Committing Authenticated Encryption

Generic Composition, NIST LWC Finalists, and Zero-Padding

Patrick Struck

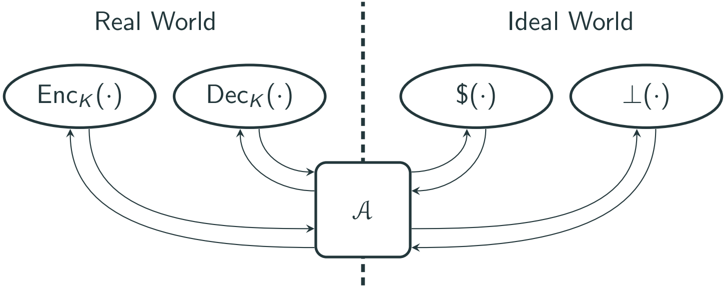
GAPS, September 2025

University of Konstanz
based on joint work
with Maximiliane Weishäupl (ToSC 2024 Issue 1) and
with Juliane Krämer and Maximiliane Weishäupl (ToSC 2024 Issue 4)

Motivation

An authenticated encryption scheme is deemed secure if:

1. an adversary cannot learn anything about the message from a ciphertext
2. an adversary cannot forge a valid ciphertext



Motivation

Attacks have shown that we sometimes require more properties from an AE scheme

- Fast message franking attack[1]
- Subscribe with Google attack[2]
- Partitioning oracle attack[3]
- possibly more attacks in the future

[1] Dodis et al. **"Fast Message Franking: From Invisible Salamanders to Encryptment".** In: *CRYPTO 2018*. 2018.
[2] Albertini et al. **"How to Abuse and Fix Authenticated Encryption Without Key Commitment".** In: *USENIX 2022*. 2022.
[3] Len, Grubbs, and Ristenpart. **"Partitioning Oracle Attacks".** In: *USENIX 2021*. 2021.

Motivation

Partitioning Oracle Attack:

Assume that $\mathcal{A}$ has a list of leaked keys which contains the correct key

Motivation

Partitioning Oracle Attack:

Assume that $\mathcal{A}$ has a list of leaked keys which contains the correct key

- Split the list into two sub-lists L_1 and L_2
- Find a ciphertext that decrypts validly under the keys in L_1 and to $\perp$ under the keys in L_2

Motivation

Partitioning Oracle Attack:

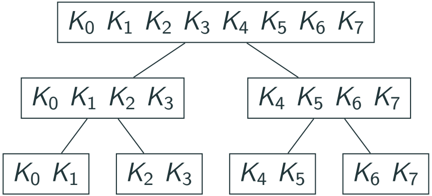
Assume that $\mathcal{A}$ has a list of leaked keys which contains the correct key

- Split the list into two sub-lists L_1 and L_2
- Find a ciphertext that decrypts validly under the keys in L_1 and to $\perp$ under the keys in L_2
- Based on the response, $\mathcal{A}$ knows if the correct key is in L_1 or L_2
- repeat using the list containing the correct key

Partitioning Oracle Attack:

Assume that $\mathcal{A}$ has a list of leaked keys which contains the correct key

- Split the list into two sub-lists L_1 and L_2
- Find a ciphertext that decrypts validly under the keys in L_1 and to $\perp$ under the keys in L_2
- Based on the response, $\mathcal{A}$ knows if the correct key is in L_1 or L_2
- repeat using the list containing the correct key



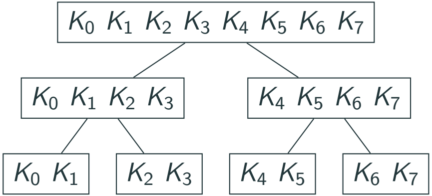
Motivation

Partitioning Oracle Attack:

Assume that $\mathcal{A}$ has a list of leaked keys which contains the correct key

- Split the list into two sub-lists L_1 and L_2
- Find a ciphertext that decrypts validly under the keys in L_1 and to $\perp$ under the keys in L_2
- Based on the response, $\mathcal{A}$ knows if the correct key is in L_1 or L_2
- repeat using the list containing the correct key

Problem: $\mathcal{A}$ can construct ciphertexts that decrypt under multiple keys



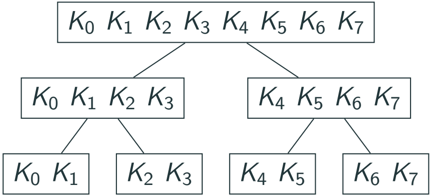
Motivation

Partitioning Oracle Attack:

Assume that $\mathcal{A}$ has a list of leaked keys which contains the correct key

- Split the list into two sub-lists L_1 and L_2
- Find a ciphertext that decrypts validly under the keys in L_1 and to $\perp$ under the keys in L_2
- Based on the response, $\mathcal{A}$ knows if the correct key is in L_1 or L_2
- repeat using the list containing the correct key

Problem: $\mathcal{A}$ can construct ciphertexts that decrypt under multiple keys
Solution: committing security



Motivation

Game $\mathsf{CMT_K}$

$(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M}) \leftarrow \mathcal{A}()$
if $K = \overline{K}$
  return 0
$(C, T) \leftarrow \textsc{Enc}(K, N, A, M)$
$(\overline{C}, \overline{T}) \leftarrow \textsc{Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M})$
return $((C, T) = (\overline{C}, \overline{T}))$

Game CMT

$(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M}) \leftarrow \mathcal{A}()$
if $(K, N, A) = (\overline{K}, \overline{N}, \overline{A})$
  return 0
$(C, T) \leftarrow \textsc{Enc}(K, N, A, M)$
$(\overline{C}, \overline{T}) \leftarrow \textsc{Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M})$
return $((C, T) = (\overline{C}, \overline{T}))$

Security games $\mathsf{CMT_K}$ (left) and CMT (right).

Generic Composition

Generic Composition: Overview

There are three methods of generic composition:

1. Encrypt-and-MAC
2. Encrypt-then-MAC
3. MAC-then-Encrypt

[4] Namprempre, Rogaway, and Shrimpton. **“Reconsidering Generic Composition”.** In: *EUROCRYPT 2014*. 2014.

Generic Composition: Overview

There are three methods of generic composition:

1. Encrypt-and-MAC
2. Encrypt-then-MAC
3. MAC-then-Encrypt

We focus on the so-called N-schemes[4]

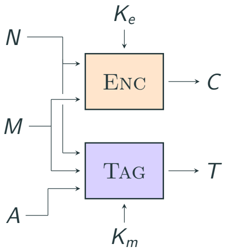
- construct an AE scheme from a nonce-based encryption scheme and a MAC

---

[4] Namprempre, Rogaway, and Shrimpton. **"Reconsidering Generic Composition".** In: *EUROCRYPT 2014*. 2014.

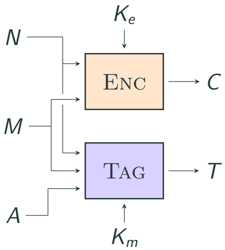
Generic Composition: Overview

N1 (Encrypt-and-MAC)

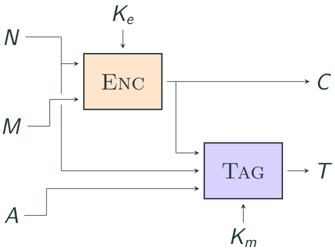


Generic Composition: Overview

N1 (Encrypt-and-MAC)

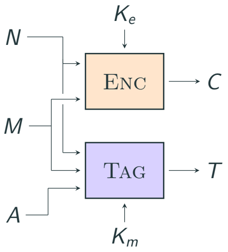


N2 (Encrypt-then-MAC)

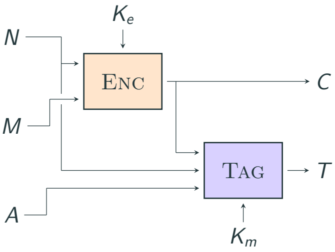


Generic Composition: Overview

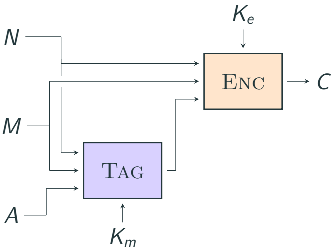
N1 (Encrypt-and-MAC)



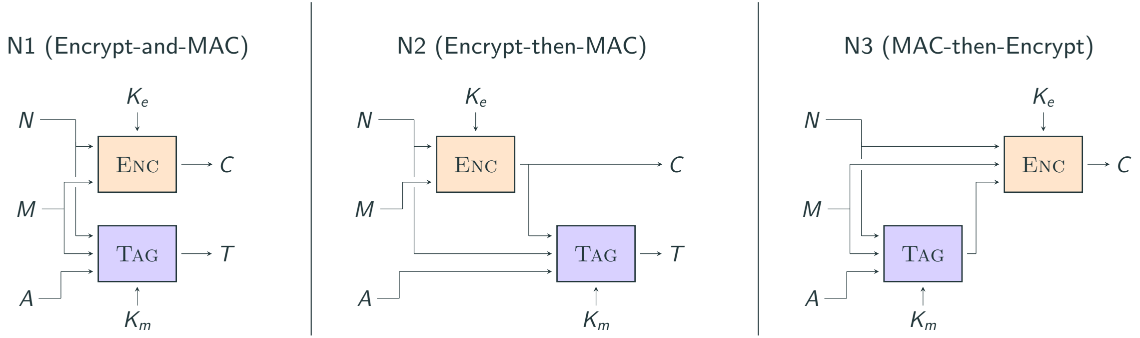
N2 (Encrypt-then-MAC)



N3 (MAC-then-Encrypt)



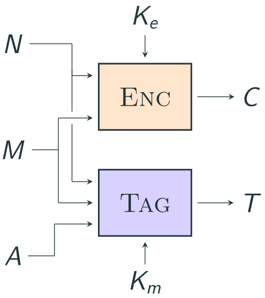
Generic Composition: Overview



We give positive results for N1 and negative results for N2

Committing Security of N1 (Encrypt-and-MAC)

N1 (Encrypt-and-MAC)



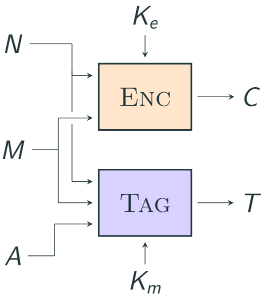
Theorem (Committing Security of N1)

Let SE *be a symmetric encryption scheme and* MAC *be a MAC. Let further* $\mathrm{N1}[\mathrm{SE}, \mathrm{MAC}]$ *be the authenticated encryption scheme obtained via the N1 construction using* SE *and* MAC*. Then for any adversary* $\mathcal{A}$ *there exist adversaries* $\mathcal{B}$ *and* $\mathcal{C}$ *such that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\mathrm{N1}[\mathrm{SE},\mathrm{MAC}]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{wCR}}_{\mathrm{SE}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{CR}}_{\mathrm{MAC}}(\mathcal{C}) \,.$$

Committing Security of N1 (Encrypt-and-MAC)

N1 (Encrypt-and-MAC)



Theorem (Committing Security of N1)

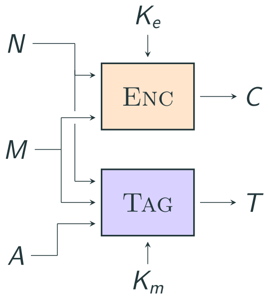
Let $\textsc{Se}$ *be a symmetric encryption scheme and* $\textsc{Mac}$ *be a MAC. Let further* $\mathrm{N1}[\textsc{Se}, \textsc{Mac}]$ *be the authenticated encryption scheme obtained via the N1 construction using* $\textsc{Se}$ *and* $\textsc{Mac}$*. Then for any adversary* $\mathcal{A}$ *there exist adversaries* $\mathcal{B}$ *and* $\mathcal{C}$ *such that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\mathrm{N1}[\textsc{Se},\textsc{Mac}]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{wCR}}_{\textsc{Se}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{CR}}_{\textsc{Mac}}(\mathcal{C}) .$$

Committing security of N1 reduces to collision resistance of the underlying MAC and a weak form of collision resistance of the underlying encryption

Committing Security of N1 (Encrypt-and-MAC)

N1 (Encrypt-and-MAC)

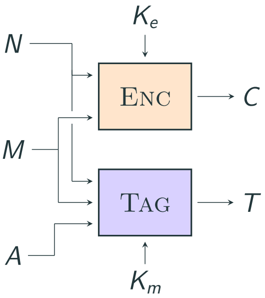


Game CR	Game wCR
$(K, X), (\overline{K}, \overline{X}) \leftarrow \mathcal{A}()$	$(K, N, M), (\overline{K}, \overline{N}, \overline{M}) \leftarrow \mathcal{A}()$
if $(K, X) = (\overline{K}, \overline{X})$	**if** $K = \overline{K} \vee (N, M) \neq (\overline{N}, \overline{M})$
return 0	**return** 0
$T \leftarrow \textsc{Tag}(K, X)$	$C \leftarrow \textsc{Enc}(K, N, M)$
$\overline{T} \leftarrow \textsc{Tag}(\overline{K}, \overline{X})$	$\overline{C} \leftarrow \textsc{Enc}(\overline{K}, \overline{N}, \overline{M})$
return $(T = \overline{T})$	**return** $(C = \overline{C})$

Security game CR for MACs and wCR for symmetric encryption.

Committing Security of N1 (Encrypt-and-MAC)

N1 (Encrypt-and-MAC)



Game CR	Game wCR
$(K, X), (\overline{K}, \overline{X}) \leftarrow \mathcal{A}()$	$(K, N, M), (\overline{K}, \overline{N}, \overline{M}) \leftarrow \mathcal{A}()$
if $(K, X) = (\overline{K}, \overline{X})$	**if** $K = \overline{K} \vee (N, M) \neq (\overline{N}, \overline{M})$
return 0	**return** 0
$T \leftarrow \textsc{Tag}(K, X)$	$C \leftarrow \textsc{Enc}(K, N, M)$
$\overline{T} \leftarrow \textsc{Tag}(\overline{K}, \overline{X})$	$\overline{C} \leftarrow \textsc{Enc}(\overline{K}, \overline{N}, \overline{M})$
return $(T = \overline{T})$	**return** $(C = \overline{C})$

Security game CR for MACs and wCR for symmetric encryption.

Weak collision-resistant encryption: adversary needs to find distinct keys and one nonce-message pair that result in the same ciphertext

- finding arbitrary collisions (for tidy encryption schemes) is easy

Committing Security of N1 (Encrypt-and-MAC)

Proof idea: for the output $((K_e, K_m), N, A, M), ((\overline{K}_e, \overline{K}_m), \overline{N}, \overline{A}, \overline{M})$ by $\mathcal{A}$, distinguish between the following cases:

Committing Security of N1 (Encrypt-and-MAC)

Proof idea: for the output $((K_e, K_m), N, A, M), ((\overline{K}_e, \overline{K}_m), \overline{N}, \overline{A}, \overline{M})$ by $\mathcal{A}$, distinguish between the following cases:

1. $K_e \neq \overline{K}_e \wedge (N, M) = (\overline{N}, \overline{M})$**:**

In this case, $\mathcal{A}$ breaks wCR security of the underlying encryption

Committing Security of N1 (Encrypt-and-MAC)

Proof idea: for the output $((K_e, K_m), N, A, M), ((\overline{K}_e, \overline{K}_m), \overline{N}, \overline{A}, \overline{M})$ by $\mathcal{A}$, distinguish between the following cases:

1. **$K_e \neq \overline{K}_e \wedge (N, M) = (\overline{N}, \overline{M})$:**
   In this case, $\mathcal{A}$ breaks wCR security of the underlying encryption
2. **$K_e = \overline{K}_e \vee (N, M) \neq (\overline{N}, \overline{M})$:**
   In this case, it holds that $(K_m, N, A, M) \neq (\overline{K}_m, \overline{N}, \overline{A}, \overline{M})$ which implies that $\mathcal{A}$ breaks CR security of the MAC

Committing Security of N1 (Encrypt-and-MAC)

Are there schemes that satisfy the required properties?

[5] Degabriele, Janson, and Struck. **"Sponges Resist Leakage: The Case of Authenticated Encryption".** In: *ASIACRYPT 2019*. 2019.

Committing Security of N1 (Encrypt-and-MAC)

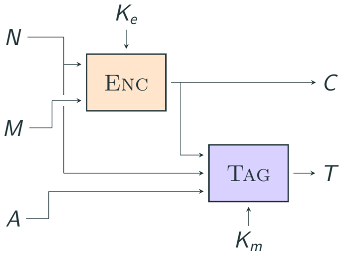
Are there schemes that satisfy the required properties?

We show that the encryption scheme and MAC of $\textsc{Slae}$[5] (a derivate of $\textsc{Isap}$) achieve wCR and CR, respectively

[5]Degabriele, Janson, and Struck. **"Sponges Resist Leakage: The Case of Authenticated Encryption".** In: *ASIACRYPT 2019*. 2019.

Committing Attack against N2 (Encrypt-then-MAC)

N2 (Encrypt-then-MAC)



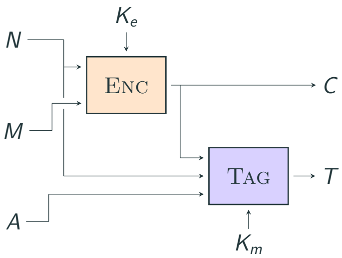
Theorem (Committing Security of N2)

Let $\textsc{Se}$ *be a symmetric encryption scheme and* $\textsc{Mac}$ *be a MAC. Let further* $\mathrm{N2}[\textsc{Se}, \textsc{Mac}]$ *be the authenticated encryption scheme obtained via the N2 construction using* $\textsc{Se}$ *and* $\textsc{Mac}$*. Then there exists an adversary* $\mathcal{A}$ *such that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\mathrm{N2}[\textsc{Se},\textsc{Mac}]}(\mathcal{A}) = 1\,.$$

Committing Attack against N2 (Encrypt-then-MAC)

N2 (Encrypt-then-MAC)



Theorem (Committing Security of N2)

Let $\textsc{Se}$ *be a symmetric encryption scheme and* $\textsc{Mac}$ *be a MAC. Let further* $\mathrm{N2}[\textsc{Se}, \textsc{Mac}]$ *be the authenticated encryption scheme obtained via the N2 construction using* $\textsc{Se}$ *and* $\textsc{Mac}$*. Then there exists an adversary* $\mathcal{A}$ *such that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\mathrm{N2}[\textsc{Se},\textsc{Mac}]}(\mathcal{A}) = 1\,.$$

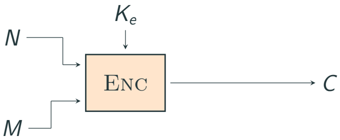
Gist: since N2 authenticates the ciphertext (not the message like N1), finding a ciphertext collision is sufficient

- not a restricted collision as was the case for N1

Committing Attack against N2 (Encrypt-then-MAC)

Adversary $\mathcal{A}$

Committing Attack against N2 (Encrypt-then-MAC)

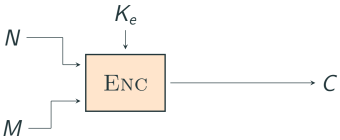


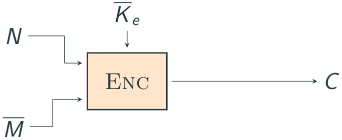
Adversary $\mathcal{A}$

$K_e, N, M \leftarrow_{\$} \mathcal{K} \times \mathcal{N} \times \mathcal{M}$

$C \leftarrow \textsc{Se.Enc}(K_e, N, M)$

Committing Attack against N2 (Encrypt-then-MAC)





Adversary $\mathcal{A}$

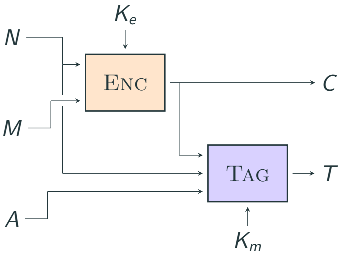
$K_e, N, M \leftarrow\$ \mathcal{K} \times \mathcal{N} \times \mathcal{M}$

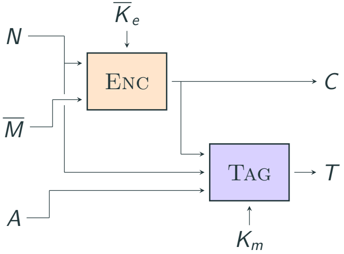
$C \leftarrow \mathrm{SE.Enc}(K_e, N, M)$

$\overline{K}_e \leftarrow\$ \mathcal{K} \setminus \{K_e\}$

$\overline{M} \leftarrow \mathrm{SE.Dec}(\overline{K}_e, N, C)$ // by tidyness: $C = \mathrm{SE.Enc}(\overline{K}_e, N, \overline{M})$

Committing Attack against N2 (Encrypt-then-MAC)





Adversary $\mathcal{A}$

$K_e, N, M \leftarrow\$\ \mathcal{K} \times \mathcal{N} \times \mathcal{M}$

$C \leftarrow \textsc{Se.Enc}(K_e, N, M)$

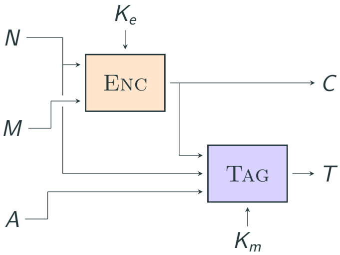
$\overline{K}_e \leftarrow\$\ \mathcal{K} \setminus \{K_e\}$

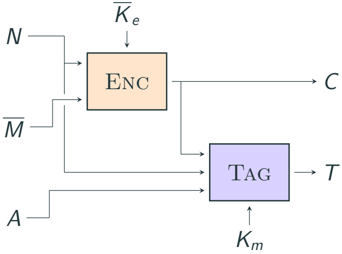
$\overline{M} \leftarrow \textsc{Se.Dec}(\overline{K}_e, N, C)$ // by tidyness: $C = \textsc{Se.Enc}(\overline{K}_e, N, \overline{M})$

$(K_m, A) \leftarrow\$\ \mathcal{K} \times \mathcal{A}$

return $((K_e, K_m), N, A, M), ((\overline{K}_e, K_m), N, A, \overline{M})$

Committing Attack against N2 (Encrypt-then-MAC)





Adversary $\mathcal{A}$

$K_e, N, M \leftarrow_{\$} \mathcal{K} \times \mathcal{N} \times \mathcal{M}$

$C \leftarrow \mathrm{SE.ENC}(K_e, N, M)$

$\overline{K}_e \leftarrow_{\$} \mathcal{K} \setminus \{K_e\}$

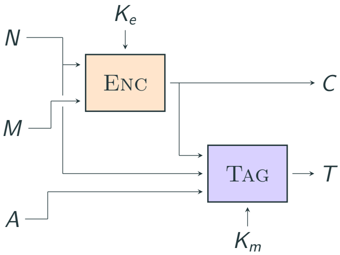
$\overline{M} \leftarrow \mathrm{SE.DEC}(\overline{K}_e, N, C)$ // by tidyness: $C = \mathrm{SE.ENC}(\overline{K}_e, N, \overline{M})$

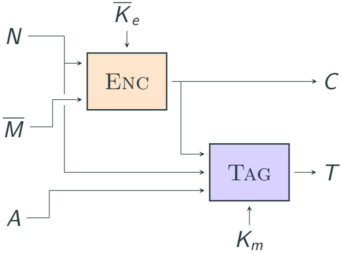
$(K_m, A) \leftarrow_{\$} \mathcal{K} \times \mathcal{A}$

return $((K_e, K_m), N, A, M), ((\overline{K}_e, K_m), N, A, \overline{M})$

- Attack exploits independent keys for underlying encryption scheme and MAC

Committing Attack against N2 (Encrypt-then-MAC)





Adversary $\mathcal{A}$

$K_e, N, M \leftarrow\$ \mathcal{K} \times \mathcal{N} \times \mathcal{M}$

$C \leftarrow \text{SE.ENC}(K_e, N, M)$

$\overline{K}_e \leftarrow\$ \mathcal{K} \setminus \{K_e\}$

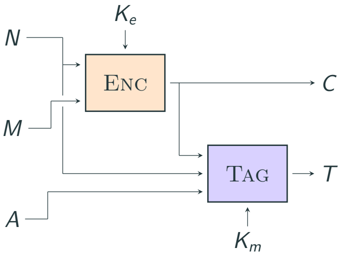
$\overline{M} \leftarrow \text{SE.DEC}(\overline{K}_e, N, C)$ // by tidyness: $C = \text{SE.ENC}(\overline{K}_e, N, \overline{M})$

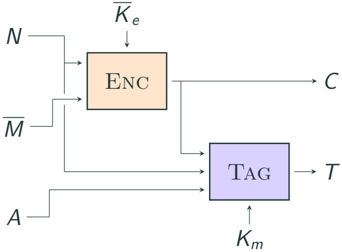
$(K_m, A) \leftarrow\$ \mathcal{K} \times \mathcal{A}$

return $((K_e, K_m), N, A, M), ((\overline{K}_e, K_m), N, A, \overline{M})$

- Attack exploits independent keys for underlying encryption scheme and MAC
- Attack does not work if keys are derived via a pseudorandom generator from some master key

Committing Attack against N2 (Encrypt-then-MAC)





Adversary $\mathcal{A}$

$K_e, N, M \leftarrow_{\$} \mathcal{K} \times \mathcal{N} \times \mathcal{M}$

$C \leftarrow \textsc{Se.Enc}(K_e, N, M)$

$\overline{K}_e \leftarrow_{\$} \mathcal{K} \setminus \{K_e\}$

$\overline{M} \leftarrow \textsc{Se.Dec}(\overline{K}_e, N, C)$ // by tidyness: $C = \textsc{Se.Enc}(\overline{K}_e, N, \overline{M})$

$(K_m, A) \leftarrow_{\$} \mathcal{K} \times \mathcal{A}$

return $((K_e, K_m), N, A, M), ((\overline{K}_e, K_m), N, A, \overline{M})$

- Attack exploits independent keys for underlying encryption scheme and MAC
- Attack does not work if keys are derived via a pseudorandom generator from some master key
- Attack also does not carry over to more practical AE schemes

NIST Lightweight Cryptography Finalists

NIST Lightweight Cryptography (LWC) Standardization

Timeline

- August 2018:
  Call for algorithms
- April 2019:
  56 round-1 candidates
- August 2019:
  32 round-2 candidates
- March 2021:
  10 finalists

NIST Lightweight Cryptography (LWC) Standardization

Timeline

- August 2018:
  Call for algorithms
- April 2019:
  56 round-1 candidates
- August 2019:
  32 round-2 candidates
- March 2021:
  10 finalists

Finalists:

1. Ascon
2. Elephant
3. Gift-Cofb
4. Grain-128aead
5. Isap
6. Photon-Beetle
7. Romulus
8. Schwaemm
9. TinyJambu
10. Xoodyak

NIST Lightweight Cryptography (LWC) Standardization

Timeline

- August 2018:
  Call for algorithms
- April 2019:
  56 round-1 candidates
- August 2019:
  32 round-2 candidates
- March 2021:
  10 finalists
- February 2023:
  Ascon selected to be standardized

Finalists:

1. Ascon
2. Elephant
3. Gift-Cofb
4. Grain-128aead
5. Isap
6. Photon-Beetle
7. Romulus
8. Schwaemm
9. TinyJambu
10. Xoodyak

NIST Lightweight Cryptography (LWC) Standardization

Timeline

- August 2018:
  Call for algorithms
- April 2019:
  56 round-1 candidates
- August 2019:
  32 round-2 candidates
- March 2021:
  10 finalists
- February 2023:
  Ascon selected to be standardized

Finalists:

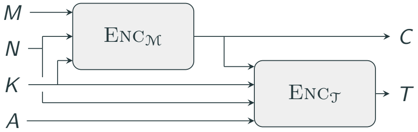
1. Ascon
2. Elephant
3. Gift-Cofb
4. Grain-128aead
5. Isap
6. Photon-Beetle
7. Romulus
8. Schwaemm
9. TinyJambu
10. Xoodyak

We analyze the committing security of all finalists except Grain-128aead (which uses a dedicated design)

- we focus on the modes of operation, assuming underlying components to be ideal

NIST LWC Finalists: Classification

Encrypt-then-MAC AE schemes

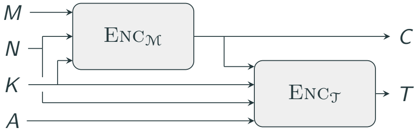


ELEPHANT and ISAP follow this design

- difference to N2: only a single key

NIST LWC Finalists: Classification

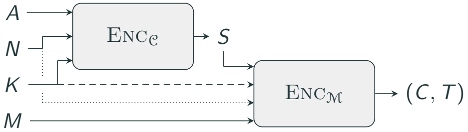
Encrypt-then-MAC AE schemes



Elephant and Isap follow this design

- difference to N2: only a single key

Context-pre-Processing AE schemes

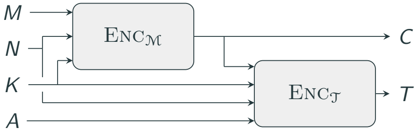


Ascon, Gift-Cofb, Photon-Beetle, Romulus, Schwaemm, TinyJambu, and Xoodyak follow this design

- dashed/dotted line only present in some schemes

NIST LWC Finalists: Classification

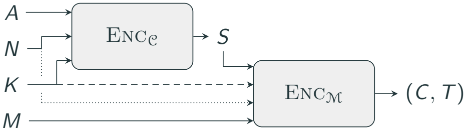
Encrypt-then-MAC AE schemes



Elephant and Isap follow this design

- difference to N2: only a single key

Context-pre-Processing AE schemes



Ascon, Gift-Cofb, Photon-Beetle, Romulus, Schwaemm, TinyJambu, and Xoodyak follow this design

- dashed/dotted line only present in some schemes

Main focus of this talk: Encrypt-then-MAC AE schemes

Results: Overview

Scheme	CMT

Results: Overview

- Attacks with minimal costs against four schemes (✗):
  Romulus, Elephant, Gift-Cofb, and Photon-Beetle

Scheme	CMT
Romulus	✗
Elephant	✗
Gift-Cofb	✗
Photon-Beetle	✗

Results: Overview

- Attacks with minimal costs against four schemes (✗):
  Romulus, Elephant, Gift-Cofb, and Photon-Beetle
- Attacks with significantly less than 2^{64} queries against two schemes (✦):
  TinyJambu and Xoodyak

Scheme	CMT
Romulus	✗
Elephant	✗
Gift-Cofb	✗
Photon-Beetle	✗
TinyJambu	✦
Xoodyak	✦

Results: Overview

- Attacks with minimal costs against four schemes (✗): ROMULUS, ELEPHANT, GIFT-COFB, and PHOTON-BEETLE
- Attacks with significantly less than 2^{64} queries against two schemes (✦): TINYJAMBU and XOODYAK
- Proofs showing about 64-bit committing security for three schemes (✓): ASCON, ISAP, and SCHWAEMM

Scheme	CMT
ROMULUS	✗
ELEPHANT	✗
GIFT-COFB	✗
PHOTON-BEETLE	✗
TINYJAMBU	✦
XOODYAK	✦
ASCON	✓
ISAP	✓
SCHWAEMM	✓

Results: Overview

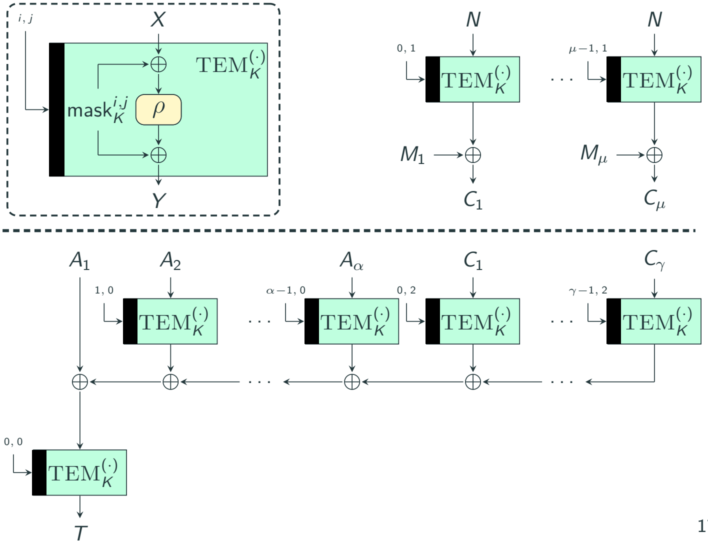
Attacks boil down to one of the following properties:

- The whole state is adversary-controlled (Romulus, Elephant, Gift-Cofb)
  - true for the initial state (Photon-Beetle)
- The adversary-controlled state is too large (Xoodyak)
- The tag is too short (TinyJambu)

Scheme	CMT
Romulus	✗
Elephant	✗
Gift-Cofb	✗
Photon-Beetle	✗
TinyJambu	✦
Xoodyak	✦

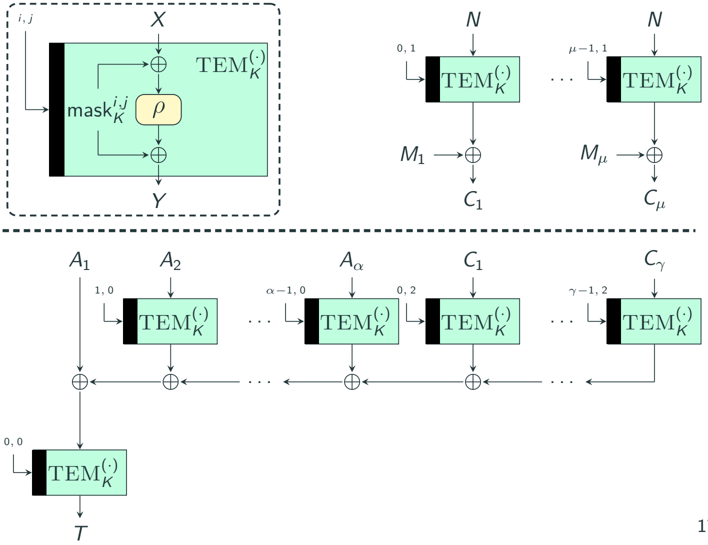
Elephant

- Elephant is based on a public permutation
- The permutation is used in a tweakable Even-Mansour style
- Upper part: encryption
- Lower part: authentication[a]



[a] A_1 contains the nonce N.

- Elephant is based on a public permutation
- The permutation is used in a tweakable Even-Mansour style
- Upper part: encryption
- Lower part: authentication[a]
- Observations:
  1. via A_1, we have full control over the state during authentication
  2. via the message, we have full control over the ciphertext during encryption



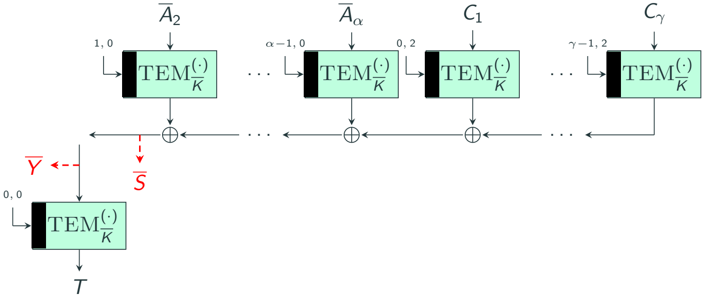
[a] A_1 contains the nonce N.

Committing attack:

1. Choose (K, N, A, M) and compute the ciphertext (C, T)

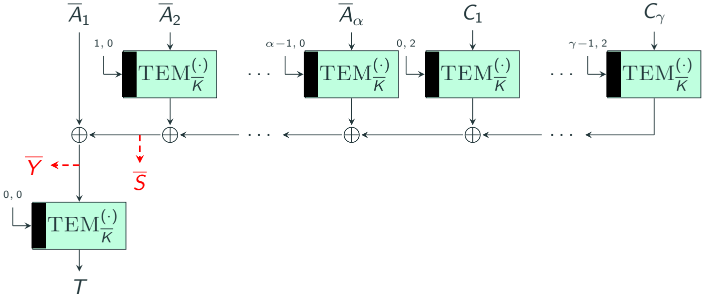
Committing attack:

1. Choose (K, N, A, M) and compute the ciphertext (C, T)
2. Choose $\overline{K}$, $\overline{A}_2, \ldots, \overline{A}_\alpha$, and compute the states $\overline{Y}$ and $\overline{S}$



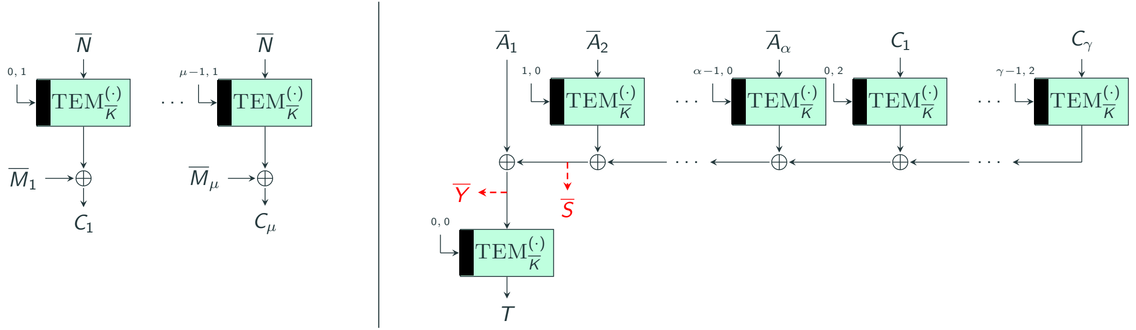
Committing attack:

1. Choose (K, N, A, M) and compute the ciphertext (C, T)
2. Choose $\overline{K}$, $\overline{A}_2, \ldots, \overline{A}_\alpha$, and compute the states $\overline{Y}$ and $\overline{S}$
3. Set $\overline{A}_1 \leftarrow \overline{Y} \oplus \overline{S}$ (note that this also determines the nonce $\overline{N}$)



Committing attack:

1. Choose (K, N, A, M) and compute the ciphertext (C, T)
2. Choose $\overline{K}$, $\overline{A}_2, \ldots, \overline{A}_\alpha$, and compute the states $\overline{Y}$ and $\overline{S}$
3. Set $\overline{A}_1 \leftarrow \overline{Y} \oplus \overline{S}$ (note that this also determines the nonce $\overline{N}$)
4. Choose $\overline{M}$ that, using $\overline{K}$ and $\overline{N}$, encrypts to C



Theorem

Consider $\textsc{Elephant}$ *as shown above. Let* TEM *be modeled as an ideal tweakable cipher* $\widetilde{\mathsf{E}}$. *Then there exists an adversary* $\mathcal{A}$, *making* q *queries to* $\widetilde{\mathsf{E}}$, *such that*

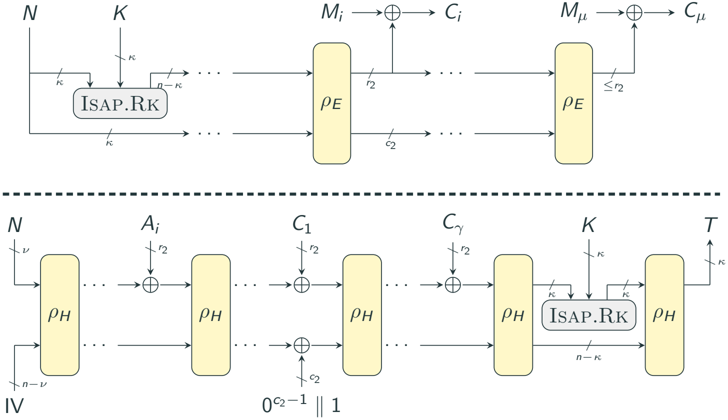
$$\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Elephant}}(\mathcal{A}) = 1\,,$$

where $q = 2\mu + 2\gamma + \alpha + \overline{\alpha}$. *Here,* μ *is the number of message blocks while computing* $\textsc{Enc}_{\mathcal{M}}$ *and* γ *is the number of ciphertext blocks while computing* $\textsc{Enc}_{\mathcal{T}}$.[6] *Furthermore,* α *and* $\overline{\alpha}$ *are the number of associated data blocks for the two tuples that* $\mathcal{A}$ *outputs.*

[6]Note that μ and γ might not be the same.

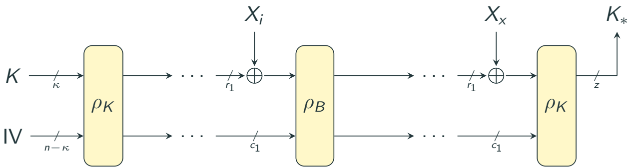
ISAP

- ISAP is based on a public permutation
- It features a re-keying function ISAP.RK to achieve resilience against side-channel leakage
- Upper part: encryption
- Lower part: authentication



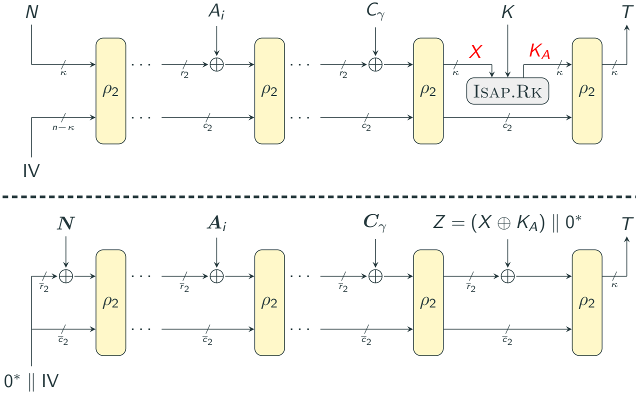
ISAP: Re-Keying Function

- Re-Keying function ISAP.RK is a plain sponge construction
- Core property: rate is set to 1 to minimize the effect of side-channel leakage



Proof idea:

- model ISAP as a plain sponge with an increased rate to handle its special features (re-keying function and domain separation)
- Collision resistance of the plain sponge construction yields committing security



Isap: Committing Security

Theorem

Consider Isap *as shown above. Let* ρ_1 *and* ρ_2 *be modeled by ideal permutations* ρ_1 *and* ρ_2*, respectively. Then for any adversary* $\mathcal{A}$ *making* q_1 *and* q_2 *queries to* ρ_1 *and* ρ_2*, respectively, it holds that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Isap}}(\mathcal{A}) \leq \frac{q_1(q_1-1)}{2^\kappa} + \frac{q_1(q_1+1)}{2^{n-\kappa}} + \frac{q_2(q_2-1)}{2^\kappa} + \frac{q_2(q_2+1)}{2^{n-\max\{\kappa, r_2+1\}}} \,.$$

Isap: Committing Security

Theorem

Consider Isap *as shown above. Let ρ_1 and ρ_2 be modeled by ideal permutations ρ_1 and ρ_2, respectively. Then for any adversary $\mathcal{A}$ making q_1 and q_2 queries to ρ_1 and ρ_2, respectively, it holds that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\mathrm{ISAP}}(\mathcal{A}) \leq \frac{q_1(q_1-1)}{2^\kappa} + \frac{q_1(q_1+1)}{2^{n-\kappa}} + \frac{q_2(q_2-1)}{2^\kappa} + \frac{q_2(q_2+1)}{2^{n-\max\{\kappa, r_2+1\}}} .$$

Dominant terms (for NIST parameter sets): $\frac{q_1(q_1-1)}{2^\kappa}$ and $\frac{q_2(q_2-1)}{2^\kappa}$

ISAP: Committing Security

Theorem

Consider ISAP *as shown above. Let ρ_1 and ρ_2 be modeled by ideal permutations ρ_1 and ρ_2, respectively. Then for any adversary $\mathcal{A}$ making q_1 and q_2 queries to ρ_1 and ρ_2, respectively, it holds that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Isap}}(\mathcal{A}) \leq \frac{q_1(q_1-1)}{2^\kappa} + \frac{q_1(q_1+1)}{2^{n-\kappa}} + \frac{q_2(q_2-1)}{2^\kappa} + \frac{q_2(q_2+1)}{2^{n-\max\{\kappa, r_2+1\}}} \,.$$

Dominant terms (for NIST parameter sets): $\frac{q_1(q_1-1)}{2^\kappa}$ and $\frac{q_2(q_2-1)}{2^\kappa}$

- Committing security can be increased by increasing κ (length of tags and session keys)
- but only up to $\kappa = n/2$ (at which point the other terms become dominant)

Zero-Padding

Zero-Padding

Prepend zeros to the message prior to encryption:

$$\textsc{zp-Ae.Enc}(K, N, A, M) \coloneqq \textsc{Ae.Enc}(K, N, A, 0^z \parallel M)$$

[7]Naito, Sasaki, and Sugawara. **"Commiting Security of Ascon: Cryptanalysis on Primitive and Proof on Mode".** In: *ToSC 2023 (4)*. 2023.

Zero-Padding

Prepend zeros to the message prior to encryption:

$$\text{ZP-AE.ENC}(K, N, A, M) := \text{AE.ENC}(K, N, A, 0^z \parallel M)$$

"Lightweight" method to achieve CMT_K security

[7] Naito, Sasaki, and Sugawara. **"Commiting Security of Ascon: Cryptanalysis on Primitive and Proof on Mode".** In: *ToSC 2023 (4)*. 2023.

Zero-Padding

Prepend zeros to the message prior to encryption:

$$\textsc{zp-Ae.Enc}(K, N, A, M) := \textsc{Ae.Enc}(K, N, A, 0^z \parallel M)$$

"Lightweight" method to achieve CMT_K security

- neither claimed nor proven to work for all schemes

[7] Naito, Sasaki, and Sugawara. **"Commiting Security of Ascon: Cryptanalysis on Primitive and Proof on Mode".** In: *ToSC 2023 (4)*. 2023.

Zero-Padding

Prepend zeros to the message prior to encryption:

$$\textsc{zp}\text{-}\textsc{Ae.Enc}(K, N, A, M) := \textsc{Ae.Enc}(K, N, A, 0^z \parallel M)$$

"Lightweight" method to achieve CMT_K security

- neither claimed nor proven to work for all schemes
- Zero-padding was shown to improve CMT security of $\textsc{Ascon}$[7]

[7] Naito, Sasaki, and Sugawara. **"Commiting Security of Ascon: Cryptanalysis on Primitive and Proof on Mode".** In: *ToSC 2023 (4)*. 2023.

Zero-Padding

We ask two questions regarding zero-padding:

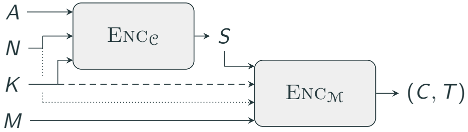
Zero-Padding

We ask two questions regarding zero-padding:

1. Can we achieve CMT_K security for the schemes that are not CMT secure?
2. Can we increase CMT security for the secure schemes?

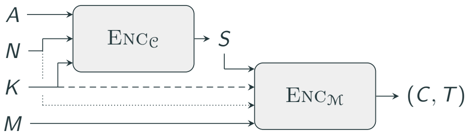
Zero-Padding: Photon-Beetle and Xoodyak

Context-pre-Processing AE schemes

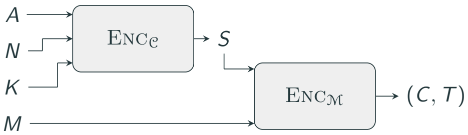


Zero-Padding: Photon-Beetle and Xoodyak

Context-pre-Processing AE schemes



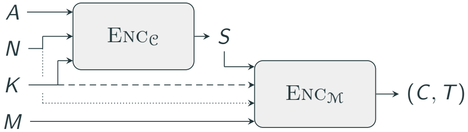
Full-Context-pre-Processing AE schemes



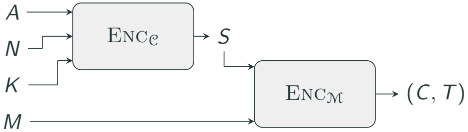
Full-Context-pre-Processing AE schemes:
Photon-Beetle and Xoodyak

Zero-Padding: Photon-Beetle and Xoodyak

Context-pre-Processing AE schemes



Full-Context-pre-Processing AE schemes

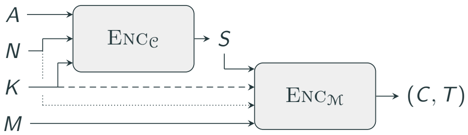


Full-Context-pre-Processing AE schemes: Photon-Beetle and Xoodyak

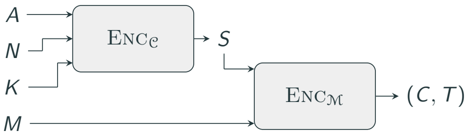
Finding a collision for $\textsc{Enc}_{\mathcal{C}}$ (for different keys) directly yields a committing attack

Zero-Padding: Photon-Beetle and Xoodyak

Context-pre-Processing AE schemes



Full-Context-pre-Processing AE schemes



Full-Context-pre-Processing AE schemes: Photon-Beetle and Xoodyak

Finding a collision for $\textsc{Enc}_{\mathcal{C}}$ (for different keys) directly yields a committing attack

- For Photon-Beetle and Xoodyak we can find such collisions
- Photon-Beetle and Xoodyak cannot be "patched" via zero-padding to achieve $\mathrm{CMT}_{\mathsf{K}}$ security

Zero-Padding: $\textsc{Elephant}$

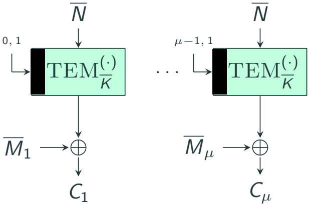
Zero-padding dos not provides $\mathrm{CMT}_{\mathsf{K}}$ security if the number of zeros is smaller than the block length

- Assume that we have a ciphertext $(C, T) = \textsc{Enc}(K, N, A, M)$. How to find $(\overline{K}, \overline{N}, \overline{A}, \overline{M})$?

Zero-Padding: Elephant

Zero-padding dos not provides $\mathrm{CMT_K}$ security if the number of zeros is smaller than the block length

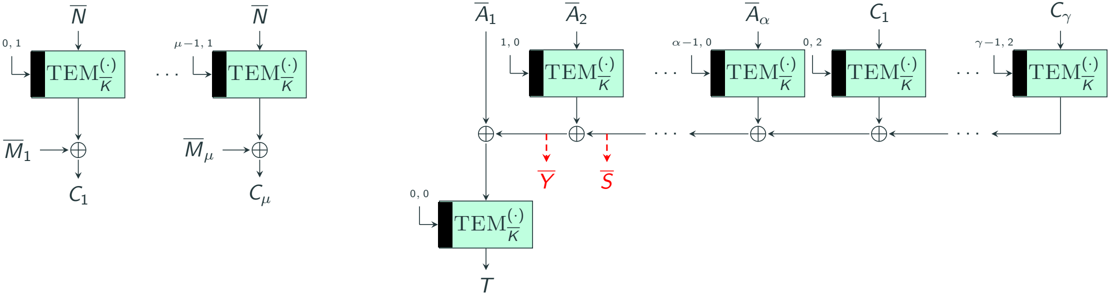
- Assume that we have a ciphertext $(C, T) = \textsc{Enc}(K, N, A, M)$. How to find $(\overline{K}, \overline{N}, \overline{A}, \overline{M})$?
- If $\overline{M}_1 = 0^n$, set $\overline{N} \leftarrow \mathrm{TEM}^{-1}(C_1)$ and choose remaining message blocks as before



Zero-Padding: Elephant

Zero-padding dos not provides CMT_K security if the number of zeros is smaller than the block length

- Assume that we have a ciphertext $(C, T) = \textsc{Enc}(K, N, A, M)$. How to find $(\overline{K}, \overline{N}, \overline{A}, \overline{M})$?
- If $\overline{M}_1 = 0^n$, set $\overline{N} \leftarrow \mathrm{TEM}^{-1}(C_1)$ and choose remaining message blocks as before
- For the authentication part, we have to target a different associated data block than $\overline{A}_1$, which contains the nonce $\overline{N}$



Zero-Padding: ISAP

Core idea: birthday attack on the tag

Zero-Padding: Isap

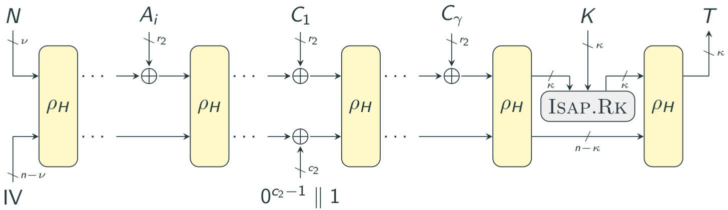
Core idea: birthday attack on the tag

- fix arbitrary key-nonce pair (K, N) and compute an honest ciphertext C by encrypting some message M

Zero-Padding: Isap

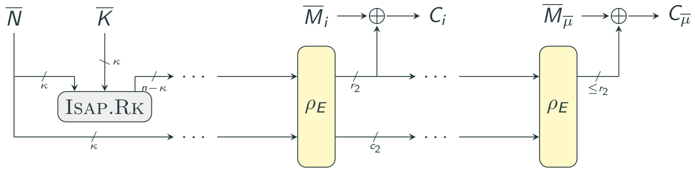
Core idea: birthday attack on the tag

- fix arbitrary key-nonce pair (K, N) and compute an honest ciphertext C by encrypting some message M
- Try various associated data until a tag collision is found ($\approx 2^{64}$ queries); let A and $\overline{A}$ denote the associated data yielding the collision



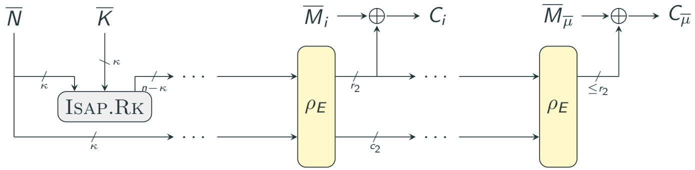
Zero-Padding: Isap

By setting $(\overline{K}, \overline{N}, \overline{M}) \leftarrow (K, N, M)$, we get the same ciphertext C during encryption



Zero-Padding: Isap

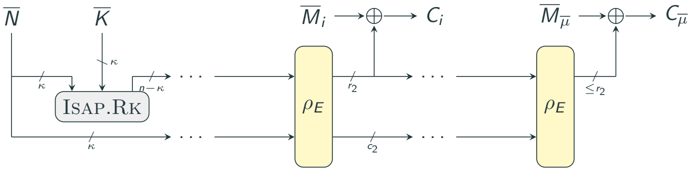
By setting $(\overline{K}, \overline{N}, \overline{M}) \leftarrow (K, N, M)$, we get the same ciphertext C during encryption



- Outputting $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$ breaks CMT security
  - Cost: $\approx 2^{64}$ (find A and $\overline{A}$)
  - Committing security of Isap does *not* increase via zero-padding

Zero-Padding: ISAP

By setting $(\overline{K}, \overline{N}, \overline{M}) \leftarrow (K, N, M)$, we get the same ciphertext C during encryption



- Outputting $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$ breaks CMT security
  - Cost: $\approx 2^{64}$ (find A and $\overline{A}$)
  - Committing security of ISAP does *not* increase via zero-padding
- Important difference to ASCON: computation of C is independent of the associated data

Conclusion

Conclusion

We analyzed the committing security of ...

- ... the generic composition paradigms

Conclusion

We analyzed the committing security of ...

- ... the generic composition paradigms
- ... the NIST LWC finalists

Conclusion

We analyzed the committing security of . . .

- . . . the generic composition paradigms
- . . . the NIST LWC finalists
- . . . the zero-padded versions of several NIST LWC finalists

Conclusion

We analyzed the committing security of . . .

- . . . the generic composition paradigms
- . . . the NIST LWC finalists
- . . . the zero-padded versions of several NIST LWC finalists

Thank You!

patrick.struck@uni.kn