Release of Unverified Plaintext: Tight Unified Model and Application to ANYDAE

Donghoon Chang and Nilanjan Datta and **Avijit Dutta** and Bart Mennink and Mridul Nandi and Somitra Sanadhya and Ferdinand Sibleyras

Institute for Advancing Intelligence, TCG-CREST, Kolkata

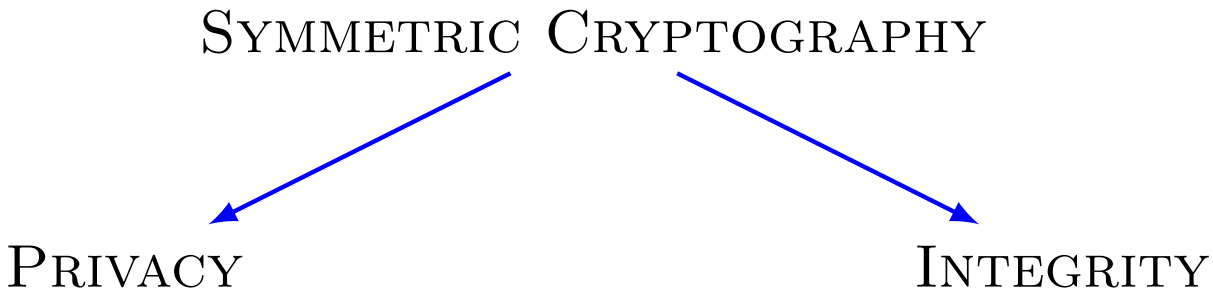
Fast Software Encryption 2020

26th October, 2020

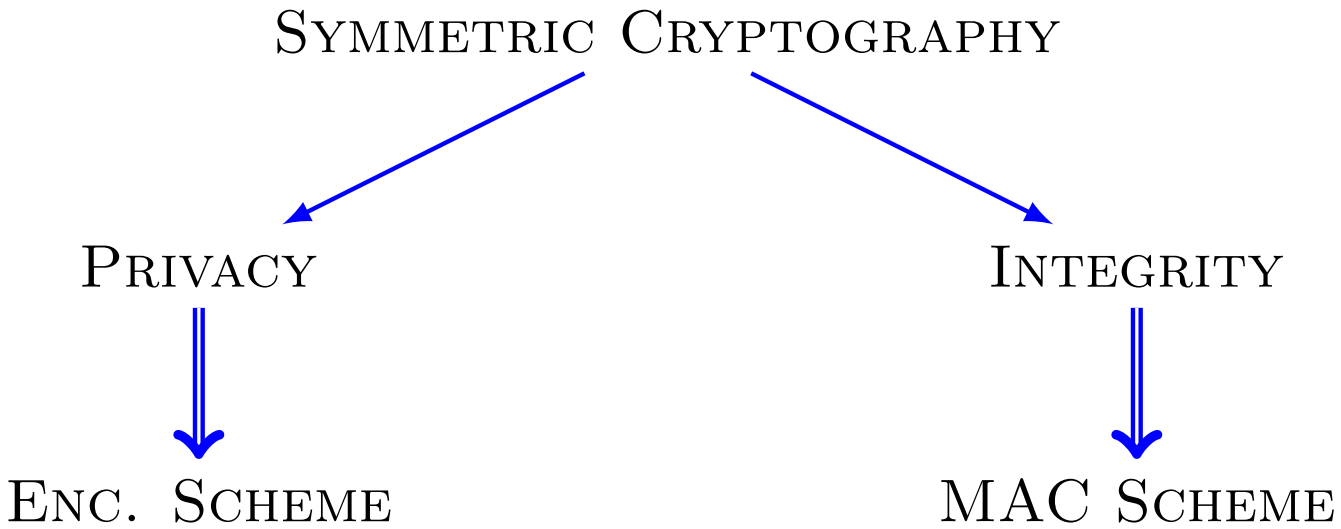
Outline of the Talk

- Definitions of AE and Security Notion.
- RUP Security.
- INT-RUP Attack on SUNDAE.
- MONDAE: An INT-RUP Secure Variant of SUNDAE.
- ANYDAE: Generic INT-RUP Design.
- TUESDAE: An Optimal Instantiation of ANYDAE.

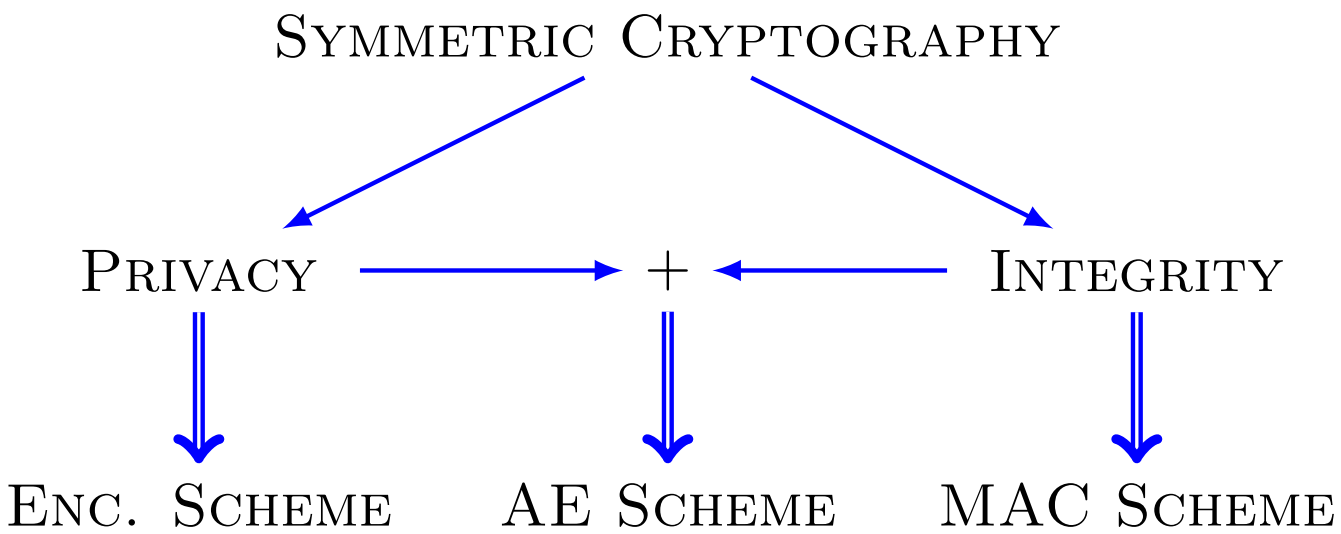
Goal of Symmetric Cryptography



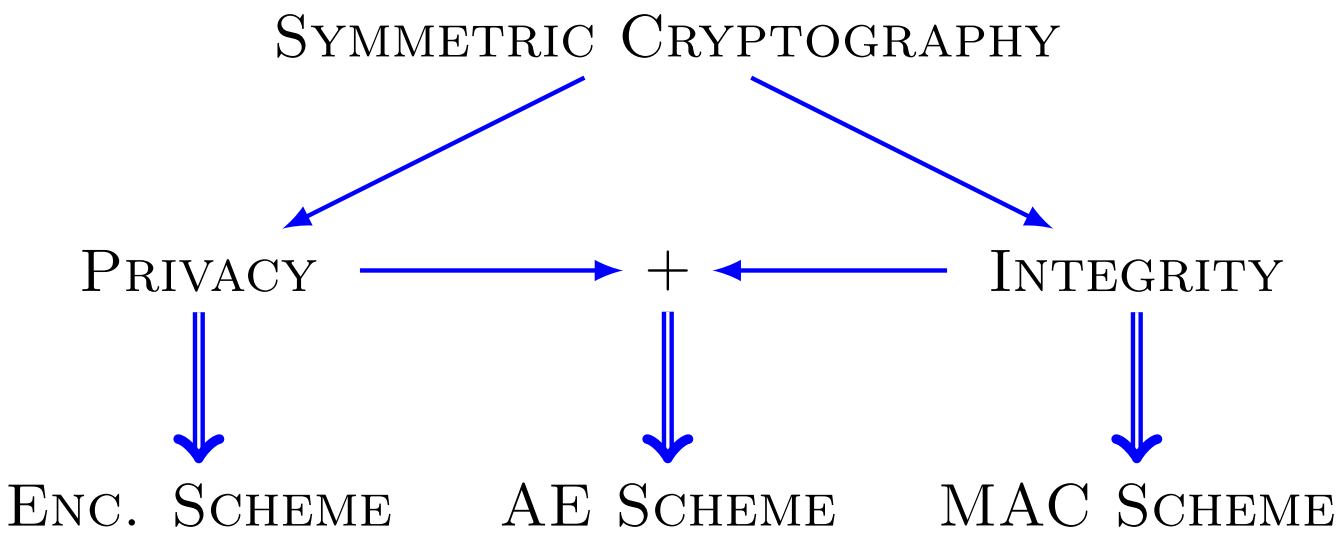
Goal of Symmetric Cryptography



Goal of Symmetric Cryptography

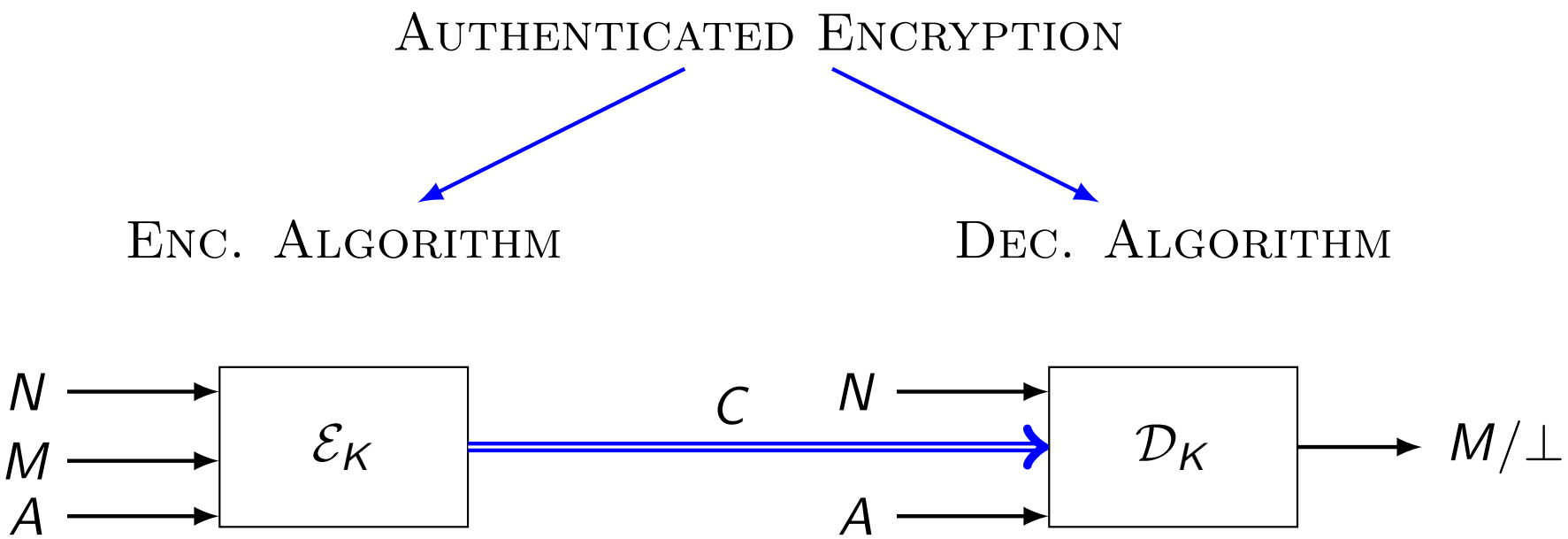


Goal of Symmetric Cryptography

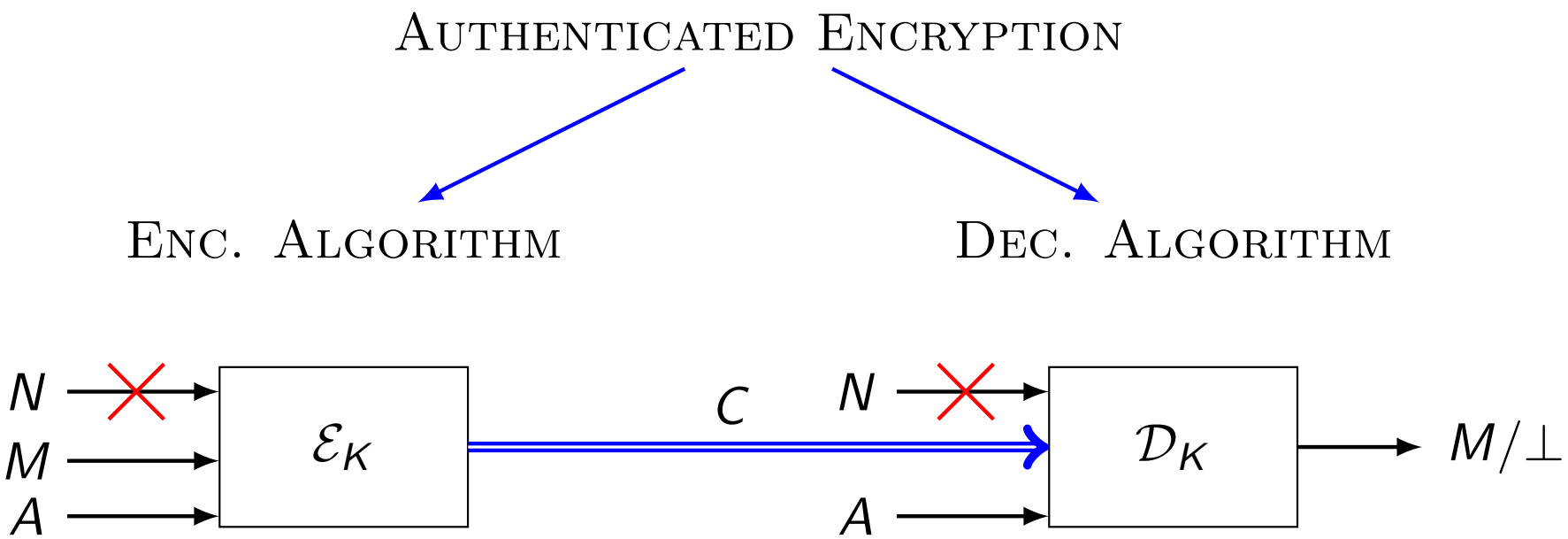


- Stateful AE (Nonce, Random IV or Arbitrary IV Based).
- Stateless AE.

Stateful Authenticated Encryption (AE)

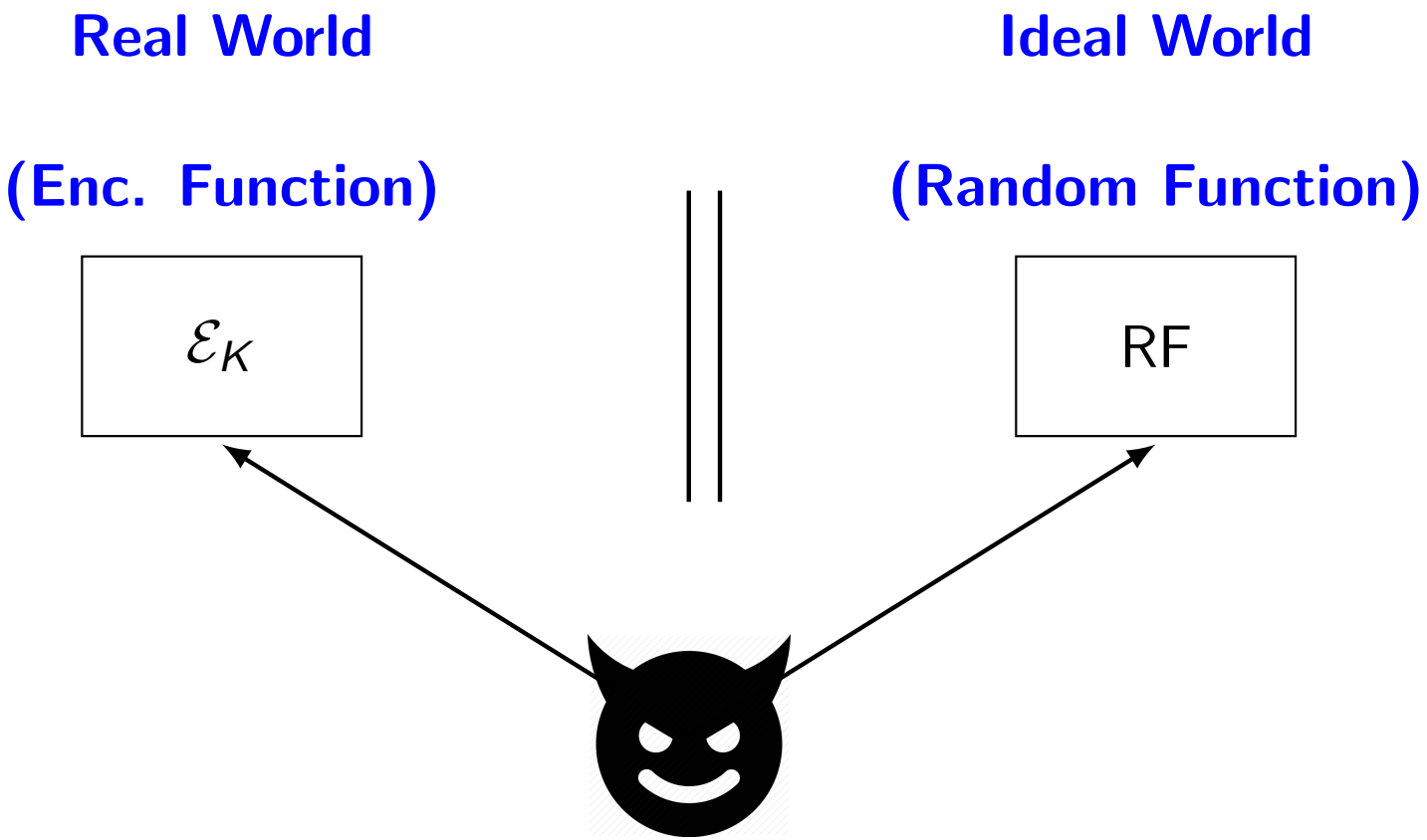


Stateless Authenticated Encryption (AE)



Security of AE

Privacy Requirement (IND-CPA).



Security of AE

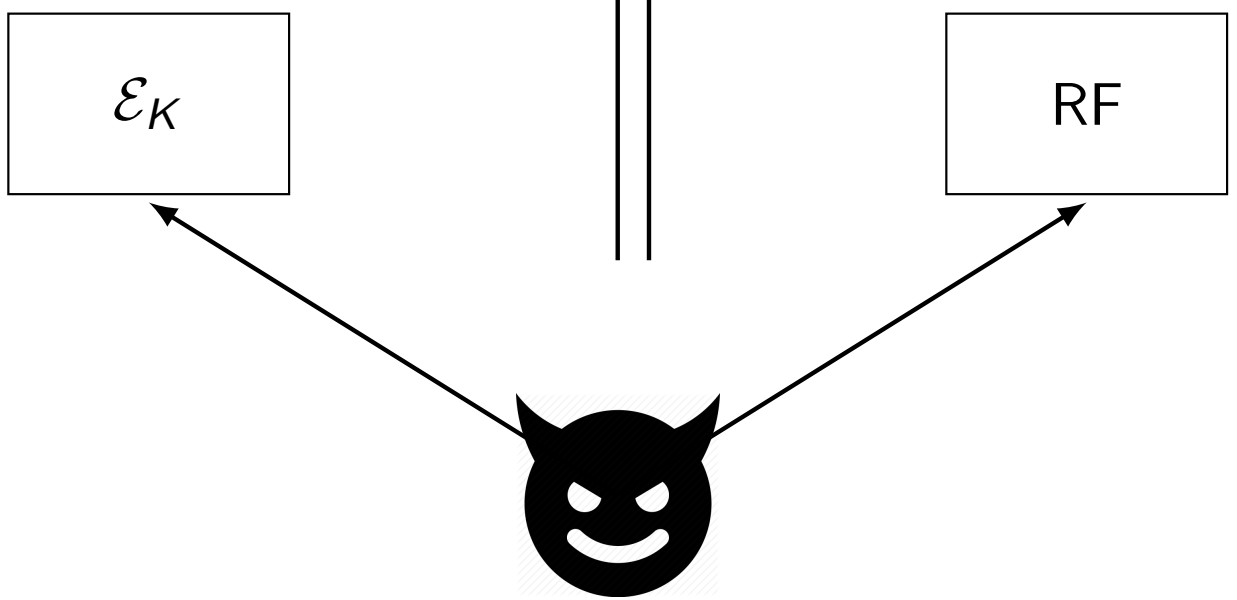
Privacy Requirement (IND-CPA).

Real World

(Enc. Function)

Ideal World

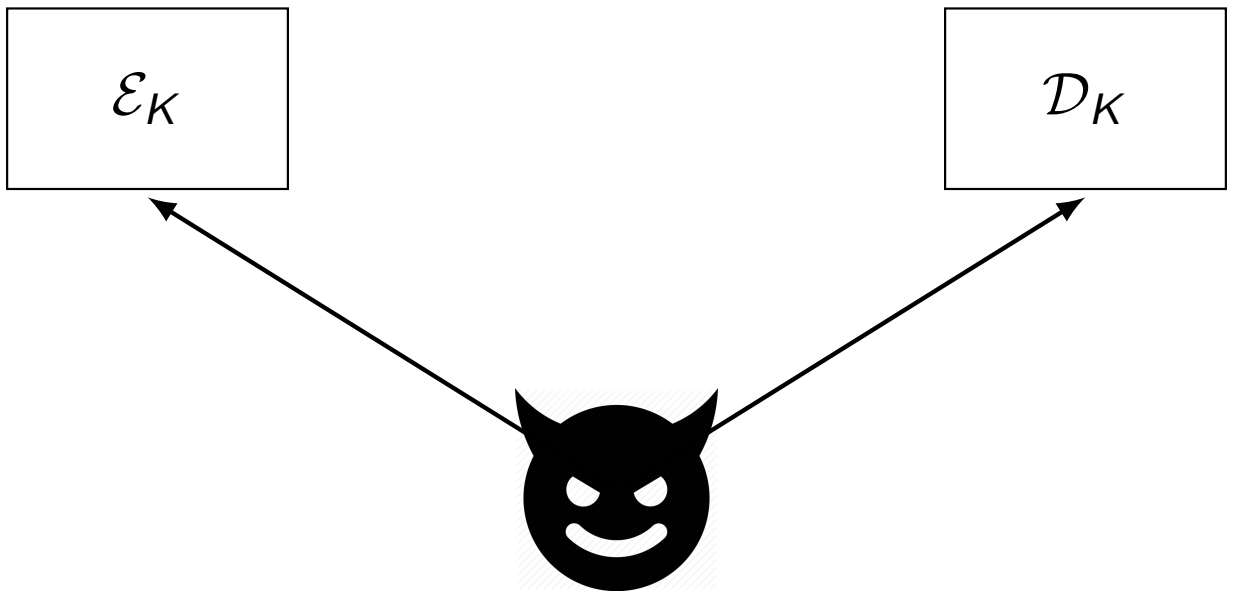
(Random Function)



For a secure AE, the distinguishing advantage is negligible.

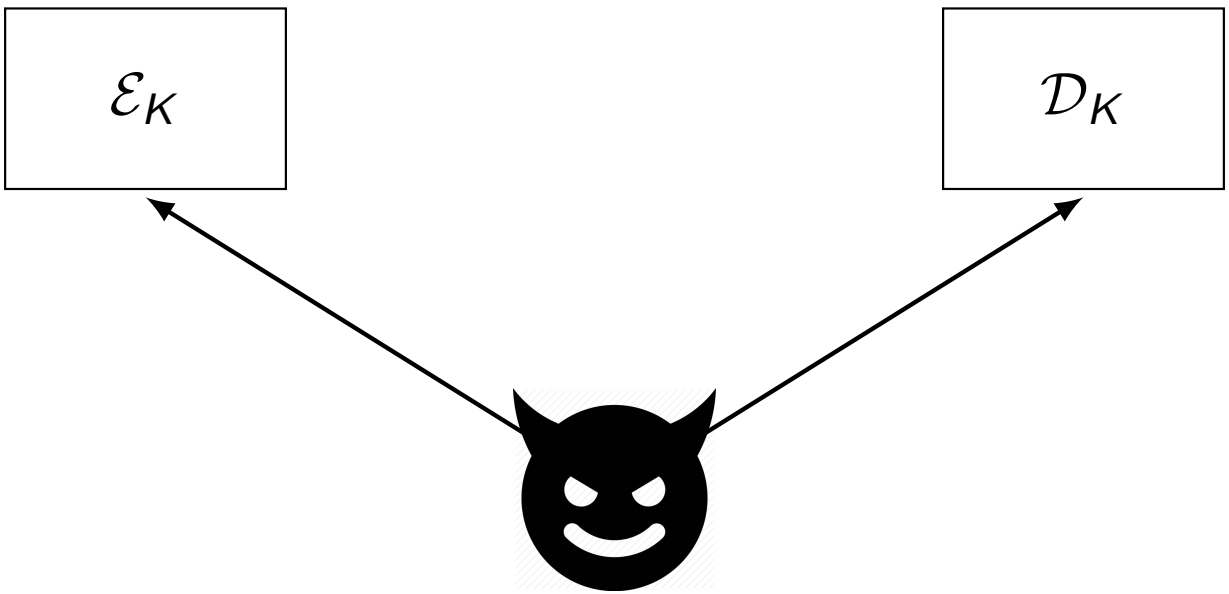
Security of AE

Integrity Requirement (INT-CTXT).



Security of AE

Integrity Requirement (INT-CTXT).

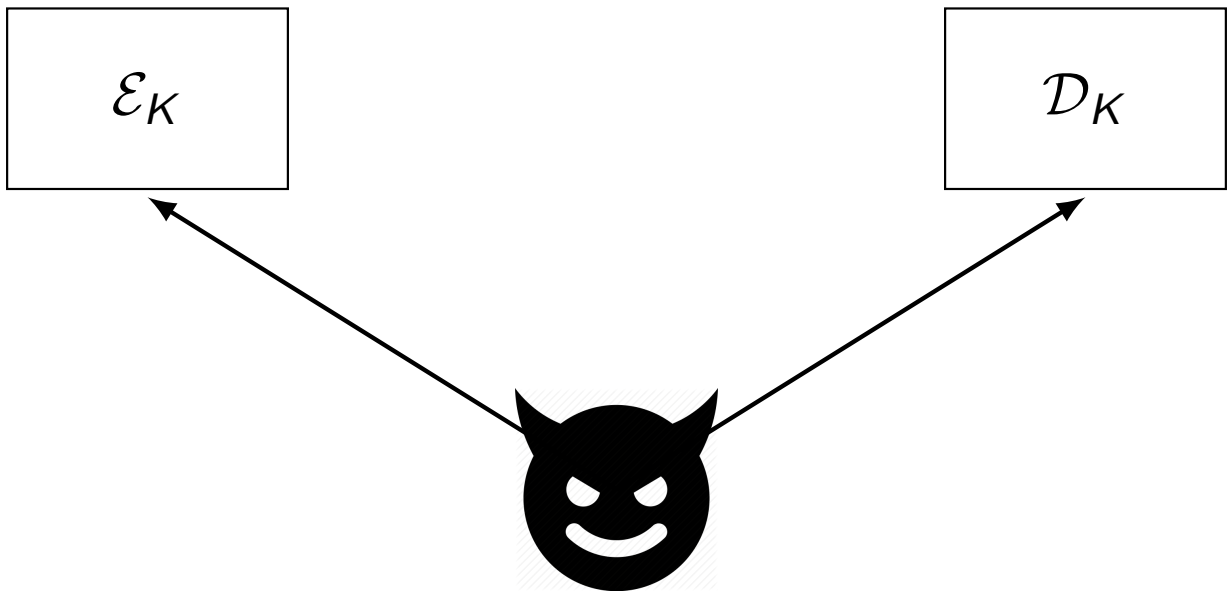


$\mathcal{A}$ forges if $\mathcal{A}$ can produce a non-trivial (N^*, A^*, C^*) tuple such that

$$\mathcal{D}_K(N^*, A^*, C^*) = M^*.$$

Security of AE

Integrity Requirement (INT-CTXT).



$\mathcal{A}$ forges if $\mathcal{A}$ can produce a non-trivial (N^*, A^*, C^*) tuple such that

$$\mathcal{D}_K(N^*, A^*, C^*) = M^*.$$

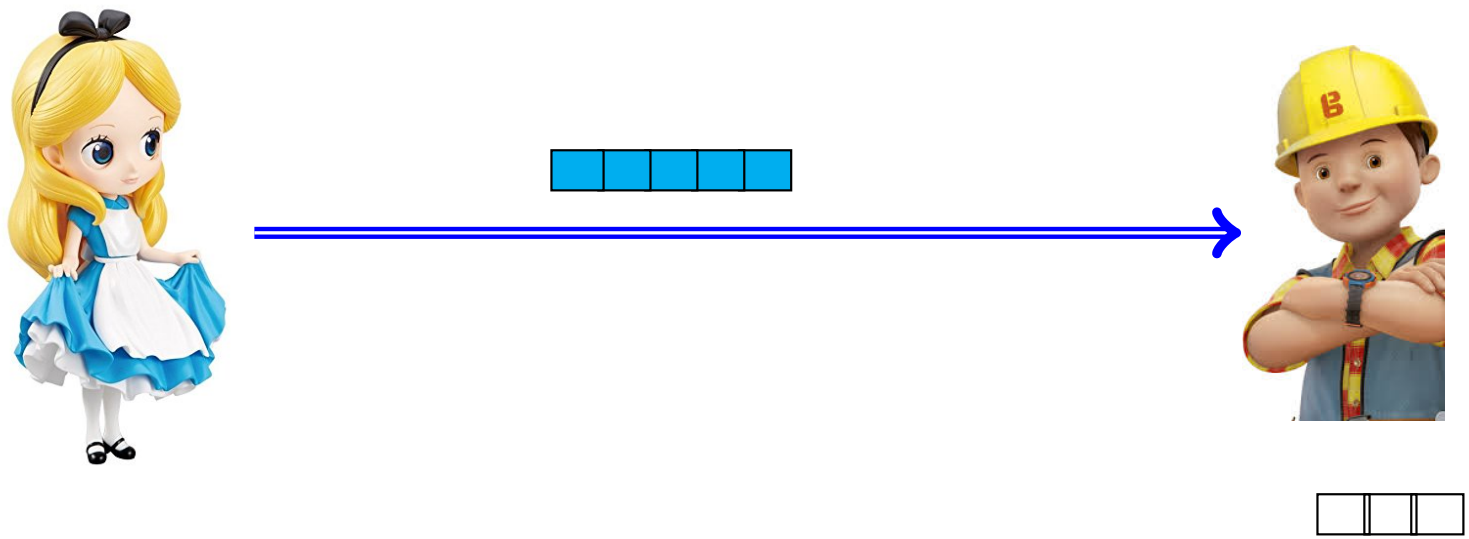
For a secure AE, the forging advantage is negligible.

Security of AE

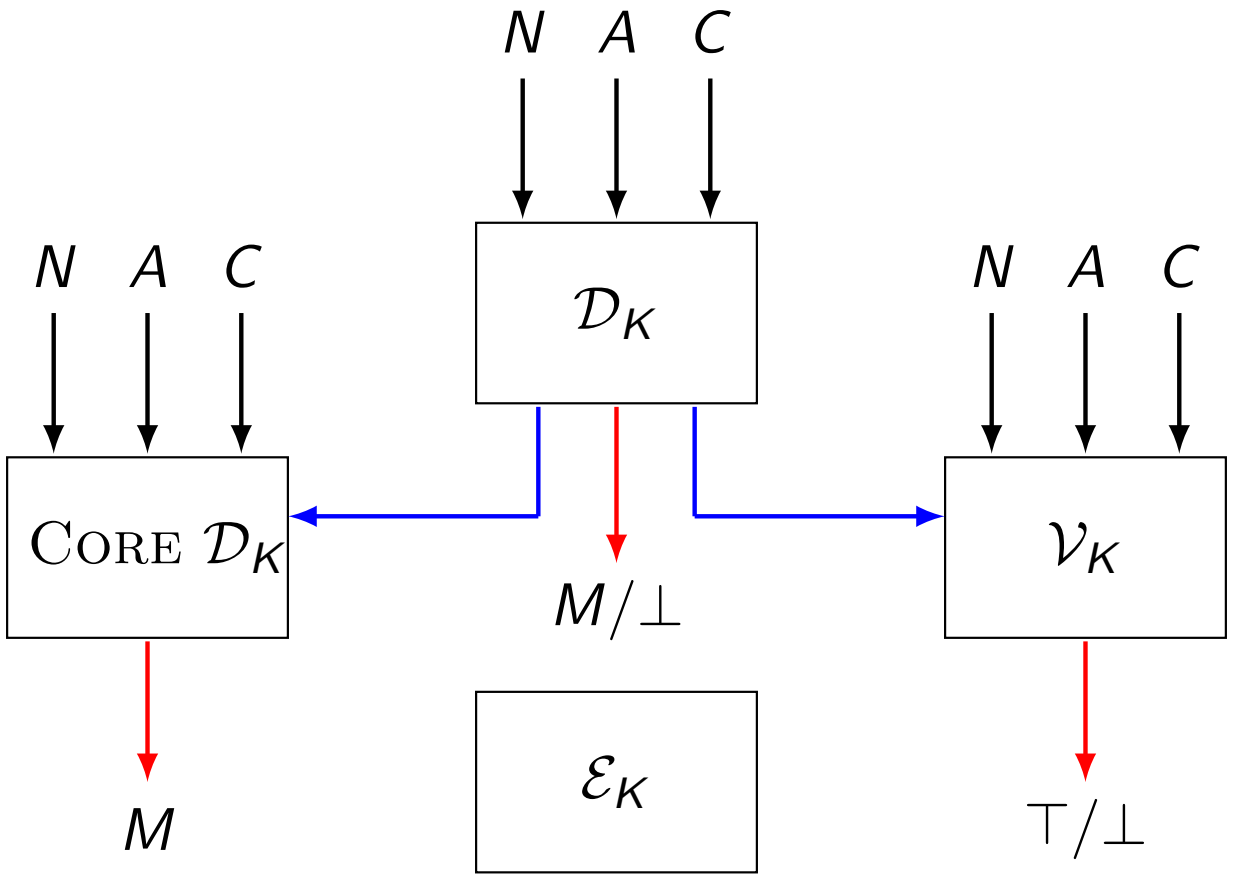
An AE scheme is secure in a conventional sense if it achieves IND-CPA and INT-CTXT security.

Release of Unverifiable Plaintext (RUP) Issue of AE

- Plaintext blocks can only be released after successful verification in the receiver end.
- But the buffer size in the receiving end is limited. As a result, it might not be able to hold the entire plaintext at once.
- **Receiver might have to release the plaintext before verifying.**



RUP Security Model



RUP Security Model

Security of AE in RUP Model formalized by Andreeva et al. (ASIACRYPT 2014).

RUP Security Model

Security of AE in RUP Model formalized by Andreeva et al. (ASIACRYPT 2014).

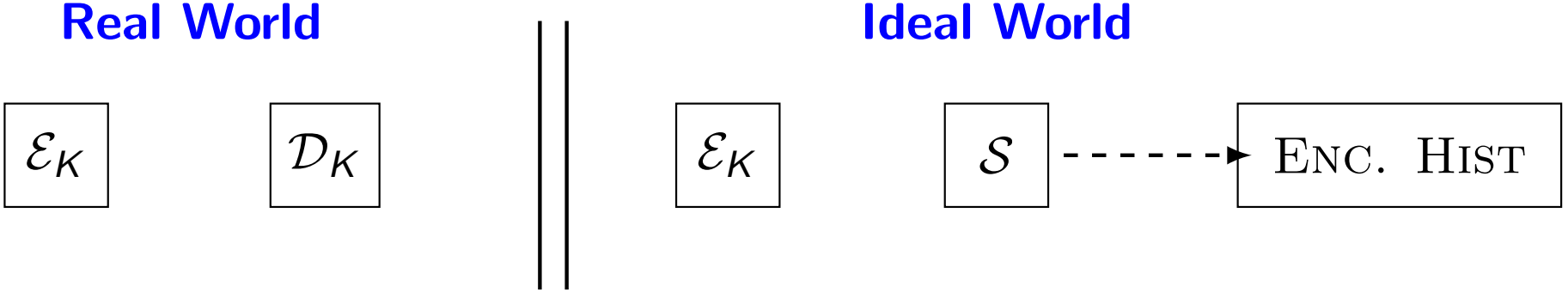
- PA1 / PA2 notion.
- INT-RUP notion.

RUP Security Model

Security of AE in RUP Model formalized by Andreeva et al. (ASIACRYPT 2014).

- PA1 / PA2 notion.
- INT-RUP notion.

PA1 Notion.

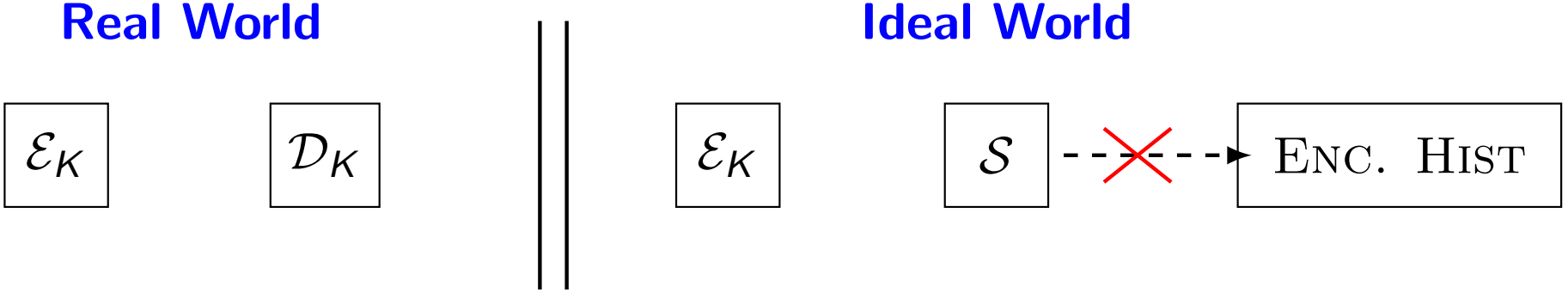


RUP Security Model

Security of AE in RUP Model formalized by Andreeva et al. (ASIACRYPT 2014).

- PA1 / PA2 notion.
- INT-RUP notion.

PA2 Notion.



RUP Secure AE

An AE scheme is RUP secure if it achieves IND-CPA and PA1 and INT-RUP security.

Different Variants of RUP Security

- Hoang et al. introduced RAE notion (EUROCRYPT 2015).
  - **Distinguish AE from a random injective function.**
- Hoang et al. introduced RAE_{sim} notion (EUROCRYPT 2015).
  - **Employs PA2 notion.**
- Barwell et al. introduced SAE notion (IMACC 2015).
  - **Refinement of RAE for nonce based AE.**
- Ashur et al. introduced RUPAE notion (CRYPTO 2017).
  - **Focuses on nonce based AE.**
  - **PA1 + INT-RUP with the ideal model decryption being a random function.**

Different Variants of RUP Security

- Encode-then-SPRP is known to achieve RAE and RUPAE security.

Different Variants of RUP Security

- Encode-then-SPRP is known to achieve RAE and RUPAE security.
- Such construction is two pass in both encryption and decryption.

Different Variants of RUP Security

- Encode-then-SPRP is known to achieve RAE and RUPAE security.
- Such construction is two pass in both encryption and decryption.
- These security notions hold for nonce based AE. Security is void when nonce is misused.

Different Variants of RUP Security

- Encode-then-SPRP is known to achieve RAE and RUPAE security.
- Such construction is two pass in both encryption and decryption.
- These security notions hold for nonce based AE. Security is void when nonce is misused.

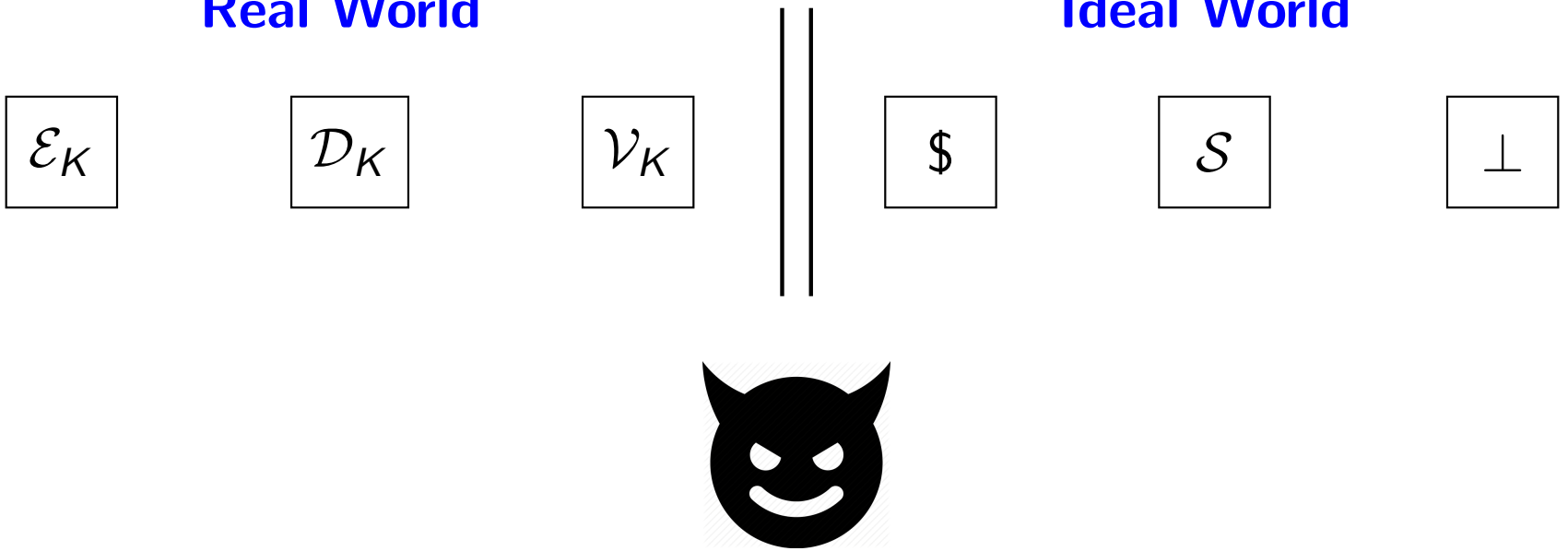
We need a security model in RUP scenario which allows

- Nonce Misuse.
- Single pass decryption feature.

AERUP Security Notion

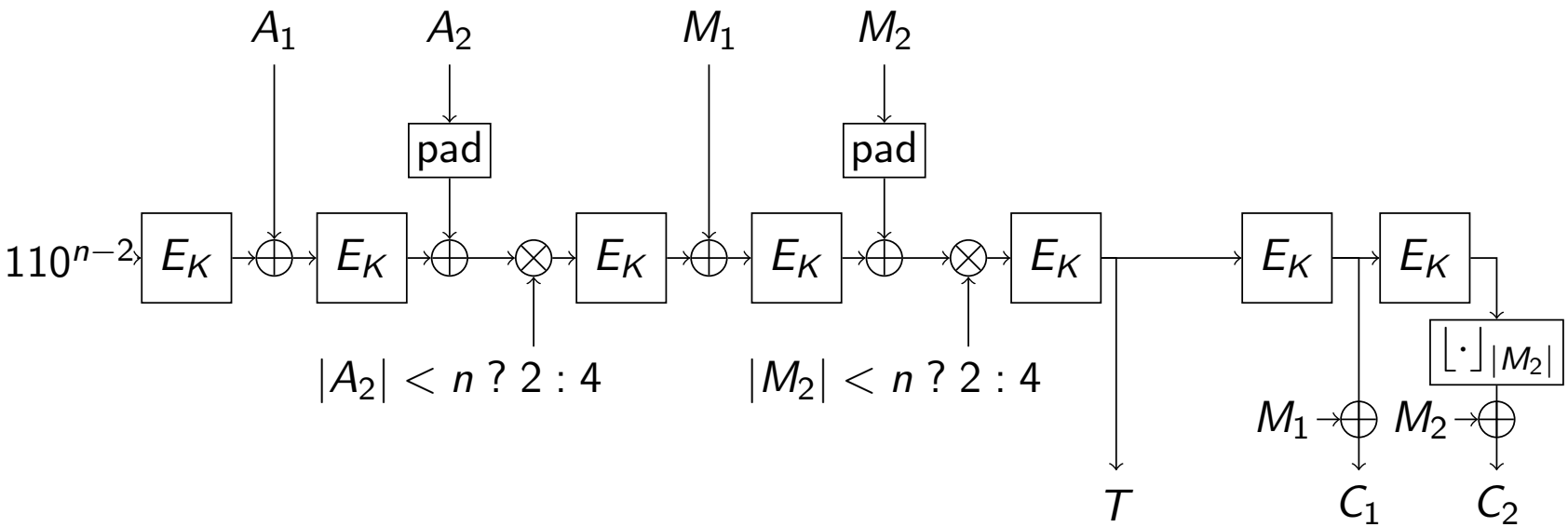
AERUP Security Notion.

Real World **Ideal World**



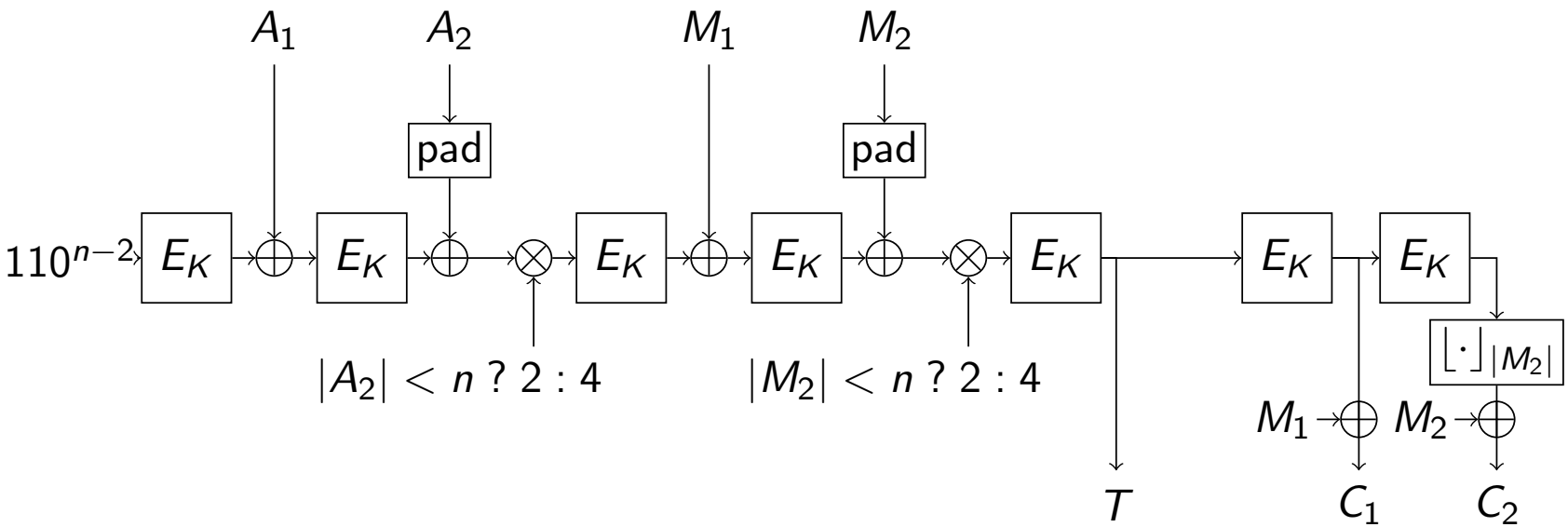
AERUP $\iff$ AE + PA1 + INT-RUP.

SUNDAE [Banik et al., FSE 2019]



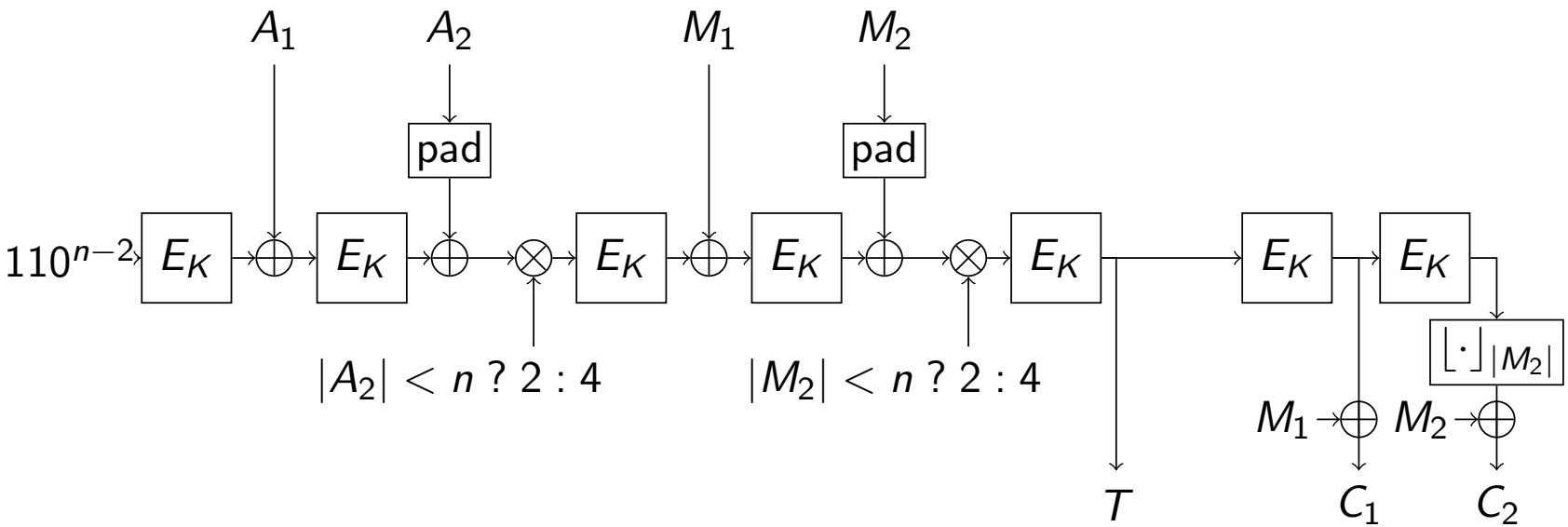
- Determinstic AE.

SUNDAE [Banik et al., FSE 2019]



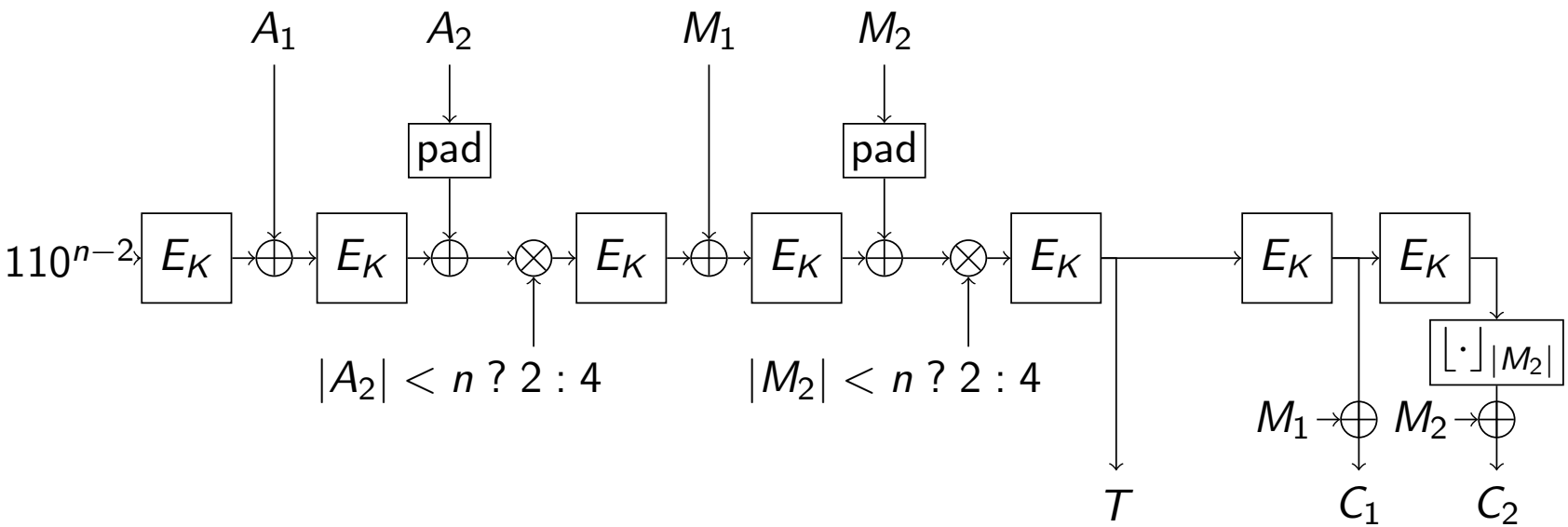
- Determinstic AE.
- Makes $a + 2m + 1$ BC invocations.

SUNDAE [Banik et al., FSE 2019]



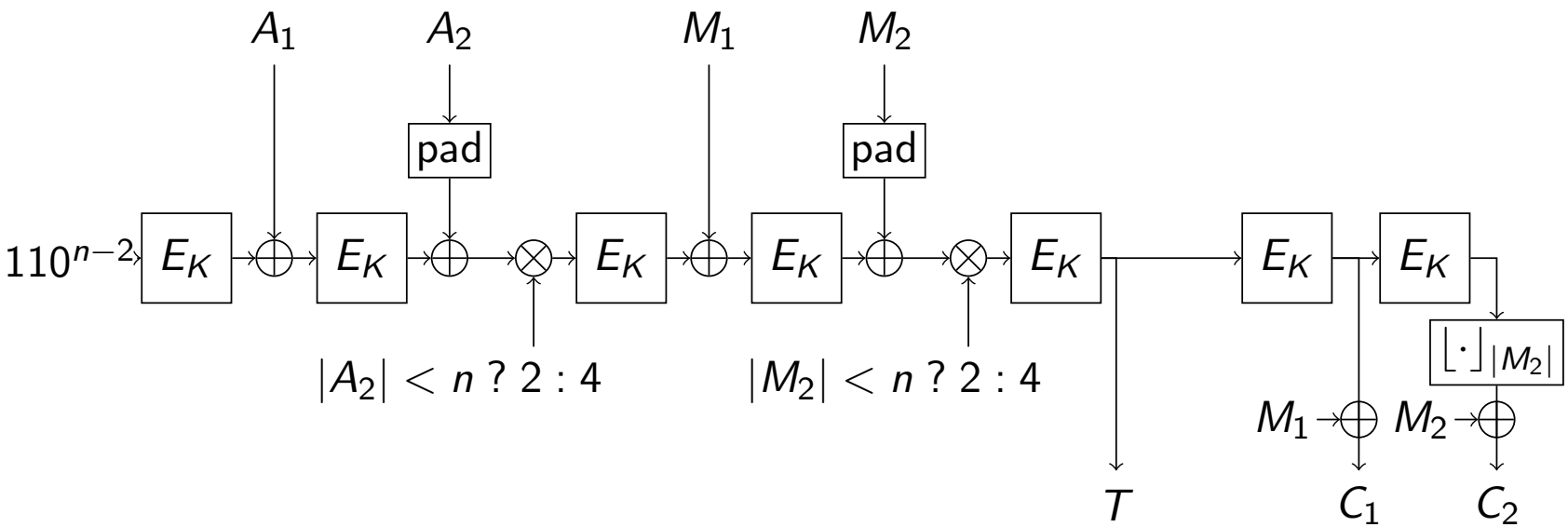
- Determinstic AE.
- Makes $a + 2m + 1$ BC invocations.
- One of the AE Candidates in NIST Lightweight Cryptography competition.

SUNDAE [Banik et al., FSE 2019]



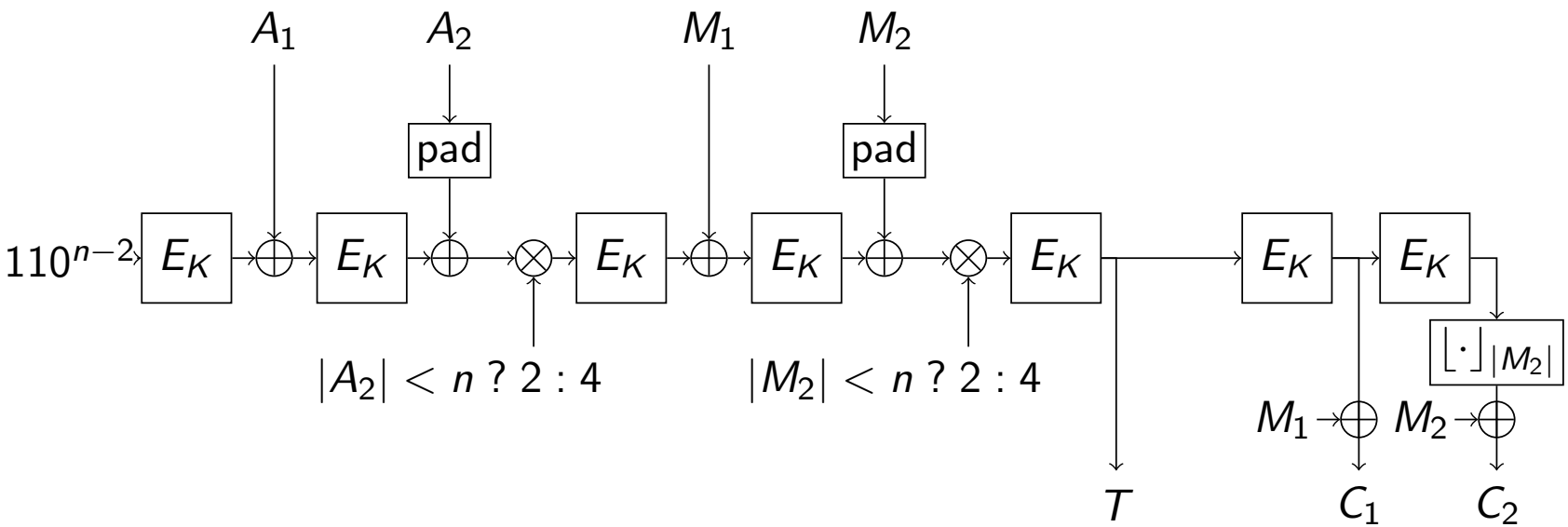
- SUNDAE is particularly efficient for short messages.

SUNDAE [Banik et al., FSE 2019]



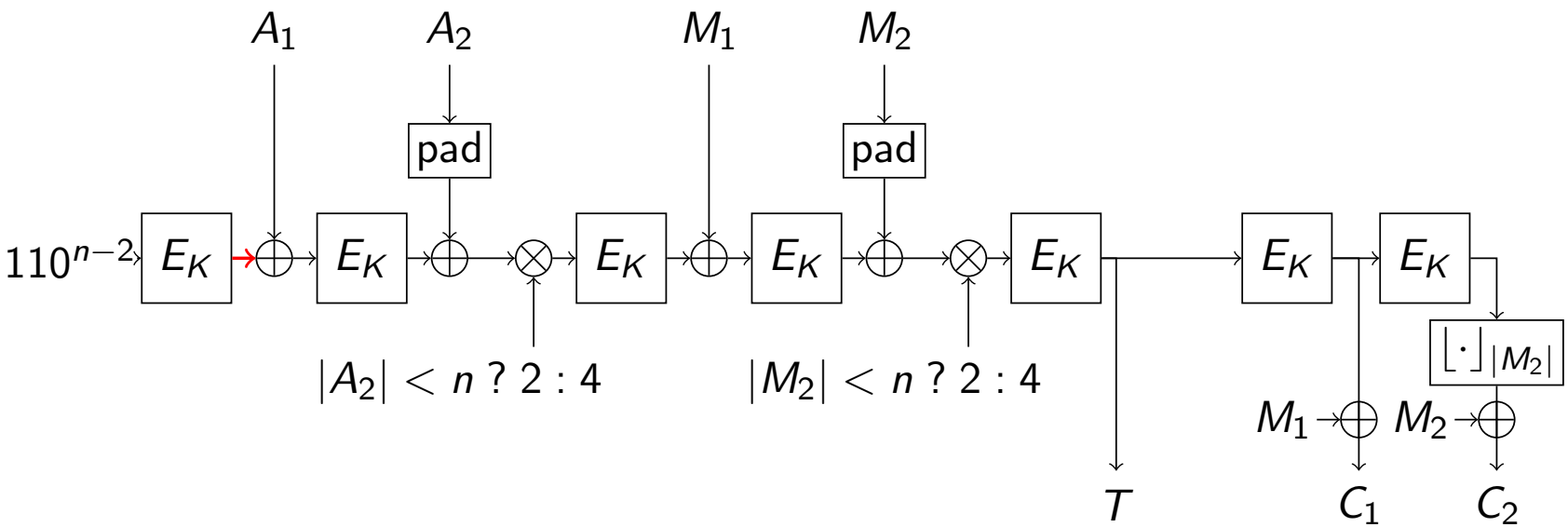
- SUNDAE is particularly efficient for short messages.
- State size as small as the block size.

SUNDAE [Banik et al., FSE 2019]



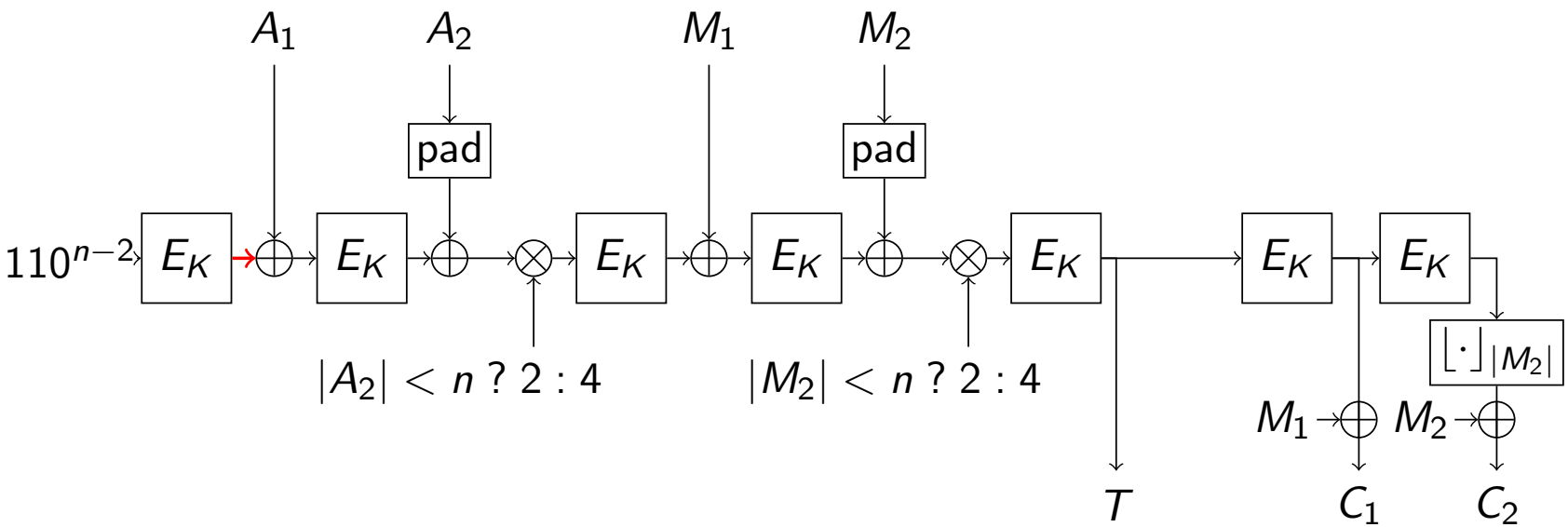
- SUNDAE is particularly efficient for short messages.
- State size as small as the block size.
- Offers good implementation characteristics both on lightweight and high-performance platforms.

SUNDAE is not RUP Secure: INT-RUP Insecurity



1. $\mathcal{A}$ makes query $\mathcal{D}_K(\epsilon, T_1, C_1[1])$, where $T_1 = 110^{n-2}$ and obtains $M_1[1]$.

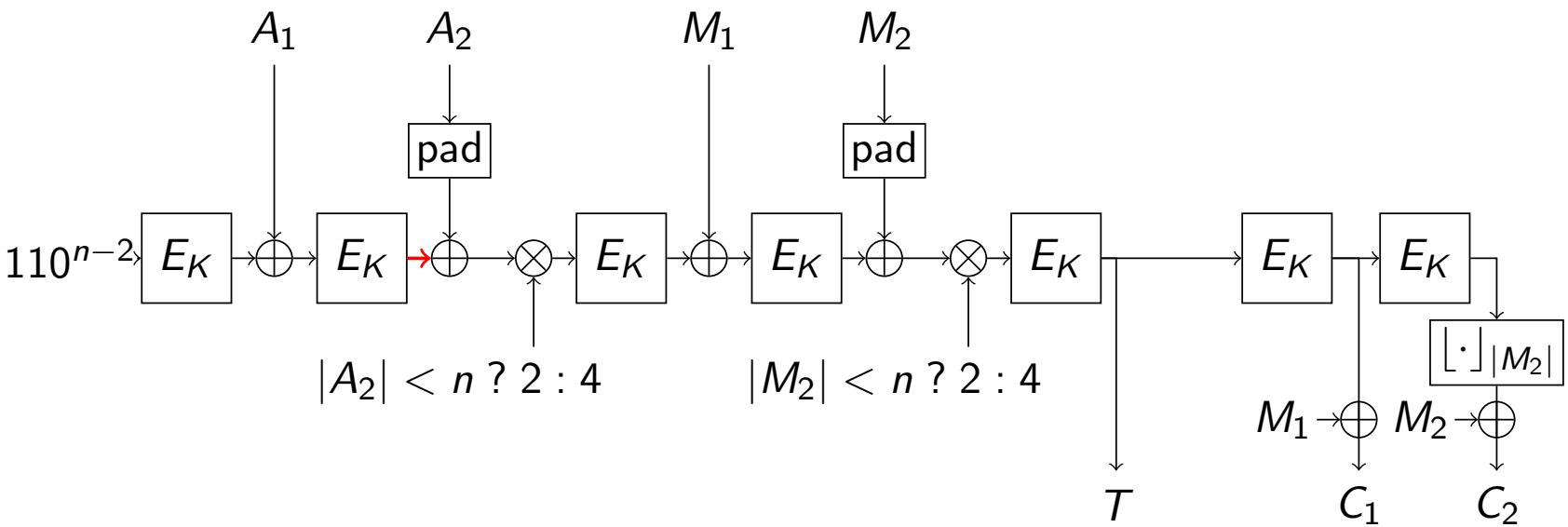
SUNDAE is not RUP Secure: INT-RUP Insecurity



1. $\mathcal{A}$ makes query $\mathcal{D}_K(\epsilon, T_1, C_1[1])$, where $T_1 = 110^{n-2}$ and obtains $M_1[1]$.

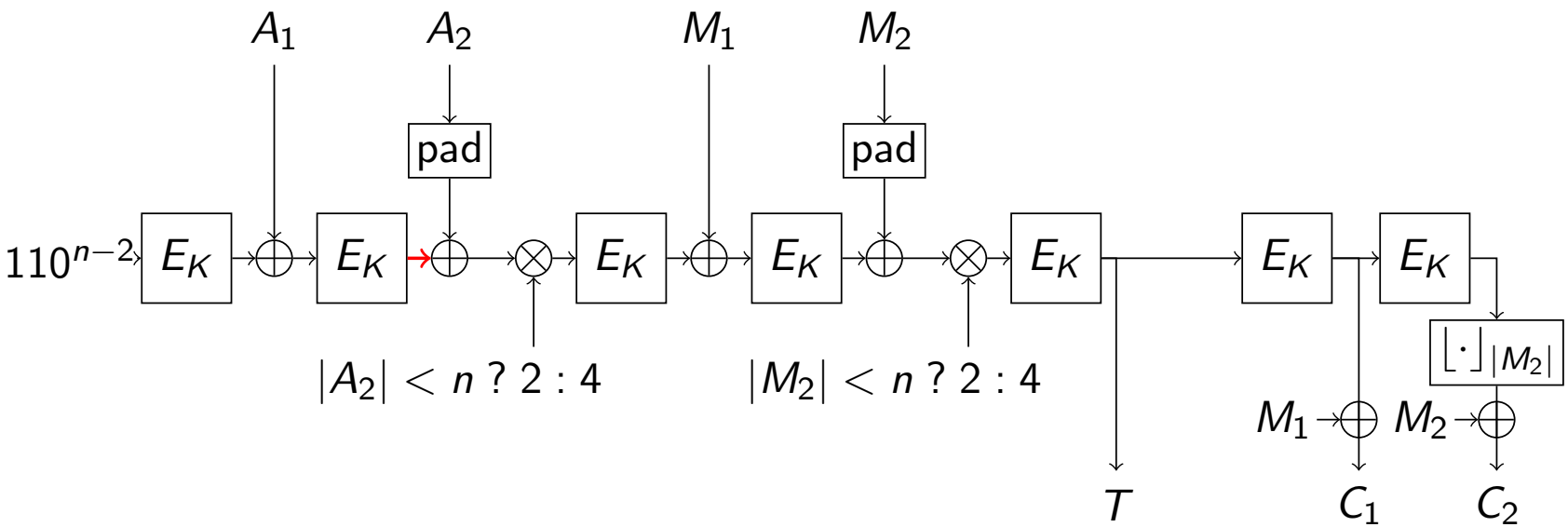
$\mathcal{A}$ **learns** $E_K(110^{n-2}) = M_1[1] \oplus C_1[1]$**.**

SUNDAE is not RUP Secure: INT-RUP Insecurity



2. $\mathcal{A}$ makes query $\mathcal{D}_K(\epsilon, T_2, C_2[1])$, where $T_2 = M_1[1] \oplus C_1[1] \oplus A[1]$ and obtains $M_2[1]$.

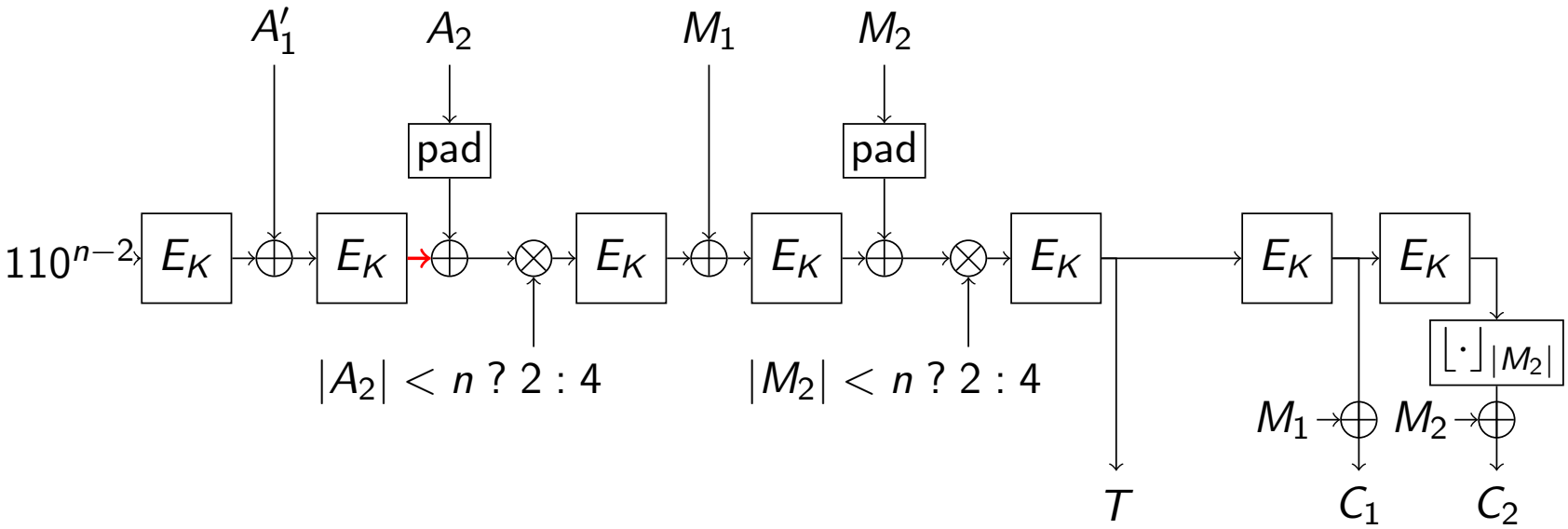
SUNDAE is not RUP Secure: INT-RUP Insecurity



2. $\mathcal{A}$ makes query $\mathcal{D}_K(\epsilon, T_2, C_2[1])$, where $T_2 = M_1[1] \oplus C_1[1] \oplus A[1]$ and obtains $M_2[1]$.

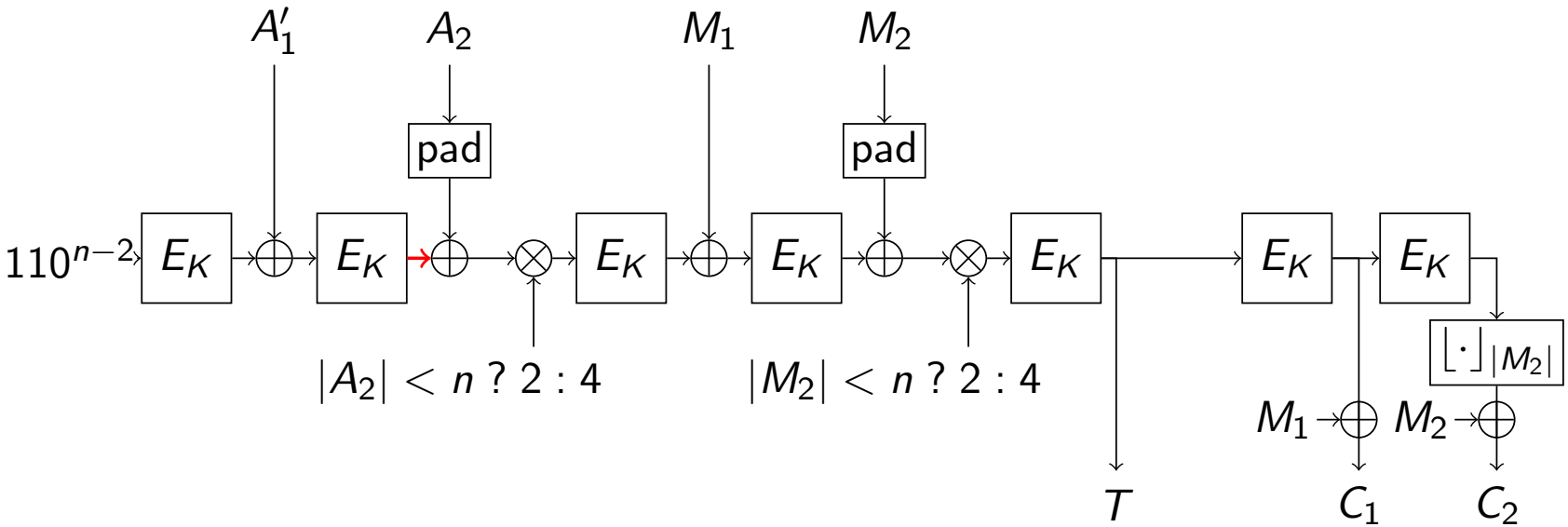
$\mathcal{A}$ **learns** $E_K(E_K(110^{n-2}) \oplus A[1]) = M_2[1] \oplus C_2[1]$**.**

SUNDAE is not RUP Secure: INT-RUP Insecurity



3. $\mathcal{A}$ makes query $\mathcal{D}_K(\epsilon, T_3, C_3[1])$, where $T_3 = M_3[1] \oplus C_3[1] \oplus A'[1] (\neq A[1])$ and obtains $M_3[1]$.

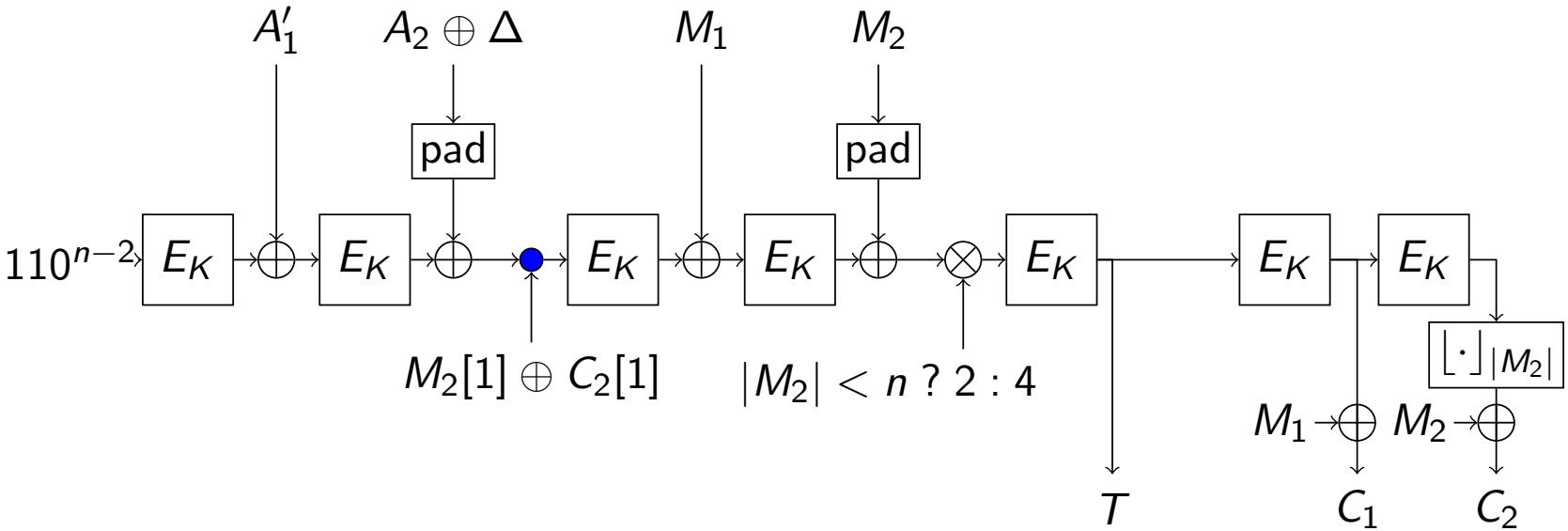
SUNDAE is not RUP Secure: INT-RUP Insecurity



3. $\mathcal{A}$ makes query $\mathcal{D}_K(\epsilon, T_3, C_3[1])$, where $T_3 = M_3[1] \oplus C_3[1] \oplus A'[1] (\neq A[1])$ and obtains $M_3[1]$.

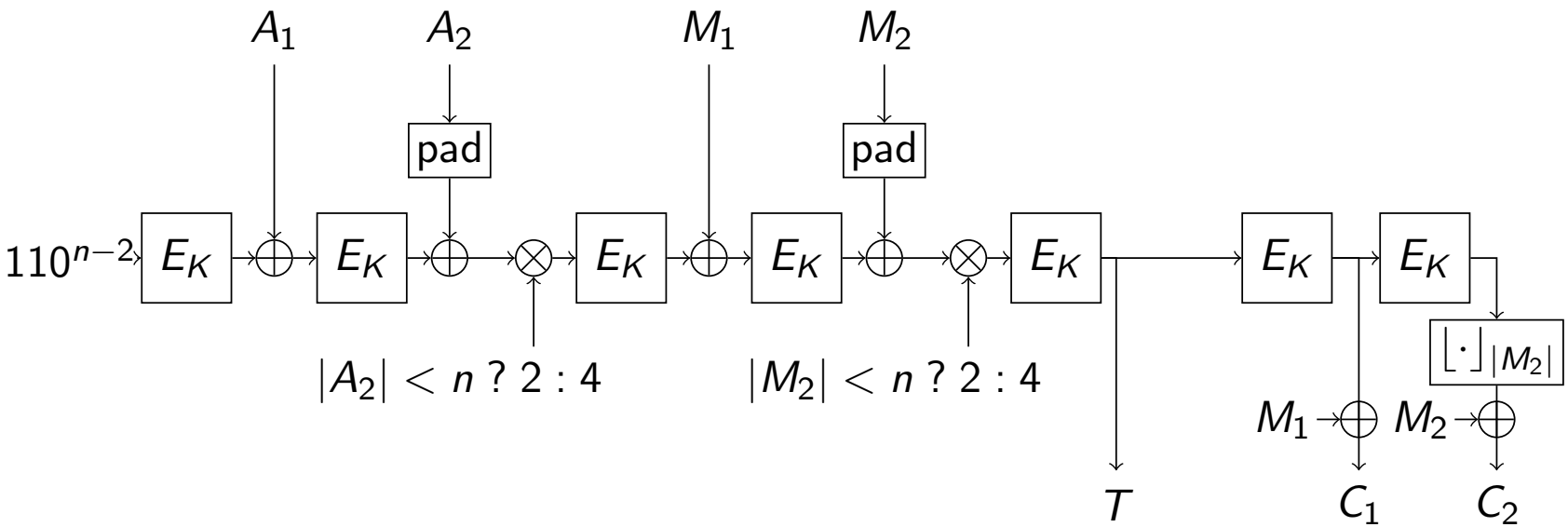
$\mathcal{A}$ **learns** $E_K(E_K(110^{n-2}) \oplus A'[1]) = M_3[1] \oplus C_3[1]$.

SUNDAE is not RUP Secure: INT-RUP Insecurity



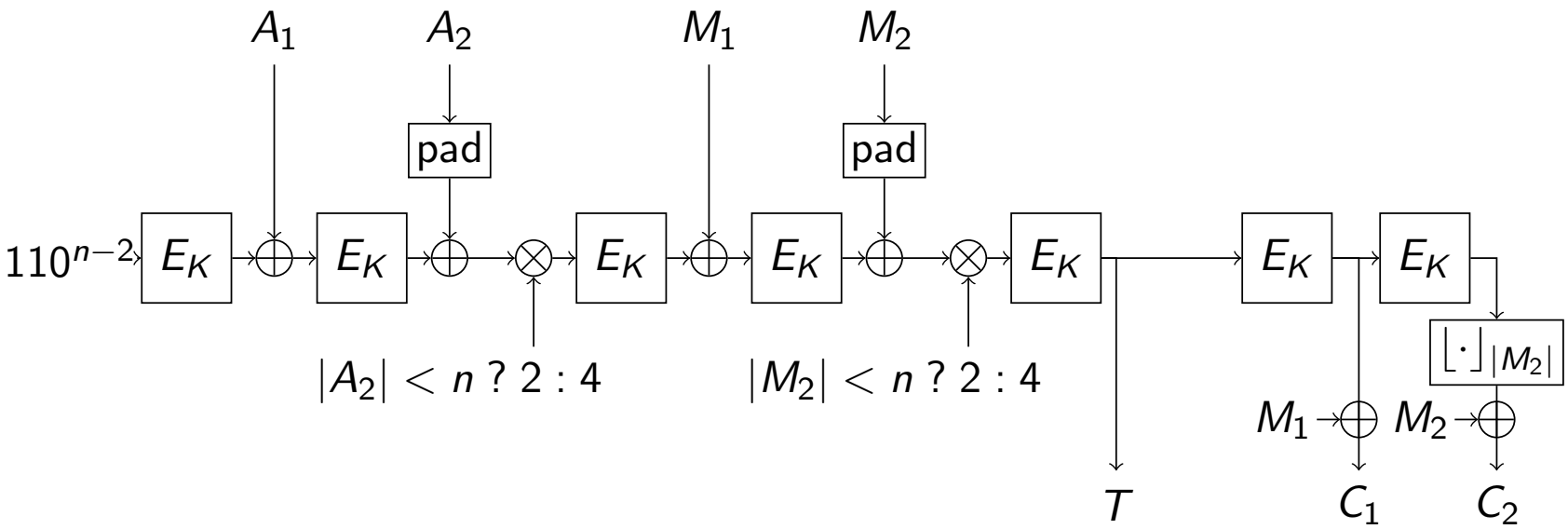
4. $\mathcal{A}$ makes query $\mathcal{E}_K(A'[1] \| (A[2] \oplus \Delta) \| A[3] \| \ldots \| A[a], M))$ and obtains (C, T), $\Delta = M_2[1] \oplus C_2[1] \oplus M_3[1] \oplus C_3[1]$.

SUNDAE is not RUP Secure: INT-RUP Insecurity



5. $\mathcal{A}$ forges with (A, T, C).

SUNDAE is not RUP Secure: INT-RUP Insecurity



5. $\mathcal{A}$ forges with (A, T, C).

SUNDAE is not INT-RUP Secure.

MONDAE: A RUP-Secure Variant of SUNDAE.

Remedy for INT-RUP Attack: MONDAE

Reason for INT-RUP attack on SUNDAE.

Adversary can learn $E_k(T)$ for any value of T.

Remedy for INT-RUP Attack: MONDAE

Reason for INT-RUP attack on SUNDAE.

Adversary can learn $E_k(T)$ for any value of T.

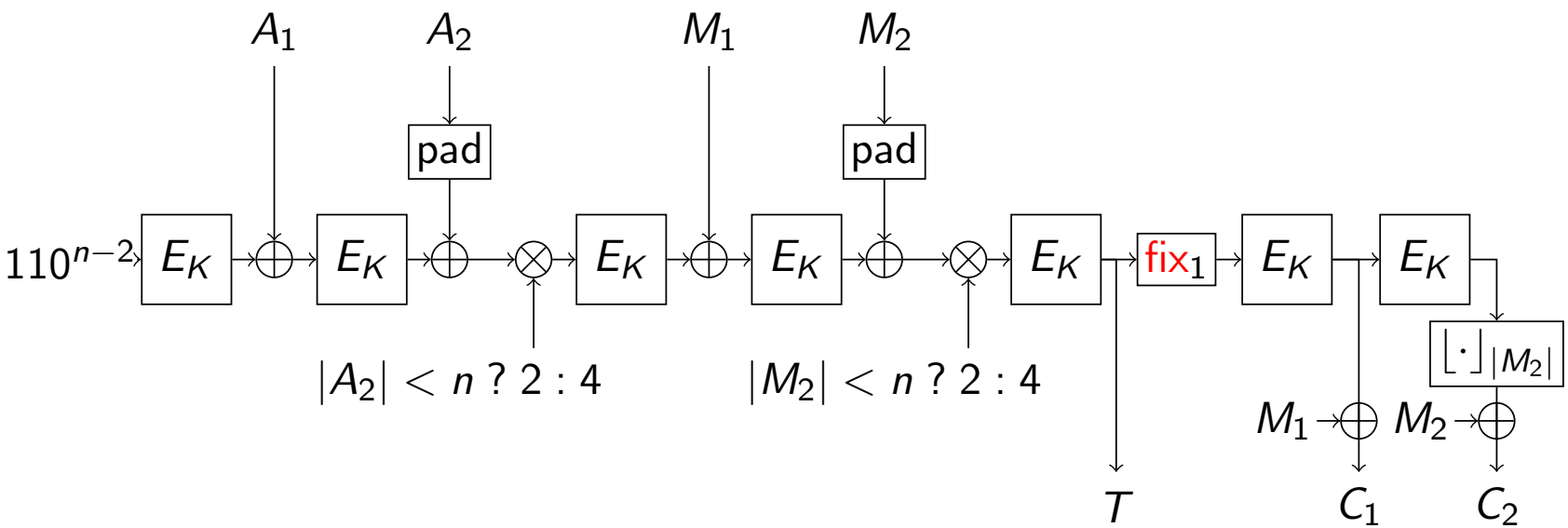
Can we make a small change to SUNDAE and make it RUP-Secure ?

Remedy for INT-RUP Attack: MONDAE

Reason for INT-RUP attack on SUNDAE.

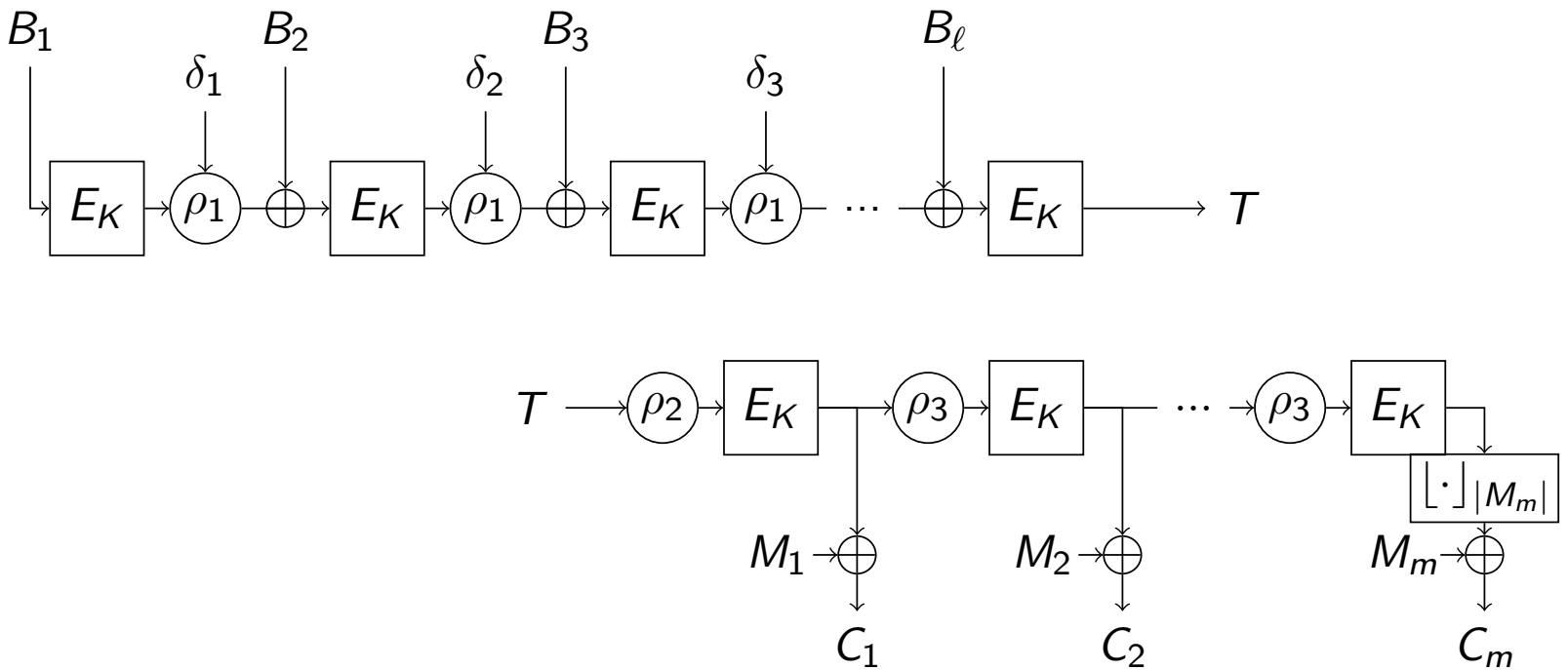
Adversary can learn $E_k(T)$ for any value of T.

Can we make a small change to SUNDAE and make it RUP-Secure ?



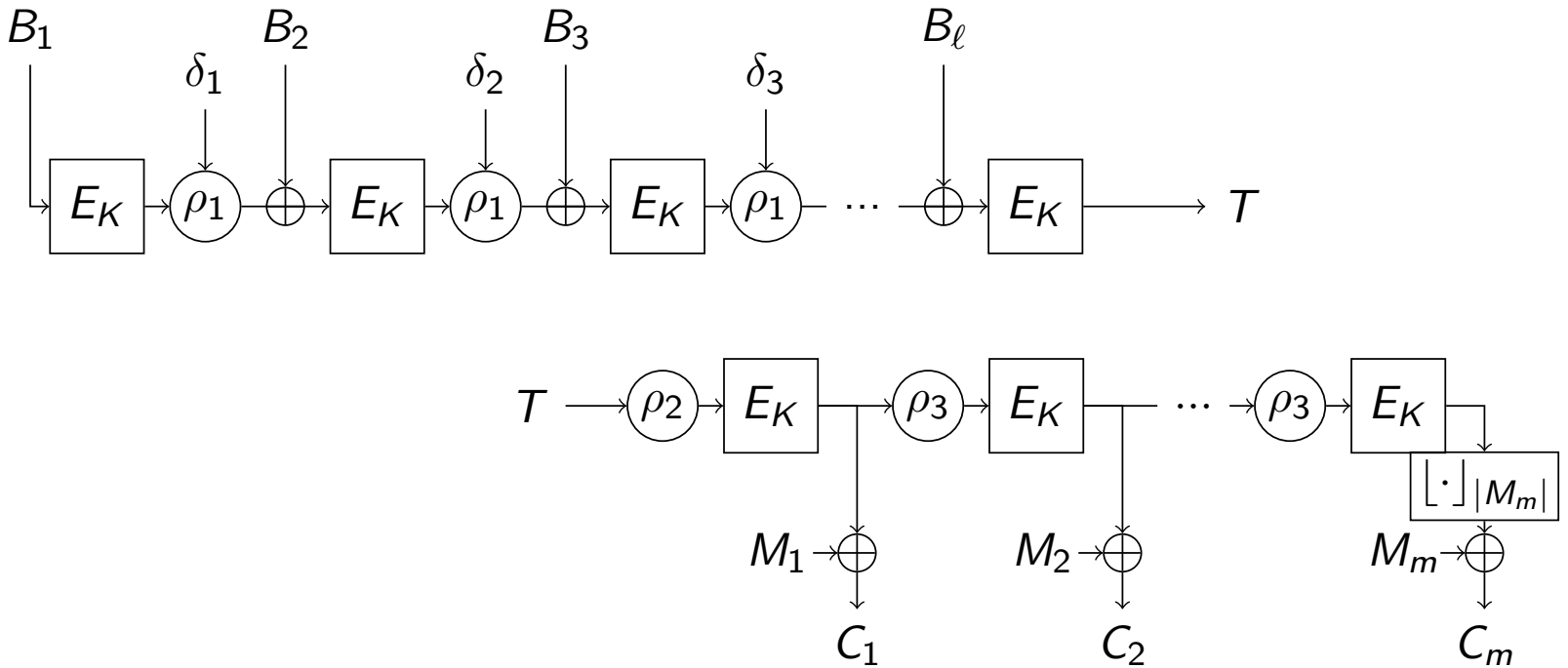
ANYDAE: A Generic RUP Secure AE

- $\mathsf{Fmt}(A, M) = ((B_1, \delta_1), \ldots, (B_{l-1}, \delta_{l-1}), B_l)$.
- $\rho_1(B_i, \delta_i) \to \{0,1\}^n$.
- $\rho_2, \rho_3 : \{0,1\}^n \to \{0,1\}^n$.



ANYDAE: A Generic RUP Secure AE

- $\mathsf{Fmt}(A, M) = ((B_1, \delta_1), \ldots, (B_{l-1}, \delta_{l-1}), B_l)$.
- $\rho_1(B_i, \delta_i) \to \{0,1\}^n$.
- $\rho_2, \rho_3 : \{0,1\}^n \to \{0,1\}^n$.



Is ANYDAE secure for any choice of Fmt, ρ_1, ρ_2 and ρ_3 function ?

Security of ANYDAE

$\mathcal{F}_1$ is the set of first block outputs of Fmt function.

Security of ANYDAE

$\mathcal{F}_1$ is the set of first block outputs of Fmt function.

- If Fmt is injective and prefix free function.
- ρ_1 is ϵ_1 differential uniform and γ_1 regular function.
- ρ_2 is γ_2 regular and ρ_3 is γ_3 regular functions.
- $\mathcal{F}_1$ is disjoint from the range of ρ_2.
- $\Omega := |\mathcal{F}_1 \cap \mathrm{range}(\rho_3)|$.

Security of ANYDAE

$\mathcal{F}_1$ is the set of first block outputs of Fmt function.

- If Fmt is injective and prefix free function.
- ρ_1 is ϵ_1 differential uniform and γ_1 regular function.
- ρ_2 is γ_2 regular and ρ_3 is γ_3 regular functions.
- $\mathcal{F}_1$ is disjoint from the range of ρ_2.
- $\Omega := |\mathcal{F}_1 \cap \mathrm{range}(\rho_3)|$.

Security Result

$$\mathbf{Adv}_{\mathrm{ANYDAE}}(\sigma, q_d) \lesssim \frac{\sigma^2}{2^n} + \Omega\sigma \cdot \gamma_3 + \frac{q_d}{2^n}.$$

MONDAE and TUESDAE: Instantiations of ANYDAE

- MONDAE is an instantiation of ANYDAE where ρ_2 is fix_1 function.
- TUESDAE is a n-bit state DAE scheme and hence optimal instantiation of ANYDAE.
- MONDAE and TUESDAE are INT-RUP secure.
- TUESDAE makes optimal number of BC calls.
- This optimality comes at the cost of some additional multiplexers which could slightly increase the hardware area.

Thank You For Your Attention.