SHAKE modes of operation

Joan Daemen[1] Seth Hoffert Silvia Mella[1] Gilles Van Assche[2]

[1]Radboud University
[2]STMicroelectronics

3rd NIST Workshop on Block Cipher Modes of Operation
Rockville, USA, October 3-4, 2023

Outline

(committing) authenticated encryption

- Authenticated encryption:
  - encryption (wrap) takes $(K, [N,], AD, P)$ and returns C (and tag T)
  - decryption (unwrap) takes $(K, [N,], AD, C[, T])$ and returns P or error $\perp$
- Ideally, C looks random for each input and unwrap of invalid ciphertext fails
- Some applications require collision-resistance of wrapping even if the key is known
- This is called *committing* AE
- In some cases a weaker property may be sufficient
- We propose committing AE schemes that have as tag the SHAKE hash of an injective encoding of $(K, [N,] AD, P)$

(committing) authenticated encryption

- Authenticated encryption:
  - encryption (wrap) takes $(K, [N,], AD, P)$ and returns C (and tag T)
  - decryption (unwrap) takes $(K, [N,], AD, C[, T])$ and returns P or error $\perp$
- Ideally, C looks random for each input and unwrap of invalid ciphertext fails
- Some applications require collision-resistance of wrapping even if the key is known
- This is called *committing* AE
- In some cases a weaker property may be sufficient
- We propose committing AE schemes that have as tag the SHAKE hash of an injective encoding of $(K, [N,] AD, P)$

(committing) authenticated encryption

- Authenticated encryption:
  - encryption (wrap) takes $(K, [N,], AD, P)$ and returns C (and tag T)
  - decryption (unwrap) takes $(K, [N,], AD, C[, T])$ and returns P or error $\perp$
- Ideally, C looks random for each input and unwrap of invalid ciphertext fails
- Some applications require collision-resistance of wrapping even if the key is known
- This is called *committing* AE
- In some cases a weaker property may be sufficient
- We propose committing AE schemes that have as tag the SHAKE hash of an injective encoding of $(K, [N,] AD, P)$

(committing) authenticated encryption

- Authenticated encryption:
  - encryption (wrap) takes $(K, [N,], AD, P)$ and returns C (and tag T)
  - decryption (unwrap) takes $(K, [N,], AD, C[, T])$ and returns P or error $\perp$
- Ideally, C looks random for each input and unwrap of invalid ciphertext fails
- Some applications require collision-resistance of wrapping even if the key is known
- This is called *committing* AE
- In some cases a weaker property may be sufficient
- We propose committing AE schemes that have as tag the SHAKE hash of an injective encoding of $(K, [N,] AD, P)$

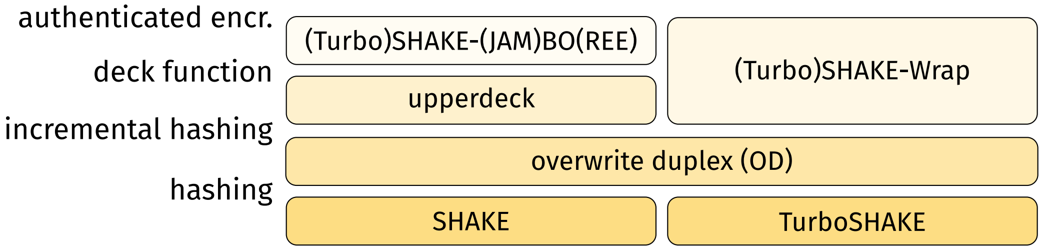
(committing) authenticated encryption

- Authenticated encryption:
  - encryption (wrap) takes $(K, [N,], AD, P)$ and returns C (and tag T)
  - decryption (unwrap) takes $(K, [N,], AD, C[, T])$ and returns P or error $\perp$
- Ideally, C looks random for each input and unwrap of invalid ciphertext fails
- Some applications require collision-resistance of wrapping even if the key is known
- This is called *committing* AE
- In some cases a weaker property may be sufficient
- We propose committing AE schemes that have as tag the SHAKE hash of an injective encoding of $(K, [N,] AD, P)$

(committing) authenticated encryption

- Authenticated encryption:
  - encryption (wrap) takes $(K, [N,], \mathrm{AD}, P)$ and returns C (and tag T)
  - decryption (unwrap) takes $(K, [N,], \mathrm{AD}, C[, T])$ and returns P or error $\perp$
- Ideally, C looks random for each input and unwrap of invalid ciphertext fails
- Some applications require collision-resistance of wrapping even if the key is known
- This is called *committing* AE
- In some cases a weaker property may be sufficient
- We propose committing AE schemes that have as tag the SHAKE hash of an injective encoding of $(K, [N,]\, \mathrm{AD}, P)$

Committing authenticated encryption based on (Turbo)SHAKE



For the paper, see `https://eprint.iacr.org/2023/1494`

SHAKE and TurboSHAKE

SHAKE

- FIPS 202 specifies two XOFs: SHAKE128 and SHAKE256
- Based on KECCAK: sponge with KECCAK-$p[24 \text{ rounds}]$ [Bertoni et al., 2008]
- 15 years of public scrutiny $\Rightarrow$ 12 rounds give comfortable safety margin

TurboSHAKE

- Sponge with KECCAK-$p[12 \text{ rounds}]$ [Bertoni et al., ePrint 2023/342]
- Same public scrutiny applies as all cryptanalysis is on reduced-round versions

Security:

- 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

SHAKE

- FIPS 202 specifies two XOFs: SHAKE128 and SHAKE256
- Based on KECCAK: sponge with KECCAK-p[24 rounds] [Bertoni et al., 2008]
- 15 years of public scrutiny $\Rightarrow$ 12 rounds give comfortable safety margin

TurboSHAKE

- 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:

- 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

SHAKE

- FIPS 202 specifies two XOFs: SHAKE128 and SHAKE256
- Based on KECCAK: sponge with KECCAK-$p[24 \text{ rounds}]$ [Bertoni et al., 2008]
- 15 years of public scrutiny ⇒ 12 rounds give comfortable safety margin

TurboSHAKE

- Sponge with KECCAK-$p[12 \text{ rounds}]$ [Bertoni et al., ePrint 2023/342]
- Same public scrutiny applies as all cryptanalysis is on reduced-round versions

Security:

- 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

SHAKE

- FIPS 202 specifies two XOFs: SHAKE128 and SHAKE256
- Based on KECCAK: sponge with KECCAK-p[24 rounds] [Bertoni et al., 2008]
- 15 years of public scrutiny ⇒ 12 rounds give comfortable safety margin

TurboSHAKE

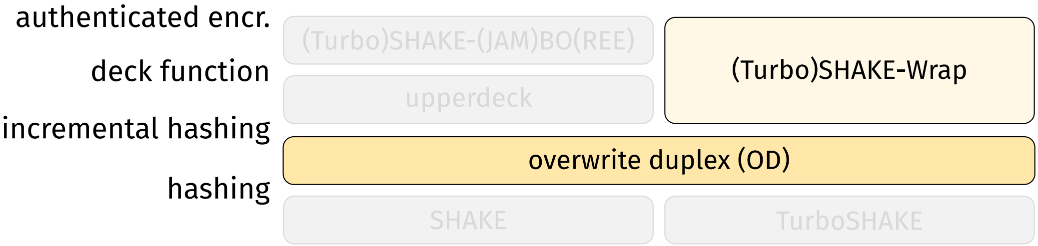
- Sponge with KECCAK-p[12 rounds] [Bertoni et al., ePrint 2023/342]
- Same public scrutiny applies as all cryptanalysis is on reduced-round versions

Security:

- 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

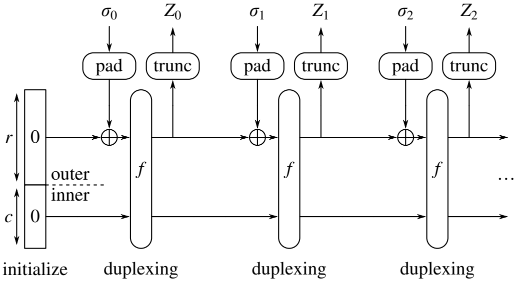
Outline

Duplex-based approach



Overwrite Duplex (OD)

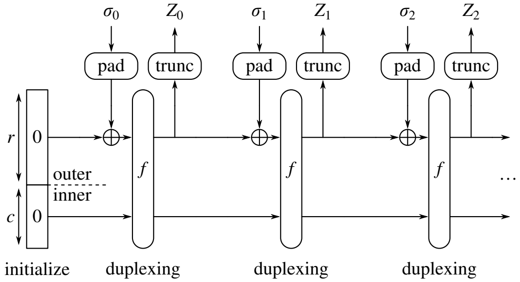
Duplex [Bertoni et al., SAC 2011]:



- Security of duplex equivalent to sponge
- Security of outer-keyed duplex equivalent to keyed sponge
- Overwrite Duplex (OD): variant where bulk of input σ overwrites state

Overwrite Duplex (OD)

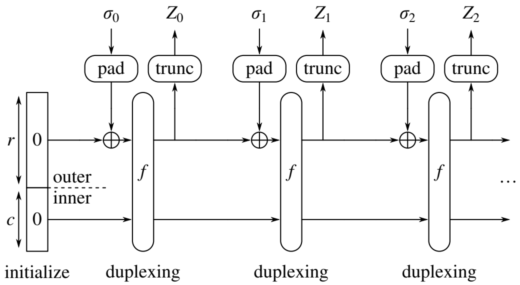
Duplex [Bertoni et al., SAC 2011]:



- Security of duplex equivalent to sponge
- Security of outer-keyed duplex equivalent to keyed sponge
- Overwrite Duplex (OD): variant where bulk of input σ overwrites state

Overwrite Duplex (OD)

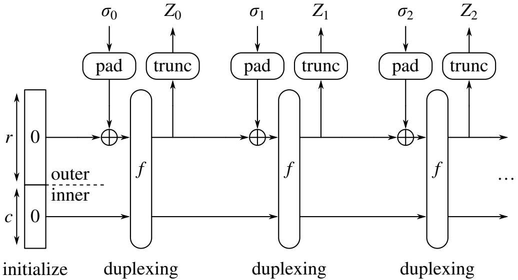
Duplex [Bertoni et al., SAC 2011]:



- Security of duplex equivalent to sponge
- Security of outer-keyed duplex equivalent to keyed sponge
- Overwrite Duplex (OD): variant where bulk of input σ overwrites state

Overwrite Duplex (OD)

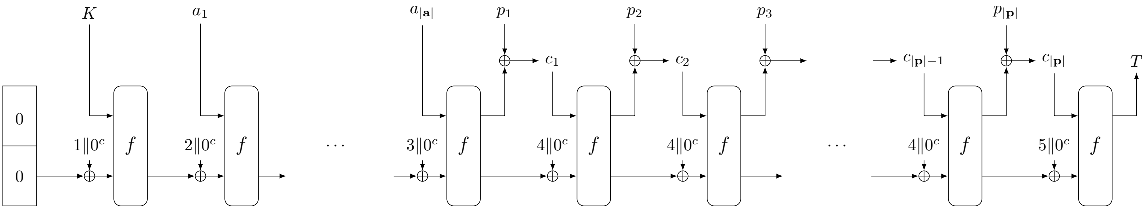
Duplex [Bertoni et al., SAC 2011]:



- Security of duplex equivalent to sponge
- Security of outer-keyed duplex equivalent to keyed sponge
- Overwrite Duplex (OD): variant where bulk of input σ overwrites state

(Turbo)SHAKE-Wrap: nonce-based session AE

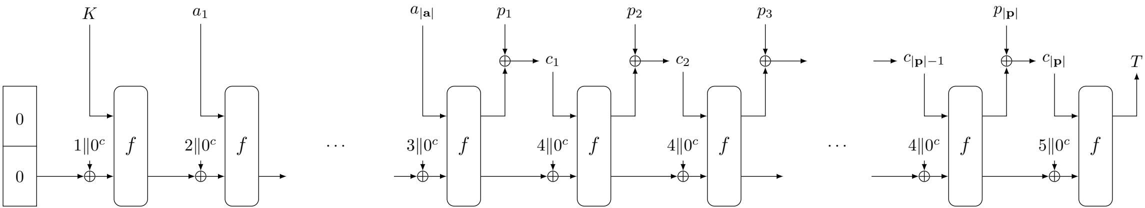
- Mode on top of ODWrap instantiated with one of the four SHAKEs
- Supports sessions: *online* AE through interm. tags and bidirectional messages



- Simple duplex-based AE with 1 domain separation byte per *f* call
- AE confidentiality and integrity follows from security of keyed SHAKE
- Committing security reduces to collision-resistance of (unkeyed) SHAKE

(Turbo)SHAKE-Wrap: nonce-based session AE

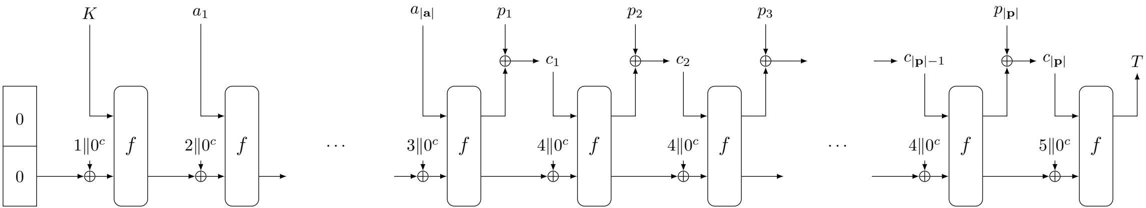
- Mode on top of ODWrap instantiated with one of the four SHAKEs
- Supports sessions: *online* AE through interm. tags and bidirectional messages



- Simple duplex-based AE with 1 domain separation byte per *f* call
- AE confidentiality and integrity follows from security of keyed SHAKE
- Committing security reduces to collision-resistance of (unkeyed) SHAKE

(Turbo)SHAKE-Wrap: nonce-based session AE

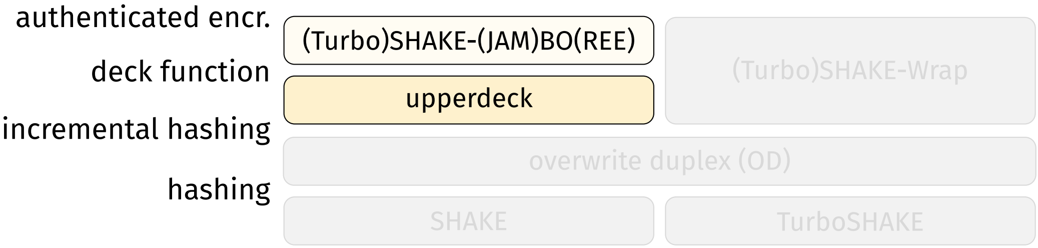
- Mode on top of ODWrap instantiated with one of the four SHAKEs
- Supports sessions: *online* AE through interm. tags and bidirectional messages



- Simple duplex-based AE with 1 domain separation byte per *f* call
- AE confidentiality and integrity follows from security of keyed SHAKE
- Committing security reduces to collision-resistance of (unkeyed) SHAKE

Outline

Deck-based approach



Definition of a deck function

A deck function F_K

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

doubly extendable cryptographic keyed function

Definition of a deck function

A deck function F_K

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

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

Definition of a deck function

A deck function F_K

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

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

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

Efficient incrementality

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

Definition of a deck function

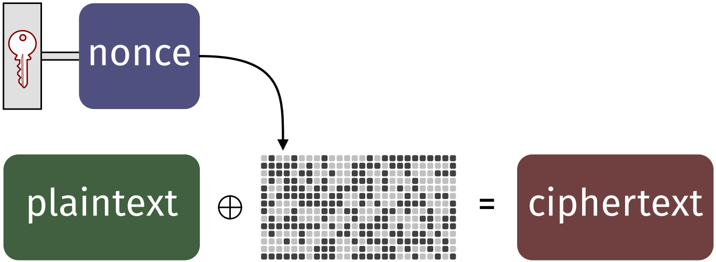
A deck function F_K

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

Efficient incrementality

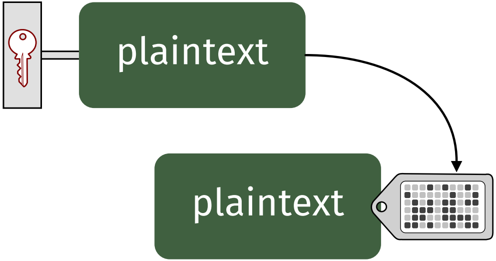
- Extendable input
  1. Compute $F_K(X)$
  2. Compute $F_K(X;Y)$: cost independent of X
- Extendable output
  1. Request n_1 bits from offset 0
  2. Request n_2 bits from offset n_1: cost independent of n_1

Stream encryption: short input, long output



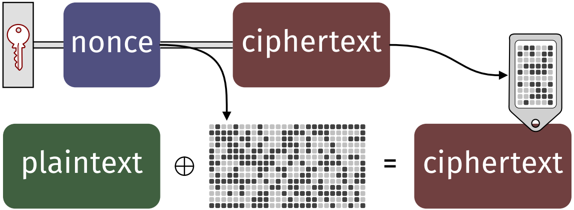
$$C \leftarrow P + F_K(N)$$

MAC computation: long input, short output



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

Authenticated encryption



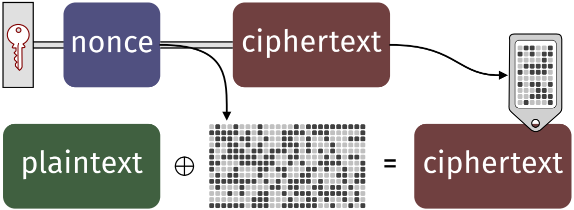
Wrap:

$$C \leftarrow P + F_K(\text{nonce})$$
$$T \leftarrow 0^t + F_K(\text{nonce}; C)$$

Unwrap:

$$T \stackrel{?}{=} 0^t + F_K(\text{nonce}; C)$$
$$P \leftarrow C + F_K(\text{nonce})$$

Authenticated encryption



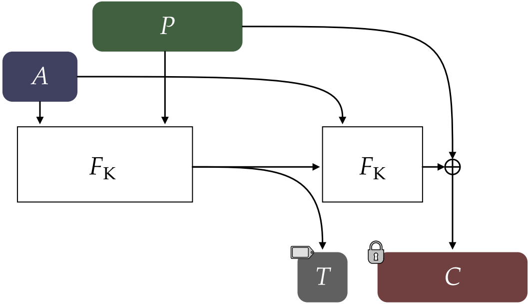
Wrap:

$$C \leftarrow P + F_K(\text{nonce})$$
$$T \leftarrow 0^t + F_K(\text{nonce}; C)$$

Unwrap:

$$T \stackrel{?}{=} 0^t + F_K(\text{nonce}; C)$$
$$P \leftarrow C + F_K(\text{nonce})$$

(SIV)-type authenticated encryption



wrap:

$T \leftarrow 0^t + F_K(\mathrm{AD}; P||0)$

$C \leftarrow P + F_K(\mathrm{AD}; T||1)$

return $C||T$

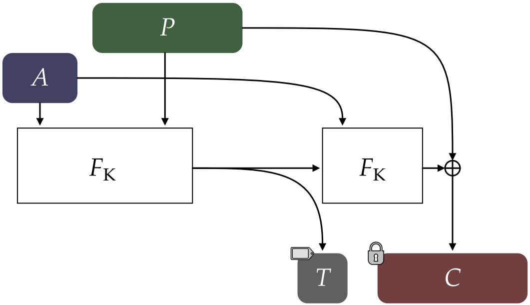
unwrap:

$P \leftarrow C + F_K(\mathrm{AD}; T||1)$

$T \stackrel{?}{=} 0^t + F_K(\mathrm{AD}; P||0)$

return P (or $\perp$)

(SIV)-type authenticated encryption



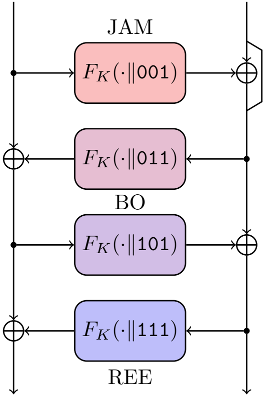
wrap:

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

unwrap:

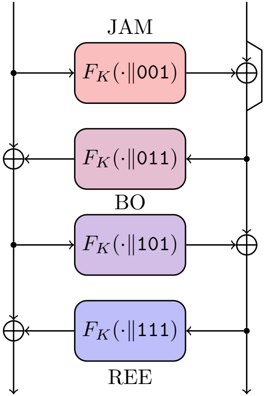
$P \leftarrow C + F_K\,(\mathrm{AD}; T||1)$
$T \stackrel{?}{=} 0^t + F_K\,(\mathrm{AD}; P||0)$
return P (or $\perp$)

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



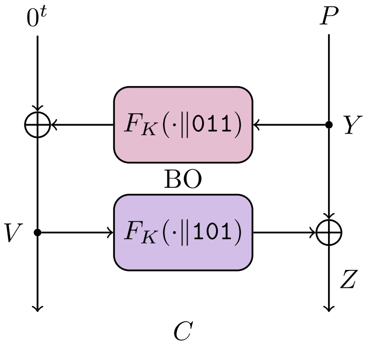
Norica Băcuieți, Joan Daemen, Seth Hoffert, Gilles Van Assche, Ronny Van Keer, Jammin' on the deck , Asiacrypt 2022, `https://eprint.iacr.org/2022/531`

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



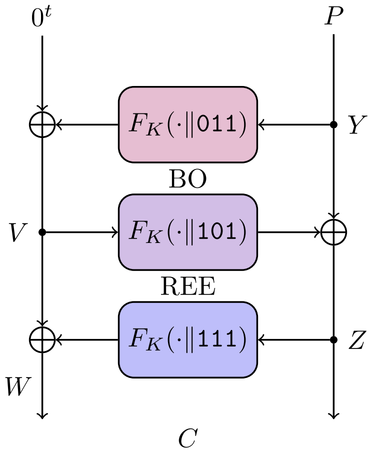
Norica Băcuieți, Joan Daemen, Seth Hoffert, Gilles Van Assche, Ronny Van Keer, Jammin' on the deck , Asiacrypt 2022, `https://eprint.iacr.org/2022/531`

Deck-BO



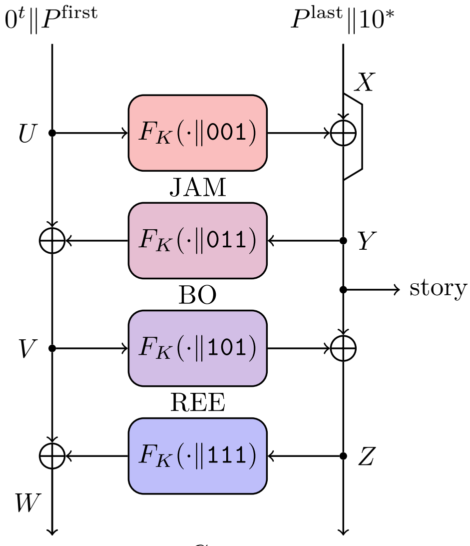
SIV [Rogaway and Shrimpton, EC 2006] + support for AD and sessions

Deck-BOREE



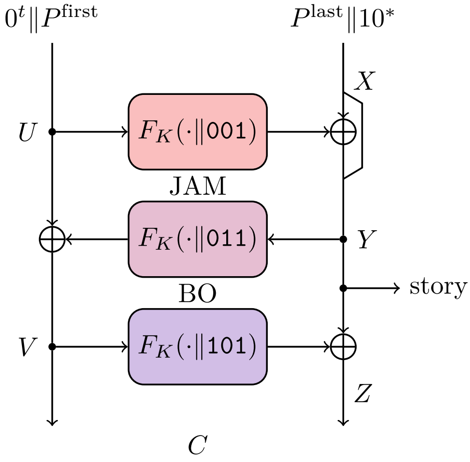
RIV [Abed et al., FSE 2016] + session support

Deck-JAMBOREE



Robust AE [Hoang, Krovetz and Rogaway, EC 2015] + session support

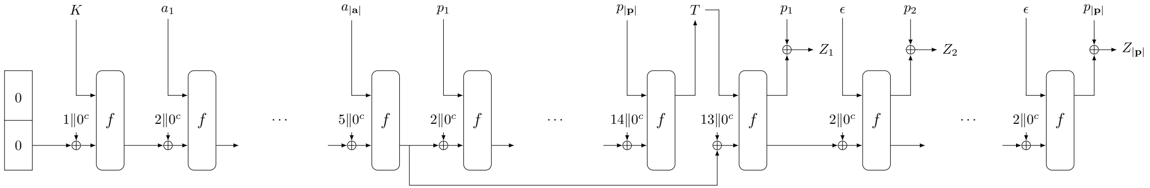
Deck-JAMBO



SIV with optimal redundancy
(but not RUP resistance)

(Turbo)SHAKE-BO: SIV AE with support for sessions

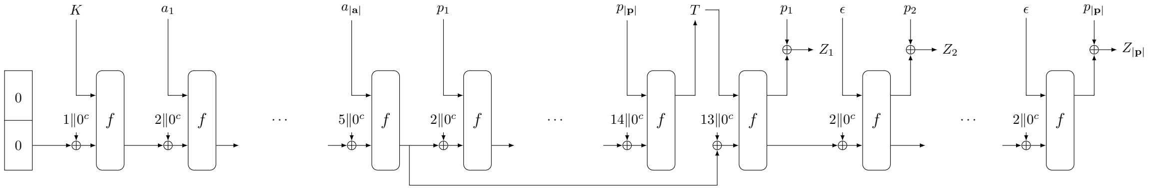
- Upperdeck: deck function on top of OD
- (Turbo)SHAKE-BO: Deck-BO on top of upperdeck instantiated with (Turbo)SHAKE



- Simple AE mode with sponge-based deck calls and 1 domain separation byte per f call
- AE confidentiality and integrity follows from the security of keyed SHAKE
- Committing security reduces to collision-resistance of (unkeyed) SHAKE

(Turbo)SHAKE-BO: SIV AE with support for sessions

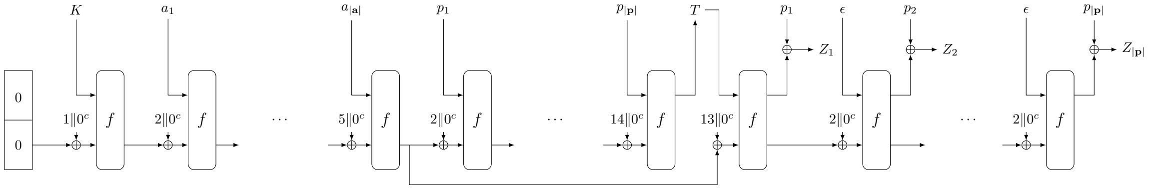
- Upperdeck: deck function on top of OD
- (Turbo)SHAKE-BO: Deck-BO on top of upperdeck instantiated with (Turbo)SHAKE



- Simple AE mode with sponge-based deck calls and 1 domain separation byte per f call
- AE confidentiality and integrity follows from the security of keyed SHAKE
- Committing security reduces to collision-resistance of (unkeyed) SHAKE

(Turbo)SHAKE-BO: SIV AE with support for sessions

- Upperdeck: deck function on top of OD
- (Turbo)SHAKE-BO: Deck-BO on top of upperdeck instantiated with (Turbo)SHAKE



- Simple AE mode with sponge-based deck calls and 1 domain separation byte per *f* call
- AE confidentiality and integrity follows from the security of keyed SHAKE
- Committing security reduces to collision-resistance of (unkeyed) SHAKE

Conclusions

Two approaches for **committing AE** on top of 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 AE** on top of 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!