

# Symmetric-Key Cryptography

---

Charlotte Lefevre

CentraleSupélec, 6 October 2026

# Roadmap

1. Introduction
2. Indistinguishability and Proofs
3. Authenticated Encryption
4. Hash Functions
5. Sponges
6. Accordion Modes
7. Conclusion

## Introduction

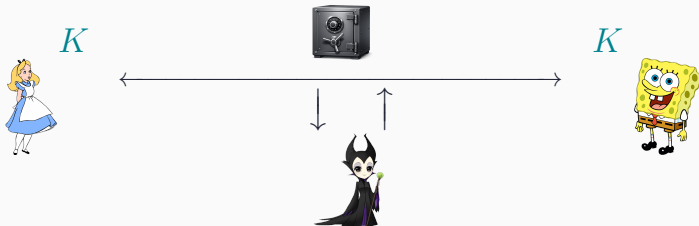
---

# Symmetric-Key: Authentication and Encryption



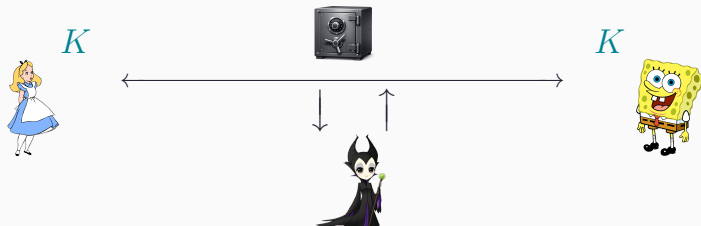
- Two parties, **Alice** and **Bob**, want to communicate over a public channel
  - They have agreed on a secret key  $K$ , used to protect their messages

# Symmetric-Key: Authentication and Encryption



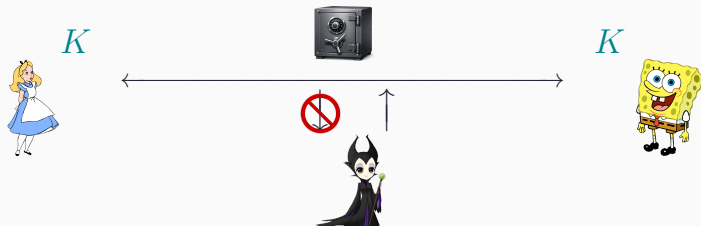
- Two parties, **Alice** and **Bob**, want to communicate over a public channel
  - They have agreed on a secret key  $K$ , used to protect their messages
- **Eve** may try to eavesdrop, tamper with, or disrupt this communication

# Symmetric-Key: Authentication and Encryption



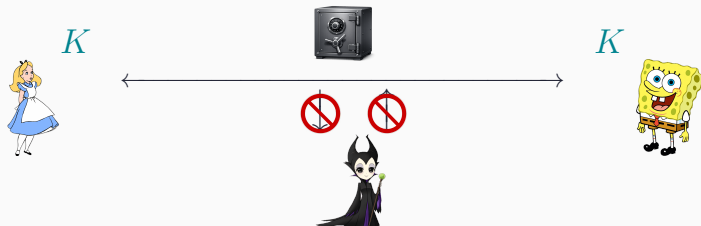
- Two parties, **Alice** and **Bob**, want to communicate over a public channel
  - They have agreed on a secret key  $K$ , used to protect their messages
- **Eve** may try to eavesdrop, tamper with, or disrupt this communication
- This leads to two core security goals:

# Symmetric-Key: Authentication and Encryption



- Two parties, **Alice** and **Bob**, want to communicate over a public channel
  - They have agreed on a secret key  $K$ , used to protect their messages
- **Eve** may try to eavesdrop, tamper with, or disrupt this communication
- This leads to two core security goals:
  - **Confidentiality**: Eve learns nothing about the message contents

# Symmetric-Key: Authentication and Encryption



- Two parties, **Alice** and **Bob**, want to communicate over a public channel
  - They have agreed on a secret key  $K$ , used to protect their messages
- **Eve** may try to eavesdrop, tamper with, or disrupt this communication
- This leads to two core security goals:
  - **Confidentiality**: Eve learns nothing about the message contents
  - **Authenticity**: Manipulation/replay/delay of messages is detected

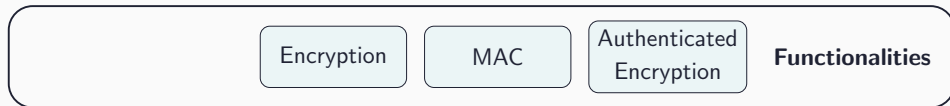
## Symmetric-Key Algorithms (Non-Exhaustive, With Some Examples)

Encryption

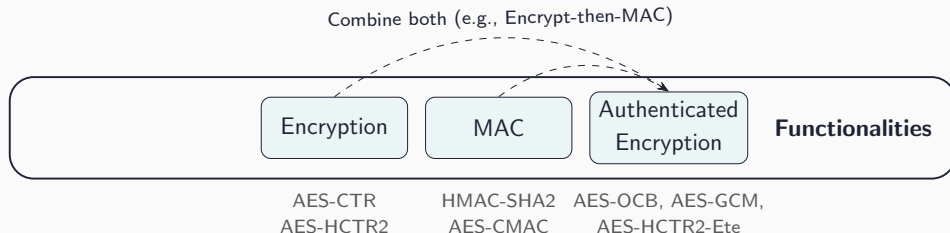
MAC

**Functionalities**

## Symmetric-Key Algorithms (Non-Exhaustive, With Some Examples)



# Symmetric-Key Algorithms (Non-Exhaustive, With Some Examples)



## Symmetric-Key Algorithms (Non-Exhaustive, With Some Examples)

Hash Function/PRF

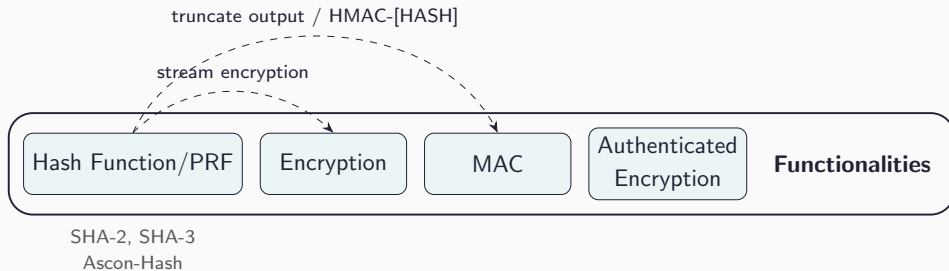
Encryption

MAC

Authenticated  
Encryption

**Functionalities**

# Symmetric-Key Algorithms (Non-Exhaustive, With Some Examples)



## Symmetric-Key Algorithms (Non-Exhaustive, With Some Examples)

Hash Function/PRF

Encryption

MAC

Authenticated  
Encryption

**Functionalities**

# Symmetric-Key Algorithms (Non-Exhaustive, With Some Examples)

Hash Function/PRF

Encryption

MAC

Authenticated  
Encryption

**Functionalities**

**Building Blocks**

# Symmetric-Key Algorithms (Non-Exhaustive, With Some Examples)

Hash Function/PRF

Encryption

MAC

Authenticated  
Encryption

**Functionalities**

Permutation

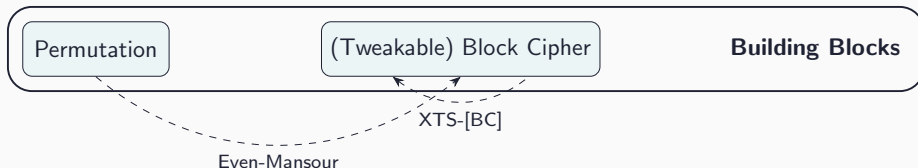
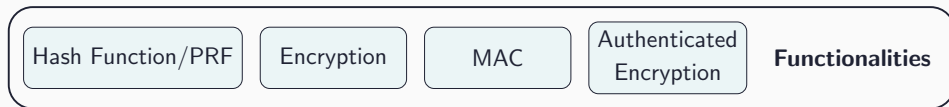
(Tweakable) Block Cipher

**Building Blocks**

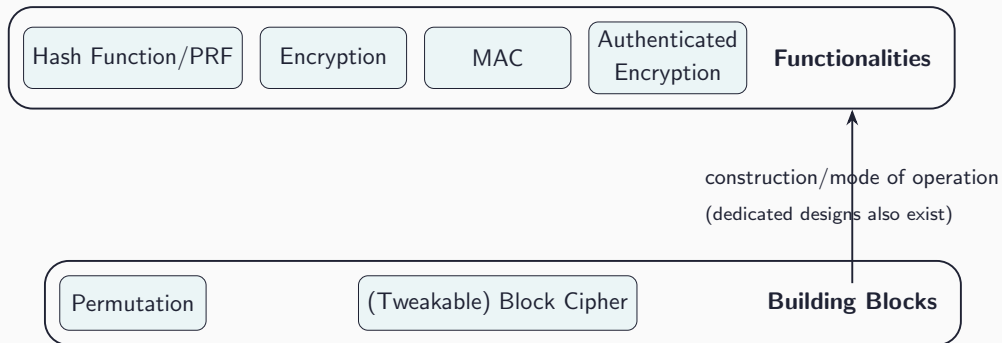
Keccak-f, Ascon-p

AES, DES

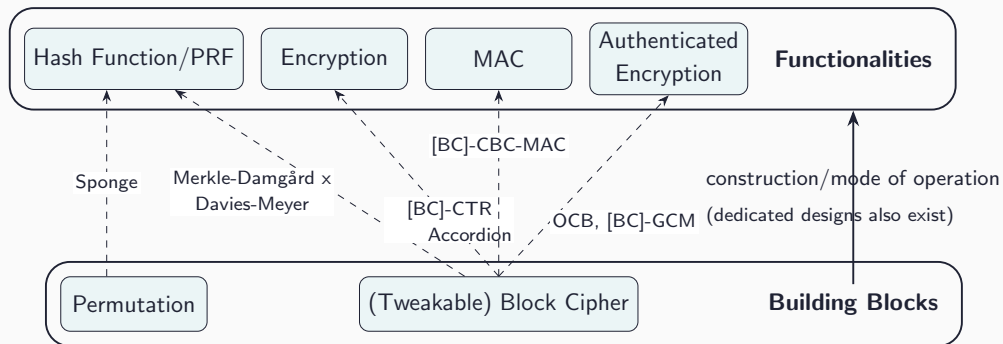
# Symmetric-Key Algorithms (Non-Exhaustive, With Some Examples)



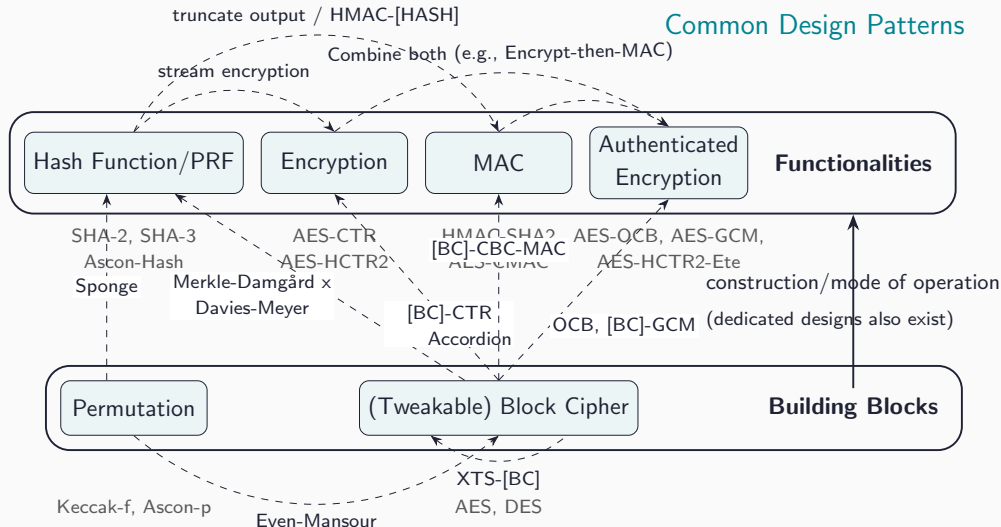
# Symmetric-Key Algorithms (Non-Exhaustive, With Some Examples)



# Symmetric-Key Algorithms (Non-Exhaustive, With Some Examples)



# Symmetric-Key Algorithms (Non-Exhaustive, With Some Examples)



## Security Arguments (1/3)

- To trust a cryptographic scheme, we need confidence it cannot be broken; ideally we would like a proof of that

## Security Arguments (1/3)

- To trust a cryptographic scheme, we need confidence it cannot be broken; ideally we would like a proof of that
- Unfortunately we **cannot** prove the security of most cryptosystems directly (No full security proof for SHA-2, SHA-3, AES-CTR, ...)

## Security Arguments (1/3)

- To trust a cryptographic scheme, we need confidence it cannot be broken; ideally we would like a proof of that
- Unfortunately we **cannot** prove the security of most cryptosystems directly (No full security proof for SHA-2, SHA-3, AES-CTR, ...)
- In practice, we rely on two complementary approaches:
  - (1) **Cryptanalysis:** extensively attack the building block, the algorithm, and/or the construction
  - (2) **Provable Security:** prove security under some assumptions on the building block/adversary

### Cryptanalysis:

- **Generic attacks:**
  - Attack the mode/construction assuming the primitive is ideal
  - Other side of the coin of provable security: tightness of bound

### Cryptanalysis:

- **Generic attacks:**
  - Attack the mode/construction assuming the primitive is ideal
  - Other side of the coin of provable security: tightness of bound
- **Attacks on the primitive/entire algorithm:**
  - Confidence = years of public extensive cryptanalysis (e.g., AES)
  - Common toolbox: differential, linear, integral, algebraic...
  - Security margin: #rounds attacked vs. #rounds of the cipher

### Cryptanalysis:

- **Generic attacks:**
  - Attack the mode/construction assuming the primitive is ideal
  - Other side of the coin of provable security: tightness of bound
- **Attacks on the primitive/entire algorithm:**
  - Confidence = years of public extensive cryptanalysis (e.g., AES)
  - Common toolbox: differential, linear, integral, algebraic...
  - Security margin: #rounds attacked vs. #rounds of the cipher
  - No time today, see [this very nice course](#)

### Provable Security:

- Roughly, the assumption is that the primitive is strong enough

### Provable Security:

- Roughly, the assumption is that the primitive is strong enough
- Two flavors:
  - **Standard model:** “If  $B$  is secure, then  $\text{mode}[B]$  is secure”, or

$$\mathbf{Adv}_{\text{mode}[B]}^{\text{sec-model}}(\mathcal{A}) \leq \text{some terms} + \mathbf{Adv}_B^{\text{other-sec-model}}(\mathcal{A}')$$

### Provable Security:

- Roughly, the assumption is that the primitive is strong enough
- Two flavors:
  - **Standard model:** “If  $B$  is secure, then  $\text{mode}[B]$  is secure”, or

$$\mathbf{Adv}_{\text{mode}[B]}^{\text{sec-model}}(\mathcal{A}) \leq \text{some terms} + \mathbf{Adv}_B^{\text{other-sec-model}}(\mathcal{A}')$$

- **Ideal model:** replaces  $B$  with an ideal object (e.g. random oracle):

$$\mathbf{Adv}_{\text{mode}[\text{ideal-}B]}^{\text{sec-model}}(\mathcal{A}) \leq \text{some terms (do not depend on } B)$$

## Provable Security:

- Roughly, the assumption is that the primitive is strong enough
- Two flavors:
  - **Standard model:** “If  $B$  is secure, then  $\text{mode}[B]$  is secure”, or

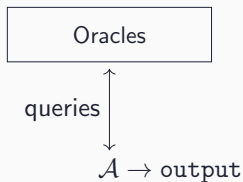
$$\mathbf{Adv}_{\text{mode}[B]}^{\text{sec-model}}(\mathcal{A}) \leq \text{some terms} + \mathbf{Adv}_B^{\text{other-sec-model}}(\mathcal{A}')$$

- **Ideal model:** replaces  $B$  with an ideal object (e.g. random oracle):

$$\mathbf{Adv}_{\text{mode}[\text{ideal-}B]}^{\text{sec-model}}(\mathcal{A}) \leq \text{some terms (do not depend on } B)$$

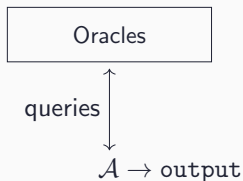
- A security proof holds only with adversaries captured by the security model! (e.g., key-commitment)

## Security Models (In Our Use-Case)



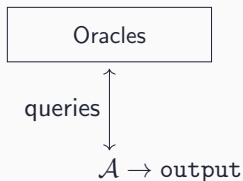
- An adversary  $\mathcal{A}$  is an algorithm interacting with a collection of oracles
- Oracles model what  $\mathcal{A}$  can do in practice: encrypt, decrypt, query the primitive (ideal model), ...

## Security Models (In Our Use-Case)



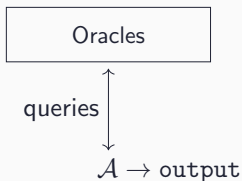
- An adversary  $\mathcal{A}$  is an algorithm interacting with a collection of oracles
- Oracles model what  $\mathcal{A}$  can do in practice: encrypt, decrypt, query the primitive (ideal model), ...
- $\mathcal{A}$  makes queries, then outputs something. It **wins** if its output satisfies some winning condition (manage to decrypt, finding a preimage, ...)

## Security Models (In Our Use-Case)



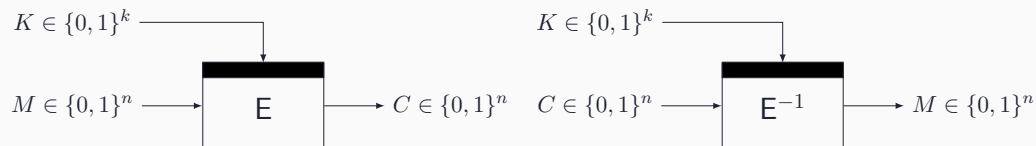
- An adversary  $\mathcal{A}$  is an algorithm interacting with a collection of oracles
- Oracles model what  $\mathcal{A}$  can do in practice: encrypt, decrypt, query the primitive (ideal model), ...
- $\mathcal{A}$  makes queries, then outputs something. It **wins** if its output satisfies some winning condition (manage to decrypt, finding a preimage, ...)
- We call the (normalized) probability that it wins the **advantage**

## Security Models (In Our Use-Case)



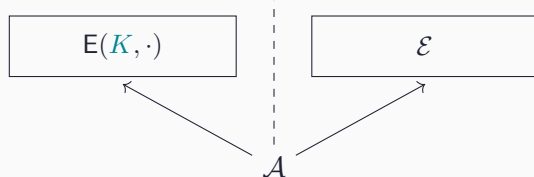
- An adversary  $\mathcal{A}$  is an algorithm interacting with a collection of oracles
- Oracles model what  $\mathcal{A}$  can do in practice: encrypt, decrypt, query the primitive (ideal model), ...
- $\mathcal{A}$  makes queries, then outputs something. It **wins** if its output satisfies some winning condition (manage to decrypt, finding a preimage, ...)
- We call the (normalized) probability that it wins the **advantage**
- Security := the advantage is negligible  $\forall \mathcal{A}$  with **reasonable** resources (running time, number of queries, ...)

## Example: Block Cipher Security (Keyed Setting)



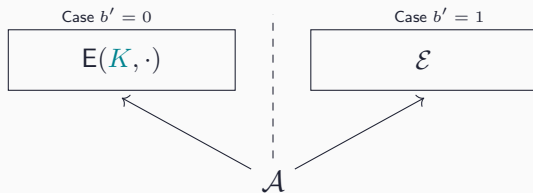
What would we informally expect from a “good” block cipher, used with a secret key?

## (S)PRP Security (1/4)



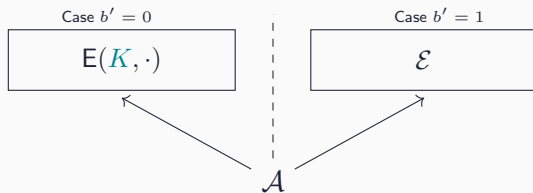
- PRP security implies protection against many concrete attacks at once
- $E(K, \cdot)$ , for some random, secret key  $K$ , should be indistinguishable from a random permutation  $\mathcal{E}$

## (S)PRP Security (1/4)



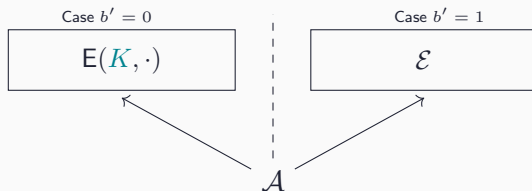
- PRP security implies protection against many concrete attacks at once
- $E(K, \cdot)$ , for some random, secret key  $K$ , should be indistinguishable from a random permutation  $\mathcal{E}$
- At the beginning of security game;  $b' \xleftarrow{\$} \{0, 1\}$ , if  $b' = 0$  implement leftmost oracle, else rightmost

## (S)PRP Security (1/4)



- PRP security implies protection against many concrete attacks at once
- $E(K, \cdot)$ , for some random, secret key  $K$ , should be indistinguishable from a random permutation  $\mathcal{E}$
- At the beginning of security game;  $b' \xleftarrow{\$} \{0, 1\}$ , if  $b' = 0$  implement leftmost oracle, else rightmost
- The adversary  $\mathcal{A}$  outputs a bit  $b$  reflecting its guess with which world it lives in

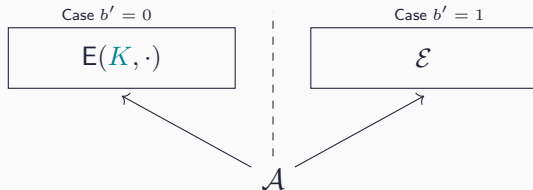
## (S)PRP Security (2/4)



- $\mathcal{A}$  wins if its guess was correct, so we want to define

$$\mathbf{Adv}_E^{\text{PRP}}(\mathcal{A}) = \mathbf{Pr}(b = b')$$

## (S)PRP Security (2/4)

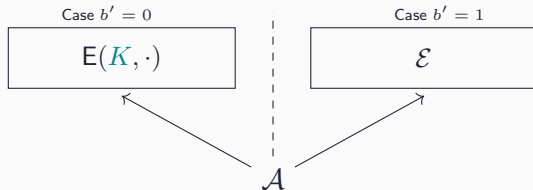


- $\mathcal{A}$  wins if its guess was correct, so we want to define

$$\text{Adv}_{\text{E}}^{\text{PRP}}(\mathcal{A}) = \Pr(b = b')$$

- But a dummy adversary wins with probability  $1/2$

## (S)PRP Security (2/4)



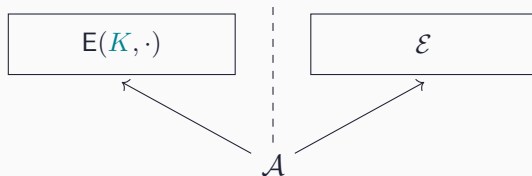
- $\mathcal{A}$  wins if its guess was correct, so we want to define

$$\mathbf{Adv}_E^{\text{PRP}}(\mathcal{A}) = \mathbf{Pr}(b = b')$$

- But a dummy adversary wins with probability  $1/2$
- Workaround: normalize the advantage:

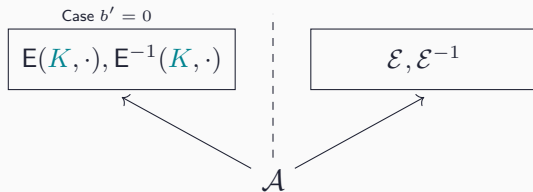
$$\begin{aligned}\mathbf{Adv}_E^{\text{PRP}}(\mathcal{A}) &= |2\mathbf{Pr}(b = b') - 1| \\ &= \left| \mathbf{Pr}\left(\mathcal{A}^{E(K, \cdot)} \rightarrow 1 : K \xleftarrow{\$} \{0, 1\}^k\right) - \mathbf{Pr}\left(\mathcal{A}^{\mathcal{E}} \rightarrow 1 : \mathcal{E} \xleftarrow{\$} \text{Perms}(\{0, 1\}^n)\right) \right|\end{aligned}$$

## (S)PRP Security (3/4)



$$\mathbf{Adv}_E^{\text{PRP}}(\mathcal{A}) = \left| \Pr \left( \mathcal{A}^{E(K, \cdot)} \rightarrow 1 : K \xleftarrow{\$} \{0, 1\}^k \right) - \Pr \left( \mathcal{A}^{\mathcal{E}} \rightarrow 1 : \mathcal{E} \xleftarrow{\$} \text{Perms}(\{0, 1\}^n) \right) \right|$$

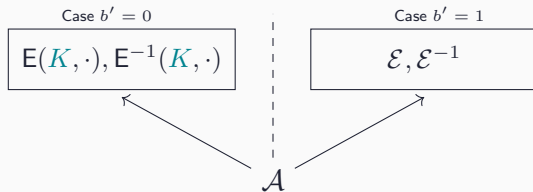
## (S)PRP Security (3/4)



$$\text{Adv}_{\text{E}}^{\text{SPRP}}(\mathcal{A}) = \left| \Pr \left( \mathcal{A}^{E(K, \cdot), E^{-1}(K, \cdot)} \rightarrow 1 : K \xleftarrow{\$} \{0, 1\}^k \right) - \Pr \left( \mathcal{A}^{\mathcal{E}, \mathcal{E}^{-1}} \rightarrow 1 : \mathcal{E} \xleftarrow{\$} \text{Perms}(\{0, 1\}^n) \right) \right|$$

- SPRP security model: also give inverse access to  $E^{-1}(K, \cdot)$

## (S)PRP Security (3/4)



$$\mathbf{Adv}_E^{\text{SPRP}}(\mathcal{A}) = \left| \Pr \left( \mathcal{A}^{E(K, \cdot), E^{-1}(K, \cdot)} \rightarrow 1 : K \xleftarrow{\$} \{0, 1\}^k \right) - \Pr \left( \mathcal{A}^{\mathcal{E}, \mathcal{E}^{-1}} \rightarrow 1 : \mathcal{E} \xleftarrow{\$} \text{Perms}(\{0, 1\}^n) \right) \right|$$

- SPRP security model: also give inverse access to  $E^{-1}(K, \cdot)$
- $\mathcal{A}$ 's resources: number of oracle queries  $D$  (*online complexity*) and running time  $T$  (*offline complexity*)

- Ideally: advantage close to 0 for any adversary with “reasonable”  $T$  and  $D$ . The adversary can always recover the key, so the best we could hope for is

$$\mathbf{Adv}^{\text{ind}}(\mathcal{A}) \underset{\approx}{\leq} \frac{T}{2^k} \quad (*)$$

$\implies$  need a sufficiently large key size, to rule out exhaustive search

## (S)PRP Security (4/4)

- Ideally: advantage close to 0 for any adversary with “reasonable”  $T$  and  $D$ . The adversary can always recover the key, so the best we could hope for is

$$\mathbf{Adv}^{\text{ind}}(\mathcal{A}) \underset{\approx}{\leq} \frac{T}{2^k} \quad (*)$$

$\implies$  need a sufficiently large key size, to rule out exhaustive search

- (\*) **believed** to hold for AES (no proof of that!)

# Indistinguishability and Proofs

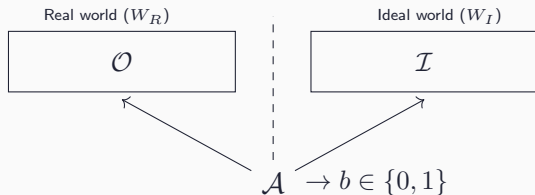
---

# Indistinguishability

The principle of comparing an algorithm to its idealized version is handy. The underlying type of security game is named **indistinguishability game**

# Indistinguishability

The principle of comparing an algorithm to its idealized version is handy. The underlying type of security game is named **indistinguishability game**

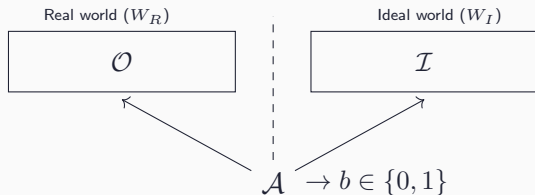


$b' = 0$  if real world, else  $b' = 1$

$$\mathbf{Adv}_{W_R; W_I}^{\text{ind}}(\mathcal{A}) = \left| \mathbf{Pr} \left( \mathcal{A}^{\mathcal{O}} \rightarrow 1 : \mathcal{O} \stackrel{\$}{\leftarrow} \mathbb{O} \right) - \mathbf{Pr} \left( \mathcal{A}^{\mathcal{I}} \rightarrow 1 : \mathcal{I} \stackrel{\$}{\leftarrow} \mathbb{I} \right) \right|$$

# Indistinguishability

The principle of comparing an algorithm to its idealized version is handy. The underlying type of security game is named **indistinguishability game**



$b' = 0$  if real world, else  $b' = 1$

$$\mathbf{Adv}_{W_R; W_I}^{\text{ind}}(\mathcal{A}) = \left| \mathbf{Pr} \left( \mathcal{A}^{\mathcal{O}} \rightarrow 1 : \mathcal{O} \xleftarrow{\$} \mathbb{O} \right) - \mathbf{Pr} \left( \mathcal{A}^{\mathcal{I}} \rightarrow 1 : \mathcal{I} \xleftarrow{\$} \mathbb{I} \right) \right|$$

Key features: triangle inequality and compositions

In security proofs, we often end up with **information-theoretic** adversaries:

- Unlimited computational power
- The only resource we measure is their number of **oracle queries**
- Not realistic, but this is what remains after a **security reduction** (replace  $E_K$  by a random permutation) or in an **idealized model**

# Information-Theoretic Adversaries

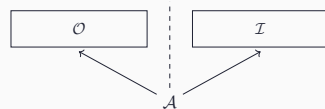
In security proofs, we often end up with **information-theoretic** adversaries:

- Unlimited computational power
- The only resource we measure is their number of **oracle queries**
- Not realistic, but this is what remains after a **security reduction** (replace  $E_K$  by a random permutation) or in an **idealized model**

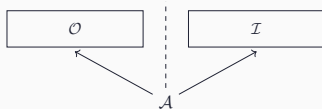
**But not for the (S)PRP security of a concrete block cipher:** with a few queries, an unbounded adversary tries all  $2^k$  keys and recovers  $K$

$\implies$  we must also bound its running time

# Indistinguishability and Statistical Distance



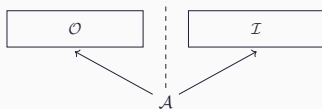
- The whole interaction between the adversary and the oracles can be summarized into a **transcript**  $\tau$ ; an ordered list containing all (input, outputs)
- The adversaries can be assumed to be deterministic without loss of generality



- The whole interaction between the adversary and the oracles can be summarized into a **transcript**  $\tau$ ; an ordered list containing all (input, outputs)
- The adversaries can be assumed to be deterministic without loss of generality

**Statistical Distance Formulation.** Omitting for readability the probability space:

$$\Delta_{\mathcal{A}}(W_R; W_I) := \frac{1}{2} \sum_{\tau} |\mathbf{Pr}(\mathcal{A}^{\mathcal{O}} \text{ produces } \tau) - \mathbf{Pr}(\mathcal{A}^{\mathcal{I}} \text{ produces } \tau)| ,$$



- The whole interaction between the adversary and the oracles can be summarized into a **transcript**  $\tau$ ; an ordered list containing all (input, outputs)
- The adversaries can be assumed to be deterministic without loss of generality

**Statistical Distance Formulation.** Omitting for readability the probability space:

$$\Delta_{\mathcal{A}}(W_R; W_I) := \frac{1}{2} \sum_{\tau} |\Pr(\mathcal{A}^{\mathcal{O}} \text{ produces } \tau) - \Pr(\mathcal{A}^{\mathcal{I}} \text{ produces } \tau)|,$$

For a set  $\mathbb{D}$  of information-theoretic distinguishers with a fixed query budget, we have

$$\max_{\mathcal{A} \in \mathbb{D}} \mathbf{Adv}_{W_R; W_I}^{\text{ind}}(\mathcal{A}) = \max_{\mathcal{A} \in \mathbb{D}} \Delta_{\mathcal{A}}(W_R; W_I)$$

$\implies$  equivalent when maximizing over all adversaries with a given query budget

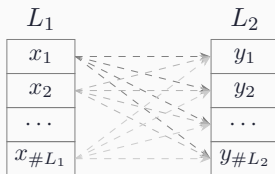
# The Birthday Bound

- Shows up everywhere
- Suppose we have two lists  $L_1$ ,  $L_2$  of  $n$ -bit random elements, and want to upper bound probability to get a collision:

| $L_1$       | $L_2$       |
|-------------|-------------|
| $x_1$       | $y_1$       |
| $x_2$       | $y_2$       |
| $\dots$     | $\dots$     |
| $x_{\#L_1}$ | $y_{\#L_2}$ |

# The Birthday Bound

- Shows up everywhere
- Suppose we have two lists  $L_1$ ,  $L_2$  of  $n$ -bit random elements, and want to upper bound probability to get a collision:

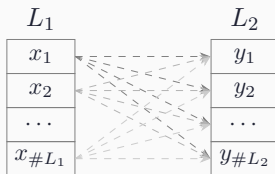


- An upper bound to get a collision is given by a union bound:

$$\Pr(\text{col}) \leq \sum_{i_1, i_2} \Pr(L_1[i_1] = L_2[i_2]) = \frac{\#L_1 \#L_2}{2^n}$$

# The Birthday Bound

- Shows up everywhere
- Suppose we have two lists  $L_1$ ,  $L_2$  of  $n$ -bit random elements, and want to upper bound probability to get a collision:



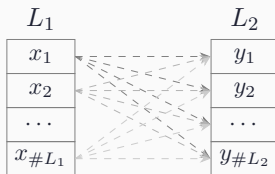
- An upper bound to get a collision is given by a union bound:

$$\Pr(\text{col}) \leq \sum_{i_1, i_2} \Pr(L_1[i_1] = L_2[i_2]) = \frac{\#L_1 \#L_2}{2^n}$$

- What about collisions within a single list  $L$ ?

# The Birthday Bound

- Shows up everywhere
- Suppose we have two lists  $L_1$ ,  $L_2$  of  $n$ -bit random elements, and want to upper bound probability to get a collision:



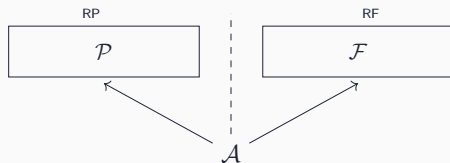
- An upper bound to get a collision is given by a union bound:

$$\Pr(\text{col}) \leq \sum_{i_1, i_2} \Pr(L_1[i_1] = L_2[i_2]) = \frac{\#L_1 \#L_2}{2^n}$$

- What about collisions within a single list  $L$ ?  $\Pr(\text{col}) \leq \frac{\binom{\#L}{2}}{2^n} = \frac{\#L(\#L-1)}{2^{n+1}}$

## Example: PRP/PRF Switching Lemma (1/3)

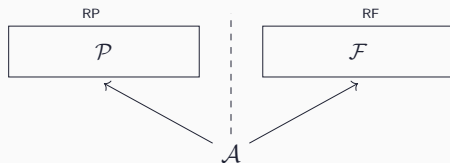
Happens frequently as an intermediate step in security proofs



- $\mathcal{P} : \{0, 1\}^n \rightarrow \{0, 1\}^n$  is a **random permutation** (forward queries only)
- $\mathcal{F} : \{0, 1\}^n \rightarrow \{0, 1\}^n$  is a **random function**

## Example: PRP/PRF Switching Lemma (1/3)

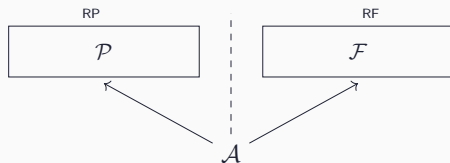
Happens frequently as an intermediate step in security proofs



- $\mathcal{P} : \{0, 1\}^n \rightarrow \{0, 1\}^n$  is a **random permutation** (forward queries only)
- $\mathcal{F} : \{0, 1\}^n \rightarrow \{0, 1\}^n$  is a **random function**
- The adversary makes at most  $q$  queries that we assume to be fresh **without loss of generality**

## Example: PRP/PRF Switching Lemma (1/3)

Happens frequently as an intermediate step in security proofs



- $\mathcal{P} : \{0, 1\}^n \rightarrow \{0, 1\}^n$  is a **random permutation** (forward queries only)
- $\mathcal{F} : \{0, 1\}^n \rightarrow \{0, 1\}^n$  is a **random function**
- The adversary makes at most  $q$  queries that we assume to be fresh **without loss of generality**
- Proof using the **fundamental lemma of game-playing** [BR06]

## Example: PRP/PRF Switching Lemma (2/3)

RP :

$$P \xleftarrow{\$} \text{Perm}(n)$$

Input  $x$  : return  $P(x)$

RF :

$$F \xleftarrow{\$} \text{Fun}(n, n)$$

Input  $x$  : return  $F(x)$

## Example: PRP/PRF Switching Lemma (2/3)

RP :

$P \xleftarrow{\$} \text{Perm}(n)$

Input  $x$  : return  $P(x)$



LRP : on fresh input  $x$ ,

$y \xleftarrow{\$} \{0, 1\}^n \setminus T$

$T \leftarrow T \cup \{y\}$

return  $y$

RF :

$F \xleftarrow{\$} \text{Fun}(n, n)$

Input  $x$  : return  $F(x)$



LRF : on fresh input  $x$ ,

$y \xleftarrow{\$} \{0, 1\}^n$

$T \leftarrow T \cup \{y\}$

return  $y$

- Lazy sampling : instead of sampling  $P$  (or  $F$ ) once and for all at the beginning, sample each output on-the-fly ( $T$  initialized to empty)

## Example: PRP/PRF Switching Lemma (2/3)

RP :

$P \xleftarrow{\$} \text{Perm}(n)$

Input  $x$  : return  $P(x)$



LRP : on fresh input  $x$ ,

$y \xleftarrow{\$} \{0, 1\}^n \setminus T$

$T \leftarrow T \cup \{y\}$

return  $y$

RF :

$F \xleftarrow{\$} \text{Fun}(n, n)$

Input  $x$  : return  $F(x)$



LRF : on fresh input  $x$ ,

$y \xleftarrow{\$} \{0, 1\}^n$

$T \leftarrow T \cup \{y\}$

return  $y$

- Lazy sampling : instead of sampling  $P$  (or  $F$ ) once and for all at the beginning, sample each output on-the-fly ( $T$  initialized to empty)
- Does not change the distribution, so  $\mathbf{Adv}_{\text{RP}; \text{RF}}^{\text{ind}}(\mathcal{A}) = \mathbf{Adv}_{\text{LRP}; \text{LRF}}^{\text{ind}}(\mathcal{A})$

## Example: PRP/PRF Switching Lemma (3/3)

On a fresh input  $x$ ,

LRP:

```
 $y \xleftarrow{\$} \{0, 1\}^n \setminus T$   
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

LRF:

```
 $y \xleftarrow{\$} \{0, 1\}^n$   
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

- $T$  is initialized as the empty set

## Example: PRP/PRF Switching Lemma (3/3)

On a fresh input  $x$ ,

LRP:

```
 $y \xleftarrow{\$} \{0, 1\}^n \setminus T$   
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

LRF':

```
 $y \xleftarrow{\$} \{0, 1\}^n$   
If  $y \in T$ :  
— set bad  
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

LRF:

```
 $y \xleftarrow{\$} \{0, 1\}^n$   
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

- $T$  is initialized as the empty set
- Rewrite LRF into LRF'

## Example: PRP/PRF Switching Lemma (3/3)

On a fresh input  $x$ ,

LRP:

```
 $y \xleftarrow{\$} \{0, 1\}^n \setminus T$   
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

LRP':

```
 $y \xleftarrow{\$} \{0, 1\}^n$   
If  $y \in T$   
— set bad  
—  $y \xleftarrow{\$} \{0, 1\}^n \setminus T$   
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

LRF':

```
 $y \xleftarrow{\$} \{0, 1\}^n$   
If  $y \in T$ :  
— set bad  
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

LRF:

```
 $y \xleftarrow{\$} \{0, 1\}^n$   
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

- $T$  is initialized as the empty set
- Rewrite LRF into LRF'; LRP into LRP':  $\mathbf{Adv}^{\text{LRP}; \text{LRF}}(\mathcal{A}) = \mathbf{Adv}^{\text{LRP}'; \text{LRF}'}(\mathcal{A})$

## Example: PRP/PRF Switching Lemma (3/3)

On a fresh input  $x$ ,

LRP:

```
 $y \xleftarrow{\$} \{0, 1\}^n \setminus T$   
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

LRP':

```
 $y \xleftarrow{\$} \{0, 1\}^n$   
If  $y \in T$   
— set bad  
—  $y \xleftarrow{\$} \{0, 1\}^n \setminus T$   
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

LRF':

```
 $y \xleftarrow{\$} \{0, 1\}^n$   
If  $y \in T$ :  
— set bad  
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

LRF:

```
 $y \xleftarrow{\$} \{0, 1\}^n$   
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

- $T$  is initialized as the empty set
- Rewrite LRF into LRF'; LRP into LRP':  $\mathbf{Adv}^{\text{LRP}; \text{LRF}}(\mathcal{A}) = \mathbf{Adv}^{\text{LRP}'; \text{LRF}'}(\mathcal{A})$
- LRP' and LRF' are identical as long as **bad** is not set

## Example: PRP/PRF Switching Lemma (3/3)

On a fresh input  $x$ ,

LRP:

```
 $y \xleftarrow{\$} \{0, 1\}^n \setminus T$   
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

LRP':

```
 $y \xleftarrow{\$} \{0, 1\}^n$   
If  $y \in T$   
— set bad  
—  $y \xleftarrow{\$} \{0, 1\}^n \setminus T$   
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

LRF':

```
 $y \xleftarrow{\$} \{0, 1\}^n$   
If  $y \in T$ :  
— set bad  
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

LRF:

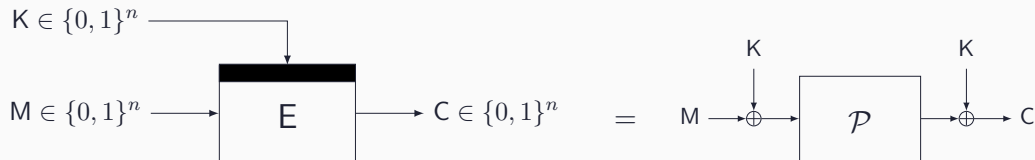
```
 $y \xleftarrow{\$} \{0, 1\}^n$   
 $T \leftarrow T \cup \{y\}$   
return  $y$ 
```

- $T$  is initialized as the empty set
- Rewrite LRF into LRF'; LRP into LRP':  $\mathbf{Adv}^{\text{LRP}; \text{LRF}}(\mathcal{A}) = \mathbf{Adv}^{\text{LRP}'; \text{LRF}'}(\mathcal{A})$
- LRP' and LRF' are identical as long as **bad** is not set
- The fundamental lemma of game playing tells us:

$$\mathbf{Adv}_{\text{LRP}'; \text{LRF}'}^{\text{ind}}(\mathcal{A}) \leq \Pr\left(\mathcal{A}^{\text{LRF}'} \text{ sets BAD}\right) \leq \frac{q(q-1)}{2^{n+1}}$$

## The Even-Mansour Construction [EM91, EM97, DKS12] (1/4)

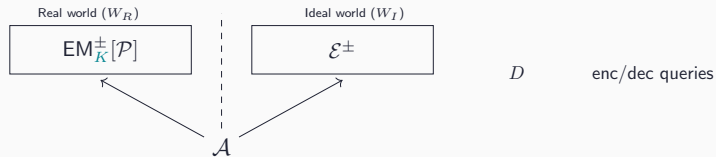
The Even-Mansour (EM) construction is a generic way to build a block cipher from a public permutation:



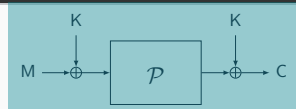
- Encryption: XOR the plaintext with  $K$ , apply  $\mathcal{P}$ , XOR the result with  $K$  again
- Decryption: same, using  $\mathcal{P}^{-1}$  instead

# The Even-Mansour Construction (2/4)

SPRP security:

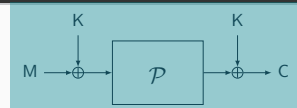
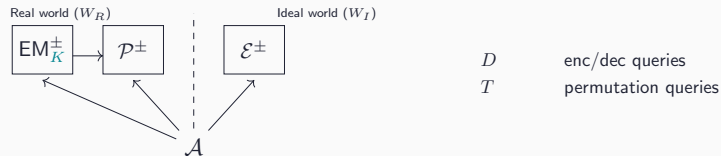


- $\mathcal{P}$  is a **public** permutation



# The Even-Mansour Construction (2/4)

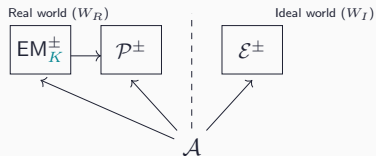
SPRP security:



- $\mathcal{P}$  is a **public** permutation
- Proof treats  $\mathcal{P}$  as a **uniformly random** permutation  
 $\implies$  Make  $\mathcal{P}$  part of the oracles

# The Even-Mansour Construction (2/4)

SPRP security:

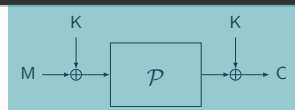


$D$

enc/dec queries

$T$

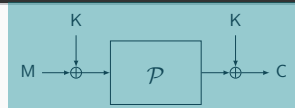
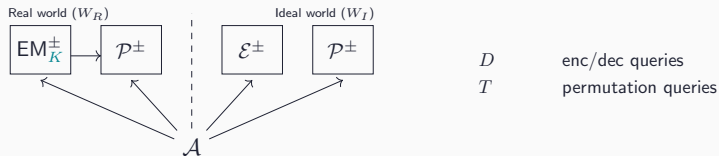
permutation queries



- $\mathcal{P}$  is a **public** permutation
- Proof treats  $\mathcal{P}$  as a **uniformly random** permutation  
 $\implies$  Make  $\mathcal{P}$  part of the oracles
- This also requires an ideal-world counterpart to compare against

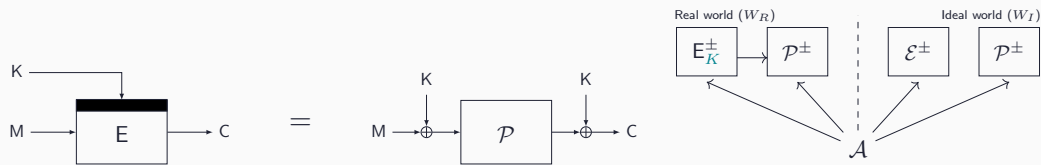
# The Even-Mansour Construction (2/4)

SPRP security:



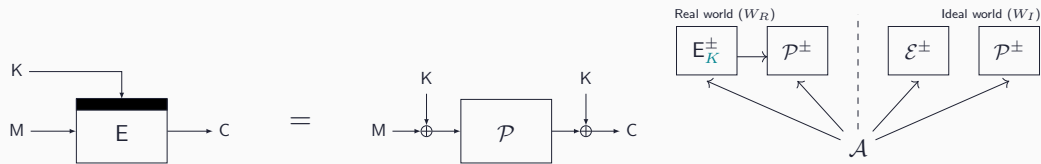
- $\mathcal{P}$  is a **public** permutation
- Proof treats  $\mathcal{P}$  as a **uniformly random** permutation  
 $\implies$  Make  $\mathcal{P}$  part of the oracles
- This also requires an ideal-world counterpart to compare against  $\implies$  give access to a random permutation, **independent** of the block cipher oracle

## The Even-Mansour Construction (3/4)



- Intuitively, things go wrong when an EM query and a permutation query **collide** on the same underlying  $\mathcal{P}$ -call
- Example: an EM query on  $M$  internally computes  $C := \mathcal{P}(M \oplus K) \oplus K$ . If  $\mathcal{A}$  later queries  $O := \mathcal{P}(M \oplus K)$ , then they get  $C \oplus O = K$   
 $\implies$  can be turned into a key recovery attack

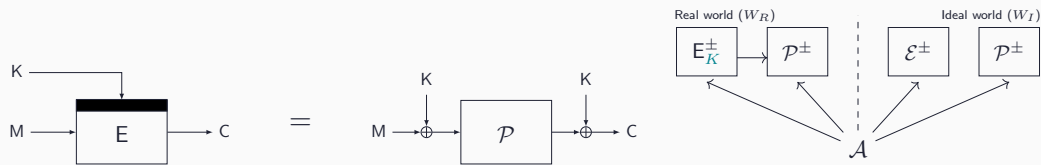
# The Even-Mansour Construction (4/4)



- As before, security reduces to a **bad event**.<sup>1</sup> Writing  $\text{Tab}_{\text{Perm}} = \{(X, Y)\}$  for the table of  $\mathcal{P}$ -queries so far, **bad** is set when
  - A query  $C \leftarrow E(M)$  has  $M \oplus K \in \text{dom}(\text{Tab}_{\text{Perm}})$  or  $C \oplus K \in \text{rng}(\text{Tab}_{\text{Perm}})$
  - Or same as above, with the roles of  $E$  and  $\text{Perm}$  swapped

<sup>1</sup>Shown for PRP security; SPRP is analogous, just more cases

# The Even-Mansour Construction (4/4)



- As before, security reduces to a **bad event**.<sup>1</sup> Writing  $\text{Tab}_{\text{Perm}} = \{(X, Y)\}$  for the table of  $\mathcal{P}$ -queries so far, **bad** is set when
  - A query  $C \leftarrow E(M)$  has  $M \oplus K \in \text{dom}(\text{Tab}_{\text{Perm}})$  or  $C \oplus K \in \text{rng}(\text{Tab}_{\text{Perm}})$
  - Or same as above, with the roles of  $E$  and  $\text{Perm}$  swapped

$$\implies \text{Adv}_{\text{EM}}^{\text{SPRP}}(\mathcal{A}) \leq \frac{2DT}{2^n}$$

<sup>1</sup>Shown for PRP security; SPRP is analogous, just more cases

## The H-Coefficient Technique [Pat08, CS14]

- Ubiquitous in the literature

# The H-Coefficient Technique [Pat08, CS14]

- Ubiquitous in the literature
- Summarize  $\mathcal{A}$ 's interaction into a transcript  $\tau$ , and let

$$p_{\text{re}}(\tau) = \mathbf{Pr}(\mathcal{A}^{\mathcal{O}} \text{ yields } \tau), \quad p_{\text{id}}(\tau) = \mathbf{Pr}(\mathcal{A}^{\mathcal{I}} \text{ yields } \tau)$$

# The H-Coefficient Technique [Pat08, CS14]

- Ubiquitous in the literature
- Summarize  $\mathcal{A}$ 's interaction into a transcript  $\tau$ , and let

$$p_{\text{re}}(\tau) = \mathbf{Pr}(\mathcal{A}^{\mathcal{O}} \text{ yields } \tau), \quad p_{\text{id}}(\tau) = \mathbf{Pr}(\mathcal{A}^{\mathcal{I}} \text{ yields } \tau)$$

- Split attainable  $\tau$  into  $\mathcal{T}_{\text{good}} \sqcup \mathcal{T}_{\text{bad}}$ . If

$$\frac{p_{\text{re}}(\tau)}{p_{\text{id}}(\tau)} \geq 1 - \varepsilon \quad \forall \tau \in \mathcal{T}_{\text{good}}, \quad \mathbf{Pr}(\mathcal{A}^{\mathcal{I}} \in \mathcal{T}_{\text{bad}}) \leq \delta \implies \mathbf{Adv}^{\text{ind}}(\mathcal{A}) \leq \varepsilon + \delta$$

# The H-Coefficient Technique [Pat08, CS14]

- Ubiquitous in the literature
- Summarize  $\mathcal{A}$ 's interaction into a transcript  $\tau$ , and let

$$p_{\text{re}}(\tau) = \mathbf{Pr}(\mathcal{A}^{\mathcal{O}} \text{ yields } \tau), \quad p_{\text{id}}(\tau) = \mathbf{Pr}(\mathcal{A}^{\mathcal{I}} \text{ yields } \tau)$$

- Split attainable  $\tau$  into  $\mathcal{T}_{\text{good}} \sqcup \mathcal{T}_{\text{bad}}$ . If

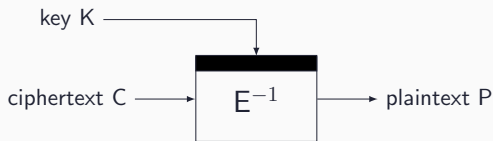
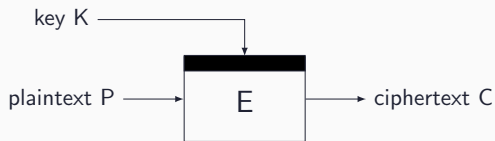
$$\frac{p_{\text{re}}(\tau)}{p_{\text{id}}(\tau)} \geq 1 - \varepsilon \quad \forall \tau \in \mathcal{T}_{\text{good}}, \quad \mathbf{Pr}(\mathcal{A}^{\mathcal{I}} \in \mathcal{T}_{\text{bad}}) \leq \delta \implies \mathbf{Adv}^{\text{ind}}(\mathcal{A}) \leq \varepsilon + \delta$$

- Identical-until-bad: special case when  $p_{\text{re}}(\tau) = p_{\text{id}}(\tau), \forall \tau \in \mathcal{T}_{\text{good}}$
- H-coefficient is more general, conceptually a bit different

# Authenticated Encryption

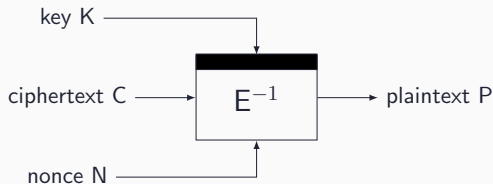
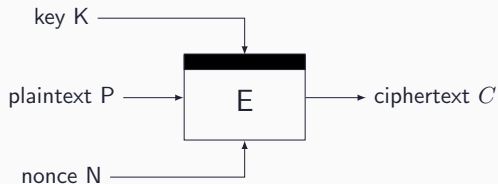
---

# Encryption



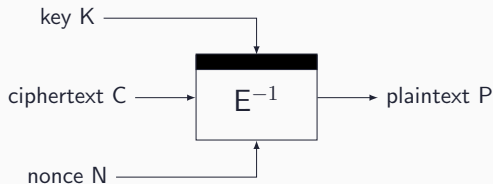
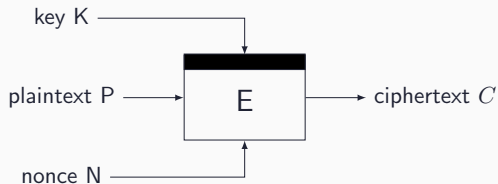
- Encryption aims to achieve **confidentiality**
- Key is typically of fixed size
- Plaintexts/ciphertexts typically arbitrarily long

# Encryption



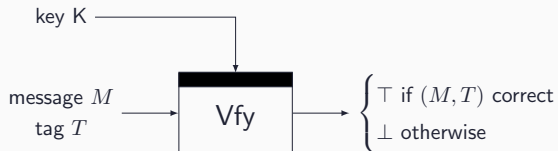
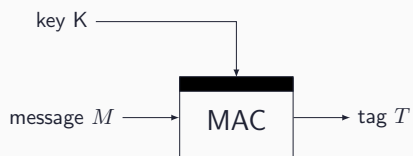
- Encryption aims to achieve **confidentiality**
- Key is typically of fixed size
- Plaintexts/ciphertexts typically arbitrarily long
- Depending on the use-case, a diversifier may be necessary

# Encryption



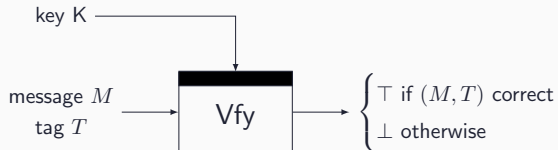
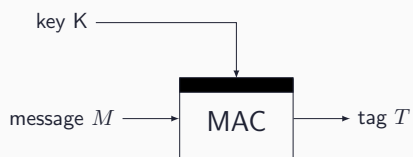
- Encryption aims to achieve confidentiality
- Key is typically of fixed size
- Plaintexts/ciphertexts typically arbitrarily long
- Depending on the use-case, a diversifier may be necessary
- Example: CTR mode

# Message Authentication Code (MAC)



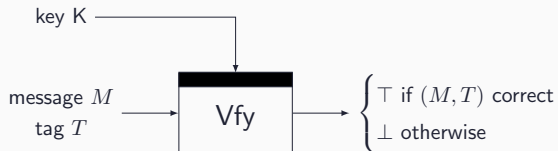
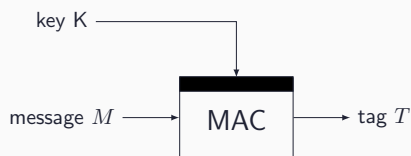
- Message authentication code aims to achieve **authenticity**
- Tag  $T$  authenticates  $M$
- Key and tag are typically of fixed size
- Message is typically arbitrarily long

# Message Authentication Code (MAC)



- Message authentication code aims to achieve **authenticity**
- Tag  $T$  authenticates  $M$
- Key and tag are typically of fixed size
- Message is typically arbitrarily long
- May also include a nonce

# Message Authentication Code (MAC)



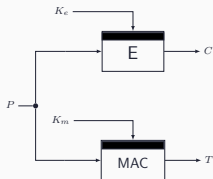
- Message authentication code aims to achieve **authenticity**
- Tag  $T$  authenticates  $M$
- Key and tag are typically of fixed size
- Message is typically arbitrarily long
- May also include a nonce
- Example: Wegman-Carter-Shoup

- In many cases we want both confidentiality and authenticity

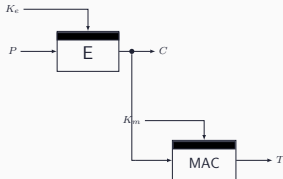
# Motivation

- In many cases we want both confidentiality and authenticity
- We can do it via generic composition [BN08]:

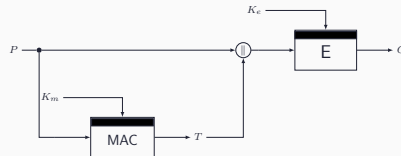
Encrypt-and-MAC



Encrypt-then-MAC



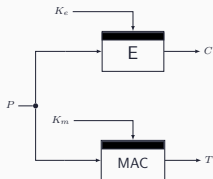
MAC-then-Encrypt



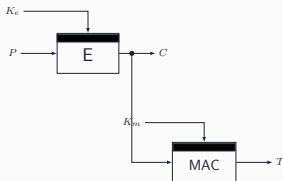
# Motivation

- In many cases we want both confidentiality and authenticity
- We can do it via generic composition [BN08]:

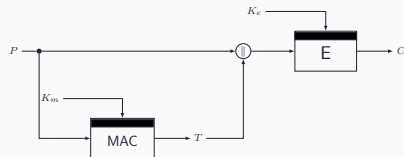
Encrypt-and-MAC



Encrypt-then-MAC



MAC-then-Encrypt

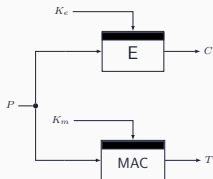


- Enc-and-MAC insecure, MAC-then-Encrypt not really, Enc-then-MAC fine

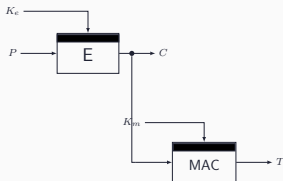
# Motivation

- In many cases we want both confidentiality and authenticity
- We can do it via generic composition [BN08]:

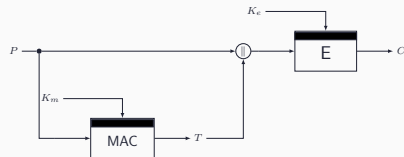
Encrypt-and-MAC



Encrypt-then-MAC

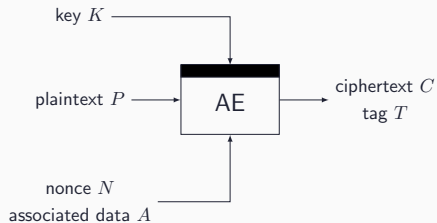


MAC-then-Encrypt



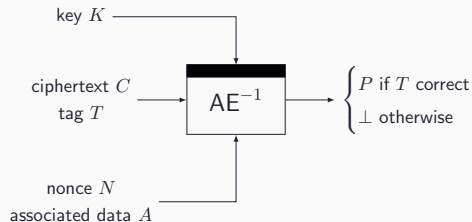
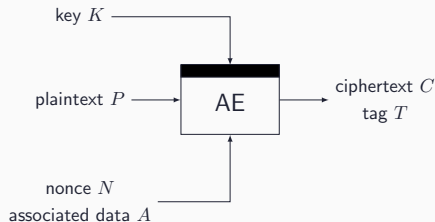
- Enc-and-MAC insecure, MAC-then-Encrypt not really, Enc-then-MAC fine
- But not necessarily the most efficient way, may be good to have a single primitive doing those two jobs

# Authenticated Encryption with Associated Data (AEAD)



- Using key  $K$ :
  - Ciphertext  $C$  encrypts plaintext  $P$
  - Tag  $T$  authenticates  $(N, A, P)$
- Nonce  $N$  must not be repeated

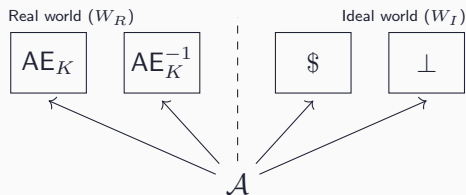
# Authenticated Encryption with Associated Data (AEAD)



- Using key  $K$ :
  - Ciphertext  $C$  encrypts plaintext  $P$
  - Tag  $T$  authenticates  $(N, A, P)$
- Nonce  $N$  must not be repeated

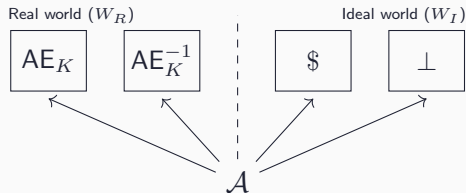
- Unwrapping needs to satisfy that
  - Plaintext disclosed if tag is **correct**
  - Plaintext is not leaked if tag is **incorrect**

# Standard AEAD Security



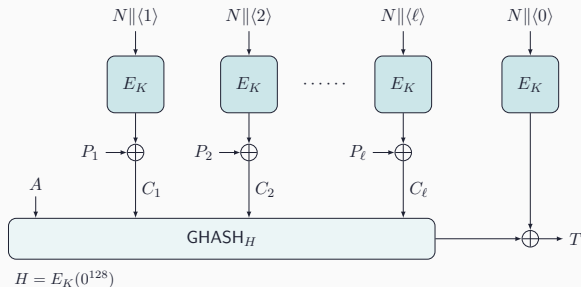
- Ideally, encryption should return random strings, and decryption never accept **fresh** inputs
- In real world, secret key  $K \xleftarrow{\$} \{0, 1\}^k$

# Standard AEAD Security



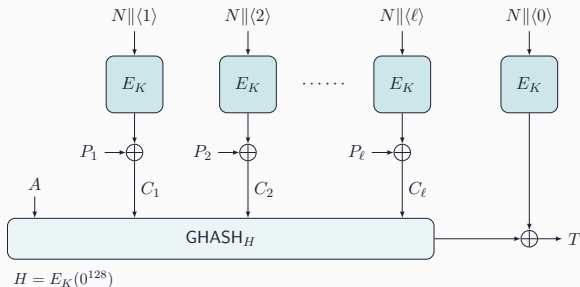
- Ideally, encryption should return random strings, and decryption never accept **fresh** inputs
- In real world, secret key  $K \xleftarrow{\$} \{0, 1\}^k$
- Distinguisher restrictions:
  - Never queries the decryption oracle with a tuple  $(C, T, N, A)$  already output by encryption oracle
  - Never repeats a nonce for enc queries (dec queries are free)

## Example: GCM Mode (Simplified)



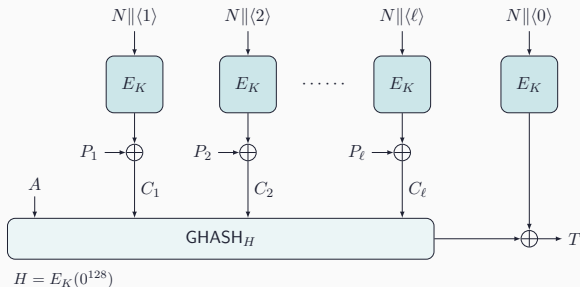
- Widely used (e.g., TLS, SSH, WPA3, ...), standardized in [Dwo07]

## Example: GCM Mode (Simplified)



- Widely used (e.g., TLS, SSH, WPA3, ...), standardized in [Dwo07]
- GHASH: polynomial (addition and multiplication over finite field) with coefficients computed from AD/CT blocks, evaluated at  $H$  (plus padding)

## Example: GCM Mode (Simplified)



- Widely used (e.g., TLS, SSH, WPA3, ...), standardized in [Dwo07]
- GHASH: polynomial (addition and multiplication over finite field) with coefficients computed from AD/CT blocks, evaluated at  $H$  (plus padding)
- $E$  good PRP  $\implies$  security (no better than birthday bound [NOMI15])

# Nonce Reuse

In practice, nonces may get repeated:

- Counters need reliable state: reboots, rollbacks, several devices sharing one key
- Random nonces collide after  $\approx 2^{m/2}$  encryptions ( $m$ : nonce size)
- Implementation or protocol bugs

# Nonce Reuse

In practice, nonces may get repeated:

- Counters need reliable state: reboots, rollbacks, several devices sharing one key
- Random nonces collide after  $\approx 2^{m/2}$  encryptions ( $m$ : nonce size)
- Implementation or protocol bugs

Consequences for GCM if  $N$  repeats:

- Same keystream:  $C \oplus C' = P \oplus P' \rightarrow$  confidentiality lost
- The two tags give a polynomial equation in  $H \rightarrow$  may solve for  $H \rightarrow$  forgeries

# Nonce Reuse

In practice, nonces may get repeated:

- Counters need reliable state: reboots, rollbacks, several devices sharing one key
- Random nonces collide after  $\approx 2^{m/2}$  encryptions ( $m$ : nonce size)
- Implementation or protocol bugs

Consequences for GCM if  $N$  repeats:

- Same keystream:  $C \oplus C' = P \oplus P' \rightarrow$  confidentiality lost
- The two tags give a polynomial equation in  $H \rightarrow$  may solve for  $H \rightarrow$  forgeries

In real life: HTTPS servers repeating GCM nonces [BZD<sup>+</sup>16]; KRACK [VP17] forcing nonce reuse in WPA2

# Nonce Reuse

In practice, nonces may get repeated:

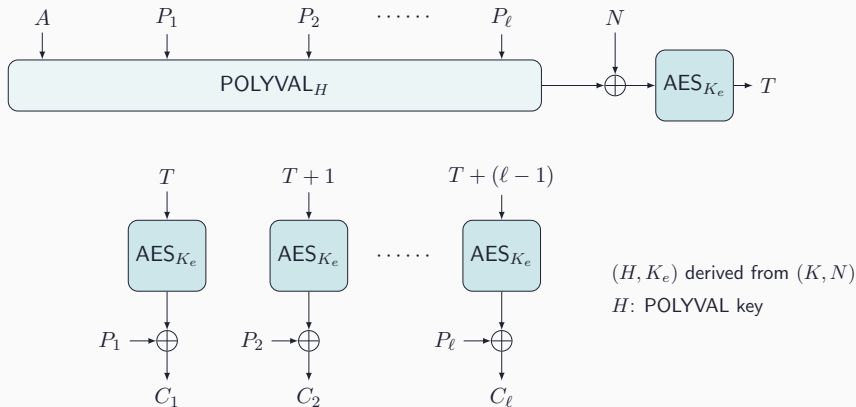
- Counters need reliable state: reboots, rollbacks, several devices sharing one key
- Random nonces collide after  $\approx 2^{m/2}$  encryptions ( $m$ : nonce size)
- Implementation or protocol bugs

Consequences for GCM if  $N$  repeats:

- Same keystream:  $C \oplus C' = P \oplus P' \rightarrow$  confidentiality lost
- The two tags give a polynomial equation in  $H \rightarrow$  may solve for  $H \rightarrow$  forgeries

In real life: HTTPS servers repeating GCM nonces [BZD<sup>+</sup>16]; KRACK [VP17] forcing nonce reuse in WPA2  $\implies$  would be nice to have some degree of resistance against nonce-reuse

# AES-GCM-SIV (Simplified Diaam)



4

- Introduced in [GL15]; the diagram above shows its evolution AES-GCM-SIV (RFC 8452)
- Proved secure, even in the event of nonce reuse [IS17]

AEAD sometimes used in contexts outside of their intended use

AEAD sometimes used in contexts outside of their intended use

- Example: **Invisible salamanders** [DGRW18] on Messenger message franking → one ciphertext shows an abusive image to the recipient, and a harmless one to Facebook when reported

AEAD sometimes used in contexts outside of their intended use

- Example: **Invisible salamanders** [DGRW18] on Messenger message franking → one ciphertext shows an abusive image to the recipient, and a harmless one to Facebook when reported
- **Key-commitment** [BH22]:
  - (CMT-1) hard to find  $(K, N, A, M), (K', N', A', M')$  with  $C = C'$  and  $K \neq K'$
  - (CMT-4, also called context commitment) hard to find  $(K, N, A, M) \neq (K', N', A', M')$  with  $C = C'$

AEAD sometimes used in contexts outside of their intended use

- Example: **Invisible salamanders** [DGRW18] on Messenger message franking  $\rightarrow$  one ciphertext shows an abusive image to the recipient, and a harmless one to Facebook when reported
- **Key-commitment** [BH22]:
  - (CMT-1) hard to find  $(K, N, A, M), (K', N', A', M')$  with  $C = C'$  and  $K \neq K'$
  - (CMT-4, also called context commitment) hard to find  $(K, N, A, M) \neq (K', N', A', M')$  with  $C = C'$
- GCM fails: given  $K, K'$ , GHASH is linear in  $C \Rightarrow$  solve one linear equation to make both tags match

AEAD sometimes used in contexts outside of their intended use

- Example: **Invisible salamanders** [DGRW18] on Messenger message franking  $\rightarrow$  one ciphertext shows an abusive image to the recipient, and a harmless one to Facebook when reported
- **Key-commitment** [BH22]:
  - (CMT-1) hard to find  $(K, N, A, M), (K', N', A', M')$  with  $C = C'$  and  $K \neq K'$
  - (CMT-4, also called context commitment) hard to find  $(K, N, A, M) \neq (K', N', A', M')$  with  $C = C'$
- GCM fails: given  $K, K'$ , GHASH is linear in  $C \Rightarrow$  solve one linear equation to make both tags match
- Possible to generically repair a non-committing AEAD, e.g., [ADG<sup>+</sup>22]

- One primitive for confidentiality and authenticity, used everywhere

## AEAD: Conclusion

- One primitive for confidentiality and authenticity, used everywhere
- Weak spot: nonces, one repetition can break security
  - Misuse resistance (AES-GCM-SIV) limits the damage, but needs two passes

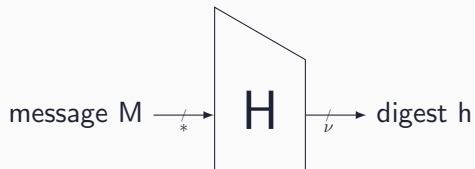
## AEAD: Conclusion

- One primitive for confidentiality and authenticity, used everywhere
- Weak spot: nonces, one repetition can break security
  - Misuse resistance (AES-GCM-SIV) limits the damage, but needs two passes
- Other requirements beyond “classical” security:
  - key-commitment: the adversary chooses the keys
  - release of unverified plaintext [ABL<sup>+</sup>14]: plaintext output before the tag is checked
  - leakage resilience: side channels

# Hash Functions

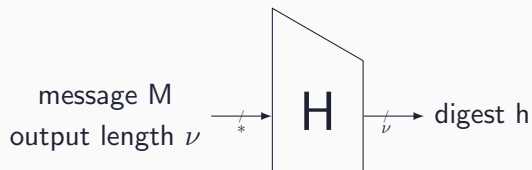
---

# Hash Functions



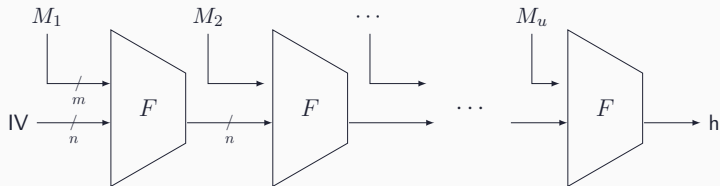
- A hash function maps an arbitrarily long message  $M$  to a fixed-size digest  $h$

# Hash Functions



- A hash function maps an arbitrarily long message  $M$  to a fixed-size digest  $h$
- Generalized version: [eXtensible Output Function \(XOF\)](#)

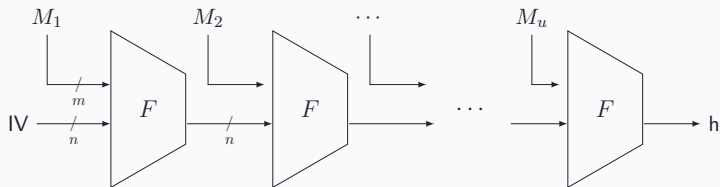
# Merkle-Damgård (MD) Construction [Mer89, Dam89]



- Construction underlying the SHA-2 family
- Message is injectively padded and split into  $m$ -bit blocks, which are processed iteratively via a **compression function**

$$\begin{aligned} F : \{0, 1\}^n \times \{0, 1\}^m &\longrightarrow \{0, 1\}^n \\ (CV, M) &\longrightarrow CV' \end{aligned}$$

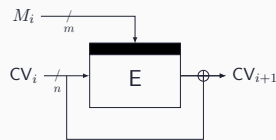
# Merkle-Damgård (MD) Construction [Mer89, Dam89]



- Construction underlying the SHA-2 family
- Message is injectively padded and split into  $m$ -bit blocks, which are processed iteratively via a **compression function**

$$\begin{aligned} F : \{0, 1\}^n \times \{0, 1\}^m &\longrightarrow \{0, 1\}^n \\ (CV, M) &\longrightarrow CV' \end{aligned}$$

- $F$  can be built with block ciphers, e.g., Davies-Meyer



- $F$  = Davies–Meyer with a dedicated block cipher
- 64 rounds for SHA-224 and SHA-256, 80 rounds for the others
- Padding: 1, then 0's, then the length

| Function    | $\nu$ | $n$ | $m$  |
|-------------|-------|-----|------|
| SHA-224     | 224   | 256 | 512  |
| SHA-256     | 256   | 256 | 512  |
| SHA-384     | 384   | 512 | 1024 |
| SHA-512     | 512   | 512 | 1024 |
| SHA-512/256 | 256   | 512 | 1024 |

## Hash Function Security: Pre/SecPre/Col (1/2)

### Preimage Resistance (Pre)

Given  $h$ , it should be hard to find  $M$  such that  $\mathcal{H}(M) = h$

## Hash Function Security: Pre/SecPre/Col (1/2)

### Preimage Resistance (Pre)

Given  $h$ , it should be hard to find  $M$  such that  $\mathcal{H}(M) = h$

### Second Preimage Resistance (SecPre)

Given  $M$ , it should be hard to find  $M' \neq M$  such that  $\mathcal{H}(M') = \mathcal{H}(M)$

# Hash Function Security: Pre/SecPre/Col (1/2)

## Preimage Resistance (Pre)

Given  $h$ , it should be hard to find  $M$  such that  $\mathcal{H}(M) = h$

## Second Preimage Resistance (SecPre)

Given  $M$ , it should be hard to find  $M' \neq M$  such that  $\mathcal{H}(M') = \mathcal{H}(M)$

## Collision Resistance (Col)

It should be hard to find  $M' \neq M$  such that  $\mathcal{H}(M') = \mathcal{H}(M)$

# Hash Function Security: Pre/SecPre/Col (1/2)

## Preimage Resistance (Pre)

Given  $h$ , it should be hard to find  $M$  such that  $\mathcal{H}(M) = h$

## Second Preimage Resistance (SecPre)

Given  $M$ , it should be hard to find  $M' \neq M$  such that  $\mathcal{H}(M') = \mathcal{H}(M)$

## Collision Resistance (Col)

It should be hard to find  $M' \neq M$  such that  $\mathcal{H}(M') = \mathcal{H}(M)$

**Those above statements are informal**

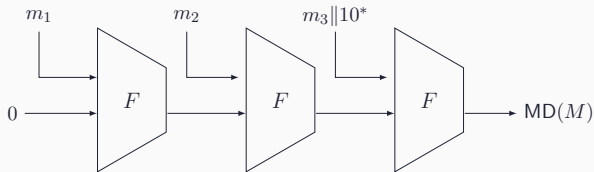
**Col** for a fixed  $\mathcal{H}$ : an adversary printing a hard-coded collision *exists*

→ [RS04]: hash family  $\mathcal{H}_K$ , seven notions for **Pre**, **SecPre**, **Col** with random or adversarial key/target; **Col** needs a random key

→ [Rog06]: keep  $\mathcal{H}$  unkeyed, give explicit reductions

## Hash Function Security: Pre/SecPre/Col (2/2)

Take MD Construction with  $10^*$  padding

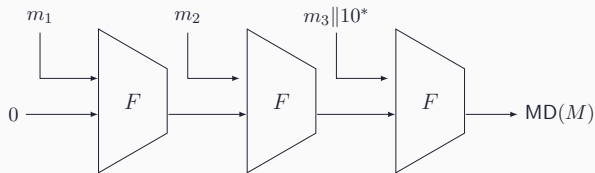


$$M = m_1 \| m_2 \| m_3$$

- $MD^F$  with a random  $F$  is Pre/SecPre/Col

## Hash Function Security: Pre/SecPre/Col (2/2)

Take MD Construction with  $10^*$  padding

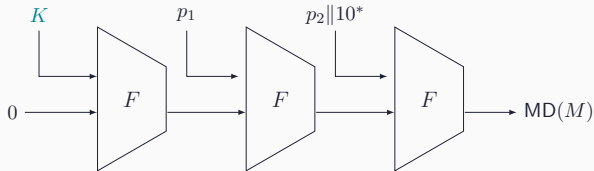


$$M = m_1 \| m_2 \| m_3$$

- $MD^F$  with a random  $F$  is Pre/SecPre/Col
- Does not imply security in every use-case due to [length extensions attacks](#)

## Hash Function Security: Pre/SecPre/Col (2/2)

Take MD Construction with  $10^*$  padding

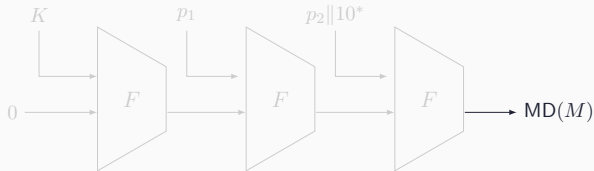


$$M = K \parallel p_1 \parallel p_2$$

- $\text{MD}^F$  with a random  $F$  is Pre/SecPre/Col
- Does not imply security in every use-case due to **length extensions attacks**

## Hash Function Security: Pre/SecPre/Col (2/2)

Take MD Construction with  $10^*$  padding

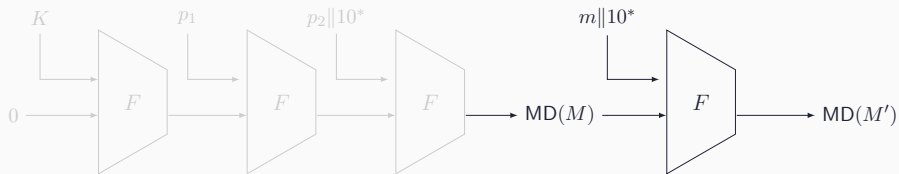


$$M = K || p_1 || p_2$$

- $MD^F$  with a random  $F$  is Pre/SecPre/Col
- Does not imply security in every use-case due to [length extensions attacks](#)

## Hash Function Security: Pre/SecPre/Col (2/2)

Take MD Construction with  $10^*$  padding

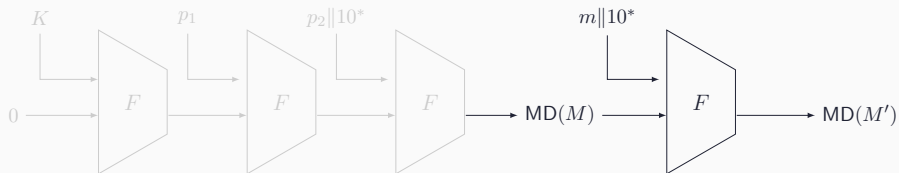


$$M' = K \parallel p_1 \parallel p_2 \parallel 10^* \parallel m$$

- $MD^F$  with a random  $F$  is Pre/SecPre/Col
- Does not imply security in every use-case due to **length extensions attacks**

## Hash Function Security: Pre/SecPre/Col (2/2)

Take MD Construction with  $10^*$  padding

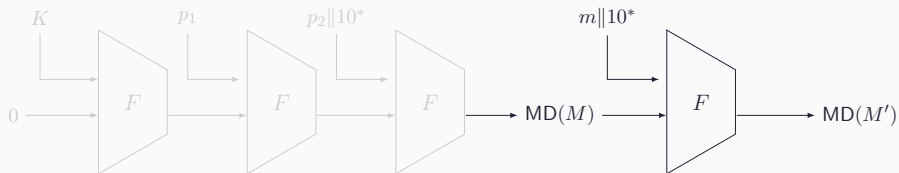


$$M' = K \parallel p_1 \parallel p_2 \parallel 10^* \parallel m$$

- $MD^F$  with a random  $F$  is Pre/SecPre/Col
- Does not imply security in every use-case due to [length extensions attacks](#)
- Note: MD can be repaired with small modifications (truncation, prefix-free padding, MDP, ...), or using HMAC

## Hash Function Security: Pre/SecPre/Col (2/2)

Take MD Construction with  $10^*$  padding



$$M' = K \| p_1 \| p_2 \| 10^* \| m$$

- $MD^F$  with a random  $F$  is Pre/SecPre/Col
- Does not imply security in every use-case due to **length extensions attacks**
- Note: MD can be repaired with small modifications (truncation, prefix-free padding, MDP, ...), or using HMAC
- What we really want: our hash function behaves like a **random oracle**

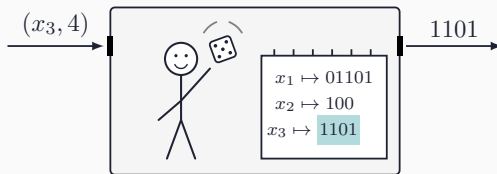
## Random Oracle

A random oracle  $\mathcal{RO}$  keeps internally a database of input-output tuples, initially empty

# Random Oracle

A random oracle  $\mathcal{RO}$  keeps internally a database of input-output tuples, initially empty

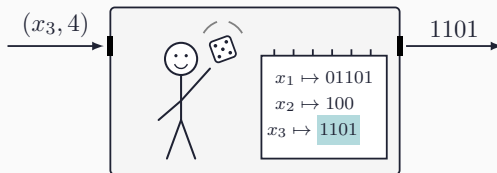
**Fresh query**  $(x, \ell)$ : uniformly random  $\ell$  bits



# Random Oracle

A random oracle  $\mathcal{RO}$  keeps internally a database of input-output tuples, initially empty

**Fresh query**  $(x, \ell)$ : uniformly random  $\ell$  bits



**Query**  $(x, \ell)$  with known  $x$ : consistent answer, sample fresh bits as needed



## Hash Function Security: a RO-like Hash Function?

- To be secure in all use-cases, we want the hash function to **behave like a random oracle**
- Many security proofs model the hash function as such

## Hash Function Security: a RO-like Hash Function?

- To be secure in all use-cases, we want the hash function to **behave like a random oracle**
- Many security proofs model the hash function as such
- But no concrete hash function can: it has a short description, a random oracle has none. This was used to build schemes provably secure in the ROM and broken for *every* instantiation [CGH04]

## Hash Function Security: a RO-like Hash Function?

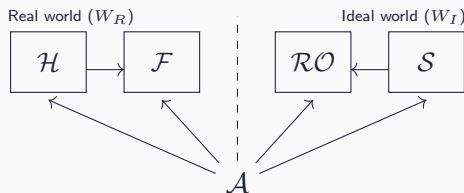
- To be secure in all use-cases, we want the hash function to **behave like a random oracle**
- Many security proofs model the hash function as such
- But no concrete hash function can: it has a short description, a random oracle has none. This was used to build schemes provably secure in the ROM and broken for *every* instantiation [CGH04]
- Still, we would like to argue that the construction rules out generic attacks that treat the underlying primitive as a black-box

## Hash Function Security: a RO-like Hash Function?

- To be secure in all use-cases, we want the hash function to **behave like a random oracle**
- Many security proofs model the hash function as such
- But no concrete hash function can: it has a short description, a random oracle has none. This was used to build schemes provably secure in the ROM and broken for *every* instantiation [CGH04]
- Still, we would like to argue that the construction rules out generic attacks that treat the underlying primitive as a black-box  $\implies$  Indifferentiability!

# Indifferentiability [MRH04, CDMP05] (1/2)

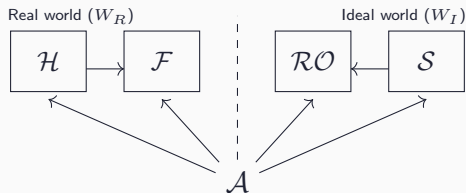
States that a hash function construction  $\mathcal{H}$  based on a **random** primitive  $\mathcal{F}$  behaves like a random oracle  $\mathcal{RO}$  in any **single-stage** game <sup>2</sup>



- $\mathcal{H}$  is indifferentiable from  $\mathcal{RO}$  **for some** simulator  $\mathcal{S}$  whenever any  $\mathcal{A}$  can distinguish the two worlds only with a negligible probability
- Advantage is denoted as  $\mathbf{Adv}_{\mathcal{H}, \mathcal{S}}^{\text{indiff}}(\mathcal{A})$

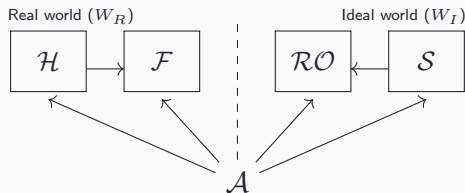
<sup>2</sup>See [RSS11] for examples where composition fails

## Indifferentiability (2/2)



- For any single-stage game, indifferentiability allows to replace  $(\mathcal{H}^{\mathcal{F}}, \mathcal{F})$  with  $(\mathcal{RO}, \mathcal{S}[\mathcal{RO}])$

## Indifferentiability (2/2)



- For any single-stage game, indifferentiability allows to replace  $(\mathcal{H}^{\mathcal{F}}, \mathcal{F})$  with  $(\mathcal{RO}, \mathcal{S}[\mathcal{RO}])$
- In particular, for  $\text{atk} \in \{\text{pre}, \text{secre}, \text{col}\}$ , we have

$$\mathbf{Adv}_{\mathcal{H}}^{\text{atk}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathcal{H}, \mathcal{S}}^{\text{indiff}}(\mathcal{A}') + \mathbf{Adv}_{\mathcal{RO}}^{\text{atk}}(\mathcal{A}'')$$

for some  $\mathcal{A}'$  and  $\mathcal{A}''$  with comparable resources

## Sponges

---

# The Sponge and Cryptographic Competitions



# The Sponge and Cryptographic Competitions

- 2007 + Introduction of sponge [BDPV07]
- 2007 + SHA-3 competition begins
- Call for hashing

# The Sponge and Cryptographic Competitions

- 2007 — Introduction of sponge [BDPV07]
- 2007 — SHA-3 competition begins  
Call for hashing
- 2011 — Introduction of the duplex with SpongeWrap [BDPV11a]

# The Sponge and Cryptographic Competitions

- 2007 — Introduction of sponge [BDPV07]
- 2007 — SHA-3 competition begins
  - Call for hashing
- 2011 — Introduction of the duplex with SpongeWrap [BDPV11a]
- 2012 — SHA-3 competition ends
  - Keccak selected as winner (NIST FIPS PUB 202)

# The Sponge and Cryptographic Competitions

- 2007 — Introduction of sponge [BDPV07]
- 2007 — SHA-3 competition begins
  - Call for hashing
- 2011 — Introduction of the duplex with SpongeWrap [BDPV11a]
- 2012 — SHA-3 competition ends
  - Keccak selected as winner (NIST FIPS PUB 202)
- 2013 — CAESAR competition begins
  - Call for AEAD; 57 submissions, of which  $\approx 10$  sponge/duplex-based

# The Sponge and Cryptographic Competitions

- 2007 — Introduction of sponge [BDPV07]
- 2007 — SHA-3 competition begins
  - Call for hashing
- 2011 — Introduction of the duplex with SpongeWrap [BDPV11a]
- 2012 — SHA-3 competition ends
  - Keccak selected as winner (NIST FIPS PUB 202)
- 2013 — CAESAR competition begins
  - Call for AEAD; 57 submissions, of which  $\approx 10$  sponge/duplex-based
- 2019 — CAESAR competition ended
  - Ascon won in the category **lightweight**

# The Sponge and Cryptographic Competitions

- 2007 — Introduction of sponge [BDPV07]
- 2007 — SHA-3 competition begins
  - Call for hashing
- 2011 — Introduction of the duplex with SpongeWrap [BDPV11a]
- 2012 — SHA-3 competition ends
  - Keccak selected as winner (NIST FIPS PUB 202)
- 2013 — CAESAR competition begins
  - Call for AEAD; 57 submissions, of which  $\approx 10$  sponge/duplex-based
- 2019 — CAESAR competition ended
  - Ascon won in the category **lightweight**
- 2019 — NIST lightweight crypto competition begins
  - Call for AEAD (optional hashing)
  - 57 submissions, of which  $\approx 22$  sponge/duplex-based

# The Sponge and Cryptographic Competitions

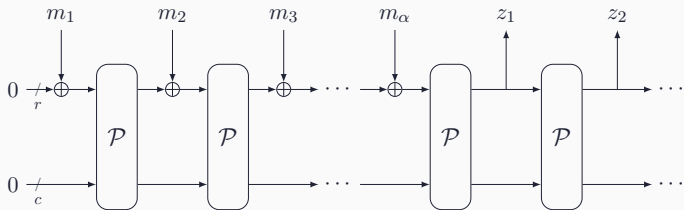
- 2007 — Introduction of sponge [BDPV07]
- 2007 — SHA-3 competition begins
  - Call for hashing
- 2011 — Introduction of the duplex with SpongeWrap [BDPV11a]
- 2012 — SHA-3 competition ends
  - Keccak selected as winner (NIST FIPS PUB 202)
- 2013 — CAESAR competition begins
  - Call for AEAD; 57 submissions, of which  $\approx 10$  sponge/duplex-based
- 2019 — CAESAR competition ended
  - Ascon won in the category **lightweight**
- 2019 — NIST lightweight crypto competition begins
  - Call for AEAD (optional hashing)
  - 57 submissions, of which  $\approx 22$  sponge/duplex-based
- 2023 — NIST Lightweight crypto competition ends
  - Ascon selected as overall winner (NIST SP 800-232)

# The Sponge and Cryptographic Competitions

Also: Arithmetization-Oriented Sponges.  
e.g., [GKR<sup>+</sup>21, BBC<sup>+</sup>23, GKL<sup>+</sup>22]

- 2007 — Introduction of sponge [BDPV07]
- 2007 — SHA-3 competition begins
  - Call for hashing
- 2011 — Introduction of the duplex with SpongeWrap [BDPV11a]
- 2012 — SHA-3 competition ends
  - Keccak selected as winner (NIST FIPS PUB 202)
- 2013 — CAESAR competition begins
  - Call for AEAD; 57 submissions, of which  $\approx 10$  sponge/duplex-based
- 2019 — CAESAR competition ended
  - Ascon won in the category **lightweight**
- 2019 — NIST lightweight crypto competition begins
  - Call for AEAD (optional hashing)
  - 57 submissions, of which  $\approx 22$  sponge/duplex-based
- 2023 — NIST Lightweight crypto competition ends
  - Ascon selected as overall winner (NIST SP 800-232)

# The Sponge Construction



- Permutation of size  $b = r + c$  bits:
  - rate  $r$ : **outer part** of the state (efficiency parameter)
  - capacity  $c$ : **inner part** of the state (security parameter)
- The message is padded with a **sponge-compliant padding** (e.g.,  $10^*$ )

## Indifferentiability Security of the Sponge [BDPV08]

- There exists  $\mathcal{S}$  (using at most  $q$   $\mathcal{RO}$ -queries) s.t., for any  $\mathcal{A}$  with query budget  $q$ :<sup>a</sup>

$$\mathbf{Adv}_{\text{Sponge}, \mathcal{S}}^{\text{indiff}}(\mathcal{A}) \leq \frac{2q(q+1)}{2^c}$$

---

<sup>a</sup>I.e., the number of permutation calls that would be induced in the real world

# Indifferentiability Security of the Sponge [BDPV08]

- There exists  $\mathcal{S}$  (using at most  $q$   $\mathcal{RO}$ -queries) s.t., for any  $\mathcal{A}$  with query budget  $q$ :<sup>a</sup>

$$\mathbf{Adv}_{\text{Sponge}, \mathcal{S}}^{\text{indiff}}(\mathcal{A}) \leq \frac{2q(q+1)}{2^c}$$

- When truncating the output to  $\nu$  bits, this implies
  - Col security up to  $\approx \min\{2^{\nu/2}, 2^{c/2}\}$  evaluations
  - SecPre security up to  $\approx \min\{2^{\nu}, 2^{c/2}\}$  evaluations
  - Pre security up to  $\approx \min\{2^{\nu}, 2^{c/2}\}$  evaluations
- Matching attacks already described by [BDPV07]

---

<sup>a</sup>I.e., the number of permutation calls that would be induced in the real world

# Indifferentiability Security of the Sponge [BDPV08]

- There exists  $\mathcal{S}$  (using at most  $q$   $\mathcal{RO}$ -queries) s.t., for any  $\mathcal{A}$  with query budget  $q$ :<sup>a</sup>

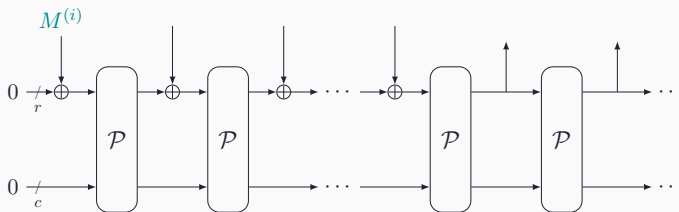
$$\mathbf{Adv}_{\text{Sponge}, \mathcal{S}}^{\text{indiff}}(\mathcal{A}) \leq \frac{2q(q+1)}{2^c}$$

- When truncating the output to  $\nu$  bits, this implies
  - Col security up to  $\approx \min\{2^{\nu/2}, 2^{c/2}\}$  evaluations **tight**
  - SecPre security up to  $\approx \min\{2^{\nu}, 2^{c/2}\}$  evaluations **tight**
  - Pre security up to  $\approx \min\{2^{\nu}, 2^{c/2}\}$  evaluations **not tight**, needs a dedicated proof [LM22]: secure up to  $\approx \min\{2^{\nu}, \max\{2^{\nu-r}, 2^{c/2}\}\}$
- Matching attacks already described by [BDPV07]

---

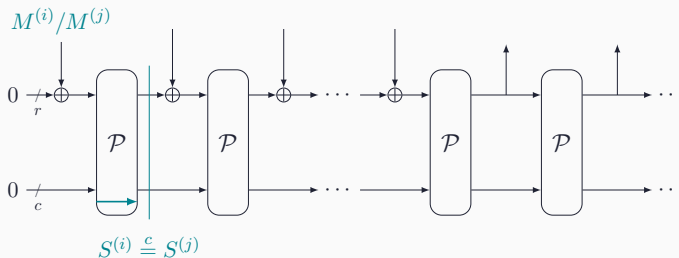
<sup>a</sup>I.e., the number of permutation calls that would be induced in the real world

## Collision Attack [BDPV07]



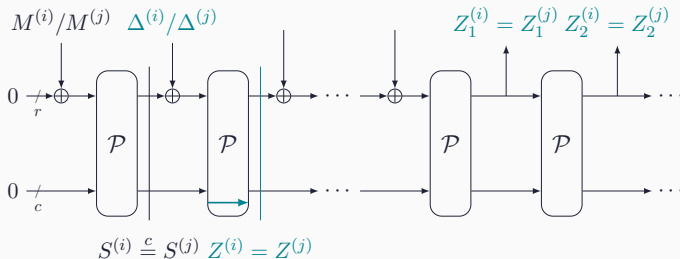
- Absorb  $2^{c/2}$  different message blocks

## Collision Attack [BDPV07]



- Absorb  $2^{c/2}$  different message blocks
- With high probability, there exists  $S^{(i)} \neq S^{(j)}$  that collide on their inner part

## Collision Attack [BDPV07]



- Absorb  $2^{c/2}$  different message blocks
- With high probability, there exists  $S^{(i)} \neq S^{(j)}$  that **collide on their inner part**
- Compensate the difference in the outer part with  $\Delta^{(i)} = [S^{(i)}]_r$ ,  $\Delta^{(j)} = [S^{(j)}]_r$

## SHA-3 in Practice

- $\mathcal{P} = \text{KECCAK-}f[1600]$ :  $b = 1600$ ,  
24 rounds
- Padding: domain bits (01 for SHA-3,  
1111 for SHAKE) then  $10^*1$

| Function | $r$  | $c$  |
|----------|------|------|
| SHA3-224 | 1152 | 448  |
| SHA3-256 | 1088 | 512  |
| SHA3-384 | 832  | 768  |
| SHA3-512 | 576  | 1024 |
| SHAKE128 | 1344 | 256  |
| SHAKE256 | 1088 | 512  |

## SHA-3 in Practice

- $\mathcal{P} = \text{KECCAK-}f[1600]$ :  $b = 1600$ ,  
24 rounds
- Padding: domain bits (01 for SHA-3,  
1111 for SHAKE) then  $10^*1$
- SHA3- $n$  uses  $c = 2n$
- SHAKE: output length free (a XOF)

| Function | $r$  | $c$  |
|----------|------|------|
| SHA3-224 | 1152 | 448  |
| SHA3-256 | 1088 | 512  |
| SHA3-384 | 832  | 768  |
| SHA3-512 | 576  | 1024 |
| SHAKE128 | 1344 | 256  |
| SHAKE256 | 1088 | 512  |

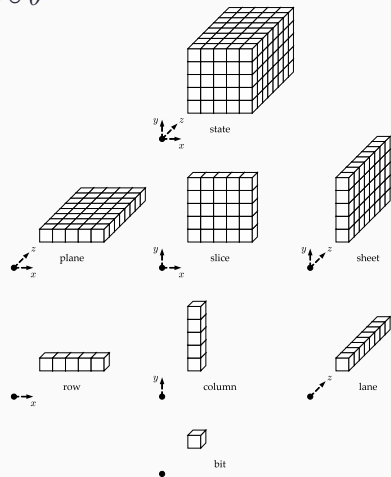
- $\mathcal{P} = \text{KECCAK-}f[1600]$ :  $b = 1600$ ,  
24 rounds
- Padding: domain bits (01 for SHA-3,  
1111 for SHAKE) then  $10^*1$

| Function | $r$  | $c$  |
|----------|------|------|
| SHA3-224 | 1152 | 448  |
| SHA3-256 | 1088 | 512  |
| SHA3-384 | 832  | 768  |
| SHA3-512 | 576  | 1024 |
| SHAKE128 | 1344 | 256  |
| SHAKE256 | 1088 | 512  |

- SHA3- $n$  uses  $c = 2n$
- SHAKE: output length free (a XOF)
- RFC 9861: specifies KangarooTwelve and TurboSHAKE (round reduced versions of SHA-3)

# Inside Keccak- $f[1600]$

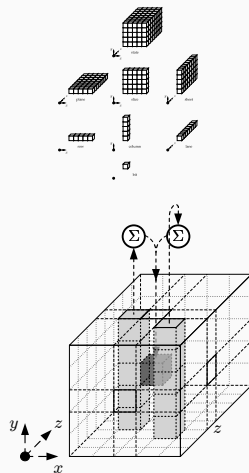
State:  $5 \times 5$  lanes of 64 bits; 24 rounds of  $\iota \circ \chi \circ \pi \circ \rho \circ \theta$



# Inside Keccak- $f[1600]$

State:  $5 \times 5$  lanes of 64 bits; 24 rounds of  $\iota \circ \chi \circ \pi \circ \rho \circ \theta$

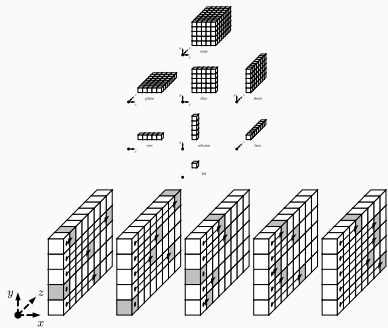
- $\theta$ : XOR each bit the bitwise sum of the parities of two columns  $\rightarrow$  diffusion
- $\rho$ : translations within the lanes  $\rightarrow$  inter-slice dispersion
- $\pi$ : transpose the lanes  $\rightarrow$  breaks alignment (long-term diffusion)
- $\chi$ :  $x_i \leftarrow x_i \oplus (\neg x_{i+1} \wedge x_{i+2})$  on rows  $\rightarrow$  only nonlinear step (algebraic degree 2)
- $\iota$ : XOR a round constant into one lane  $\rightarrow$  breaks symmetry between rounds



# Inside Keccak- $f[1600]$

State:  $5 \times 5$  lanes of 64 bits; 24 rounds of  $\iota \circ \chi \circ \pi \circ \rho \circ \theta$

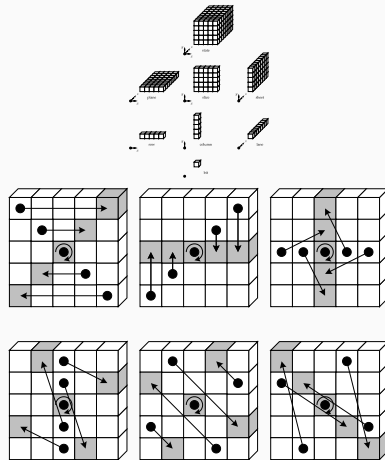
- $\theta$ : XOR each bit the bitwise sum of the parities of two columns  $\rightarrow$  diffusion
- $\rho$ : translations within the lanes  $\rightarrow$  inter-slice dispersion
- $\pi$ : transpose the lanes  $\rightarrow$  breaks alignment (long-term diffusion)
- $\chi$ :  $x_i \leftarrow x_i \oplus (\neg x_{i+1} \wedge x_{i+2})$  on rows  $\rightarrow$  only nonlinear step (algebraic degree 2)
- $\iota$ : XOR a round constant into one lane  $\rightarrow$  breaks symmetry between rounds



# Inside Keccak-f[1600]

State:  $5 \times 5$  lanes of 64 bits; 24 rounds of  $\iota \circ \chi \circ \pi \circ \rho \circ \theta$

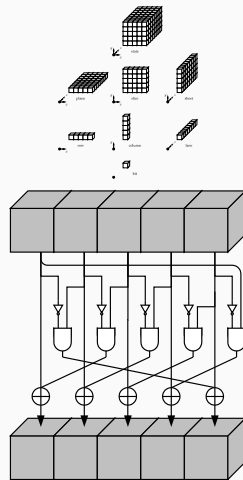
- $\theta$ : XOR each bit the bitwise sum of the parities of two columns  $\rightarrow$  diffusion
- $\rho$ : translations within the lanes  $\rightarrow$  inter-slice dispersion
- $\pi$ : transpose the lanes  $\rightarrow$  breaks alignment (long-term diffusion)
- $\chi$ :  $x_i \leftarrow x_i \oplus (\neg x_{i+1} \wedge x_{i+2})$  on rows  $\rightarrow$  only nonlinear step (algebraic degree 2)
- $\iota$ : XOR a round constant into one lane  $\rightarrow$  breaks symmetry between rounds



# Inside Keccak- $f[1600]$

State:  $5 \times 5$  lanes of 64 bits; 24 rounds of  $\iota \circ \chi \circ \pi \circ \rho \circ \theta$

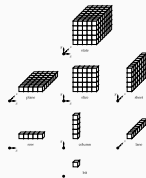
- $\theta$ : XOR each bit the bitwise sum of the parities of two columns  $\rightarrow$  diffusion
- $\rho$ : translations within the lanes  $\rightarrow$  inter-slice dispersion
- $\pi$ : transpose the lanes  $\rightarrow$  breaks alignment (long-term diffusion)
- $\chi$ :  $x_i \leftarrow x_i \oplus (\neg x_{i+1} \wedge x_{i+2})$  on rows  $\rightarrow$  only nonlinear step (algebraic degree 2)
- $\iota$ : XOR a round constant into one lane  $\rightarrow$  breaks symmetry between rounds



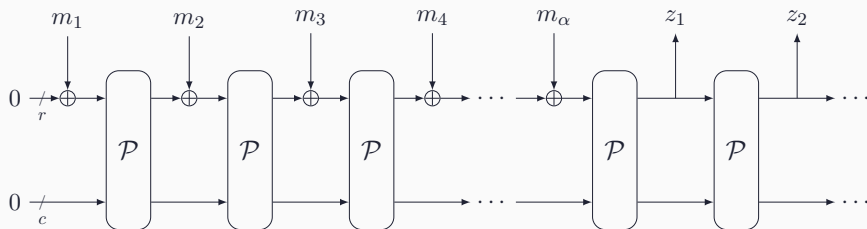
# Inside Keccak- $f[1600]$

State:  $5 \times 5$  lanes of 64 bits; 24 rounds of  $\iota \circ \chi \circ \pi \circ \rho \circ \theta$

- $\theta$ : XOR each bit the bitwise sum of the parities of two columns  $\rightarrow$  diffusion
- $\rho$ : translations within the lanes  $\rightarrow$  inter-slice dispersion
- $\pi$ : transpose the lanes  $\rightarrow$  breaks alignment (long-term diffusion)
- $\chi$ :  $x_i \leftarrow x_i \oplus (\neg x_{i+1} \wedge x_{i+2})$  on rows  $\rightarrow$  only nonlinear step (algebraic degree 2)
- $\iota$ : XOR a round constant into one lane  $\rightarrow$  breaks symmetry between rounds

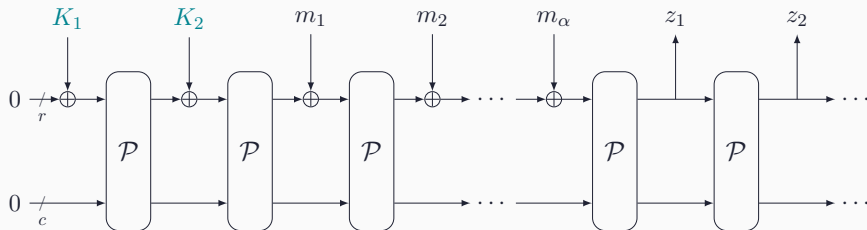


# Sponge-Based PRF Constructions



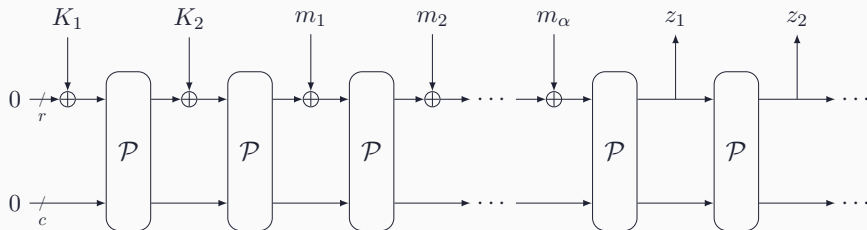
- The sponge can be tweaked

# Sponge-Based PRF Constructions



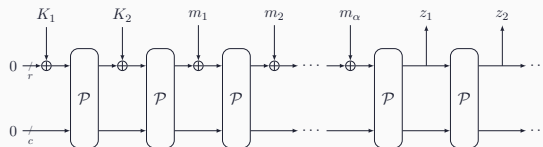
- The sponge can be tweaked to build a pseudorandom function

# Sponge-Based PRF Constructions



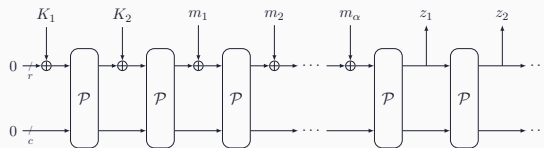
- The sponge can be tweaked to build a pseudorandom function
- Blackbox invocation of sponge with input  $K\|M$
- Useful for MAC or keystream generation
- The construction is named **Outer Keyed Sponge** [BDPV11b]

## Outer Keyed Sponge: Security



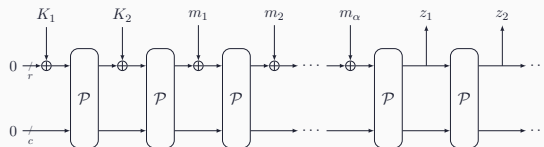
- PRF security of OKS can be deduced from indifferentiability ...

## Outer Keyed Sponge: Security



- PRF security of OKS can be deduced from indistinguishability ...
- ... but indistinguishability does not distinguish between offline and online complexities

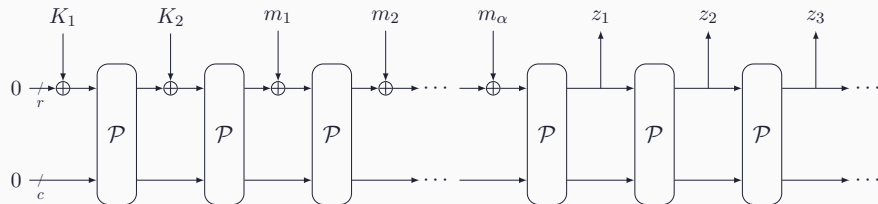
## Outer Keyed Sponge: Security



- PRF security of OKS can be deduced from indistinguishability ...
- ... but indistinguishability does not distinguish between offline and online complexities
- A dedicated analysis yields a tighter bound [GPT15, ADMV15, NY16, Men18]. Ignoring constant/log factors, and assuming  $q_{\text{off}} \geq q_{\text{on}}$ :

$$\text{Adv}_{\text{OKS}}^{\text{PRF}}(\mathcal{A}) = \tilde{\mathcal{O}}\left(\frac{q_{\text{off}}}{2^k} + \frac{q_{\text{off}}q_{\text{on}}}{2^c}\right)$$

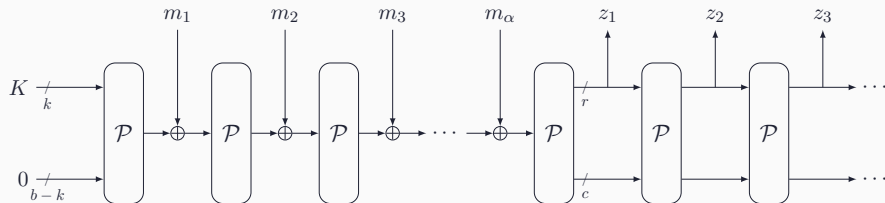
## Sponge-Based PRF Constructions: Optimizations



A long line of research showed:



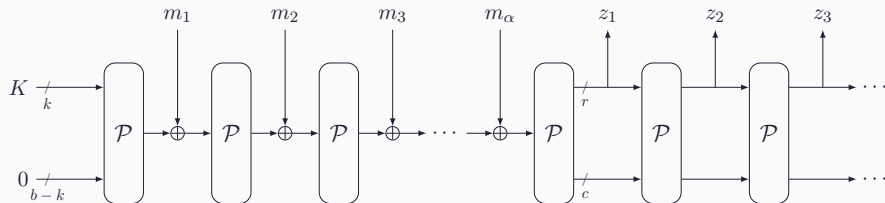
# Sponge-Based PRF Constructions: Optimizations



A long line of research showed:

- State may be initialized with key, in various ways [BDPV12, DM24], ...
- May absorb over the entire state [BDPV11b, BDPV12, CDH<sup>+</sup>12, ADMV15], ...

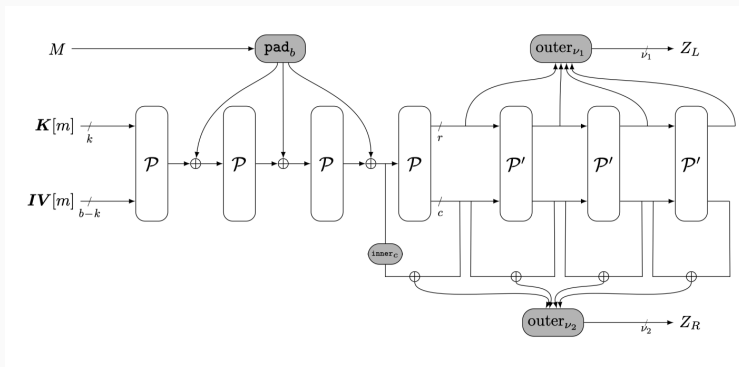
# Sponge-Based PRF Constructions: Optimizations



A long line of research showed:

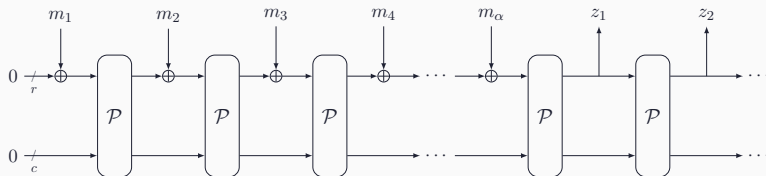
- State may be initialized with key, in various ways [BDPV12, DM24], ...
- May absorb over the entire state [BDPV11b, BDPV12, CDH<sup>+</sup>12, ADMV15], ...
- ... without decreasing generic security. Also, security bounds that capture fine-grained adversarial power were conducted

# Sponge-Based PRF Constructions: Optimizations



Recently: squeezing can also be optimized, extracting **more than  $r$  bits per permutation call** without losing generic security (MacaKey [LB25])

# Sponge-Based AEAD Constructions

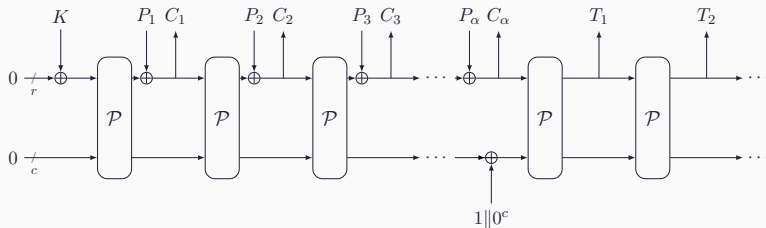


- The sponge can be tweaked

---

<sup>3</sup>Figure simplifies SpongeWrap

# Sponge-Based AEAD Constructions

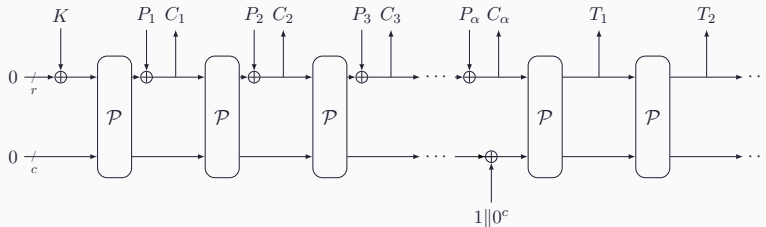


- The sponge can be tweaked for authenticated encryption purposes

---

<sup>3</sup>Figure simplifies SpongeWrap

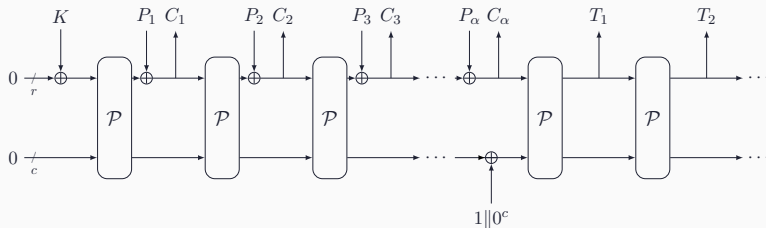
# Sponge-Based AEAD Constructions



- The sponge can be tweaked for authenticated encryption purposes
- The first construction to achieve that is named [SpongeWrap](#) [BDPV11a]<sup>3</sup>

<sup>3</sup>Figure simplifies SpongeWrap

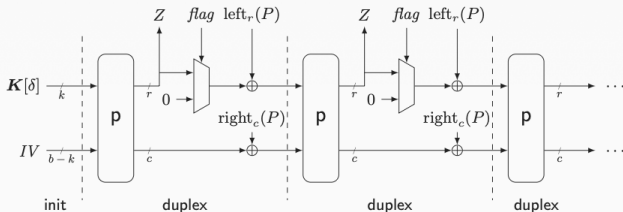
# Sponge-Based AEAD Constructions



- The sponge can be tweaked for authenticated encryption purposes
- The first construction to achieve that is named [SpongeWrap](#) [BDPV11a]<sup>3</sup>
- Built on top of [the duplex](#) [BDPV11a], a sort of online version of the sponge

<sup>3</sup>Figure simplifies SpongeWrap

# Full-State Keyed Duplex [DMV17]

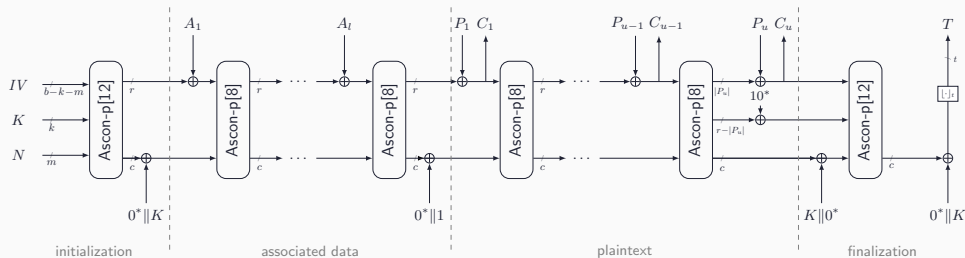


(picture taken from Bart Mennink's talk Understanding the Duplex and Its Security) [Men23]

- Very general object that captures a lot of use-cases for **keyed applications**<sup>4</sup>
- Bounds parametrized by various use-cases, complex to interpret

<sup>4</sup>Unkeyed versions of the (not full-state) duplex were also treated, see [DFG23]

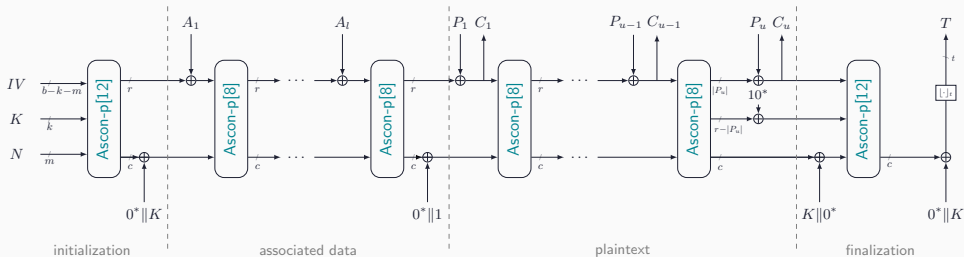
# Ascon-AEAD128 – NIST SP 800-232 [SMKK25]



## Variant of SpongeWrap

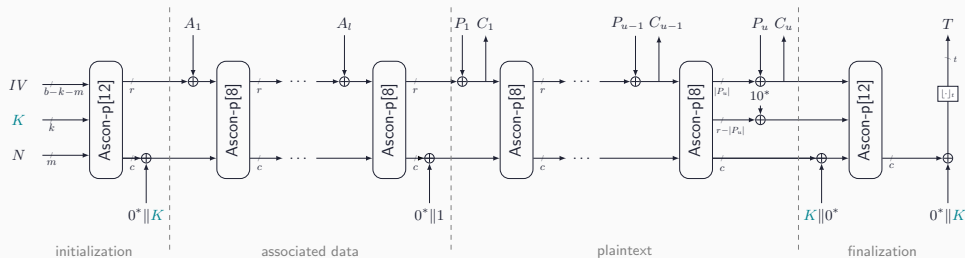
- $b = 320, m, t = 128, c = 192, r = 128, k = 128$

# Ascon-AEAD128 – NIST SP 800-232 [SMKK25]



## Variant of SpongeWrap

- $b = 320, m, t = 128, c = 192, r = 128, k = 128$
- Initial and final permutations are stronger than inner ones



## Variant of SpongeWrap

- $b = 320, m, t = 128, c = 192, r = 128, k = 128$
- Initial and final permutations are stronger than inner ones
- Additional **key blindings** around “outer” permutations  $\implies$  better authenticity and leakage resilience

## Accordion Modes

---

# Motivation: NIST's Review of AES and Its Modes

In 2021, NIST began reviewing the AES standard and the SP 800-38 modes

## Some findings [MD24]

- **Data limits** are increasingly relevant [Dwo07]:
  - at most  $2^{39}$  – 256 bits per GCM call
  - at most  $2^{32}$  calls per key with random IVs (or IVs that are not 96 bits)
- **Security expectations have evolved**: growing demand for key- and context-committing AEAD

# Motivation: NIST's Review of AES and Its Modes

In 2021, NIST began reviewing the AES standard and the SP 800-38 modes

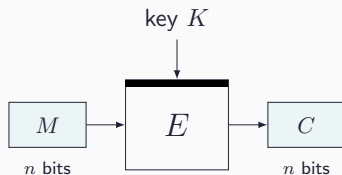
## Some findings [MD24]

- **Data limits** are increasingly relevant [Dwo07]:
  - at most  $2^{39}$  – 256 bits per GCM call
  - at most  $2^{32}$  calls per key with random IVs (or IVs that are not 96 bits)
- **Security expectations have evolved**: growing demand for key- and context-committing AEAD

**One implication:** new constructions are needed that have more robust security, and scale to larger data volumes  $\implies$  **accordions** are a promising solution

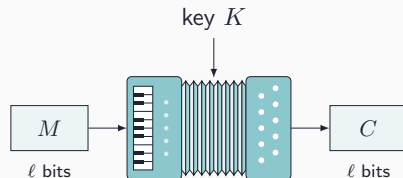
NIST IR 8459, *Report on the Block Cipher Modes of Operation in the NIST SP 800-38 Series* (2024).

## Accordions: The Idea (1/2)



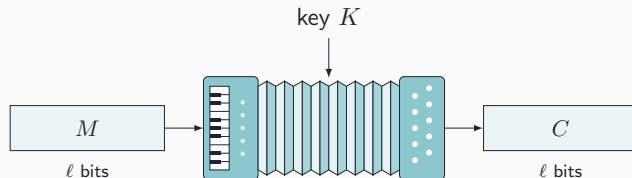
- A block cipher only enciphers inputs of **one** size:  $n$  bits

## Accordions: The Idea (1/2)



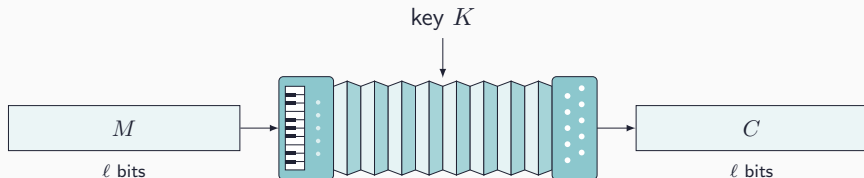
- A block cipher only enciphers inputs of **one** size:  $n$  bits
- We would like the same object for messages of **any** length

## Accordions: The Idea (1/2)



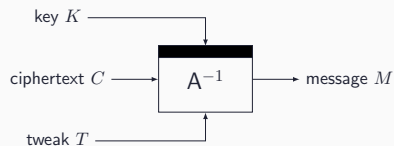
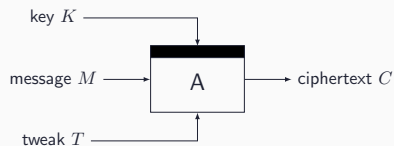
- A block cipher only enciphers inputs of **one** size:  $n$  bits
- We would like the same object for messages of **any** length. . .

## Accordions: The Idea (1/2)



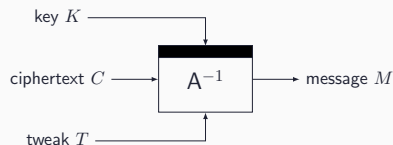
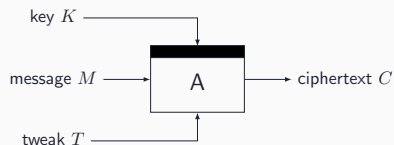
- A block cipher only enciphers inputs of **one** size:  $n$  bits
- We would like the same object for messages of **any** length. . .
- . . . a cipher that **stretches** to the length of the message: an **accordion**!

## Accordions: The Idea (2/2)



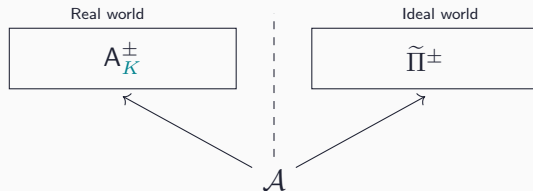
- Formally: a **tweakable variable-input-length wide block cipher** (TVIL-WBC) built from a block cipher  $E$  [CDD<sup>+</sup>25]
  - For all  $K, T$  and allowed  $\ell$ :  $A(K, T, \cdot)$  is a permutation of  $\{0, 1\}^\ell \implies |C| = |M|$

## Accordions: The Idea (2/2)



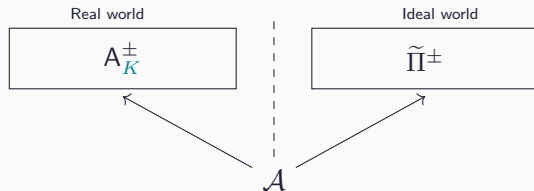
- Formally: a **tweakable variable-input-length wide block cipher** (TVIL-WBC) built from a block cipher  $E$  [CDD<sup>+</sup>25]
  - For all  $K, T$  and allowed  $\ell$ :  $A(K, T, \cdot)$  is a permutation of  $\{0, 1\}^\ell \implies |C| = |M|$
- Intuition: changing **any** bit of the input randomizes **every** bit of the output (unlike, e.g., CTR)

# Accordion Security



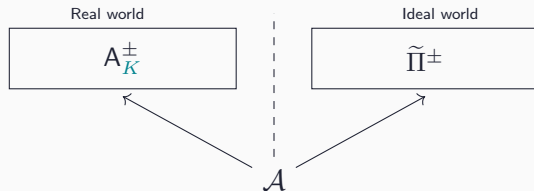
- Same game as for SPRP security, but **tweakable** and **variable-input-length**

# Accordion Security



- Same game as for SPRP security, but **tweakable** and **variable-input-length**
- Real world:  $K \xleftarrow{\$} \{0, 1\}^k$ , queries  $\text{enc}(T, x)$  and  $\text{dec}(T, x)$  answered with  $A^\pm(K, T, \cdot)$
- Ideal world: for each **distinct**  $(T, |x|)$ , an **independent** random permutation  $\tilde{\Pi}_{T, |x|}$  of  $\{0, 1\}^{|x|}$

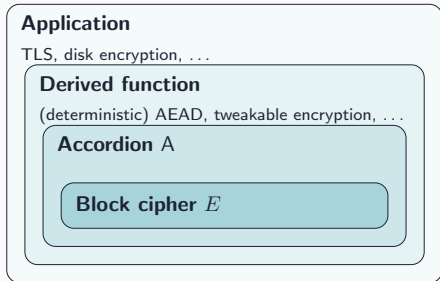
# Accordion Security



- Same game as for SPRP security, but **tweakable** and **variable-input-length**
- Real world:  $K \xleftarrow{\$} \{0, 1\}^k$ , queries  $\text{enc}(T, x)$  and  $\text{dec}(T, x)$  answered with  $A^\pm(K, T, \cdot)$
- Ideal world: for each **distinct**  $(T, |x|)$ , an **independent** random permutation  $\tilde{\Pi}_{T, |x|}$  of  $\{0, 1\}^{|x|}$
- $\mathcal{A}$ 's resources:  $q$  queries,  $\sigma$  blocks in total, running time  $t$

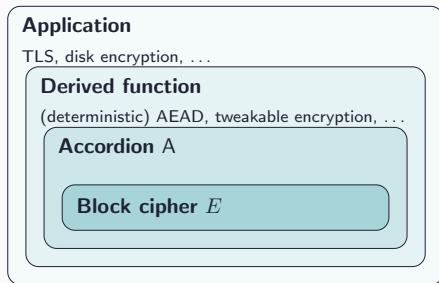
$$\text{Adv}_A^{\text{TSPRP}}(\mathcal{A}) = \left| \Pr \left( \mathcal{A}^{A^\pm(K, \cdot, \cdot)} \rightarrow 1 : K \xleftarrow{\$} \{0, 1\}^k \right) - \Pr \left( \mathcal{A}^{\tilde{\Pi}^\pm} \rightarrow 1 \right) \right|$$

# Many Functionalities



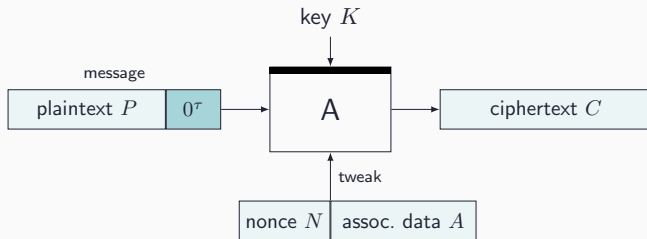
- A **derived function** is an **input encoding** that enables a specific functionality from the accordion [CDD<sup>+</sup>25]

# Many Functionalities



- A **derived function** is an **input encoding** that enables a specific functionality from the accordion [CDD<sup>+</sup>25]
- Separates enciphering from the role of each input (plaintext, AD, nonce, ...)

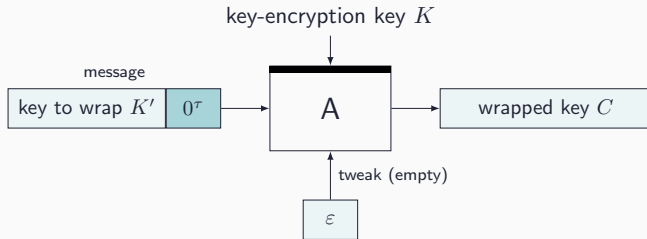
## Derived Functions: Examples



**Encode-then-encipher** [BR00] add redundancy, then encipher

- **AEAD:** tweak =  $(N, A)$ , message =  $P || 0^\tau$ ; decryption checks the  $\tau$  zeros

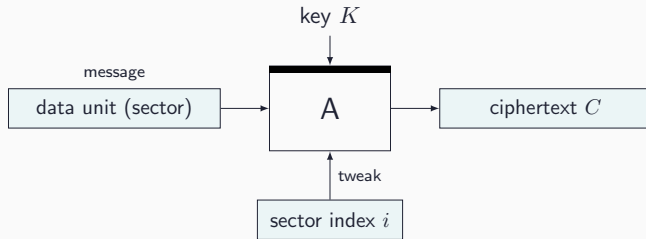
## Derived Functions: Examples



**Encode-then-encipher** [BR00] add redundancy, then encipher

- **AEAD**: tweak =  $(N, A)$ , message =  $P \parallel 0^\tau$ ; decryption checks the  $\tau$  zeros
- **Deterministic AE** (e.g., key wrapping [RS06]): no tweak needed, message =  $K' \parallel 0^\tau$

## Derived Functions: Examples



**Encode-then-encipher** [BR00] add redundancy, then encipher

- **AEAD:** tweak =  $(N, A)$ , message =  $P \parallel 0^\tau$ ; decryption checks the  $\tau$  zeros
- **Deterministic AE** (e.g., key wrapping [RS06]): no tweak needed, message =  $K' \parallel 0^\tau$
- **Tweakable encryption** (storage): tweak = index of the data unit, message = its content; no expansion, and changing the tweak should be cheap

## AEAD from an Accordion: What Do We Get?

With  $C = A(K, (N, A), P\|0^\tau)$ , assuming a good TSPRP:

- **Confidentiality:** ok
- **Authenticity:** a forgery must decrypt to  $\tau$  zeros, which happens w.p.  $\approx 2^{-\tau}$  per attempt

## AEAD from an Accordion: What Do We Get?

With  $C = A(K, (N, A), P \| 0^\tau)$ , assuming a good TSPRP:

- **Confidentiality:** ok
- **Authenticity:** a forgery must decrypt to  $\tau$  zeros, which happens w.p.  $\approx 2^{-\tau}$  per attempt
- **Nonce-misuse resistance:** repeating  $(N, A, P)$  only reveals that the plaintexts are equal

## AEAD from an Accordion: What Do We Get?

With  $C = A(K, (N, A), P \| 0^\tau)$ , assuming a good TSPRP:

- **Confidentiality:** ok
- **Authenticity:** a forgery must decrypt to  $\tau$  zeros, which happens w.p.  $\approx 2^{-\tau}$  per attempt
- **Nonce-misuse resistance:** repeating  $(N, A, P)$  only reveals that the plaintexts are equal
- **Key-commitment:** not implied (the adversary knows the keys!)  $\implies$  has to be proven separately (may follow from, e.g., indifferentiability)

Besides the aforementioned requirements, the accordion should operate on an underlying block cipher, which

- Can take advantage of AES instructions
- Simplifies the security analysis: reduce to the SPRP security of  $E$
- Has a solid base of existing research

## Requirements [CDD<sup>+</sup>25]

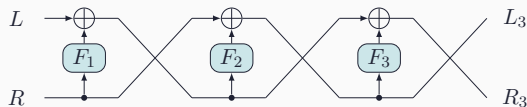
Besides the aforementioned requirements, the accordion should operate on an underlying block cipher, which

- Can take advantage of AES instructions
- Simplifies the security analysis: reduce to the SPRP security of  $E$
- Has a solid base of existing research

In addition it should have nice performance:

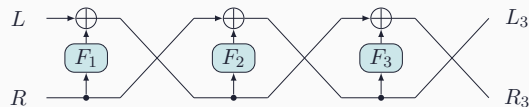
- Target: not significantly more than **twice the cost of GCM**
- Parallelizable, small overhead on short inputs

## Intermezzo: Feistel Networks (1/2)



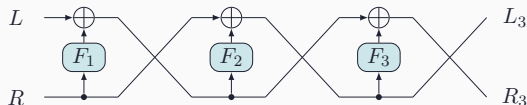
- Used, among others, for designing block ciphers: DES, Blowfish, Camellia, ...
- One round on  $2n$  bits:  $(L, R) \mapsto (R, L \oplus F_i(R))$
- $F_i$  is a **keyed** function

## Intermezzo: Feistel Networks (1/2)



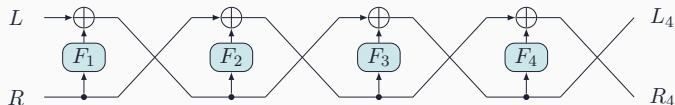
- Used, among others, for designing block ciphers: DES, Blowfish, Camellia, ...
- One round on  $2n$  bits:  $(L, R) \mapsto (R, L \oplus F_i(R))$
- $F_i$  is a **keyed** function
- Invertible, even if the  $F_i$ 's are not

## Intermezzo: Feistel Networks (2/2)



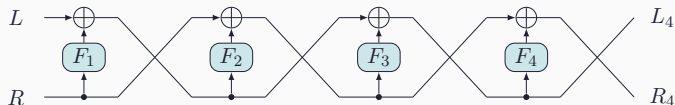
- Luby–Rackoff [LR88]: with independent random functions  $F_i : \{0, 1\}^n \rightarrow \{0, 1\}^n$ ,
  - 3 rounds: PRP, 4 rounds: SPRP, with advantage  $\lesssim q^2/2^n$  (birthday bound on the half-block)

## Intermezzo: Feistel Networks (2/2)



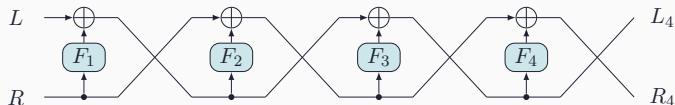
- Luby–Rackoff [LR88]: with independent random functions  $F_i : \{0, 1\}^n \rightarrow \{0, 1\}^n$ ,
  - 3 rounds: PRP, 4 rounds: SPRP, with advantage  $\lesssim q^2/2^n$  (birthday bound on the half-block)

## Intermezzo: Feistel Networks (2/2)



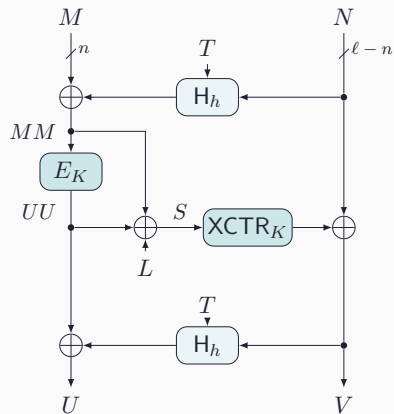
- Luby–Rackoff [LR88]: with independent random functions  $F_i : \{0, 1\}^n \rightarrow \{0, 1\}^n$ ,
  - 3 rounds: PRP, 4 rounds: SPRP, with advantage  $\lesssim q^2/2^n$  (birthday bound on the half-block)
- Naor–Reingold [NR99]: first and last rounds can be replaced by cheap pairwise-independent permutations  $\implies$  universal hashing

## Intermezzo: Feistel Networks (2/2)



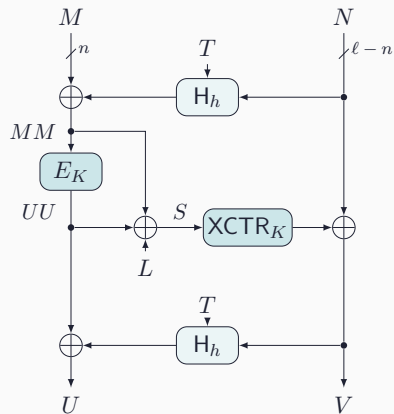
- Luby–Rackoff [LR88]: with independent random functions  $F_i : \{0, 1\}^n \rightarrow \{0, 1\}^n$ ,
  - 3 rounds: PRP, 4 rounds: SPRP, with advantage  $\lesssim q^2/2^n$  (birthday bound on the half-block)
- Naor–Reingold [NR99]: first and last rounds can be replaced by cheap pairwise-independent permutations  $\implies$  **universal hashing**
- **Unbalanced** Feistel: branches of different sizes ( $n$  and  $\ell - n$  bits)  $\implies$  variable input length!

## NIST Choice: HCTR2 [CHB21] (1/3)



- Crowley, Huckleberry, Biggers (Google, 2021); fixes two bugs of HCTR [WFW05]
- Deployed in Linux Kernel
- **Unbalanced** Feistel-like network: narrow branch  $n$  bits, wide branch  $\ell - n$  bits

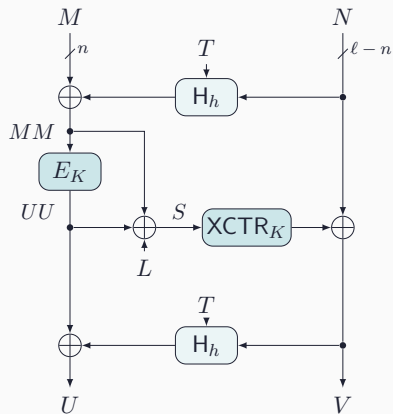
## NIST Choice: HCTR2 [CHB21] (1/3)



- Crowley, Huckleberry, Biggers (Google, 2021); fixes two bugs of HCTR [WFW05]
- Deployed in Linux Kernel
- **Unbalanced** Feistel-like network: narrow branch  $n$  bits, wide branch  $\ell - n$  bits
- Takes as input any length  $\ell \geq n$

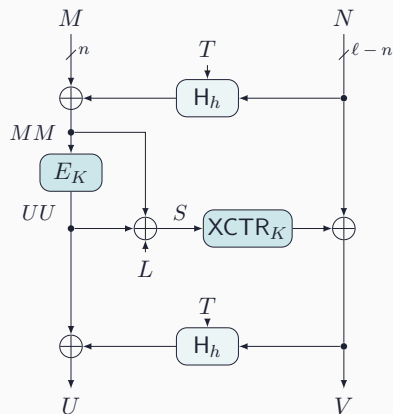


## NIST Choice: HCTR2 [CHB21] (1/3)



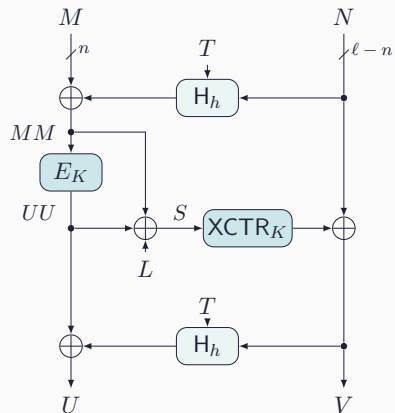
- Crowley, Huckleberry, Biggers (Google, 2021); fixes two bugs of HCTR [WFW05]
- Deployed in Linux Kernel
- **Unbalanced** Feistel-like network: narrow branch  $n$  bits, wide branch  $\ell - n$  bits
- Takes as input any length  $\ell \geq n$
- $P = M || N$  with  $|M| = n$ ,  $C = U || V$
- One key:  $h = E_K(\langle 0 \rangle)$ ,  $L = E_K(\langle 1 \rangle)$
- $H_h$ : polynomial hash (uses POLYVAL)

# NIST Choice: HCTR2 [CHB21] (1/3)

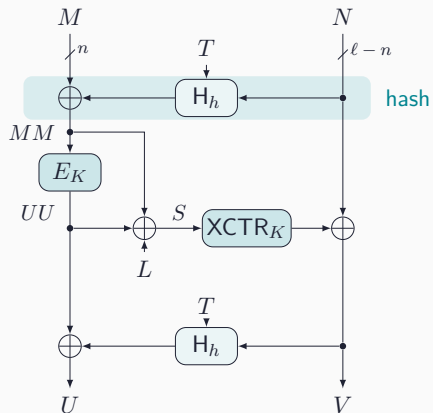


- Crowley, Huckleberry, Biggers (Google, 2021); fixes two bugs of HCTR [WFW05]
- Deployed in Linux Kernel
- **Unbalanced** Feistel-like network: narrow branch  $n$  bits, wide branch  $\ell - n$  bits
- Takes as input any length  $\ell \geq n$
- $P = M \| N$  with  $|M| = n$ ,  $C = U \| V$
- One key:  $h = E_K(\langle 0 \rangle)$ ,  $L = E_K(\langle 1 \rangle)$
- $H_h$ : polynomial hash (uses POLYVAL)
- $XCTR_K(S) = E_K(S \oplus \langle 1 \rangle) \| E_K(S \oplus \langle 2 \rangle) \| \dots$

## NIST Choice: HCTR2 (2/3)

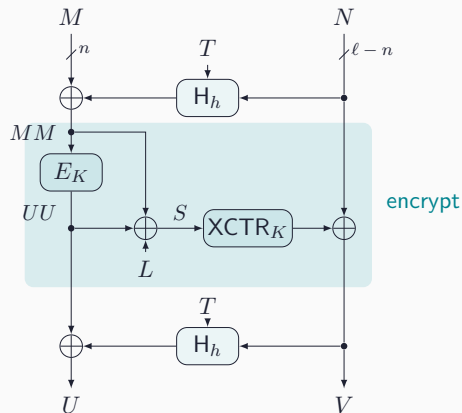


## NIST Choice: HCTR2 (2/3)



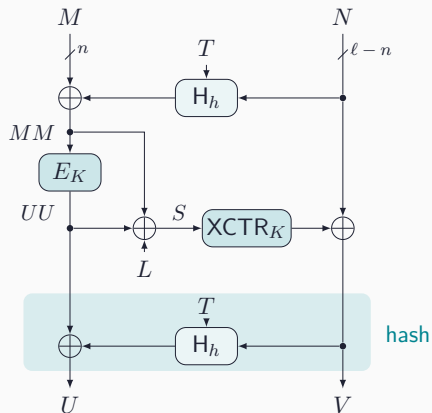
- Hash:  $MM = M \oplus H_h(T, N)$

## NIST Choice: HCTR2 (2/3)



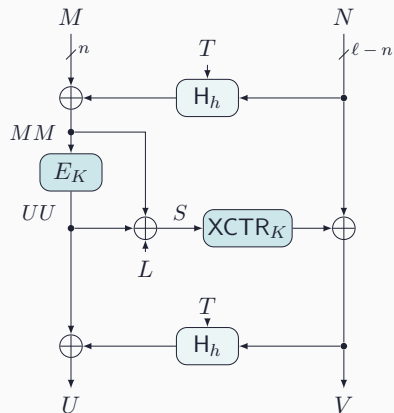
- Hash:  $MM = M \oplus H_h(T, N)$
- **Encrypt:**  $UU = E_K(MM)$ ,  
 $V = N \oplus XCTR_K(MM \oplus UU \oplus L)$

## NIST Choice: HCTR2 (2/3)



- Hash:  $MM = M \oplus H_h(T, N)$
- Encrypt:  $UU = E_K(MM)$ ,  
 $V = N \oplus XCTR_K(MM \oplus UU \oplus L)$
- Hash:  $U = UU \oplus H_h(T, V)$

## NIST Choice: HCTR2 (2/3)



- Hash:  $MM = M \oplus H_h(T, N)$
- Encrypt:  $UU = E_K(MM)$ ,  
 $V = N \oplus XCTR_K(MM \oplus UU \oplus L)$
- Hash:  $U = UU \oplus H_h(T, V)$
- Decryption: same picture upside down,  
 $E_K^{-1}$  only on the narrow branch

**TSPRP Security** [CHB21]: for any  $\mathcal{A}$  making  $q$  queries of  $\sigma$  blocks in total,

$$\mathbf{Adv}_{\text{HCTR2}[E]}^{\text{TSPRP}}(\mathcal{A}) \leq \mathbf{Adv}_E^{\text{SPRP}}(\mathcal{A}') + \frac{3\sigma^2 + 2q\sigma + q^2 + 7\sigma + 2}{2^{n+1}}$$

for some  $\mathcal{A}'$  making  $\sigma + 2$  queries

**TSPRP Security** [CHB21]: for any  $\mathcal{A}$  making  $q$  queries of  $\sigma$  blocks in total,

$$\mathbf{Adv}_{\text{HCTR2}[E]}^{\text{TSPRP}}(\mathcal{A}) \leq \mathbf{Adv}_E^{\text{SPRP}}(\mathcal{A}') + \frac{3\sigma^2 + 2q\sigma + q^2 + 7\sigma + 2}{2^{n+1}}$$

for some  $\mathcal{A}'$  making  $\sigma + 2$  queries

- Standard model: security reduces to the SPRP security of the inner primitive
- Proof with H-coefficient technique

**TSPRP Security** [CHB21]: for any  $\mathcal{A}$  making  $q$  queries of  $\sigma$  blocks in total,

$$\mathbf{Adv}_{\text{HCTR2}[E]}^{\text{TSPRP}}(\mathcal{A}) \leq \mathbf{Adv}_E^{\text{SPRP}}(\mathcal{A}') + \frac{3\sigma^2 + 2q\sigma + q^2 + 7\sigma + 2}{2^{n+1}}$$

for some  $\mathcal{A}'$  making  $\sigma + 2$  queries

- Standard model: security reduces to the SPRP security of the inner primitive
- Proof with H-coefficient technique

**Key-Commitment** proven separately, latest result from CRYPTO 2026 [CCH<sup>+</sup>26]

**HCTR2 with AES** ( $n = 128$ )  
= Acc128

ready to standardize, but birthday bound:  $\leq 2^{48}$  bits per key

**HCTR2 with AES ( $n = 128$ )**  
= Acc128

ready to standardize, but birthday bound:  $\leq 2^{48}$  bits per key

more security?

**1. Wider block**

HCTR2 with Rijndael-256 = Acc256

**2. Beyond birthday bound**

HCTR-based mode with AES = BBBAcc

**HCTR2 with AES ( $n = 128$ )**  
= Acc128

ready to standardize, but birthday bound:  $\leq 2^{48}$  bits per key

more security?

**1. Wider block**

HCTR2 with Rijndael-256 = Acc256

same mode,  $n = 256$

$\Rightarrow$  birthday bound = 128 bits

but a less studied cipher

**2. Beyond birthday bound**

HCTR-based mode with AES = BBBAcc

**HCTR2 with AES ( $n = 128$ )**  
= Acc128

ready to standardize, but birthday bound:  $\leq 2^{48}$  bits per key

more security?

## 1. Wider block

HCTR2 with Rijndael-256 = Acc256

same mode,  $n = 256$

$\Rightarrow$  birthday bound = 128 bits

but a less studied cipher

## 2. Beyond birthday bound

HCTR-based mode with AES = BBBAcc

e.g. HCTR2-TwKD, Cascaded

HCTR2 [CHM<sup>+</sup>25]

but more complex; often BBB only  
if a tweak is not reused too much

## Side Quest: Rijndael-256

- AES = Rijndael restricted to 128-bit blocks; Rijndael supports blocks and keys from 128 to 256 bits
- 128-bit blocks  $\implies$  birthday-bound limits on the data per key
- December 2024: NIST proposes to standardize Rijndael-256 [Nat24]: 256-bit block, 256-bit key, same design and high-level round structure as AES (14 rounds)

- Accordion = tweakable wide block cipher from a block cipher: **one** primitive, many derived functions (AEAD, storage encryption, key wrapping, ...)
- Encode-then-encipher turns it into robust AE
- The price: several passes
- NIST: HCTR2-based Acc128 first, then Acc256 (with Rijndael-256) and BBBAcc
- This is an on-going standardization effort

## Conclusion

---

## What About Quantum Computers? (1/3)

- Some public-key crypto (RSA, ECC) is **broken** by Shor's algorithm  
     $\implies$  new post-quantum standards

# What About Quantum Computers? (1/3)

- Some public-key crypto (RSA, ECC) is **broken** by Shor's algorithm  
     $\implies$  new post-quantum standards
- Symmetric crypto is generally less affected
  - Best generic attack: Grover, finds a  $k$ -bit key in  $\approx 2^{k/2}$  steps instead of  $2^k$
  - Grover parallelizes badly

## What About Quantum Computers? (1/3)

- Some public-key crypto (RSA, ECC) is **broken** by Shor's algorithm  
     $\implies$  new post-quantum standards
- Symmetric crypto is generally less affected
  - Best generic attack: Grover, finds a  $k$ -bit key in  $\approx 2^{k/2}$  steps instead of  $2^k$
  - Grover parallelizes badly
- Conservative choice anyway: 256-bit keys (AES-256), longer hashes (SHA-384/512, sponges with  $c = 512$ )

## What About Quantum Computers? (2/3)

### Doubling the key size is not always enough!

- Two models: Q1 = quantum computer, but classical queries to the keyed oracle;  
Q2 = superposition queries (stronger, debatable in practice)

## What About Quantum Computers? (2/3)

### Doubling the key size is not always enough!

- Two models: Q1 = quantum computer, but **classical** queries to the keyed oracle; Q2 = **superposition** queries (stronger, debatable in practice)
- Q2: Simon's algorithm finds a hidden period in **polynomial** time, whatever the key size:
  - distinguishes the 3-round Feistel (Luby–Rackoff PRP) [KM10]
  - recovers the key of (single-iterated) Even–Mansour [KM12]
  - many constructions are broken, see [this talk](#)

## What About Quantum Computers? (2/3)

### Doubling the key size is not always enough!

- Two models: Q1 = quantum computer, but classical queries to the keyed oracle; Q2 = superposition queries (stronger, debatable in practice)
- Q2: Simon's algorithm finds a hidden period in polynomial time, whatever the key size:
  - distinguishes the 3-round Feistel (Luby–Rackoff PRP) [KM10]
  - recovers the key of (single-iterated) Even–Mansour [KM12]
  - many constructions are broken, see [this talk](#)
- Q1: Simon still helps “offline” [BHN<sup>+</sup>19], e.g. [BSS22]

## What About Quantum Computers? (3/3)

### Proofs against quantum adversaries are sometimes harder

- **Easy case** (Q1, standard model): the adversary sends classical queries to the device, and we reduce to the SPRP security of the block cipher  $\implies$  the classical proof **carries over** (but the underlying assumption is stronger)

# What About Quantum Computers? (3/3)

## Proofs against quantum adversaries are sometimes harder

- **Easy case** (Q1, standard model): the adversary sends classical queries to the device, and we reduce to the SPRP security of the block cipher  $\implies$  the classical proof **carries over** (but the underlying assumption is stronger)
- **Hard case**: Q2 or ideal models, where the adversary evaluates the public primitive itself, hence in superposition
  - “lazy sampling” does not carry over easily, especially for permutations
  - The sponge was proven quantum indifferentiable only in 2025 [ACMT25], with a **loose** bound: we don't know if the bound is tight

## What About Quantum Computers? (3/3)

### Proofs against quantum adversaries are sometimes harder

- **Easy case** (Q1, standard model): the adversary sends classical queries to the device, and we reduce to the SPRP security of the block cipher  $\implies$  the classical proof **carries over** (but the underlying assumption is stronger)
- **Hard case**: Q2 or ideal models, where the adversary evaluates the public primitive itself, hence in superposition
  - “lazy sampling” does not carry over easily, especially for permutations
  - The sponge was proven quantum indifferentiable only in 2025 [ACMT25], with a **loose** bound: we don't know if the bound is tight

Symmetric crypto is in  $\pm$  good shape, but our understanding still needs refining

## Lightweight Cryptography (1/2): Why?

- Crypto now runs on **extremely constrained** devices:  
RFID tags, IoT sensors, medical implants, transport tickets, supply-chain tracking, ...

## Lightweight Cryptography (1/2): Why?

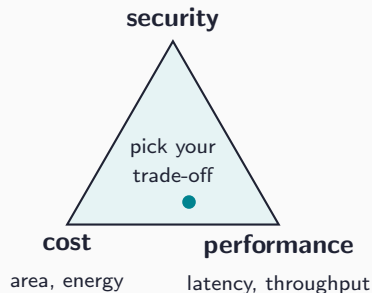
- Crypto now runs on **extremely constrained** devices:  
RFID tags, IoT sensors, medical implants, transport tickets, supply-chain tracking, ...
- AES got massive speedups, but not always enough

## Lightweight Cryptography (1/2): Why?

- Crypto now runs on **extremely constrained** devices:  
RFID tags, IoT sensors, medical implants, transport tickets, supply-chain tracking, ...
- AES got massive speedups, but not always enough
- Many dimensions to optimize:
  - area (GE, FPGA slices, RAM/ROM)
  - energy, power
  - latency (memory encryption)
  - short messages (e.g., 96-bit EPC)
  - side channels (attacker holds the device!)

## Lightweight Cryptography (1/2): Why?

- Crypto now runs on **extremely constrained** devices: RFID tags, IoT sensors, medical implants, transport tickets, supply-chain tracking, ...
- AES got massive speedups, but not always enough
- Many dimensions to optimize:
  - area (GE, FPGA slices, RAM/ROM)
  - energy, power
  - latency (memory encryption)
  - short messages (e.g., 96-bit EPC)
  - side channels (attacker holds the device!)



## Lightweight Cryptography (2/2): How?

Push the designs to their limits:

- simple operations; many iterations of very simple rounds, simple key schedules
- smaller parameters
- smaller security margins by design  $\implies$  many of them broken

## Lightweight Cryptography (2/2): How?

Push the designs to their limits:

- simple operations; many iterations of very simple rounds, simple key schedules
- smaller parameters
- smaller security margins by design  $\implies$  many of them broken

### Examples

- Block ciphers: NOEKEON, PRESENT, PRINCE (low latency), SKINNY and GIFT (tweakable / smaller), SIMON
- Ascon

## Lightweight Cryptography (2/2): How?

Push the designs to their limits:

- simple operations; many iterations of very simple rounds, simple key schedules
- smaller parameters
- smaller security margins by design  $\implies$  many of them broken

### Examples

- Block ciphers: NOEKEON, PRESENT, PRINCE (low latency), SKINNY and GIFT (tweakable / smaller), SIMON
- Ascon

The choice of a primitive depends also on the [deployment context](#)

## A Few Initiatives / Competitions / Standardization Efforts (non-exhaustive)

|         |   |                                                                 |
|---------|---|-----------------------------------------------------------------|
| 1997–00 | + | AES (NIST): Rijndael                                            |
| 2000–03 | + | NESSIE (EU): block ciphers, MACs, hashes                        |
| 2004–08 | + | eSTREAM (ECRYPT): stream ciphers (Salsa20, Trivium, ...)        |
| 2007–12 | + | SHA-3 (NIST): Keccak                                            |
| 2013–15 | + | PHC: password hashing, Argon2                                   |
| 2013–19 | + | CAESAR: authenticated encryption (Ascon, AEGIS, Deoxys-II, ...) |
| 2018–23 | + | Lightweight (NIST): Ascon                                       |
| 2023–   | + | Accordions (NIST)                                               |

## Take-Home Messages

- A handful of **recurring design patterns**: SPNs, Feistel, Even–Mansour, sponge and duplex, . . .

## Take-Home Messages

- A handful of **recurring design patterns**: SPNs, Feistel, Even–Mansour, sponge and duplex, . . .
- A security proof only holds **in its model** (nonce-respecting adversaries, key-commitment, classical vs Q1 vs Q2, . . .) and should be interpreted with care

## Take-Home Messages

- A handful of **recurring design patterns**: SPNs, Feistel, Even–Mansour, sponge and duplex, ...
- A security proof only holds **in its model** (nonce-respecting adversaries, key-commitment, classical vs Q1 vs Q2, ...) and should be interpreted with care
- The choice of an algorithm depends on the **deployment context**, not only on the target security level

## Take-Home Messages

- A handful of **recurring design patterns**: SPNs, Feistel, Even–Mansour, sponge and duplex, . . .
- A security proof only holds **in its model** (nonce-respecting adversaries, key-commitment, classical vs Q1 vs Q2, . . .) and should be interpreted with care
- The choice of an algorithm depends on the **deployment context**, not only on the target security level
- Standardization efforts and competitions stir a lot of research

## Take-Home Messages

- A handful of **recurring design patterns**: SPNs, Feistel, Even–Mansour, sponge and duplex, . . .
- A security proof only holds **in its model** (nonce-respecting adversaries, key-commitment, classical vs Q1 vs Q2, . . .) and should be interpreted with care
- The choice of an algorithm depends on the **deployment context**, not only on the target security level
- Standardization efforts and competitions stir a lot of research
- Symmetric crypto is maybe more stable than public-key crypto, but there is still a lot to explore

[ABL<sup>+</sup>14] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda.

**How to Securely Release Unverified Plaintext in Authenticated Encryption.**

In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 105–125. Springer, 2014.

- [ACMT25] Gorjan Alagic, Joseph Carolan, Christian Majenz, and Saliha Tokat.  
**The Sponge Is Quantum Indifferentiable.**  
*In 66th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2025, Sydney, Australia, December 14-17, 2025, pages 2591–2630. IEEE, 2025.*
- [ADG<sup>+</sup>22] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg.  
**How to Abuse and Fix Authenticated Encryption Without Key Commitment.**  
*In Kevin R. B. Butler and Kurt Thomas, editors, 31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022, pages 3291–3308. USENIX Association, 2022.*

- [ADMV15] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche.  
**Security of Keyed Sponge Constructions Using a Modular Proof Approach.**  
In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 364–384. Springer, 2015.

- [BBC<sup>+</sup>23] Clémence Bouvier, Pierre Briaud, Pyrros Chaidos, Léo Perrin, Robin Salen, Vesselin Velichkov, and Danny Willems.  
**New Design Techniques for Efficient Arithmetization-Oriented Hash Functions: ttAnemoi Permutations and ttJive Compression Mode.**  
In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 507–539. Springer, 2023.
- [BDPV07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.  
**Sponge Functions.**  
Ecrypt Hash Workshop 2007, May 2007.

- [BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.  
**On the Indifferentiability of the Sponge Construction.**  
In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

- [BDPV11a] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.  
**Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications.**  
In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, 2011.
- [BDPV11b] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.  
**On the Security of the Keyed Sponge Construction.**  
Symmetric Key Encryption Workshop (SKEW 2011), 2011.

- [BDPV11c] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.  
**On the Security of the Keyed Sponge Construction.**  
Symmetric Key Encryption Workshop, February 2011.
- [BDPV12] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.  
**Permutation-based encryption, authentication and authenticated encryption.**  
Directions in Authenticated Ciphers, July 2012.

- [BH22] Mihir Bellare and Viet Tung Hoang.  
**Efficient Schemes for Committing Authenticated Encryption.**  
In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 845–875. Springer, 2022.

- [BHN<sup>+</sup>19] Xavier Bonnetain, Akinori Hosoyamada, María Naya-Plasencia, Yu Sasaki, and André Schrottenloher.

**Quantum attacks without superposition queries: The offline simon's algorithm.**

In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 552–583. Springer, 2019.

- [BN08]      Mihir Bellare and Chanathip Namprempre.  
**Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm.**  
*J. Cryptol.*, 21(4):469–491, 2008.
- [BR00]      Mihir Bellare and Phillip Rogaway.  
**Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography.**  
In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*, pages 317–330. Springer, 2000.

- [BR06] Mihir Bellare and Phillip Rogaway.  
**The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs.**  
In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

- [BSS22] Xavier Bonnetain, André Schrottenloher, and Ferdinand Sibleyras.  
**Beyond quadratic speedups in quantum attacks on symmetric schemes.**  
In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 315–344. Springer, 2022.

[BZD<sup>+</sup>16] Hanno Böck, Aaron Zauner, Sean Devlin, Juraj Somorovsky, and Philipp Jovanovic.

**Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS.**

In Natalie Silvanovich and Patrick Traynor, editors, *10th USENIX Workshop on Offensive Technologies, WOOT 16, Austin, TX, USA, August 8-9, 2016*. USENIX Association, 2016.

- [CCH<sup>+</sup>26] Donghoon Chang, Yu Long Chen, Yukihiro Hiraga, Kazuhiko Minematsu, Nicky Mouha, Yusuke Naito, Yu Sasaki, and Takeshi Sugawara.

**Key committing security of hctr2, revisited.**

In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology - CRYPTO 2026 - 46th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-20, 2026, Proceedings, Part VI*, volume 16805 of *Lecture Notes in Computer Science*, pages 98–130. Springer, 2026.

- [CDD<sup>+</sup>25] Yu Long Chen, Michael Davidson, Morris Dworkin, John Kelsey, Yu Sasaki, Meltem Sönmez Turan, Donghoon Chang, Nicky Mouha, and Alyssa Thompson.

**Requirements for cryptographic accordions.**

NIST IR 8552, National Institute of Standards and Technology, 2025.

- [CDH<sup>+</sup>12] Donghoon Chang, Morris Dworkin, Seokhie Hong, John Kelsey, and Mridul Nandi.

**A Keyed Sponge Construction with Pseudorandomness in the Standard Model.**

NIST SHA-3 Workshop, March 2012.

- [CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya.  
**Merkle-Damgård Revisited: How to Construct a Hash Function.**  
In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, 2005.
- [CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi.  
**The random oracle methodology, revisited.**  
*J. ACM*, 51(4):557–594, 2004.

- [CHB21] Paul Crowley, Nathan Huckleberry, and Eric Biggers.  
**Length-preserving encryption with HCTR2.**  
Cryptology ePrint Archive, Paper 2021/1441, 2021.
- [CHM<sup>+</sup>25] Yu Long Chen, Yukihiro Hiraga, Nicky Mouha, Yusuke Naito, Yu Sasaki, and Takeshi Sugawara.  
**Beyond-birthday-bound security with HCTR2: cascaded construction and tweak-based key derivation.**  
In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part I*, volume 16245 of *Lecture Notes in Computer Science*, pages 3–34. Springer, 2025.

- [CS14] Shan Chen and John P. Steinberger.  
**Tight Security Bounds for Key-Alternating Ciphers.**  
In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 327–350. Springer, 2014.

[Dam89] Ivan Damgård.

**A Design Principle for Hash Functions.**

In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.

- [DFG23] Jean Paul Degabriele, Marc Fischlin, and Jérôme Govinden.  
**The Indifferentiability of the Duplex and Its Practical Applications.**  
In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part VIII*, volume 14445 of *Lecture Notes in Computer Science*, pages 237–269. Springer, 2023.

- [DGRW18] Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. **Fast message franking: From invisible salamanders to encryptment.** In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 155–186. Springer, 2018.

- [DKS12] Orr Dunkelman, Nathan Keller, and Adi Shamir.  
**Minimalism in cryptography: The even-mansour scheme revisited.**  
In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 336–354. Springer, 2012.

- [DM24] Christoph Dobraunig and Bart Mennink.  
**Generalized Initialization of the Duplex Construction.**  
In Christina Pöpper and Lejla Batina, editors, *Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part II*, volume 14584 of *Lecture Notes in Computer Science*, pages 460–484. Springer, 2024.

[DMMT25] Christoph Dobraunig, Krystian Matusiewicz, Bart Mennink, and Alexander Tereschenko.

**Efficient instances of docked double decker with aes, and application to authenticated encryption.**

In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part I*, volume 15601 of *Lecture Notes in Computer Science*, pages 62–92. Springer, 2025.

- [DMV17] Joan Daemen, Bart Mennink, and Gilles Van Assche.  
**Full-State Keyed Duplex with Built-In Multi-user Support.**  
In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 606–637. Springer, 2017.
- [Dwo07] Morris Dworkin.  
**Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.**  
NIST SP 800-38D, 2007.  
<https://doi.org/10.6028/NIST.SP.800-38D>.

- [EM91] Shimon Even and Yishay Mansour.  
**A Construction of a Cipher From a Single Pseudorandom Permutation.**  
In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors,  
*Advances in Cryptology - ASIACRYPT '91, International Conference on  
the Theory and Applications of Cryptology, Fujiyoshida, Japan, November  
11-14, 1991, Proceedings*, volume 739 of *Lecture Notes in Computer  
Science*, pages 210–224. Springer, 1991.
- [EM97] Shimon Even and Yishay Mansour.  
**A Construction of a Cipher from a Single Pseudorandom Permutation.**  
*J. Cryptol.*, 10(3):151–162, 1997.

- [GKL<sup>+</sup>22] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch.

**Reinforced Concrete: A Fast Hash Function for Verifiable Computation.**

In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 1323–1335. ACM, 2022.

[GKR<sup>+</sup>21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger.

**Poseidon: A New Hash Function for Zero-Knowledge Proof Systems.**

In Michael D. Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 519–535. USENIX Association, 2021.

- [GL15] Shay Gueron and Yehuda Lindell.  
**GCM-SIV: full nonce misuse-resistant authenticated encryption at under one cycle per byte.**  
In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 109–119. ACM, 2015.

- [GPT15] Peter Gazi, Krzysztof Pietrzak, and Stefano Tessaro.  
**The Exact PRF Security of Truncation: Tight Bounds for Keyed Sponges and Truncated CBC.**  
In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 368–387. Springer, 2015.
- [IS17] Tetsu Iwata and Yannick Seurin.  
**Reconsidering the security bound of AES-GCM-SIV.**  
*IACR Trans. Symmetric Cryptol.*, 2017(4):240–267, 2017.

- [KM10] Hidenori Kuwakado and Masakatu Morii.  
**Quantum distinguisher between the 3-round feistel cipher and the random permutation.**  
In *IEEE International Symposium on Information Theory, ISIT 2010, June 13-18, 2010, Austin, Texas, USA, Proceedings*, pages 2682–2685. IEEE, 2010.
- [KM12] Hidenori Kuwakado and Masakatu Morii.  
**Security on the quantum-type even-mansour cipher.**  
In *Proceedings of the International Symposium on Information Theory and its Applications, ISITA 2012, Honolulu, HI, USA, October 28-31, 2012*, pages 312–316. IEEE, 2012.

- [LB25] Charlotte Lefevre and Mario Marhuenda Beltrán.  
**MacaKey: Full-State Keyed Sponge Meets the Summation-Truncation Hybrid.**  
*IACR Cryptol. ePrint Arch.*, 2025:893, 2025.
- [LM22] Charlotte Lefevre and Bart Mennink.  
**Tight Preimage Resistance of the Sponge Construction.**  
In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 185–204. Springer, 2022.

- [LR88] Michael Luby and Charles Rackoff.  
**How to construct pseudorandom permutations from pseudorandom functions.**  
*SIAM J. Comput.*, 17(2):373–386, 1988.
- [MD24] Nicky Mouha and Morris Dworkin.  
**Report on the block cipher modes of operation in the NIST SP 800-38 series.**  
NIST IR 8459, National Institute of Standards and Technology, 2024.
- [Men18] Bart Mennink.  
**Key Prediction Security of Keyed Sponges.**  
*IACR Trans. Symmetric Cryptol.*, 2018(4):128–149, 2018.

- [Men23] Bart Mennink.  
**Understanding the Duplex and Its Security.**  
*IACR Trans. Symmetric Cryptol.*, 2023(2):1–46, 2023.
- [Mer89] Ralph C. Merkle.  
**One Way Hash Functions and DES.**  
In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.

- [MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein.  
**Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology.**  
In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.
- [Nat24] National Institute of Standards and Technology.  
**Pre-draft call for comments: NIST proposes to standardize a wider variant of AES (SP 800-197).**  
<https://csrc.nist.gov/pubs/sp/800/197/iprd>, 2024.

- [Nat25] National Institute of Standards and Technology.  
**Pre-draft call for comments: NIST launches development of cryptographic accordions (SP 800-197a).**  
<https://csrc.nist.gov/pubs/sp/800/197/a/iprd>, 2025.
- [NOMI15] Yuichi Niwa, Keisuke Ohashi, Kazuhiko Minematsu, and Tetsu Iwata.  
**GCM Security Bounds Reconsidered.**  
In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 385–407. Springer, 2015.

- [NR99] Moni Naor and Omer Reingold.  
**On the construction of pseudorandom permutations: Luby-rackoff revisited.**  
*J. Cryptol.*, 12(1):29–66, 1999.
- [NY16] Yusuke Naito and Kan Yasuda.  
**New Bounds for Keyed Sponges with Extendable Output: Independence Between Capacity and Message Length.**  
In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2016.

- [Pat08] Jacques Patarin.  
**The “Coefficients H” Technique.**  
In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors,  
*Selected Areas in Cryptography, 15th International Workshop, SAC 2008,*  
*Sackville, New Brunswick, Canada, August 14-15, Revised Selected*  
*Papers*, volume 5381 of *Lecture Notes in Computer Science*, pages  
328–345. Springer, 2008.

[Rog06]

Phillip Rogaway.

**Formalizing human ignorance.**

In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 2006, First International Conference on Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers*, volume 4341 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2006.

- [RS04] Phillip Rogaway and Thomas Shrimpton.  
**Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance.**  
In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004.

- [RS06] Phillip Rogaway and Thomas Shrimpton.  
**A Provable-Security Treatment of the Key-Wrap Problem.**  
In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, 2006.

- [RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton.  
**Careful with Composition: Limitations of the Indifferentiability Framework.**

In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, 2011.

- [SMKK25] Meltem Sönmez Turan, Kerry A. McKay, Jinkeon Kang, and John Kelsey.  
**Ascon-Based Lightweight Cryptography Standards for Constrained Devices.**  
NIST SP 800-232, August 2025.  
<https://doi.org/10.6028/NIST.SP.800-232>.
- [VP17] Mathy Vanhoef and Frank Piessens.  
**Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2.**  
In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1313–1328. ACM, 2017.

- [WFW05] Peng Wang, Dengguo Feng, and Wenling Wu.  
**HCTR: A variable-input-length enciphering mode.**  
In Dengguo Feng, Dongdai Lin, and Moti Yung, editors, *Information Security and Cryptology, First SKLOIS Conference, CISC 2005, Beijing, China, December 15-17, 2005, Proceedings*, volume 3822 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2005.