Overloading the Nonce:

Rugged PRPs, Nonce-Set AEAD, and Order-Resilient Channels

Jean Paul Degabriele and Vukašin Karadžić

Cryptography Research Centre

Technology Innovation Institute

TECHNISCHE UNIVERSITÄT DARMSTADT

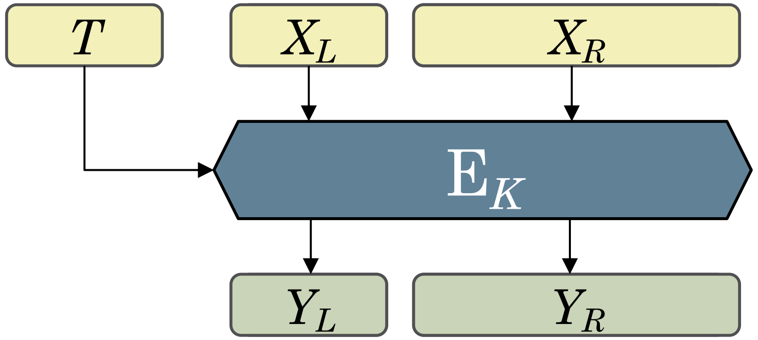
Technology Innovation Institute

Outline

- Defining Rugged PRPs
- The UIV Construction
- Transforming Rugged PRPs into AEAD
- Nonce-Set AEAD
- Order-Resilient Channels

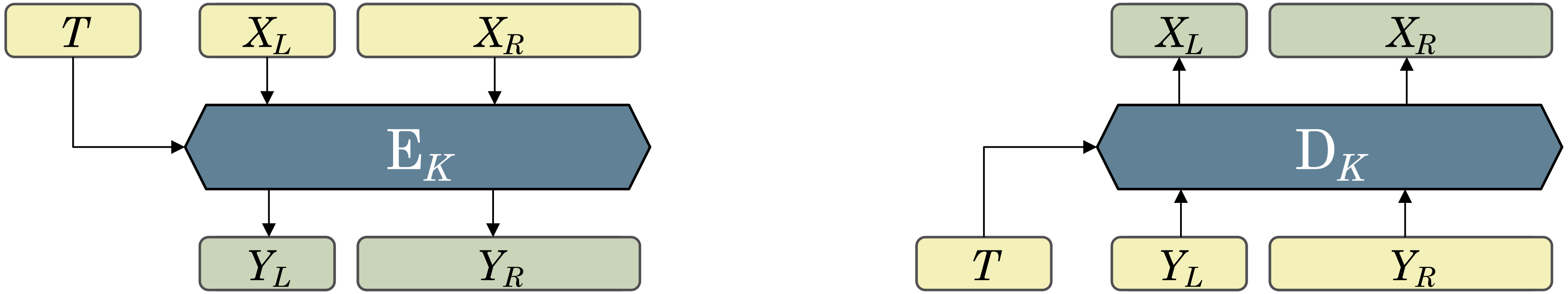
Defining Rugged PRPs

Rugged Pseudorandom Permutations



- Syntactically a Rugged PRP is a **(VIL) tweakable cipher** over a **split domain**: $\{0,1\}^n \times \{0,1\}^*$, where n is in the range 128-256 bits.

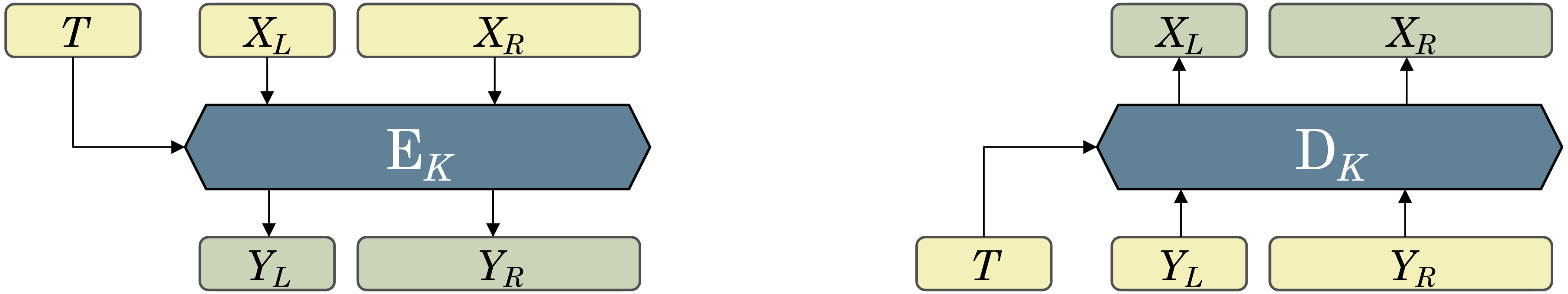
Rugged Pseudorandom Permutations



- Syntactically a Rugged PRP is a **(VIL) tweakable cipher** over a **split domain**: $\{0,1\}^n \times \{0,1\}^*$, where n is in the range 128-256 bits.

Rugged Pseudorandom Permutations

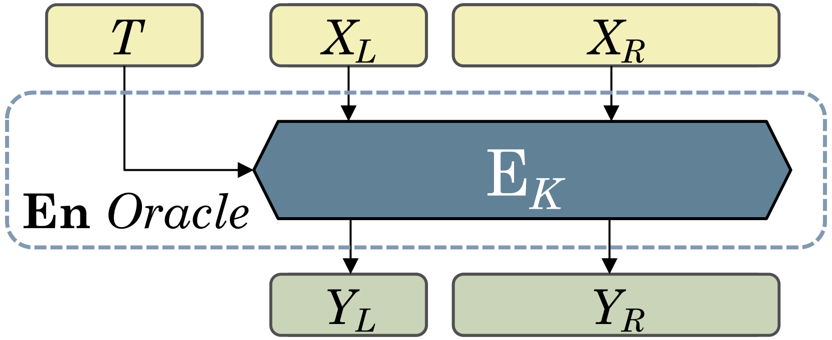
Cryptography Research Centre

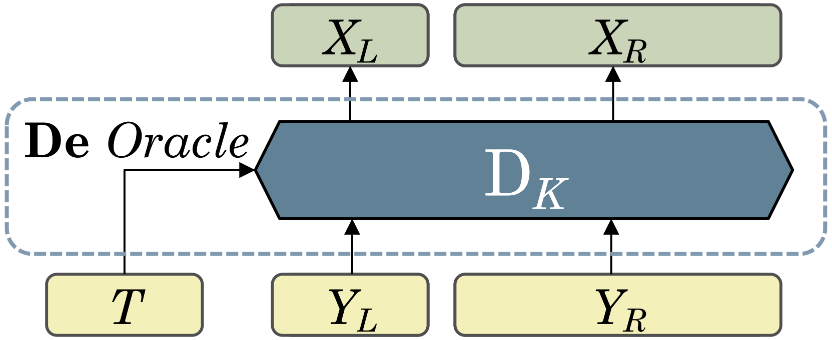


- Syntactically a Rugged PRP is a **(VIL) tweakable cipher** over a **split domain**: $\{0,1\}^n \times \{0,1\}^*$, where n is in the range 128-256 bits.
- **Intermediate Security**, between PRP and SPRP security.
- The adversary is only given **partial access** to the deciphering algorithm.

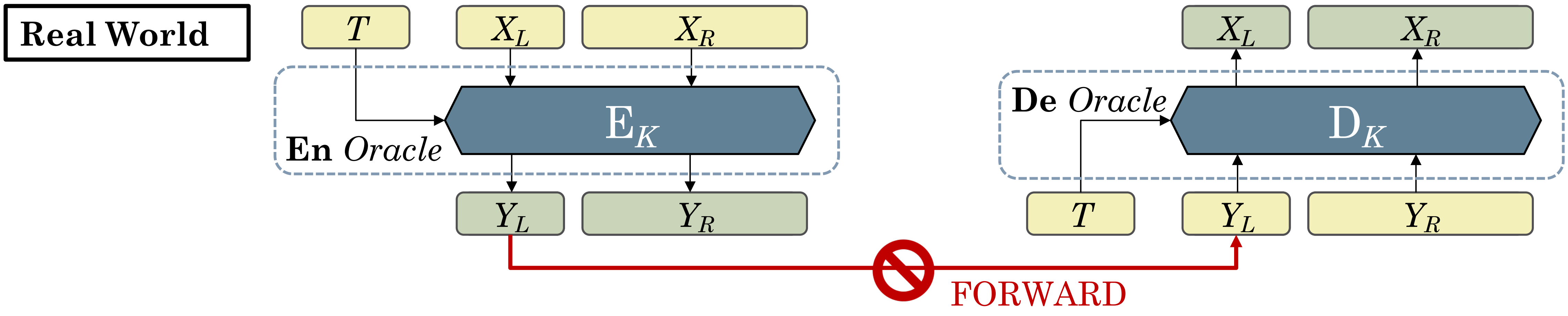
Rugged Pseudorandom Permutations

Real World

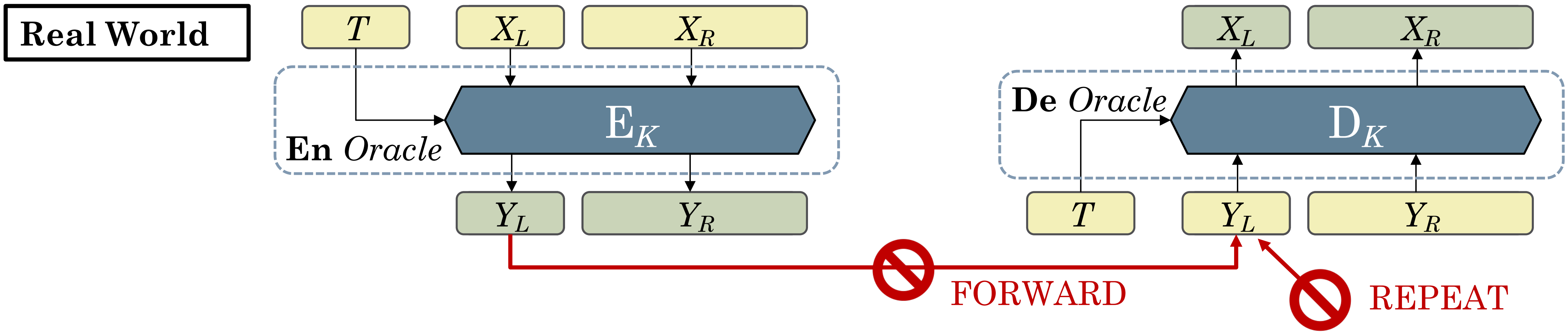




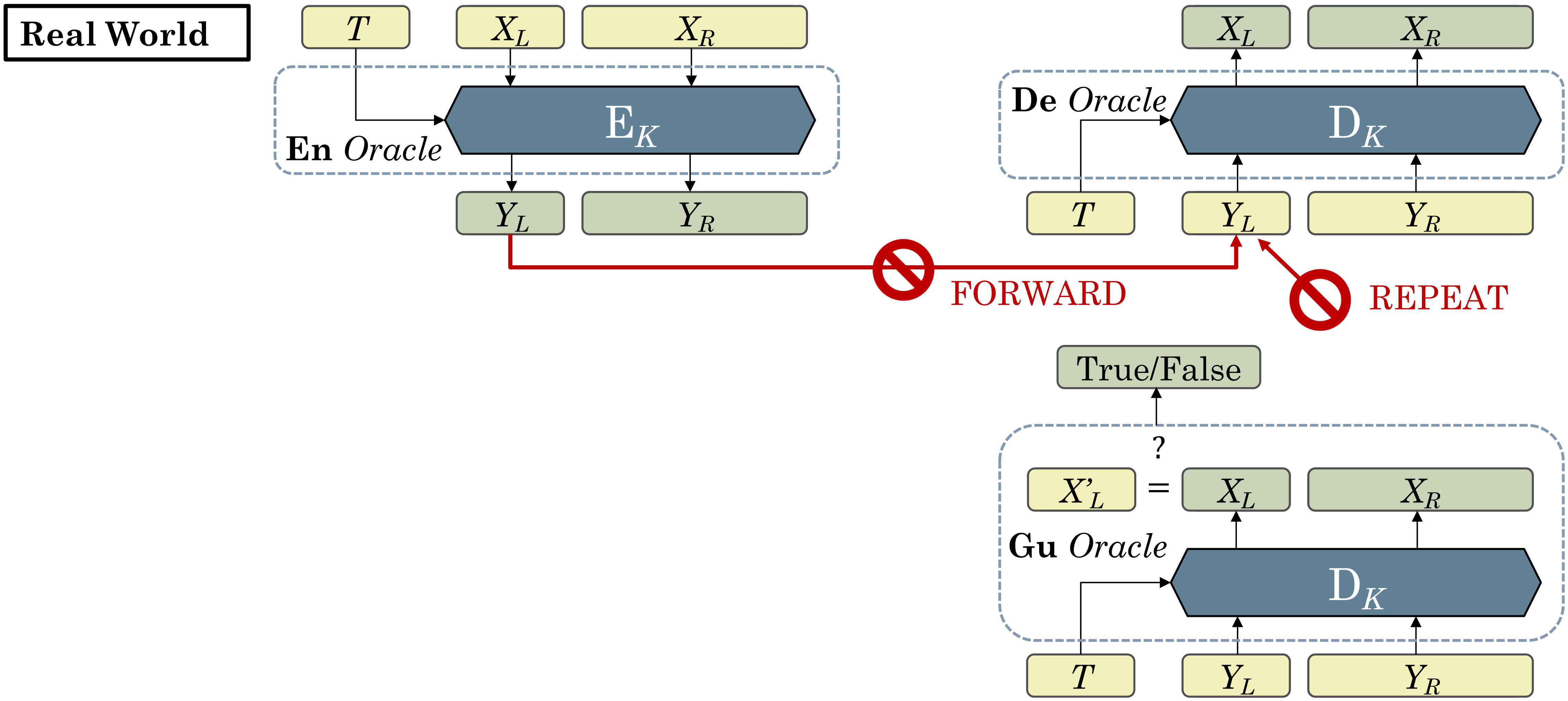
Rugged Pseudorandom Permutations



Rugged Pseudorandom Permutations

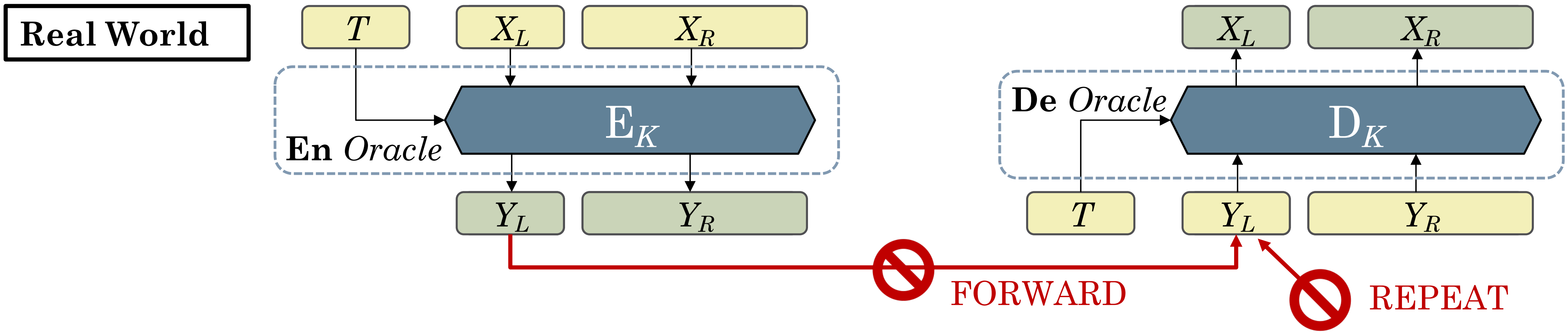


Rugged Pseudorandom Permutations



Rugged Pseudorandom Permutations

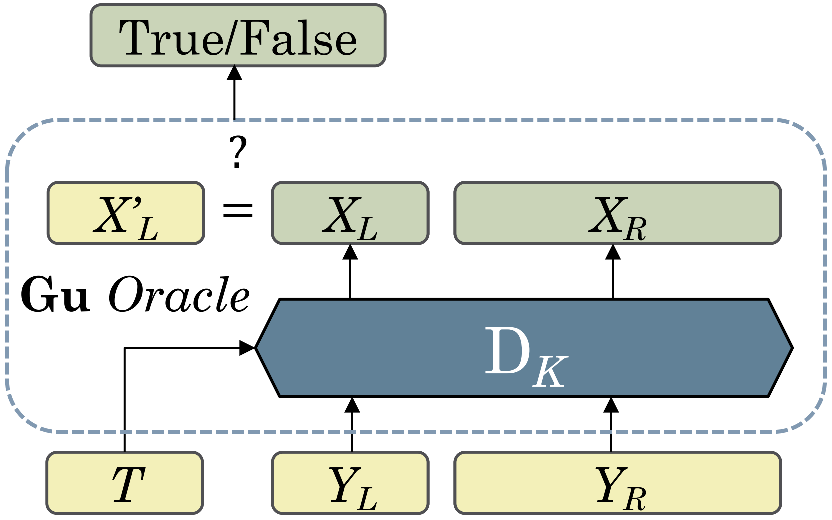
Cryptography Research Centre



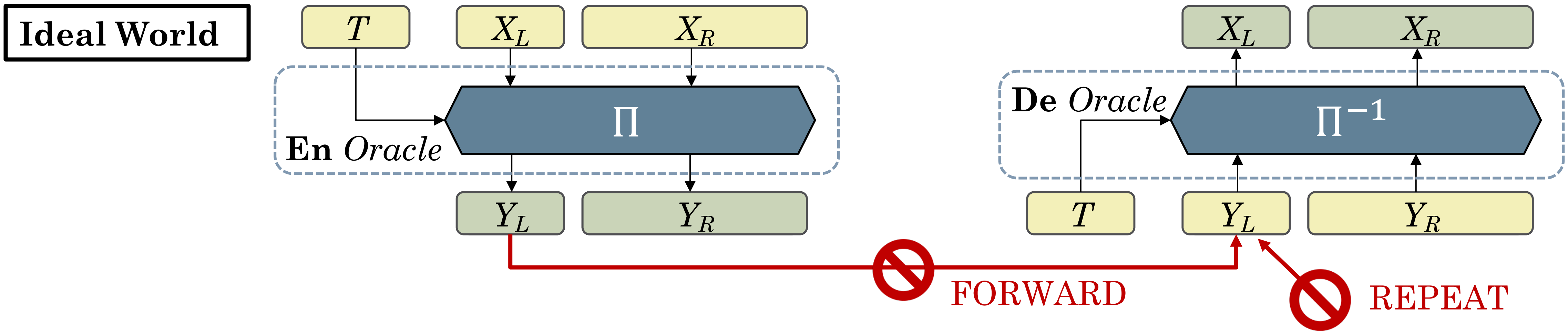
- Deciphering can be accessed via two separate oracles:

- **De** - restricted queries, full output.

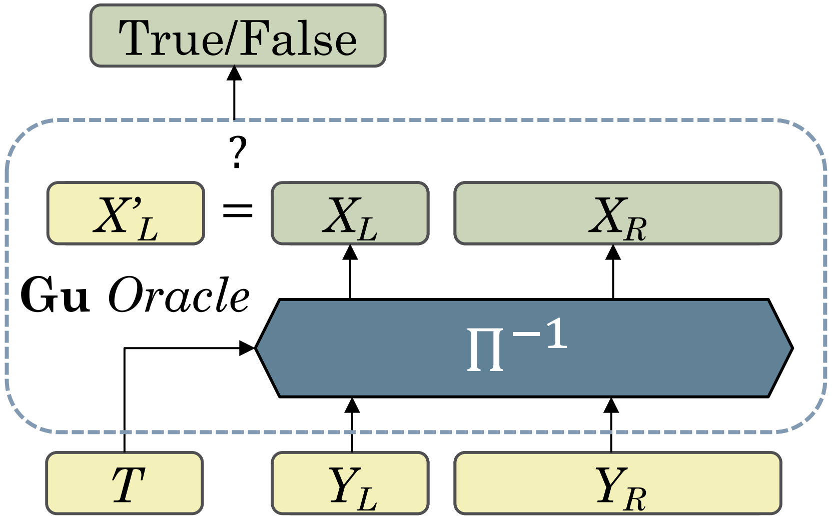
- **Gu** – unrestricted queries, 1-bit output.



Rugged Pseudorandom Permutations

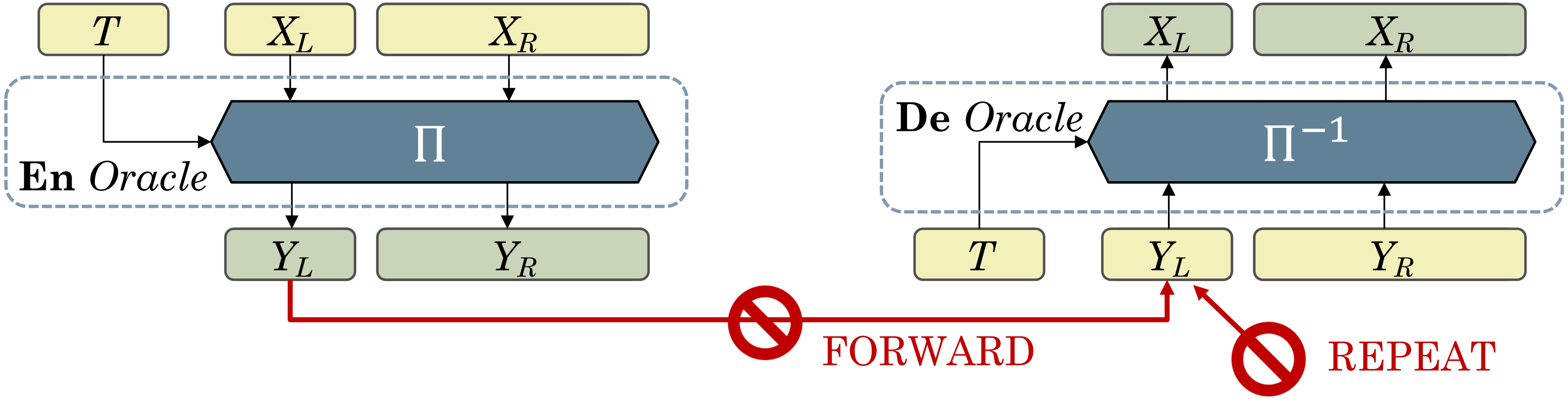


- Replace E_K with an ideal cipher Π.

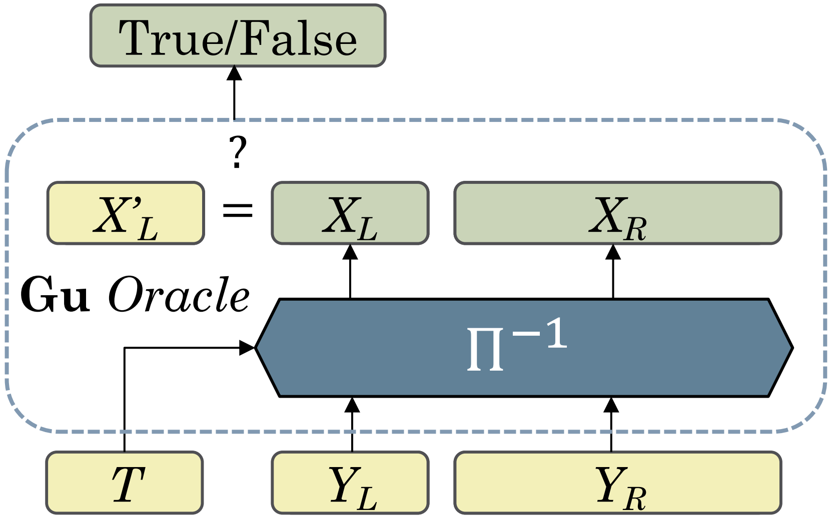


Rugged Pseudorandom Permutations

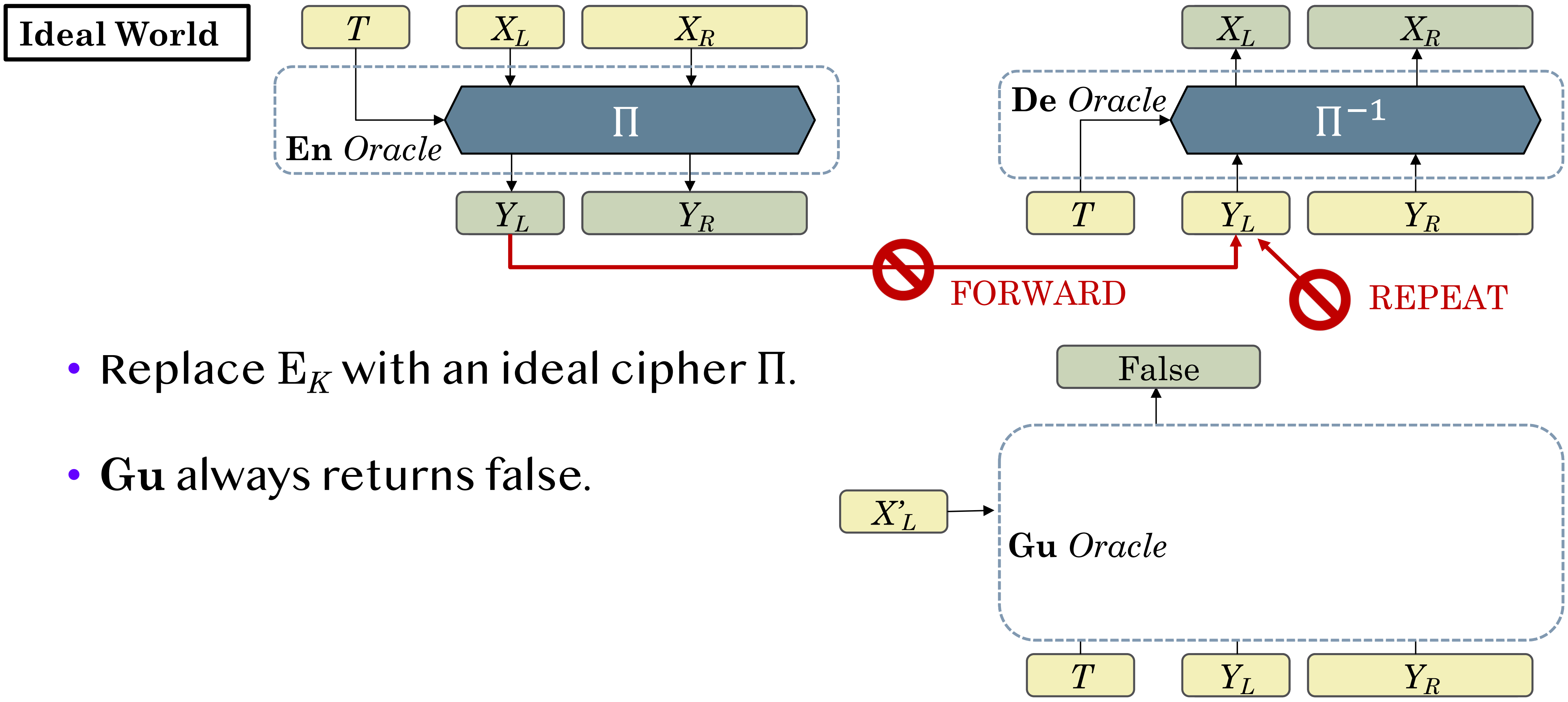
Ideal World



- Replace E_K with an ideal cipher Π.
- **Gu** always returns false.

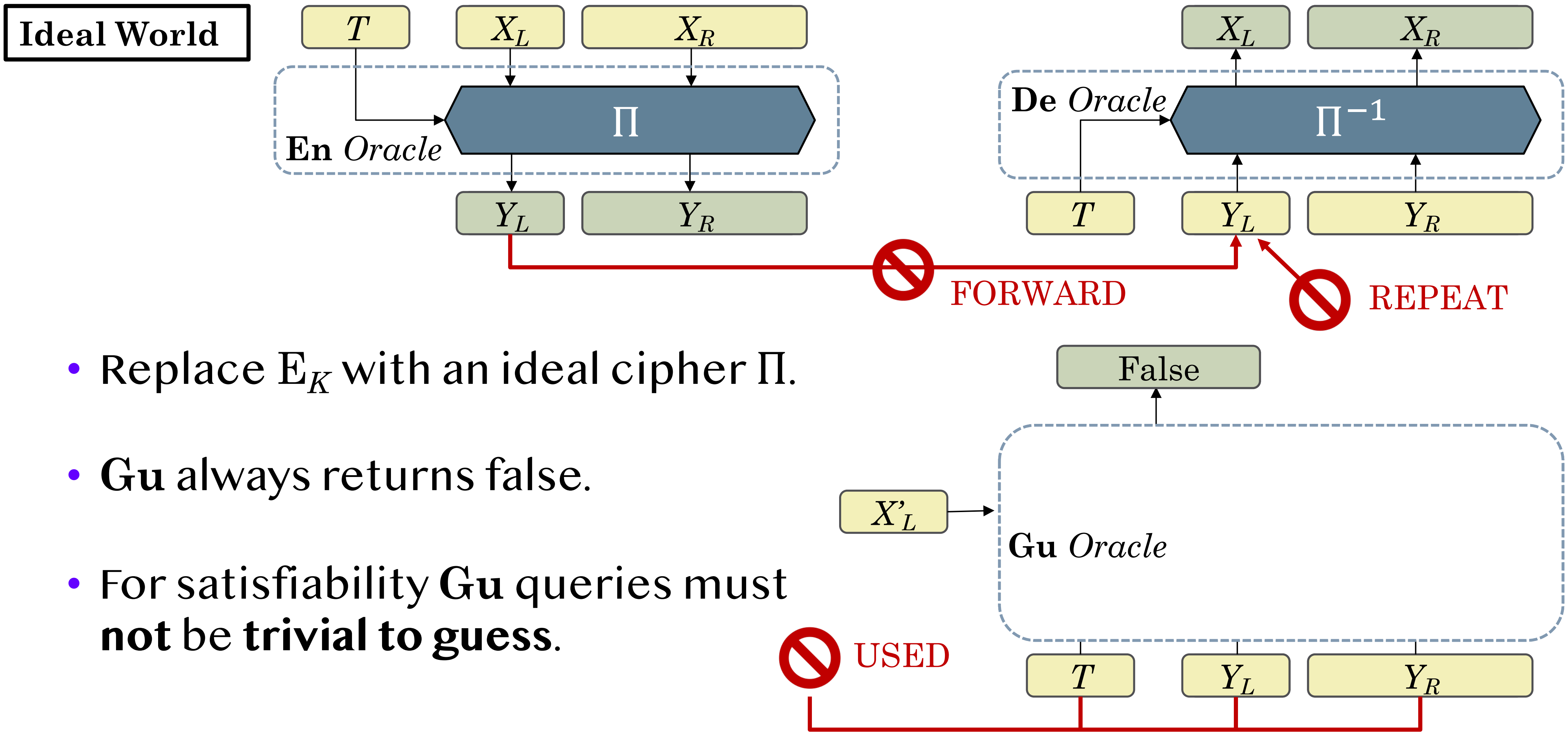


Rugged Pseudorandom Permutations



- Replace E_K with an ideal cipher Π.
- **Gu** always returns false.

Rugged Pseudorandom Permutations



- Replace E_K with an ideal cipher Π.
- **Gu** always returns false.
- For satisfiability **Gu** queries must **not** be **trivial to guess**.

Notes on the Definition

Cryptography Research Centre

- The term **rugged** is meant to reflect the **intermediate overall security** and the **asymmetry in security** between **enciphering** and **deciphering**.

Notes on the Definition

Cryptography Research Centre

- The term **rugged** is meant to reflect the **intermediate overall security** and the **asymmetry in security** between **enciphering** and **deciphering**.

- It is mainly intended for **variable-length ciphers** (not blockciphers) in the context of the **encode-then-encipher paradigm**.

- RPRPs are variable-length tweakable ciphers that can be easily transformed into AEAD with varying security properties.

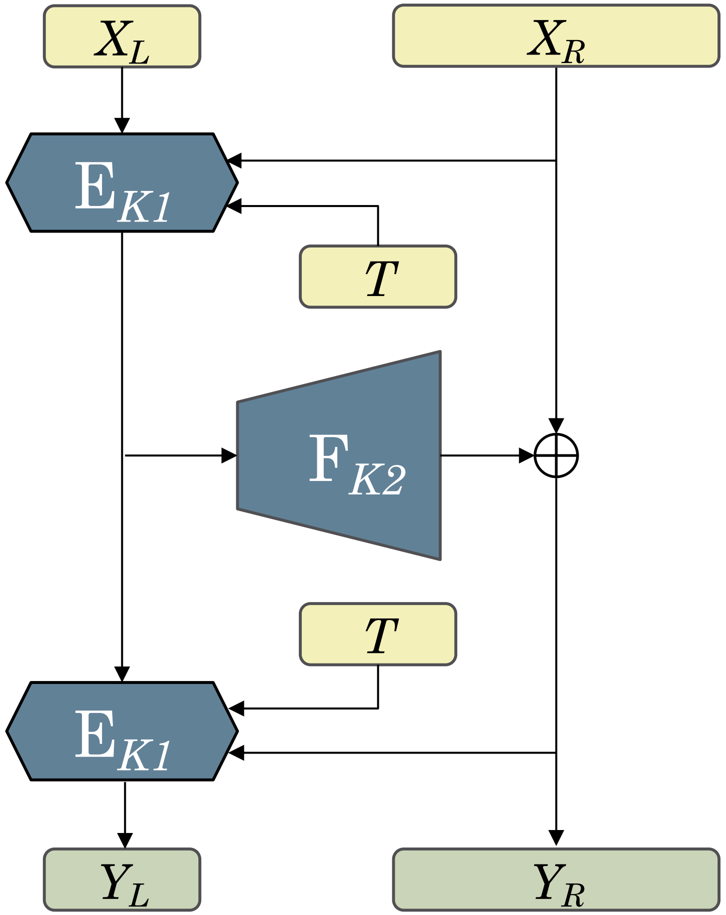
Notes on the Definition

Cryptography Research Centre

- The term **rugged** is meant to reflect the **intermediate overall security** and the **asymmetry in security** between **enciphering** and **deciphering**.
- It is mainly intended for **variable-length ciphers** (not blockciphers) in the context of the **encode-then-encipher paradigm**.
- RPRPs are variable-length tweakable ciphers that can be easily transformed into AEAD with varying security properties.
- The definition is itself motivated by the **encode-then-encipher paradigm** and **features common** to variable-length cipher **constructions**.

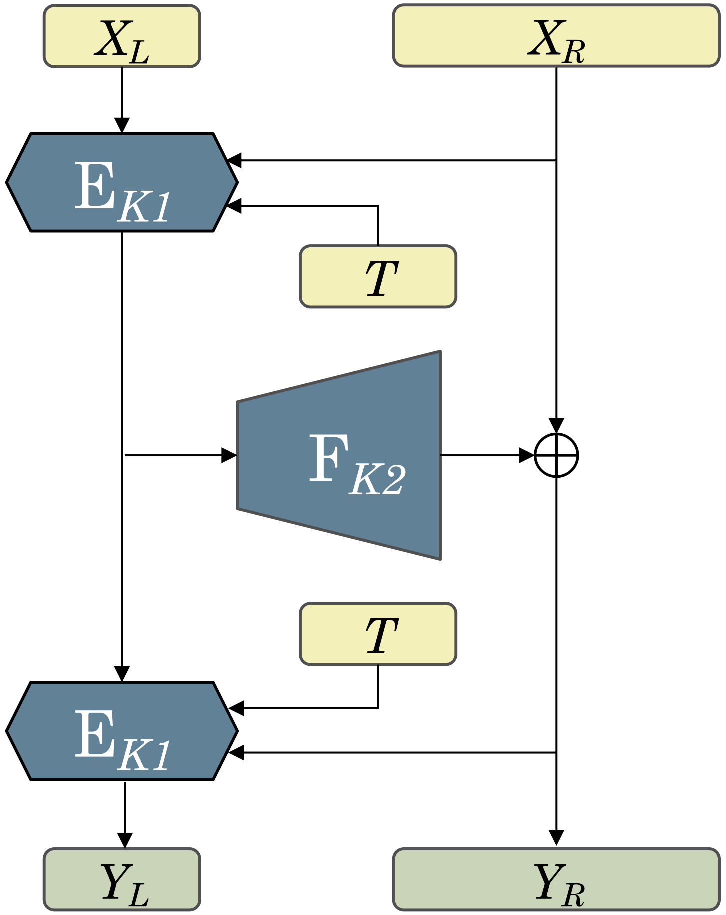
The UIV Construction

Protected IV [ShrTer13]



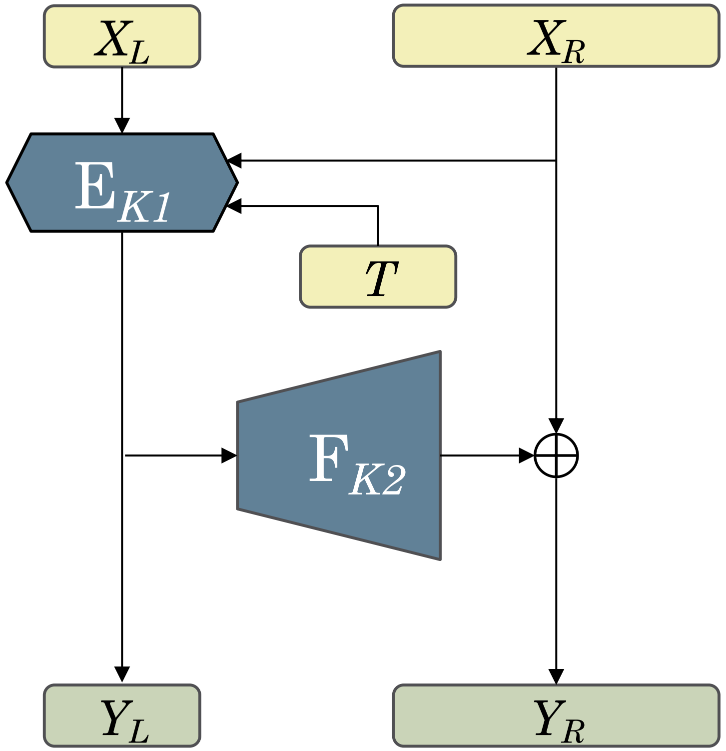
- PIV is a **(VIL) tweakable cipher construction** that is **SPRP** secure.

Protected IV [ShrTer13]



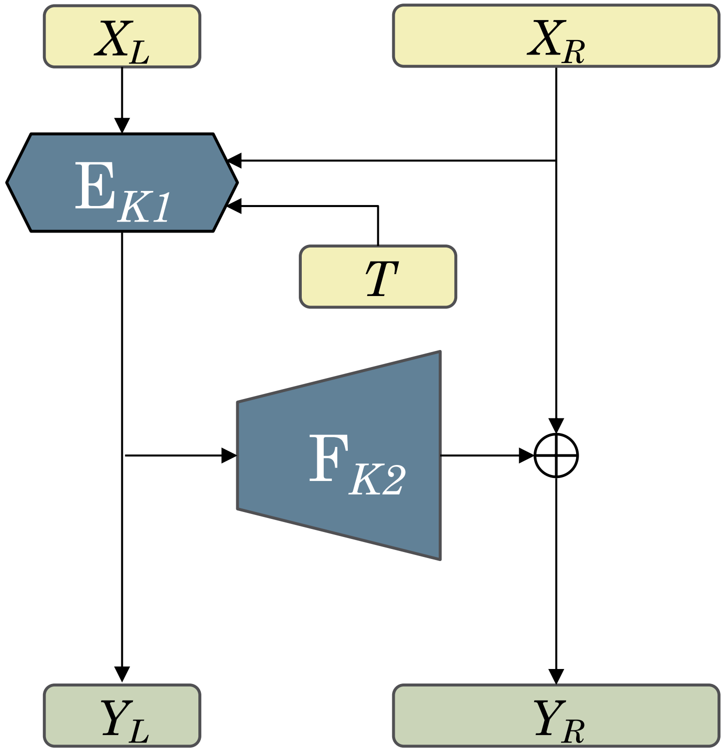
- PIV is a **(VIL) tweakable cipher construction** that is **SPRP** secure.
- Shown here as consisting of a **VOL-PRF** F_{K2} and two **FIL tweakable cipher** instances E_{K1}.
- A typical instantiation of F_{K2} is **AES-CTR** where the IV acts as the VOL-PRF input.

Unilaterally-Protected IV



- **UIV** is obtained simply by dropping the **third layer** and it can be shown to be **RPRP secure**.

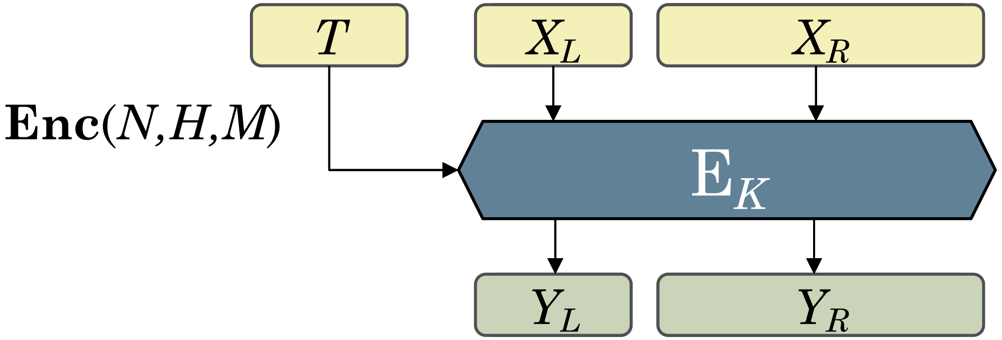
Unilaterally-Protected IV



- **UIV** is obtained simply by dropping the **third layer** and it can be shown to be **RPRP secure**.
- It can be instantiated with **GCM components** leading to a **performance** similar to GCM-SIV.
- It is closely related to **MiniCTR** [Min15] and **GCM-RUP** [ADL17].

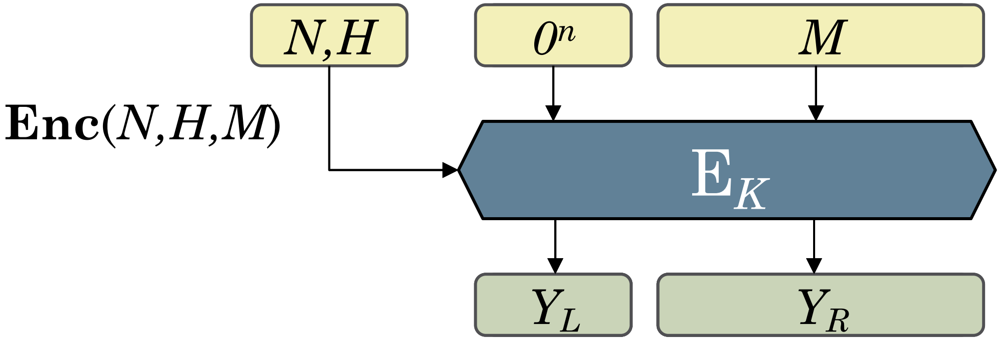
Transforming RPRPs into AEAD

The EtE Transform



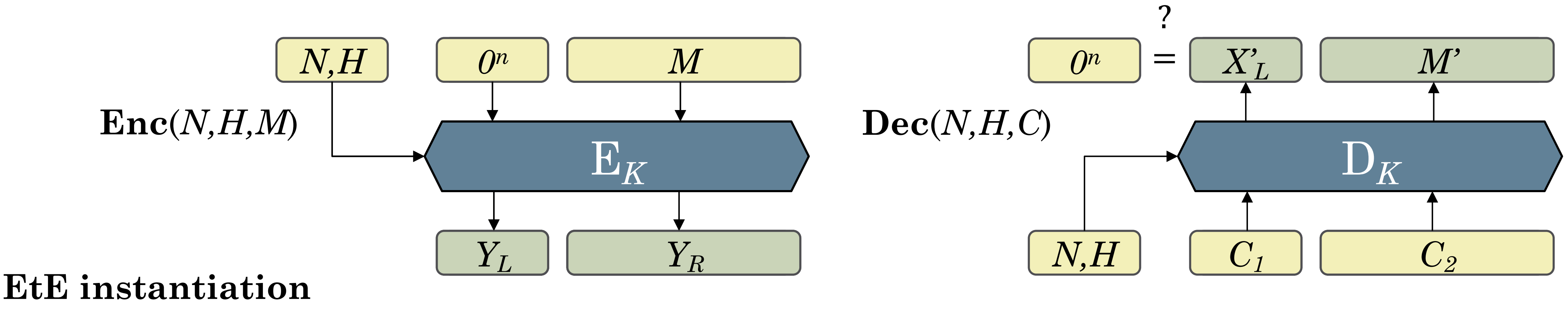
- We revisit and adapt the **Encode-then-Encipher paradigm** [BelRog00, ShrTer13] in the context of RPRPs.

The EtE Transform



- We revisit and adapt the **Encode-then-Encipher paradigm** [BelRog00, ShrTer13] in the context of RPRPs.

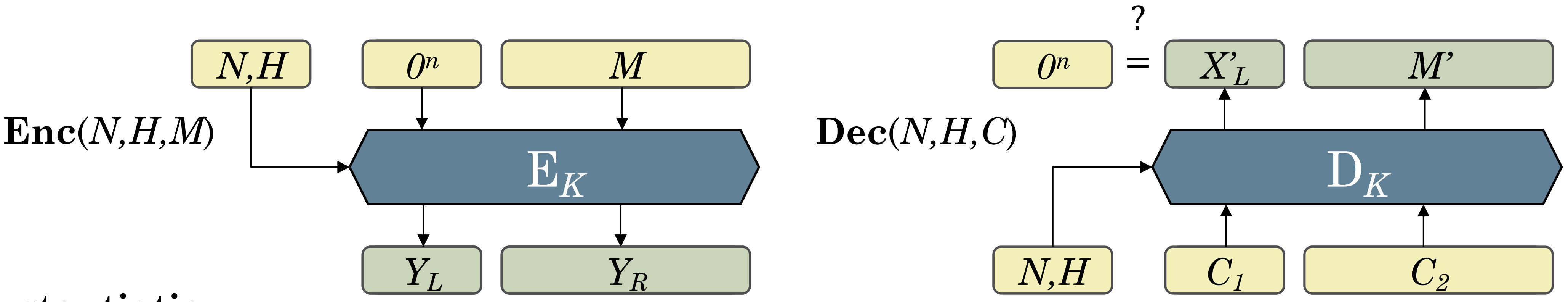
The EtE Transform



- We revisit and adapt the **Encode-then-Encipher paradigm** [BelRog00, ShrTer13] in the context of RPRPs.

The EtE Transform

Cryptography Research Centre

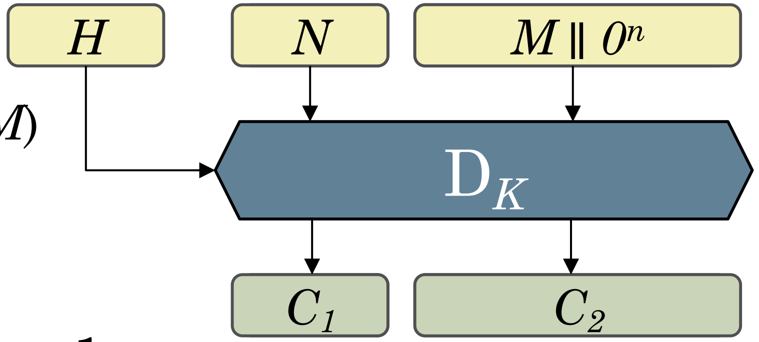


EtE instantiation

- We revisit and adapt the **Encode-then-Encipher paradigm** [BelRog00, ShrTer13] in the context of RPRPs.
- EtE is slightly more general, the above is a specific instantiation of it.
- $(\mathbf{E}_K, \mathbf{D}_K)$ is RPRP secure $\Longrightarrow$ EtE is **Misuse-Resistant AEAD**.

The EtD Transform

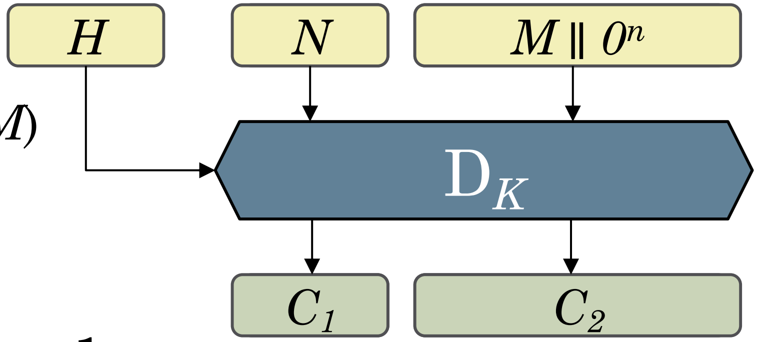
Enc(N,H,M)



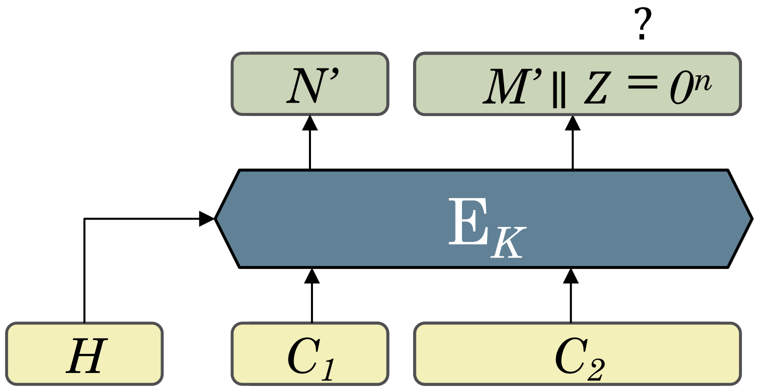
EtD instantiation 1

The EtD Transform

Enc(N,H,M)

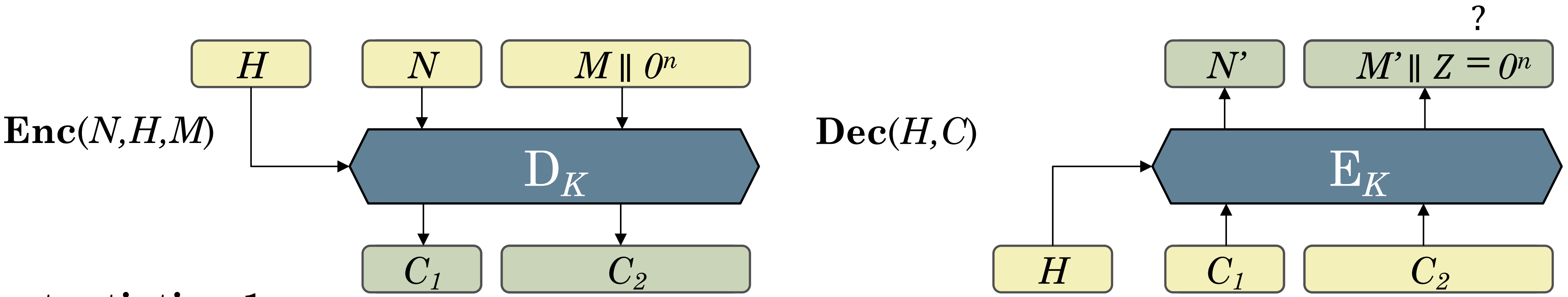


Dec(H,C)



EtD instantiation 1

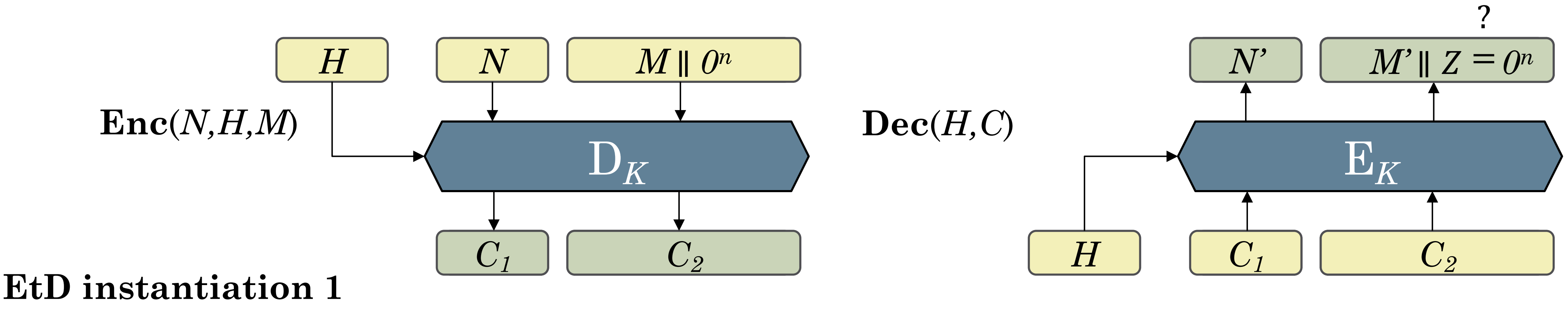
The EtD Transform



EtD instantiation 1

- $(\mathbf{E}_K, \mathbf{D}_K)$ is RPRP secure $\Longrightarrow$ EtD yields a **RUPAE nonce-hiding AEAD**.

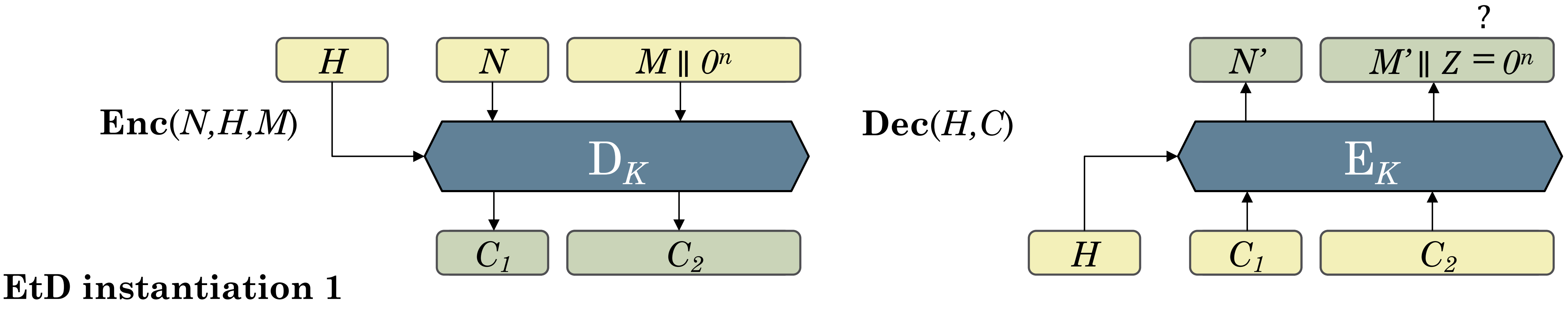
The EtD Transform



- $(\mathbf{E}_K, \mathbf{D}_K)$ is RPRP secure $\Longrightarrow$ EtD yields a **RUPAE nonce-hiding AEAD**.

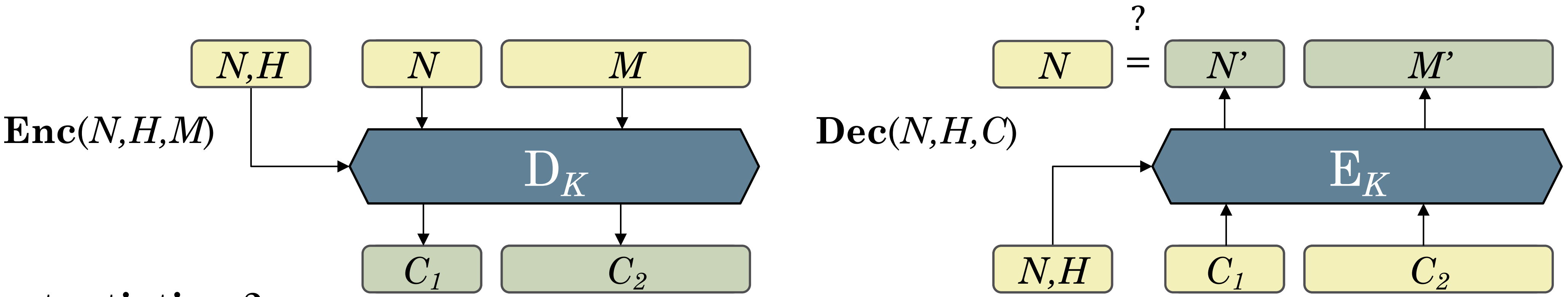
- When the tweakable cipher is GCM-UIV this instantiation of EtD corresponds to **GCM-RUP** [ADL17].

The EtD Transform



- However we can instantiate it differently to reduce the ciphertext expansion by using the **nonce to authenticate** the ciphertext.

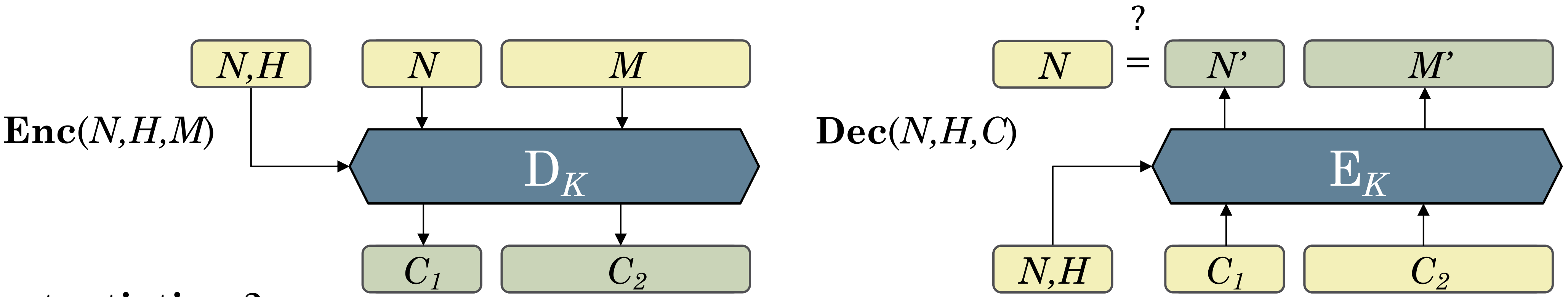
The EtD Transform



EtD instantiation 2

- However we can instantiate it differently to reduce the ciphertext expansion by using the **nonce to authenticate** the ciphertext.

The EtD Transform



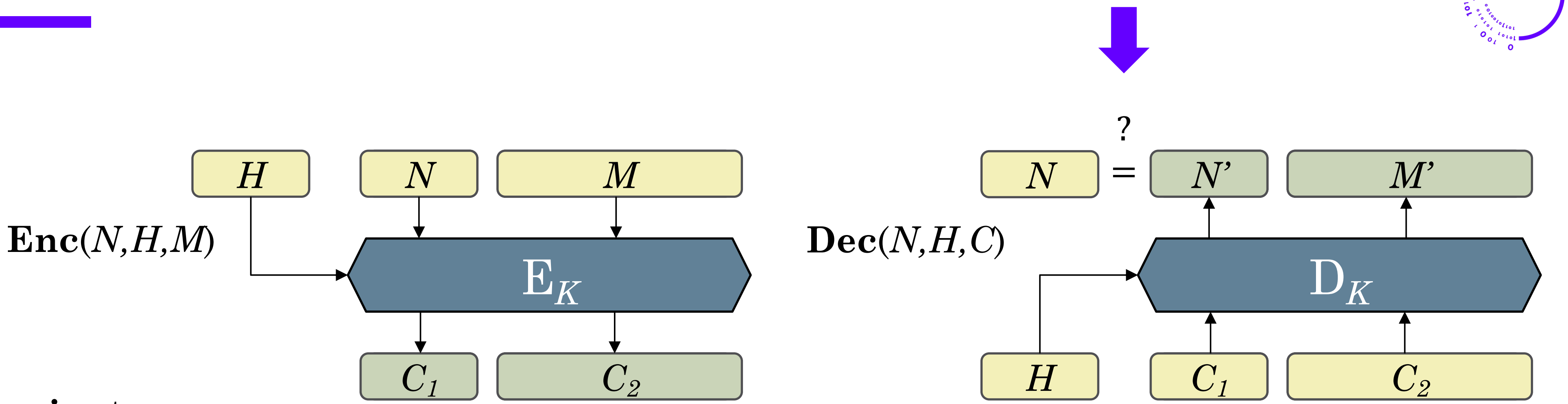
EtD instantiation 2

- However we can instantiate it differently to reduce the ciphertext expansion by using the **nonce to authenticate** the ciphertext.

- $(\mathbf{E}_K, \mathbf{D}_K)$ is RPRP secure $\Longrightarrow$ EtD is a (standard) **AEAD** that is **RUPAE** secure.

Nonce-Set AEAD

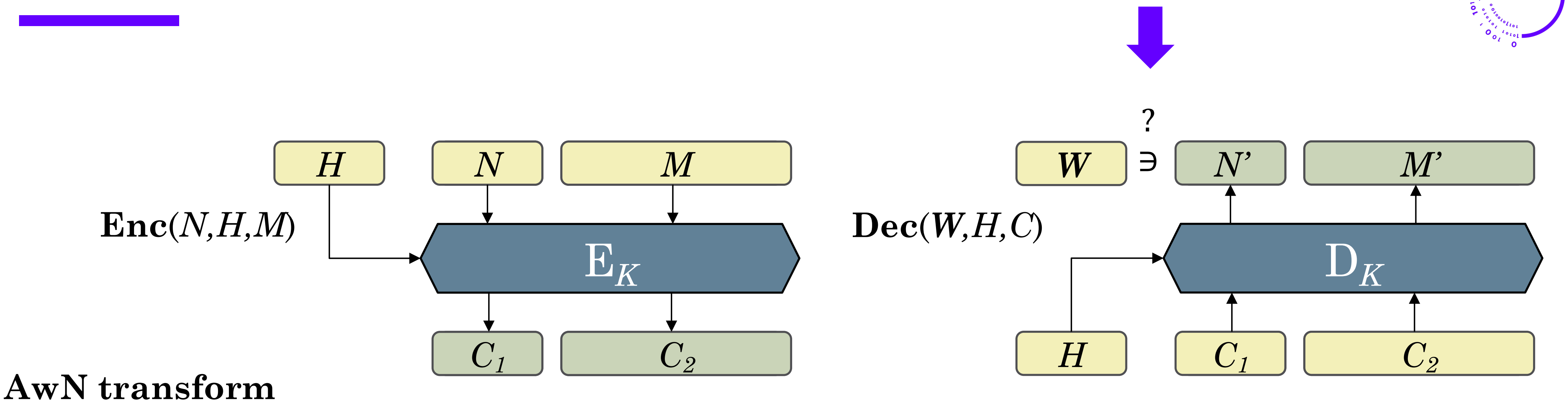
The AwN Transform



EtE variant

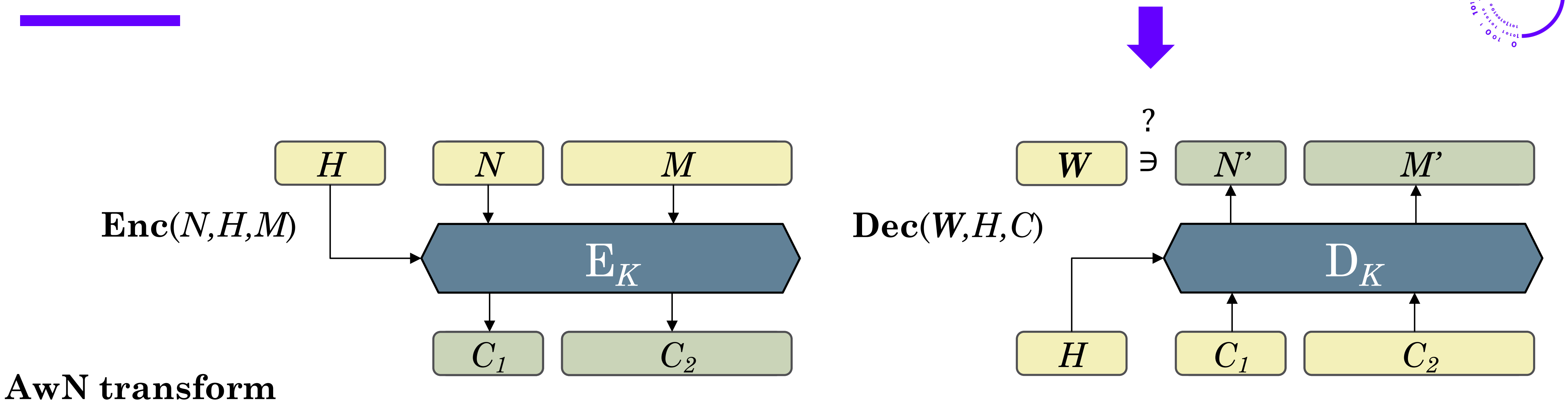
- We can also use the **nonce to authenticate** in the **EtE** transform and obtain a nonce-hiding AEAD (E_K = UIV ⇒ MiniCTR [Min15]).

The AwN Transform



- We can also use the **nonce to authenticate** in the **EtE** transform and obtain a nonce-hiding AEAD (E_K = UIV ⇒ MiniCTR [Min15]).
- We can generalize this further by **testing the nonce for set membership** instead of equality, yielding the **AwN** transform.

The AwN Transform



- We can also use the **nonce to authenticate** in the **EtE** transform and obtain a nonce-hiding AEAD (E_K = UIV ⇒ MiniCTR [Min15]).

- We can generalize this further by **testing the nonce for set membership** instead of equality, yielding the **AwN** transform.

- **AwN** transforms an RPRP into a **Nonce-Set AEAD** that is **Misuse-Resistant**.

Nonce-Set AEAD Formally

- **Syntactically** the difference is in the decryption algorithm:

$$(N', M')/(\perp, \perp) \leftarrow \mathrm{Dec}_K(\boldsymbol{W}, H, C) \text{ where } N' \in \boldsymbol{W}$$

Nonce-Set AEAD Formally

- **Syntactically** the difference is in the decryption algorithm:

$$(N', M')/(\perp, \perp) \leftarrow \mathrm{Dec}_K(\boldsymbol{W}, H, C) \text{ where } N' \in \boldsymbol{W}$$

- **Correctness** requires that for all $K, N, H, M, \boldsymbol{W}$ such that $N \in \boldsymbol{W}$,

$$\text{If } C \leftarrow \mathrm{Enc}_{\mathrm{K}}(N, H, M) \text{ then } (N, M) \leftarrow \mathrm{Dec}_{\mathrm{K}}(\boldsymbol{W}, H, C).$$

Nonce-Set AEAD Formally

- **Syntactically** the difference is in the decryption algorithm:

$$(N', M')/(\perp, \perp) \leftarrow \mathrm{Dec}_K(\boldsymbol{W}, H, C) \text{ where } N' \in \boldsymbol{W}$$

- **Correctness** requires that for all $K, N, H, M, \boldsymbol{W}$ such that $N \in \boldsymbol{W}$,

$$\text{If } C \leftarrow \mathrm{Enc}_K(N, H, M) \text{ then } (N, M) \leftarrow \mathrm{Dec}_K(\boldsymbol{W}, H, C).$$

- **(MR)AE security** translates in a straightforward manner, we only need to adapt the prohibited queries:

$$\text{If } C \leftarrow \mathrm{Enc}_K(N, H, M) \text{ then no queries } \mathrm{Dec}_K(\boldsymbol{W}, H, C) \text{ where } N \in \boldsymbol{W} \text{ can be made by the adversary.}$$

Why Nonce-Set AEAD?

Cryptography Research Centre

- It is a natural primitive in the context of **order-resilient channels** such as **QUIC** and **DTLS** which employ window mechanisms.
- Nonce-Set AEAD serves as a **stepping stone** from which a variety of **secure channel functionalities** can be easily realized.

Why Nonce-Set AEAD?

Cryptography Research Centre

- It is a natural primitive in the context of **order-resilient channels** such as **QUIC** and **DTLS** which employ window mechanisms.
- Nonce-Set AEAD serves as a **stepping stone** from which a variety of **secure channel functionalities** can be easily realized.
- Nonce-Set AEAD can also be constructed from **any nonce-hiding AEAD** via a straightforward generic transform.
- However **AwN** realizes Nonce-Set AEAD directly resulting in **more compact ciphertexts** than this generic transform.

Order-Resilient Channels

Order-Resilient Channels

- **QUIC** and **DTLS** realize secure channels over UDP and need to handle out-of-order delivery.

- Several possibilities arise for handling **reorderings**, **replays**, **modifications**, and **deletions**, and how much of each to tolerate.

Order-Resilient Channels

Cryptography Research Centre

- **QUIC** and **DTLS** realize secure channels over UDP and need to handle out-of-order delivery.
- Several possibilities arise for handling **reorderings**, **replays**, **modifications**, and **deletions**, and how much of each to tolerate.
- Typical constructions employ one or more **window mechanisms**, which add complexity—making them **hard to understand and analyze**.
- In general, it is unclear how these **additional mechanisms** interact with AEAD and what the **overall security** of the channel is.

The Support Predicate

- The various functionalities of such channels can be formally characterized by a **support predicate**:

$$\boldsymbol{accept/reject \leftarrow supp(C, C_S, DC_R)}$$

The Support Predicate

- The various functionalities of such channels can be formally characterized by a **support predicate**:

$$\boldsymbol{accept/reject \leftarrow supp(C, C_S, DC_R)}$$

- It was developed in [Bac19, FGJ20] as a **generalization** of the **silencing approach** by [RogZha18].

- The support predicate permeates into all aspects of the secure channel **correctness**, **security**, and **robustness** [FGJ20].

Order-Resilient Channels from NS-AEAD

$\mathsf{Init}()$	$\mathsf{Send}(\mathrm{stk}_s, A, M)$	$\mathsf{Recv}(\mathrm{stk}_r, A, C)$
$(\mathrm{st}_s, \mathrm{st}_r) \leftarrow\$ \mathsf{StInit}()$	$(\mathrm{st}_s, K) \leftarrow \mathrm{stk}_s$	$(\mathrm{st}_r, K) \leftarrow \mathrm{stk}_r$
$K \leftarrow\$ \{0,1\}^k$	$(\mathrm{st}'_s, N) \leftarrow \mathsf{NonceExtract}(\mathrm{st}_s)$	$\boldsymbol{W} \leftarrow \mathsf{NonceSetPolicy}(\mathrm{st}_r)$
$\mathrm{stk}_s \leftarrow (\mathrm{st}_s, K)$	$\mathbf{if}\ N = \perp\ \mathbf{then}$	$(N, M) \leftarrow \mathsf{Dec}(K, \boldsymbol{W}, A, C)$
$\mathrm{stk}_r \leftarrow (\mathrm{st}_r, K)$	$\quad \mathbf{return}\ (\mathrm{st}'_s, \perp)$	$\mathbf{if}\ (N, M) = (\perp, \perp)\ \mathbf{then}$
$\mathbf{return}\ (\mathrm{stk}_s, \mathrm{stk}_r)$	$C \leftarrow \mathsf{Enc}(K, N, A, M)$	$\quad mn \leftarrow \perp$
	$\mathrm{stk}'_s \leftarrow (\mathrm{st}'_s, K)$	$\mathbf{else}$
	$\mathbf{return}\ (\mathrm{stk}'_s, C)$	$\quad (\mathrm{st}'_r, mn) \leftarrow \mathsf{StUpdate}(\mathrm{st}_r, N)$
		$\mathrm{stk}'_r \leftarrow (\mathrm{st}'_r, K)$
		$\mathbf{return}\ (\mathrm{stk}'_r, mn, M)$

- We present a **universal** and **generic** channel construction from Nonce-Set AEAD for **any desired support predicate**!

Order-Resilient Channels from NS-AEAD

$\mathsf{Init}()$	$\mathsf{Send}(\mathrm{stk}_s, A, M)$	$\mathsf{Recv}(\mathrm{stk}_r, A, C)$
$(\mathrm{st}_s, \mathrm{st}_r) \leftarrow\$ \mathsf{StInit}()$ $K \leftarrow\$ \{0,1\}^k$ $\mathrm{stk}_s \leftarrow (\mathrm{st}_s, K)$ $\mathrm{stk}_r \leftarrow (\mathrm{st}_r, K)$ $\mathbf{return}\ (\mathrm{stk}_s, \mathrm{stk}_r)$	$(\mathrm{st}_s, K) \leftarrow \mathrm{stk}_s$ $(\mathrm{st}'_s, N) \leftarrow \mathsf{NonceExtract}(\mathrm{st}_s)$ $\mathbf{if}\ N = \perp\ \mathbf{then}$ $\quad \mathbf{return}\ (\mathrm{st}'_s, \perp)$ $C \leftarrow \mathsf{Enc}(K, N, A, M)$ $\mathrm{stk}'_s \leftarrow (\mathrm{st}'_s, K)$ $\mathbf{return}\ (\mathrm{stk}'_s, C)$	$(\mathrm{st}_r, K) \leftarrow \mathrm{stk}_r$ $\boldsymbol{W} \leftarrow \mathsf{NonceSetPolicy}(\mathrm{st}_r)$ $(N, M) \leftarrow \mathsf{Dec}(K, \boldsymbol{W}, A, C)$ $\mathbf{if}\ (N, M) = (\perp, \perp)\ \mathbf{then}$ $\quad mn \leftarrow \perp$ $\mathbf{else}$ $\quad (\mathrm{st}'_r, mn) \leftarrow \mathsf{StUpdate}(\mathrm{st}_r, N)$ $\mathrm{stk}'_r \leftarrow (\mathrm{st}'_r, K)$ $\mathbf{return}\ (\mathrm{stk}'_r, mn, M)$

- We present a **universal** and **generic** channel construction from Nonce-Set AEAD for **any desired support predicate**!
- The construction consists of a **Nonce-Set AEAD** (blue) scheme and a **Nonce-Set Processing (NSP)** scheme (red).

Order-Resilient Channels from NS-AEAD

Cryptography Research Centre

$\mathsf{Init}()$

$(\mathrm{st}_s, \mathrm{st}_r) \leftarrow\$ \mathsf{StInit}()$

$K \leftarrow\$ \{0,1\}^k$

$\mathrm{stk}_s \leftarrow (\mathrm{st}_s, K)$

$\mathrm{stk}_r \leftarrow (\mathrm{st}_r, K)$

$\mathbf{return}\ (\mathrm{stk}_s, \mathrm{stk}_r)$

$\mathsf{Send}(\mathrm{stk}_s, A, M)$

$(\mathrm{st}_s, K) \leftarrow \mathrm{stk}_s$

$(\mathrm{st}'_s, N) \leftarrow \mathsf{NonceExtract}(\mathrm{st}_s)$

$\mathbf{if}\ N = \perp\ \mathbf{then}$

$\quad \mathbf{return}\ (\mathrm{st}'_s, \perp)$

$C \leftarrow \boxed{\mathsf{Enc}(K, N, A, M)}$

$\mathrm{stk}'_s \leftarrow (\mathrm{st}'_s, K)$

$\mathbf{return}\ (\mathrm{stk}'_s, C)$

$\mathsf{Recv}(\mathrm{stk}_r, A, C)$

$(\mathrm{st}_r, K) \leftarrow \mathrm{stk}_r$

$\boldsymbol{W} \leftarrow \mathsf{NonceSetPolicy}(\mathrm{st}_r)$

$(N, M) \leftarrow \boxed{\mathsf{Dec}(K, \boldsymbol{W}, A, C)}$

$\mathbf{if}\ (N, M) = (\perp, \perp)\ \mathbf{then}$

$\quad mn \leftarrow \perp$

$\mathbf{else}$

$\quad (\mathrm{st}'_r, mn) \leftarrow \mathsf{StUpdate}(\mathrm{st}_r, N)$

$\mathrm{stk}'_r \leftarrow (\mathrm{st}'_r, K)$

$\mathbf{return}\ (\mathrm{stk}'_r, mn, M)$

- We present a **universal** and **generic** channel construction from Nonce-Set AEAD for **any desired support predicate**!
- The construction consists of a **Nonce-Set AEAD** (blue) scheme and a **Nonce-Set Processing (NSP)** scheme (red).

Order-Resilient Channels from NS-AEAD

$\mathsf{Init}()$	$\mathsf{Send}(\mathrm{stk}_s, A, M)$	$\mathsf{Recv}(\mathrm{stk}_r, A, C)$
$(\mathrm{st}_s, \mathrm{st}_r) \leftarrow\$ \boxed{\mathsf{StInit}()}$	$(\mathrm{st}_s, K) \leftarrow \mathrm{stk}_s$	$(\mathrm{st}_r, K) \leftarrow \mathrm{stk}_r$
$K \leftarrow\$ \{0,1\}^k$	$(\mathrm{st}'_s, N) \leftarrow \boxed{\mathsf{NonceExtract}(\mathrm{st}_s)}$	$\boldsymbol{W} \leftarrow \boxed{\mathsf{NonceSetPolicy}(\mathrm{st}_r)}$
$\mathrm{stk}_s \leftarrow (\mathrm{st}_s, K)$	**if** $N = \perp$ **then**	$(N, M) \leftarrow \boxed{\mathsf{Dec}(K, \boldsymbol{W}, A, C)}$
$\mathrm{stk}_r \leftarrow (\mathrm{st}_r, K)$	**return** $(\mathrm{st}'_s, \perp)$	**if** $(N, M) = (\perp, \perp)$ **then**
return $(\mathrm{stk}_s, \mathrm{stk}_r)$	$C \leftarrow \boxed{\mathsf{Enc}(K, N, A, M)}$	$mn \leftarrow \perp$
	$\mathrm{stk}'_s \leftarrow (\mathrm{st}'_s, K)$	**else**
	return (stk'_s, C)	$(\mathrm{st}'_r, mn) \leftarrow \boxed{\mathsf{StUpdate}(\mathrm{st}_r, N)}$
		$\mathrm{stk}'_r \leftarrow (\mathrm{st}'_r, K)$
		return (stk'_r, mn, M)

- We present a **universal** and **generic** channel construction from Nonce-Set AEAD for **any desired support predicate**!
- The construction consists of a **Nonce-Set AEAD** (blue) scheme and a **Nonce-Set Processing (NSP)** scheme (red).

Order-Resilient Channels from NS-AEAD

- We prove this channel construction **correct**, **robust**, and **secure** in a generic way for **any support predicate**.

- We only require that the **Nonce-Set AEAD** is secure and that the **NSP scheme** satisfy a functionality property called **faithfulness.**

Order-Resilient Channels from NS-AEAD

Cryptography Research Centre

- We prove this channel construction **correct**, **robust**, and **secure** in a generic way for **any support predicate**.
- We only require that the **Nonce-Set AEAD** is secure and that the **NSP scheme** satisfy a functionality property called **faithfulness.**
- Informally, faithfulness says that the **NSP scheme** accurately reproduces the **support predicate logic over the nonces**.
- One can simply **tune the NSP** to the **desired functionality** and plug in their favourite **Nonce-Set AEAD** and **security/robustness** will be **automatic**.

Concluding Remarks

Summary

- Rugged PRPs strike a new **tradeoff** between **security** and **performance**.
- In particular, we have shown that the **Encode-then-Encipher** paradigm can be made to work with **weaker variable-length ciphers** than SPRPs.

Summary

- Rugged PRPs strike a new **tradeoff** between **security** and **performance**.
- In particular, we have shown that the **Encode-then-Encipher** paradigm can be made to work with **weaker variable-length ciphers** than SPRPs.
- The new notion allowed a **systematic exploration** of the different **AEAD** and **NS-AEAD** schemes that can be realized from **UIV**.
- We can look for **alternative RPRP constructions** and plug them into our template constructions.

Summary

Cryptography Research Centre

- Rugged PRPs strike a new **tradeoff** between **security** and **performance**.
- In particular, we have shown that the **Encode-then-Encipher** paradigm can be made to work with **weaker variable-length ciphers** than SPRPs.
- The new notion allowed a **systematic exploration** of the different **AEAD** and **NS-AEAD** schemes that can be realized from **UIV**.
- We can look for **alternative RPRP constructions** and plug them into our template constructions.
- NS-AEAD draws a **clean abstraction boundary** for understanding order-resilient channels, **separating security** from **channel functionality**.