Advanced Cryptography
CS 655

Week 2:

- Authenticated Encryption with Associated Data
- Concrete (Multi-User) Security Analysis of AES-GCM
- Partitioning Oracle Attacks
- AES-GCM-SIV

Spring 2023

Authenticated Encryption with Associated Data

- AE.KeyGen: Generates random key K
- AE.Enc(K,N,M,H)
  - **Inputs:** Key: K, Nonce: N, Message: M, Header: H (associated data)
  - **Output:** ciphertext C
- AE.Dec(K,C,H)
  - **Inputs:** Key: K, Ciphertext**:** C, Header**:** H (associated data)
  - **Output:** message m (or “Invalid Ciphertext”)

Ideal Cipher Model

- For all keys K $E(K,.)$ is a truly random permutation with inverse $E^{-1}(K,.)$
- All parties (adversary + honest) have access to oracles $E(.,.)$ and $E^{-1}(.,.)$

- AE.Enc(K,N,M,H)
  - **Inputs:** Key: K, Nonce: N, Message: M, Header: H (associated data)
  - **Output:** ciphertext C
  - Will query $E(K,.)$ and/or $E^{-1}(K,.)$ to generate C

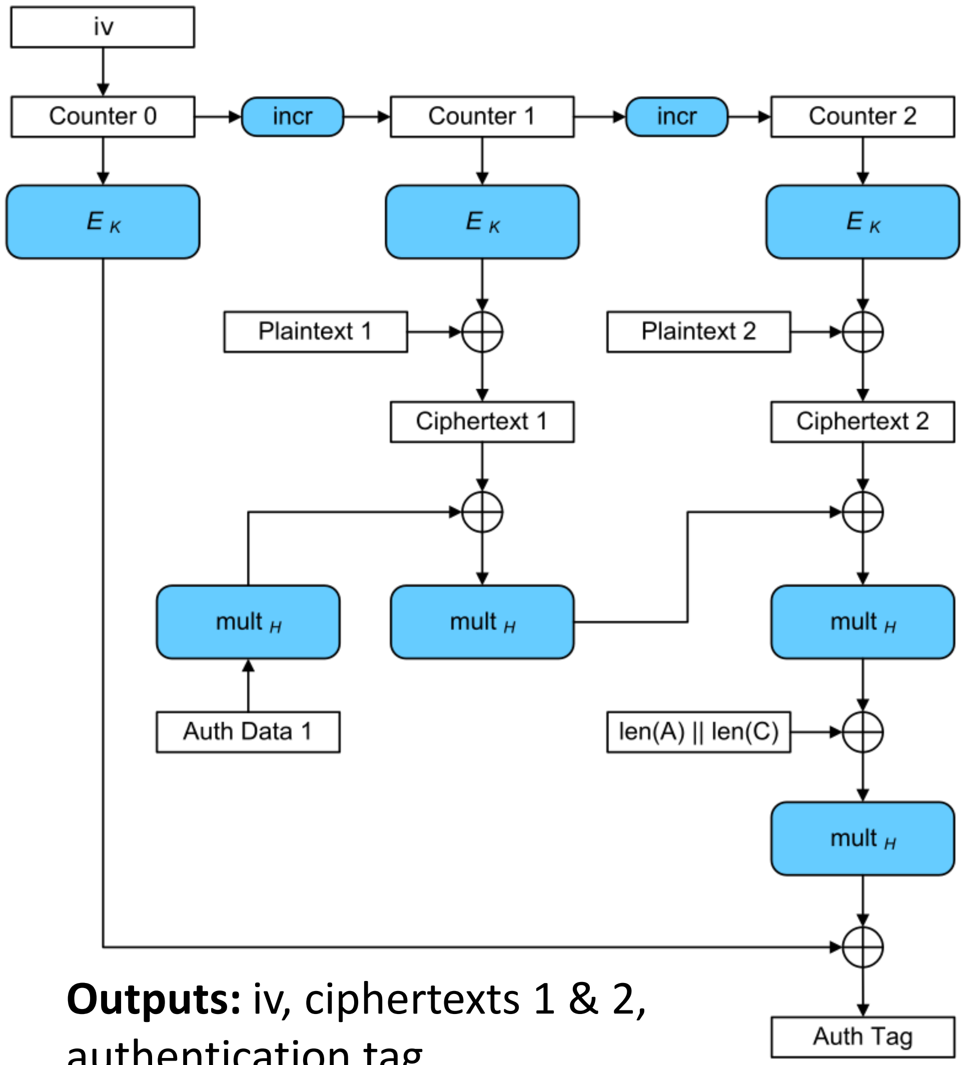
- AE.Dec(K,C,H)
  - **Inputs:** Key: K, Ciphertext**:** C, Header**:** H (associated data)
  - **Output:** message m (or "Invalid Ciphertext")
  - Will query $E(K,.)$ and/or $E^{-1}(K,.)$ to generate C

- Attacker my query $E(.,.)$ and $E^{-1}(.,.)$, but does not know secret key K

Galois Counter Mode (GCM)

- AES-GCM
- **Security Guarantee:** Authentication Encryption with Associated Data
  - Message can be arbitrarily long
  - Length of message and authentication data is authenticated to avoid truncation attacks etc…
  - Public Associated Data is Authenticated
    - Source IP
    - Destination IP
    - Why can't these values be encrypted?

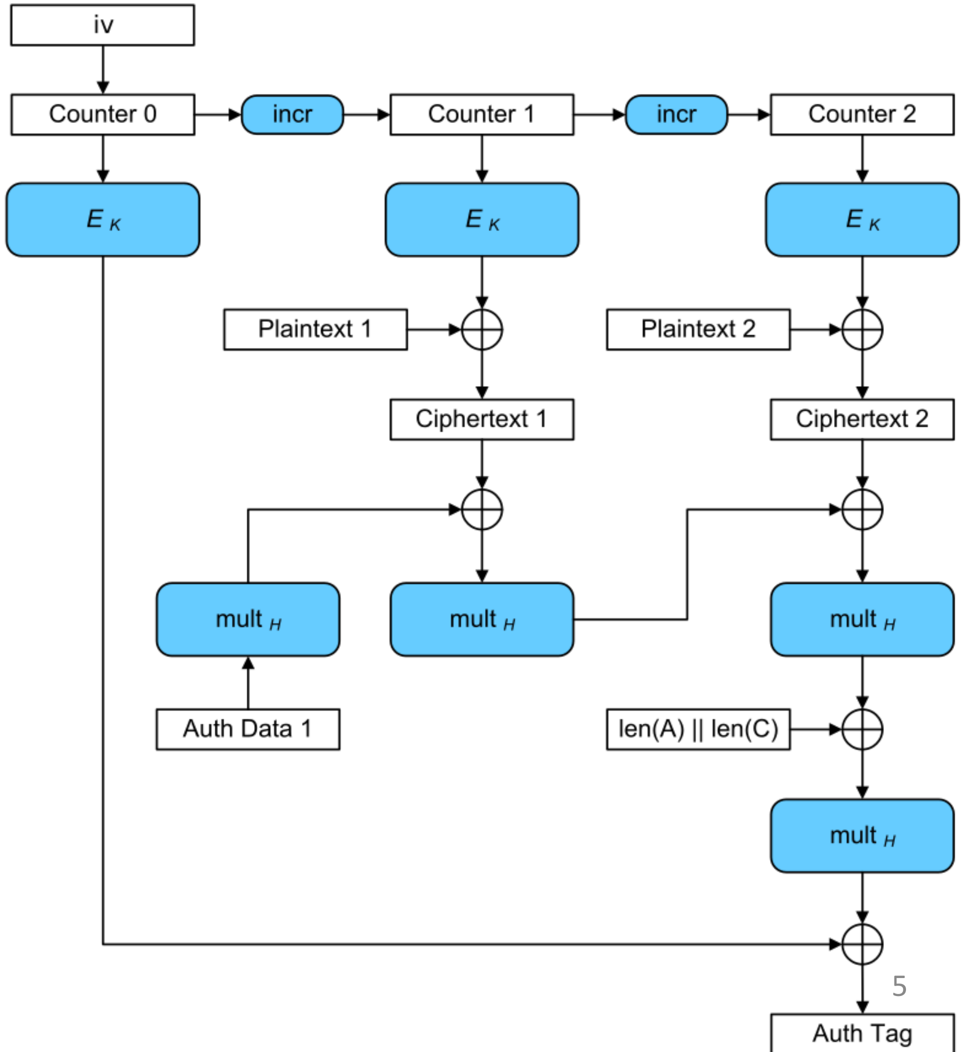
Input: plaintexts 1 & 2



Outputs: iv, ciphertexts 1 & 2, authentication tag

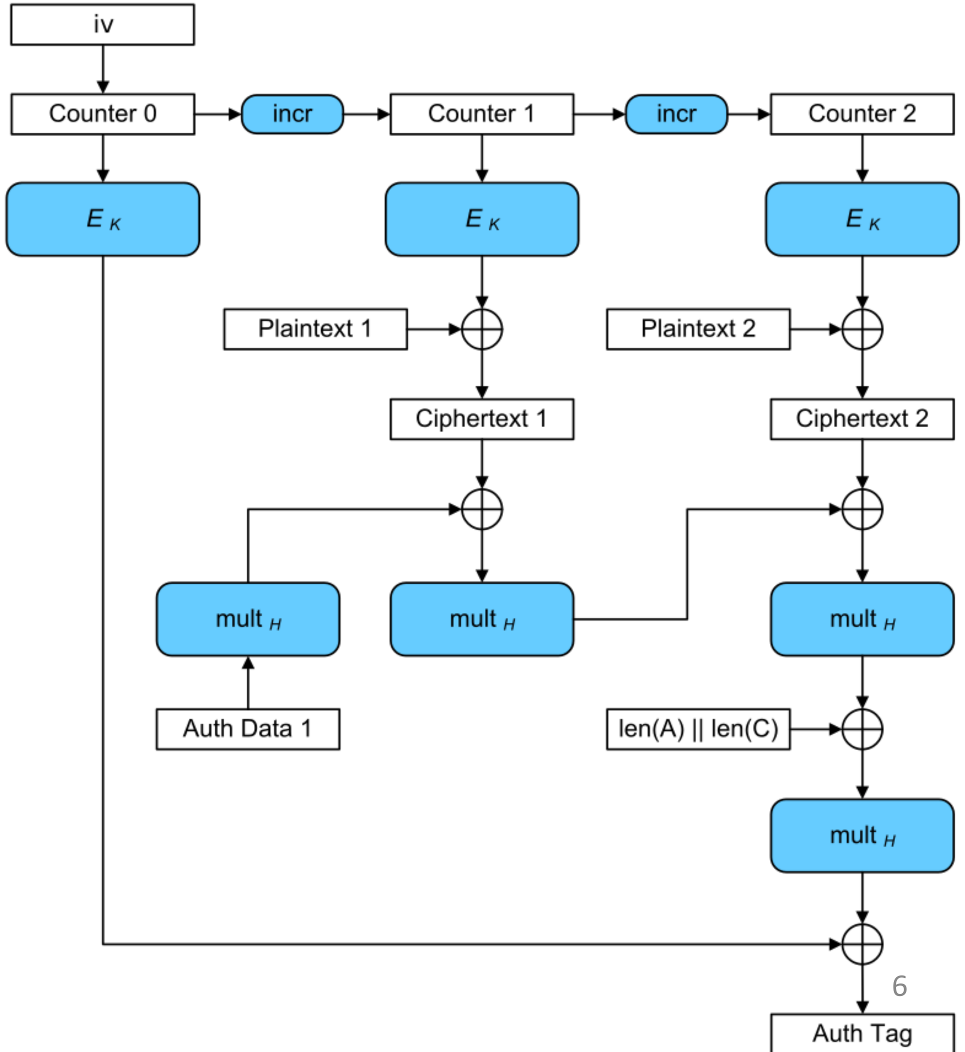
GCM: Nonce Collision

- AES-GCM
- **Suppose that message m_1 is b_1 blocks long and message m_2 is b_2 block long.**
- **Suppose that we pick nonces N_1 and N_2**
- **How should we define nonce collision?**
- **What is the probability of this event?**



GCM: Nonce Collision

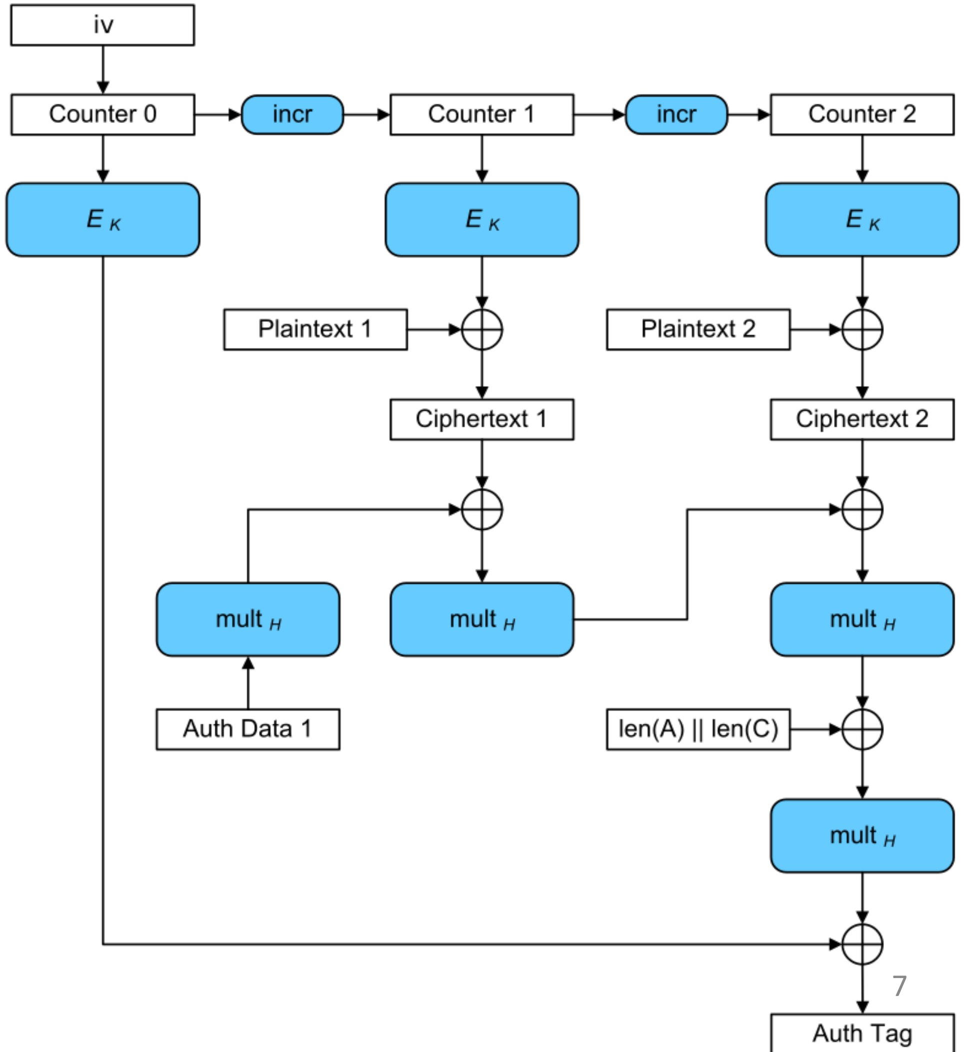
- AES-GCM
- **Suppose that message m_1 is b_1 blocks long and message m_2 is b_2 block long.**
- **Suppose that we pick nonces N_1 and N_2**
- **How should we define nonce collision?**
  - **If interval $[N_1, N_1+b_1]$ intersects with $[N_2, N_2+b_2]$ then there could be problems. Why?**



GCM: Nonce Collision

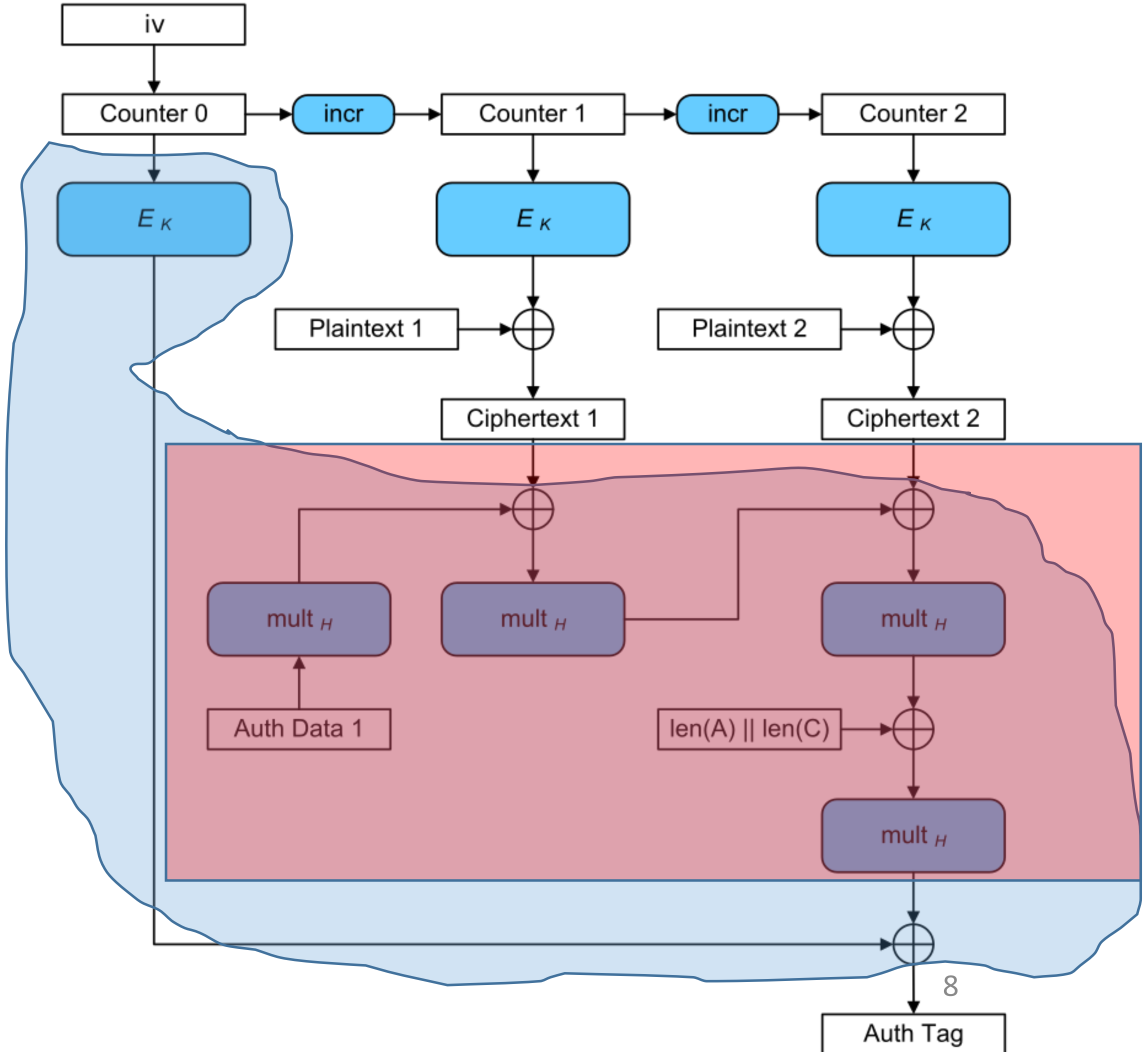
- AES-GCM
- **Suppose that message m_1 is b_1 blocks long and message m_2 is b_2 block long.**
- **Suppose that we pick nonces N_1 and N_2**
- **How should we define nonce collision?**
  - **If interval $[N_1,N_1+b_1]$ intersects with $[N_2, N_2+b_2]$ then there could be problems. Why?**
  - **Collision if N_2 is in $[N_1-b_2,N_1+b_1]$**
  - **Probability of a collision $2^{-\lambda}(b_1 + b_2 + 1)$**
- **Union Bound: Probability of any nonce collision over all pairs of queries**

$$2^{-\lambda} \sum_{i<j\leq q_e} (bi + bj + 1)$$



Galois Counter Mode (GCM)

- AES-GCM
- **Decryption?**
  - **Step 1: Recompute authentication tag from available data**
    - $\mathbf{H(k, A, C, |C|, |A|) \coloneqq E_k}\,(\boldsymbol{N}) \oplus \mathbf{G(A, C, |C|, |A|)}$
    - **Nonce: N, Authentication Data: A**
    - **Length: |C|**
    - **Length: |A|**
    - **Ciphertext Blocks: C1,C2,**
    - **If authentication tag does not match then output "Invalid Ciphertext"**
  - **Step 2:** $\mathbf{m_i = Ek(N + i)} \oplus \boldsymbol{C_i}$ **for each block i**



Parameters and Definitions

- κ: length of secret key (bits)
- λ: length of block (bits)

Definition: We say that a hash function H is ε-almost XOR-universal if for all distinct messages m_1 and m_2 and all strings s we have

$$\Pr[H(k, m_1) \oplus H(k, m_2) = s] \leq \varepsilon$$

Where the randomness is taken over the selection of the secret key k.

McGrew and Viega [24, Lemma 2] show that H has this property for $\epsilon(m, n) = (\lceil m/\lambda \rceil + \lceil n/\lambda \rceil + 1)/2^{\lambda}$.

AES-GCM: Nonces

- **Option 1:** Random N
  - **Advantage:** Stateless + simple to implement,
  - **Disadvantage:** It is possible for a nonce to collide (typical solution: generate fresh keys after 2^{32} messages to keep probability of a nonce collision small)

- **Option 2:** Both parties increment N after each message
  - **Advantage:** Avoids nonce collisions ☺
  - **Disadvantage:**
    - Requires keeping track of current value.
    - Implementation Challenges. What if packets are dropped?
    - Security issue if implementation is buggy or if counter is accidently reset (e.g., radiation)

Multi-User Security

- Suppose that u users generate independent κ bit keys $K_1,\ldots,K_u$
- Attacker may be happy to decrypt just one ciphertext intercepted from any of these use (or just tamper with just one ciphertext for sent to any of these users)
- **General Reduction:** If the encryption scheme is (t,q,eps)-secure with respect to a single user then it provides (t,q,u*eps)-multi-user security
- **Reduction? Can we do better for AES-GCM?**

Multi-User Security Game for AEAD

- Challenger picks a random bit b and Generates u independent keys $K_1, \ldots, Ku$
  - **Real Mode: b=1**
  - **Ideal Mode: b=0**
- **Attacker Goal: guess b**

- **Attacker Oracles:**
  - **Ideal Cipher**
  - **Encryption oracle** (Takes as input an individual $i \leq u$, nonce N, message M, header H) :
    - Outputs: “Invalid” if pair (i,N) is repeated (Attacker not allowed to repeat nonce for individual user)
    - **Real Mode:** Encrypts message using key K_i and outputs ciphertext
    - **Ideal Mode:** Returns random string instead of ciphertext
  - **Verification Oracle:** (Takes as input individual $i \leq u$, nonce N, ciphertext M, header H):
    - Outputs 1 if this ciphertext was generated via a query to the encryption oracle with same user/nonce/header; otherwise
    - **Ideal Mode:** Output 0
    - **Real Mode:** Attempt to decrypt using key K_i; output 0 if decryption fails and 1 otherwise

Source: Bellare, Tackmann, Multi-User Security of Authenticated Encryption: AES-GCM in TLS 1.3

Multi-User Security Game Oracles

Game $\mathbf{G}^{\text{mu-ind}}_{\mathsf{AE}}(A)$

$b \leftarrow_{\$} \{0,1\}$; $b' \leftarrow_{\$} A^{\text{New},\text{Enc},\text{Vf},\text{E},\text{E}^{-1}}$
Return $(b' = b)$

$\text{New}()$

$v \leftarrow v + 1$; $K_v \leftarrow_{\$} \{0,1\}^{\mathsf{AE.kl}}$

$\text{Enc}(i, N, M, H)$

If not $(1 \le i \le v)$ then return $\perp$
If $((i, N) \in U)$ then return $\perp$
$C_1 \leftarrow \mathsf{AE.Enc}^{\text{E},\text{E}^{-1}}(K_i, N, M, H)$
$C_0 \leftarrow_{\$} \{0,1\}^{\mathsf{AE.cl}(|M|)}$
$U \leftarrow U \cup \{(i, N)\}$; $V \leftarrow V \cup \{(i, N, C_b, H)\}$
Return C_b

$\text{Vf}(i, N, C, H)$

If not $(1 \le i \le v)$ then return $\perp$
If $((i, N, C, H) \in V)$ then return true
If $(b = 0)$ then return false
$M \leftarrow \mathsf{AE.Dec}^{\text{E},\text{E}^{-1}}(K_i, N, C, H)$
Return $(M \neq \perp)$

$\text{E}(L, x)$

If $T[L, x] = \perp$ then
  $T[L, x] \leftarrow_{\$} \overline{\text{im}\, T[L, \cdot]}$
  $T^{-1}[L, T[L, x]] \leftarrow x$
Return $T[L, x]$

$\text{E}^{-1}(L, y)$

If $T^{-1}[L, y] = \perp$ then
  $T^{-1}[L, y] \leftarrow_{\$} \overline{\text{im}\, T^{-1}[L, \cdot]}$
  $T[L, T^{-1}[L, y]] \leftarrow y$
Return $T^{-1}[L, y]$

Source: Bellare, Tackmann, Multi-User Security of Authenticated Encryption: AES-GCM in TLS 1.3

Theorem 8. *Let $\kappa, \lambda, \nu \geq 1$ be such that $\nu \leq \lambda - 2$. Let $\mathsf{H}\colon \{0,1\}^\lambda \times (\{0,1\}^* \times \{0,1\}^*) \to \{0,1\}^\lambda$ be an ϵ-almost XOR-universal hash function, for some $\epsilon\colon \mathbb{N} \times \mathbb{N} \to [0,1]$. Let $\mathsf{CAU} = \mathbf{CAU}[\mathsf{H}, \kappa, \lambda, \nu]$. Let A be an adversary that makes at most u queries to its* New *oracle, q_e queries to its* Enc *oracle with messages of length at most ℓ_{bit} bits, q_v queries to its* Vf *oracle with messages of length at most $\ell_{\text{bit}} + \lambda$ bits, and p queries to its* E *and* E^{-1} *oracles. Assume furthermore that $q_e \leq 2^\nu$ and $\ell_{\text{bit}} \leq \lambda(2^{\lambda-\nu} - 2)$. Then*

$$\mathsf{Adv}^{\text{mu-ind}}_{\mathsf{CAU}}(A) \leq \frac{up}{2^\kappa} + \frac{u(\ell_{\text{blk}}(q_e + q_v) + 1)^2 \cdot}{2^{\lambda+1}} + \frac{u(u-1)}{2^{\kappa+1}} + uq_v \cdot \epsilon(\ell_{\text{bit}}, \ell_{\text{head}}),$$

for $\ell_{\text{blk}} = \lceil \ell_{\text{bit}}/\lambda \rceil + 1$ and where the AEAD headers are restricted to ℓ_{head} bits.

- **Though Question:** Which parameters do we expect to be large in practice? qe, qv or p?

Theorem 8. *Let $\kappa, \lambda, \nu \geq 1$ be such that $\nu \leq \lambda - 2$. Let* $\mathsf{H}\colon \{0,1\}^\lambda \times (\{0,1\}^* \times \{0,1\}^*) \to \{0,1\}^\lambda$ *be an ϵ-almost XOR-universal hash function, for some* $\epsilon\colon \mathbb{N} \times \mathbb{N} \to [0,1]$. *Let* $\mathsf{CAU} = \mathbf{CAU}[\mathsf{H}, \kappa, \lambda, \nu]$. *Let A be an adversary that makes at most u queries to its* New *oracle, q_e queries to its* Enc *oracle with messages of length at most ℓ_{bit} bits, q_v queries to its* Vf *oracle with messages of length at most $\ell_{\mathrm{bit}} + \lambda$ bits, and p queries to its* E *and* E^{-1} *oracles. Assume furthermore that $q_e \leq 2^\nu$ and $\ell_{\mathrm{bit}} \leq \lambda(2^{\lambda-\nu} - 2)$. Then*

$$\mathsf{Adv}^{\mathrm{mu\text{-}ind}}_{\mathsf{CAU}}(A) \leq \frac{up}{2^\kappa} + \frac{u(\ell_{\mathrm{blk}}(q_e + q_v) + 1)^2 \cdot}{2^{\lambda+1}} + \frac{u(u-1)}{2^{\kappa+1}} + uq_v \cdot \epsilon(\ell_{\mathrm{bit}}, \ell_{\mathrm{head}}),$$

for $\ell_{\mathrm{blk}} = \lceil \ell_{\mathrm{bit}}/\lambda \rceil + 1$ and where the AEAD headers are restricted to ℓ_{head} bits.

- **P: may be very large (can compute E(.,.) offline)**
- qe, qv require cooperation from a party who knows secret key

Hybrid Argument: Slowly Make Real/Ideal Oracles Identical

- **Hybrid 0: Original Game**
  - **Challenger Generates u <u>independent</u> keys $K_1, \ldots, Ku$**
  - Note: It is possible that the attacker gets lucky and that $K_i = K_j$ for some users i and j.

- **Question:** How could attacker attacker exploit this?
- **Question 2:** What is the probability of the bad event KCOLLISION that there exists a key collision?

- **Hybrid 1:** Original game, but random keys are selected subject to the constraint that they all are distinct .

- **Question:** What is the probability that an attacker can distinguish between hybrids 0 and 1?

Hybrid Argument: Slowly Make Real/Ideal Oracles Identical

- **Hybrid 0: Original Game in Real Mode (b=0):**
  - **Challenger Generates u <u>independent</u> keys $\boldsymbol{K_1}, \dots, \boldsymbol{Ku}$**
  - Note: It is possible that the attacker gets lucky and that $\boldsymbol{K_i} = \boldsymbol{K_j}$ for some users i and j.

- **Question 2:** What is the probability of the bad event KCOLLISION that there exists a key collision?

- **Hybrid 1:** Original game, but random keys are selected subject to the constraint that they all are distinct .

- **Question:** What is the probability that an attacker can distinguish between hybrids 0 and 1?
- **Answer:** at most $\Pr[\text{KCOLLISION}] \leq 2^{-\kappa}\binom{u}{2}$

Theorem 8. *Let $\kappa, \lambda, \nu \geq 1$ be such that $\nu \leq \lambda - 2$. Let $\mathsf{H}\colon \{0,1\}^\lambda \times (\{0,1\}^* \times \{0,1\}^*) \to \{0,1\}^\lambda$ be an ϵ-almost XOR-universal hash function, for some $\epsilon\colon \mathbb{N} \times \mathbb{N} \to [0,1]$. Let $\mathsf{CAU} = \mathbf{CAU}[\mathsf{H}, \kappa, \lambda, \nu]$. Let A be an adversary that makes at most u queries to its* New *oracle, q_e queries to its* Enc *oracle with messages of length at most ℓ_{bit} bits, q_v queries to its* Vf *oracle with messages of length at most $\ell_{\mathrm{bit}} + \lambda$ bits, and p queries to its* E *and* E^{-1} *oracles. Assume furthermore that $q_e \leq 2^\nu$ and $\ell_{\mathrm{bit}} \leq \lambda(2^{\lambda-\nu} - 2)$. Then*

$$\mathsf{Adv}^{\mathrm{mu\text{-}ind}}_{\mathsf{CAU}}(A) \leq \frac{up}{2^\kappa} + \frac{u(\ell_{\mathrm{blk}}(q_e + q_v) + 1)^2}{2^{\lambda+1}} \cdot + \frac{u(u-1)}{2^{\kappa+1}} + uq_v \cdot \epsilon(\ell_{\mathrm{bit}}, \ell_{\mathrm{head}}),$$

for $\ell_{\mathrm{blk}} = \lceil \ell_{\mathrm{bit}}/\lambda \rceil + 1$ and where the AEAD headers are restricted to ℓ_{head} bits.

Hybrid Argument: Slowly Make Real/Ideal Oracles Identical

- **Hybrid 2:**
  - Instead of using E(K_i,.) in the encryption oracle the we replace E(K_i,.) with a fresh random permutation f_i for each user

- **Tempting Argument:** Hybrid 1 is indistinguishable from Hybrid 2 since E(K_i,.) is already a truly random permutation.

- What is the flaw in this argument?

Hybrid Argument: Slowly Make Real/Ideal Oracles Identical

- **Hybrid 2:**
  - Instead of using E(K_i,.) in the encryption oracle the we replace E(K_i,.) with a fresh random permutation f_i for each user

- **Tempting Argument:** Hybrid 1 is indistinguishable from Hybrid 2 since E(K_i,.) is already a truly random permutation.

- What is the flaw in this argument?
- **Answer:** Attacker might get lucky and query E(K_i ,.), while f_i is completely independent of E(K_i,.)
- However, hybrids are indistinguishable if attacker never submits query of the form E(K_i ,.). Let BADQ be the event that the attacker submits a query to ideal cipher with key K_i for some user.

$$\Pr[\text{BADQ}] \leq pu2^{-\kappa}$$

Theorem 8. *Let $\kappa, \lambda, \nu \geq 1$ be such that $\nu \leq \lambda - 2$. Let $\mathsf{H}\colon \{0,1\}^\lambda \times (\{0,1\}^* \times \{0,1\}^*) \to \{0,1\}^\lambda$ be an ϵ-almost XOR-universal hash function, for some $\epsilon\colon \mathbb{N} \times \mathbb{N} \to [0,1]$. Let $\mathsf{CAU} = \mathbf{CAU}[\mathsf{H}, \kappa, \lambda, \nu]$. Let A be an adversary that makes at most u queries to its* New *oracle, q_e queries to its* Enc *oracle with messages of length at most ℓ_{bit} bits, q_v queries to its* Vf *oracle with messages of length at most $\ell_{\text{bit}} + \lambda$ bits, and p queries to its* E *and* E^{-1} *oracles. Assume furthermore that $q_e \leq 2^\nu$ and $\ell_{\text{bit}} \leq \lambda(2^{\lambda-\nu} - 2)$. Then*

$$\mathsf{Adv}^{\text{mu-ind}}_{\mathsf{CAU}}(A) \leq \frac{up}{2^\kappa} + \frac{u(\ell_{\text{blk}}(q_e + q_v) + 1)^2 \cdot}{2^{\lambda+1}} + \frac{u(u-1)}{2^{\kappa+1}} + uq_v \cdot \epsilon(\ell_{\text{bit}}, \ell_{\text{head}}),$$

for $\ell_{\text{blk}} = \lceil \ell_{\text{bit}}/\lambda \rceil + 1$ and where the AEAD headers are restricted to ℓ_{head} bits.

Hybrid Argument: Slowly Make Real/Ideal Oracles Identical

- **Hybrid 2:**
  - Instead of using E(K_i,.) in the encryption oracle the we replace E(K_i,.) with a fresh random permutation f_i for each user

- **Hybrid 3:**
  - Change f_i for each user to a truly random function

- Hybrid 2 is statistically indistinguishable from Hybrid 2

- At most q_v (resp. q_E) queries to encryption/decryption oracle per user
- Each query generates at most ℓ_{blk} queries to f_i per user
- Hybrid 3 and 2 are equivalent unless there is a collision in one of the queries to f_i

$$\Pr[COLLISION] \leq u\left(\ell_{blk}(q_E + q_v)\right)^2 2^{-\lambda-1}$$

Theorem 8. *Let $\kappa, \lambda, \nu \geq 1$ be such that $\nu \leq \lambda - 2$. Let $\mathsf{H}\colon \{0,1\}^\lambda \times (\{0,1\}^* \times \{0,1\}^*) \to \{0,1\}^\lambda$ be an ϵ-almost XOR-universal hash function, for some $\epsilon\colon \mathbb{N} \times \mathbb{N} \to [0,1]$. Let $\mathsf{CAU} = \mathbf{CAU}[\mathsf{H}, \kappa, \lambda, \nu]$. Let A be an adversary that makes at most u queries to its* New *oracle, q_e queries to its* Enc *oracle with messages of length at most ℓ_{bit} bits, q_v queries to its* Vf *oracle with messages of length at most $\ell_{\mathrm{bit}} + \lambda$ bits, and p queries to its E and E^{-1} oracles. Assume furthermore that $q_e \leq 2^\nu$ and $\ell_{\mathrm{bit}} \leq \lambda(2^{\lambda-\nu} - 2)$. Then*

$$\mathsf{Adv}^{\mathrm{mu\text{-}ind}}_{\mathsf{CAU}}(A) \leq \frac{up}{2^\kappa} + \frac{u(\ell_{\mathrm{blk}}(q_e + q_v) + 1)^2}{2^{\lambda+1}} + \frac{u(u-1)}{2^{\kappa+1}} + uq_v \cdot \epsilon(\ell_{\mathrm{bit}}, \ell_{\mathrm{head}}),$$

for $\ell_{\mathrm{blk}} = \lceil \ell_{\mathrm{bit}}/\lambda \rceil + 1$ and where the AEAD headers are restricted to ℓ_{head} bits.

Game G_4 $\boxed{G_5}$

$b \leftarrow\!\!\$ \{0,1\}$; $b' \leftarrow\!\!\$ A^{\text{New},\text{Enc},\text{Vf}}$

Return $(b' = b)$

New()

$v \leftarrow v+1$; $K[v] \leftarrow\!\!\$ \overline{\{K[1],\ldots,K[v-1]\}}$

Enc(i, N, M, H)

$V \leftarrow V \cup \{(i,N)\}$

$C_1 \leftarrow \mathsf{CAU.Enc}^{\mathrm{E}}(K[i], N, M, H)$

$C_0 \leftarrow\!\!\$ \{0,1\}^{\mathsf{CAU.cl}(|M|)}$

Return C_b

Vf(i, N, C, H)

If $(b = 0)$ then return false

$M \leftarrow \mathsf{CAU.Dec}^{\mathrm{E}}(K[i], N, C, H)$

If $M \neq \perp$ and $(i,N) \notin V$ then

  bad $\leftarrow$ true ; $\boxed{\text{return false}}$

Return $(M \neq \perp)$

E(K, x)

If $U[K,x] = \perp$ then

  $U[K,x] \leftarrow\!\!\$ \{0,1\}^{\lambda}$

Return $U[K,x]$

Figure 23: Between the games G_4 and G_5, we change the behavior of the Vf oracle to reject forgery attempts also for $b = 1$.

- Hybrid 4 is equivalent to Hybrid 3 (introduces bad flag)
- Hybrid 5 returns false if nonce i has not been used for user i ➔ Can view $f_i(N)$ as random λ bit string that is yet to be picked.

$$|\Pr[G4] - \Pr[G5]| \leq \frac{uq_v}{2^{\lambda}}$$

Theorem 8. *Let* $\kappa, \lambda, \nu \geq 1$ *be such that* $\nu \leq \lambda - 2$. *Let* $\mathsf{H}\colon \{0,1\}^\lambda \times (\{0,1\}^* \times \{0,1\}^*) \to \{0,1\}^\lambda$ *be an* ϵ*-almost XOR-universal hash function, for some* $\epsilon\colon \mathbb{N} \times \mathbb{N} \to [0,1]$. *Let* $\mathsf{CAU} = \mathbf{CAU}[\mathsf{H}, \kappa, \lambda, \nu]$. *Let* A *be an adversary that makes at most* u *queries to its* New *oracle,* q_e *queries to its* Enc *oracle with messages of length at most* ℓ_{bit} *bits,* q_v *queries to its* Vf *oracle with messages of length at most* $\ell_{\mathrm{bit}} + \lambda$ *bits, and* p *queries to its* E *and* E^{-1} *oracles. Assume furthermore that* $q_e \leq 2^\nu$ *and* $\ell_{\mathrm{bit}} \leq \lambda(2^{\lambda-\nu} - 2)$. *Then*

$$\mathsf{Adv}^{\mathsf{mu\text{-}ind}}_{\mathsf{CAU}}(A) \leq \frac{up}{2^\kappa} + \frac{u(\ell_{\mathrm{blk}}(q_e + q_v) + 1)^2 \cdot}{2^{\lambda+1}} + \frac{u(u-1)}{2^{\kappa+1}} + uq_v \cdot \epsilon(\ell_{\mathrm{bit}}, \ell_{\mathrm{head}}),$$

for $\ell_{\mathrm{blk}} = \lceil \ell_{\mathrm{bit}}/\lambda \rceil + 1$ *and where the AEAD headers are restricted to* ℓ_{head} *bits.*

Game G_6 $\boxed{\mathrm{G}_7}$	$\text{VF}(i, N, T \| C, H)$
$b \leftarrow\$ \{0,1\}$; $b' \leftarrow\$ A^{\text{NEW,ENC,VF}}$	If $(b = 0$ or $(i, N) \notin V)$ then return false
Return $(b' = b)$	$G \leftarrow \mathrm{E}(K[i], 0^\lambda)$; $Y \leftarrow N \| \langle 1 \rangle$
	Let C', H' such that $(i, N, C', H') \in W$
$\text{NEW}()$	$\Delta \leftarrow T \oplus \mathrm{E}(K[i], Y + 0)$
$v \leftarrow v + 1$; $K[v] \leftarrow\$ \overline{\{K[1], \ldots, K[v-1]\}}$	If $\mathsf{H}(G, H', C') \oplus \mathsf{H}(G, H, C) = \Delta$ then
	bad $\leftarrow$ true ; $\boxed{\text{return false}}$
$\text{ENC}(i, N, M, H)$	Return $\mathsf{H}(G, H', C') \oplus \mathsf{H}(G, H, C) = \Delta$
$G \leftarrow \mathrm{E}(K[i], 0^\lambda)$; $Y \leftarrow N \| \langle 1 \rangle$	
// Compute C as in $\mathsf{CAU.Enc}^{\mathrm{E}}(K[i], N, M, H)$	$\mathrm{E}(K, x)$
$C_1 \leftarrow \mathrm{E}(K[i], Y + 0) \| C$	If $U[K, x] = \bot$ then
$V \leftarrow V \cup \{(i, N)\}$; $W \leftarrow W \cup \{(i, N, C, H)\}$	$U[K, x] \leftarrow\$ \{0,1\}^\lambda$
$C_0 \leftarrow\$ \{0,1\}^{\mathsf{CAU.cl}(\|M\|)}$	Return $U[K, x]$
Return C_b	

Figure 24: Game G_6 is equivalent to G_5. The outputs of ENC are sampled differently, but VF is adapted in a consistent way.

What is Probability Attacker wins in Hybrid 7

- What is the probability attacker wins in Hybrid 7?
- Exactly ½
- Why? In hybrid 7 of all oracles is identical when b=0 and b=1.

Question:

- What is the probability of distinguishing between Hybrid 6 and 7?

- For each query to verification oracle hybrids 6 and 7 are equivalent unless we have a hash collision

$$\Pr\left[\mathsf{H}(G, H, C) \oplus \mathsf{H}(G, H', C') = T \oplus \mathrm{E}(K[i], Y + 0)\right] \leq \epsilon(\ell_{\mathrm{bit}}, \ell_{\mathrm{head}}),$$

- Union Bound over all uq_v queries

Theorem 8. *Let* $\kappa, \lambda, \nu \geq 1$ *be such that* $\nu \leq \lambda - 2$*. Let* $\mathsf{H}\colon \{0,1\}^\lambda \times (\{0,1\}^* \times \{0,1\}^*) \to \{0,1\}^\lambda$ *be an* ϵ*-almost XOR-universal hash function, for some* $\epsilon\colon \mathbb{N} \times \mathbb{N} \to [0,1]$*. Let* $\mathsf{CAU} = \mathbf{CAU}[\mathsf{H}, \kappa, \lambda, \nu]$*. Let* A *be an adversary that makes at most* u *queries to its* NEW *oracle,* q_e *queries to its* ENC *oracle with messages of length at most* ℓ_{bit} *bits,* q_v *queries to its* VF *oracle with messages of length at most* $\ell_{\text{bit}} + \lambda$ *bits, and* p *queries to its* E *and* E^{-1} *oracles. Assume furthermore that* $q_e \leq 2^\nu$ *and* $\ell_{\text{bit}} \leq \lambda(2^{\lambda-\nu} - 2)$*. Then*

$$\mathsf{Adv}^{\text{mu-ind}}_{\mathsf{CAU}}(A) \leq \frac{up}{2^\kappa} + \frac{u(\ell_{\text{blk}}(q_e + q_v) + 1)^2 \cdot}{2^{\lambda+1}} + \frac{u(u-1)}{2^{\kappa+1}} + uq_v \cdot \epsilon(\ell_{\text{bit}}, \ell_{\text{head}}),$$

for $\ell_{\text{blk}} = \lceil \ell_{\text{bit}}/\lambda \rceil + 1$ *and where the AEAD headers are restricted to* ℓ_{head} *bits.*

We combine all bounds shown in the above paragraphs:

$$\begin{aligned}
\mathsf{Adv}^{\text{mu-ind}}_{\mathsf{CAU}}(A) &= 2\Pr[\mathbf{G}^{\text{mu-ind}}_{\mathsf{CAU}}(A)] - 1 = 2\Pr[\mathrm{G}_0] - 1 \\
&\leq 2\Pr[\mathrm{G}_1] - 1 + \frac{u(u-1)}{2^{\kappa+1}} \\
&\leq 2\Pr[\mathrm{G}_3] - 1 + \frac{u(u-1)}{2^{\kappa+1}} + \frac{u((q_e+q_v)\cdot \ell_{\text{blk}})^2}{2^{\lambda+1}} \\
&\leq 2\Pr[\mathrm{G}_5] - 1 + \frac{u(u-1)}{2^{\kappa+1}} + \frac{u((q_e+q_v)\cdot \ell_{\text{blk}})^2}{2^{\lambda+1}} + uq_v \cdot 2^{-\lambda} \\
&\leq 2\Pr[\mathrm{G}_7] - 1 + \frac{u(u-1)}{2^{\kappa+1}} + \frac{u((q_e+q_v)\cdot \ell_{\text{blk}})^2}{2^{\lambda+1}} + uq_v \cdot (2^{-\lambda} + \epsilon(\ell_{\text{bit}}, \ell_{\text{head}})) \,,
\end{aligned}$$

which concludes the proof. ▮

McGrew and Viega [24, Lemma 2] show that H has this property for $\epsilon(m,n) = (\lceil m/\lambda \rceil + \lceil n/\lambda \rceil + 1)/2^{\lambda}$.

Multi-User Security Game for AEAD

- Challenger picks a random bit b and Generates u independent keys $K_1, \ldots, Ku$
  - **Real Mode: b=1**
  - **Ideal Mode: b=0**
- **Attacker Goal: guess b**

- **Attacker Oracles:**
  - **Ideal Cipher**
  - **Encryption oracle** (Takes as input an individual $i \leq u$, nonce N, message M, header H) :
    - Outputs: "Invalid" if pair (i,N) is repeated (Attacker not allowed to repeat nonce for individual user)
    - **Real Mode:** Encrypts message using key K_i and outputs ciphertext
    - **Ideal Mode:** Returns random string instead of ciphertext
  - **Verification Oracle:** (Takes as input individual $i \leq u$, nonce N, ciphertext M, header H):
    - Outputs 1 if this ciphertext was generated via a query to the encryption oracle with same user/nonce/header; otherwise
    - **Ideal Mode:** Output 0
    - **Real Mode:** Attempt to decrypt using key K_i; output 0 if decryption fails and 1 otherwise

Source: Bellare, Tackmann, Multi-User Security of Authenticated Encryption: AES-GCM in TLS 1.3

Reminder: Last Class

Theorem 8. *Let* $\kappa, \lambda, \nu \geq 1$ *be such that* $\nu \leq \lambda - 2$. *Let* $\mathsf{H}\colon \{0,1\}^\lambda \times (\{0,1\}^* \times \{0,1\}^*) \to \{0,1\}^\lambda$ *be an* ϵ*-almost XOR-universal hash function, for some* $\epsilon\colon \mathbb{N} \times \mathbb{N} \to [0,1]$. *Let* $\mathsf{CAU} = \mathbf{CAU}[\mathsf{H}, \kappa, \lambda, \nu]$. *Let* A *be an adversary that makes at most* u *queries to its* New *oracle,* q_e *queries to its* Enc *oracle with messages of length at most* ℓ_{bit} *bits,* q_v *queries to its* Vf *oracle with messages of length at most* $\ell_{\mathrm{bit}} + \lambda$ *bits, and* p *queries to its* E *and* E^{-1} *oracles. Assume furthermore that* $q_e \leq 2^\nu$ *and* $\ell_{\mathrm{bit}} \leq \lambda(2^{\lambda-\nu} - 2)$. *Then*

$$\mathsf{Adv}^{\mathrm{mu\text{-}ind}}_{\mathsf{CAU}}(A) \leq \frac{up}{2^\kappa} + \frac{u(\ell_{\mathrm{blk}}(q_e + q_v) + 1)^2}{2^{\lambda+1}} + \frac{u(u-1)}{2^{\kappa+1}} + uq_v \cdot \epsilon(\ell_{\mathrm{bit}}, \ell_{\mathrm{head}}),$$

for $\ell_{\mathrm{blk}} = \lceil \ell_{\mathrm{bit}}/\lambda \rceil + 1$ *and where the AEAD headers are restricted to* ℓ_{head} *bits.*

GHASH in AES-GCM

$$\mathrm{GHASH}(H, A, C) = X_{t+1}$$

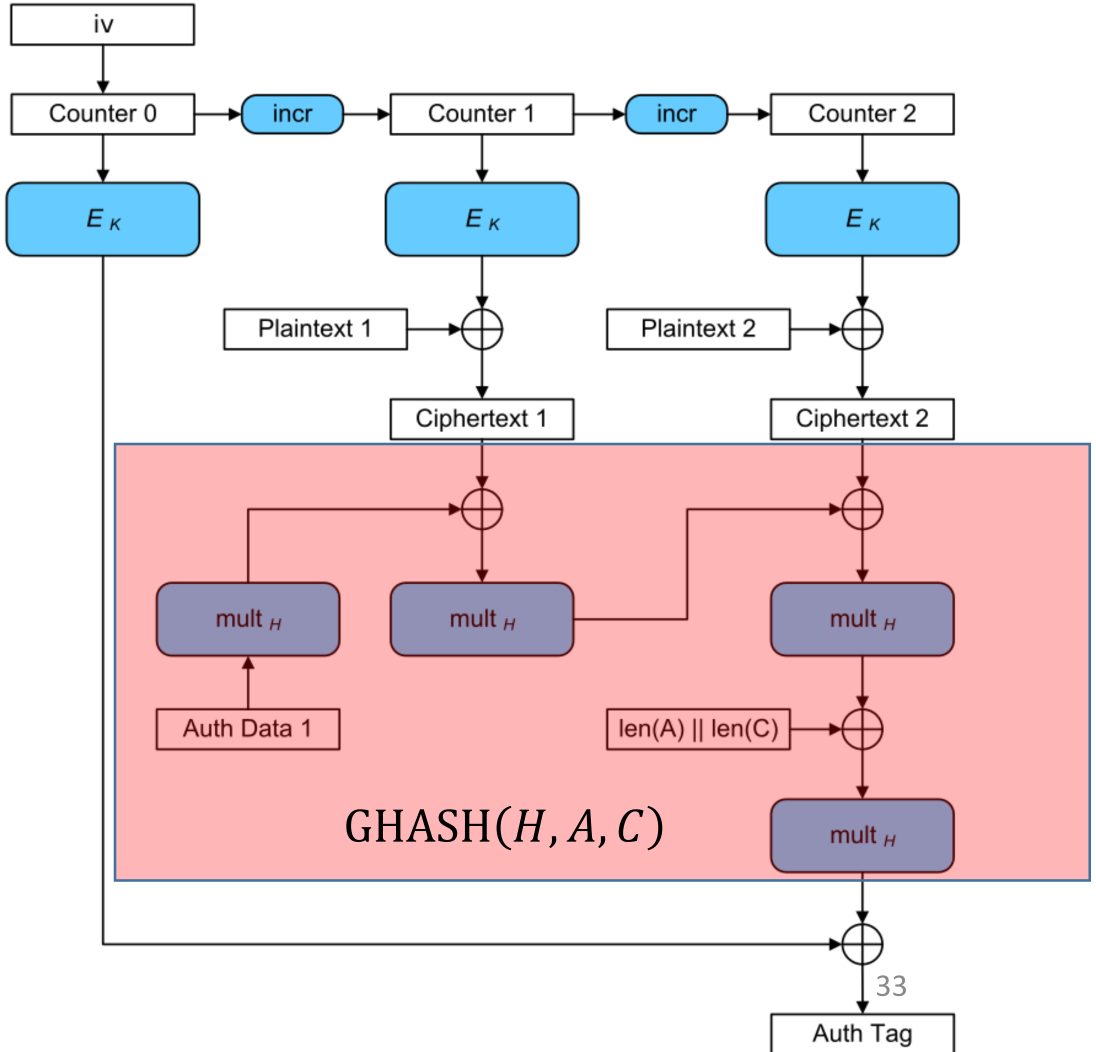
Where

- $X_0 = 0,$
- $(S_1, \ldots, S_t) = A \circ C \circ len(A) \circ len(C)$ and

$$X_i = (S_i \oplus X_{i-1}) \cdot H$$

AES-GCM: $H = \mathrm{E_K}(0^\lambda)$ (secret value)

Authentication Tag: $\mathbf{E_K(N) \oplus GHASH}(\boldsymbol{H}, \boldsymbol{A}, \boldsymbol{C})$



GHASH in AES-GCM

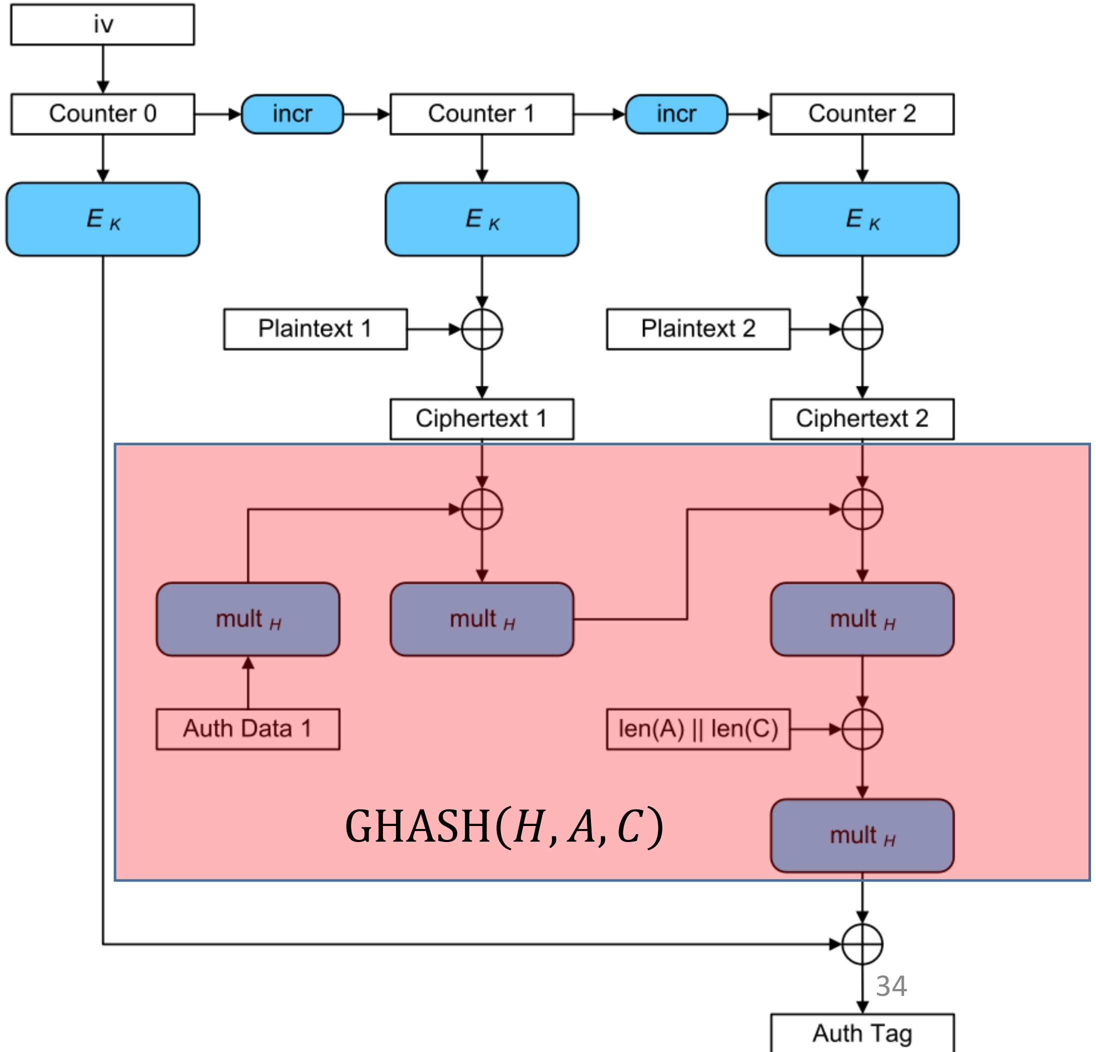
$$\mathrm{GHASH}(H, A, C) = X_{t+1}$$

Where

- $X_0 = 0,$
- $(S_1, \ldots, S_t) = A \circ C \circ len(A) \circ len(C)$ and

$$X_i = (S_i \oplus X_{i-1}) \cdot H$$

$$X_{t+1} = \sum_{i \leq t} S_i \cdot H^{t-i+1}$$



Back to the Nonces

- Prior Security Analysis Assumes no Nonce Collisions
- If nonces are randomized in $\{0,1\}^{\lambda}$ we need to add a term
- $\boldsymbol{2^{-\lambda}\sum_{i<j\le q_e}(bi+bj+1)} \le \boldsymbol{2^{-\lambda}\binom{q_e}{2}}(2\ell_{blk}+1)$

Back to the Nonces: AES GCM

- In AES-GCM $\boldsymbol{\lambda = 128}$, but the nonce is typically 96-bits

$$Counter0 = N \circ 0^{31} \circ 1$$

Constraint: plaintext/associated is at most $2^{32} - 1$ blocks long
➔If all nonces are unique then all counters are unique

$$\mathbf{Pr[Exists\ Nonce\ Collision]} \leq \mathbf{2^{-96}\binom{q_e}{2} = 2^{-96}\binom{q_e}{2}}$$

$$\sout{2^{-\lambda} \sum_{i<j\leq q_e} (bi + bj + 1) \leq 2^{-\lambda} \binom{q_e}{2} (2\ell_{blk} + 1)}$$

Back to the Nonces: AES GCM

- In AES-GCM $\boldsymbol{\lambda = 128}$, but the nonce is typically 96-bits

$$Counter0 = N \circ 0^{31} \circ 1$$

Constraint: plaintext/associated is at most $2^{32} - 1$ blocks long

➔If all nonces are unique then all counters are unique

$$\mathbf{Pr[Exists\ Nonce\ Collision]} \leq \mathbf{2^{-96}} \binom{\boldsymbol{q_e}}{\mathbf{2}} = \mathbf{2^{-96}} \binom{\boldsymbol{q_e}}{\mathbf{2}}$$

Practice: Pick fresh key once $\boldsymbol{q_e} = 2^{32}$

Nonce-Misuse Resistance

- Recall Encryption Scheme $Enc(K, m) = \langle r, F_k(r) \oplus m \rangle$

- If attacker intercepts two ciphertexts with repeated nonce

$c = \langle r, s = F_k(r) \oplus m \rangle$ and $c' = \langle r, s' = F_k(r) \oplus m' \rangle$

Attacker can obtain $s \oplus s' = m \oplus m'$ which often reveals both m and m'

AES-GCM suffers similar weaknesses

Nonce-Misuse Resistance

Generally, for any encryption scheme Enc(K,N,m) if the nonces are repeated for messages m and m' then the attacker will learn whether or not $m = m'$ (Assume that N is the only randomness)

Ideally this is the <u>only</u> thing the attacker should learn!

Game $\mathbf{G}^{\mathsf{mu\text{-}mrae}}_{\mathsf{AE},\mathsf{KeyGen},\Pi}(\mathcal{A})$

$\mathsf{st}_0 \leftarrow \varepsilon;\ v \leftarrow 0;\ b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\text{New},\text{Enc},\text{Vf},\text{Prim}}$
Return $(b' = b)$

$\text{Vf}(i, N, C, A)$

If $i \notin \{1, \ldots, v\}$ then return $\perp$
If $(i, N, C, A) \in V[i]$ then return true
If $b = 0$ then return false
$M \leftarrow \mathsf{AE.D}^{\text{Prim}}(K_i, N, C, A)$
Return $(M \neq \perp)$

$\text{New}(\mathsf{aux})$

$v \leftarrow v + 1$
$(K_v, \mathsf{st}_v) \leftarrow\!\!\$\ \mathsf{KeyGen}(\mathsf{st}_{v-1}, \mathsf{aux})$

$\text{Enc}(i, N, M, A)$

If $i \notin \{1, \ldots, v\}$ then return $\perp$
If $(i, N, M, A) \in U[i]$ then return $\perp$
$C_1 \leftarrow \mathsf{AE.E}^{\text{Prim}}(K_i, N, M, A)$
$C_0 \leftarrow\!\!\$\ \{0,1\}^{|C_1|}$
$U[i] \leftarrow U[i] \cup \{(i, N, M, A)\}$
$V[i] \leftarrow V[i] \cup \{(i, N, C_b, A)\}$
Return C_b

Attacker is allowed to repeat nonce N for same user i as long as the message M (or authentication headers A) are different.

Game $\mathbf{G}^{\mathsf{mu\text{-}mrae}}_{\mathsf{AE},\mathsf{KeyGen},\Pi}(\mathcal{A})$

$\mathsf{st}_0 \leftarrow \varepsilon;\ v \leftarrow 0;\ b \leftarrow\!\!\$\ \{0,1\}$

$b' \leftarrow\!\!\$\ \mathcal{A}^{\text{New},\text{Enc},\text{Vf},\text{Prim}}$

Return $(b' = b)$

$\text{Vf}(i, N, C, A)$

If $i \notin \{1, \ldots, v\}$ then return $\bot$

If $(i, N, C, A) \in V[i]$ then return true

If $b = 0$ then return false

$M \leftarrow \mathsf{AE.D}^{\text{Prim}}(K_i, N, C, A)$

Return $(M \neq \bot)$

$\text{New}(\mathsf{aux})$

$v \leftarrow v + 1$

$(K_v, \mathsf{st}_v) \leftarrow\!\!\$\ \mathsf{KeyGen}(\mathsf{st}_{v-1}, \mathsf{aux})$

$\text{Enc}(i, N, M, A)$

If $i \notin \{1, \ldots, v\}$ then return $\bot$

If $(i, N, M, A) \in U[i]$ then return $\bot$

$C_1 \leftarrow \mathsf{AE.E}^{\text{Prim}}(K_i, N, M, A)$

$C_0 \leftarrow\!\!\$\ \{0,1\}^{|C_1|}$

$U[i] \leftarrow U[i] \cup \{(i, N, M, A)\}$

$V[i] \leftarrow V[i] \cup \{(i, N, C_b, A)\}$

Return C_b

Attacker is allowed to repeat nonce N for same user i as long as the message M (or authentication headers A) are different.

Generic Attack

- Fix nonce N, message |M|> $\kappa + 4$ and associated data A.
- Attacker queries C_i=Enc(i,N,M,A) for q different users.
- Output 1 If we find a collision C_j=C_i ; otherwise 0;

- Analysis:
  - **Real World:** two users will have the same key with probability at least $\frac{q(q-1)}{2^{\kappa+2}}$
  - **Ideal World:** two users will have the same ciphertext with probability at most $\frac{q(q-1)}{2^{|M|+1}} \leq \frac{q(q-1)}{2^{\kappa+5}}$
  - Advantage: at least $\frac{q(q-1)}{2^{\kappa+2}} - \frac{q(q-1)}{2^{\kappa+5}} > \frac{q(q-1)}{2^{\kappa+3}}$

AES-GCM-SIV

- Key Ideas:
  - Pick two keys K_1 and K_2
  - Final authentication TAG derived using K_2 based on nonce and hash T which in turn derived from A, M and K_1
  - $Counter_0$ is derived from TAG
  - **Note:** If we repeat the same nonce, but message M and or authentication data A changes then so will the counter $Counter_i$

GCM-SIV$^+$ (encryption-keylength, K1, K2, N, AAD, MSG)

```
1.    Context: encryption-keylength (= 128 or 256)
               0 <=  m <= 32 such that MSG length is at most 2^m-1 blocks.
2.    Keys: K1 (128 bits), K2 (128 or 256 bits)
3.    If encryption-keylength = 128, AES = AES128, else AES = AES256
4.    Input: AAD, MSG, N (96 bits)
5.    Padding:
6.      A = Zero pad AAD to the next 16 bytes boundary (d blocks)
7.      M = Zero pad MSG to the next 16 bytes boundary (v blocks)
8.        (denote M by blocks as: M0, M1, ..., M(v-1).)
9.    Encrypting and Authenticating:
10.     L1 = (bytelen(AAD)*8); L2 = (bytelen(MSG)*8)
11.     LENBLK = IntToString64(L1) || IntToString64(L2)
12*.    T = POLYVAL (K1, A || M || LENBLK)
13.     TAG = AES (K2, 0 || (T XOR N) [126:0])
14.     for i = 0, 1, ..., v-1 do
15*.      Low32(i)  = (StringToInt32(TAG[31:0]) + i) mod 2^{32}
16*.      CTRBLK_i = 1 || TAG[126:32] || IntToString32(Low32(i))
17.       CTi = AES (K2, CTRBLK_i) XOR Mi
18.     end do
19.     Set C = CT0, CT1, ..., CT(v-1)
20.     if length(MSG) != length(CT)
21.       Chop off lsbytes of CT(v-1) to make lengths equal
22.   Output: C = (CT0, CT1, ..., CT(v-1)), TAG

------------GCM-SIV-------------
12*.     GCM-SIV used GHASH instead of POLYVAL
15-16*. GCM-SIV set CTRBLK_i = 1 || TAG[126:k] || IntToString32(i)
--------------------------------
```

Fig. 1. Specification of GCM-SIV$^+$. The differences between GCM-SIV$^+$and GCM-SIV are in Steps 12*, 15* and 16*.

Security Bounds

$$\mathsf{Adv}^{\text{mu-mrae}}_{\mathsf{AE},\mathsf{KeyGen},E}(\mathcal{A}) \leq \frac{1}{2^{n/2}} + \frac{\beta a p}{2^k} + \frac{(3\beta c + 7\beta)L^2 + 4\beta c L p}{2^{n+k}} + \frac{(4c\beta + 0.5\beta + 6.5)LB}{2^n} + \frac{dp + (2d + a)L}{2^k},$$

n – blocksize; k – key length; B – blocks encrypted per user,

$\beta, c, a = O(1)$ are constants

d – upper bound on the number of users re-using a given nonce

$p < 2^{(0.9)n}$ (num queries to ideal cipher)

$L < 2^{(0.9)n}$ (total #block encrypted)

Revisiting AES-GCM-SIV: Multi-user Security, Faster Key Derivation and Better Bounds [BHT18]

Nonce Multi-Collisions (d)

- Suppose we sample q nonces $N_1, \ldots, N_q \leq 2^{\lambda}$. What is the probability that some nonce N appears d time?

$$\Pr[\text{exists d collision}] \leq \binom{q}{d} 2^{-(d-1)\lambda} \leq q^d 2^{-(d-1)\lambda}$$

If $q < 2^{\lambda(1-\varepsilon)}$ and $d = \frac{2}{\varepsilon}$ then

$$\Pr[\text{exists d collision}] \leq 2^{\lambda(1-\varepsilon)d} 2^{-(d-1)\lambda} = 2^{\lambda(1-\varepsilon d)} = 2^{\lambda}$$

Point: We can safely assume d is a small constant.

Partitioning Oracle Attacks

Julia Len Paul Grubbs Thomas Ristenpart

Cornell Tech

USENIX Security 2021

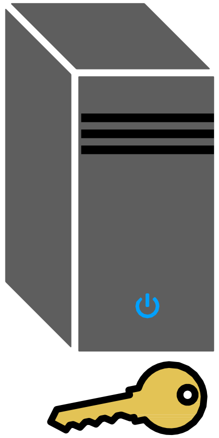
Authenticated Encryption

For simplicity, we ignore associated data in this presentation

Nonce N

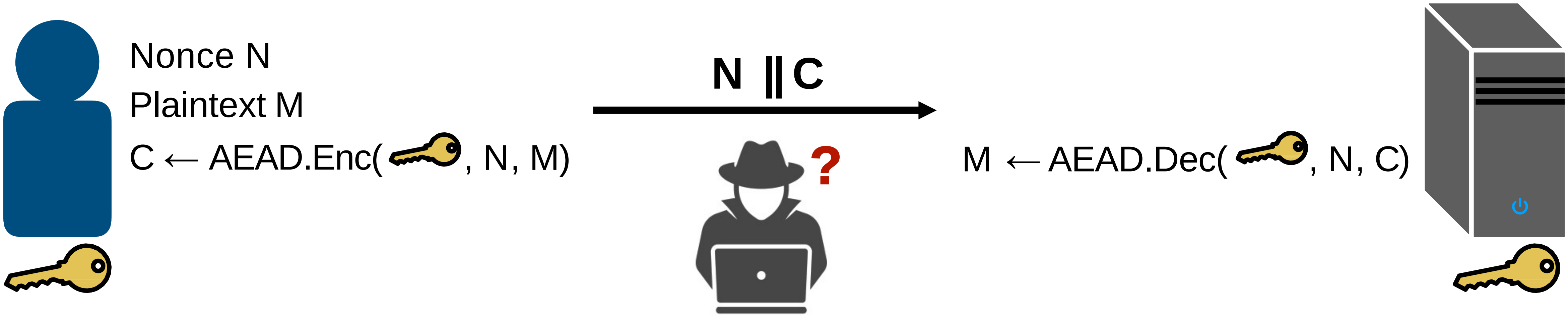
Plaintext M

C ← AEAD.Enc(, N, M)



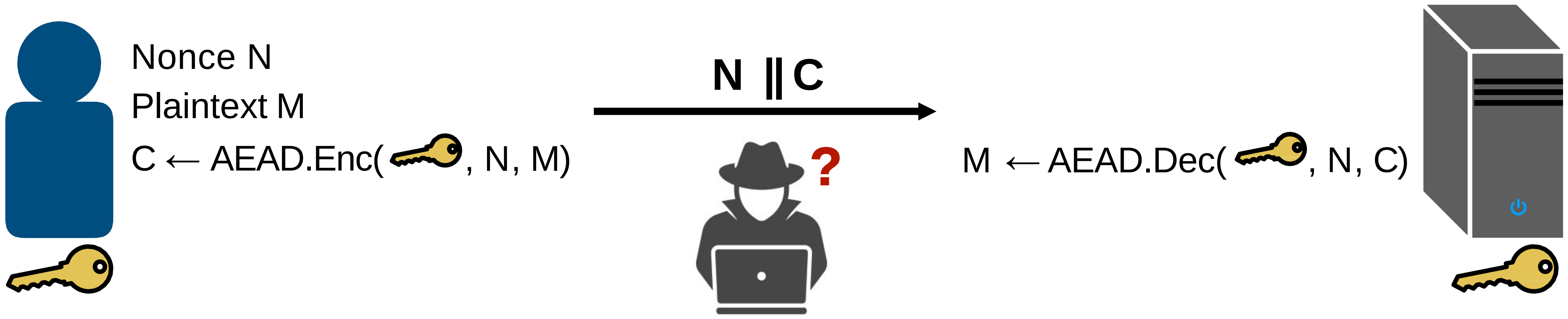
Authenticated Encryption

For simplicity, we ignore associated data in this presentation



Authenticated Encryption

For simplicity, we ignore associated data in this presentation



Popular

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

Easy to use

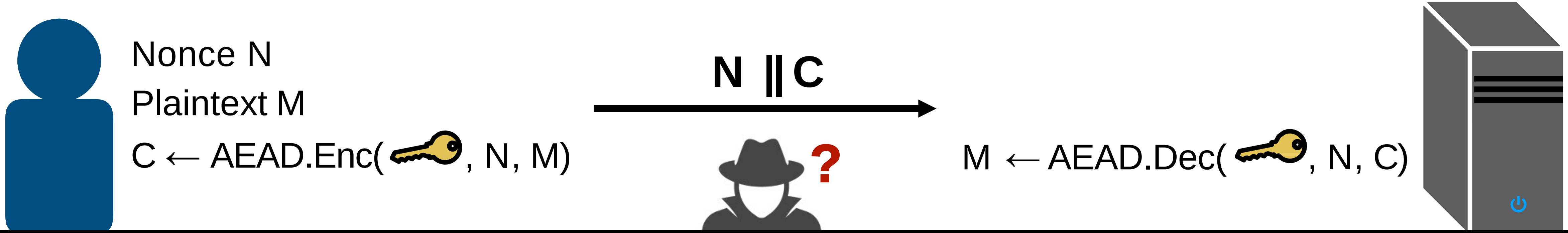
- Efficient
- Standardized
- Widely supported

Secure

- Proven CCA-secure
- Confidentiality
- Integrity

Authenticated Encryption

For simplicity, we ignore 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

- Efficient
- Standardized
- Widely supported

- Proven CCA-secure
- Confidentiality
- Integrity

(Non-) Committing AEAD

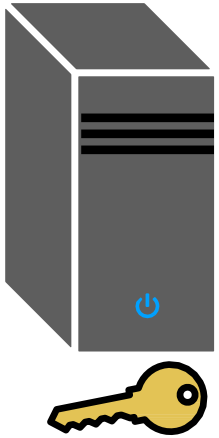
For simplicity, we ignore associated data in this presentation

Nonce N’

Ciphertext C’

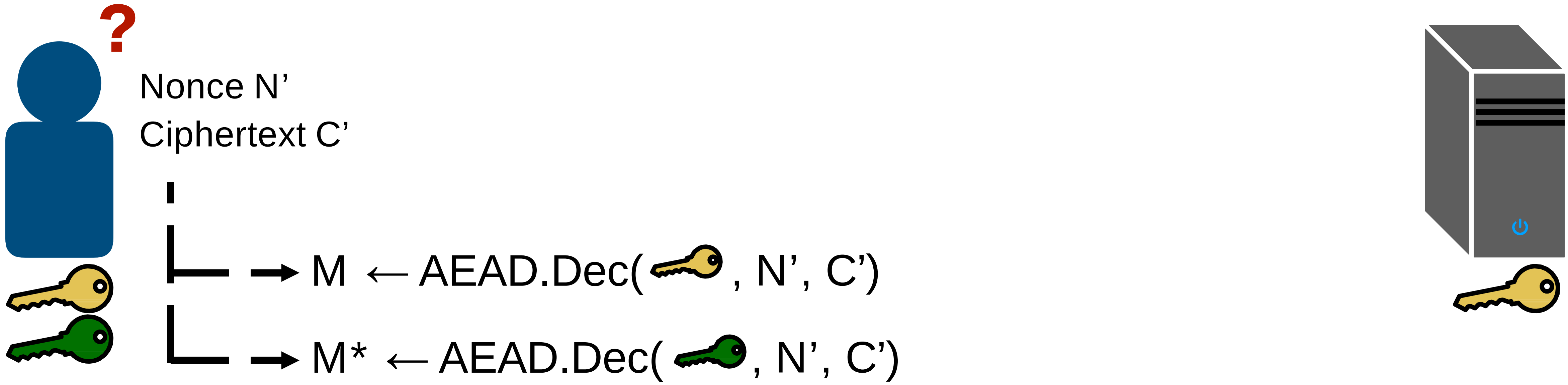
$M \leftarrow \text{AEAD.Dec}(\;, N', C')$

$M^* \leftarrow \text{AEAD.Dec}(\;, N', C')$



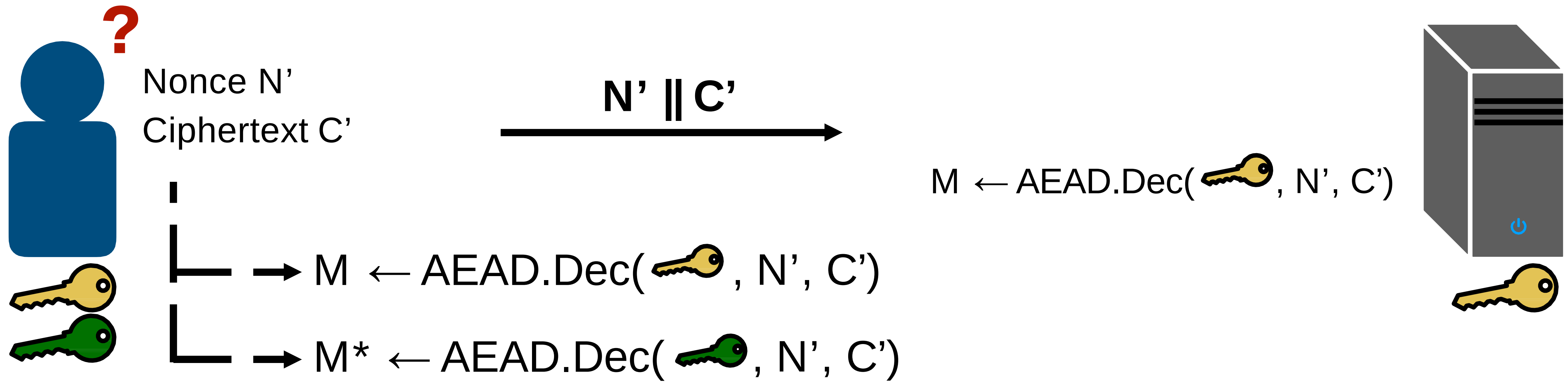
(Non-) Committing AEAD

For simplicity, we ignore associated data in this presentation



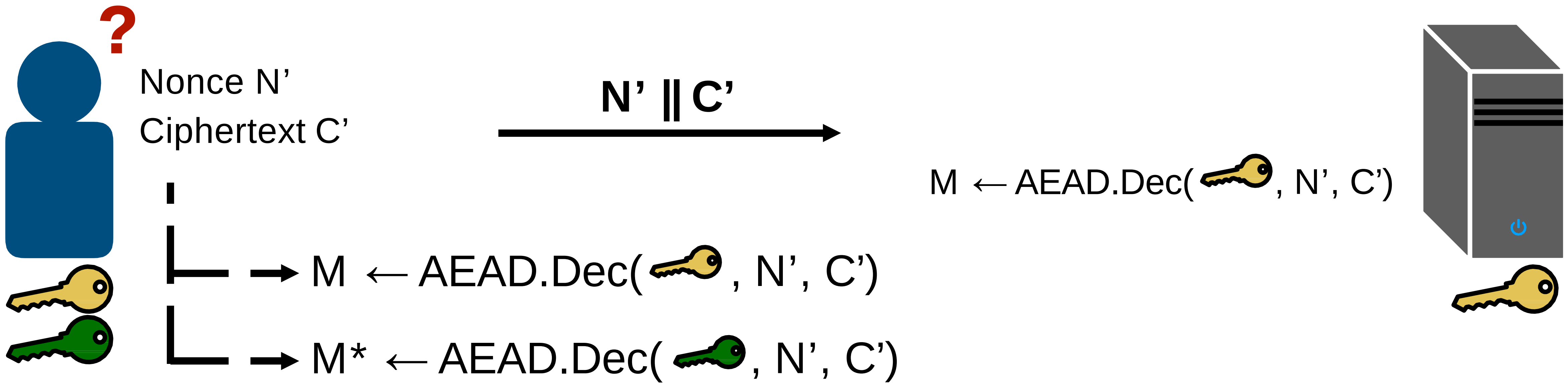
(Non-) Committing AEAD

For simplicity, we ignore associated data in this presentation



(Non-) Committing AEAD

For simplicity, we ignore associated data in this presentation



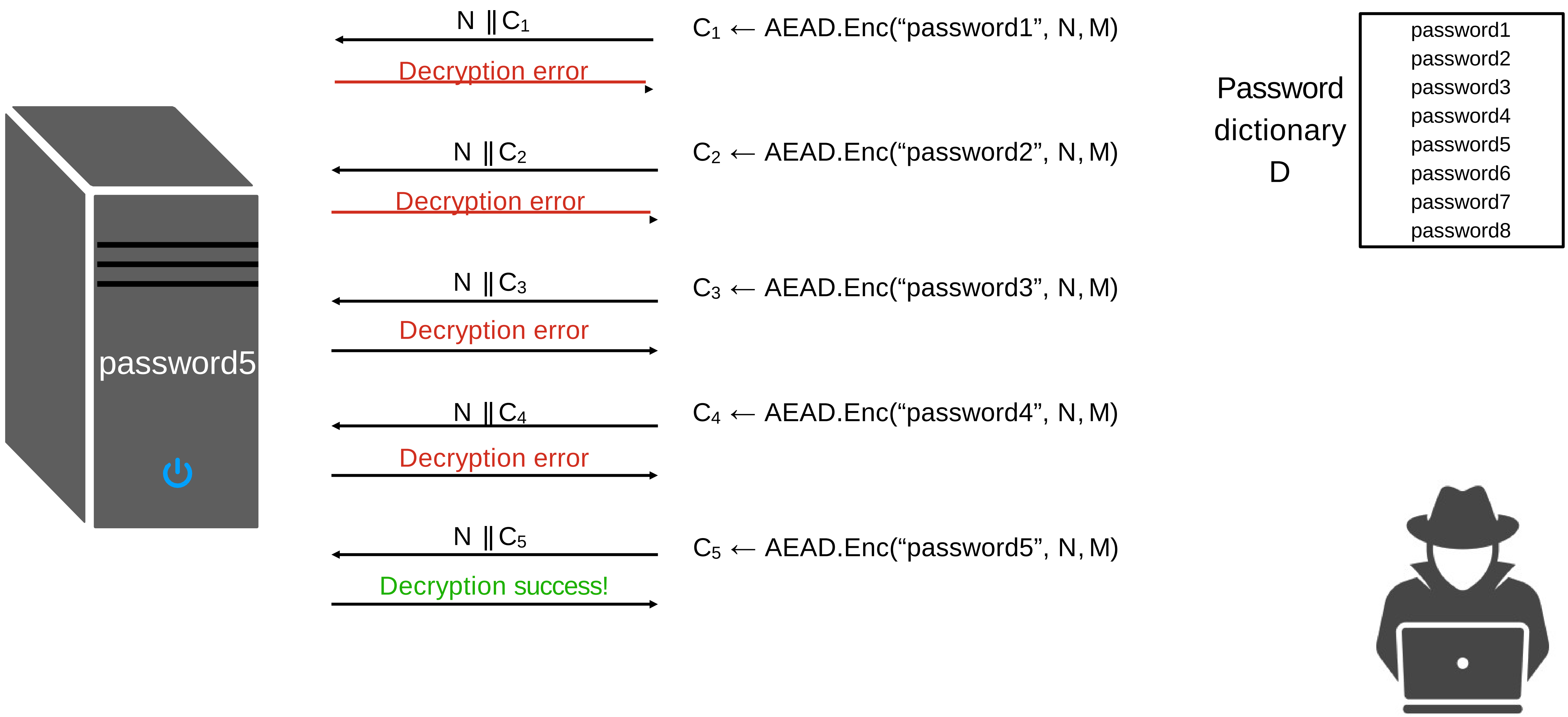
No guarantee the sender actually knows the *exact* key the recipient will use to decrypt!

Not considered an essential security goal, except in moderation settings [GLR CRYPTO’17], [DGRW CRYPTO’18]

Password dictionary D

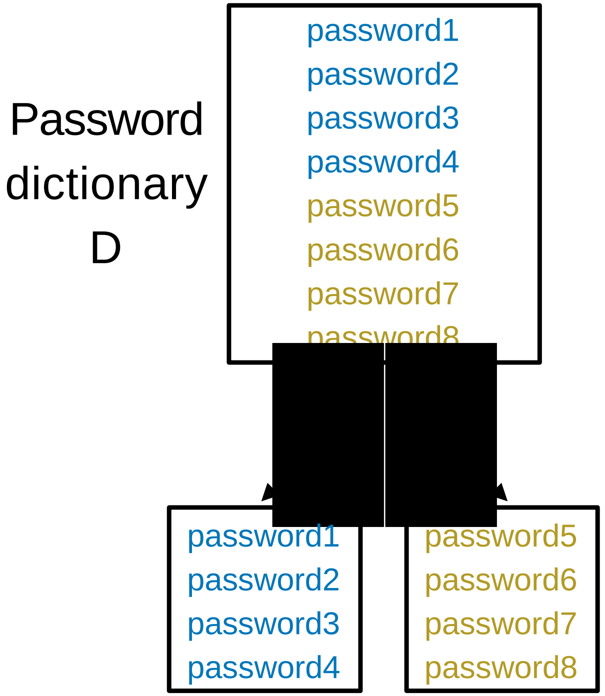
password1
password2
password3
password4
password5
password6
password7
password8

Brute-force Dictionary Attack



Partitioning Oracle Attack

A high level overview of our attack



Partitioning Oracle Attack

A high level overview of our attack

Password dictionary D

password1
password2
password3
password4
password5
password6
password7
[redacted]

password1
password2
password3
password4

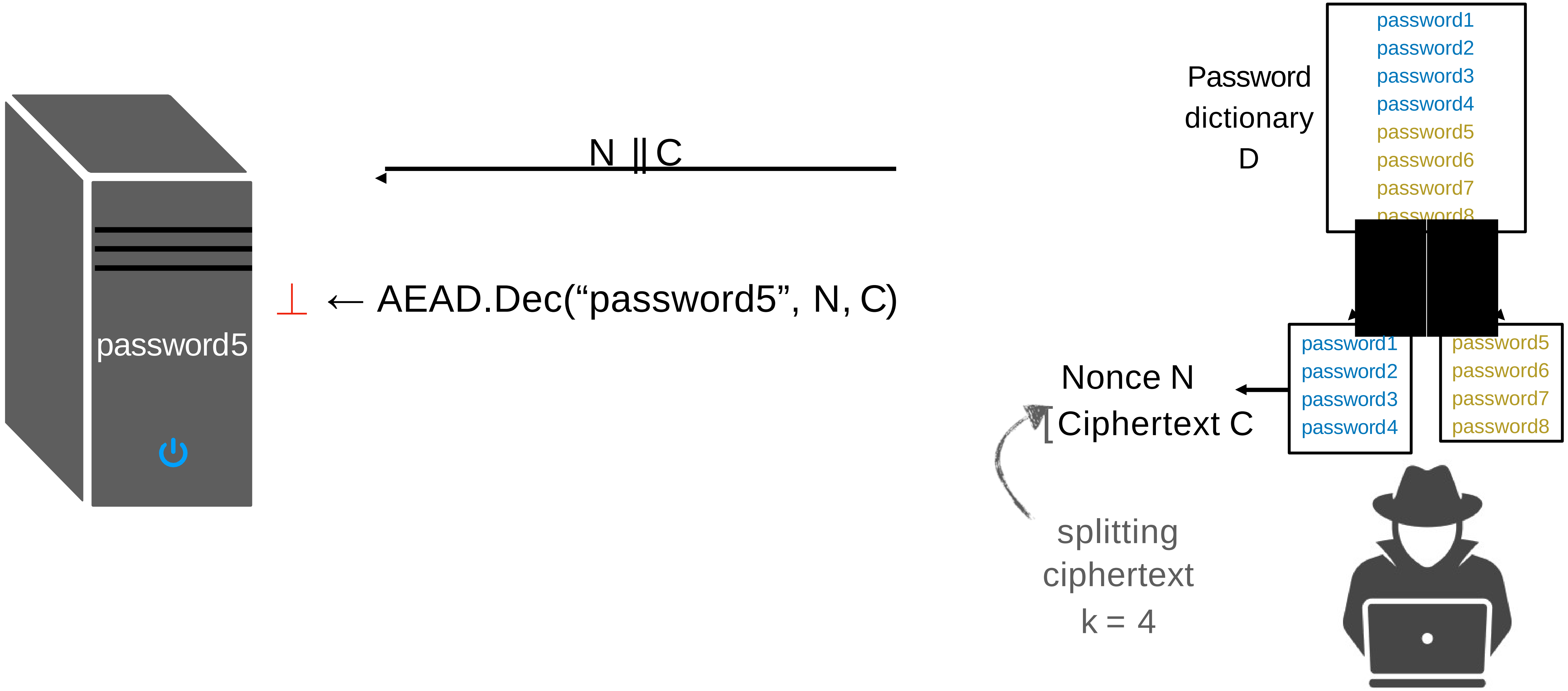
password5
password6
password7
password8

Nonce N
Ciphertext C

splitting ciphertext
k = 4

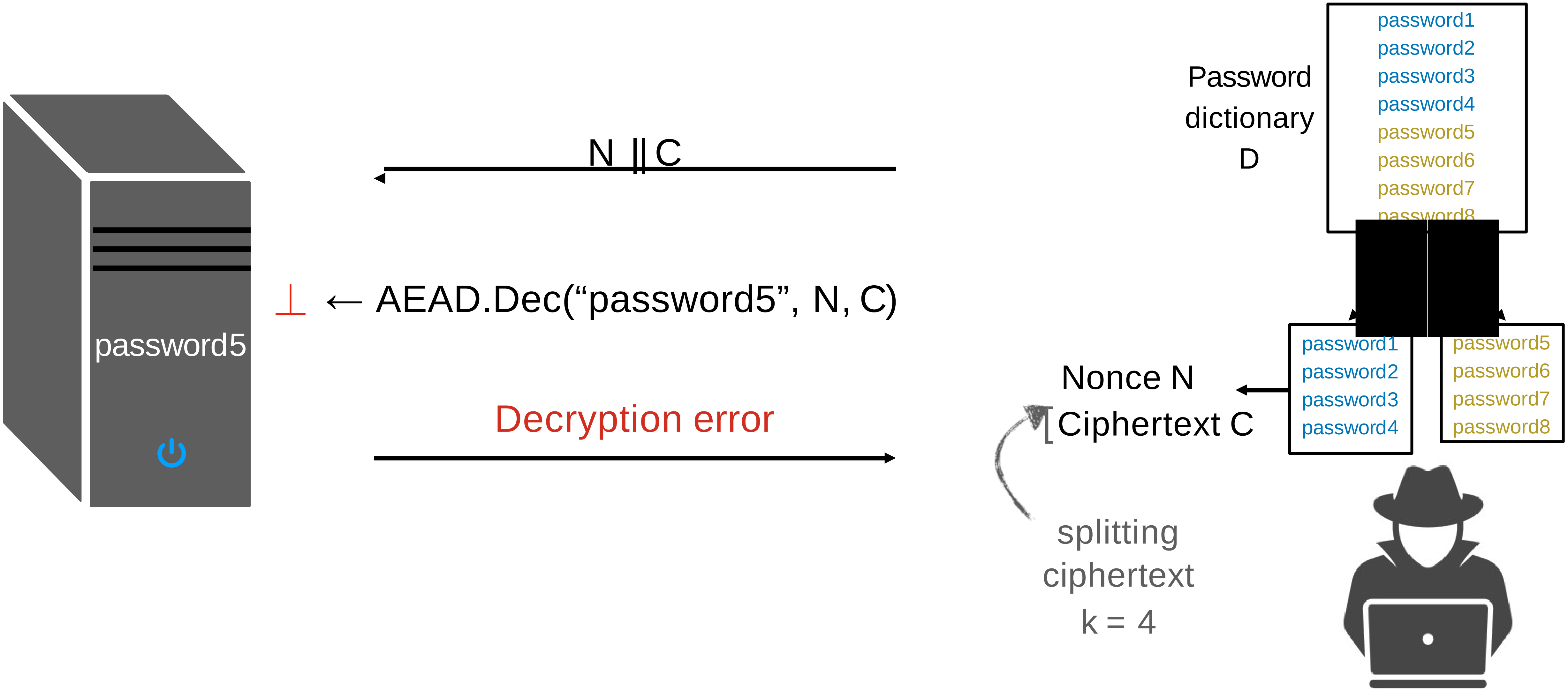
Partitioning Oracle Attack

A high level overview of our attack



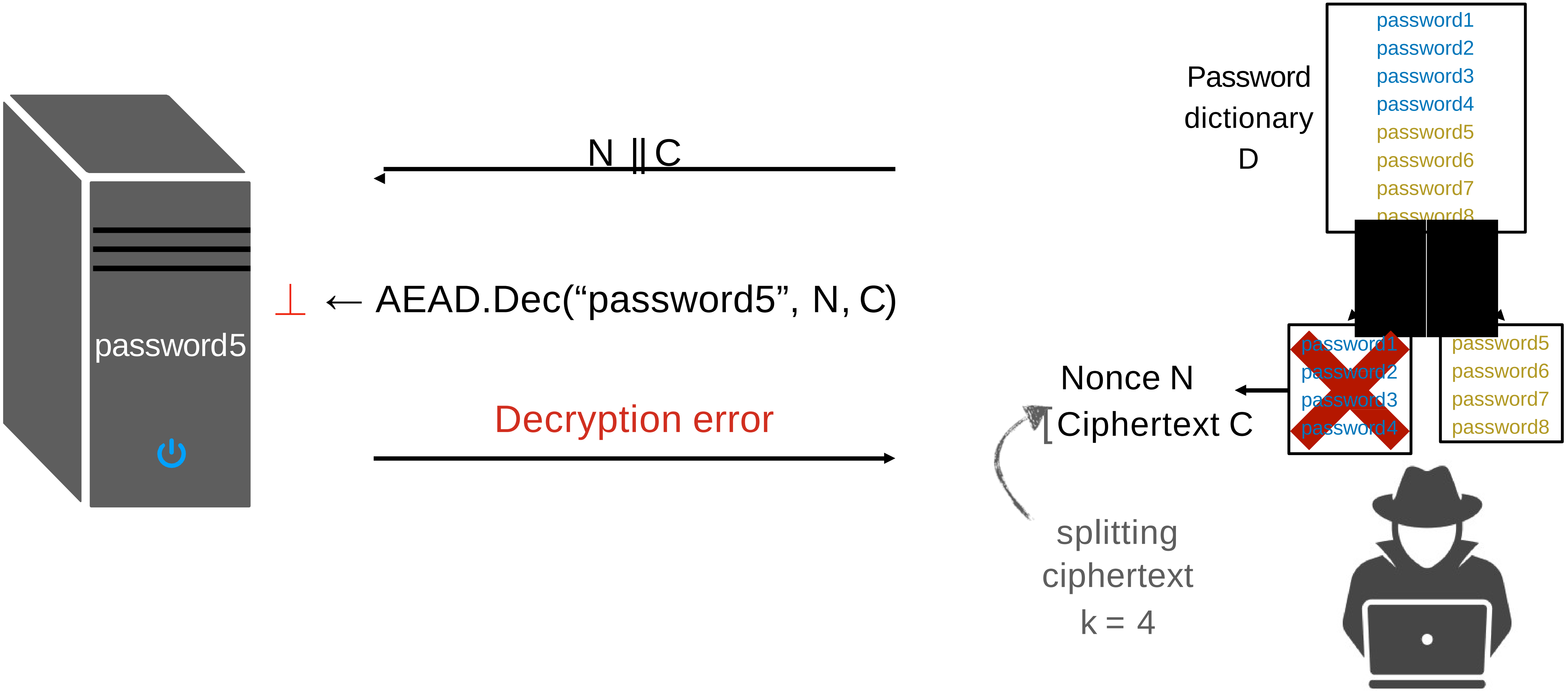
Partitioning Oracle Attack

A high level overview of our attack



Partitioning Oracle Attack

A high level overview of our attack



Partitioning Oracle Attack

D → D_1 → D_2 → D_3 → …

k = 1

k = 1

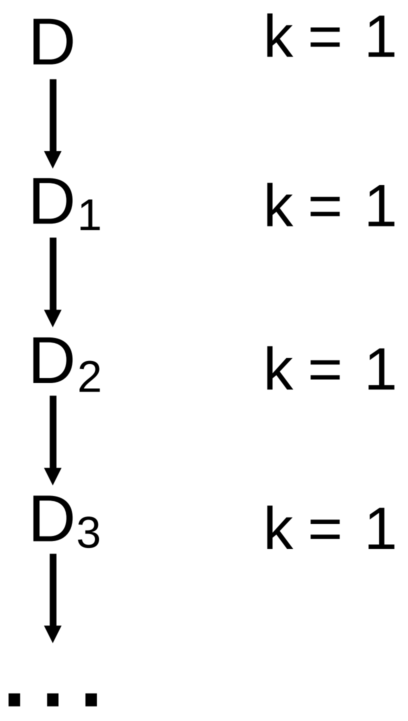
k = 1

k = 1

Brute-force dictionary attack

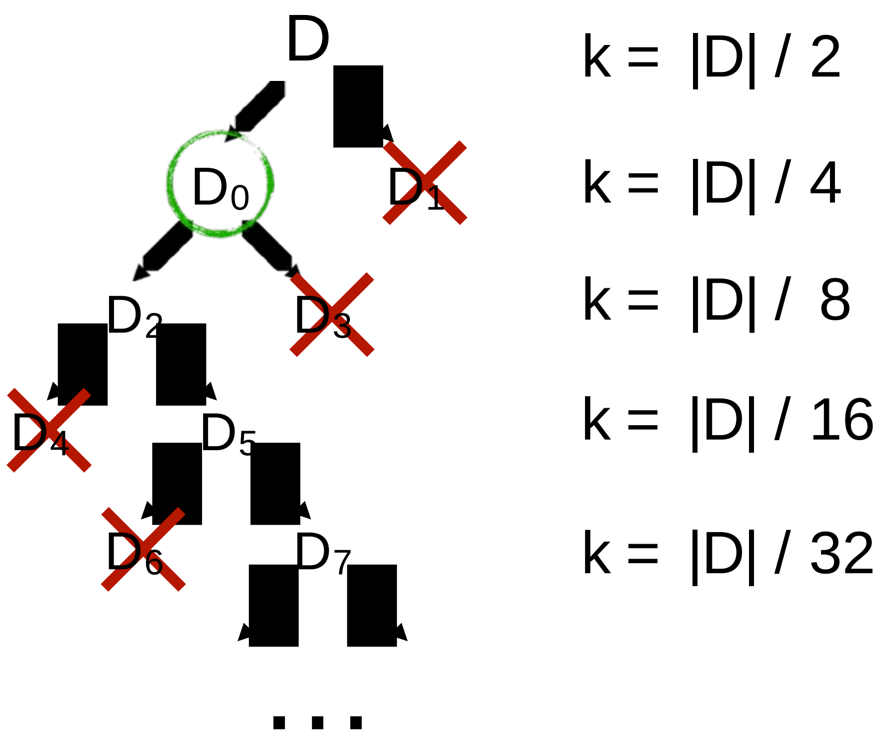
Requires $\mathcal{O}(|D|)$ queries to learn the password

Partitioning Oracle Attack



Brute-force dictionary attack

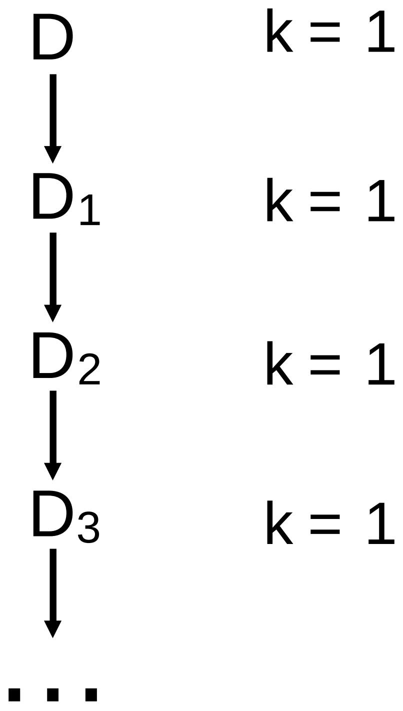
Requires $\mathcal{O}(|D|)$ queries to learn
the password



Requires $\mathcal{O}(\log |D|)$ queries to learn the password

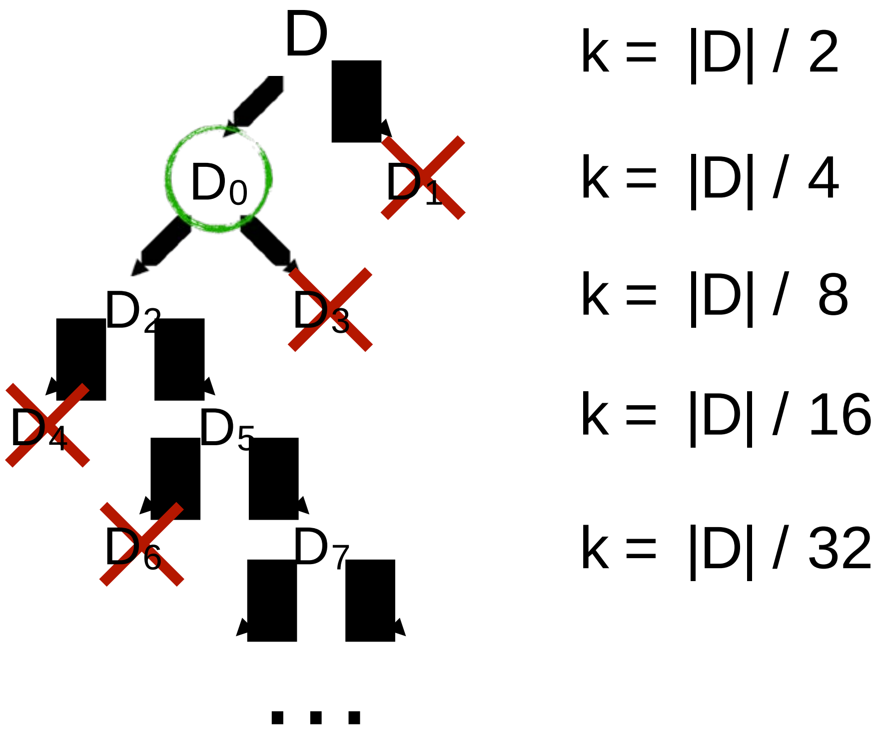
Exponential speedup over brute-force dictionary attack!

Partitioning Oracle Attack



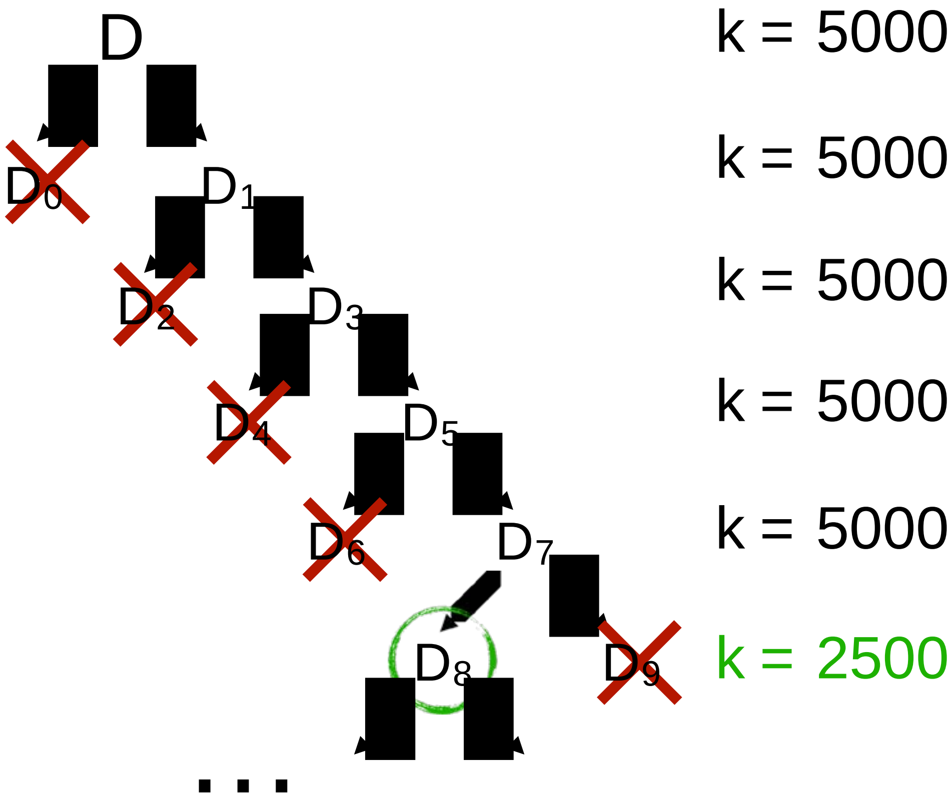
Brute-force dictionary attack

Requires $\mathcal{O}(|D|)$ queries to learn the password



Requires $\mathcal{O}(\log |D|)$ queries to learn the password

Exponential speedup over brute-force dictionary attack!



|D| is large so a more realistic case is k = 5000

This still offers a ***good speedup*** over brute-force

Partitioning oracle attacks rely on:

1. Building splitting ciphertexts that can decrypt under $k > 1$ different keys

2. Access to a partitioning oracle

Partitioning oracle attacks rely on:

1. Building splitting ciphertexts that can decrypt under k > 1 different keys

   Key Multi-collision Attacks

   [GLR CRYPTO'17] first showed an attack against AES-GCM for k = 2

2. Access to a partitioning oracle

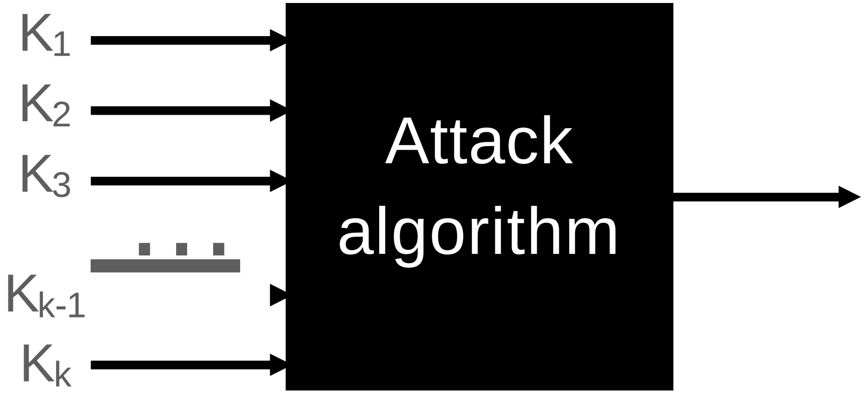
Computing Key Multi-Collisions: AES-GCM

Encrypt - then - MAC

Counter mode encryption of AES

GHASH: polynomial MAC

Run time: $\mathcal{O}(k^2)$



Ciphertext that decrypts under all k keys

<u>Length</u>: k 16-byte blocks

Computing Key Multi-Collisions: AES-GCM

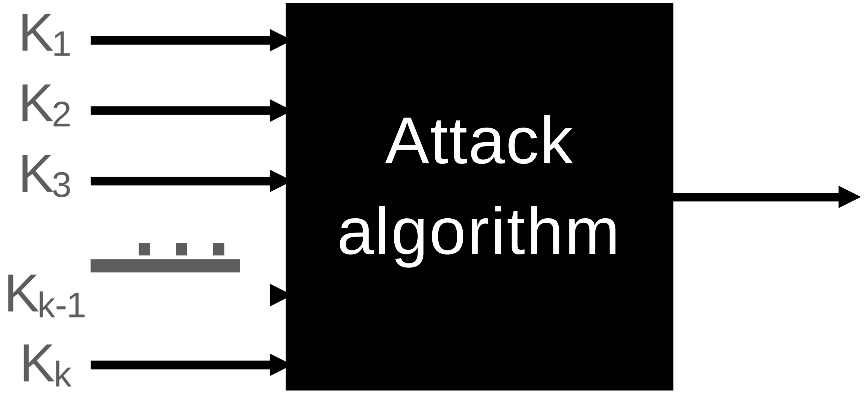
Encrypt - then - MAC

Encrypt → Counter mode encryption of AES

MAC → ~~GHASH:~~ polynomial MAC

Run time: $\mathcal{O}(k^2)$

Reduces finding ciphertext to solving set of linear equations



Ciphertext that decrypts under all k keys

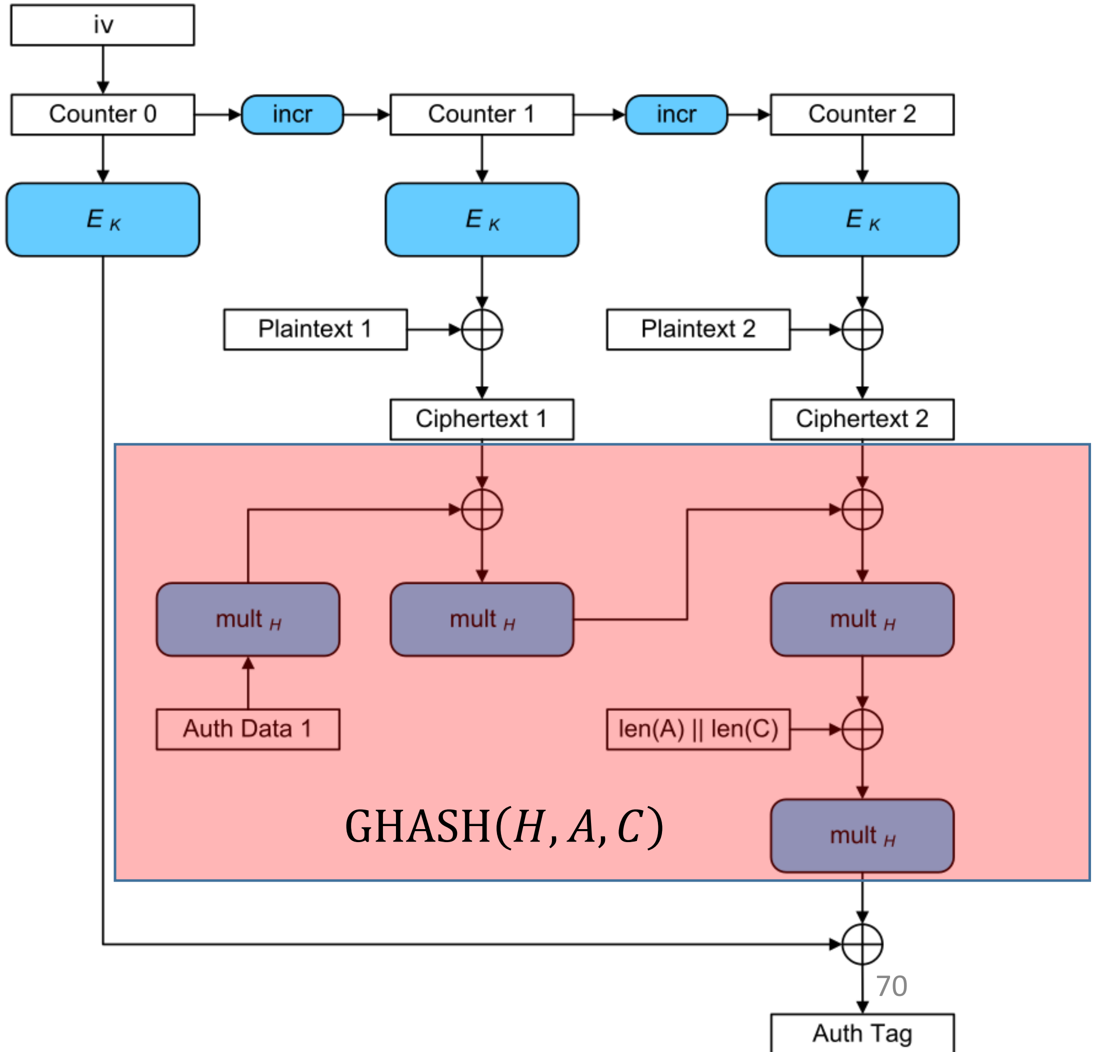
Length: k 16-byte blocks

GHASH in AES-GCM

$$\mathrm{GHASH}(H, \cancel{A}, C) = X_{t+1}$$

Where

$$T = X_{t+1} = \sum_{i \leq t} C_i \cdot H^{t-i+1}$$



Multi-Collision

Goal: Find $C = (C_1 \ , C_3 \ , C_3 \ )$ and K_1, K_2, K_3 such that

$$\mathrm{T} = \mathrm{GHASH}(H_1, C) = \mathrm{GHASH}(H_2, C) = \mathrm{GHASH}(H_3, C) = T$$

$Where\, \mathrm{H}_j = E_{K_j}(0^\lambda)$

Linear Constraints

$$\sum_i C_i \cdot H_1^{t-i+1} = \sum_i C_i \cdot H_2^{t-i+1} = \sum_i C_i \cdot H_3^{t-i+1}$$

Multi-Collision

Goal: Find $C = \left(C_1\ , C_2\ , C_3\ \right), N_1\ , N_2\ , N_3$ and K_1, K_2, K_3 such that

$$\mathrm{T} = \mathrm{GHASH}(H_1, C) \oplus E_K(N_1) = \mathrm{GHASH}(H_2, C) \oplus E_K(N_2)$$
$$= \mathrm{GHASH}(H_3, C) \oplus E_K(N_3)$$

$where\ H_j = E_{K_j}\left(0^\lambda\right)$

Three Linear Constraints: *For each* $j = 1{,}2{,}3$

$$\mathrm{T} = \boldsymbol{C_1} \cdot H_j^4 \oplus \boldsymbol{C_2} \cdot H_j^3 \oplus\ \boldsymbol{C_3} \cdot H_j^2 \oplus \mathrm{L} \cdot H_j^1 \oplus E_K\left(N_j\ \right)$$

Three Unknowns: $\boldsymbol{C_1}, \boldsymbol{C_2}$ *and* $\boldsymbol{C_3}$

Computing Key Multi-Collisions: AES-GCM

Input: Let nonce N, authentication tag T, and keys K_1, K_2, K_3 be arbitrary

Goal: Compute ciphertext C that decrypts under all 3 keys

Pre-compute: $H_i = AES_{K_i}(0^{128})$, $P_i = AES_{K_i}(N \| 0^{31}1)$, $L = |C|$

$$H_{1_4} \cdot C_1 \oplus H_{1_3} \cdot C_2 \oplus H_{1_2} \cdot C_3 \oplus H_1 \cdot L \oplus P_1 = T$$

$$H_{2_4} \cdot C_1 \oplus H_{2_3} \cdot C_2 \oplus H_{2_2} \cdot C_3 \oplus H_2 \cdot L \oplus P_2 = T$$

$$H_{3_4} \cdot C_1 \oplus H_{3_3} \cdot C_2 \oplus H_{3_2} \cdot C_3 \oplus H_3 \cdot L \oplus P_3 = T$$

Computing Key Multi-Collisions: AES-GCM

Input: Let nonce N, authentication tag T, and keys K_1, K_2, K_3 be arbitrary

Goal: Compute ciphertext C that decrypts under all 3 keys

Pre-compute: $H_i = AES_{K_i}(0^{128})$, $P_i = AES_{K_i}(N \,||\, 0^{31}1)$, $L = |C|$

$$\begin{bmatrix} H_{1_2} & H_1 & 1 \\ H_2^2 & H_2 & 1 \\ H_3^2 & H_3 & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} (T \oplus H_1 \cdot L \oplus P_1) \cdot H_{1_{-2}} \\ (T \oplus H_2 \cdot L \oplus P_2) \cdot H_2^{-2} \\ (T \oplus H_3 \cdot L \oplus P_3) \cdot H_3^{-2} \end{bmatrix}$$

Vandermonde matrix: we can use polynomial interpolation!

Computing Key Multi-Collisions: AES-GCM

- Implemented Multi-Collide-GCM using SageMath and Magma computational algebra system
- Timing experiments performed on desktop with Intel Core i9 processor and 128 GB RAM, running Linux x86-64

We make a ciphertext that decrypts under > 4000 keys in < 30 seconds!

k	Time (s)	Size (B)
2	0.18	48
2^{10}	6.6	16,400
2^{12}	29	65,552
2^{16}	1,820	1,048,592

Computing Key Multi-Collisions: AES-GCM

- Implemented Multi-Collide-GCM using SageMath and Magma computational algebra system
- Timing experiments performed on desktop with Intel Core i9 processor and 128 GB RAM, running Linux x86-64

We make a ciphertext that decrypts under > 4000 keys in < 30 seconds!

k	Time (s)	Size (B)
2	0.18	48
2^{10}	6.6	16,400
2^{12}	29	65,552
2^{16}	[illegible]	1,048,592

There exists an algorithm that does polynomial interpolation in $\mathcal{O}(k \log^2 k)$ using FFTs, so it's possible to create multi-collisions much faster [BM '74]

Computing Key Multi-Collisions

XSalsa20/Poly1305

ChaCha20/Poly1305

AES-GCM-SIV

⇒

Also vulnerable to key multi-collision attacks!

Attacks are more complex and less scalable than those for AES-GCM

Partitioning oracle attacks rely on:

1. Building splitting ciphertexts that can decrypt under k > 1 different keys

   Key Multi-collision Attacks

   [GLR CRYPTO'17] first showed an attack against AES-GCM for k = 2

2. Access to a partitioning oracle

Partitioning oracle attacks rely on:

1. Building splitting ciphertexts that can decrypt under k > 1 different keys

Key Multi-collision Attacks

[GLR CRYPTO'17] first showed an attack against AES-GCM for k = 2

2. Access to a partitioning oracle

Where do partitioning oracles arise?

Partitioning Oracles

Schemes we looked at in depth

- Shadowsocks proxy servers for UDP
  - Popular Internet censorship evasion tool
  - Partitioning oracle attacks enable an attacker to efficiently recover a password from a Shadowsocks server

Partitioning Oracles

Schemes we looked at in depth

- Shadowsocks proxy servers for UDP
  - Popular Internet censorship evasion tool
  - Partitioning oracle attacks enable an attacker to efficiently recover a password from a Shadowsocks server
- Early implementations of the OPAQUE asymmetric PAKE protocol
  - Selected by the IETF CFRG for standardization
  - Many early implementations went against protocol specification to use a non-committing AEAD scheme
  - These schemes are vulnerable to partitioning oracle attacks

Partitioning Oracles

Schemes we looked at in depth

- Shadowsocks proxy servers for UDP
  - Popular Internet censorship evasion tool
  - Partitioning oracle attacks enable an attacker to efficiently recover a password from a Shadowsocks server
- Early implementations of the OPAQUE asymmetric PAKE protocol
  - Selected by the IETF CFRG for standardization
  - Many early implementations went against protocol specification to use a non-committing AEAD scheme
  - These schemes are vulnerable to partitioning oracle attacks

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 do?

- Our paper is the latest in a growing body of evidence that non-committing AEAD can lead to vulnerabilities*
- So which committing AEAD scheme do we use?
  - None currently standardized!

We need a committing AEAD standard, and it should be the default choice for AEAD

* After we published our results, [ADGKLS ’20] also discussed the importance of committing AEAD

Conclusion

Contact: jlen@cs.cornell.edu

Full version: https://eprint.iacr.org/2020/1491.pdf

- Described partitioning oracle attacks, which exploit non-committing AEAD to recover secrets
- Widely-used AEAD schemes, such as AES-GCM, XSalsa20/Poly1305, ChaCha20/Poly1305, and AES-GCM-SIV, are ***not*** committing
- Partitioning oracle attacks can be used to recover passwords from Shadowsocks proxy servers and incorrect implementations of OPAQUE
- **Recommendation**: Design and standardize committing AEAD for deployment

Thank you to my co-authors and Hugo Krawczyk, Mihir Bellare, Scott Fluhrer, David McGrew, Kenny Patterson, Chris Wood, Steven Bellovin, and Samuel Neves!

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

AES-GCM SIV

$\underline{\mathsf{AE.E}(K_{\mathsf{in}} \parallel K_{\mathrm{out}}, N, M, A)}$	$\underline{\mathsf{AE.D}(K_{\mathsf{in}} \parallel K_{\mathrm{out}}, N, C, A)}$
$\mathsf{IV} \leftarrow \mathsf{F}(K_{\mathsf{in}} \parallel K_{\mathrm{out}}, N, M, A)$	$\mathsf{IV} \parallel C' \leftarrow C;\ M \leftarrow \mathsf{SE.D}^E(K_{\mathrm{out}}, C)$
$C \leftarrow \mathsf{SE.E}^E(K_{\mathrm{out}}, M; \mathsf{IV})$	$T \leftarrow \mathsf{F}^E(K_{\mathsf{in}} \parallel K_{\mathrm{out}}, N, M, A)$
Return C	If $T \neq \mathsf{IV}$ then return $\perp$ else return M

Fig. 4: **The SIV construction (with key reuse)** $\mathsf{AE} = \mathsf{SIV}[\mathsf{F}, \mathsf{SE}]$ **that is built on top of an ideal cipher** E**.**