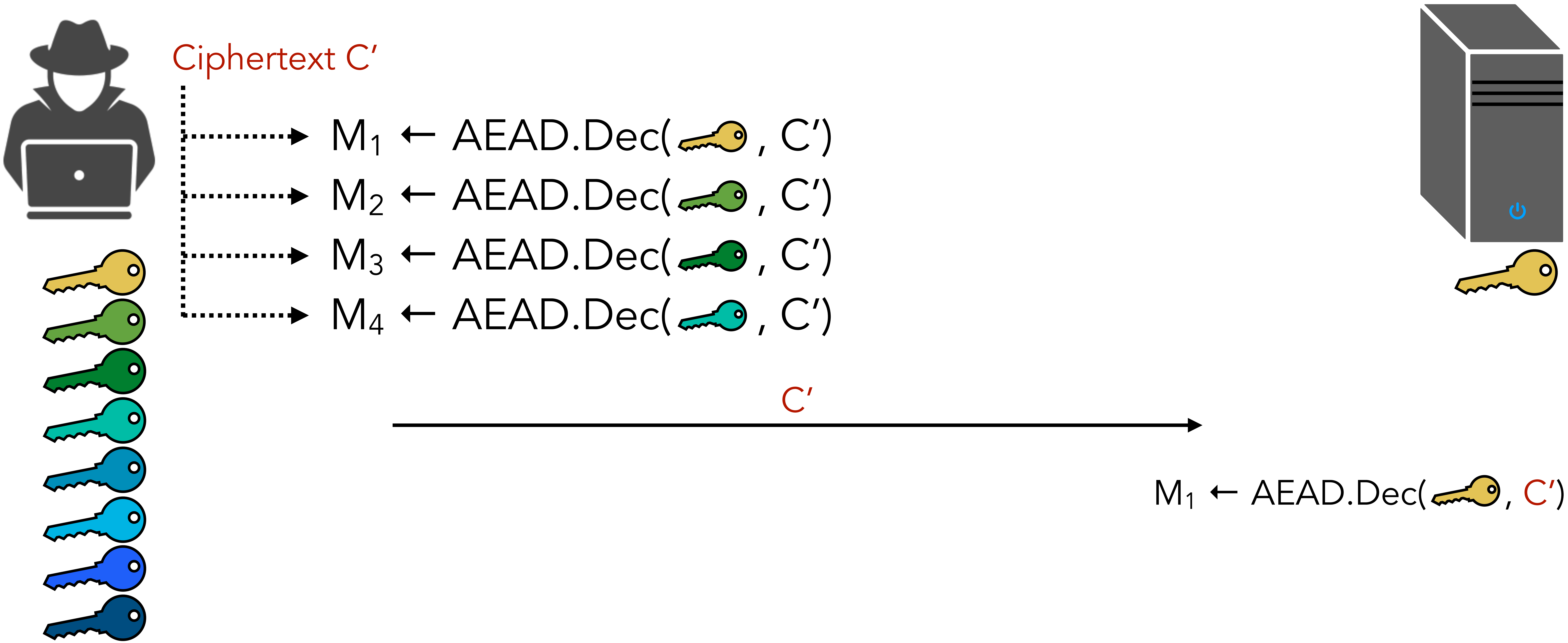
$M_2 \leftarrow$ AEAD.Dec(, C′)

$M_3 \leftarrow$ AEAD.Dec(, C′)

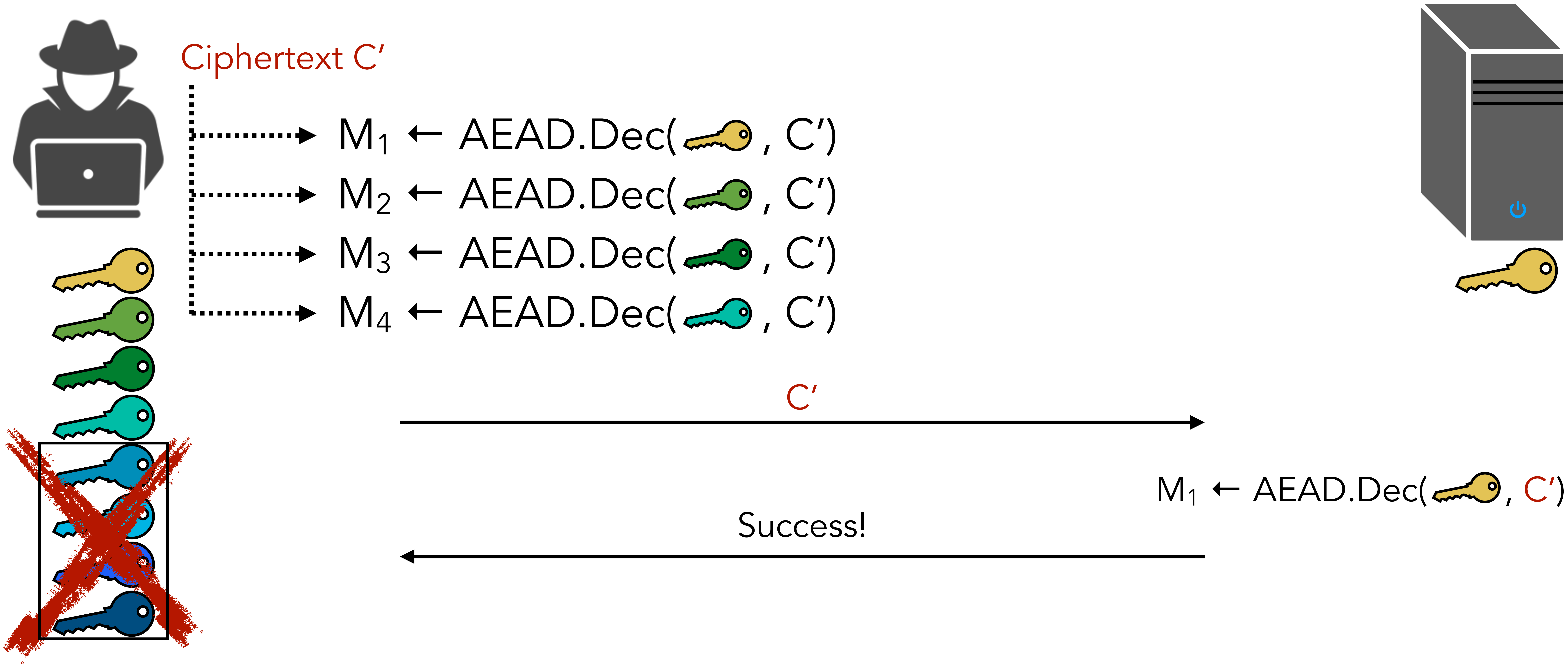
$M_4 \leftarrow$ AEAD.Dec(, C′)



Partitioning Oracle Attacks [LGR USENIX'21]

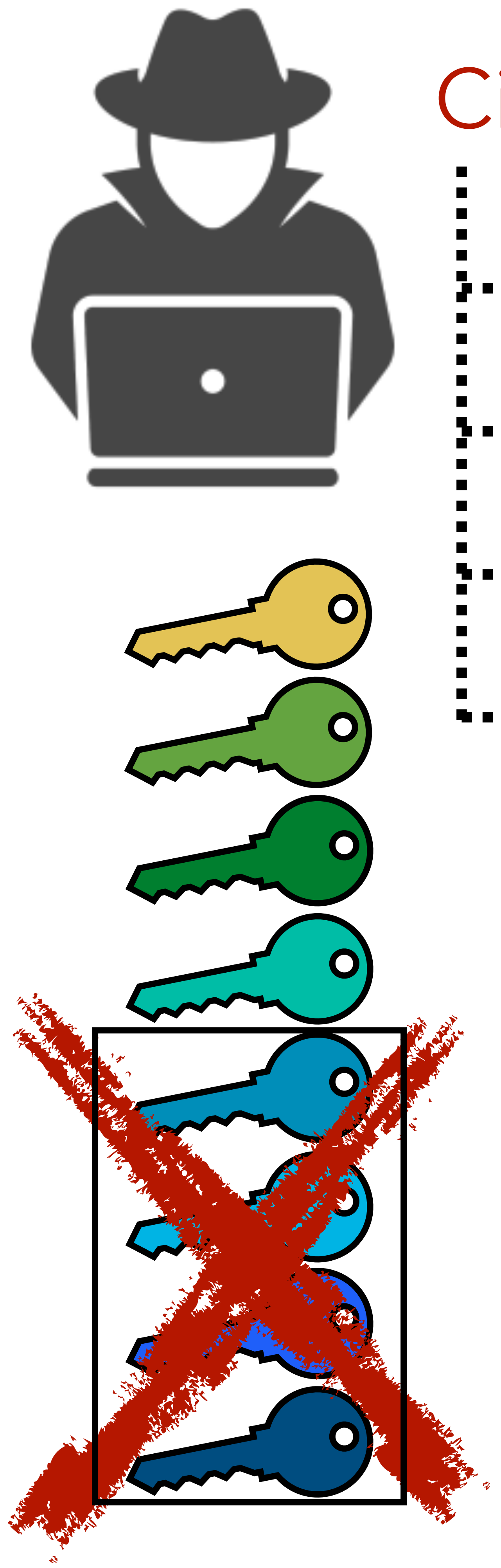


Partitioning Oracle Attacks [LGR USENIX'21]



Partitioning Oracle Attacks [LGR USENIX'21]

The attacker learns 1-bit of information about the key!



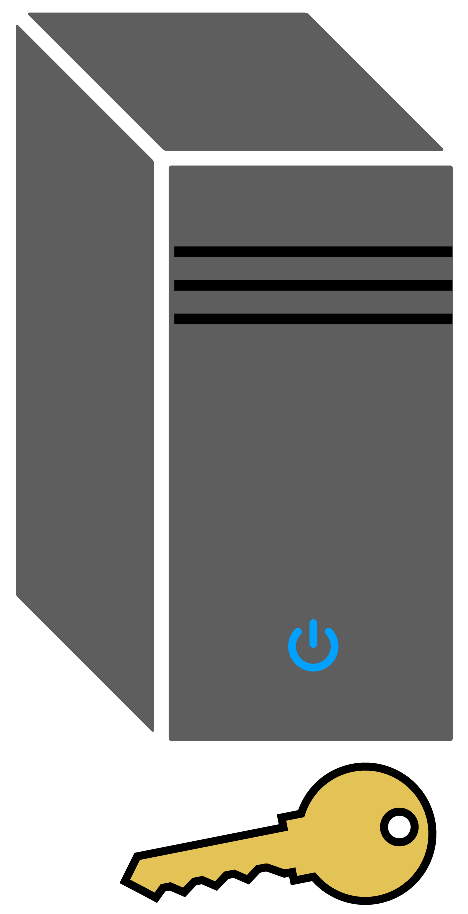
Ciphertext C′

$M_1 \leftarrow$ AEAD.Dec(, C′)

$M_2 \leftarrow$ AEAD.Dec(, C′)

$M_3 \leftarrow$ AEAD.Dec(, C′)

$M_4 \leftarrow$ AEAD.Dec(, C′)



C′

$M_1 \leftarrow$ AEAD.Dec(, C′)

Success!

Vulnerabilities from non-committing AEAD (so far)

- Facebook Messenger
- Message franking

Content moderation

[GLR CRYPTO'17]
[DGRW CRYPTO'18]

- Key rotation in key management services
- Envelope encryption in the AWS encryption SDK
- Subscribe with Google

Services by Google & Amazon

[ADGKLS '20]

Partitioning oracle attacks

Schemes looked at in depth

- Shadowsocks proxy servers for UDP
- Early implementations of the OPAQUE asymmetric PAKE protocol

Possible partitioning oracles

- Hybrid encryption: Hybrid Public-Key Encryption (HPKE)
- Age file encryption tool
- Kerberos drafts (not adopted)
- JavaScript Object Signing and Encryption (JOSE)
- Anonymity systems: use partitioning oracles to learn which public key a recipient is using from a set of public keys

What do we use for key-committing AEAD?

What do we use for key-committing AEAD?

- None currently standardized!

What do we use for key-committing AEAD?

- None currently standardized!

Scheme	Description	Adopted by…	Extra overhead over base scheme	Potential issues?
Zeros Block Check	Modifies AEAD scheme to check that a block of recovered plaintext is all-zeros string	Libsodium	Adds **64 bytes** to each ciphertext	• Side-channel if implemented incorrectly • Need to implement and analyze separately for each AEAD scheme
Hash Key Check	Modifies AEAD scheme to check SHA256 hash of the key during decryption	AWS Encryption SDK	Adds at least **32 bytes** to each ciphertext	Side-channel if implemented incorrectly
Single-key Encrypt-then-HMAC	Plain Encrypt-then-HMAC using single key	-	None!	Less efficient

What do we use for key-committing AEAD?

- None currently standardized!
- As we begin the process of making an internet-draft, we would love to hear your thoughts about needs and requirements

Scheme	Description	Adopted by…	Extra overhead over base scheme	Potential issues?
Zeros Block Check	Modifies AEAD scheme to check that a block of recovered plaintext is all-zeros string	Libsodium	Adds **64 bytes** to each ciphertext	• Side-channel if implemented incorrectly • Need to implement and analyze separately for each AEAD scheme
Hash Key Check	Modifies AEAD scheme to check SHA256 hash of the key during decryption	AWS Encryption SDK	Adds at least **32 bytes** to each ciphertext	Side-channel if implemented incorrectly
Single-key Encrypt-then-HMAC	Plain Encrypt-then-HMAC using single key	-	None!	Less efficient

References

[ABN TCC’10] Michel Abdalla, Mihir Bellare, Gregory Neven. Robust Encryption. TCC, 2010.

[FLPQ PKC’13] Pooya Farshim, Benoît Libert, Kenneth Paterson, Elizabeth Quaglia. Robust encryption, revisited. PKC, 2013.

[FOR FSE’17] Pooya Farshim, Claudio Orlandi, Răzvan Roşie. Security of symmetric primitives under incorrect usage of keys. FSE, 2017.

[GLR CRYPTO’17] Paul Grubbs, Jiahui Lu, Thomas Ristenpart. Message franking via committing authenticated encryption. CRYPTO, 2017.

[DGRW CRYPTO’18] Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, Joanne Woodage. Fast message franking: From invisible salamanders to encryptment. CRYPTO, 2018.

[LGR USENIX’21] Julia Len, Paul Grubbs, Thomas Ristenpart. Partitioning Oracle Attacks. USENIX Security, 2021.

[ADGKLS ’20] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, Sophie Schmieg. How to abuse and fix authenticated encryption without key commitment. ePrint 2020/1456.