Jammin' on the deck

Norica Băcuieți[1] Joan Daemen[1] Seth Hoffert
Gilles Van Assche[2] Ronny Van Keer[2]

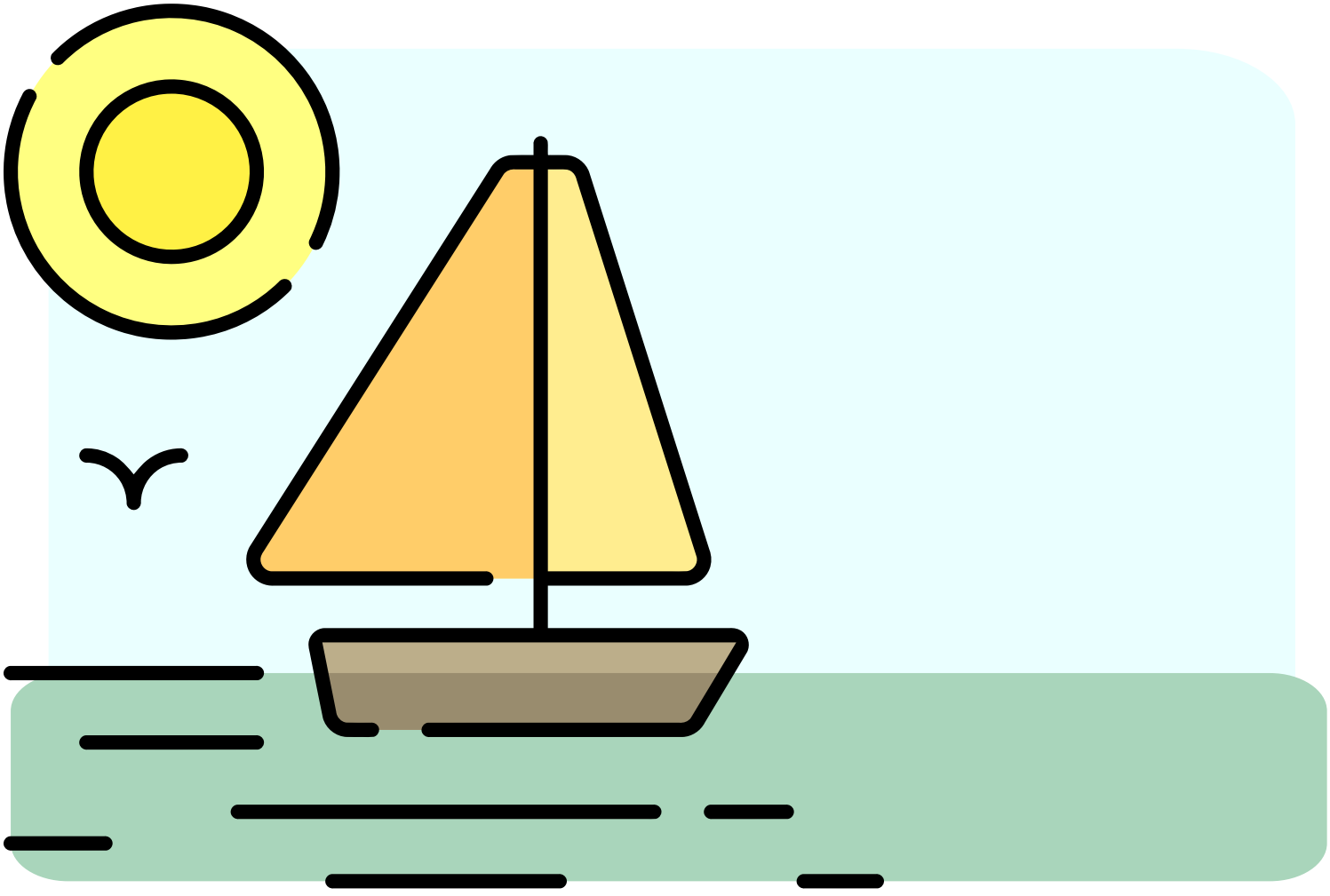
[1]Radboud University
[2]STMicroelectronics

FrisiaCrypt
Terschelling, The Netherlands, September 26, 2022

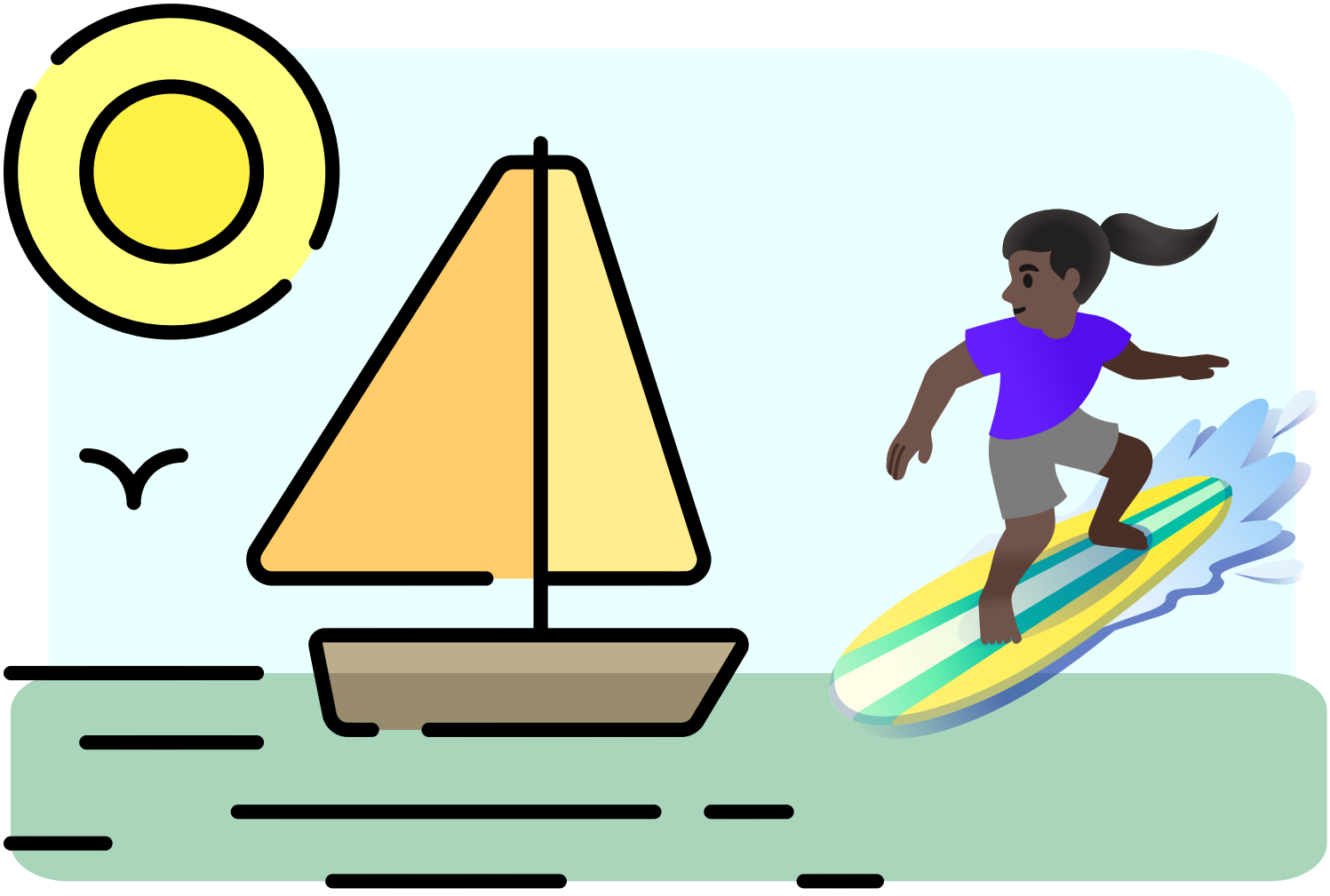
Outline

Outline

The primitive/mode interface

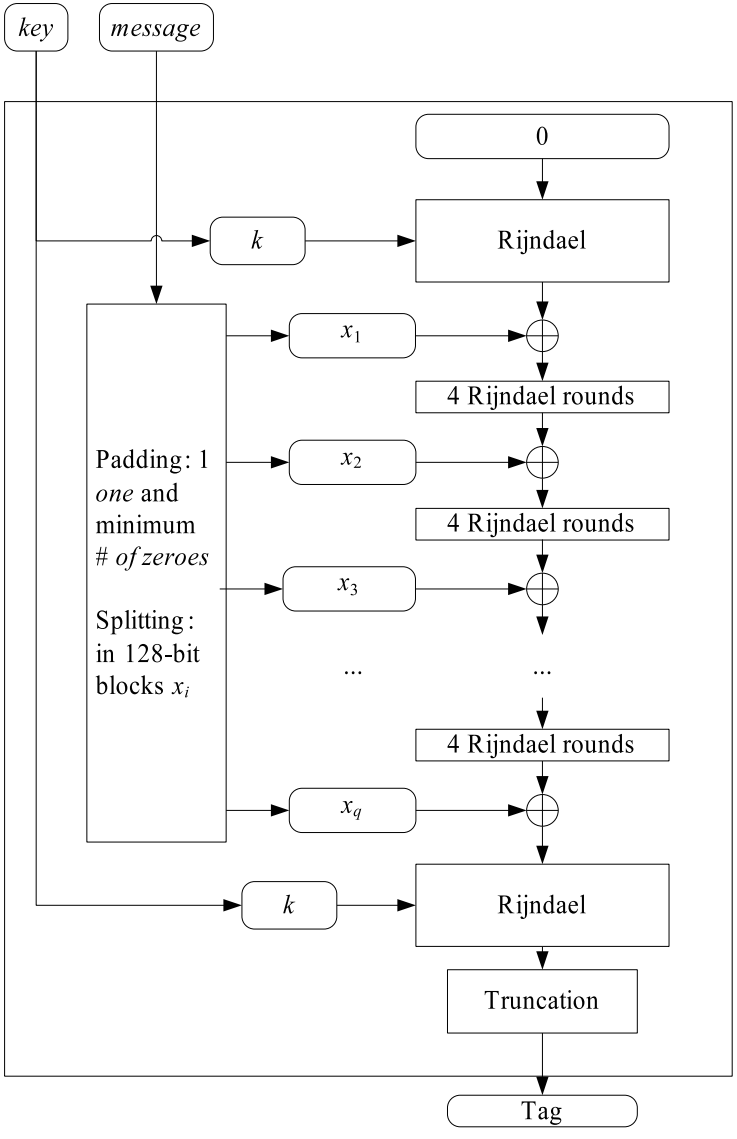


The primitive/mode interface



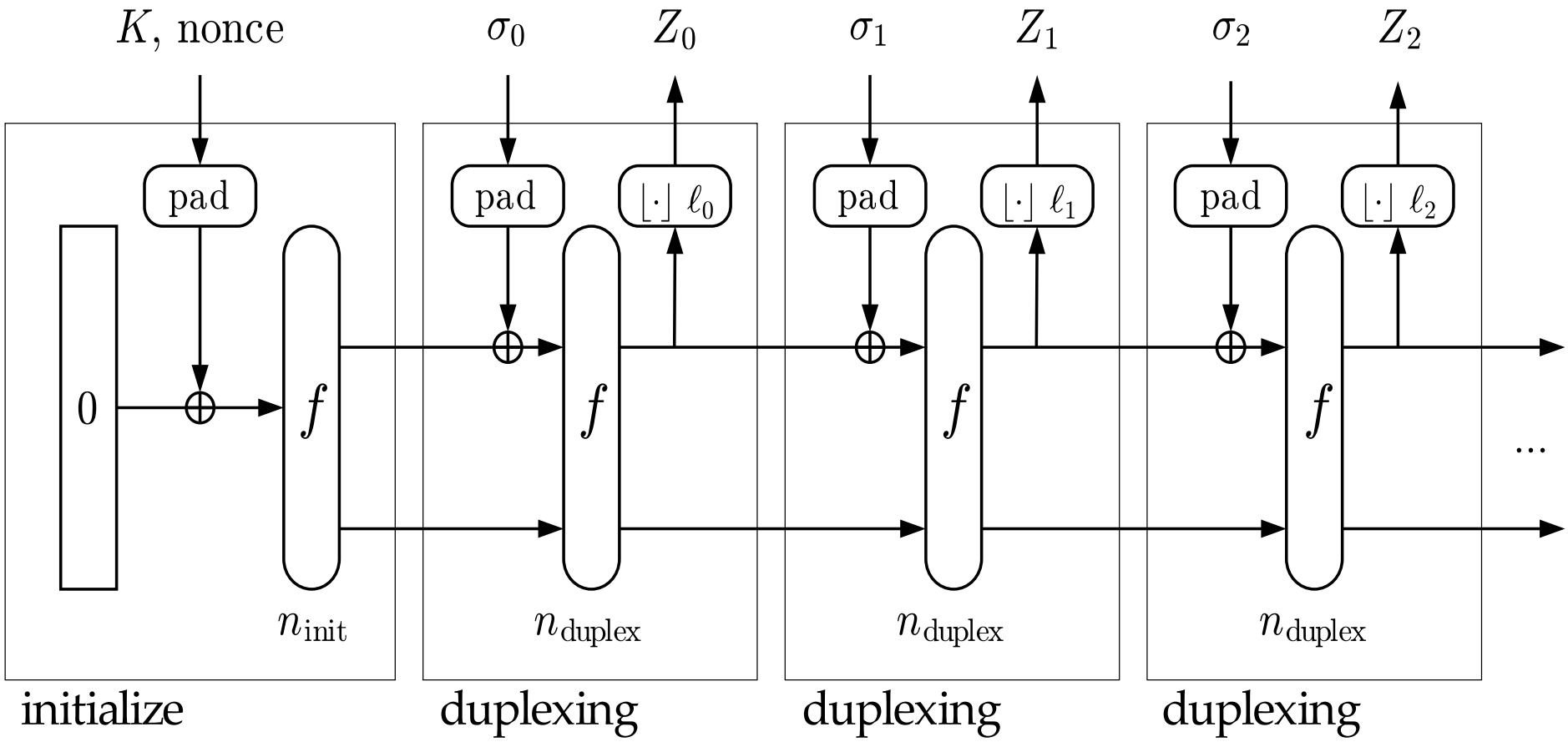
The primitive/mode interface

Variable-length primitives: Pelican 2.0



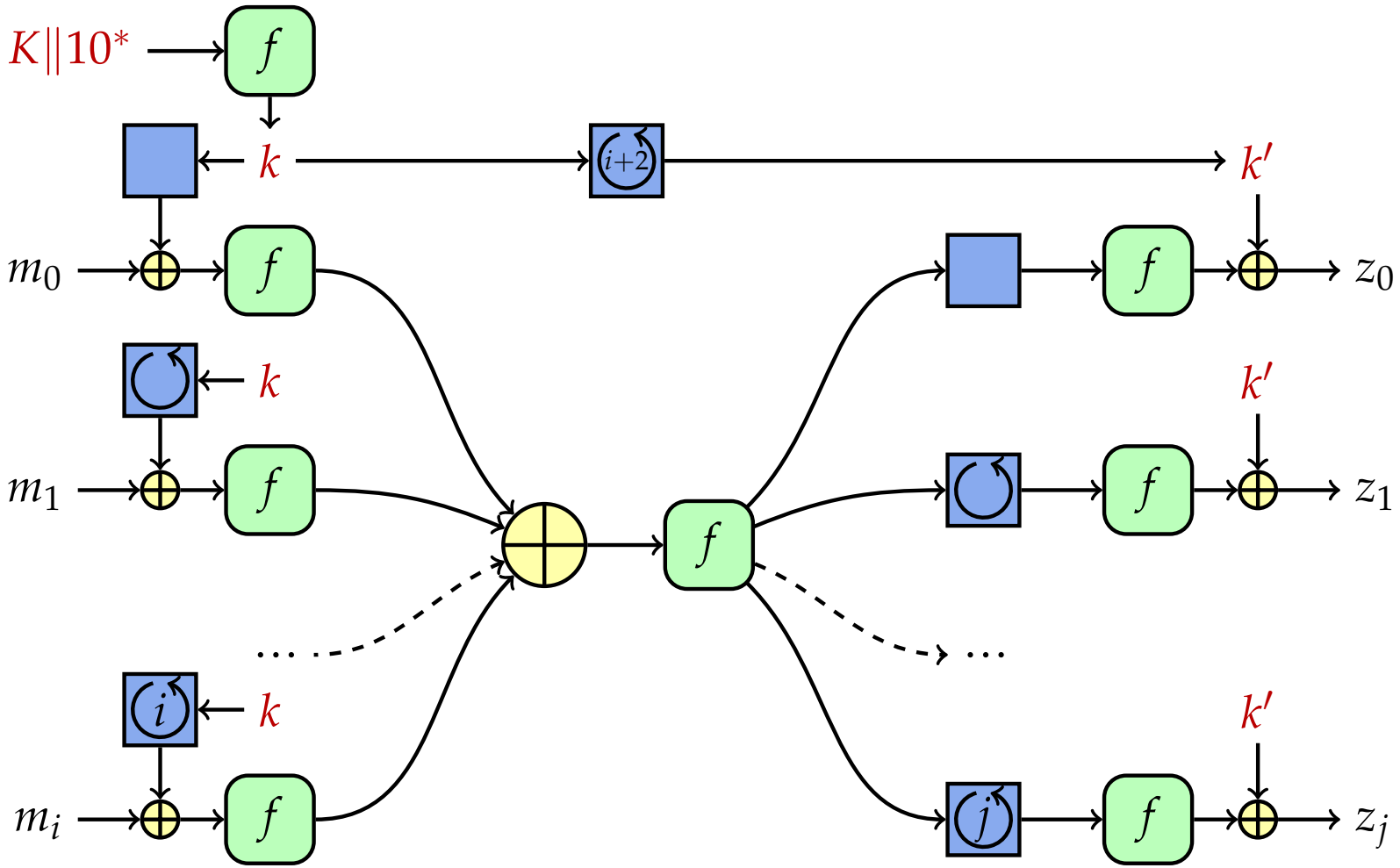
[Daemen and Rijmen, The MAC function Pelican 2.0, ePrint 2005/088]

Variable-length primitives: Monkey Duplex



[Bertoni, Daemen, Peeters and VA, DIAC 2012]

Variable-length primitives: Farfalle



[FSE 2018]

Kravatte and Xoofff

Kravatte [FSE 2018]

- $f = \text{Keccak-}p[1600, n_r = 6]$
- Input mask rolling with LFSR, state rolling with NLFSR
- Target security: ≥ 128 bits (including post-quantum)

Xoofff [FSE 2019]

- $f = \text{Xoodoo}[6]$
  384-bit permutation $4 \times 3 \times 32$ bits
- Target security: ≥ 128 bits (≥ 96 bits post-quantum)

Kravatte and Xoofff

Kravatte [FSE 2018]

- $f = \text{Keccak-}p[1600, n_r = 6]$
- Input mask rolling with LFSR, state rolling with NLFSR
- Target security: ≥ 128 bits (including post-quantum)

Xoofff [FSE 2019]

- $f = \text{Xoodoo}[6]$
  384-bit permutation $4 \times 3 \times 32$ bits
- Target security: ≥ 128 bits (≥ 96 bits post-quantum)

Outline

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: potentially infinite output
  - **pseudo-random function of the input**
  - taking n bits starting from offset q

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

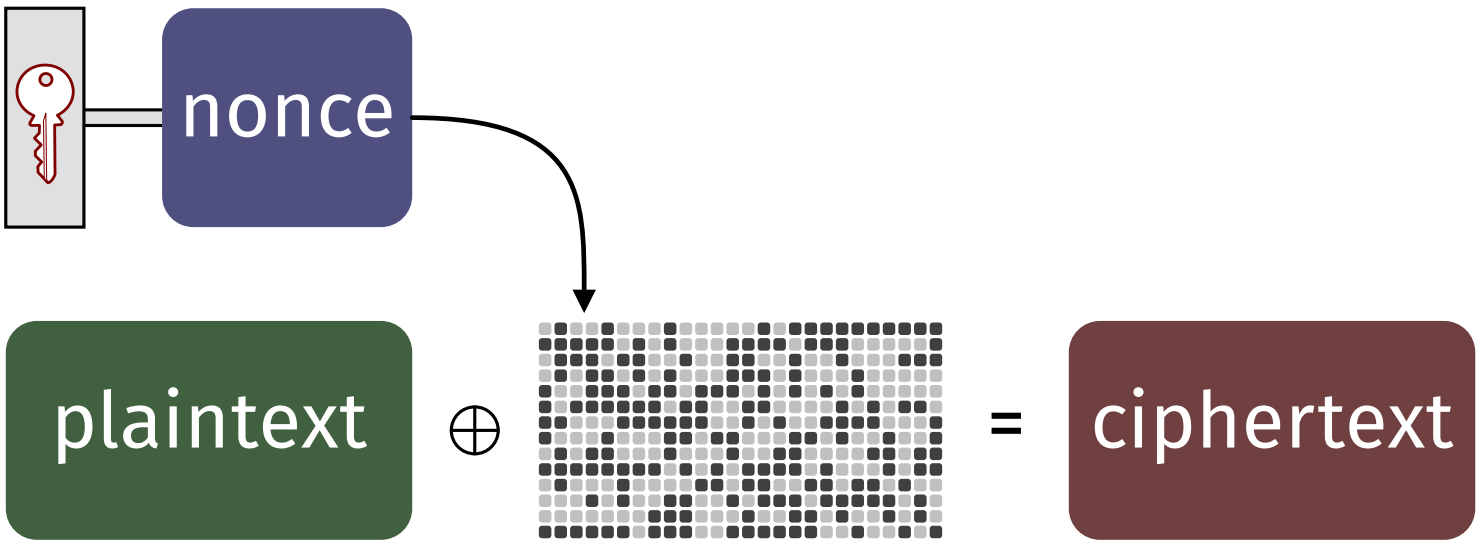
A deck function F_K

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

Efficient incrementality

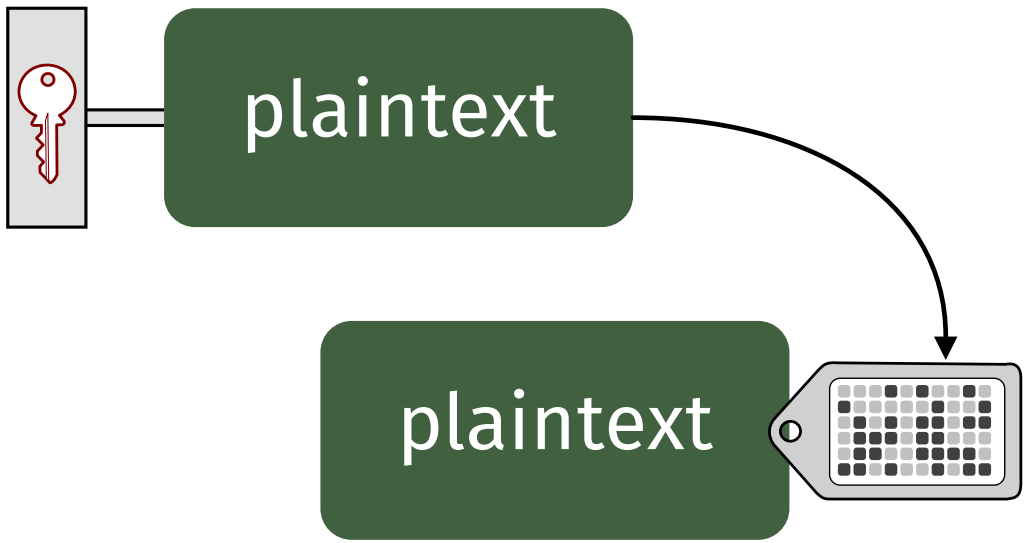
- Extendable input
  1. Compute $F_K(X)$
  2. Compute $F_K(X; Y)$: cost independent of X
- Extendable output
  1. Request n_1 bits from offset 0
  2. Request n_2 bits from offset n_1: cost independent of n_1

Stream cipher: short input, long output



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

MAC: long input, short output



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

Outline

Deck-PLAIN: session-supporting and nonce-based

Encipher first (or single) message (associated data A_1, plaintext P_1)

$Z_1 \leftarrow P_1 + F_K(A_1||10)$

$T_1 \leftarrow 0^t + F_K(A_1||10; Z_1||1)$

return $C_1 = Z_1||T_1$

Encipher second message (P_2, no associated data)

$Z_2 \leftarrow P_2 + F_K(A_1||10; Z_1||1) \ll t$

$T_2 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1)$

return $C_2 = Z_2||T_2$

Encipher third message (A_3, no plaintext)

$T_3 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1; A_3||00)$

return T_3

Deck-PLAIN: session-supporting and nonce-based

Encipher first (or single) message (associated data A_1, plaintext P_1)
$Z_1 \leftarrow P_1 + F_K(A_1||10)$
$T_1 \leftarrow 0^t + F_K(A_1||10; Z_1||1)$
return $C_1 = Z_1||T_1$

Encipher second message (P_2, no associated data)
$Z_2 \leftarrow P_2 + F_K(A_1||10; Z_1||1) \ll t$
$T_2 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1)$
return $C_2 = Z_2||T_2$

Encipher third message (A_3, no plaintext)
$T_3 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1; A_3||00)$
return T_3

Deck-PLAIN: session-supporting and nonce-based

Encipher first (or single) message (associated data A_1, plaintext P_1)
$Z_1 \leftarrow P_1 + F_K(A_1||10)$
$T_1 \leftarrow 0^t + F_K(A_1||10; Z_1||1)$
return $C_1 = Z_1||T_1$

Encipher second message (P_2, no associated data)
$Z_2 \leftarrow P_2 + F_K(A_1||10; Z_1||1) \ll t$
$T_2 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1)$
return $C_2 = Z_2||T_2$

Encipher third message (A_3, no plaintext)
$T_3 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1; A_3||00)$
return T_3

Deck-PLAIN: session-supporting and nonce-based

Encipher first (or single) message (associated data A_1, plaintext P_1)
$Z_1 \leftarrow P_1 + F_K(A_1||10)$
$T_1 \leftarrow 0^t + F_K(A_1||10; Z_1||1)$
return $C_1 = Z_1||T_1$

Encipher second message (P_2, no associated data)
$Z_2 \leftarrow P_2 + F_K(A_1||10; Z_1||1) \ll t$
$T_2 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1)$
return $C_2 = Z_2||T_2$

Encipher third message (A_3, no plaintext)
$T_3 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1; A_3||00)$
return T_3

Deck-PLAIN: session-supporting and nonce-based

Encipher first (or single) message (associated data A_1, plaintext P_1)
$Z_1 \leftarrow P_1 + F_K(A_1||10)$
$T_1 \leftarrow 0^t + F_K(A_1||10; Z_1||1)$
return $C_1 = Z_1||T_1$

Encipher second message (P_2, no associated data)
$Z_2 \leftarrow P_2 + F_K(A_1||10; Z_1||1) \ll t$
$T_2 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1)$
return $C_2 = Z_2||T_2$

Encipher third message (A_3, no plaintext)
$T_3 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1; A_3||00)$
return T_3

Deck-PLAIN: session-supporting and nonce-based

Encipher first (or single) message (associated data A_1, plaintext P_1)
$Z_1 \leftarrow P_1 + F_K(A_1||10)$
$T_1 \leftarrow 0^t + F_K(A_1||10; Z_1||1)$
return $C_1 = Z_1||T_1$

Encipher second message (P_2, no associated data)
$Z_2 \leftarrow P_2 + F_K(A_1||10; Z_1||1) \ll t$
$T_2 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1)$
return $C_2 = Z_2||T_2$

Encipher third message (A_3, no plaintext)
$T_3 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1; A_3||00)$
return T_3

Deck-PLAIN: session-supporting and nonce-based

Encipher first (or single) message (associated data A_1, plaintext P_1)
$Z_1 \leftarrow P_1 + F_K(A_1||10)$
$T_1 \leftarrow 0^t + F_K(A_1||10; Z_1||1)$
return $C_1 = Z_1||T_1$

Encipher second message (P_2, no associated data)
$Z_2 \leftarrow P_2 + F_K(A_1||10; Z_1||1) \ll t$
$T_2 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1)$
return $C_2 = Z_2||T_2$

Encipher third message (A_3, no plaintext)
$T_3 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1; A_3||00)$
return T_3

Deck-PLAIN: session-supporting and nonce-based

Encipher first (or single) message (associated data A_1, plaintext P_1)
$Z_1 \leftarrow P_1 + F_K(A_1||10)$
$T_1 \leftarrow 0^t + F_K(A_1||10; Z_1||1)$
return $C_1 = Z_1||T_1$

Encipher second message (P_2, no associated data)
$Z_2 \leftarrow P_2 + F_K(A_1||10; Z_1||1) \ll t$
$T_2 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1)$
return $C_2 = Z_2||T_2$

Encipher third message (A_3, no plaintext)
$T_3 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1; A_3||00)$
return T_3

Deck-PLAIN: session-supporting and nonce-based

Encipher first (or single) message (associated data A_1, plaintext P_1)
$Z_1 \leftarrow P_1 + F_K(A_1||10)$
$T_1 \leftarrow 0^t + F_K(A_1||10; Z_1||1)$
return $C_1 = Z_1||T_1$

Encipher second message (P_2, no associated data)
$Z_2 \leftarrow P_2 + F_K(A_1||10; Z_1||1) \ll t$
$T_2 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1)$
return $C_2 = Z_2||T_2$

Encipher third message (A_3, no plaintext)
$T_3 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1; A_3||00)$
return T_3

Deck-PLAIN: session-supporting and nonce-based

Encipher first (or single) message (associated data A_1, plaintext P_1)
$Z_1 \leftarrow P_1 + F_K(A_1||10)$
$T_1 \leftarrow 0^t + F_K(A_1||10; Z_1||1)$
return $C_1 = Z_1||T_1$

Encipher second message (P_2, no associated data)
$Z_2 \leftarrow P_2 + F_K(A_1||10; Z_1||1) \ll t$
$T_2 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1)$
return $C_2 = Z_2||T_2$

Encipher third message (A_3, no plaintext)
$T_3 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1; A_3||00)$
return T_3

Deck-PLAIN: session-supporting and nonce-based

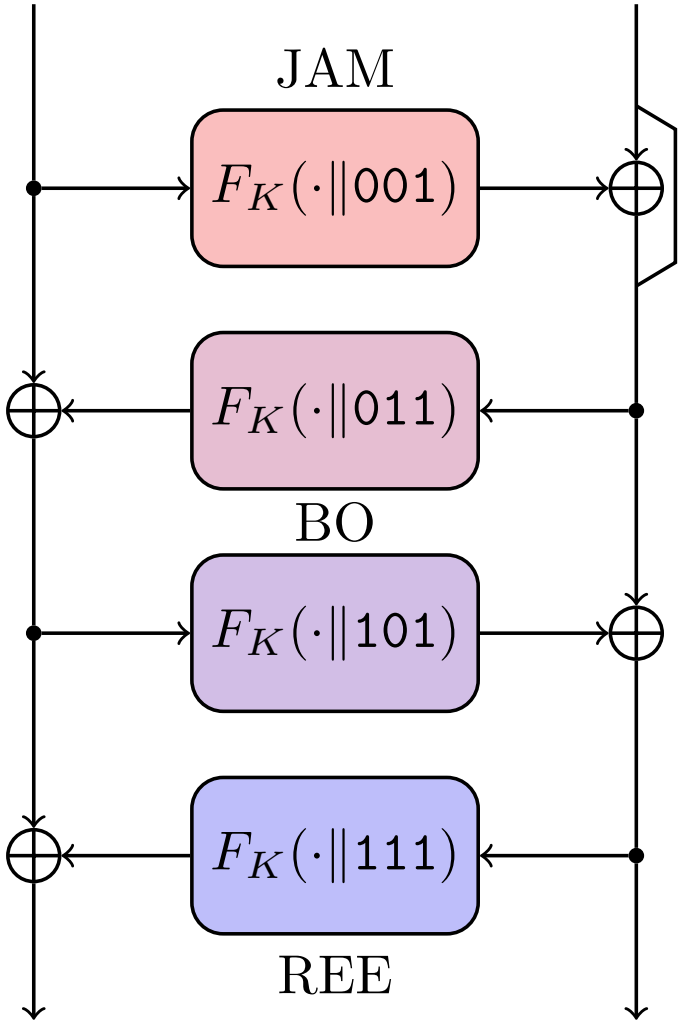
Encipher first (or single) message (associated data A_1, plaintext P_1)
$Z_1 \leftarrow P_1 + F_K(A_1||10)$
$T_1 \leftarrow 0^t + F_K(A_1||10; Z_1||1)$
return $C_1 = Z_1||T_1$

Encipher second message (P_2, no associated data)
$Z_2 \leftarrow P_2 + F_K(A_1||10; Z_1||1) \ll t$
$T_2 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1)$
return $C_2 = Z_2||T_2$

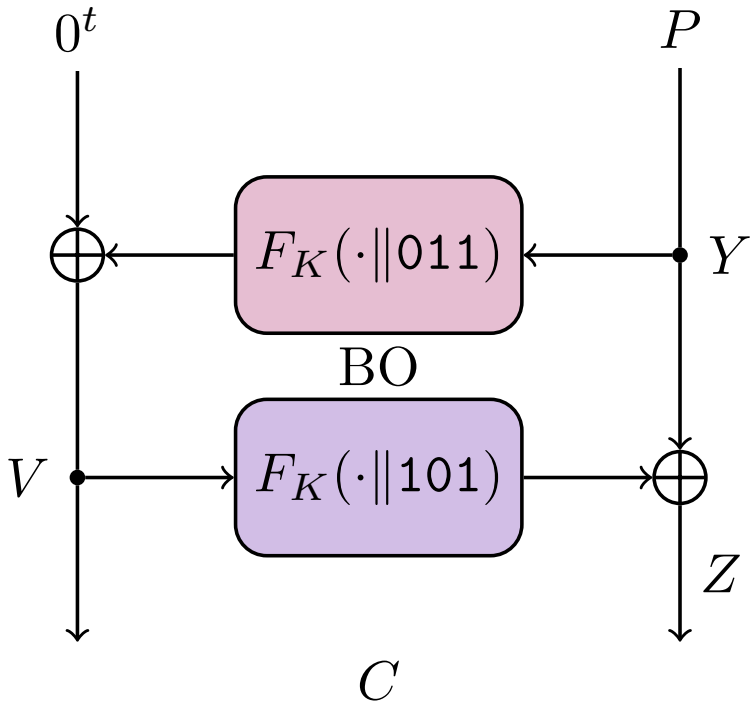
Encipher third message (A_3, no plaintext)
$T_3 \leftarrow 0^t + F_K(A_1||10; Z_1||1; Z_2||1; A_3||00)$
return T_3

Outline

Feistel network



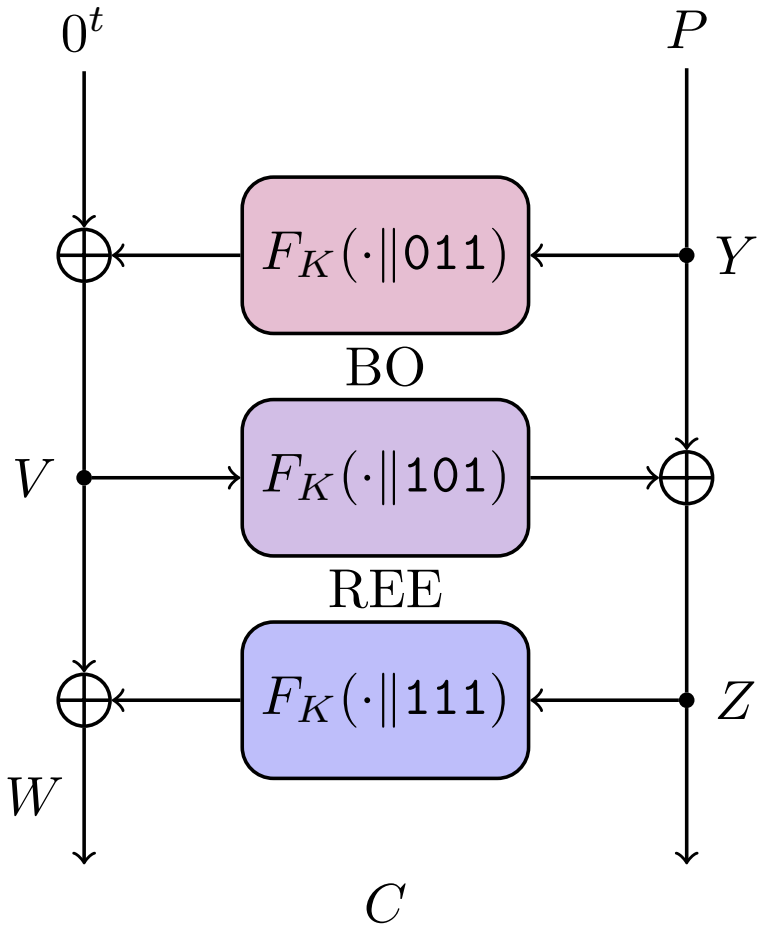
Deck-BO



SIV + session support

[Rogaway and Shrimpton, EUROCRYPT 2006]

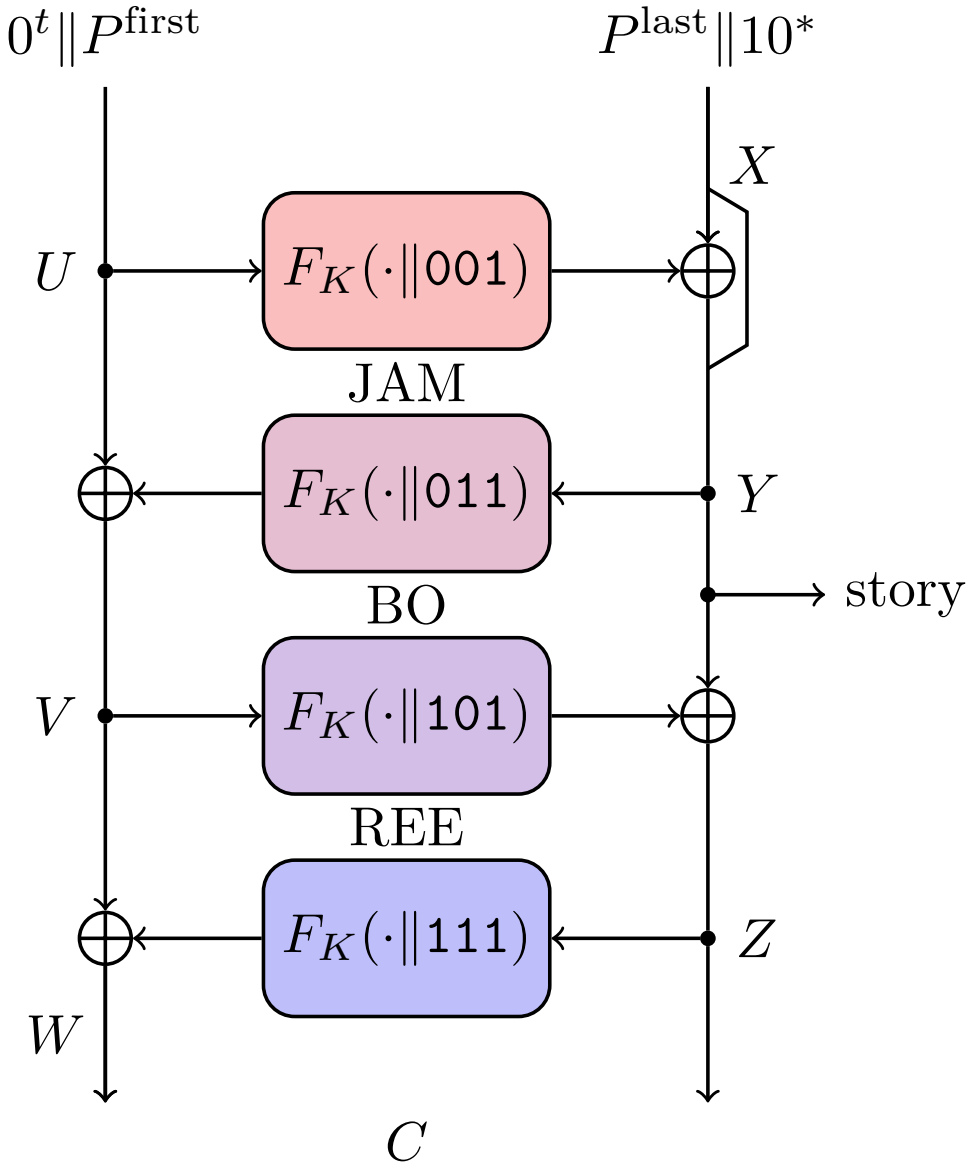
Deck-BOREE



RIV + session support

[Abed, Forler, List, Lucks and Wenzel, FSE 2016]

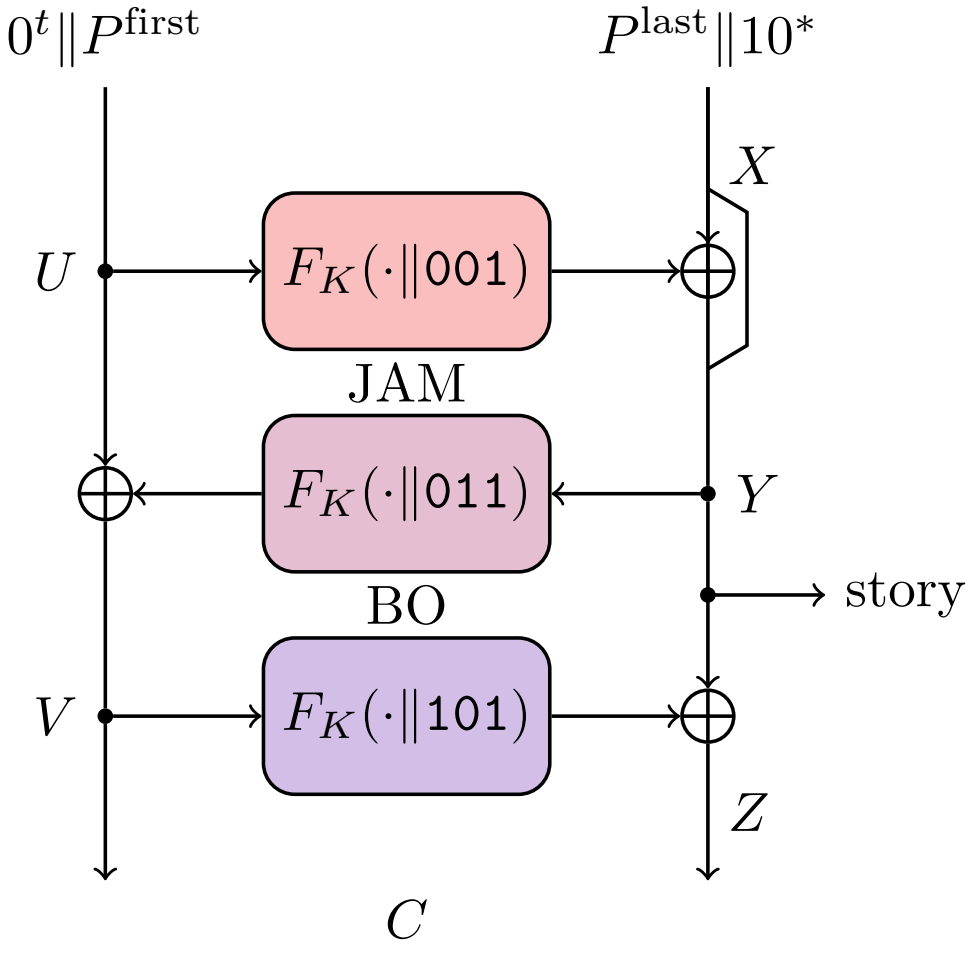
Deck-JAMBOREE



Robust AE + session support

[Hoang, Krovetz and Rogaway, EUROCRYPT 2015]

Deck-JAMBO



SIV with optimal redundancy
(but not RUP resistance)

Outline

An ideal model

Desired properties:

- Operational and referential
- Nonce-enforcing and nonce-misuse-resistant
- Sessions and bi-directional communications
- Parameterized ciphertext expansion
- Multi-key security

An ideal model

Desired properties:

- Operational and referential
- Nonce-enforcing and nonce-misuse-resistant
- Sessions and bi-directional communications
- Parameterized ciphertext expansion
- Multi-key security

An ideal model

Desired properties:

- Operational and referential
- Nonce-enforcing and nonce-misuse-resistant
- Sessions and bi-directional communications
- Parameterized ciphertext expansion
- Multi-key security

An ideal model

Desired properties:

- Operational and referential
- Nonce-enforcing and nonce-misuse-resistant
- Sessions and bi-directional communications
- Parameterized ciphertext expansion
- Multi-key security

An ideal model

Desired properties:

- Operational and referential
- Nonce-enforcing and nonce-misuse-resistant
- Sessions and bi-directional communications
- Parameterized ciphertext expansion
- Multi-key security

An ideal model

Desired properties:

- Operational and referential
- Nonce-enforcing and nonce-misuse-resistant
- Sessions and bi-directional communications
- Parameterized ciphertext expansion
- Multi-key security

The jammin cipher (no sessions)

Global variables: codebook initially set to $\perp$ for all, taboo initially set to *empty*
Instance constructor: init(ID), **return** new instance with inst.ID $=$ ID
Instance cloner: inst.clone(), **return** new instance with same inst.ID

Interface: inst.wrap(A, P) returns C
context $\leftarrow$ inst.ID; A
if codebook(context; P) $= \perp$ **then**
  $\mathcal{C} = \mathbb{Z}_2^{\mathsf{WrapExpand}(|P|)} \setminus (\text{codebook}(\text{context}; *) \cup \text{taboo}(\text{context}))$
  if $\mathcal{C} = \emptyset$ **then return** $\perp$
  codebook(context; P) $\xleftarrow{\$} \mathcal{C}$
...
return codebook(context; P)

Interface: inst.unwrap(A, C) returns P or $\perp$
context $\leftarrow$ inst.ID; A
if $\exists! P :$ codebook(context; P) $= C$
then return P
else taboo(context) $\leftarrow C$, **return** $\perp$

The jammin cipher (no sessions)

Global variables: codebook initially set to $\perp$ for all, taboo initially set to *empty*
Instance constructor: init(ID), **return** new instance with inst.ID $=$ ID
Instance cloner: inst.clone(), **return** new instance with same inst.ID

Interface: inst.wrap(A, P) returns C
context $\leftarrow$ inst.ID; A
if codebook(context; P) $= \perp$ **then**
  $\mathcal{C} = \mathbb{Z}_2^{\mathsf{WrapExpand}(|P|)} \setminus (\text{codebook}(\text{context}; *) \cup \text{taboo}(\text{context}))$
  if $\mathcal{C} = \emptyset$ **then return** $\perp$
  codebook(context; P) $\xleftarrow{\$} \mathcal{C}$
...
return codebook(context; P)

Interface: inst.unwrap(A, C) returns P or $\perp$
context $\leftarrow$ inst.ID; A
if $\exists! P :$ codebook(context; P) $= C$
then return P
else taboo(context) $\leftarrow C$, **return** $\perp$

The jammin cipher (no sessions)

Global variables: codebook initially set to $\perp$ for all, taboo initially set to *empty*
Instance constructor: init(ID), **return** new instance with inst.ID $=$ ID
Instance cloner: inst.clone(), **return** new instance with same inst.ID

Interface: inst.wrap(A, P) returns C
context $\leftarrow$ inst.ID; A
if codebook(context; P) $= \perp$ **then**
  $\mathcal{C} = \mathbb{Z}_2^{\mathsf{WrapExpand}(|P|)} \setminus (\text{codebook}(\text{context}; *) \cup \text{taboo}(\text{context}))$
  if $\mathcal{C} = \emptyset$ **then return** $\perp$
  codebook(context; P) $\xleftarrow{\$} \mathcal{C}$
...
return codebook(context; P)

Interface: inst.unwrap(A, C) returns P or $\perp$
context $\leftarrow$ inst.ID; A
if $\exists! P :$ codebook(context; P) $= C$
then return P
else taboo(context) $\leftarrow C$, **return** $\perp$

The jammin cipher (no sessions)

Global variables: codebook initially set to $\perp$ for all, taboo initially set to *empty*
Instance constructor: init(ID), **return** new instance with inst.ID $=$ ID
Instance cloner: inst.clone(), **return** new instance with same inst.ID

Interface: inst.wrap(A, P) returns C
context $\leftarrow$ inst.ID; A
if codebook(context; P) $= \perp$ **then**
  $\mathcal{C} = \mathbb{Z}_2^{\mathsf{WrapExpand}(|P|)} \setminus (\text{codebook}(\text{context}; *) \cup \text{taboo}(\text{context}))$
  if $\mathcal{C} = \emptyset$ **then return** $\perp$
  codebook(context; P) $\xleftarrow{\$} \mathcal{C}$
...
return codebook(context; P)

Interface: inst.unwrap(A, C) returns P or $\perp$
context $\leftarrow$ inst.ID; A
if $\exists! P$: codebook(context; P) $= C$
then return P
else taboo(context) $\leftarrow C$, **return** $\perp$

The jammin cipher (no sessions)

Global variables: codebook initially set to $\perp$ for all, taboo initially set to *empty*
Instance constructor: init(ID), **return** new instance with inst.ID $=$ ID
Instance cloner: inst.clone(), **return** new instance with same inst.ID

Interface: inst.wrap(A, P) returns C
context $\leftarrow$ inst.ID; A
if codebook(context; P) $= \perp$ **then**
  $\mathcal{C} = \mathbb{Z}_2^{\mathsf{WrapExpand}(|P|)} \setminus (\text{codebook}(\text{context}; *) \cup \text{taboo}(\text{context}))$
  if $\mathcal{C} = \emptyset$ **then return** $\perp$
  codebook(context; P) $\xleftarrow{\$} \mathcal{C}$
...
return codebook(context; P)

Interface: inst.unwrap(A, C) returns P or $\perp$
context $\leftarrow$ inst.ID; A
if $\exists! P$: codebook(context; P) $= C$
then return P
else taboo(context) $\leftarrow C$, **return** $\perp$

The jammin cipher (no sessions)

Global variables: codebook initially set to $\perp$ for all, taboo initially set to *empty*
Instance constructor: init(ID), **return** new instance with inst.ID $=$ ID
Instance cloner: inst.clone(), **return** new instance with same inst.ID

Interface: inst.wrap(A, P) returns C
context $\leftarrow$ inst.ID; A
if codebook(context; P) $= \perp$ **then**
  $\mathcal{C} = \mathbb{Z}_2^{\mathsf{WrapExpand}(|P|)} \setminus (\mathsf{codebook}(\mathsf{context}; *) \cup \mathsf{taboo}(\mathsf{context}))$
  if $\mathcal{C} = \emptyset$ **then return** $\perp$
  codebook(context; P) $\xleftarrow{\$} \mathcal{C}$
...
return codebook(context; P)

Interface: inst.unwrap(A, C) returns P or $\perp$
context $\leftarrow$ inst.ID; A
if $\exists! P$: codebook(context; P) $= C$
then return P
else taboo(context) $\leftarrow C$, **return** $\perp$

The jammin cipher

Global variables: codebook initially set to $\perp$ for all, taboo initially set to *empty*
Instance constructor: init(ID), **return** new instance with inst.history $=$ ID
Instance cloner: inst.clone(), **return** new instance with same inst.history

Interface: inst.wrap(A, P) returns C
context $\leftarrow$ inst.history; A
if codebook(context; P) $= \perp$ **then**
  $\mathcal{C} = \mathbb{Z}_2^{\mathsf{WrapExpand}(|P|)} \setminus (\mathsf{codebook}(\mathsf{context}; *) \cup \mathsf{taboo}(\mathsf{context}))$
  if $\mathcal{C} = \emptyset$ **then return** $\perp$
  codebook(context; P) $\xleftarrow{\$} \mathcal{C}$
inst.history $\leftarrow$ inst.history; A; P
return codebook(context; P)

Interface: inst.unwrap(A, C) returns P or $\perp$
context $\leftarrow$ inst.history; A
if $\exists! P$: codebook(context; P) $= C$
then inst.history $\leftarrow$ inst.history; A; P, **return** P
else taboo(context) $\leftarrow C$, **return** $\perp$

The jammin cipher and OAE2

Theorem

Let $\mathcal{J}^{+t}$ be the jammin cipher with $\mathsf{WrapExpand}(p) = p + t$. Then, for any adversary $\mathcal{D}$ that makes at most q queries, we have

$$\mathbf{Adv}^{\text{oae2-priv}}_{\mathcal{J}^{+t}}(\mathcal{D}) \leq \frac{q}{2^{t+1}} \quad \text{and} \quad \mathbf{Adv}^{\text{oae2-auth}}_{\mathcal{J}^{+t}}(\mathcal{D}) = 0\,.$$

Furthermore, when the encryption context is a nonce, we have

$$\mathbf{Adv}^{\text{oae2-priv}}_{\mathcal{J}^{+t}}(\mathcal{D}) = \mathbf{Adv}^{\text{oae2-auth}}_{\mathcal{J}^{+t}}(\mathcal{D}) = 0\,.$$

The jammin cipher can replace OAE2a ∪ OAE2b ∪ OAE2c ∪ nOAE ∪ dOAE. [Hoang, Reyhanitabar, Rogaway and Vizár, CRYPTO 2015]

Conclusions

Deck functions

- bring a new useful API to simplify modes
- put safety margin at the right place
- allow efficient ciphers

The jammin cipher

- provides a simple yet powerful model for AE
- works for both session and non-session AE
- is the model of choice for Deck-* modes

Conclusions

Deck functions

- bring a new useful API to simplify modes
- put safety margin at the right place
- allow efficient ciphers

The jammin cipher

- provides a simple yet powerful model for AE
- works for both session and non-session AE
- is the model of choice for Deck-* modes

Any questions?

Thanks for your attention!

See [ePrint 2022/531] for more details