Authenticated Encryption with Variable Stretch

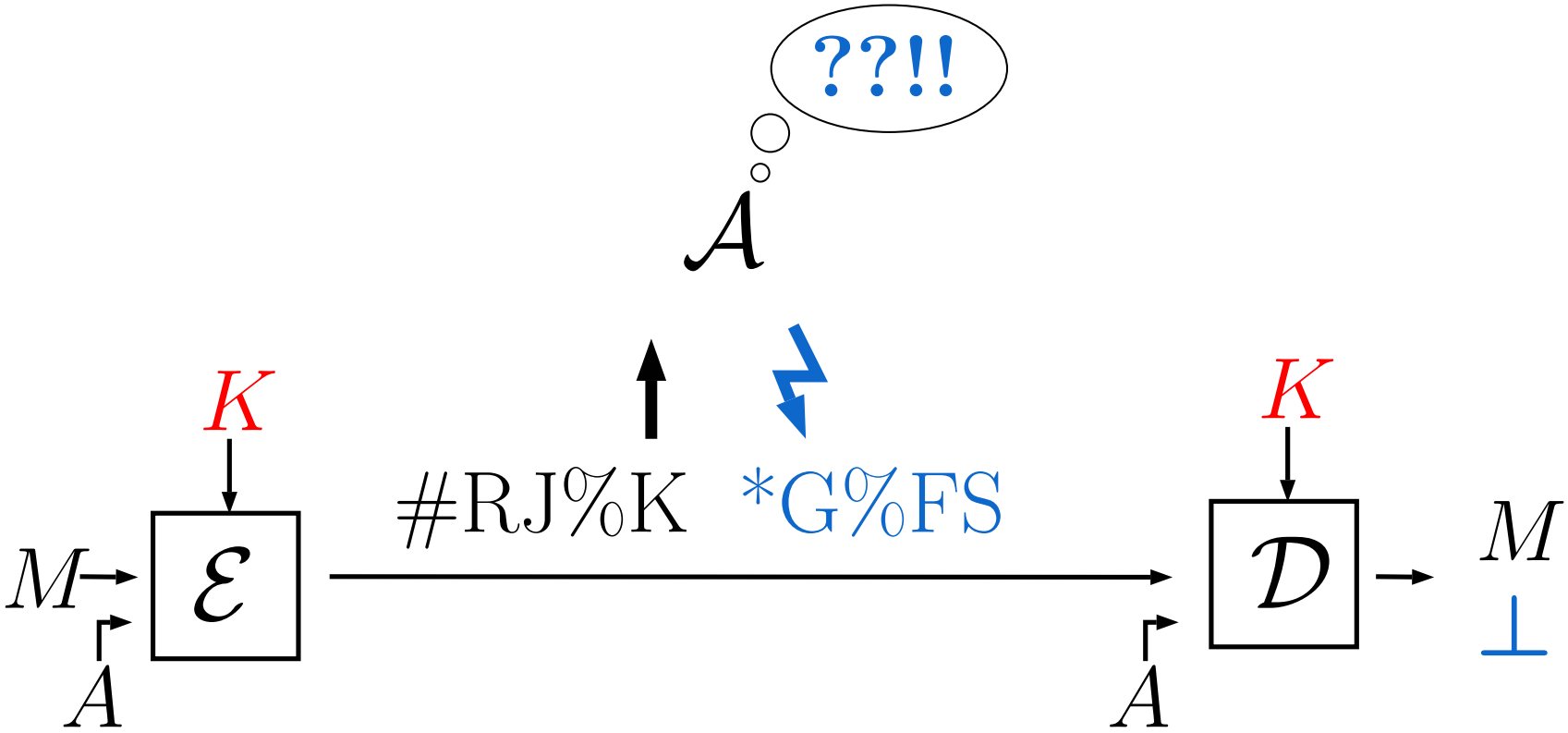
Reza Reyhanitabar[1] Serge Vaudenay[2] Damian Vizár[2]

[1] NEC Laboratories Europe, Germany [2] EPFL, Switzerland

ASIACRYPT 2016, Hanoi

This work was partly supported by the EU H2020 TREDISEC project, funded by the European Commission under grant agreement no. 644412. Damian Vizár was supported in part by Microsoft Research.

Authenticated Encryption



- **Confidentiality+Authenticity/Integrity for M**

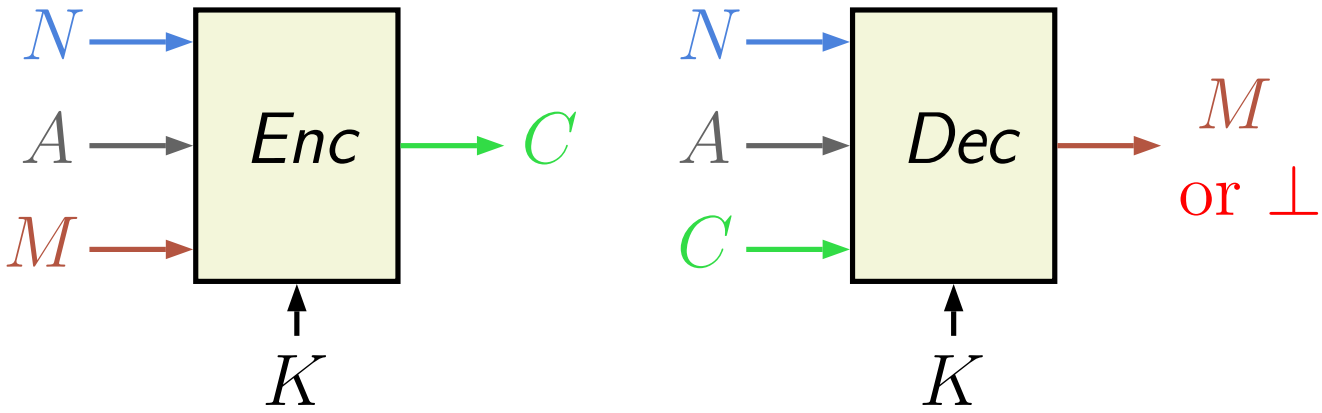
  [Bellare,Namprempre 00],[Katz,Yung 00]

- **Authenticity for A**

  [Rogaway 02]

Nonce-based AE with Associated Data (AEAD)

[Rogaway 02]

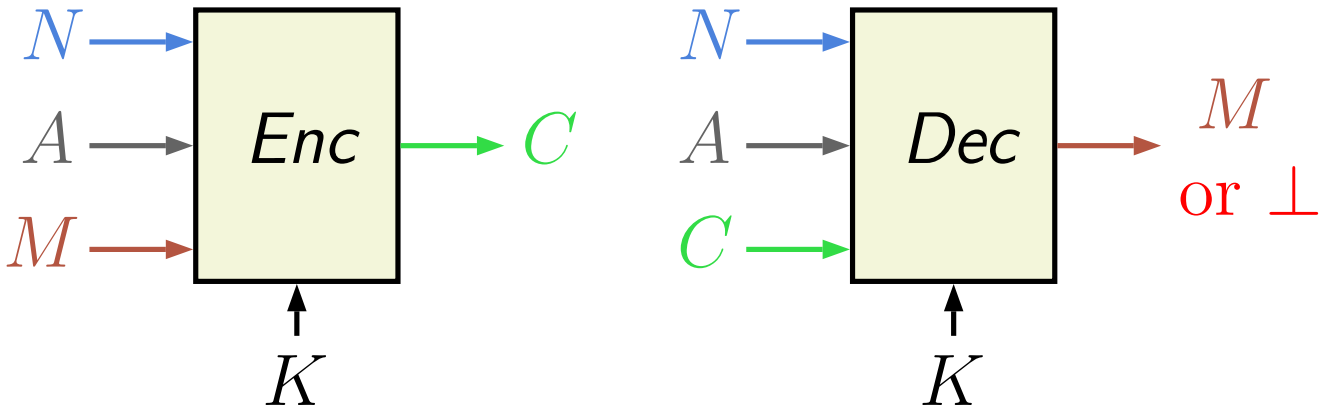


- *Enc*, *Dec*: deterministic algorithms
- *N*: Nonce, must not repeat
- *A*: Associated Data, must be authenticated, but not encrypted
- *M*: Plaintext, must be encrypted and authenticated
- *C*: Ciphertext
- *K*: Secret key

- for all K, N, A, M: $Dec(K, N, A, Enc(K, N, A, M)) = M$

Nonce-based AE with Associated Data (AEAD)

[Rogaway 02]

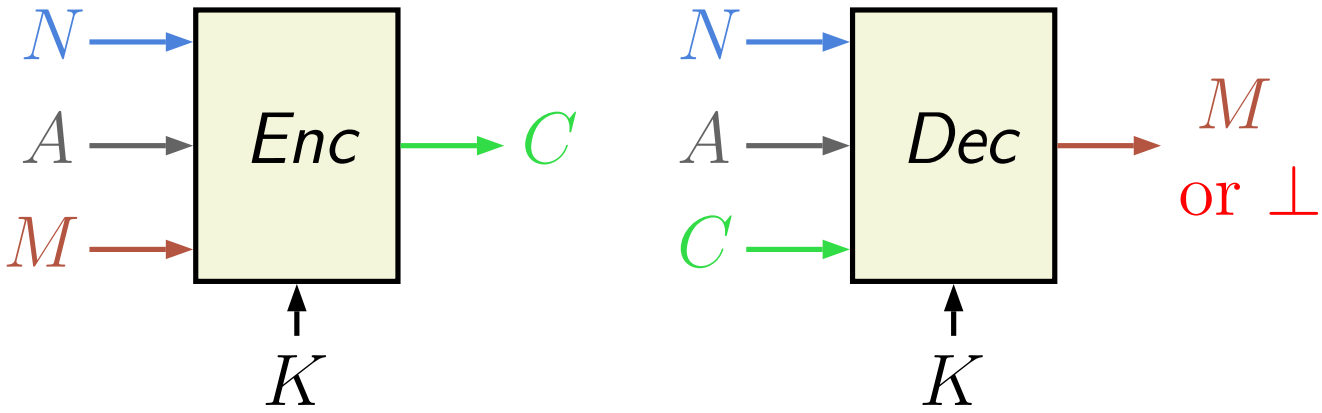


- *Enc*, *Dec*: deterministic algorithms
- *N*: Nonce, must not repeat
- *A*: Associated Data, must be authenticated, but not encrypted
- *M*: Plaintext, must be encrypted and authenticated
- *C*: Ciphertext
- *K*: Secret key

- for all K, N, A, M: $Dec(K, N, A, Enc(K, N, A, M)) = M$

Nonce-based AE with Associated Data (AEAD)

[Rogaway 02]



- *Enc*, *Dec*: deterministic algorithms
- *N*: Nonce, must not repeat
- *A*: Associated Data, must be authenticated, but not encrypted
- *M*: Plaintext, must be encrypted and authenticated
- *C*: Ciphertext
- *K*: Secret key

- for all K, N, A, M: $Dec(K, N, A, Enc(K, N, A, M)) = M$

Nonce-based AEAD and Misuse

Misuse: What if a scheme is not used as it should be?

Nonce-based AEAD and Misuse

Misuse: What if a scheme is not used as it should be?

- **Nonce misuse:** Uniqueness of nonces not guaranteed
  - Problems, possibly complete failure
  - **MRAE** [Rogaway, Shrimpton 06]
  - **OAE** [Fleischmann, Forler, Lucks 12]
  - **OAE2** [Hoang, Reyhanitabar, Rogaway, V. 15]

Nonce-based AEAD and Misuse

Misuse: What if a scheme is not used as it should be?

- **Nonce misuse:** Uniqueness of nonces not guaranteed
  - Problems, possibly complete failure
  - **MRAE** [Rogaway, Shrimpton 06]
  - **OAE** [Fleischmann, Forler, Lucks 12]
  - **OAE2** [Hoang, Reyhanitabar, Rogaway, V. 15]
- **Decryption misuse:** Leakage of unverified plaintext
  - Problems, possibly complete failure
  - **AE-RUP** [Andreeva, Bogdanov, Luykx, Mennink, Mouha, Yasuda 14]

Nonce-based AEAD and Misuse

Misuse: What if a scheme is not used as it should be?

- **Nonce misuse:** Uniqueness of nonces not guaranteed
  - Problems, possibly complete failure
  - **MRAE** [Rogaway, Shrimpton 06]
  - **OAE** [Fleischmann, Forler, Lucks 12]
  - **OAE2** [Hoang, Reyhanitabar, Rogaway, V. 15]
- **Decryption misuse:** Leakage of unverified plaintext
  - Problems, possibly complete failure
  - **AE-RUP** [Andreeva, Bogdanov, Luykx, Mennink, Mouha, Yasuda 14]
- **"Best possible security"**
  - **RAE** [Hoang, Krovetz, Rogaway 15]
  - cannot be online, hard to achieve

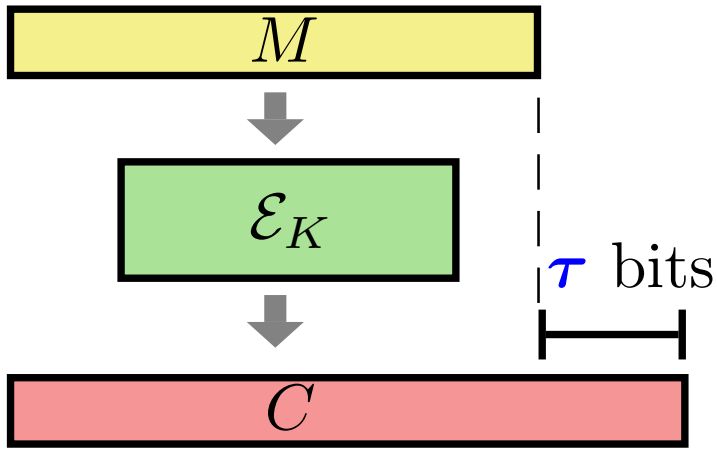
Nonce-based AEAD and Misuse

Misuse: What if a scheme is not used as it should be?

- **Nonce misuse:** Uniqueness of nonces not guaranteed
  - Problems, possibly complete failure
  - **MRAE** [Rogaway, Shrimpton 06]
  - **OAE** [Fleischmann, Forler, Lucks 12]
  - **OAE2** [Hoang, Reyhanitabar, Rogaway, V. 15]
- **Decryption misuse:** Leakage of unverified plaintext
  - Problems, possibly complete failure
  - **AE-RUP** [Andreeva, Bogdanov, Luykx, Mennink, Mouha, Yasuda 14]
- **"Best possible security"**
  - **RAE** [Hoang, Krovetz, Rogaway 15]
  - cannot be online, hard to achieve
- **Stretch misuse:** Varying "tag-length"

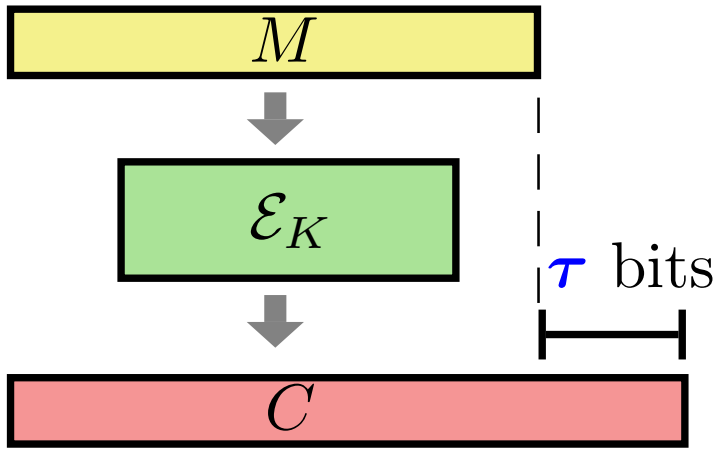
Stretch-Misuse

- Ciphertext expansion (a.k.a. stretch)
- Non-zero τ for authenticity
- A constant parameter
- Expected cost of forgery: $\approx \mathbf{2}^{\tau}$ decryption queries



Stretch-Misuse

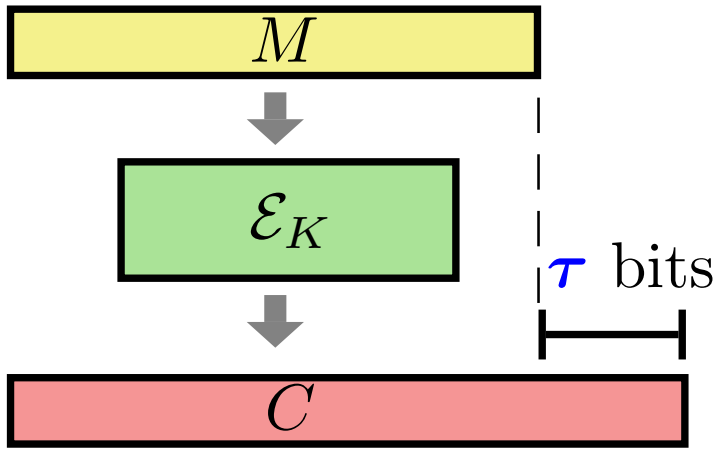
- Ciphertext expansion (a.k.a. stretch)
- Non-zero τ for authenticity
- A constant parameter
- Expected cost of forgery: $\approx \mathbf{2}^{\tau}$ decryption queries



Stretch misuse: varying stretch with the same key

Stretch-Misuse

- Ciphertext expansion (a.k.a. stretch)
- Non-zero τ for authenticity
- A constant parameter
- Expected cost of forgery: $\approx \mathbf{2}^{\tau}$ decryption queries

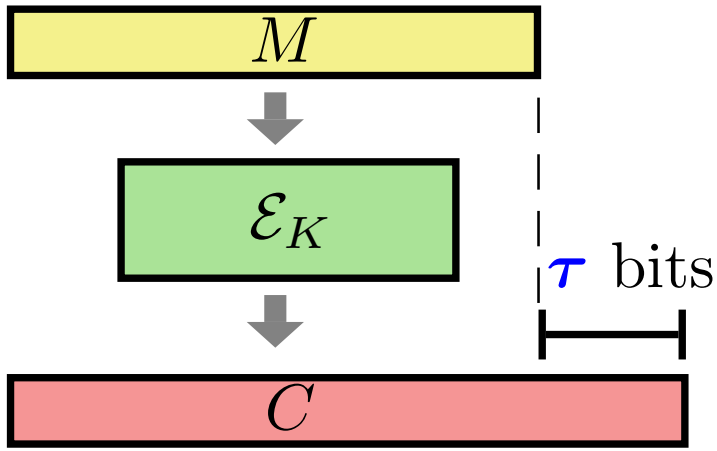


Stretch misuse: varying stretch with the same key

- **It can be attractive:** "sliding scale authenticity" for same K

Stretch-Misuse

- Ciphertext expansion (a.k.a. stretch)
- Non-zero τ for authenticity
- A constant parameter
- Expected cost of forgery: $\approx \mathbf{2}^{\tau}$ decryption queries

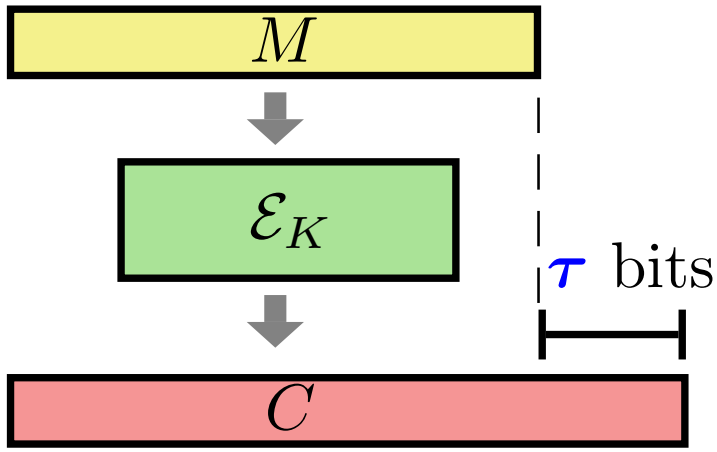


Stretch misuse: varying stretch with the same key

- **It can be attractive:** "sliding scale authenticity" for same K
- **It is easy to implement:** truncated tags

Stretch-Misuse

- Ciphertext expansion (a.k.a. stretch)
- Non-zero τ for authenticity
- A constant parameter
- Expected cost of forgery: $\approx 2^{\tau}$ decryption queries



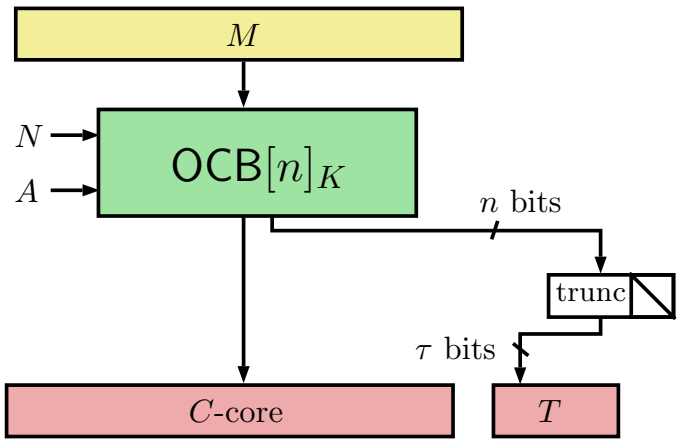
Stretch misuse: varying stretch with the same key

- **It can be attractive:** "sliding scale authenticity" for same K
- **It is easy to implement:** truncated tags
- **It impacts security**
  - "Attacks" that violate the intuitive security

Trivial Tag Length-Variation Attack on AEAD

[Manger 13, CFRG discussion]

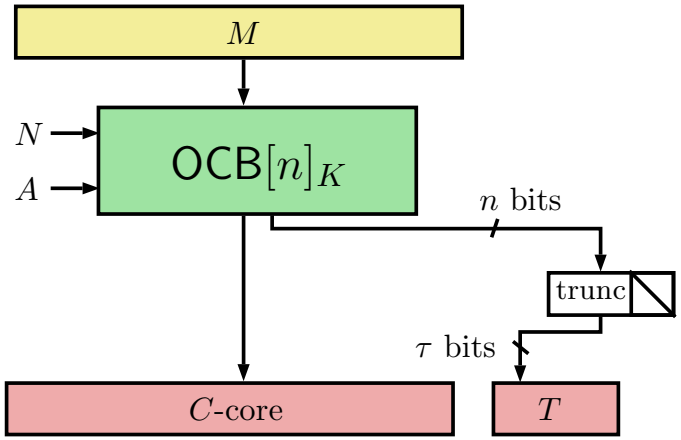
Tag computation in OCB (and others): compute tag of n bits, then truncate



Trivial Tag Length-Variation Attack on AEAD

[Manger 13, CFRG discussion]

Tag computation in OCB (and others): compute tag of n bits, then truncate

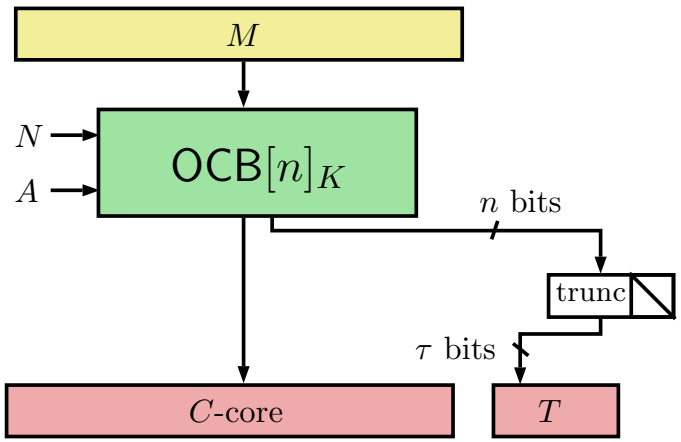


1. Query $C \| T \leftarrow \mathrm{OCB}[128]_K(N, A, M)$ for target (N, A, M)
2. Compute $T' \leftarrow \mathrm{trunc}_{64}(T)$
3. "Forge" $C \| T' \leftarrow \mathrm{OCB}[64]_K^{-1}(N, A, C \| T')$

Trivial Tag Length-Variation Attack on AEAD

[Manger 13, CFRG discussion]

Tag computation in OCB (and others): compute tag of n bits, then truncate



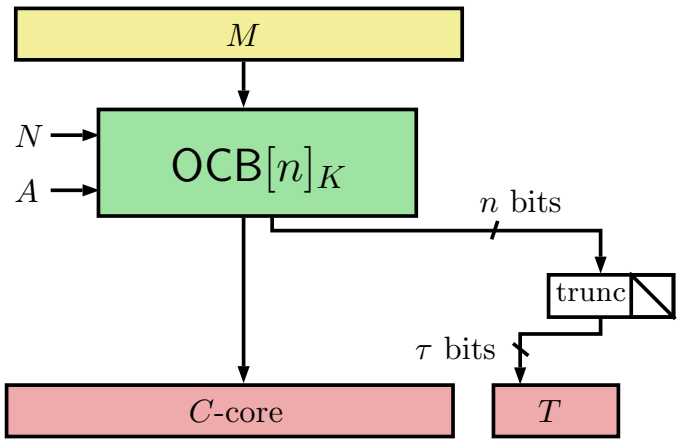
1. Query $C \| T \leftarrow \mathrm{OCB}[128]_K(N, A, M)$ for target (N, A, M)
2. Compute $T' \leftarrow \mathrm{trunc}_{64}(T)$
3. "Forge" $C \| T' \leftarrow \mathrm{OCB}[64]_K^{-1}(N, A, C \| T')$

Obvious, but contradicts the intuition of τ-bit resistance to forgery

Trivial Tag Length-Variation Attack on AEAD

[Manger 13, CFRG discussion]

Tag computation in OCB (and others): compute tag of n bits, then truncate



1. Query $C \| T \leftarrow \mathrm{OCB}[128]_K(N, A, M)$ for target (N, A, M)
2. Compute $T' \leftarrow \mathrm{trunc}_{64}(T)$
3. "Forge" $C \| T' \leftarrow \mathrm{OCB}[64]_K^{-1}(N, A, C \| T')$

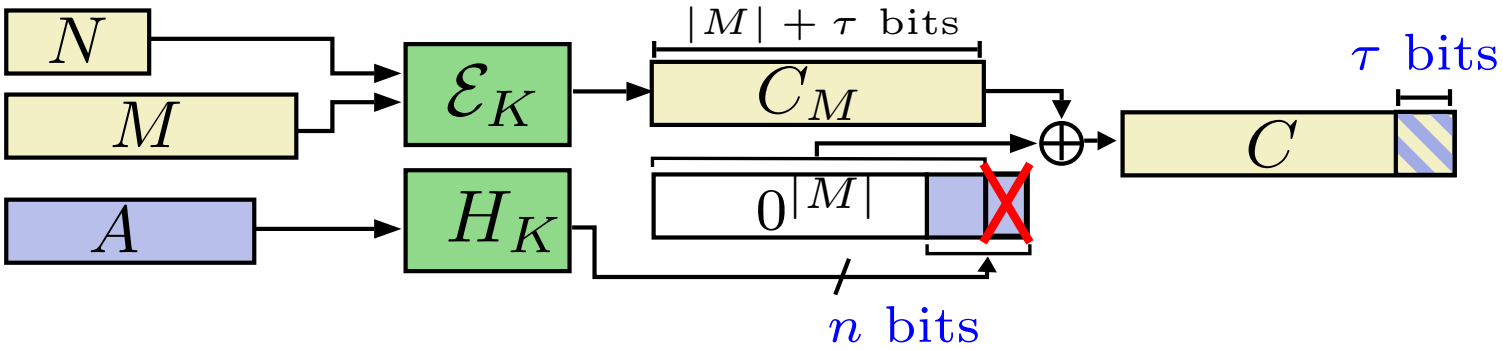
Obvious, but contradicts the intuition of τ-bit resistance to forgery

Heuristic countermeasures proposed
(OCB, OMD, OTR, CLOC&SILC)

Gradual Forgery for Ciphertext Translation

Original attack on OMD [Dobraunig, Eichlseder, Mendel, Schläffer 14]

Message-only core + AD-"hash" (H_K can be AXU)

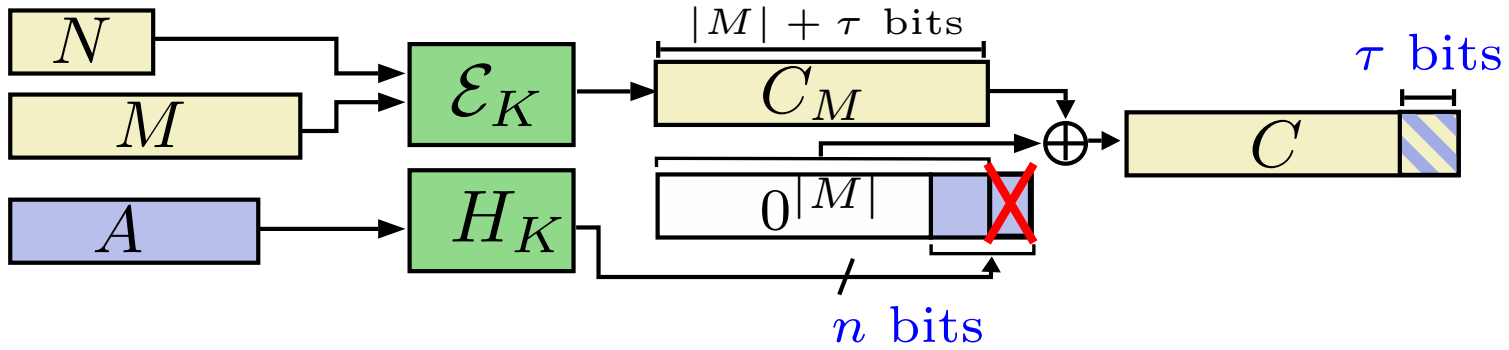


Key property: $Enc(K, N, A, M) \oplus Enc(K, N, A^*, M) = \mathrm{trunc}_\tau(H_K(A) \oplus H_K(A^*))$

Gradual Forgery for Ciphertext Translation

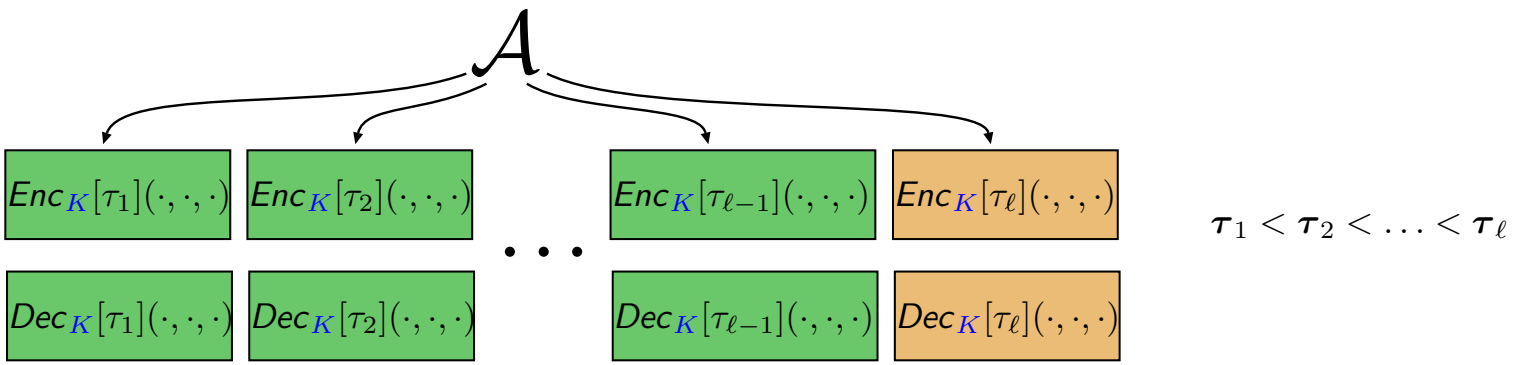
Original attack on OMD [Dobraunig, Eichlseder, Mendel, Schläffer 14]

Message-only core + AD-"hash" (H_K can be AXU)



Key property: $Enc(K, N, A, M) \oplus Enc(K, N, A^*, M) = \mathrm{trunc}_\tau(H_K(A) \oplus H_K(A^*))$

Attack: forge $Enc(N, A^*, M)$ with τ_ℓ bits of stretch



Gradual Forgery for Ciphertext Translation

Forgery for N, A^*, M with τ_g bits of stretch

1. Pick some $\mathbf{A} \neq \mathbf{A}^*$
2. Get $\mathbf{C}\|\mathbf{T} \leftarrow \mathbf{Enc}[\tau_1](\mathbf{N}, \mathbf{A}, \mathbf{M})$
3. Find $\delta \in \{\mathbf{0},\mathbf{1}\}^{\tau_1}$ s.t. $\mathbf{Dec}[\tau_1](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \delta))$ **succeeds**
4. Set $\mathbf{\Delta_A} \leftarrow \delta$
5. Get $\mathbf{C}\|\mathbf{T} \leftarrow \mathbf{Enc}[\tau_2](\mathbf{N}, \mathbf{A}, \mathbf{M})$
6. Find $\delta \in \{\mathbf{0},\mathbf{1}\}^{\tau_2 - \tau_1}$ s.t. $\mathbf{Dec}[\tau_2](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \mathbf{\Delta_A}\|\delta))$ **succeeds**
7. Set $\mathbf{\Delta_A} \leftarrow \mathbf{\Delta_A}\|\delta$
8. **...**
9. Get $\mathbf{C}\|\mathbf{T} \leftarrow \mathbf{Enc}[\tau_\ell](\mathbf{N}, \mathbf{A}, \mathbf{M})$
10. Find $\delta \in \{\mathbf{0},\mathbf{1}\}^{\tau_\ell - \tau_{\ell-1}}$ s.t. $\mathbf{Dec}[\tau_\ell](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \delta))$ **succeeds**
11. Output forgery $\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{\Delta_A}\|\delta)$

Gradual Forgery for Ciphertext Translation

Forgery for N, A^*, M with τ_g bits of stretch

1. Pick some $\mathbf{A} \neq \mathbf{A}^*$
2. Get $\mathbf{C}\|\mathbf{T} \leftarrow \mathbf{Enc}[\tau_1](\mathbf{N}, \mathbf{A}, \mathbf{M})$
3. Find $\delta \in \{\mathbf{0},\mathbf{1}\}^{\tau_1}$ s.t. $\mathbf{Dec}[\tau_1](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \delta))$ **succeeds**
4. Set $\mathbf{\Delta_A} \leftarrow \delta$ $\quad \triangleright\ \Delta_A = \mathrm{trunc}(H_K(A) \oplus H_K(A^*), \tau_1)$
5. Get $\mathbf{C}\|\mathbf{T} \leftarrow \mathbf{Enc}[\tau_2](\mathbf{N}, \mathbf{A}, \mathbf{M})$
6. Find $\delta \in \{\mathbf{0},\mathbf{1}\}^{\tau_2-\tau_1}$ s.t. $\mathbf{Dec}[\tau_2](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \mathbf{\Delta_A}\|\delta))$ **succeeds**
7. Set $\mathbf{\Delta_A} \leftarrow \mathbf{\Delta_A}\|\delta$
8. ...
9. Get $\mathbf{C}\|\mathbf{T} \leftarrow \mathbf{Enc}[\tau_\ell](\mathbf{N}, \mathbf{A}, \mathbf{M})$
10. Find $\delta \in \{\mathbf{0},\mathbf{1}\}^{\tau_\ell-\tau_{\ell-1}}$ s.t. $\mathbf{Dec}[\tau_\ell](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \delta))$ **succeeds**
11. Output forgery $\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{\Delta_A}\|\delta)$

Find first τ_1 bits
of
$H_K(A) \oplus H_K(A^*)$

Gradual Forgery for Ciphertext Translation

Forgery for N, A^*, M with τ_g bits of stretch

1. Pick some $\mathbf{A} \neq \mathbf{A}^*$
2. Get $\mathbf{C}\|\mathbf{T} \leftarrow \mathbf{Enc}[\tau_1](\mathbf{N}, \mathbf{A}, \mathbf{M})$
3. Find $\delta \in \{\mathbf{0},\mathbf{1}\}^{\tau_1}$ s.t. $\mathbf{Dec}[\tau_1](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \delta))$ **succeeds**
4. Set $\mathbf{\Delta_A} \leftarrow \delta$ $\quad \triangleright\ \Delta_A = \mathrm{trunc}(H_K(A) \oplus H_K(A^*), \tau_1)$
5. Get $\mathbf{C}\|\mathbf{T} \leftarrow \mathbf{Enc}[\tau_2](\mathbf{N}, \mathbf{A}, \mathbf{M})$
6. Find $\delta \in \{\mathbf{0},\mathbf{1}\}^{\tau_2-\tau_1}$ s.t. $\mathbf{Dec}[\tau_2](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \mathbf{\Delta_A}\|\delta))$ **succeeds**
7. Set $\mathbf{\Delta_A} \leftarrow \mathbf{\Delta_A}\|\delta$ $\quad \triangleright\ \Delta_A = \mathrm{trunc}(H_K(A) \oplus H_K(A^*), \tau_2)$
8. ...
9. Get $\mathbf{C}\|\mathbf{T} \leftarrow \mathbf{Enc}[\tau_\ell](\mathbf{N}, \mathbf{A}, \mathbf{M})$
10. Find $\delta \in \{\mathbf{0},\mathbf{1}\}^{\tau_\ell-\tau_{\ell-1}}$ s.t. $\mathbf{Dec}[\tau_\ell](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \delta))$ **succeeds**
11. Output forgery $\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{\Delta_A}\|\delta)$

Find first τ_1 bits of $H_K(A) \oplus H_K(A^*)$

Find next $\tau_2 - \tau_1$ bits of $H_K(A) \oplus H_K(A^*)$

• • •

Gradual Forgery for Ciphertext Translation

Forgery for N, A^*, M with τ_g bits of stretch

1. Pick some $\mathbf{A} \neq \mathbf{A}^*$
2. Get $\mathbf{C}\|\mathbf{T} \leftarrow \mathbf{Enc}[\tau_1](\mathbf{N}, \mathbf{A}, \mathbf{M})$
3. Find $\delta \in \{\mathbf{0},\mathbf{1}\}^{\tau_1}$ s.t. $\mathbf{Dec}[\tau_1](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \delta))$ **succeeds**
4. Set $\mathbf{\Delta_A} \leftarrow \delta$ ▷ $\Delta_A = \mathrm{trunc}(H_K(A) \oplus H_K(A^*), \tau_1)$
5. Get $\mathbf{C}\|\mathbf{T} \leftarrow \mathbf{Enc}[\tau_2](\mathbf{N}, \mathbf{A}, \mathbf{M})$
6. Find $\delta \in \{\mathbf{0},\mathbf{1}\}^{\tau_2-\tau_1}$ s.t. $\mathbf{Dec}[\tau_2](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \mathbf{\Delta_A}\|\delta))$ **succeeds**
7. Set $\mathbf{\Delta_A} \leftarrow \mathbf{\Delta_A}\|\delta$ ▷ $\Delta_A = \mathrm{trunc}(H_K(A) \oplus H_K(A^*), \tau_2)$
8. ...
9. Get $\mathbf{C}\|\mathbf{T} \leftarrow \mathbf{Enc}[\tau_\ell](\mathbf{N}, \mathbf{A}, \mathbf{M})$
10. Find $\delta \in \{\mathbf{0},\mathbf{1}\}^{\tau_\ell-\tau_{\ell-1}}$ s.t. $\mathbf{Dec}[\tau_\ell](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \delta))$ **succeeds**
11. Output forgery $\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{\Delta_A}\|\delta)$ ▷ $\Delta_A = \mathrm{trunc}(H_K(A) \oplus H_K(A^*), \tau_\ell)$

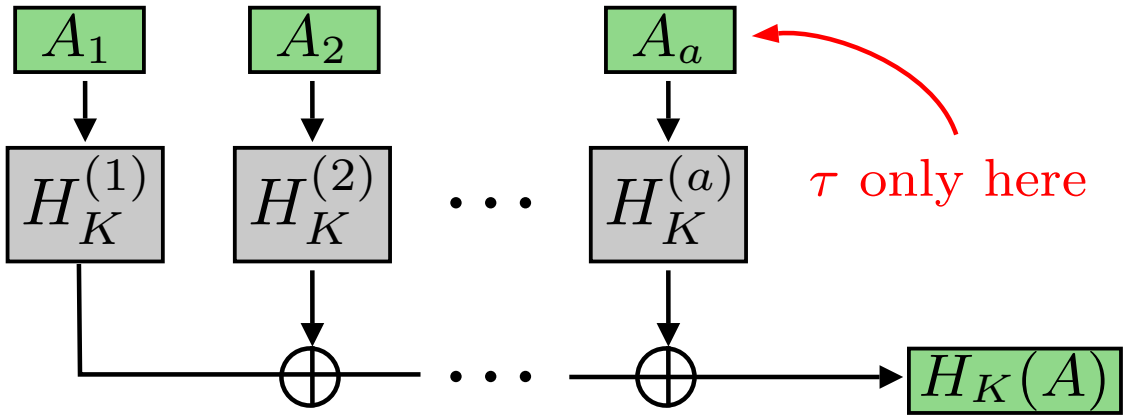
Find first τ_1 bits of $H_K(A) \oplus H_K(A^*)$ | Find next $\tau_2 - \tau_1$ bits of $H_K(A) \oplus H_K(A^*)$ | • • • | Find last $\tau_\ell - \tau_{\ell-1}$ bits of $H_K(A) \oplus H_K(A^*)$

Gradual Forgery for Ciphertext Translation

Applicability

- If **no countermeasures** OR τ **in nonce** $\Rightarrow$ works for **arbitrary** $\mathbf{H_K}$
  - OTR

- If τ **in AD** (or **in both AD and nonce**) $\Rightarrow$ works for $\mathbf{H_K}$ **like below**
  - Deoxys, OCB, GCM

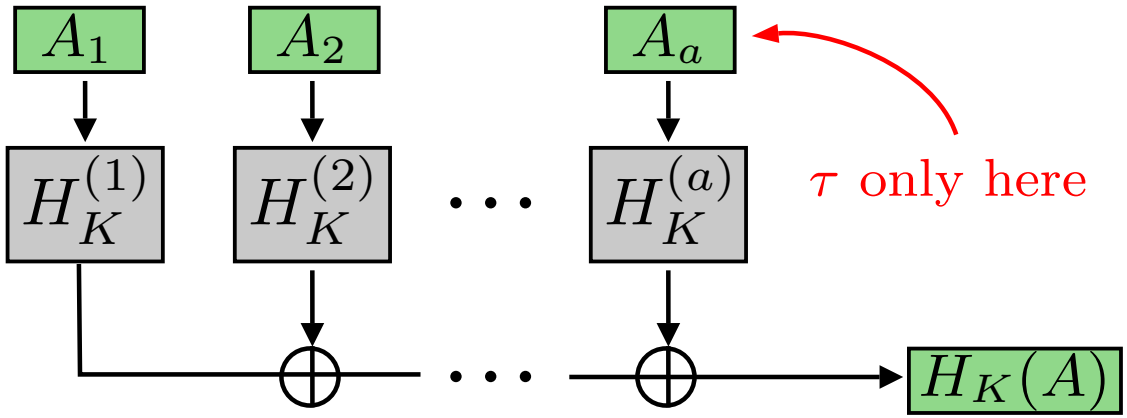


Gradual Forgery for Ciphertext Translation

Applicability

- If **no countermeasures** OR τ **in nonce** $\Rightarrow$ works for **arbitrary** $\mathbf{H_K}$
  - ▷ OTR

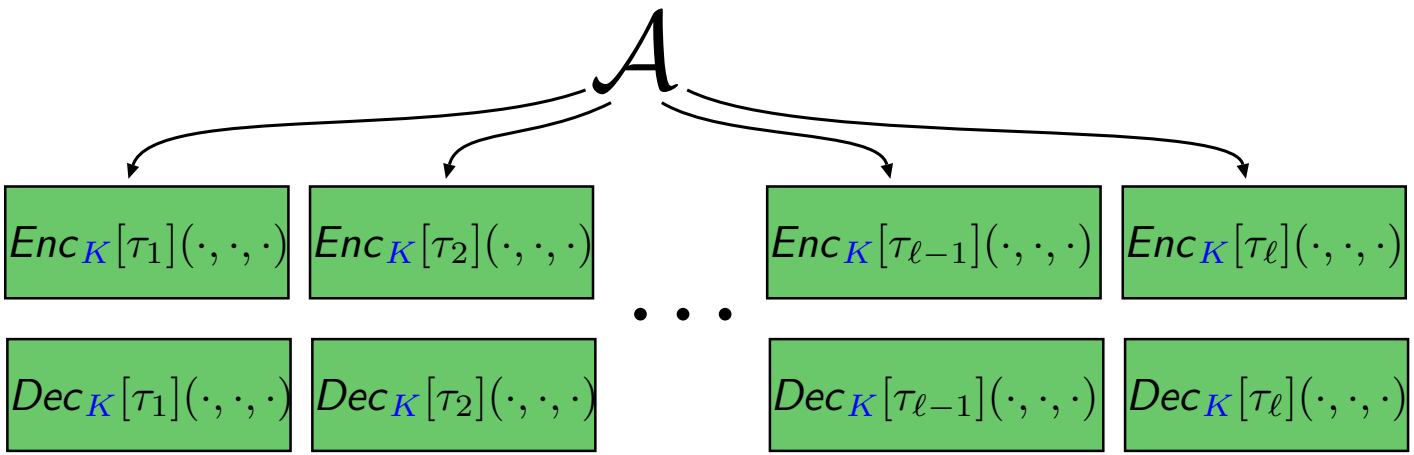
- If τ **in AD** (or **in both AD and nonce**) $\Rightarrow$ works for $\mathbf{H_K}$ **like below**
  - ▷ Deoxys, OCB, GCM



$\Rightarrow$ **Need systematic treatment!**

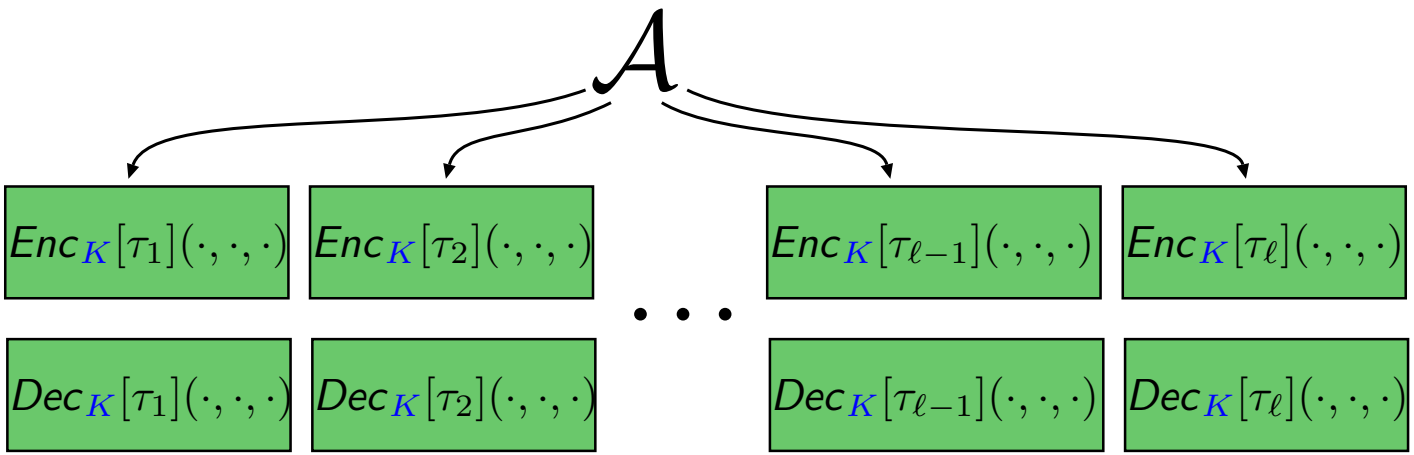
Capturing AEAD Security with Variable Stretch

- ✓ Define *stretch space* $\mathcal{I}_T = \{\tau_1, \tau_2, \ldots, \tau_\ell\}$
- ✓ Capture the attacks
- ✓ Show interactions between different amounts of stretch
- ✓ Security achievable by efficient, nonce-based schemes



Capturing AEAD Security with Variable Stretch

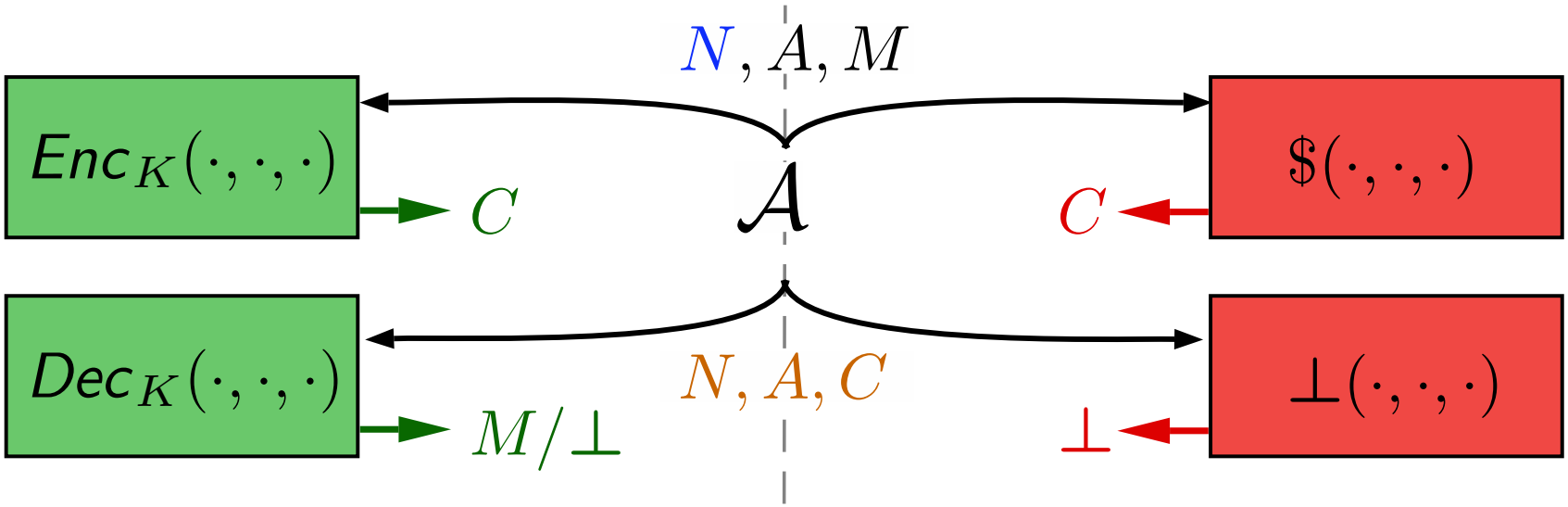
- ✓ Define *stretch space* $\mathcal{I}_T = \{\tau_1, \tau_2, \ldots, \tau_\ell\}$
- ✓ Capture the attacks
- ✓ Show interactions between different amounts of stretch
- ✓ Security achievable by efficient, nonce-based schemes
- ? What is the ideal system
- ? How to define advantage function



Nonce-based AE with Associated Data

[Rogaway 02], [Rogaway, Shrimpton 06]

$K \xleftarrow{\$} \mathcal{K}$, N never repeats, (N, A, C) not from an encryption query:

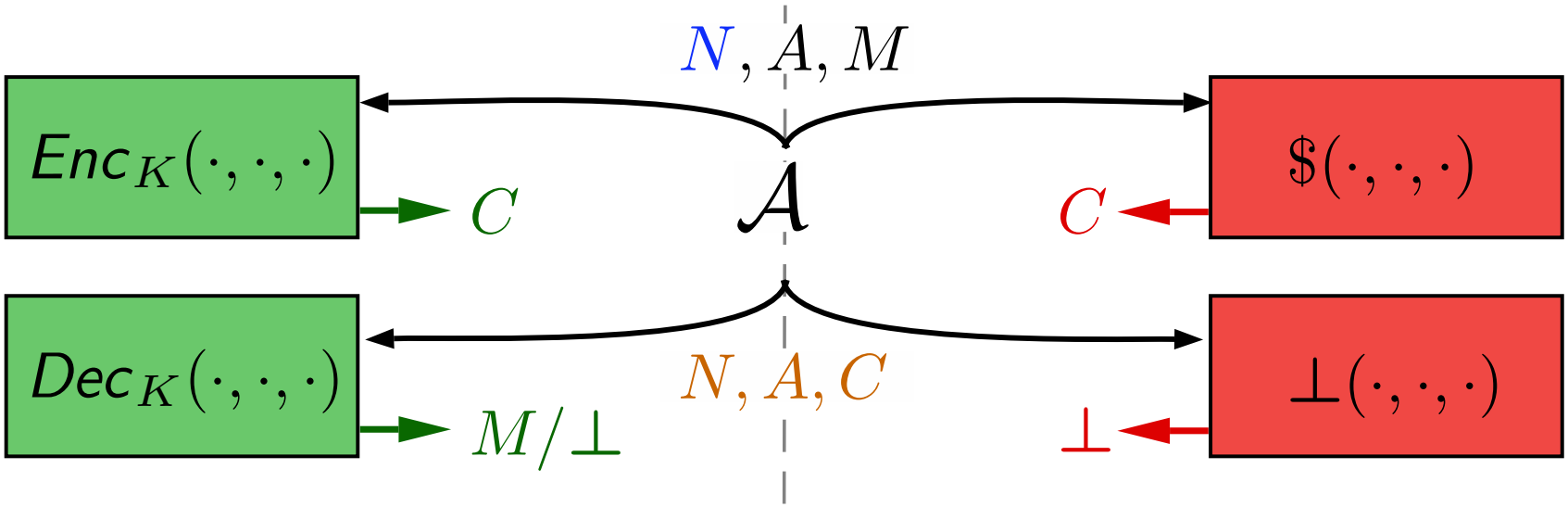


$$\mathbf{Adv}^{aead}_{\Pi}(\mathcal{A}) = \Pr\left[\mathcal{A}^{Enc_K(\cdot,\cdot,\cdot),Dec_K(\cdot,\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\$(\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} \Rightarrow 1\right]$$

Nonce-based AE with Associated Data

[Rogaway 02], [Rogaway, Shrimpton 06]

$K \xleftarrow{\$} \mathcal{K}$, N never repeats, (N, A, C) not from an encryption query:

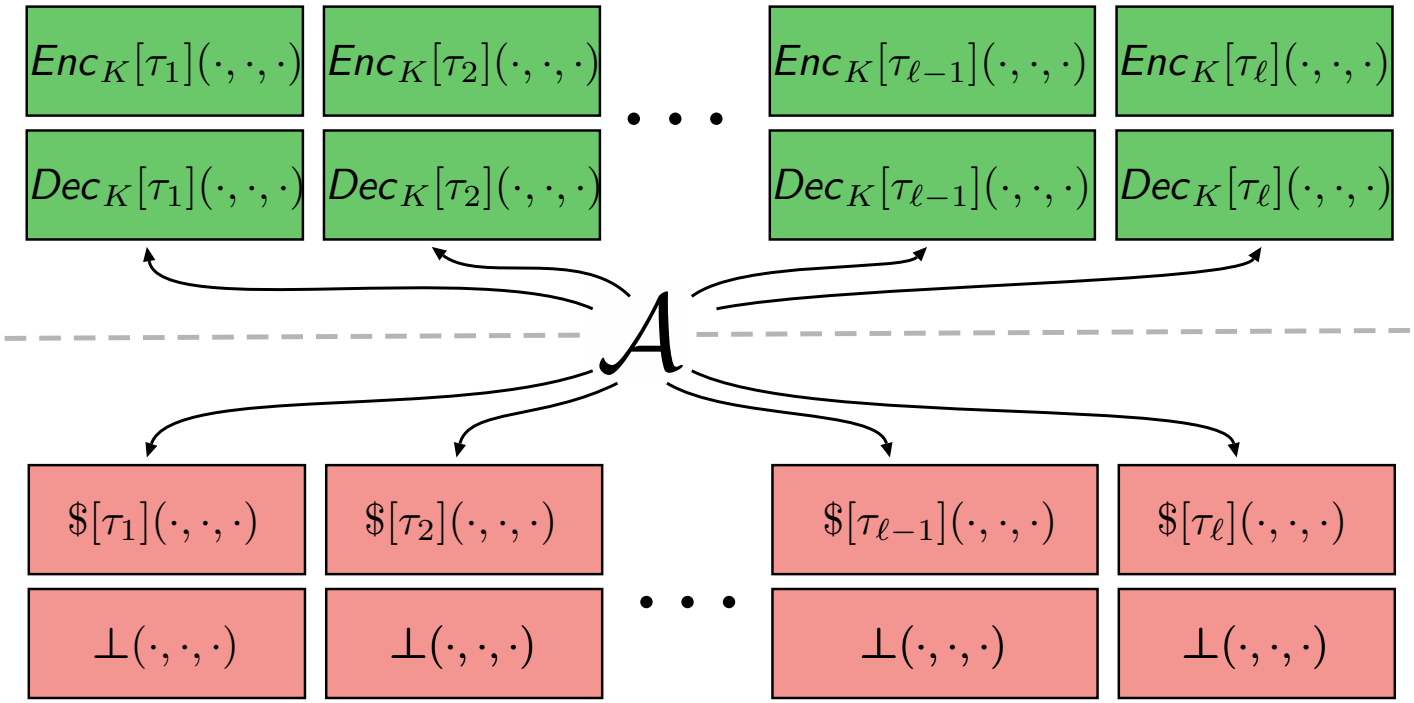


. . . and the ciphertext expansion is assumed to be constant

$$\mathbf{Adv}^{aead}_{\Pi}(\mathcal{A}) = \Pr\left[\mathcal{A}^{Enc_K(\cdot,\cdot,\cdot),Dec_K(\cdot,\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\$(\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} \Rightarrow 1\right]$$

(Not) Capturing AEAD Security with Variable Stretch

- $\Pi = (\mathsf{Enc}, \mathsf{Dec}, \mathcal{K})$ defined with $\tau \in \mathcal{I}_T = \{\tau_1, \tau_2, \ldots, \tau_\ell\}$
- Distinguishing **all** instances: not capturing intuition

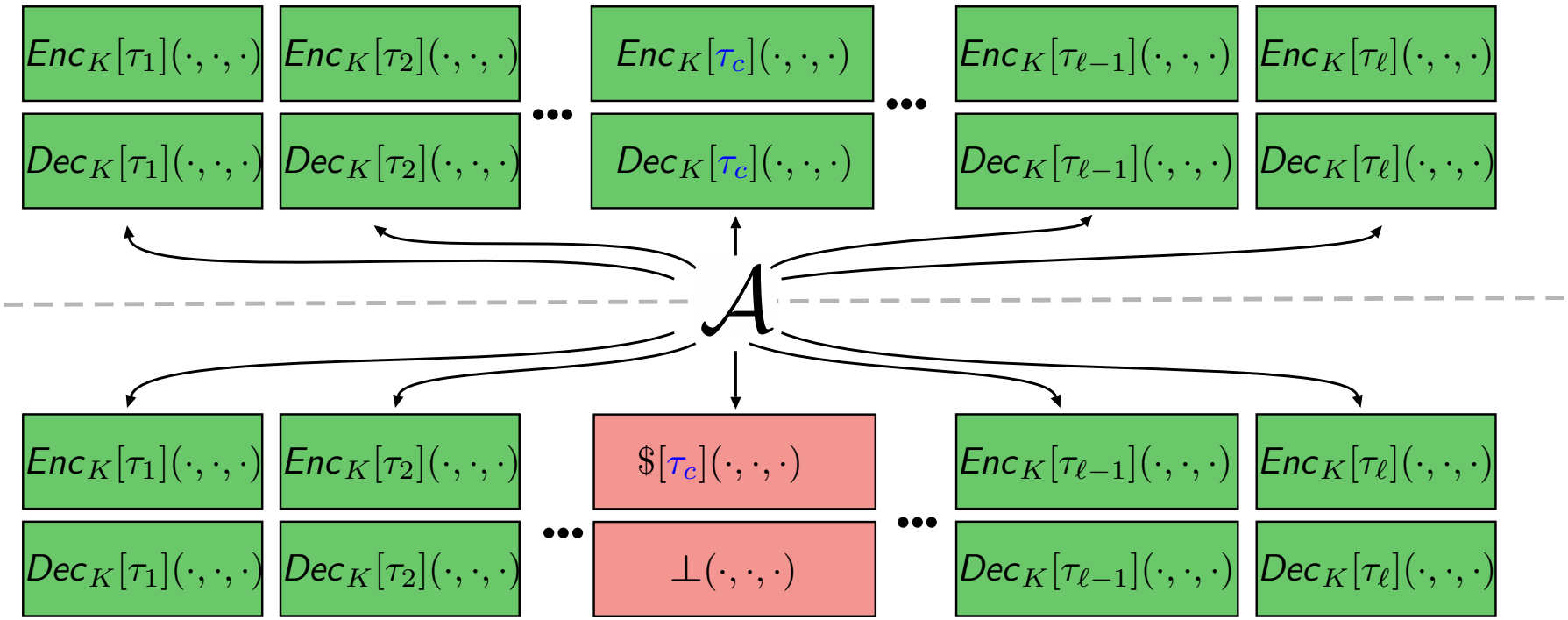


- **$\mathcal{A}$ can trivially win with $2^{\min \mathcal{I}_T}$ queries**
- **Other problems (All-in-one $\not\Leftrightarrow$ priv+auth)**

AEAD Security with Variable Stretch: $nvae(\tau_c)$

fixed but arbitrary "challenge" stretch $\tau_c \in \mathcal{I}_T = \{\tau_1, \tau_2, \ldots, \tau_\ell\}$:

- Unique (nonce,stretch) pairs
- Only non-trivial forgeries stretched by τ_c bits



$$\mathbf{Adv}_{\Pi}^{nvae(\tau_c)}(\mathcal{A}) = \Pr\left[\mathcal{A}^{\text{top system}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\text{lower system}} \Rightarrow 1\right]$$

Interpreting $nvae(\tau_c)$

Exact security level for queries stretched by τ_c

- Captures interactions with queries stretched by $\tau \neq \tau_c$ thanks to focusing on τ_c
- A proven bound applies to *every* $\tau_c \in \mathcal{I}_T$

Interpreting $nvae(\tau_c)$

Exact security level for queries stretched by τ_c

- Captures interactions with queries stretched by $\tau \neq \tau_c$ thanks to focusing on τ_c
- A proven bound applies to *every* $\tau_c \in \mathcal{I}_T$

Expected advantage

$$\mathbf{Adv}_{\Pi}^{nvae(\tau_c)} \leq \text{“small”} + c \cdot (q_{d(\tau_c)}/2^{\tau_c})^{\alpha}$$

- $c \cdot (q_{d(\tau_c)}/2^{\tau_c})^{\alpha}$ related to forging with τ_c bits of stretch
- “small” due to construction, should only depend on on total resources

E.g. “small”$= \mathbf{Adv}_{\mathsf{B}}^{prp}(t, \sigma) + \sigma^2/2^n$ with B an n-bit blockcipher and $c = \alpha = 1$.

Interpreting $nvae(\tau_c)$

Exact security level for queries stretched by τ_c

- Captures interactions with queries stretched by $\tau \neq \tau_c$ thanks to focusing on τ_c
- A proven bound applies to *every* $\tau_c \in \mathcal{I}_T$

Expected advantage

$$\mathbf{Adv}_{\Pi}^{nvae(\tau_c)} \leq \text{“small”} + c \cdot (q_{d(\tau_c)}/2^{\tau_c})^{\alpha}$$

- $c \cdot (q_{d(\tau_c)}/2^{\tau_c})^{\alpha}$ related to forging with τ_c bits of stretch
- “small” due to construction, should only depend on on total resources

E.g. “small”$= \mathbf{Adv}_{\mathsf{B}}^{prp}(t, \sigma) + \sigma^2/2^n$ with B an n-bit blockcipher and $c = \alpha = 1$.

Does not say small stretch is secure!

- Advantage always big for small τ_c

Capturing the attacks

Resources and advantage for stretch-variation forgery attacks

	Truncation	Gradual*
q_e	1	$\ell - 1$
q_d	0	$2^{\tau_2} + \sum_{i=1}^{\ell-1} 2^{\tau_i - \tau_{i-1}}$
$q_{e(\tau_c)}$	0	1
$q_{d(\tau_c)}$	1	$2^{\tau_c - \tau_{\ell-1}}$
Adv	1	1
Expected Adv	$1/2^{\tau_c}$	$2^{\tau_\ell - \tau_{\ell-1}}/2^{\tau_c}$

*where $\tau_1 < \tau_2 < \ldots < \tau_{\ell-1} < \tau_c$

Capturing the attacks

Resources and advantage for stretch-variation forgery attacks

	Truncation	Gradual*
q_e	1	$\ell - 1$
q_d	0	$2^{\tau_2} + \sum_{i=1}^{\ell-1} 2^{\tau_i - \tau_{i-1}}$
$q_{e(\tau_c)}$	0	1
$q_{d(\tau_c)}$	1	$2^{\tau_c - \tau_{\ell-1}}$
Adv	1	1
Expected Adv	$1/2^{\tau_c}$	$2^{\tau_\ell - \tau_{\ell-1}}/2^{\tau_c}$

*where $\tau_1 < \tau_2 < \ldots < \tau_{\ell-1} < \tau_c$

Capturing the attacks

Resources and advantage for stretch-variation forgery attacks

	Truncation	Gradual*
q_e	1	$\ell - 1$
q_d	0	$2^{\tau_2} + \sum_{i=1}^{\ell-1} 2^{\tau_i - \tau_{i-1}}$
$q_{e(\tau_c)}$	0	1
$q_{d(\tau_c)}$	1	$2^{\tau_c - \tau_{\ell-1}}$
Adv	1	1
Expected Adv	$1/2^{\tau_c}$	$2^{\tau_\ell - \tau_{\ell-1}}/2^{\tau_c}$

*where $\tau_1 < \tau_2 < \ldots < \tau_{\ell-1} < \tau_c$

Capturing the attacks

Resources and advantage for stretch-variation forgery attacks

	Truncation	Gradual*
q_e	1	$\ell - 1$
q_d	0	$2^{\tau_2} + \sum_{i=1}^{\ell-1} 2^{\tau_i - \tau_{i-1}}$
$q_{e(\tau_c)}$	0	1
$q_{d(\tau_c)}$	1	$2^{\tau_c - \tau_{\ell-1}}$
Adv	1	1
Expected Adv	$1/2^{\tau_c}$	$2^{\tau_\ell - \tau_{\ell-1}}/2^{\tau_c}$

*where $\tau_1 < \tau_2 < \ldots < \tau_{\ell-1} < \tau_c$

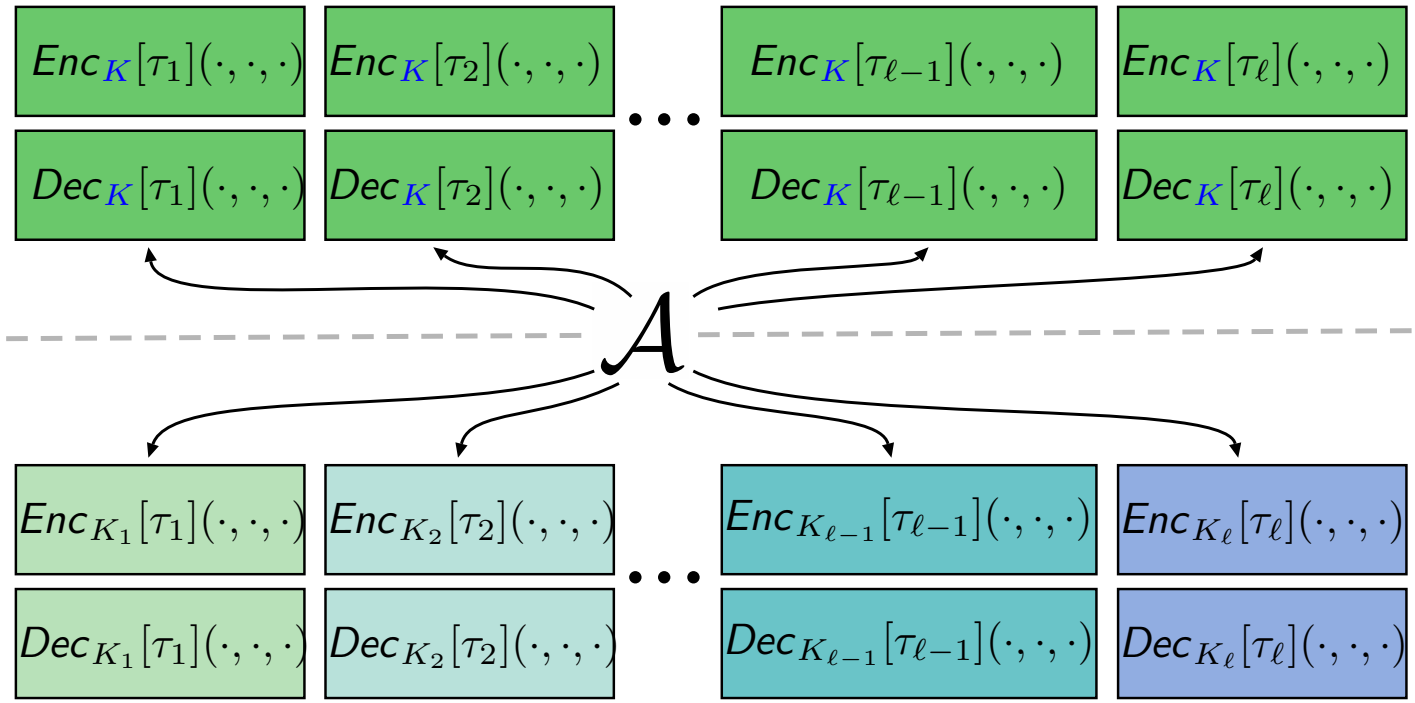
Example: $\{\tau_1, \tau_2, \tau_3, \tau_c\} = \{32, 64, 96, 128\}$

- 2^{32} decryption queries stretched by 2^{128}
- advantage 1 (compared to expected adv. $2^{32-128} = 2^{-96}$)

Achieving nvAE Modularly

Key-Equivalent Separation by Stretch

- Working with stretch space $\mathcal{I}_T = \{\tau_1, \tau_2, \ldots, \tau_\ell\}$
- Encryptions with fresh nonces per stretch



$$\mathbf{Adv}^{\mathit{kess}}_{\Pi}(\mathcal{A}) = \Pr\left[\mathcal{A}^{\text{top system}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\text{lower system}} \Rightarrow 1\right]$$

Achieving nvAE Modularly

Key-Equivalent Separation by Stretch

Low kess advantage $\neq$ AE security, but for any AEAD scheme Π with stretch space $\mathcal{I}_T = \{\tau_1, \tau_2, \ldots, \tau_\ell\}$:

$$\mathbf{Adv}_{\Pi}^{nvae(\tau_c)}(t, \mathbf{q_e}, \mathbf{q_d}, \sigma) \leq \mathbf{Adv}_{\Pi}^{kess}(t', \mathbf{q_e}, \mathbf{q_d}, \sigma) + \mathbf{Adv}_{\Pi[\tau_c]}^{aead}(t'', q_e^{\tau_c}, q_d^{\tau_c}, \sigma^{\tau_c})$$

where $\Pi[\tau_c]$ is Π used with τ_c-bit stretch, and

- $\mathbf{q_e}$ the encryption query complexities $(q_e^\tau | \tau \in \mathcal{I}_T)$
- $\mathbf{q_d}$ the decryption query complexities $(q_d^\tau | \tau \in \mathcal{I}_T)$
- σ the data complexities $(\sigma^\tau | \tau \in \mathcal{I}_T)$

- **Easier analysis if AEAD security already established!**

Achieving nvAE Security

Proof of concept: vOCB

OCB modified to be nvAE secure

- Add τ as tweak component **in all tweaks**
- Modified XEX construction
- Should not harm efficiency
- Analysis: only kess security (easy with TBC!)

$$\mathbf{Adv}^{nvae(\tau_c)}_{\mathrm{vOCB}[E]}(t, \mathbf{q_e}, \mathbf{q_d}, \sigma) \leq (|\mathcal{I}_T| + 2) \cdot \mathbf{Adv}^{\pm prp}_{E}(t', 2q) + \frac{28.5q^2}{2^n} + q_d^{\tau_c} \cdot \frac{2^{n-\tau_c}}{2^n - 1}$$

Achieving nvAE Security

Proof of concept: vOCB

OCB modified to be nvAE secure

- Add τ as tweak component **in all tweaks**
- Modified XEX construction
- Should not harm efficiency
- Analysis: only kess security (easy with TBC!)

$$\mathbf{Adv}^{nvae(\tau_c)}_{\mathrm{vOCB}[E]}(t, \mathbf{q_e}, \mathbf{q_d}, \sigma) \leq (|\mathcal{I}_T| + 2) \cdot \mathbf{Adv}^{\pm prp}_{E}(t', 2q) + \frac{28.5q^2}{2^n} + q_d^{\tau_c} \cdot \frac{2^{n-\tau_c}}{2^n - 1}$$

Achieving nvAE Security

Proof of concept: vOCB

OCB modified to be nvAE secure

- Add τ as tweak component **in all tweaks**
- Modified XEX construction
- Should not harm efficiency
- Analysis: only kess security (easy with TBC!)

$$\mathbf{Adv}^{nvae(\tau_c)}_{\mathrm{vOCB}[E]}(t, \mathbf{q_e}, \mathbf{q_d}, \sigma) \leq (|\mathcal{I}_T| + 2) \cdot \mathbf{Adv}^{\pm prp}_{E}(t', 2q) + \frac{28.5q^2}{2^n} + q_d^{\tau_c} \cdot \frac{2^{n-\tau_c}}{2^n - 1}$$

Achieving nvAE Security

Proof of concept: vOCB

OCB modified to be nvAE secure

- Add τ as tweak component **in all tweaks**
- Modified XEX construction
- Should not harm efficiency
- Analysis: only kess security (easy with TBC!)

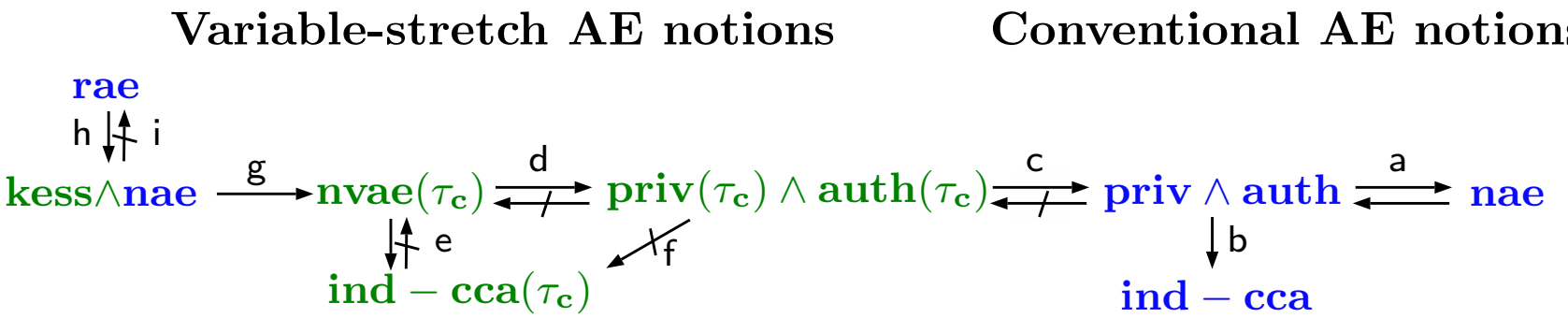
$$\mathbf{Adv}^{nvae(\tau_c)}_{\mathrm{vOCB}[E]}(t, \mathbf{q_e}, \mathbf{q_d}, \sigma) \leq (|\mathcal{I}_T| + 2) \cdot \mathbf{Adv}^{\pm prp}_{E}(t', 2q) + \frac{28.5q^2}{2^n} + q_d^{\tau_c} \cdot \frac{2^{n-\tau_c}}{2^n - 1}$$

Beyond proof of concept:

- Applicable to all AE schemes based on tweakable primitives
- Can treat OTR, Deoxys etc.

Conclusions

- We define security of nonce-based AEAD with variable stretch ☺
- We determine relations with existing notions ☺



- We show that nvAE security can be achieved and that schemes based on tweakable primitives are easily patched ☺
- Other schemes? ☹ **Generic transformation: open problem**

Thank you!

Postdocs-to-be mailto:**job_lasec@epfl.ch**

$nvae(\tau_c)$: Adversarial Resources

Default resources:

- Time t
- For **every** value of stretch $\tau \in \mathcal{I}_T$ watch:
  - Number of encryption queries q_e^τ
  - Number of decryption queries q_d^τ
  - Amount of data σ^τ

Fine granularity, generality

$nvae(\tau_c)$: Adversarial Resources

Default resources:

- Time t
- For **every** value of stretch $\tau \in \mathcal{I}_T$ watch:
  - Number of encryption queries q_e^τ
  - Number of decryption queries q_d^τ
  - Amount of data σ^τ

Fine granularity, generality

Coarser granularity best in most cases:

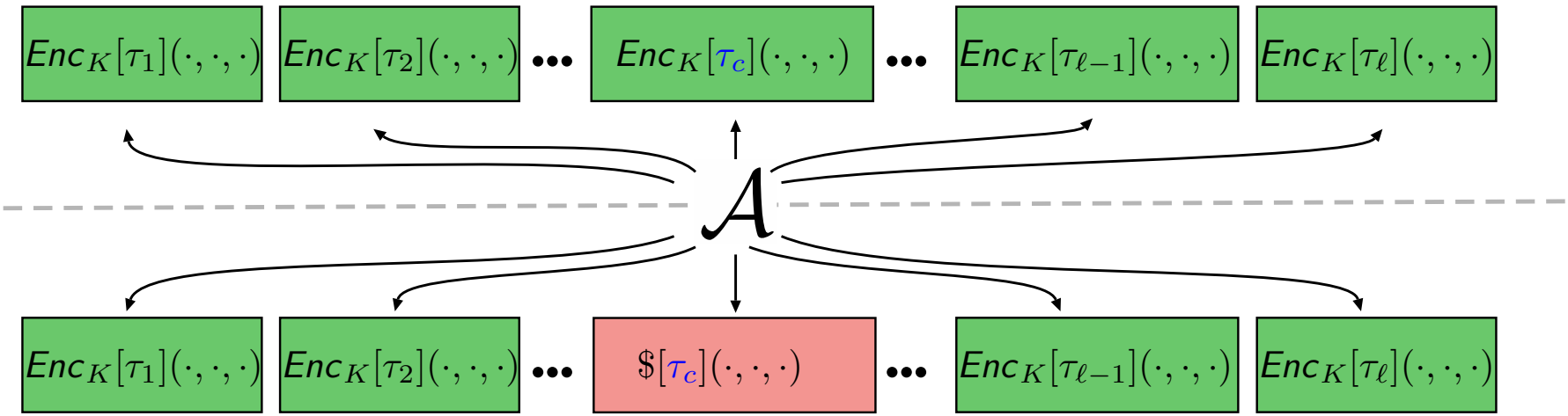
- Total number of encryptions $q_e = \sum_{\tau \in \mathcal{I}_T} q_e^\tau$
- Total number of decryptions $q_d = \sum_{\tau \in \mathcal{I}_T} q_d \tau$
- Total amount of data $\sigma = \sum_{\tau \in \mathcal{I}_T} \sigma_d \tau$
- Keep $q_e^{\tau_c}, q_d^{\tau_c}, \sigma^{\tau_c}$ apart

AEAD Security with Variable Stretch

Priv + Auth $\not\Leftrightarrow$ All-in-One

fixed but arbitrary "challenge" stretch $\tau_c \in \mathcal{I}_T = \{\tau_1, \tau_2, \ldots, \tau_\ell\}$:

- Unique nonces for (nonce,stretch) pairs



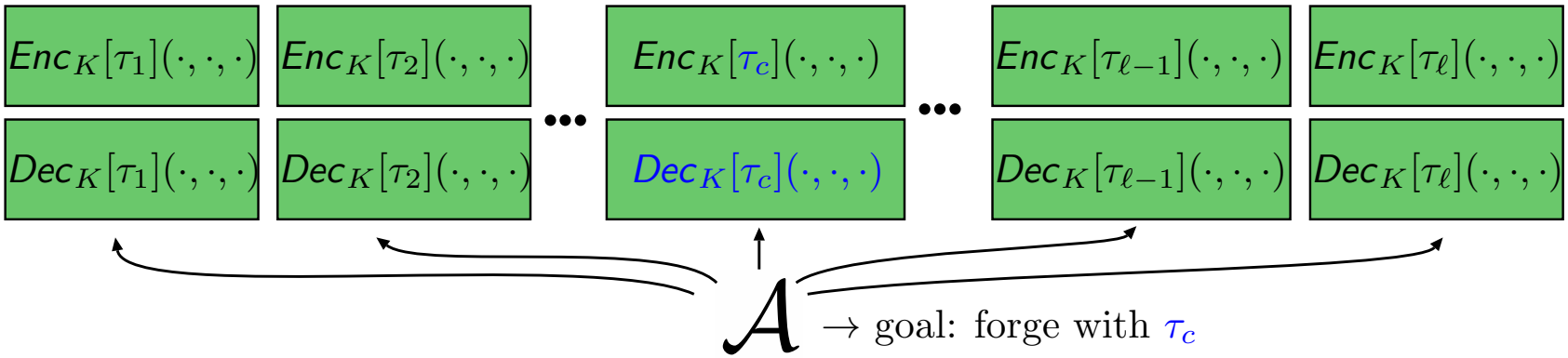
$$\mathbf{Adv}_{\Pi}^{priv(\tau_c)}(\mathcal{A}) = \Pr\left[\mathcal{A}^{\text{top system}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\text{lower system}} \Rightarrow 1\right]$$

AEAD Security with Variable Stretch

Priv + Auth $\not\Leftrightarrow$ All-in-One

fixed but arbitrary "challenge" stretch $\tau_c \in \mathcal{I}_T = \{\tau_1, \tau_2, \ldots, \tau_\ell\}$:

- Unique nonces for (nonce,stretch) pairs
- Only non-trivial forgeries stretched by τ_c bits



$$\mathbf{Adv}_{\Pi}^{auth(\tau_c)}(\mathcal{A}) = \Pr\left[\mathcal{A}^{\text{the system}} \text{ forges }\right]$$

AEAD Security with Variable Stretch

Priv + Auth $\not\Leftrightarrow$ All-in-One

We have

$$nvae(\tau_c) \Rightarrow priv(\tau_c) \quad \text{and} \quad nvae(\tau_c) \Rightarrow auth(\tau_c)$$
$$nvae(\tau_c) \Rightarrow ind-cca(\tau_c)$$

but

$$priv(\tau_c) \wedge auth(\tau_c) \not\Rightarrow nvae(\tau_c)$$
$$priv(\tau_c) \wedge auth(\tau_c) \not\Rightarrow ind-cca(\tau_c)$$

AEAD Security with Variable Stretch

Priv + Auth $\not\Leftrightarrow$ All-in-One

We have

$$nvae(\tau_c) \Rightarrow priv(\tau_c) \quad \text{and} \quad nvae(\tau_c) \Rightarrow auth(\tau_c)$$
$$nvae(\tau_c) \Rightarrow ind-cca(\tau_c)$$

but

$$priv(\tau_c) \wedge auth(\tau_c) \not\Rightarrow nvae(\tau_c)$$
$$priv(\tau_c) \wedge auth(\tau_c) \not\Rightarrow ind-cca(\tau_c)$$

Decryption queries expanded by $\tau \neq \tau_c$ may leak information about queries expanded by τ_c

AEAD Security with Variable Stretch

Priv + Auth $\not\Leftrightarrow$ All-in-One

All-in-one AE security $\Leftrightarrow$ privacy+confidentiality [Rogaway, Shrimpton 06]

AEAD Security with Variable Stretch

Priv + Auth $\not\Leftrightarrow$ All-in-One

All-in-one AE security $\Leftrightarrow$ privacy+confidentiality [Rogaway, Shrimpton 06]

Define *ind-cca*(τ_c)**,** *priv*(τ_c) **and** *auth*(τ_c)**:** ▷ similarly as *nvae*(τ_c)

$$\mathit{nvae}(\tau_c) \Rightarrow \mathit{priv}(\tau_c)$$
$$\mathit{nvae}(\tau_c) \Rightarrow \mathit{auth}(\tau_c)$$
$$\mathit{nvae}(\tau_c) \Rightarrow \mathit{ind\text{-}cca}(\tau_c)$$

but

$$\mathit{priv}(\tau_c) \wedge \mathit{auth}(\tau_c) \not\Rightarrow \mathit{nvae}(\tau_c)$$
$$\mathit{priv}(\tau_c) \wedge \mathit{auth}(\tau_c) \not\Rightarrow \mathit{ind\text{-}cca}(\tau_c)$$

AEAD Security with Variable Stretch

Priv + Auth $\not\Leftrightarrow$ All-in-One

All-in-one AE security $\Leftrightarrow$ privacy+confidentiality [Rogaway, Shrimpton 06]

Define *ind-cca*(τ_c)**,** *priv*(τ_c) **and** *auth*(τ_c)**:** ▷ similarly as *nvae*(τ_c)

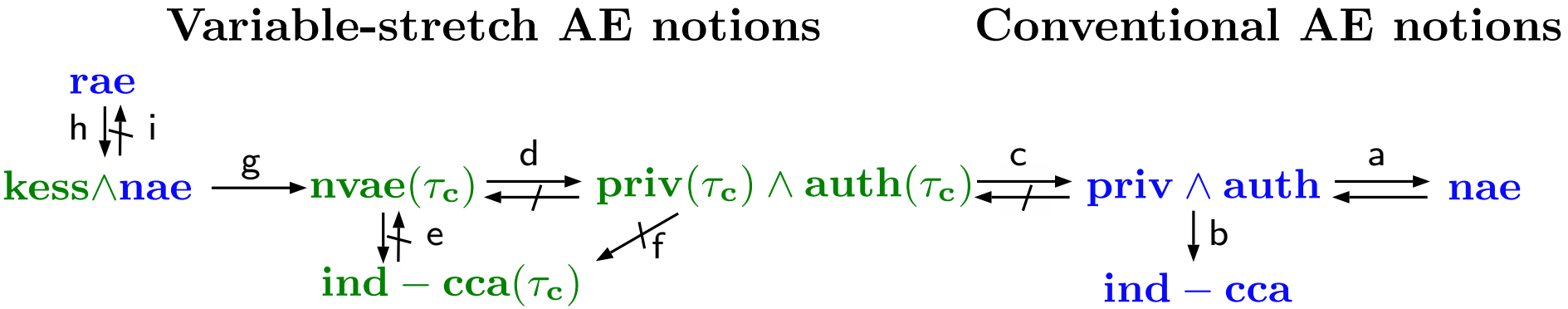
$$\textit{nvae}(\tau_c) \Rightarrow \textit{priv}(\tau_c)$$
$$\textit{nvae}(\tau_c) \Rightarrow \textit{auth}(\tau_c)$$
$$\textit{nvae}(\tau_c) \Rightarrow \textit{ind-cca}(\tau_c)$$

but

$$\textit{priv}(\tau_c) \wedge \textit{auth}(\tau_c) \not\Rightarrow \textit{nvae}(\tau_c)$$
$$\textit{priv}(\tau_c) \wedge \textit{auth}(\tau_c) \not\Rightarrow \textit{ind-cca}(\tau_c)$$

$\Rightarrow$ Decryption queries expanded by $\tau \neq \tau_c$ may leak information about queries expanded by τ_c

Relations among Notions



Previous works: a [Rogaway, Shrimpton 06] b [Bellare, Namprempre 00]
This work: c, d, e, f, g, h, i

Extending XEX

- Label every $\tau \in \mathcal{I}_T$ bijectively with $\lambda : \mathcal{I}_T \to \{0, 1, \ldots, |\mathcal{I}_T| - 1\}$.
- Compute $m = \lceil \log_2 |\mathcal{I}_T| \rceil$ and
  - $L_* = E_K(0^n)$
  - $L_\tau = \lambda(\tau) \cdot 2^2 \cdot L_*$ for $\tau \in \mathcal{I}_T$
  - $L(0) = 2^{2+m} \cdot L_*$
  - $L(\ell) = 2 \cdot L(\ell - 1)$ for $\ell > 0$.
- Compute Δ-values:

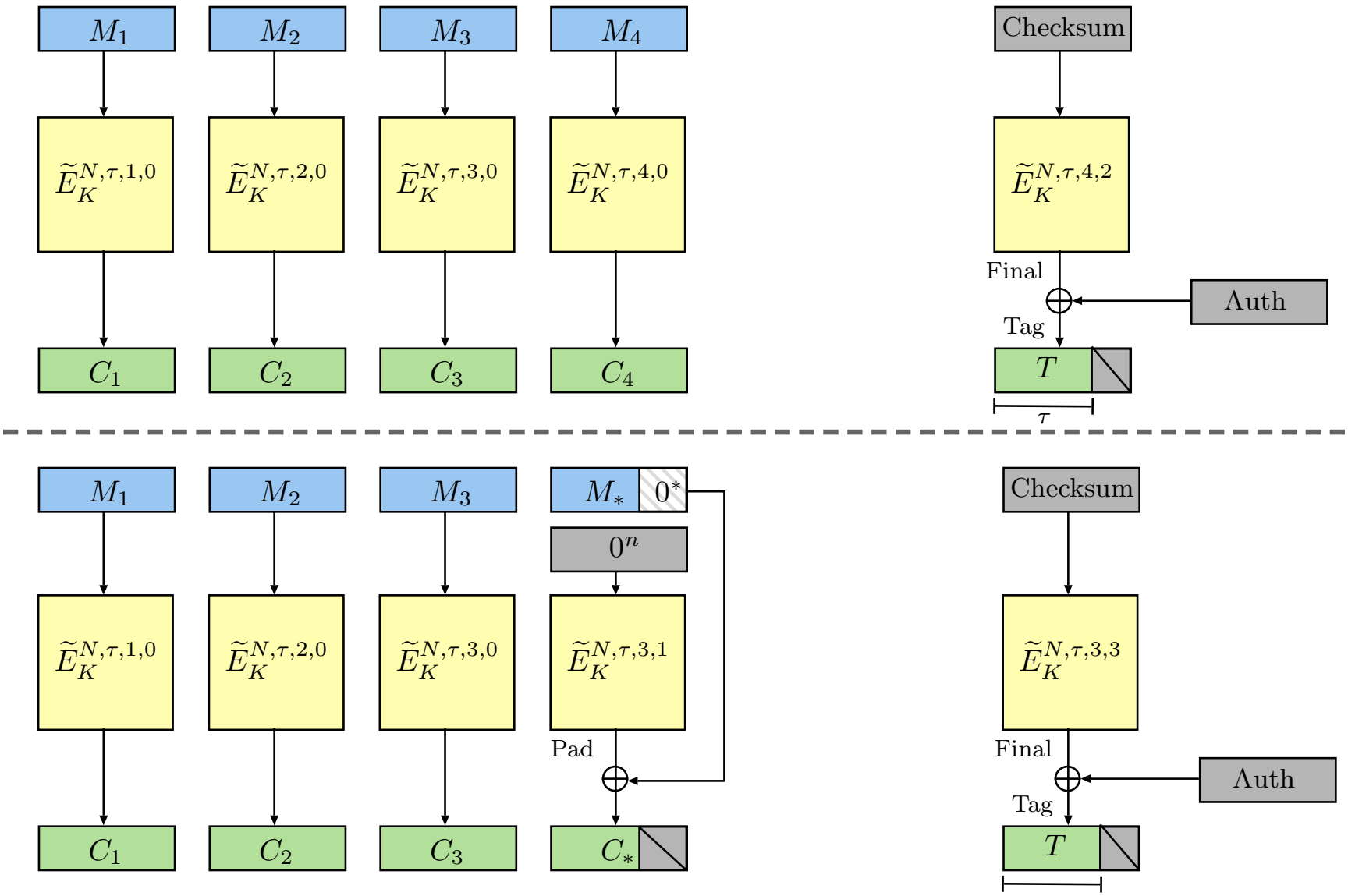
$$
\begin{aligned}
&\Delta_{N,0,0,0} = H(K, N),\\
&\Delta_{N,\tau,0,0} = \Delta_{N,0,0,0} \oplus L_\tau,\\
&\Delta_{N,\tau,i+1,0} = \Delta_{N,\tau,i,0} \oplus L(\mathrm{ntz}(i+1)) \text{ for } i \geq 0,\\
&\Delta_{N,\tau,i,j} = \Delta_{N,\tau,i,0} \oplus j \cdot L_* \text{ for } j \in \{0, 1, 2, 3\},\\
&\Delta_{\tau,0,0} = L_\tau,\\
&\Delta_{\tau,i+1,0} = \Delta_{\tau,i,0} \oplus L(\mathrm{ntz}(i+1)) \text{ for } i \geq 0,\\
&\Delta_{\tau,i,j} = \Delta_{\tau,i,0} \oplus j \cdot L_* \text{ for } j \in \{0, 1, 2, 3\}.
\end{aligned}
$$

A call to $\widetilde{E}$ is evaluated as follows:

$$\widetilde{E}_K^{N,\tau,i,j}(X) = E_K(X \oplus \Delta_{N,\tau,i,j}) \oplus \Delta_{N,\tau,i,j}, \quad \text{or} \quad \widetilde{E}_K^{\tau,i,j}(X) = E_K(X \oplus \Delta_{\tau,i,j}).$$

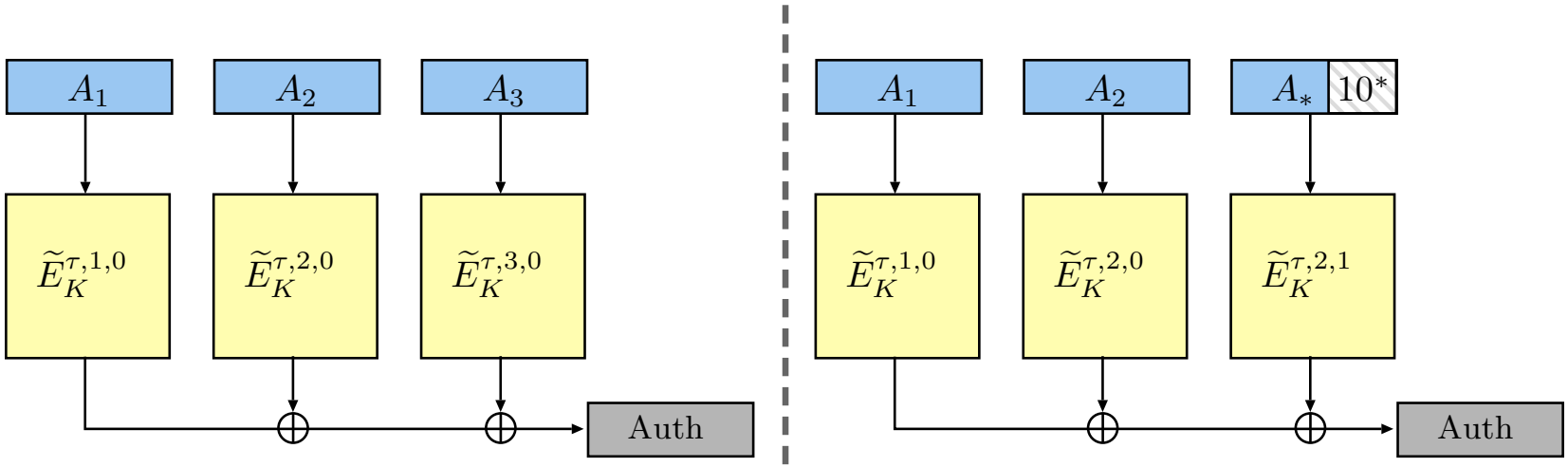
Achieving nvAE security

vΘCB



Achieving nvAE security

vΘCB



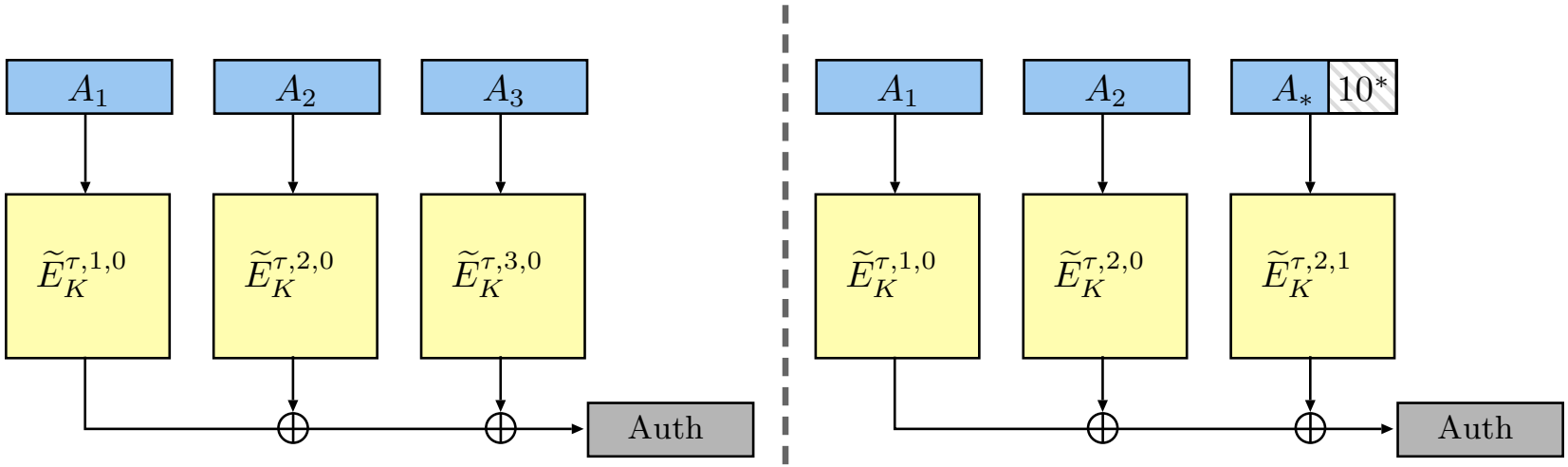
With a suitable tweakable blockcipher $\widetilde{E}$

- With modified XEX (small impact on performance):

$$\mathbf{Adv}^{nvae(\tau_c)}_{\mathrm{vOCB}[E]}(t, \mathbf{q_e}, \mathbf{q_d}, \sigma) \leq (|\mathcal{I}_T| + 2) \cdot \mathbf{Adv}^{\pm prp}_{E}(t', 2q) + \frac{28.5q^2}{2^n} + q_d^{\tau_c} \cdot \frac{2^{n-\tau_c}}{2^n - 1}$$

Achieving nvAE security

vΘCB



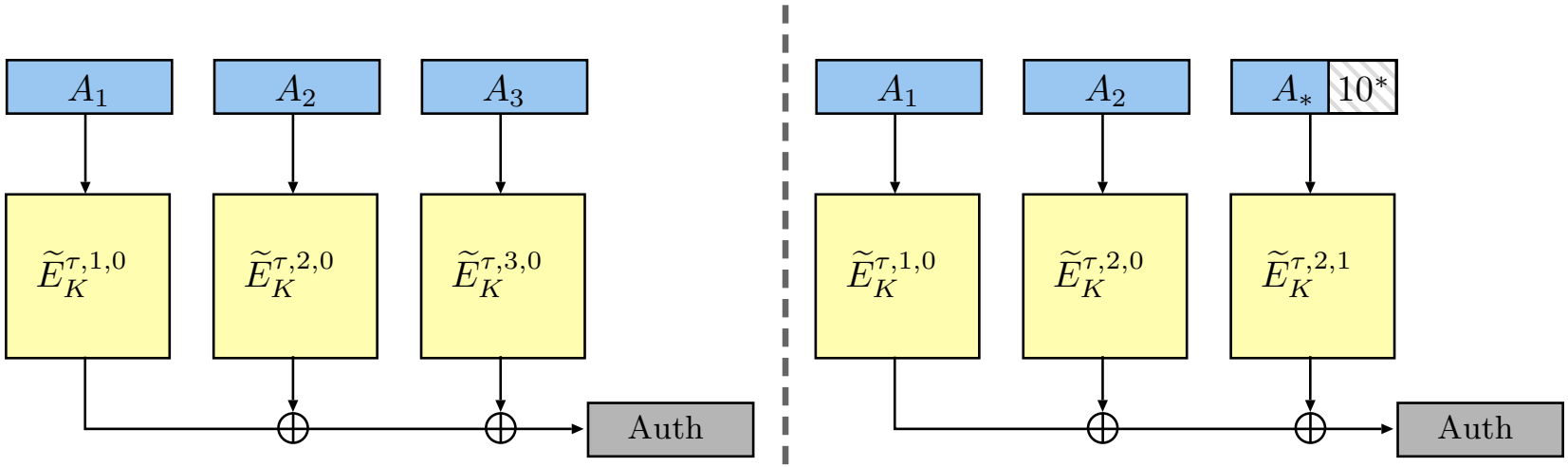
With a suitable tweakable blockcipher $\widetilde{E}$

- With modified XEX (small impact on performance):

$$\mathbf{Adv}^{nvae(\tau_c)}_{\mathrm{vOCB}[E]}(t, \mathbf{q_e}, \mathbf{q_d}, \sigma) \leq (|\mathcal{I}_T| + 2) \cdot \mathbf{Adv}^{\pm prp}_{E}(t', 2q) + \frac{28.5q^2}{2^n} + q_d^{\tau_c} \cdot \frac{2^{n-\tau_c}}{2^n - 1}$$

Achieving nvAE security

vΘCB



With a suitable tweakable blockcipher $\widetilde{E}$

- With modified XEX (small impact on performance):

$$\mathbf{Adv}^{nvae(\tau_c)}_{\mathrm{vOCB}[E]}(t, \mathbf{q_e}, \mathbf{q_d}, \sigma) \leq (|\mathcal{I}_T| + 2) \cdot \mathbf{Adv}^{\pm prp}_{E}(t', 2q) + \frac{28.5q^2}{2^n} + q_d^{\tau_c} \cdot \frac{2^{n-\tau_c}}{2^n - 1}$$