

Shaking up authenticated encryption

Joan Daemen¹ Seth Hoffert
Silvia Mella¹ Gilles Van Assche² Ronny Van Keer²

¹Radboud University

²STMicroelectronics

End-of-ESCADA Workshop
Nijmegen, The Netherlands, August 2024

What we want to do

Define concrete authenticated encryption schemes on top of Keccak

- with security reduction to (Turbo)SHAKE
- with robustness properties
 - nonce-based or nonce-misuse-resistant
 - committing
- with user-friendly features
 - nonce requirement on AD instead of dedicated short nonce field
 - session-supporting for online operation, intermediate MACs

What we want to do

Define concrete authenticated encryption schemes on top of Keccak

- with security reduction to (Turbo)SHAKE
- with robustness properties
 - nonce-based or nonce-misuse-resistant
 - committing
- with user-friendly features
 - nonce requirement on AD instead of dedicated short nonce field
 - session-supporting for online operation, intermediate MACs

What we want to do

Define concrete authenticated encryption schemes on top of Keccak

- with security reduction to (Turbo)SHAKE
- with robustness properties
 - nonce-based or nonce-misuse-resistant
 - committing
- with user-friendly features
 - nonce requirement on AD instead of dedicated short nonce field
 - session-supporting for online operation, intermediate MACs

What we want to do

Define concrete authenticated encryption schemes on top of Keccak

- with security reduction to (Turbo)SHAKE
- with robustness properties
 - nonce-based or nonce-misuse-resistant
 - committing
- with user-friendly features
 - nonce requirement on AD instead of dedicated short nonce field
 - session-supporting for online operation, intermediate MACs

What is authenticated encryption

Authenticated encryption

- encryption (wrap) takes (K, AD, P) and returns $C||T$, with AD a nonce
- decryption (unwrap) takes $(K, AD, C||T)$ and returns P or error $\perp$

Ideally

- C looks random for each input
- unwrap of invalid ciphertext fails

What is authenticated encryption

Authenticated encryption

- encryption (wrap) takes (K, AD, P) and returns $C||T$, with AD a nonce
- decryption (unwrap) takes $(K, AD, C||T)$ and returns P or error $\perp$

Ideally

- C looks random for each input
- unwrap of invalid ciphertext fails

What is the nonce requirement in authenticated encryption

Authenticated encryption

- encryption (wrap) takes (K, AD, P) and returns $C||T$, **with AD a nonce**
- decryption (unwrap) takes $(K, AD, C||T)$ and returns P or error $\perp$

Nonce-based authenticated encryption

- AD must be **different** at each wrap (for a given key and a given session)
- otherwise, **no security is claimed**

Nonce misuse resistant authenticated encryption

- AD must be **different** at each wrap (for a given key and a given session)
- otherwise, **equality of plaintexts** can be detected, but **nothing more**

What is the nonce requirement in authenticated encryption

Authenticated encryption

- encryption (wrap) takes (K, AD, P) and returns $C||T$, **with AD a nonce**
- decryption (unwrap) takes $(K, AD, C||T)$ and returns P or error $\perp$

Nonce-based authenticated encryption

- AD must be **different** at each wrap (for a given key and a given session)
- otherwise, **no security is claimed**

Nonce misuse resistant authenticated encryption

- AD must be **different** at each wrap (for a given key and a given session)
- otherwise, **equality of plaintexts** can be detected, but **nothing more**

What is the nonce requirement in authenticated encryption

Authenticated encryption

- encryption (wrap) takes (K, AD, P) and returns $C||T$, **with AD a nonce**
- decryption (unwrap) takes $(K, AD, C||T)$ and returns P or error $\perp$

Nonce-based authenticated encryption

- AD must be **different** at each wrap (for a given key and a given session)
- otherwise, **no security is claimed**

Nonce misuse resistant authenticated encryption

- AD must be **different** at each wrap (for a given key and a given session)
- otherwise, **equality of plaintexts** can be detected, but **nothing more**

What is committing authenticated encryption

Fully committing authenticated encryption

- encryption (wrap) takes (K, AD, P) and returns $C||T$, with AD a nonce
- decryption (unwrap) takes $(K, AD, C||T)$ and returns P or error $\perp$
- **hard to generate collisions** in $C||T$ by an attacker with free choice of (K, AD, P)

⇒ One way is to make T a **hash** of (K, AD, P) .

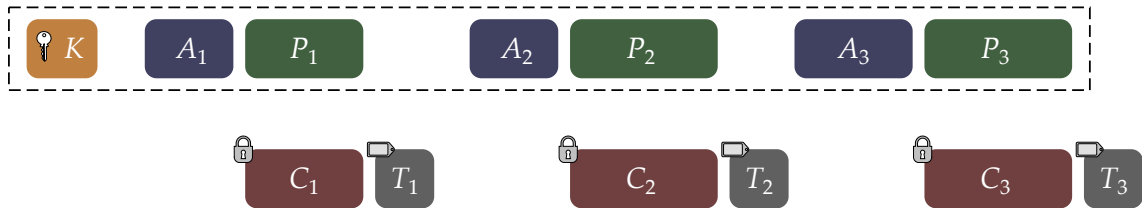
What is committing authenticated encryption

Fully committing authenticated encryption

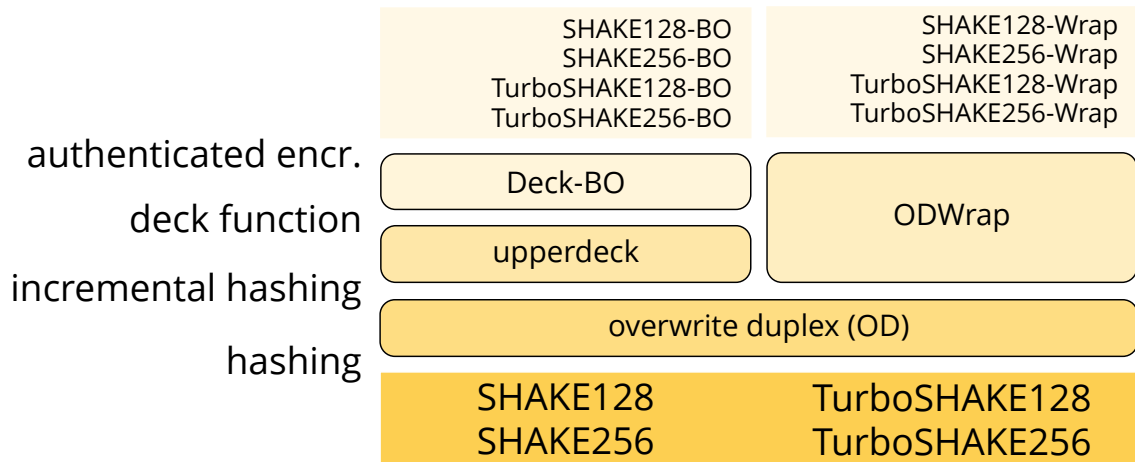
- encryption (wrap) takes (K, AD, P) and returns $C||T$, with AD a nonce
- decryption (unwrap) takes $(K, AD, C||T)$ and returns P or error $\perp$
- **hard to generate collisions** in $C||T$ by an attacker with free choice of (K, AD, P)

⇒ One way is to make T a **hash** of (K, AD, P) .

What are sessions



How we want to do it



Hashing

	SHAKE128-BO SHAKE256-BO TurboSHAKE128-BO TurboSHAKE256-BO	SHAKE128-Wrap SHAKE256-Wrap TurboSHAKE128-Wrap TurboSHAKE256-Wrap
authenticated encr.	Deck-BO	ODWrap
deck function	upperdeck	
incremental hashing	overwrite duplex (OD)	
hashing	SHAKE128 SHAKE256	TurboSHAKE128 TurboSHAKE256

SHAKE and TurboSHAKE

SHAKE128 and SHAKE256 [FIPS 202]

- Sponge with Keccak- p [**24 rounds**] [Bertoni et al., 2008]
- 15 years of public scrutiny $\Rightarrow$ 12 rounds give comfortable safety margin

TurboSHAKE128 and TurboSHAKE256

- Sponge with Keccak- p [**12 rounds**] [Bertoni et al., ePrint 2023/342]
- Same public scrutiny applies as all cryptanalysis is on reduced-round versions

Security of (Turbo)SHAKE

- Unkeyed: flat sponge claim with security strength 128/256
- Keyed:
 - When input to (Turbo)SHAKE is prefixed with a secret key K ...
 - ... it is hard to distinguish from a random oracle

SHAKE and TurboSHAKE

SHAKE128 and SHAKE256 [FIPS 202]

- Sponge with Keccak- p [**24 rounds**] [Bertoni et al., 2008]
- 15 years of public scrutiny $\Rightarrow$ 12 rounds give comfortable safety margin

TurboSHAKE128 and TurboSHAKE256

- Sponge with Keccak- p [**12 rounds**] [Bertoni et al., ePrint 2023/342]
- Same public scrutiny applies as all cryptanalysis is on reduced-round versions

Security of (Turbo)SHAKE

- Unkeyed: flat sponge claim with security strength 128/256
- Keyed:
 - When input to (Turbo)SHAKE is prefixed with a secret key K ...
 - ... it is hard to distinguish from a random oracle

SHAKE and TurboSHAKE

SHAKE128 and SHAKE256 [FIPS 202]

- Sponge with Keccak- p [**24 rounds**] [Bertoni et al., 2008]
- 15 years of public scrutiny $\Rightarrow$ 12 rounds give comfortable safety margin

TurboSHAKE128 and TurboSHAKE256

- Sponge with Keccak- p [**12 rounds**] [Bertoni et al., ePrint 2023/342]
- Same public scrutiny applies as all cryptanalysis is on reduced-round versions

Security of (Turbo)SHAKE

- Unkeyed: flat sponge claim with security strength 128/256
- Keyed:
 - When input to (Turbo)SHAKE is prefixed with a secret key K ...
 - ... it is hard to distinguish from a random oracle

SHAKE and TurboSHAKE

SHAKE128 and SHAKE256 [FIPS 202]

- Sponge with Keccak- p [**24 rounds**] [Bertoni et al., 2008]
- 15 years of public scrutiny $\Rightarrow$ 12 rounds give comfortable safety margin

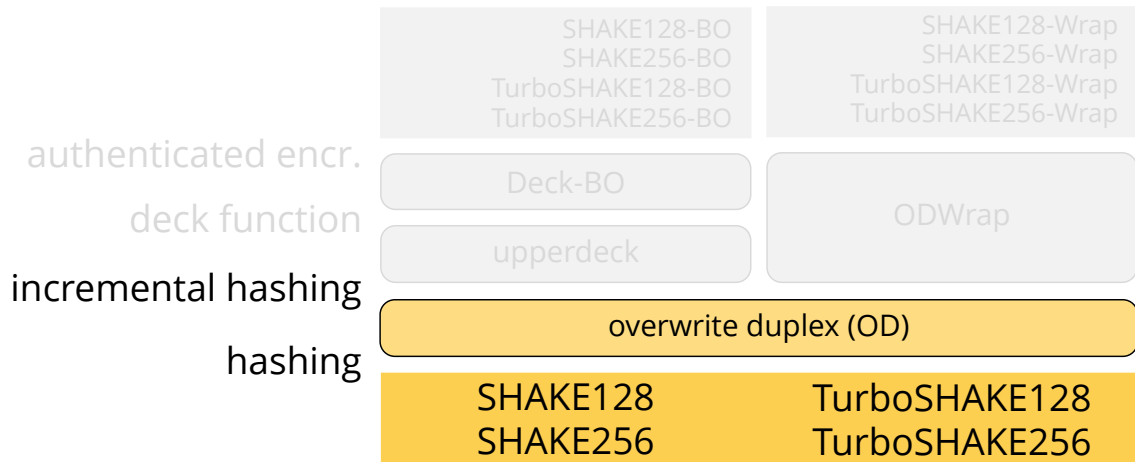
TurboSHAKE128 and TurboSHAKE256

- Sponge with Keccak- p [**12 rounds**] [Bertoni et al., ePrint 2023/342]
- Same public scrutiny applies as all cryptanalysis is on reduced-round versions

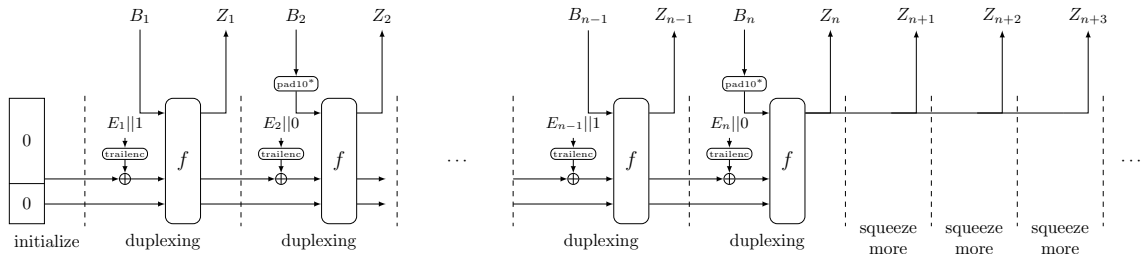
Security of (Turbo)SHAKE

- Unkeyed: flat sponge claim with security strength 128/256
- Keyed:
 - When input to (Turbo)SHAKE is prefixed with a secret key K ...
 - ... it is hard to distinguish from a random oracle

Incremental hashing



The OD construction

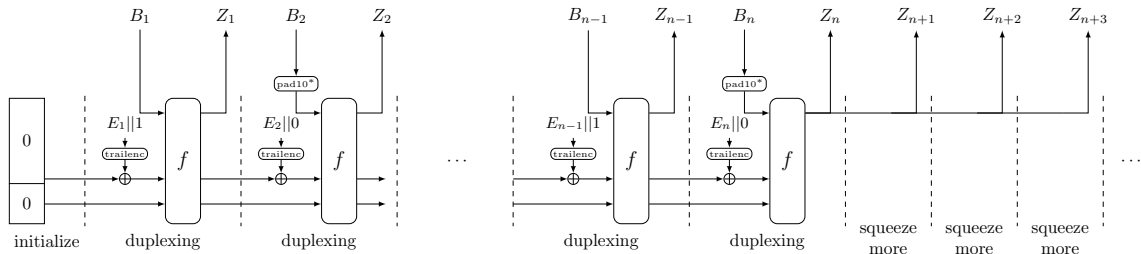


Sequence of $Z \leftarrow \text{OD.duplexing}(B, E, \ell)$, with

- B : a byte string that fits in a block
- E : a few trailing bits represented as an integer
- ℓ : the requested length, up to one block

$\Rightarrow Z$ is a cryptographic hash of $(B_1; E_1; B_2; E_2; \dots)$ received so far!

The OD construction



Sequence of $Z \leftarrow \text{OD.duplexing}(B, E, \ell)$, with

- B : a byte string that fits in a block
- E : a few trailing bits represented as an integer
- ℓ : the requested length, up to one block

$\Rightarrow Z$ is a cryptographic hash of $(B_1; E_1; B_2; E_2; \dots)$ received so far!

Intermezzo: trailing bits

Short bit string $e = (e_0, e_1, \dots, e_{n-1})$
mapped to an integer:

$$E = \text{padint}(e) = 2^n + \sum_{i=0}^{n-1} 2^i e_i$$

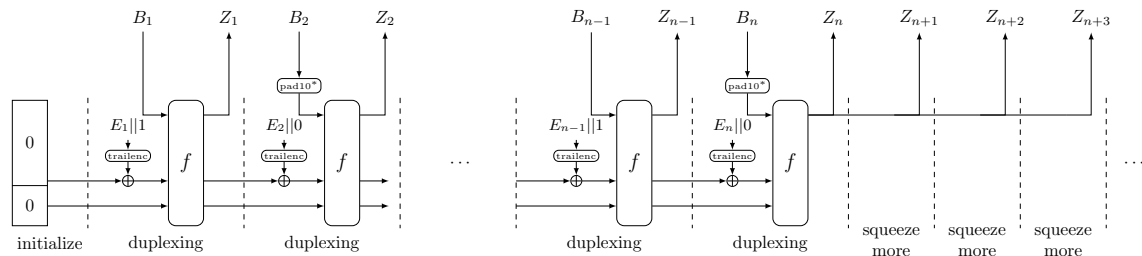
To prepend a bit:

$$e \leftarrow 0 \parallel e \Leftrightarrow E \leftarrow 2E$$

$$e \leftarrow 1 \parallel e \Leftrightarrow E \leftarrow 2E + 1$$

ϵ	1
0	2
1	3
00	4
10	5
01	6
11	7
000	8
100	9
...	...
1111111	255

Reduction of OD to the underlying sponge function



Lemma 1

Let F be a (Turbo)SHAKE instance with permutation f and capacity c . The output of the n -th duplexing call to $\text{od}[f, \rho, c]$ is the output of F applied to an input that is an injective mapping of $(B_1; E_1; \dots; B_n; E_n)$. And the input to F has B_1 as a prefix.

Concrete consequences

If (Turbo)SHAKE stands by its security claim, then ...

- Z is a collision-resistant hash of $(B_1; E_1; B_2; E_2; \dots)$, and
- (Turbo)SHAKE($K \parallel \dots$) is a PRF.

If we do OD.duplexing($K, E = 1, \ell = 0$) just after initialization, then ...

- Z is a PRF of $(B_2; E_2; B_3; E_3; \dots)$.

Concrete consequences

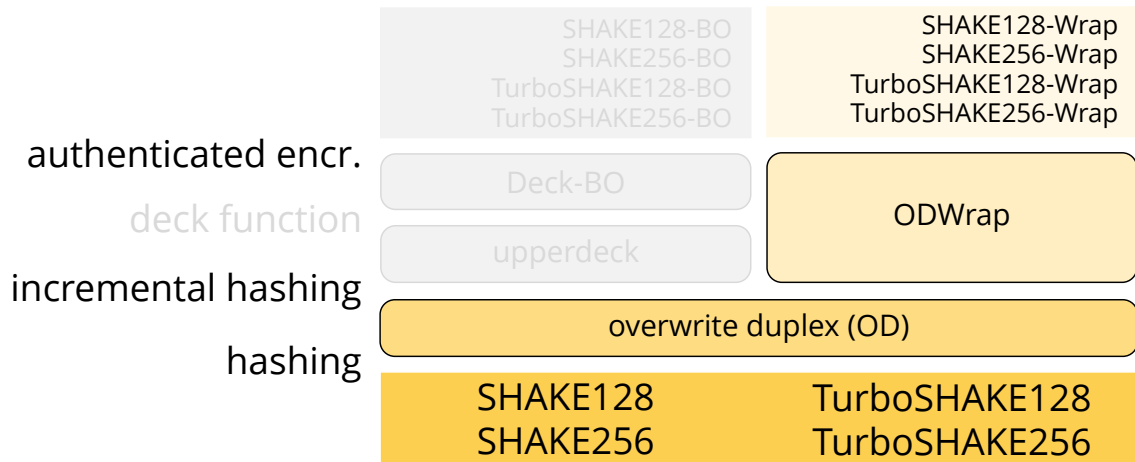
If (Turbo)SHAKE stands by its security claim, then ...

- Z is a collision-resistant hash of $(B_1; E_1; B_2; E_2; \dots)$, and
- $(\text{Turbo})\text{SHAKE}(K \parallel \dots)$ is a PRF.

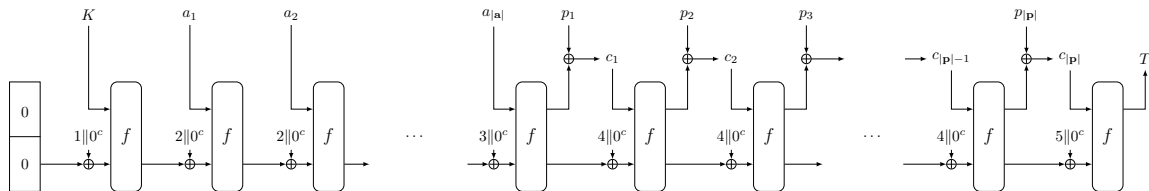
If we do OD.duplexing($K, E = 1, \ell = 0$) just after initialization, then ...

- Z is a PRF of $(B_2; E_2; B_3; E_3; \dots)$.

Duplex-based authenticated encryption

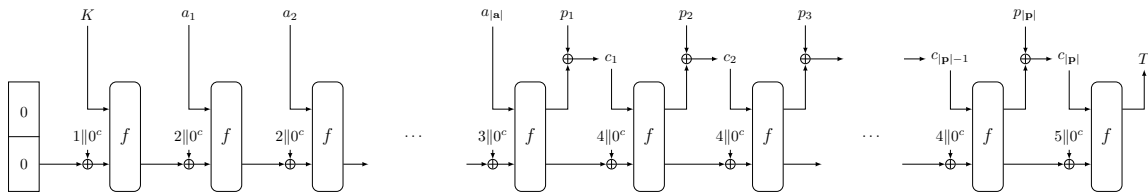


(Turbo)SHAKE-Wrap: nonce-based AE with sessions



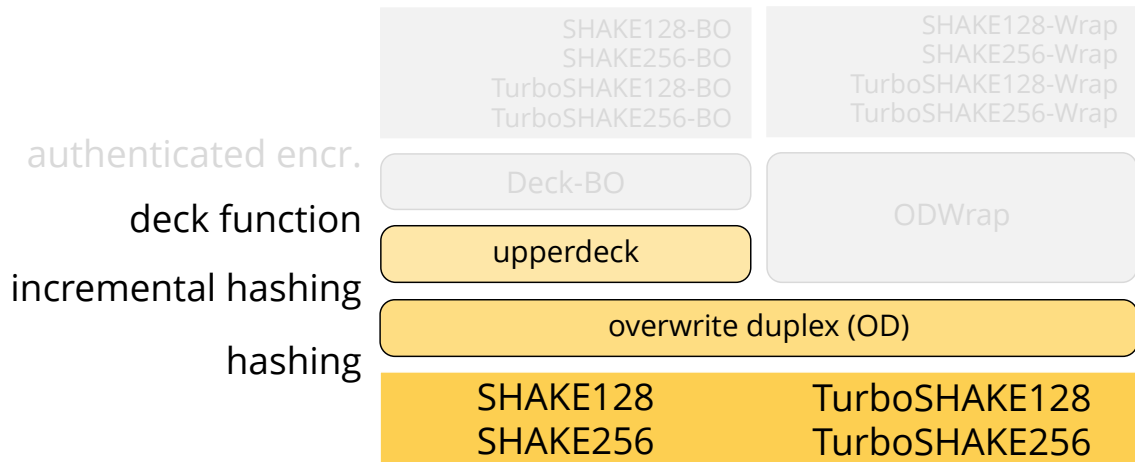
- **ODWrap** $\circ$ **OD**: nonce-based session-supporting AE mode
 - duplex-based, with 1 domain separation byte per f call
- Confidentiality and integrity $\Leftarrow$ PRF security of (Turbo)SHAKE
- Committing security $\Leftarrow$ collision resistance of (Turbo)SHAKE

(Turbo)SHAKE-Wrap: nonce-based AE with sessions



- **ODWrap** $\circ$ **OD**: nonce-based session-supporting AE mode
 - duplex-based, with 1 domain separation byte per f call
- Confidentiality and integrity $\Leftarrow$ PRF security of (Turbo)SHAKE
- Committing security $\Leftarrow$ collision resistance of (Turbo)SHAKE

Deck functions



Definition of a deck function

A deck function F_K

$$Z = 0^n + F_K(X^{(1)}; \dots; X^{(m)}) \ll q$$

doubly extendable cryptographic keyed function

Definition of a deck function

A deck function F_K

$$Z = 0^n + F_K(X^{(1)}; \dots; X^{(m)}) \ll q$$

■ Input: sequence of strings $X^{(1)}; \dots; X^{(m)}$

Definition of a deck function

A deck function F_K

$$Z = 0^n + F_K(X^{(1)}; \dots; X^{(m)}) \ll q$$

- Input: sequence of strings $X^{(1)}; \dots; X^{(m)}$
- Output: arbitrary length
 - **pseudo-random function of the input**
 - taking n bits starting from offset q
- Security model: shall be hard to distinguish from a random oracle

Definition of a deck function

A deck function F_K

$$Z = 0^n + F_K(X^{(1)}; \dots; X^{(m)}) \ll q$$

Efficient incrementality

- Extendable input

- 1 Compute $F_K(X)$
- 2 Compute $F_K(X; Y)$: cost independent of X

Definition of a deck function

A deck function F_K

$$Z = 0^n + F_K(X^{(1)}; \dots; X^{(m)}) \ll q$$

Efficient incrementality

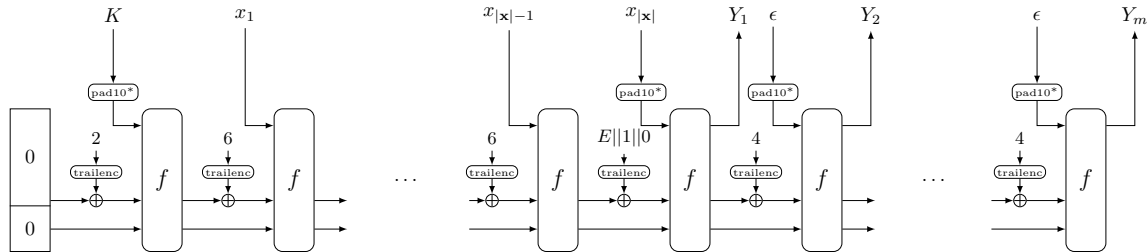
■ Extendable input

- 1 Compute $F_K(X)$
- 2 Compute $F_K(X; Y)$: cost independent of X

■ Extendable output

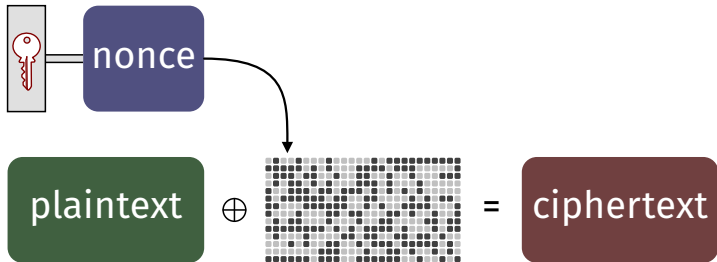
- 1 Request n_1 bits from offset 0
- 2 Request n_2 bits from offset n_1 : cost independent of n_1

The upperdeck construction



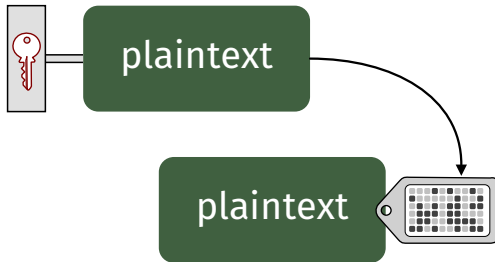
$$Y \leftarrow F_K(X||e) \quad \text{with } E = \text{padint}(e)$$

Stream encryption: short input, long output



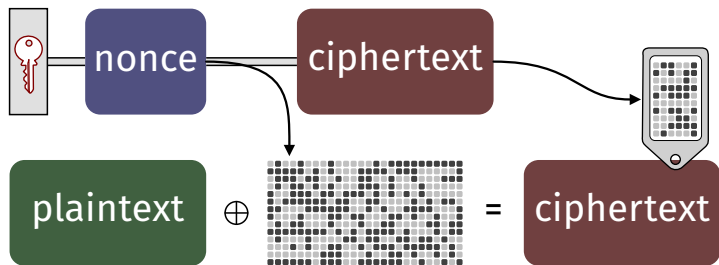
$$C \leftarrow P + F_K(\text{nonce})$$

MAC computation: long input, short output



$$T \leftarrow \mathbf{0}^t + F_K(P)$$

Authenticated encryption



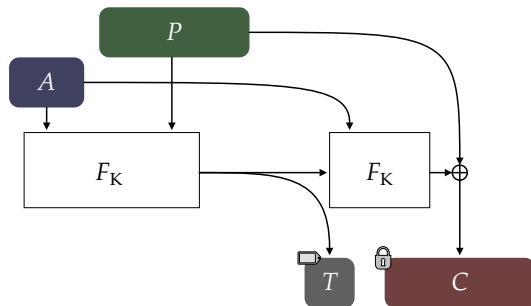
Wrap:

$$\begin{aligned} C &\leftarrow P + F_K(\text{nonce}) \\ T &\leftarrow 0^t + F_K(\text{nonce}; C) \\ \textbf{return } C \parallel T \end{aligned}$$

Unwrap:

$$\begin{aligned} T &\stackrel{?}{=} 0^t + F_K(\text{nonce}; C) \\ P &\leftarrow C + F_K(\text{nonce}) \\ \textbf{return } P \text{ (or } \perp) \end{aligned}$$

SIV-type authenticated encryption



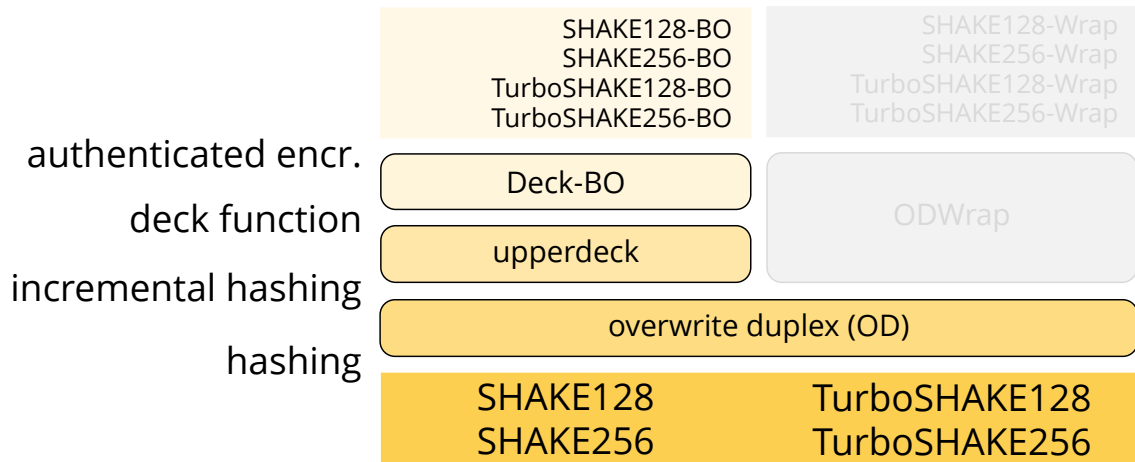
Wrap:

$T \leftarrow 0^t + F_K(AD; P || 0)$
 $C \leftarrow P + F_K(AD; T || 1)$
return $C || T$

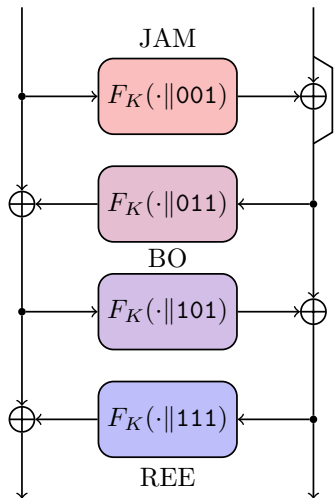
Unwrap:

$P \leftarrow C + F_K(AD; T || 1)$
 $T \stackrel{?}{=} 0^t + F_K(AD; P || 0)$
return P (or $\perp$)

Deck-based authenticated encryption

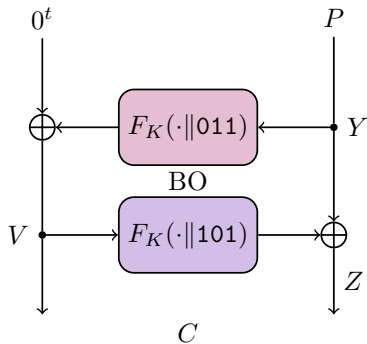


Deck-[JAM]BO[REE]: Feistel network



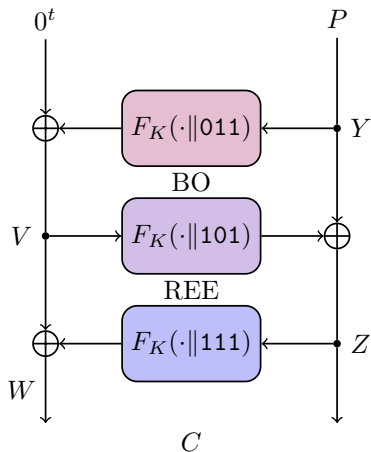
[Băcuiș, Daemen, Hoffert, Van Assche, Van Keer, Jammin' on the deck, Asiacrypt 2022]

Deck-BO



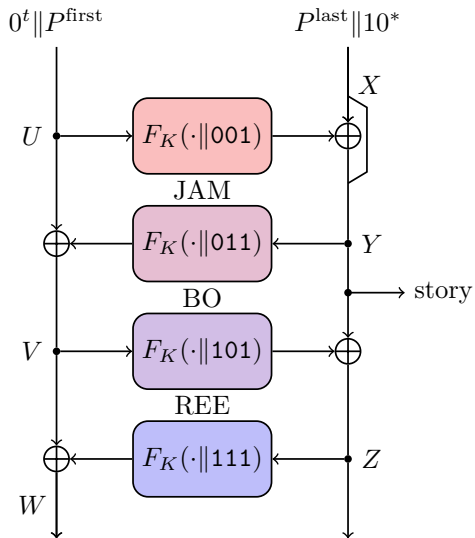
SIV [Rogaway and Shrimpton, EC 2006]
+ support for AD and sessions

Deck-BOREE



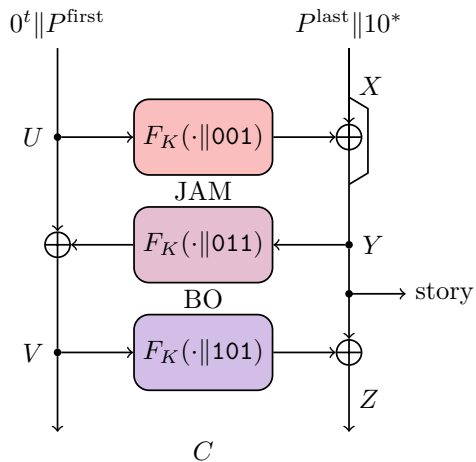
RIV [Abed et al., FSE 2016]
+ support for sessions

Deck-JAMBOREE



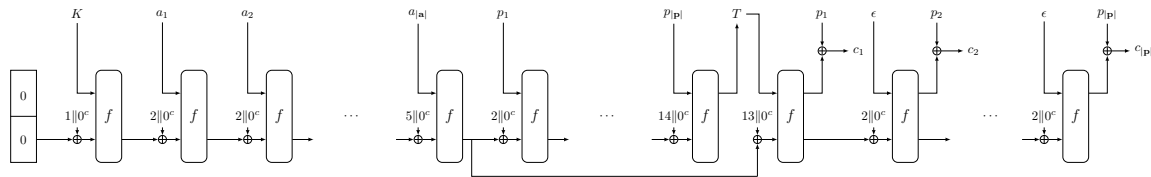
Robust AE [Hoang et al., EC 2015]
+ support for sessions

Deck-JAMBO



SIV with optimal redundancy
(but not RUP resistance)

(Turbo)SHAKE-BO: SIV-type AE with sessions



- **Deck-BO** ◦ **upperdeck** ◦ **OD**: SIV-type session-supporting AE mode
 - duplex-based, with 1 domain separation byte per f call
- Confidentiality and integrity $\Leftarrow$ PRF security of (Turbo)SHAKE
- Committing security $\Leftarrow$ collision resistance of (Turbo)SHAKE

Conclusions

Two approaches for **committing** and **session-supporting** AE with (Turbo)SHAKE:

- performance of duplex-based mode
- robustness and flexibility of deck-based modes
 - See also [nonce-encrypting](#) modes [Hoffert, ePrint 2022/1711]

And **simplicity** of the modes once the layers are merged

Thanks for your attention!

Conclusions

Two approaches for **committing** and **session-supporting** AE with (Turbo)SHAKE:

- performance of duplex-based mode
- robustness and flexibility of deck-based modes
 - See also [nonce-encrypting](#) modes [Hoffert, ePrint 2022/1711]

And **simplicity** of the modes once the layers are merged

Thanks for your attention!