Authenticated Encryption with Variable Stretch

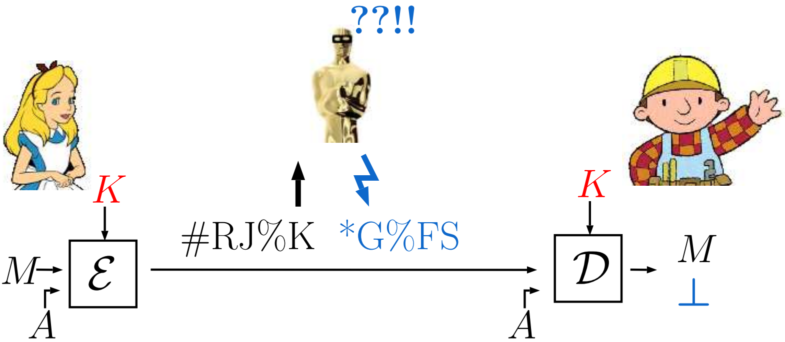
Reza Reyhanitabar[1] Serge Vaudenay[2] Damian Vizár[2]

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

DIAC 2016: Directions in Authenticated Ciphers 2016

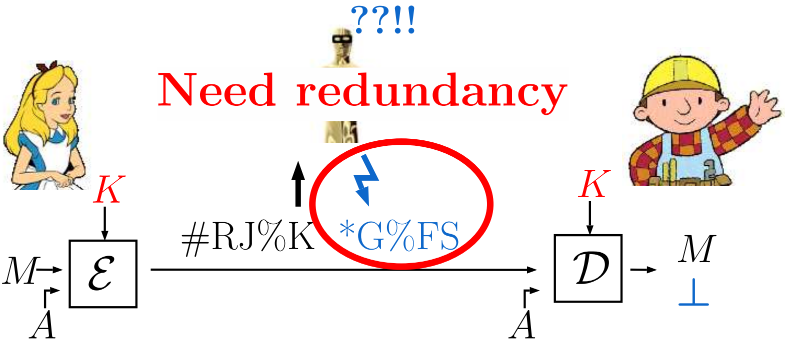
This work was partially supported by Microsoft Research

Authenticated Encryption



- **Confidentiality+Authenticity/Integrity for M**
  [Bellare,Namprempre 00],[Katz,Yung 00]
- **Authenticity for A**
  [Rogaway 02]

Authenticated Encryption

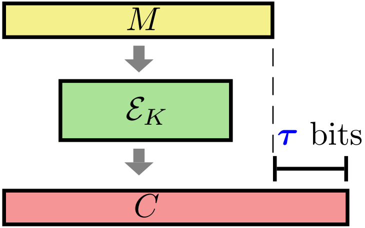


- **Confidentiality+Authenticity/Integrity for M**
  [Bellare,Namprempre 00],[Katz,Yung 00]
- **Authenticity for A**
  [Rogaway 02]

Ciphertext Epxansion

a.k.a. Stretch

Redundancy in AE: **ciphertext expansion**



Ciphertext expanded by τ bits

$\Rightarrow$ **Expected cost of forgery: $\approx 2^{\tau}$ queries**

How to Stretch?

w.r.t. the Syntax of Security Notions

- **Group 1: (Mostly) constant τ, parameter of the scheme**
  - **nAE** [Rogaway, Bellare, Black, Krovetz 01]
  - **AEAD** [Rogaway 02]
  - **DAE** and **MRAE** [Rogaway, Shrimpton 06]
  - **OAE** [Fleischmann, Forler, Lucks 12]
  - **AE-RUP** [Andreeva, Bogdanov, Luykx, Mennink, Mouha, Yasuda 14]
  - **OAE2** [Hoang, Reyhanitabar, Rogaway, V 15]

- **Group 2: User-selectable τ per query**
  - **RAE** [Hoang, Krovetz, Rogaway 15]

How to Stretch?

w.r.t. the Syntax of Security Notions

- **Group 1: (Mostly) constant τ, parameter of the scheme**
  - **nAE** [Rogaway, Bellare, Black, Krovetz 01]
  - **AEAD** [Rogaway 02]
  - **DAE** and **MRAE** [Rogaway, Shrimpton 06]
  - **OAE** [Fleischmann, Forler, Lucks 12]
  - **AE-RUP** [Andreeva, Bogdanov, Luykx, Mennink, Mouha, Yasuda 14]
  - **OAE2** [Hoang, Reyhanitabar, Rogaway, V 15]

  ▶ Different tag lengths $\Rightarrow$ independent keys

- **Group 2: User-selectable τ per query**
  - **RAE** [Hoang, Krovetz, Rogaway 15]

  ▶ "Best possible security", hard to achieve
  ▶ Cannot be "online"
  ▶ Complicated, difficult to implement

Stretch-Misuse

- **Group 1: Constant τ, parameter of the scheme**
  - **nAE**
  - **AEAD**
  - **DAE** and **MRAE**
  - **OAE**
  - **AE-RUP**
  - **OAE2**

Stretch-Misuse

- **Group 1: Constant τ, parameter of the scheme**
  - **nAE**
  - **AEAD**
  - **DAE** and **MRAE**
  - **OAE**
  - **AE-RUP**
  - **OAE2**

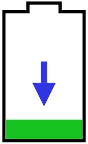
What happens if stretch is (mis)treated as a user input?

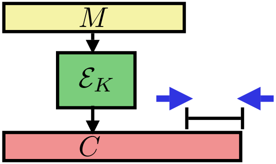
Stretch-Misuse

Why Should we Consider It?

Because it is tempting:

- Handling multiple keys is annoying
- "Sliding-scale" authenticity as a feature
  - (τ bits of stretch $\Rightarrow$ τ bits of authenticity *for individual messages*)
  - E.g. moderate τ_1 for most messages and huge τ_2 for critical
- Saving resources in constrained systems
  - E.g. sensor nodes: wireless communication is expensive
  - Reducing security to increase battery life (key exchange way too expensive)



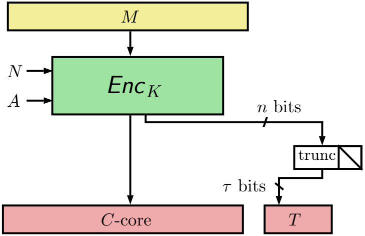


Stretch-Misuse

Why Should we Consider It?

Because it is easy to do:

- Most often: a default authentication tag that is truncated

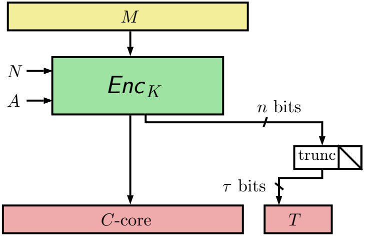


Stretch-Misuse

Why Should we Consider It?

Because it is easy to do:

- Most often: a default authentication tag that is truncated



Because it is a matter of "when", not "if" a misuse occurs

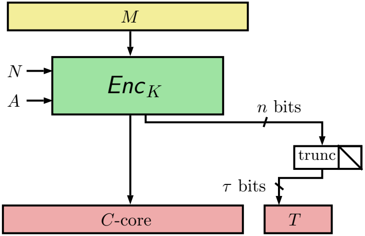
- Past examples of this for other misuses

Stretch-Misuse

Why Should we Consider It?

Because it is easy to do:

- Most often: a default authentication tag that is truncated

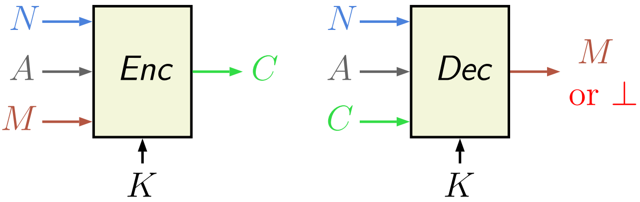


Because it is a matter of "when", not "if" a misuse occurs

- Past examples of this for other misuses

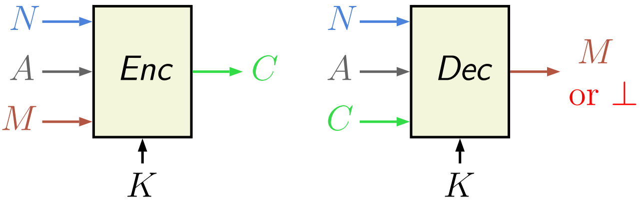
. . . and because there are attacks

Nonce-based AE with Associated Data (AEAD)



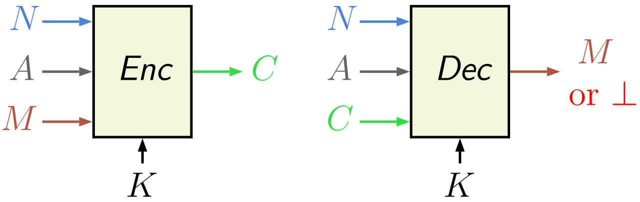
- *Enc*, *Dec*: deterministic algorithms
- *N*: Nonce (public message number) that must not repeat
- *A*: Associated Data that must be authenticated, but not encrypted
- *M*: Plaintext that must be encrypted and authenticated
- *C*: Ciphertext (stretched by τ bits)
- *K*: Secret key

Nonce-based AE with Associated Data (AEAD)



- *Enc*, *Dec*: deterministic algorithms
- *N*: Nonce (public message number) that must not repeat
- *A*: Associated Data that must be authenticated, but not encrypted
- *M*: Plaintext that must be encrypted and authenticated
- *C*: Ciphertext (stretched by τ bits)
- *K*: Secret key

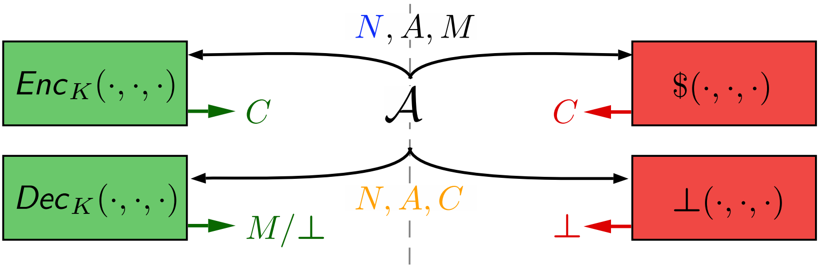
Nonce-based AE with Associated Data (AEAD)



- *Enc*, *Dec*: deterministic algorithms
- *N*: Nonce (public message number) that must not repeat
- *A*: Associated Data that must be authenticated, but not encrypted
- *M*: Plaintext that must be encrypted and authenticated
- *C*: Ciphertext (stretched by τ bits)
- *K*: Secret key

Nonce-based AE with Associated Data

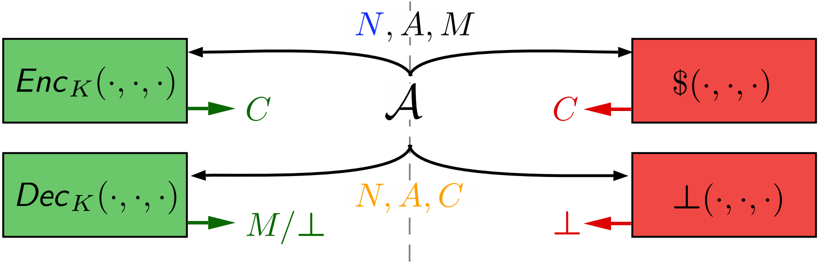
N never repeats, (*N*, *A*, *C*) not trivially correct:



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

Nonce-based AE with Associated Data

N never repeats, (N, A, C) not trivially correct:

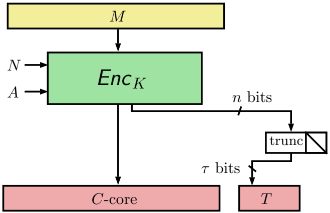


…and the ciphertext expansion is assumed to be constant

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

Trivial Tag Length-Variation Attack on AEAD

"Versions of OCB with different tag lengths exist, tag truncation trivially correct if used under same key" [Manger 13, CFRG discussion]



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

Obvious property, but ...

... **contradicts the intuition of τ-bit resistance to forgery**

Trivial Tag Length-Variation Attack on AEAD

“Would it be better if the algorithms with different tag lengths could not affect each other?”

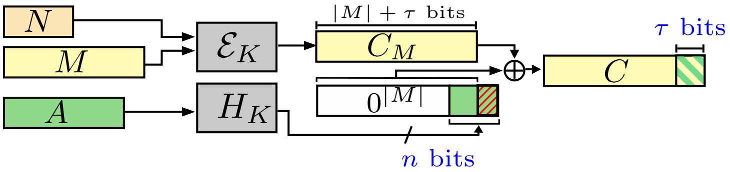
Probably! **Ad-hoc solutions proposed:**

- OCB adopts fix proposed by Manger: “just drop the tag length into the nonce”
- Nandi proposes to do the same with AD
- CLOC&SILC, OTR and OMD heuristically tweaked for round 2 of CAESAR competition

Gradual Forgery for Ciphertext Translation

Ciphertext Translation

Message-only core + AD-“hash”



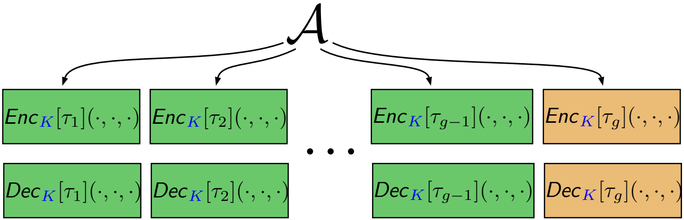
- message-ciphertext already “looks random”
- H_K can be AXU

Gradual Forgery for Ciphertext Translation

The Attack

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

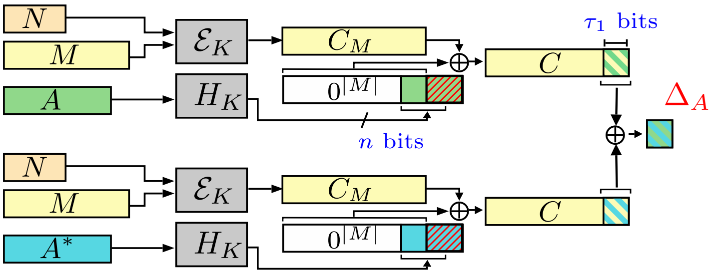
- Access to Enc and Dec oracles with stretch $\tau_1 < \tau_2 < \ldots < \tau_g$ using the same key, scheme with ciphertext translation structure
- **Forgery for N, A^*, M with τ_g bits of stretch**



Gradual Forgery for Ciphertext Translation

The Attack

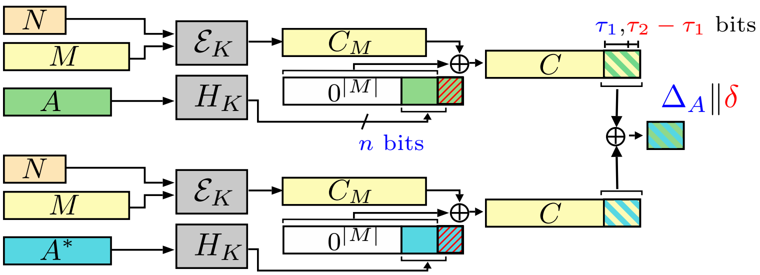
1. Pick some $\mathbf{A} \neq \mathbf{A}^*$
2. Get $\mathbf{C} \| \mathbf{T} \leftarrow \mathbf{Enc}[\tau_1](\mathbf{N}, \mathbf{A}, \mathbf{M})$
3. Find $\delta \in \{0, 1\}^{\tau_1}$ s.t. $\mathbf{Dec}[\tau_1](\mathbf{N}, \mathbf{A}^*, \mathbf{C} \| (\mathbf{T} \oplus \delta))$ **succeeds**
4. Set $\mathbf{\Delta_A} \leftarrow \delta$ $\quad \triangleright\ \Delta_A = \mathrm{trunc}(H_K(A) \oplus H_K(A^*), \tau_1)$



Gradual Forgery for Ciphertext Translation

The Attack

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. $\mathbf{Dec}[\tau_g](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \mathbf{\Delta_A}\|\delta))$ **succeeds**
7. Set $\mathbf{\Delta_A} \leftarrow \mathbf{\Delta_A}\|\delta$



Gradual Forgery for Ciphertext Translation

The Attack

...

f Get $\mathbf{C}\|\mathbf{T} \leftarrow \mathbf{Enc}[\tau_\mathbf{g}](\mathbf{N}, \mathbf{A}, \mathbf{M})$

i Find $\delta \in \{\mathbf{0}, \mathbf{1}\}^{\tau_{\mathbf{g}-1} - \tau_\mathbf{g}}$ s.t. $\mathbf{Dec}[\tau_\mathbf{g}](\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{T} \oplus \delta))$ **succeeds**

n Output forgery $\mathbf{N}, \mathbf{A}^*, \mathbf{C}\|(\mathbf{\Delta_A}\|\delta)$

Gradual Forgery for Ciphertext Translation

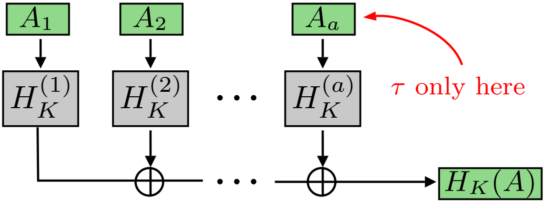
Complexity

- single encryption query per stretch
- $2^{\tau_i - \tau_{i-1}}$ decryption queries stretched by τ_i bits for $1 < i \leq \ell$
- 2^{τ_1} decryption queries stretched by τ_1
- **Forgery for τ_g bits of stretch with $2^{\tau_g - \tau_{g-1}}$ decryption queries stretched by τ_g bits** versus the intuition of τ_g bit security

E.g. if $\mathcal{I}_T = \{32, 64, 96, 128\}$, then forging a 128-bit tag takes $4 \cdot 2^{32}$ decryption queries in total

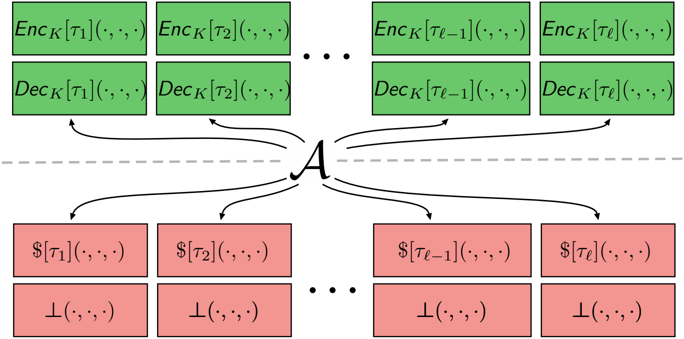
Gradual Forgery for Ciphertext Translation
Applicability

- 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



Capturing AEAD Security with Variable Tags

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

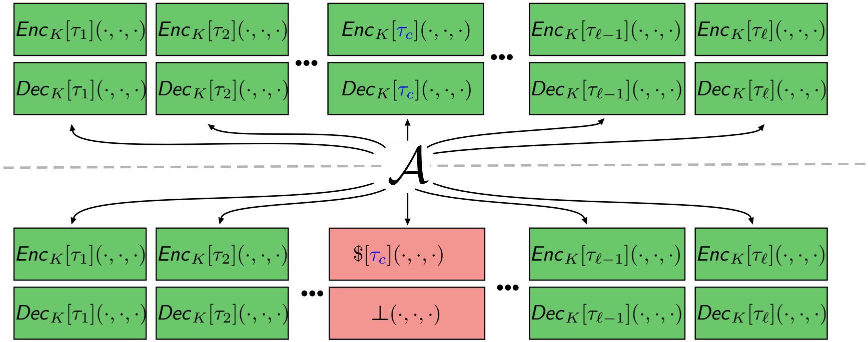


- **$\mathcal{A}$ can always win with $2^{\min \mathcal{I}_T}$ queries** (conservative evaluation)
- **Interactions between stretches not captured**

Capturing AEAD Security with Variable Tags: *nvae*(τ_c)

fixed but arbitrary "challenge" stretch τ_c:

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



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

nvae(τ_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, flexibility and generality

nvae(τ_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, flexibility and 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

nvae(τ_c)

Capturing AEAD Security with Variable tags?

- **Only distinguishable by queries stretched by τ_c**
  - E.g. forging with $\min \mathcal{I}_T$ bits of stretch alone does not help
- **Queries stretched by $\tau \neq \tau_c$ bits can still help**
  - Both truncation and gradual forgery attacks advantage$= 1$
  - Truncation: single decryption with stretch τ_c
  - Gradual: resources depend on <u>other stretch values</u>

nvae(τ_c)

Capturing AEAD Security with Variable tags?

- **Only distinguishable by queries stretched by τ_c**
  - E.g. forging with min $\mathcal{I}_T$ bits of stretch alone does not help
- **Queries stretched by $\tau \neq \tau_c$ bits can still help**
  - Both truncation and gradual forgery attacks advantage$= 1$
  - Truncation: single decryption with stretch τ_c
  - Gradual: resources depend on other stretch values

Good advantage?

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

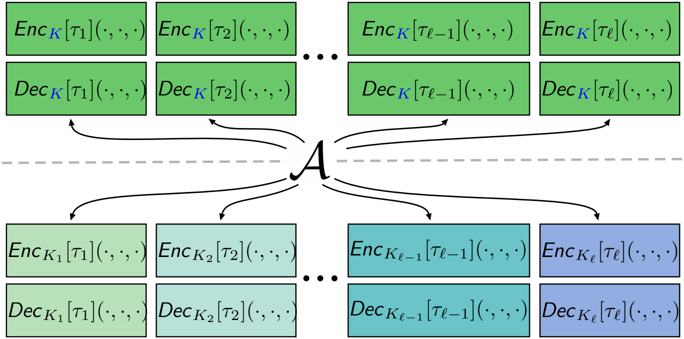
“small” due to construction, no direct dependence on τ_c

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

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



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

Achieving nvAE Modularly

Key-Equivalent Separation by Stretch

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

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

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 $(q_d^\tau | \tau \in \mathcal{I}_T)$
- $\boldsymbol{\sigma}$ 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

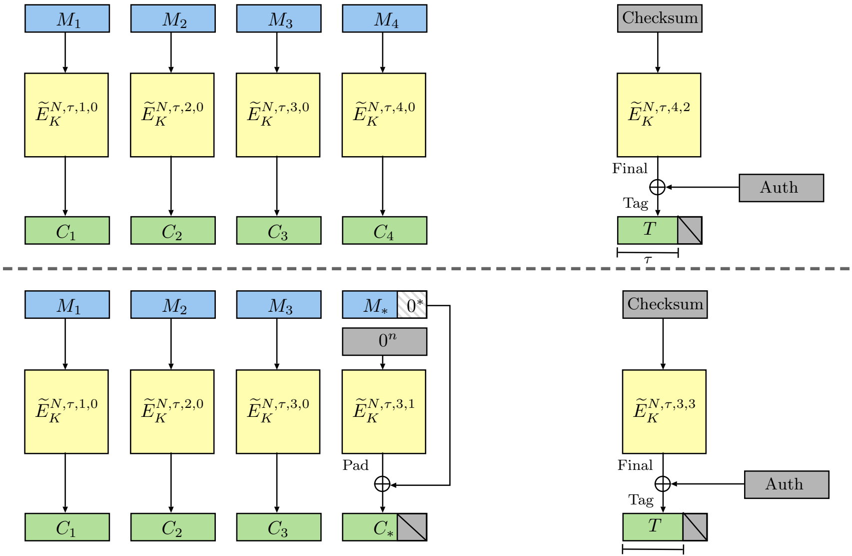
- Add τ as tweak component **in all tweaks**
- Show kess security (easy with TBC!)
- AE security inherited

Beyond proof of concept:

- Modification independent of scheme
- kess security easy to show
- nvAE security automatic
- Can treat also OTR, Deoxys etc.

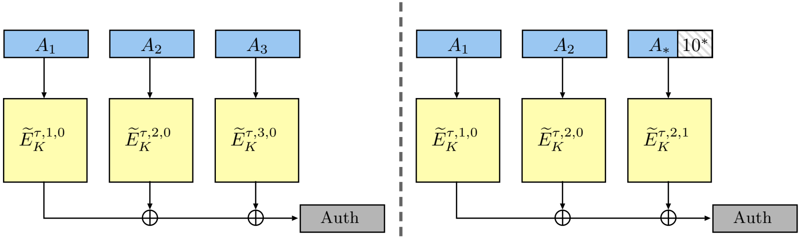
Achieving nvAE security

vΘCB



Achieving nvAE security

vΘCB



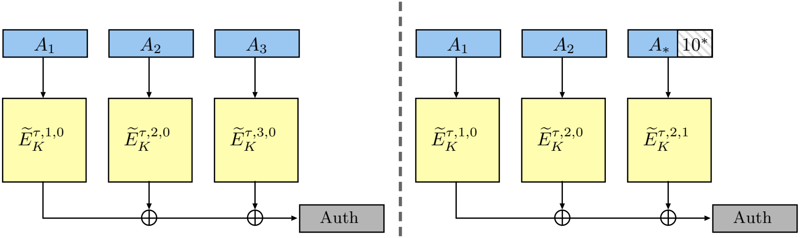
With a suitable tweakable blockcipher $\widetilde{E}$

- With modified XEX (small impact on performance):

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

Achieving nvAE security

vΘCB



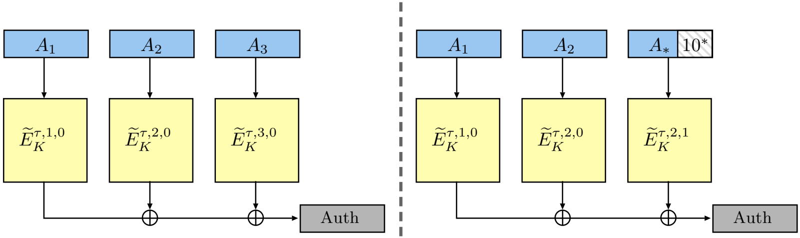
With a suitable tweakable blockcipher $\widetilde{E}$

- With modified XEX (small impact on performance):

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

Achieving nvAE security

v⊖CB



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

- With modified XEX (small impact on performance):

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

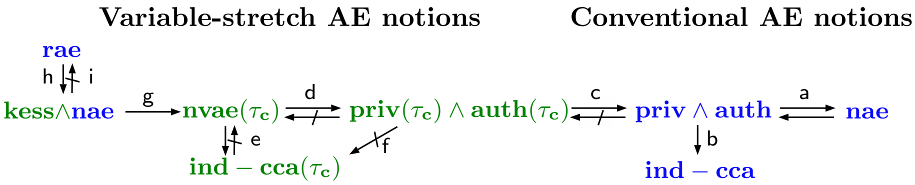
Conclusions

- AEAD schemes “insecure” with variable stretch
  - Even with ad-hoc counter measures
- We define what it means to be secure ☺
- We determine relations with existing notions ☺ (backup slide!)
- We show that
  - nvAE security can be achieved ☺
  - Schemes based on tweakable primitives easily patched ☺
- Other schemes? ☹
  - Other classes of schemes easily fixed, e.g. **encoding τ in nonce works for sponges**
  - Generic transformation: open problem

Questions?

Thank you for your attention!

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:

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