Radbound University

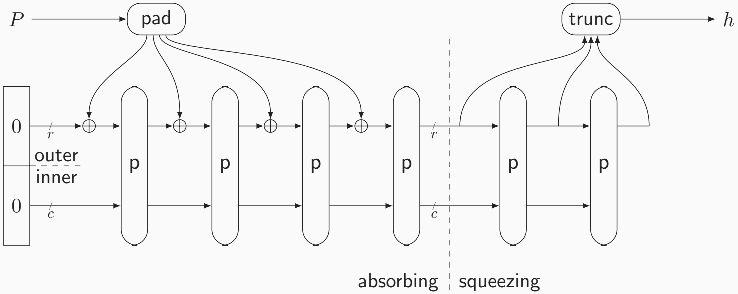
Security of Permutation-Based Modes and Its Application to Ascon

Bart Mennink
Radboud University (The Netherlands)
NIST Lightweight Cryptography Workshop 2023
June 22, 2023

ESCADA

Sponges and Ascon-Hash Mode

Sponges [BDPV07]



- p is a b-bit permutation, with $b = r + c$
  - r is the rate
  - c is the capacity (security parameter)
- SHA-3, XOFs, lightweight hashing, ...

Indifferentiability of the Sponge [BDPV08]

- Assume that p is a random permutation

Indifferentiability of the Sponge [BDPV08]

- Assume that p is a random permutation
- Sponge indifferentiable from random oracle:

$$\Delta_{\mathcal{D}}(\mathsf{sponge}, \mathsf{p}; \mathsf{ro}, \mathsf{sim}) \leq N^2/2^{c+1}$$

  - N is number of permutation evaluations that attacker can make
  - Collisions in the inner part break security of the sponge

Indifferentiability of the Sponge [BDPV08]

- Assume that p is a random permutation
- Sponge indifferentiable from random oracle:

$$\Delta_{\mathcal{D}}(\mathsf{sponge}, \mathsf{p}; \mathsf{ro}, \mathsf{sim}) \leq N^2/2^{c+1}$$

  - N is number of permutation evaluations that attacker can make
  - Collisions in the inner part break security of the sponge
- Security of sponge truncated to n bits against classical attacks:

Collision resistance:	$N^2/2^{c+1} + N^2/2^{n+1}$
Second preimage resistance:	$N^2/2^{c+1} + N/2^n$
Preimage resistance:	$N^2/2^{c+1} + N/2^n$

Indifferentiability of the Sponge [BDPV08]

- Assume that p is a random permutation
- Sponge indifferentiable from random oracle:

$$\Delta_{\mathcal{D}}(\mathsf{sponge}, \mathsf{p}; \mathsf{ro}, \mathsf{sim}) \le N^2/2^{c+1}$$

  - N is number of permutation evaluations that attacker can make
  - Collisions in the inner part break security of the sponge
- Security of sponge truncated to n bits against classical attacks:

Collision resistance:	$N^2/2^{c+1} + N^2/2^{n+1}$
Second preimage resistance:	$N^2/2^{c+1} + N/2^n$
Preimage resistance:	$N^2/2^{c+1} + N/2^n$

distance from sponge to RO
(N is # primitive evaluations)

classical attacks against RO
(N is # oracle evaluations)

Indifferentiability of the Sponge [BDPV08]

- Assume that p is a random permutation
- Sponge indifferentiable from random oracle:

$$\Delta_{\mathcal{D}}(\mathsf{sponge}, \mathsf{p}; \mathsf{ro}, \mathsf{sim}) \leq N^2/2^{c+1}$$

  - N is number of permutation evaluations that attacker can make
  - Collisions in the inner part break security of the sponge
- Security of sponge truncated to n bits against classical attacks:

Collision resistance:	$N^2/2^{c+1} + N^2/2^{n+1}$	$\leftarrow$ attack in $\min\{2^{c/2}, 2^{n/2}\}$
Second preimage resistance:	$N^2/2^{c+1} + N/2^n$	$\leftarrow$ attack in $\min\{2^{c/2}, 2^n\}$
Preimage resistance:	$N^2/2^{c+1} + N/2^n$	

distance from sponge to RO
(N is # primitive evaluations)

classical attacks against RO
(N is # oracle evaluations)

Indifferentiability of the Sponge [BDPV08]

- Assume that p is a random permutation
- Sponge indifferentiable from random oracle:

$$\Delta_{\mathcal{D}}(\text{sponge}, \text{p}; \text{ro}, \text{sim}) \leq N^2/2^{c+1}$$

  - N is number of permutation evaluations that attacker can make
  - Collisions in the inner part break security of the sponge
- Security of sponge truncated to n bits against classical attacks:

Collision resistance:	$N^2/2^{c+1} + N^2/2^{n+1}$	$\leftarrow$ attack in $\min\{2^{c/2}, 2^{n/2}\}$
Second preimage resistance:	$N^2/2^{c+1} + N/2^n$	$\leftarrow$ attack in $\min\{2^{c/2}, 2^n\}$
Preimage resistance:	$N^2/2^{c+1} + N/2^n$	$\leftarrow$ attack in $\min\{2^{n-r} + 2^{c/2}, 2^n\}$

distance from sponge to RO
(N is # primitive evaluations)

classical attacks against RO
(N is # oracle evaluations)

Tightened Preimage Bound [LM22]

Tight Preimage Resistance

- Security proven up to $\approx \min \left\{2^{c/2}, 2^n \right.$ evaluations
- Best attack in $\approx \min\{2^{n-r} + 2^{c/2}, 2^n\}$ evaluations
- Gap if $c/2 \leq n - r$

Tightened Preimage Bound [LM22]

Tight Preimage Resistance

- Security proven up to $\approx \min \left\{2^{c/2}, 2^n \right.$ evaluations
- Best attack in $\approx \min\{2^{n-r} + 2^{c/2}, 2^n\}$ evaluations
- Gap if $c/2 \leq n - r$
- Lefevre and Mennink [LM22]: preimage resistance with bound

$$\mathcal{O}\left(\frac{q}{2^n} + \min\left\{\frac{q}{2^{n-r}}, \frac{q}{2^{c/2}}\right\}\right)$$

Tightened Preimage Bound [LM22]

Tight Preimage Resistance

- Security proven up to $\approx \min\left\{2^{c/2}, 2^{n}\right\}$ evaluations
- Best attack in $\approx \min\{2^{n-r} + 2^{c/2}, 2^{n}\}$ evaluations
- Gap if $c/2 \leq n - r$
- Lefevre and Mennink [LM22]: preimage resistance with bound

$$\mathcal{O}\left(\frac{q}{2^{n}} + \min\left\{\frac{q}{2^{n-r}}, \frac{q}{2^{c/2}}\right\}\right)$$

Implication for Ascon-Hash Mode with $(b, c, r, n) = (320, 256, 64, 256)$

- 128-bit collision resistance
- 128-bit second preimage resistance
- 192-bit preimage resistance

Keyed Sponges and Duplexes

Keyed Sponge

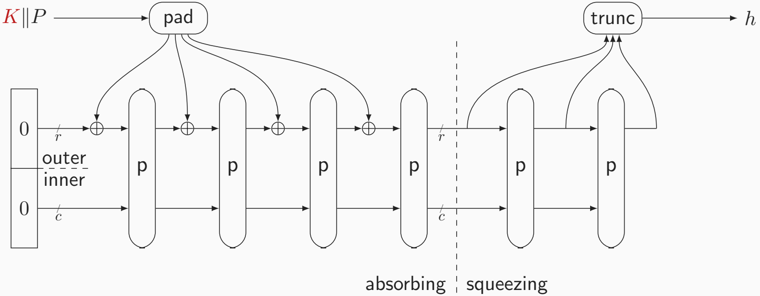
- $\text{PRF}(K, P) = \text{sponge}(K \| P)$
- Message authentication with tag size t: $\text{MAC}(K, P, t) = \text{sponge}(K \| P, t)$
- Keystream generation of length ℓ: $\text{SC}(K, D, \ell) = \text{sponge}(K \| D, \ell)$
- (All assuming K is fixed-length)

Keyed Sponge

- PRF$(K, P) =$ sponge$(K \| P)$
- Message authentication with tag size t: MAC$(K, P, t) =$ sponge$(K \| P, t)$
- Keystream generation of length ℓ: SC$(K, D, \ell) =$ sponge$(K \| D, \ell)$
- (All assuming K is fixed-length)

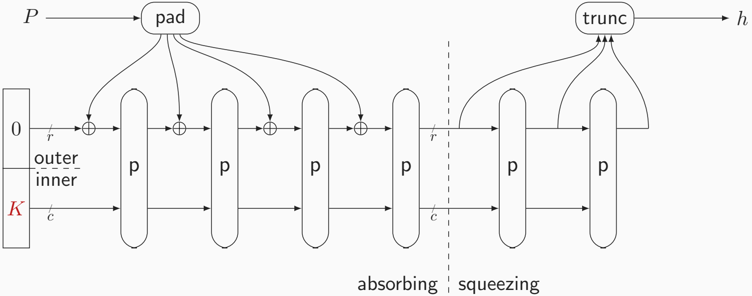
Keyed Duplex

- Authenticated encryption
- Multiple CAESAR and NIST LWC submissions



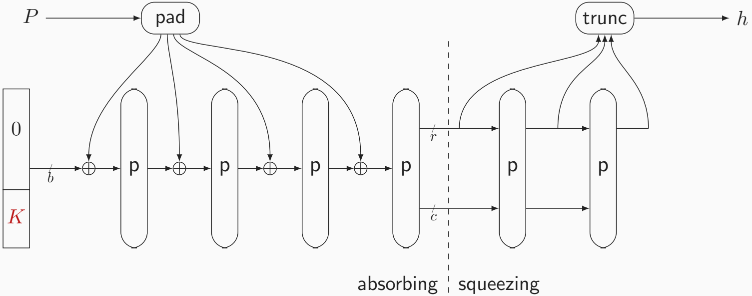
- Outer-Keyed Sponge [BDPV11b, ADMV15, NY16, Men18]

Evolution of Keyed Sponges



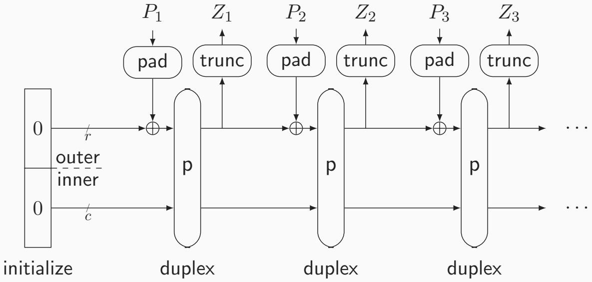
- Outer-Keyed Sponge [BDPV11b, ADMV15, NY16, Men18]
- Inner-Keyed Sponge [CDH$^+$12, ADMV15, NY16]

Evolution of Keyed Sponges



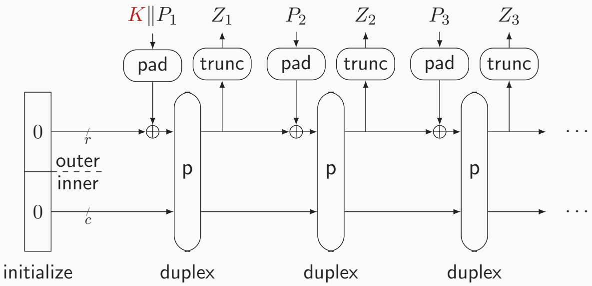
- Outer-Keyed Sponge [BDPV11b, ADMV15, NY16, Men18]
- Inner-Keyed Sponge [CDH$^+$12, ADMV15, NY16]
- Full-Keyed Sponge [BDPV12, GT16, MRV15]

Evolution of Keyed Duplexes



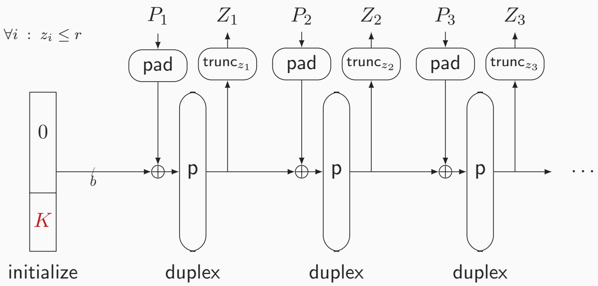
- Unkeyed Duplex [BDPV11a]

Evolution of Keyed Duplexes



- Unkeyed Duplex [BDPV11a]
- Outer-Keyed Duplex [BDPV11a]

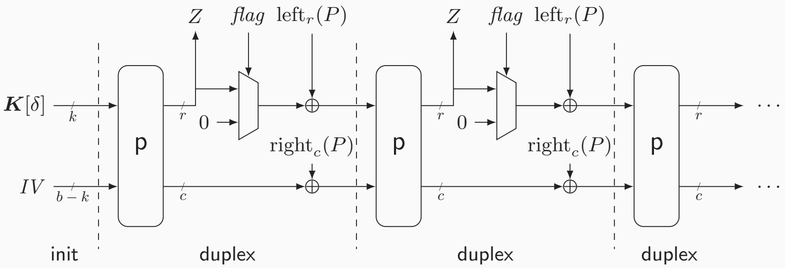
Evolution of Keyed Duplexes



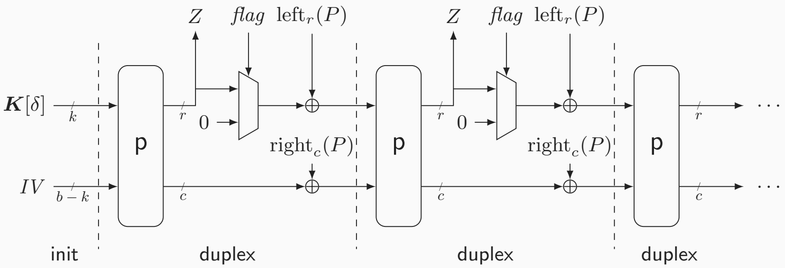
- Unkeyed Duplex [BDPV11a]
- Outer-Keyed Duplex [BDPV11a]
- Full-Keyed Duplex [MRV15, DMV17, DM19a, Men23]

Understanding the Duplex

Generalized Keyed Duplex ([DMV17, DM19a, Men23])



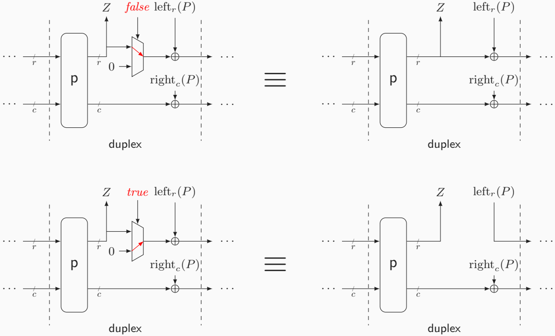
Generalized Keyed Duplex ([DMV17, DM19a, Men23])



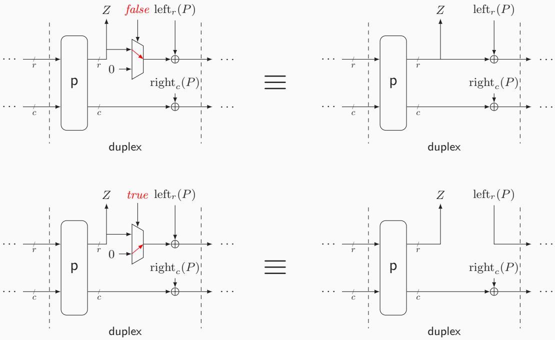
Features

- Multi-user by design: index δ specifies key in array
- Initial state: concatenation of $\boldsymbol{K}[\delta]$ and IV
- Full-state absorption, no padding
- Refined adversarial strength

Generalized Keyed Duplex: Flag (1)

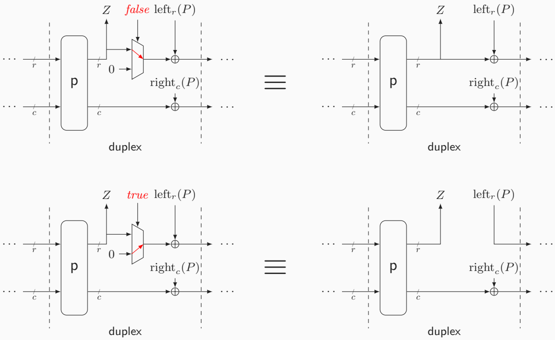


Generalized Keyed Duplex: Flag (1)



- Typical use case: authenticated encryption using duplex

Generalized Keyed Duplex: Flag (1)

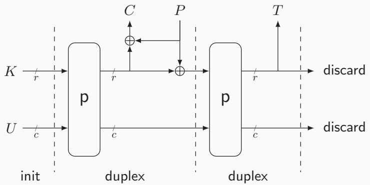


- Typical use case: authenticated encryption using duplex
- Security decreases for increasing number of calls with $\mathit{flag} = \mathit{true}$

- Consider extreme simplification of SpongeWrap authenticated encryption
- Key K, plaintext P, ciphertext C, and tag T all r bits; nonce U c bits
- General case will be discussed later in this presentation

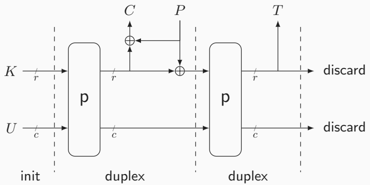
- Consider extreme simplification of SpongeWrap authenticated encryption
- Key K, plaintext P, ciphertext C, and tag T all r bits; nonce U c bits
- General case will be discussed later in this presentation

Encryption

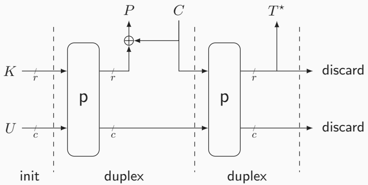


- Consider extreme simplification of SpongeWrap authenticated encryption
- Key K, plaintext P, ciphertext C, and tag T all r bits; nonce U c bits
- General case will be discussed later in this presentation

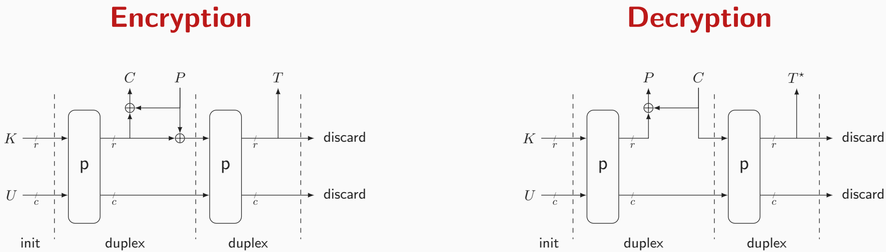
Encryption



Decryption



- Consider extreme simplification of SpongeWrap authenticated encryption
- Key K, plaintext P, ciphertext C, and tag T all r bits; nonce U c bits
- General case will be discussed later in this presentation



- Duplex call with $flag = true$ upon decryption
- Adversary can choose C and thus fix outer part to value of its choice

Security Model ([DMV17, DM19a, Men23])

Algorithm Keyed duplex construction $\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}$

Interface: KD.init
Input: $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{IV}$
Output: $\varnothing$

$S \leftarrow \mathrm{rot}_\alpha(\boldsymbol{K}[\delta] \parallel IV)$
return $\varnothing$

Interface: KD.duplex
Input: $(\mathit{flag}, P) \in \{\mathit{true}, \mathit{false}\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$

$S \leftarrow \mathsf{p}(S)$
$Z \leftarrow \mathrm{left}_r(S)$
$S \leftarrow S \oplus [\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P$
return Z

Security Model ([DMV17, DM19a, Men23])

Algorithm Keyed duplex construction $\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}$

Interface: KD.init
Input: $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{IV}$
Output: $\varnothing$

$S \leftarrow \mathrm{rot}_\alpha(\boldsymbol{K}[\delta] \parallel IV)$
return $\varnothing$

Interface: KD.duplex
Input: $(\mathit{flag}, P) \in \{\mathit{true}, \mathit{false}\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$

$S \leftarrow \mathsf{p}(S)$
$Z \leftarrow \mathrm{left}_r(S)$
$S \leftarrow S \oplus [\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P$
return Z

Algorithm Ideal extendable input function $\mathsf{IXIF}[\mathsf{ro}]$

Interface: IXIF.init
Input: $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{IV}$
Output: $\varnothing$

$\mathit{path} \leftarrow \mathrm{encode}[\delta] \parallel IV$
return $\varnothing$

Interface: IXIF.duplex
Input: $(\mathit{flag}, P) \in \{\mathit{true}, \mathit{false}\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$

$Z \leftarrow \mathsf{ro}(\mathit{path}, r)$
$\mathit{path} \leftarrow \mathit{path} \parallel ([\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P)$
return Z

Security Model ([DMV17, DM19a, Men23])

Algorithm Keyed duplex construction $\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}$

Interface: KD.init
Input: $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{IV}$
Output: $\varnothing$
$S \leftarrow \mathrm{rot}_\alpha(\boldsymbol{K}[\delta] \parallel IV)$
return $\varnothing$

Interface: KD.duplex
Input: $(\mathit{flag}, P) \in \{\mathit{true}, \mathit{false}\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$
$S \leftarrow \mathsf{p}(S)$
$Z \leftarrow \mathrm{left}_r(S)$
$S \leftarrow S \oplus [\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P$
return Z

Algorithm Ideal extendable input function $\mathsf{IXIF}[\mathsf{ro}]$

Interface: IXIF.init
Input: $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{IV}$
Output: $\varnothing$
$\mathit{path} \leftarrow \mathrm{encode}[\delta] \parallel IV$
return $\varnothing$

Interface: IXIF.duplex
Input: $(\mathit{flag}, P) \in \{\mathit{true}, \mathit{false}\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$
$Z \leftarrow \mathsf{ro}(\mathit{path}, r)$
$\mathit{path} \leftarrow \mathit{path} \parallel ([\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P)$
return Z

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}, \mathsf{p}^{\pm} \; ; \; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$$

Security Model ([DMV17, DM19a, Men23])

Algorithm Keyed duplex construction $\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}$

Interface: KD.init
Input: $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{IV}$
Output: $\varnothing$
  $S \leftarrow \mathrm{rot}_\alpha(\boldsymbol{K}[\delta] \parallel IV)$
  return $\varnothing$

Interface: KD.duplex
Input: $(\mathit{flag}, P) \in \{\mathit{true}, \mathit{false}\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$
  $S \leftarrow \mathsf{p}(S)$
  $Z \leftarrow \mathrm{left}_r(S)$
  $S \leftarrow S \oplus [\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P$
  return Z

Algorithm Ideal extendable input function $\mathsf{IXIF}[\mathsf{ro}]$

Interface: IXIF.init
Input: $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{IV}$
Output: $\varnothing$
  $\mathit{path} \leftarrow \mathrm{encode}[\delta] \parallel IV$
  return $\varnothing$

Interface: IXIF.duplex
Input: $(\mathit{flag}, P) \in \{\mathit{true}, \mathit{false}\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$
  $Z \leftarrow \mathsf{ro}(\mathit{path}, r)$
  $\mathit{path} \leftarrow \mathit{path} \parallel ([\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P)$
  return Z

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}, \mathsf{p}^{\pm} \; ; \; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$$

- $\mathsf{IXIF}[\mathsf{ro}]$ is basically random oracle in disguise

Security Model ([DMV17, DM19a, Men23])

Algorithm Keyed duplex construction $\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}$

Interface: KD.init
Input: $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{IV}$
Output: $\varnothing$

$S \leftarrow \mathrm{rot}_\alpha(\boldsymbol{K}[\delta] \parallel IV)$
return $\varnothing$

Interface: KD.duplex
Input: $(\mathit{flag}, P) \in \{\mathit{true}, \mathit{false}\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$

$S \leftarrow \mathsf{p}(S)$
$Z \leftarrow \mathrm{left}_r(S)$
$S \leftarrow S \oplus [\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P$
return Z

Algorithm Ideal extendable input function $\mathsf{IXIF}[\mathsf{ro}]$

Interface: IXIF.init
Input: $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{IV}$
Output: $\varnothing$

$\mathit{path} \leftarrow \mathrm{encode}[\delta] \parallel IV$
return $\varnothing$

Interface: IXIF.duplex
Input: $(\mathit{flag}, P) \in \{\mathit{true}, \mathit{false}\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$

$Z \leftarrow \mathsf{ro}(\mathit{path}, r)$
$\mathit{path} \leftarrow \mathit{path} \parallel ([\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P)$
return Z

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}, \mathsf{p}^{\pm} \;;\; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$$

- $\mathsf{IXIF}[\mathsf{ro}]$ is basically random oracle in disguise
- If $\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}$ is hard to distinguish from $\mathsf{IXIF}[\mathsf{ro}]$ for certain bound on adversarial resources, $\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}$ roughly "behaves like" random oracle

Security Model ([DMV17, DM19a, Men23])

Algorithm Keyed duplex construction $\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}$

Interface: KD.init
Input: $(\delta, IV) \in \{1,\dots,\mu\} \times \mathcal{IV}$
Output: $\varnothing$
  $S \leftarrow \mathrm{rot}_\alpha(\boldsymbol{K}[\delta] \parallel IV)$
  return $\varnothing$

Interface: KD.duplex
Input: $(\mathit{flag}, P) \in \{\mathit{true}, \mathit{false}\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$
  $S \leftarrow \mathsf{p}(S)$
  $Z \leftarrow \mathrm{left}_r(S)$
  $S \leftarrow S \oplus [\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P$
  return Z

Algorithm Ideal extendable input function $\mathsf{IXIF}[\mathsf{ro}]$

Interface: IXIF.init
Input: $(\delta, IV) \in \{1,\dots,\mu\} \times \mathcal{IV}$
Output: $\varnothing$
  $\mathit{path} \leftarrow \mathrm{encode}[\delta] \parallel IV$
  return $\varnothing$

Interface: IXIF.duplex
Input: $(\mathit{flag}, P) \in \{\mathit{true}, \mathit{false}\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$
  $Z \leftarrow \mathsf{ro}(\mathit{path}, r)$
  $\mathit{path} \leftarrow \mathit{path} \parallel ([\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P)$
  return Z

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}, \mathsf{p}^{\pm} \; ; \; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$$

- $\mathsf{IXIF}[\mathsf{ro}]$ is basically random oracle in disguise
- If $\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}$ is hard to distinguish from $\mathsf{IXIF}[\mathsf{ro}]$ for certain bound on adversarial resources, $\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}$ roughly “behaves like” random oracle
- Bound on adversarial resources is in turn determined by use case!

Security Bounds From [DMV17] and [DM19a]

- M: data complexity (calls to construction)
- N: time complexity (calls to primitive)
- Q: number of init calls
- Q_{IV}: max # init calls for single IV
- L: # queries with repeated path (e.g., nonce-violation)
- Ω: # queries with overwriting outer part (e.g., RUP)
- $\nu_{r,c}^{M}$: some multicollision coefficient (often small)

Simplified Security Bound

$$\frac{Q_{IV}N}{2^k} + \frac{(L + \Omega + \nu_{r,c}^{M})N}{2^c}$$

Security Bounds From [DMV17] and [DM19a]

- M: data complexity (calls to construction)
- N: time complexity (calls to primitive)
- Q: number of init calls
- Q_{IV}: max # init calls for single IV
- L: # queries with repeated path (e.g., nonce-violation)
- Ω: # queries with overwriting outer part (e.g., RUP)
- $\nu_{r,c}^{M}$: some multicollision coefficient (often small)

Simplified Security Bound

$$\frac{Q_{IV}N}{2^k} + \frac{(L+\Omega+\nu_{r,c}^{M})N}{2^c}$$

Actual Security Bounds (Retained)

- [DMV17]:

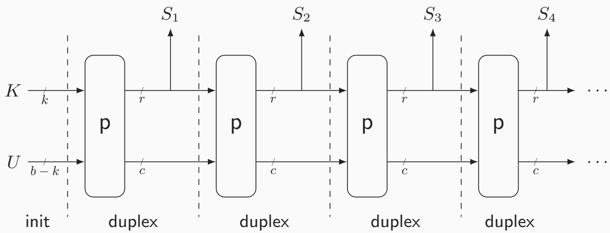
$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) \leq \frac{(L+\Omega)N}{2^c} + \frac{2\nu_{r,c}^{2(M-L)}(N+1)}{2^c} + \frac{\binom{L+\Omega+1}{2}}{2^c} + \frac{(M-L-Q)Q}{2^b-Q} + \frac{M(M-L-1)}{2^b} + \frac{Q(M-L-Q)}{2^{\min\{c+k,\max\{b-\alpha,c\}\}}} + \frac{Q_{IV}N}{2^k} + \frac{\binom{\mu}{2}}{2^k}$$

- [DM19a] (with one simplification):

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) \leq \frac{(L+\Omega)N}{2^c} + \frac{2\nu_{r,c}^{M}(N+1)}{2^c} + \frac{\nu_{r,c}^{M}(L+\Omega)+\binom{L+\Omega}{2}}{2^c} + \frac{\binom{M-L-Q}{2}+(M-L-Q)(L+\Omega)}{2^b} + \frac{\binom{M+N}{2}+\binom{N}{2}}{2^b} + \frac{Q(M-Q)}{2^{\min\{c+k,\max\{b-\alpha,c\}\}}} + \frac{Q_{IV}N}{2^k} + \frac{\binom{\mu}{2}}{2^k}$$

Duplex Application: Keystream Generation

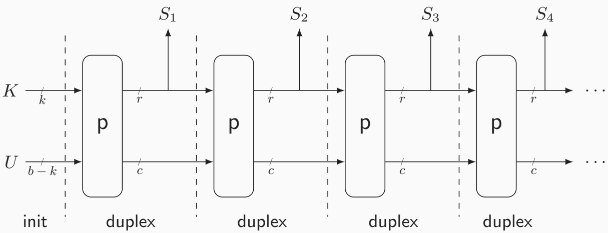
Keystream Generation



- Input: key K, nonce U
- Output: keystream S of requested length

Algorithm Keystream generation SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \mathbb{N}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: KD[p]$_{(K)}$
  $S \leftarrow \varnothing$
  KD.init$(1, U)$
  for $i = 1, \ldots, \lceil \ell/r \rceil$ **do**
    $S \leftarrow S \parallel$ KD.duplex$(\mathit{false}, 0^b)$
  return $\mathrm{left}_\ell(S)$



- Input: key K, nonce U
- Output: keystream S of requested length
- Keystream generation can be described using duplex

Algorithm Keystream generation SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \mathbb{N}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$

```
S ← ∅
KD.init(1, U)
for i = 1, ..., ⌈ℓ/r⌉ do
    S ← S || KD.duplex(false, 0^b)
return left_ℓ(S)
```

Keystream Generation: Security (1)

- Consider distinguisher D against PRF security of SC[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) $+$ N primitive queries

Keystream Generation: Security (1)

- Consider distinguisher D against PRF security of SC[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{p}]_K, \mathsf{p}^{\pm} \; ; \; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) $+$ N primitive queries
- $\mathsf{SC}[\mathsf{p}]_K$ is basically just $\mathsf{SC}[\mathsf{KD}[\mathsf{p}]_K]$

- Consider distinguisher D against PRF security of SC[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) $+$ N primitive queries
- $\mathsf{SC}[\mathsf{p}]_K$ is basically just $\mathsf{SC}[\mathsf{KD}[\mathsf{p}]_K]$
- Triangle inequality:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(\mathsf{D}) &= \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right) \\
&= \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{KD}[\mathsf{p}]_K], \mathsf{p}^{\pm} \;;\; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right) \\
&\leq \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{KD}[\mathsf{p}]_K], \mathsf{p}^{\pm} \;;\; \mathsf{SC}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{p}^{\pm}\right) + \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{p}^{\pm} \;;\; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)
\end{aligned}$$

- Consider distinguisher D against PRF security of SC[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) + N primitive queries
- $\mathsf{SC}[\mathsf{p}]_K$ is basically just $\mathsf{SC}[\mathsf{KD}[\mathsf{p}]_K]$
- Triangle inequality:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(\mathsf{D}) &= \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right) \\
&= \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{KD}[\mathsf{p}]_K], \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right) \\
&\le \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{KD}[\mathsf{p}]_K], \mathsf{p}^{\pm}\ ;\ \mathsf{SC}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{p}^{\pm}\right) + \underbrace{\Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)}_{=0}
\end{aligned}$$

Keystream Generation: Security (1)

- Consider distinguisher D against PRF security of SC[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) + N primitive queries
- $\mathsf{SC}[\mathsf{p}]_K$ is basically just $\mathsf{SC}[\mathsf{KD}[\mathsf{p}]_K]$
- Triangle inequality:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(\mathsf{D}) &= \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right) \\
&= \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{KD}[\mathsf{p}]_K], \mathsf{p}^{\pm} \;;\; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right) \\
&\leq \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{KD}[\mathsf{p}]_K], \mathsf{p}^{\pm} \;;\; \mathsf{SC}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{p}^{\pm}\right) + \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{p}^{\pm} \;;\; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)
\end{aligned}$$

$$\leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right) \qquad = 0$$

Keystream Generation: Security (1)

- Consider distinguisher D against PRF security of SC[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

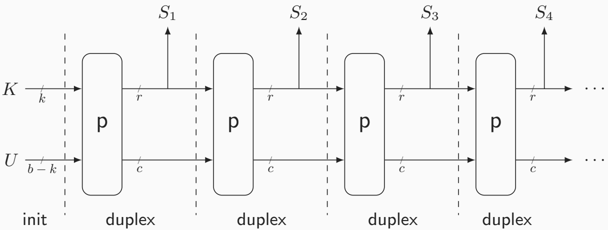
- D can make q construction queries (total σ blocks) $+$ N primitive queries
- $\mathsf{SC}[\mathsf{p}]_K$ is basically just $\mathsf{SC}[\mathsf{KD}[\mathsf{p}]_K]$
- Triangle inequality:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(\mathsf{D}) &= \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)\\ &= \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{KD}[\mathsf{p}]_K], \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)\\ &\le \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{KD}[\mathsf{p}]_K], \mathsf{p}^{\pm}\ ;\ \mathsf{SC}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{p}^{\pm}\right) + \Delta_{\mathsf{D}}\left(\mathsf{SC}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)\end{aligned}$$

$$\le \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right) \qquad = 0$$

- What are the resources of D′?

Keystream Generation: Security (2)

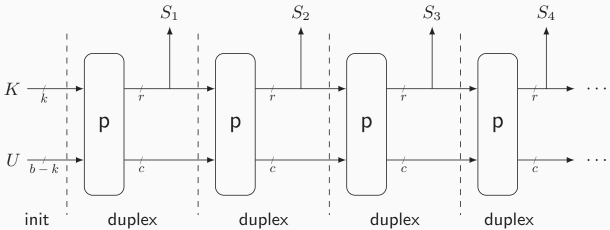


Algorithm Keystream generation SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \mathbb{N}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: KD[p]$_{(K)}$
  $S \leftarrow \varnothing$
  KD.init$(1, U)$
  for $i = 1, \ldots, \lceil \ell / r \rceil$ **do**
    $S \leftarrow S \parallel$ KD.duplex$(\mathit{false}, 0^b)$
  return $\text{left}_\ell(S)$

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)		
N: time complexity (calls to primitive)		
Q: number of init calls		
Q_{IV}: max # init calls for single IV		
L: # queries with repeated path		
Ω: # queries with overwriting outer part		

Keystream Generation: Security (2)



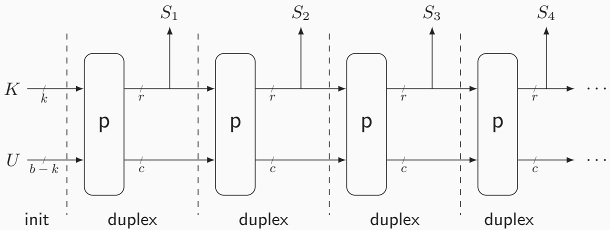
Algorithm Keystream generation SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \mathbb{N}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$

$S \leftarrow \varnothing$
$\mathsf{KD.init}(1, U)$
for $i = 1, \ldots, \lceil \ell / r \rceil$ **do**
  $S \leftarrow S \parallel \mathsf{KD.duplex}(\mathit{false}, 0^b)$
return $\mathrm{left}_\ell(S)$

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)		
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls		
Q_{IV}: max # init calls for single IV		
L: # queries with repeated path		
Ω: # queries with overwriting outer part		

Keystream Generation: Security (2)

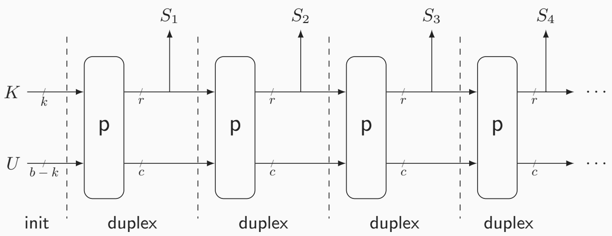


Algorithm Keystream generation SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \mathbb{N}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$

$S \leftarrow \varnothing$
$\mathsf{KD.init}(1, U)$
for $i = 1, \ldots, \lceil \ell/r \rceil$ **do**
  $S \leftarrow S \parallel \mathsf{KD.duplex}(\mathit{false}, 0^b)$
return $\mathrm{left}_\ell(S)$

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	σ
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV		
L: # queries with repeated path		
Ω: # queries with overwriting outer part		

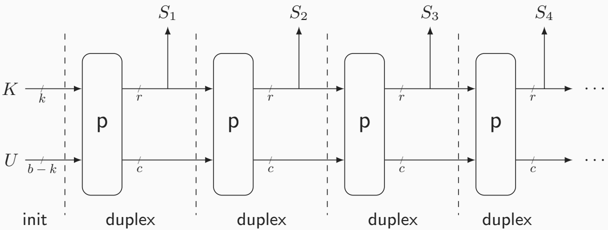


Algorithm Keystream generation SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \mathbb{N}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: KD[p]$_{(K)}$

$S \leftarrow \varnothing$
KD.init$(1, U)$
for $i = 1, \ldots, \lceil \ell/r \rceil$ **do**
  $S \leftarrow S \parallel$ KD.duplex$(\mathit{false}, 0^b)$
return $\text{left}_\ell(S)$

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	σ
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path		
Ω: # queries with overwriting outer part		



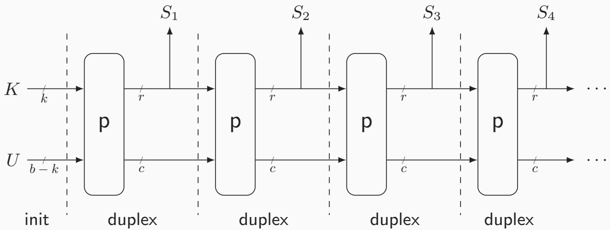
Algorithm Keystream generation SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \mathbb{N}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$

$S \leftarrow \varnothing$
$\mathsf{KD.init}(1, U)$
for $i = 1, \ldots, \lceil \ell/r \rceil$ **do**
  $S \leftarrow S \,\|\, \mathsf{KD.duplex}(\mathit{false}, 0^b)$
return $\mathrm{left}_\ell(S)$

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	σ
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	0
Ω: # queries with overwriting outer part		

Keystream Generation: Security (2)



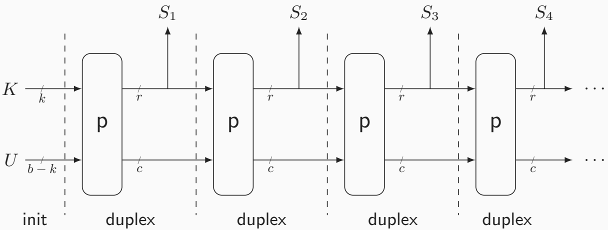
Algorithm Keystream generation SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \mathbb{N}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$

$S \leftarrow \varnothing$
$\mathsf{KD.init}(1, U)$
for $i = 1, \ldots, \lceil \ell / r \rceil$ **do**
  $S \leftarrow S \parallel \mathsf{KD.duplex}(\mathit{false}, 0^b)$
return $\mathrm{left}_\ell(S)$

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	σ
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	0
Ω: # queries with overwriting outer part	$\longrightarrow$	0

Keystream Generation: Security (2)



Algorithm Keystream generation SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \mathbb{N}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$

$S \leftarrow \varnothing$
$\mathsf{KD.init}(1, U)$
for $i = 1, \ldots, \lceil \ell / r \rceil$ **do**
  $S \leftarrow S \parallel \mathsf{KD.duplex}(\mathit{false}, 0^b)$
return $\mathrm{left}_\ell(S)$

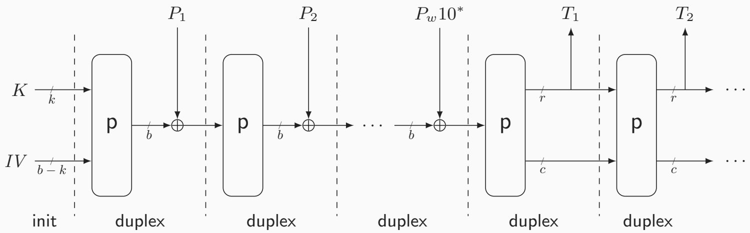
resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	σ
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	0
Ω: # queries with overwriting outer part	$\longrightarrow$	0

From [DMV17] (in single-user setting):

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}') \leq \frac{2\nu_{r,c}^{2\sigma}(N+1)}{2^c} + \frac{(\sigma-q)q}{2^b-q} + \frac{2\binom{\sigma}{2}}{2^b} + \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{N}{2^k}$$

Duplex Application: Message Authentication and Ascon-PRF

Full-State Keyed Sponge [BDPV12]



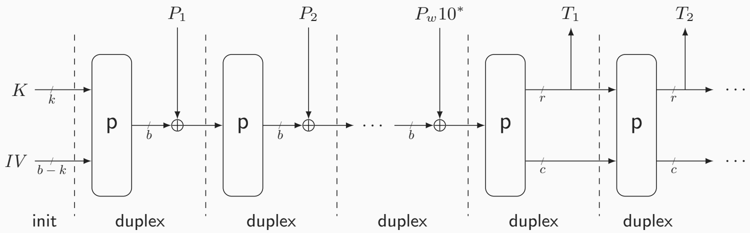
- Input: key K, initial value IV, message P
- Output: tag T

Algorithm Full-state keyed sponge FSKS[p]

Input: $(K, IV, P) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
Underlying keyed duplex: KD[p]$_{(K)}$

  $(P_1, P_2, \ldots, P_w) \leftarrow \mathrm{pad}_b^{10^*}(P)$
  $T \leftarrow \varnothing$
  KD.init$(1, IV)$
  for $i = 1, \ldots, w$ **do**
    KD.duplex(false, P_i) ▷ discard output
  for $i = 1, \ldots, \lceil t/r \rceil$ **do**
    $T \leftarrow T \,\|\,$ KD.duplex$(\mathit{false}, 0^b)$
  return $\mathrm{left}_t(T)$

Full-State Keyed Sponge [BDPV12]



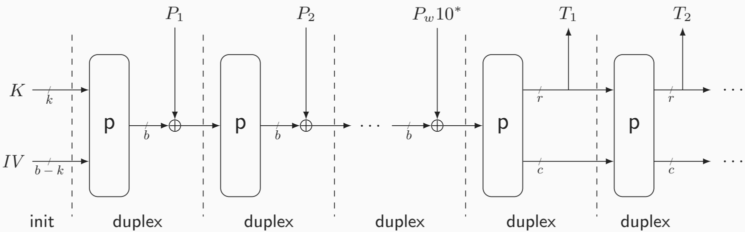
- Input: key K, initial value IV, message P
- Output: tag T
- Analysis of [MRV15] applies

Algorithm Full-state keyed sponge FSKS[p]

Input: $(K, IV, P) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
Underlying keyed duplex: KD[p]$_{(K)}$

$(P_1, P_2, \ldots, P_w) \leftarrow \mathrm{pad}_b^{10^*}(P)$
$T \leftarrow \varnothing$
KD.init$(1, IV)$
for $i = 1, \ldots, w$ **do**
KD.duplex(false, P_i) ▷ discard output
for $i = 1, \ldots, \lceil t/r \rceil$ **do**
$T \leftarrow T \parallel$ KD.duplex$(\mathit{false}, 0^b)$
return $\mathrm{left}_t(T)$

Full-State Keyed Sponge [BDPV12]

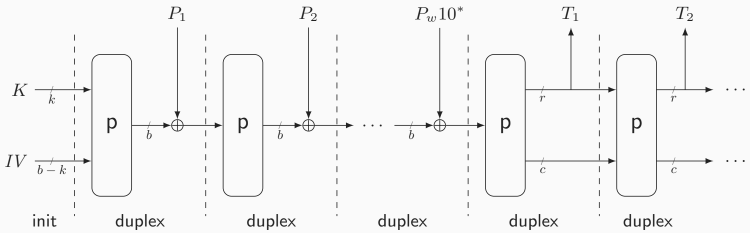


- Input: key K, initial value IV, message P
- Output: tag T
- Analysis of [MRV15] applies
- PRF security of FSKS[p]:
  - Comparable analysis as for SC[p]

Algorithm Full-state keyed sponge FSKS[p]

Input: $(K, IV, P) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
Underlying keyed duplex: KD[p]$_{(K)}$

$(P_1, P_2, \ldots, P_w) \leftarrow \mathrm{pad}_b^{10^*}(P)$
$T \leftarrow \varnothing$
KD.init$(1, IV)$
for $i = 1, \ldots, w$ **do**
  KD.duplex(false, P_i) ▷ discard output
for $i = 1, \ldots, \lceil t/r \rceil$ **do**
  $T \leftarrow T \parallel$ KD.duplex$(\mathit{false}, 0^b)$
return $\mathrm{left}_t(T)$



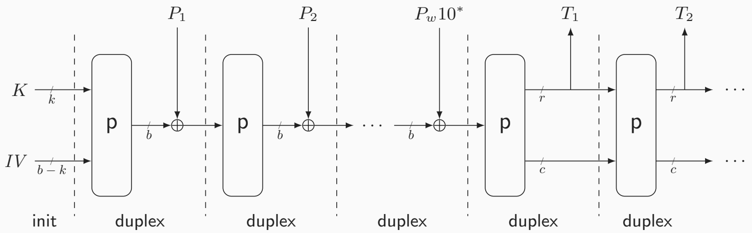
- Input: key K, initial value IV, message P
- Output: tag T
- Analysis of [MRV15] applies
- PRF security of FSKS[p]:
  - Comparable analysis as for SC[p]
  - …but distinguisher can repeat paths

Algorithm Full-state keyed sponge FSKS[p]

Input: $(K, IV, P) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
Underlying keyed duplex: KD[p]$_{(K)}$

$(P_1, P_2, \ldots, P_w) \leftarrow \mathrm{pad}_b^{10^*}(P)$
$T \leftarrow \varnothing$
KD.init$(1, IV)$
for $i = 1, \ldots, w$ **do**
  KD.duplex(false, P_i) ▷ discard output
for $i = 1, \ldots, \lceil t/r \rceil$ **do**
  $T \leftarrow T \parallel$ KD.duplex$(\mathit{false}, 0^b)$
return $\mathrm{left}_t(T)$

Full-State Keyed Sponge [BDPV12]



- Input: key K, initial value IV, message P
- Output: tag T
- Analysis of [MRV15] applies
- PRF security of FSKS[p]:
  - Comparable analysis as for SC[p]
  - ... but distinguisher can repeat paths
  - Impacts resources of D′

Algorithm Full-state keyed sponge FSKS[p]

Input: $(K, IV, P) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
Underlying keyed duplex: KD[p]$_{(K)}$
  $(P_1, P_2, \ldots, P_w) \leftarrow \mathrm{pad}_b^{10^*}(P)$
  $T \leftarrow \varnothing$
  KD.init$(1, IV)$
  for $i = 1, \ldots, w$ **do**
    KD.duplex(false, P_i) ▷ discard output
  for $i = 1, \ldots, \lceil t/r \rceil$ **do**
    $T \leftarrow T \,\|\,$ KD.duplex$(\mathit{false}, 0^b)$
  return $\mathrm{left}_t(T)$

Full-State Keyed Sponge: Security

- Consider distinguisher D against PRF security of FSKS[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{FSKS}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) $+$ N primitive queries

Full-State Keyed Sponge: Security

- Consider distinguisher D against PRF security of FSKS[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{FSKS}[\mathsf{p}]_K, \mathsf{p}^{\pm} \; ; \; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) + N primitive queries
- Triangle inequality: $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm} \; ; \; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$

Full-State Keyed Sponge: Security

- Consider distinguisher D against PRF security of FSKS[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{FSKS}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) $+$ N primitive queries
- Triangle inequality: $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$
- What are the resources of D′?

Full-State Keyed Sponge: Security

- Consider distinguisher D against PRF security of FSKS[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{FSKS}[\mathsf{p}]_K, \mathsf{p}^{\pm} \; ; \; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) + N primitive queries
- Triangle inequality: $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm} \; ; \; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$
- What are the resources of D′?

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)		
N: time complexity (calls to primitive)		
Q: number of init calls		
Q_{IV}: max # init calls for single IV		
L: # queries with repeated path		
Ω: # queries with overwriting outer part		

Full-State Keyed Sponge: Security

- Consider distinguisher D against PRF security of FSKS[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{FSKS}[\mathsf{p}]_K, \mathsf{p}^{\pm} \; ; \; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) + N primitive queries
- Triangle inequality: $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm} \; ; \; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$
- What are the resources of D′?

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	σ
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path		
Ω: # queries with overwriting outer part		

Full-State Keyed Sponge: Security

- Consider distinguisher D against PRF security of FSKS[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{FSKS}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) $+$ N primitive queries
- Triangle inequality: $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$
- What are the resources of D′?

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	σ
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path		
Ω: # queries with overwriting outer part	$\longrightarrow$	0

Full-State Keyed Sponge: Security

- Consider distinguisher D against PRF security of FSKS[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{FSKS}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) + N primitive queries
- Triangle inequality: $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$
- What are the resources of D′?

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	σ
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q-1$
Ω: # queries with overwriting outer part	$\longrightarrow$	0

- Consider distinguisher D against PRF security of FSKS[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{FSKS}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) $+$ N primitive queries
- Triangle inequality: $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$
- What are the resources of D′?

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	σ
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q-1$
Ω: # queries with overwriting outer part	$\longrightarrow$	0

From [DMV17] (in single-user setting):

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}') \leq \frac{2\nu_{r,c}^{2\sigma}(N+1)}{2^c} + \frac{(q-1)N + \binom{q}{2}}{2^c} + \frac{(\sigma-q)q}{2^b-q} + \frac{2\binom{\sigma}{2}}{2^b} + \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{N}{2^k}$$

Full-State Keyed Sponge: Security

- Consider distinguisher D against PRF security of FSKS[p]

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{FSKS}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right)$$

- D can make q construction queries (total σ blocks) $+$ N primitive queries
- Triangle inequality: $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FSKS}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$
- What are the resources of D′?

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	σ
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q-1$
Ω: # queries with overwriting outer part	$\longrightarrow$	0

From [DMV17] (in single-user setting):

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}') \leq \frac{2\nu_{r,c}^{2\sigma}(N+1)}{2^c} + \frac{(q-1)N+\binom{q}{2}}{2^c} + \frac{(\sigma-q)q}{2^b-q} + \frac{2\binom{\sigma}{2}}{2^b} + \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{N}{2^k}$$

influence of L

Full-State Keyed Sponge: Adversarial Power in Influencing Outer Part

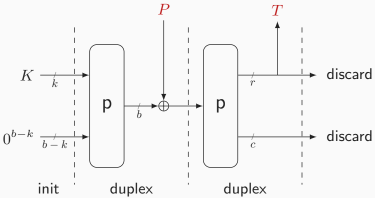
- Repeated paths (i.e., large L) can seriously affect security

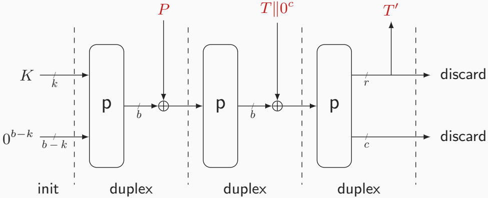
Full-State Keyed Sponge: Adversarial Power in Influencing Outer Part

- Repeated paths (i.e., large L) can seriously affect security
- Consider simplified FSKS$[\mathsf{p}]$: no IV, no padding, r-bit tag

Full-State Keyed Sponge: Adversarial Power in Influencing Outer Part

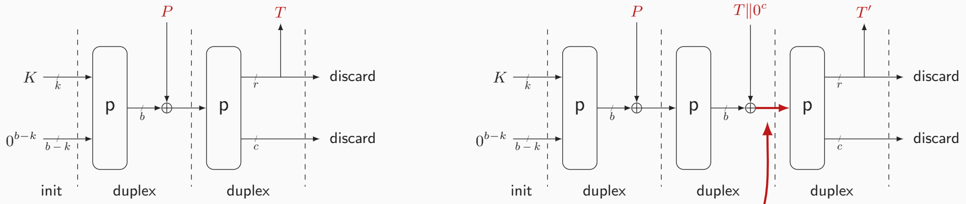
- Repeated paths (i.e., large L) can seriously affect security
- Consider simplified FSKS[p]: no IV, no padding, r-bit tag
- Distinguisher makes two queries: $P \mapsto T$ and $P\|T\|0^c \mapsto T'$





Full-State Keyed Sponge: Adversarial Power in Influencing Outer Part

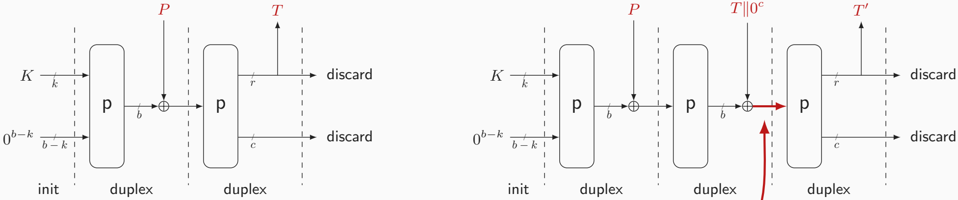
- Repeated paths (i.e., large L) can seriously affect security
- Consider simplified FSKS[p]: no IV, no padding, r-bit tag
- Distinguisher makes two queries: $P \mapsto T$ and $P \| T \| 0^c \mapsto T'$



- State of second query before squeezing equals $0^r \| *^c$

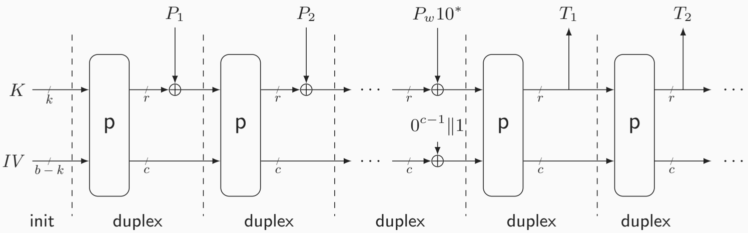
Full-State Keyed Sponge: Adversarial Power in Influencing Outer Part

- Repeated paths (i.e., large L) can seriously affect security
- Consider simplified FSKS[p]: no IV, no padding, r-bit tag
- Distinguisher makes two queries: $P \mapsto T$ and $P \| T \| 0^c \mapsto T'$



- State of second query before squeezing equals $0^r \| *^c$
- Key recovery attack:
  - Make q twin queries as above and N primitive queries of form $0^r \| *^c$
  - Construction-primitive collision (likely if $\frac{q \cdot N}{2^c} \approx 1$) $\longrightarrow$ derive K

Ascon-PRF [DEMS21]



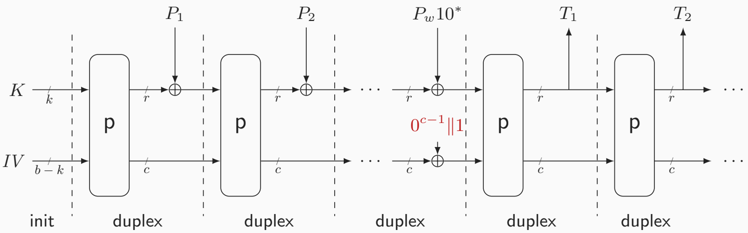
- Input: key K, initial value IV, message P
- Output: tag T

Algorithm Ascon-PRF[p]

Input: $(K, IV, P) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
Underlying keyed duplex: KD[p]$_{(K)}$

$(P_1, P_2, \ldots, P_w) \leftarrow \text{pad}_r^{10^*}(P)$
$T \leftarrow \varnothing$
KD.init$(1, IV)$
for $i = 1, \ldots, w-1$ **do**
  KD.duplex(false, P_i) ▷ discard output
KD.duplex$(\mathit{false}, P_w \| 0^{c-1}1)$
for $i = 1, \ldots, \lceil t/r \rceil$ **do**
  $T \leftarrow T \;\|$ KD.duplex$(\mathit{false}, 0^b)$
return $\text{left}_t(T)$

Ascon-PRF [DEMS21]

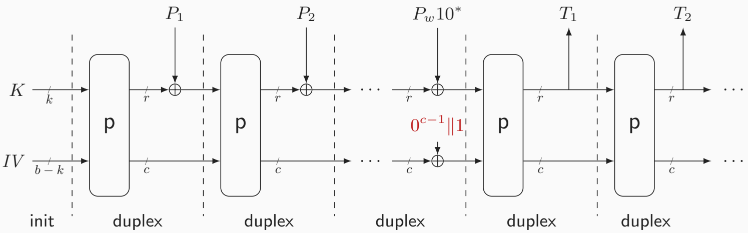


- Input: key K, initial value IV, message P
- Output: tag T
- Domain separation solves problem of repeated paths

Algorithm Ascon-PRF[p]

Input: $(K, IV, P) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
Underlying keyed duplex: KD[p]$_{(K)}$

$(P_1, P_2, \ldots, P_w) \leftarrow \mathrm{pad}_r^{10^*}(P)$
$T \leftarrow \varnothing$
KD.init$(1, IV)$
for $i = 1, \ldots, w-1$ **do**
  KD.duplex(false, P_i) ▷ discard output
KD.duplex$(\mathit{false}, P_w \| 0^{c-1}1)$
for $i = 1, \ldots, \lceil t/r \rceil$ **do**
  $T \leftarrow T \,\|\,$ KD.duplex$(\mathit{false}, 0^b)$
return $\mathrm{left}_t(T)$



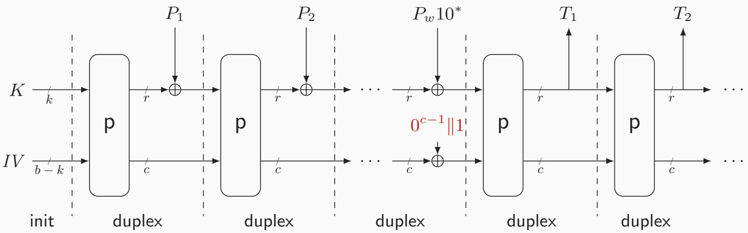
- Input: key K, initial value IV, message P
- Output: tag T
- Domain separation solves problem of repeated paths
  - Repeated paths may still occur...

Algorithm Ascon-PRF[p]

Input: $(K, IV, P) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
Underlying keyed duplex: KD[p]$_{(K)}$

```
(P_1, P_2, ..., P_w) ← pad_r^{10*}(P)
T ← ∅
KD.init(1, IV)
for i = 1, ..., w − 1 do
    KD.duplex(false, P_i)                ▷ discard output
KD.duplex(false, P_w || 0^{c−1} 1)
for i = 1, ..., ⌈t/r⌉ do
    T ← T || KD.duplex(false, 0^b)
return left_t(T)
```

Ascon-PRF [DEMS21]



- Input: key K, initial value IV, message P
- Output: tag T
- Domain separation solves problem of repeated paths
  - Repeated paths may still occur. . .
  - . . . but adversary cannot exploit them

Algorithm Ascon-PRF[p]

Input: $(K, IV, P) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
Underlying keyed duplex: KD[p]$_{(K)}$

$(P_1, P_2, \ldots, P_w) \leftarrow \mathrm{pad}_r^{10^*}(P)$
$T \leftarrow \varnothing$
KD.init$(1, IV)$
for $i = 1, \ldots, w-1$ **do**
  KD.duplex(false, P_i) ▷ discard output
KD.duplex$(\mathit{false}, P_w \| 0^{c-1}1)$
for $i = 1, \ldots, \lceil t/r \rceil$ **do**
  $T \leftarrow T \;\|$ KD.duplex$(\mathit{false}, 0^b)$
return $\mathrm{left}_t(T)$

Ascon-PRF: Security

- Unfortunately, (bounds on) the resources of D′ do not change:

Ascon-PRF: Security

- Unfortunately, (bounds on) the resources of D′ do not change:

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	σ
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q-1$
Ω: # queries with overwriting outer part	$\longrightarrow$	0

Ascon-PRF: Security

- Unfortunately, (bounds on) the resources of D′ do not change:

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	σ
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q-1$
Ω: # queries with overwriting outer part	$\longrightarrow$	0

- Improved bound from [DMV17]:
  - Loose bounding in original proof
  - Resolving this loose bounding makes $\frac{(q-1)N+\binom{q}{2}}{2^c}$ vanish

Ascon-PRF: Security

- Unfortunately, (bounds on) the resources of D′ do not change:

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	σ
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q-1$
Ω: # queries with overwriting outer part	$\longrightarrow$	0

- Improved bound from [DMV17]:
  - Loose bounding in original proof
  - Resolving this loose bounding makes $\frac{(q-1)N+\binom{q}{2}}{2^c}$ vanish
- Improved bound from [DM19a]:
  - Defines additional parameter $\nu_{\text{fix}} \leq L + \Omega$
  - In most cases $\nu_{\text{fix}} = L + \Omega$; for current case $\nu_{\text{fix}} = 0$
  - Dominant term $\frac{(q-1)N+\binom{q}{2}}{2^c}$ never appears in the first place

Ascon-PRF: Implication

Multi-user bound from [DMV17]

$$\mathbf{Adv}^{\mu\text{-prf}}_{\mathsf{Ascon\text{-}PRF}}(\mathsf{D}) \leq \frac{2\nu_{r,c}^{2\sigma}(N+1)}{2^c} + \frac{(\sigma-q)q}{2^b-q} + \frac{2\binom{\sigma}{2}}{2^b} + \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}$$

Ascon-PRF: Implication

Multi-user bound from [DMV17]

$$\mathbf{Adv}^{\mu\text{-prf}}_{\text{Ascon-PRF}}(\mathsf{D}) \leq \frac{2\nu^{2\sigma}_{r,c}(N+1)}{2^c} + \frac{(\sigma-q)q}{2^b-q} + \frac{2\binom{\sigma}{2}}{2^b} + \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}$$

Application to Ascon-PRF Parameters

- $(k, b, c, r) = (128, 320, 192, 128)$
- Assume online complexity of $q, \sigma \ll 2^{64}$ (could be taken higher)
- The multicollision term $\nu^{2^{65}}_{128,192}$ is at most 5

Ascon-PRF: Implication

Multi-user bound from [DMV17]

$$\mathbf{Adv}^{\mu\text{-prf}}_{\text{Ascon-PRF}}(\mathsf{D}) \leq \frac{2\nu^{2\sigma}_{r,c}(N+1)}{2^c} + \frac{(\sigma-q)q}{2^b-q} + \frac{2\binom{\sigma}{2}}{2^b} + \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}$$

$$\frac{10(N+1)}{2^{192}} + \frac{2^{128}}{2^{320}} + \frac{2^{128}}{2^{320}} + \frac{2^{128}}{2^{320}} + \frac{\mu N}{2^{128}} + \frac{\binom{\mu}{2}}{2^{128}}$$

Application to Ascon-PRF Parameters

- $(k, b, c, r) = (128, 320, 192, 128)$
- Assume online complexity of $q, \sigma \ll 2^{64}$ (could be taken higher)
- The multicollision term $\nu^{2^{65}}_{128,192}$ is at most 5

Multi-user bound from [DMV17]

$$\mathbf{Adv}^{\mu\text{-prf}}_{\text{Ascon-PRF}}(\mathsf{D}) \le \frac{2\nu^{2\sigma}_{r,c}(N+1)}{2^c} + \frac{(\sigma-q)q}{2^b-q} + \frac{2\binom{\sigma}{2}}{2^b} + \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}$$

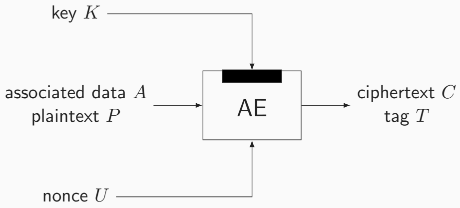
$$\le \quad \frac{10(N+1)}{2^{192}} + \frac{2^{128}}{2^{320}} + \frac{2^{128}}{2^{320}} + \frac{2^{128}}{2^{320}} + \frac{\mu N}{2^{128}} + \frac{\binom{\mu}{2}}{2^{128}}$$

Application to Ascon-PRF Parameters

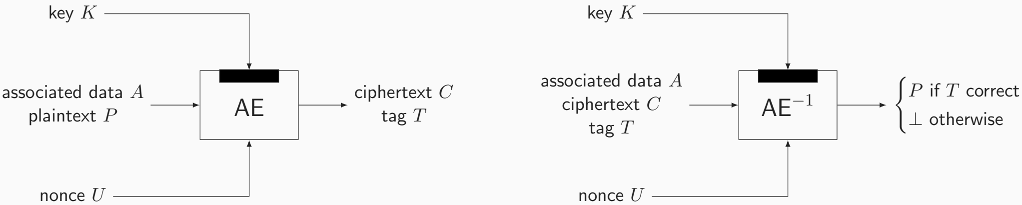
- $(k, b, c, r) = (128, 320, 192, 128)$
- Assume online complexity of $q, \sigma \ll 2^{64}$ (could be taken higher)
- The multicollision term $\nu^{2^{65}}_{128,192}$ is at most 5
- Generic security as long as $N \ll 2^{128}/\mu$

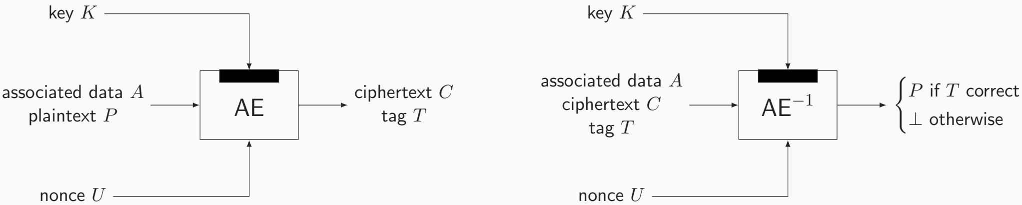
Duplex Application: MonkeySpongeWrap

Authenticated Encryption



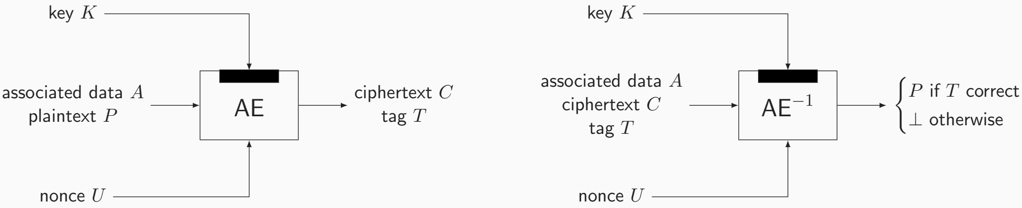
Authenticated Encryption





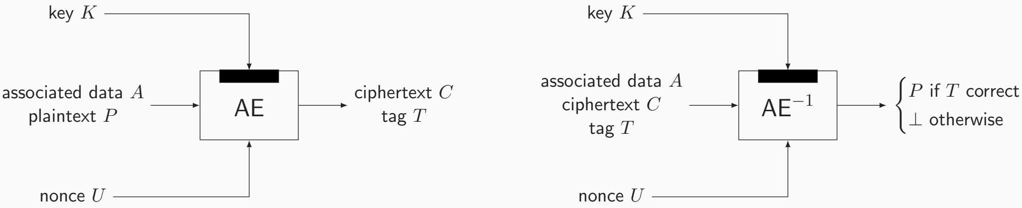
Role of Duplex

- Blockwise construction allows for processing different types of in-/output



Role of Duplex

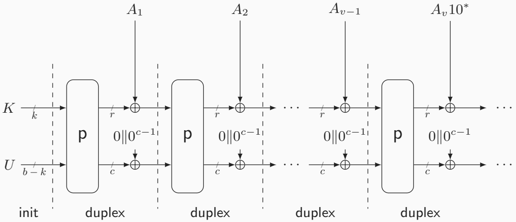
- Blockwise construction allows for processing different types of in-/output
- Usage of flag makes duplex-style encryption decryptable



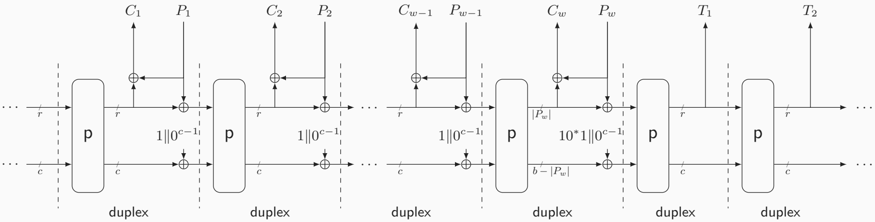
Role of Duplex

- Blockwise construction allows for processing different types of in-/output
- Usage of flag makes duplex-style encryption decryptable
  (Although the flag is not a necessity for this)

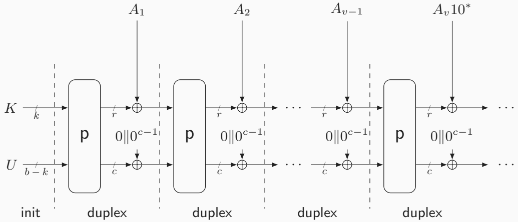
MonkeySpongeWrap: Encryption



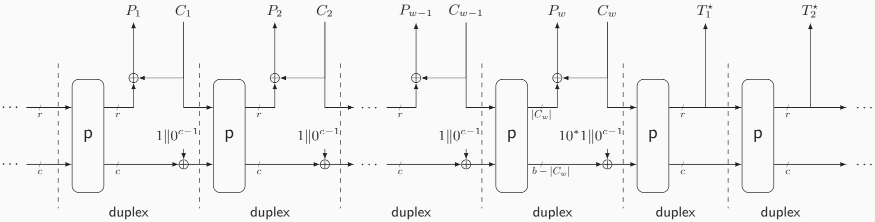
- Improvement over SpongeWrap [BDPV11a]
- State initialized using key and nonce
- Domain separation spill-over into inner part



MonkeySpongeWrap: Decryption



- Decryption similar to encryption
- Notable difference:
  - Processing of C
  - Duplexing with $\mathit{flag} = \mathit{true}$



MonkeySpongeWrap Versus Ascon-AEAD

- MonkeySpongeWrap can be described using duplex

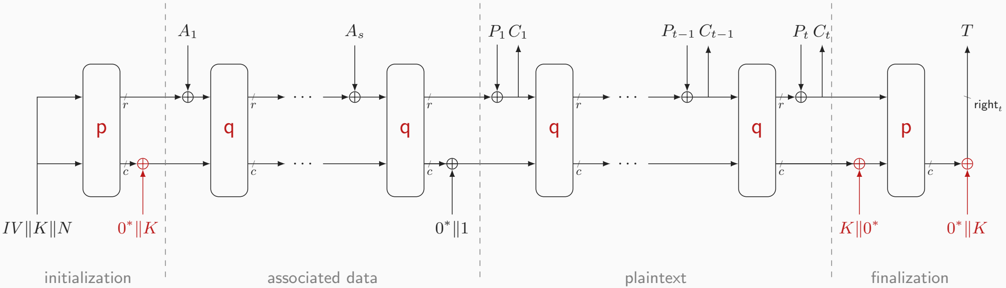
MonkeySpongeWrap Versus Ascon-AEAD

- MonkeySpongeWrap can be described using duplex
- Applications to modes of Xoodyak and Gimli (a.o.)

MonkeySpongeWrap Versus Ascon-AEAD

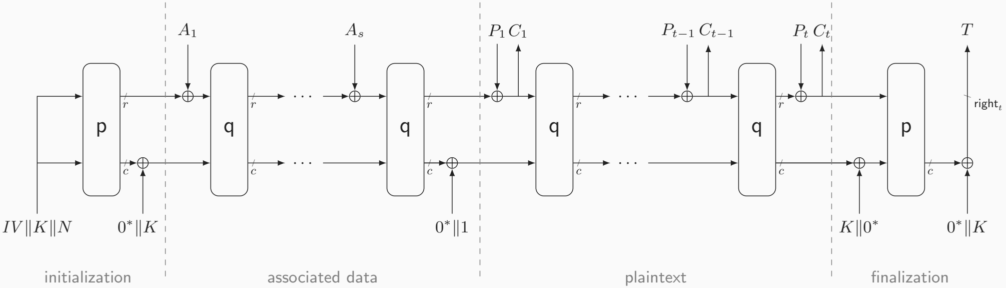
- MonkeySpongeWrap can be described using duplex
- Applications to modes of Xoodyak and Gimli (a.o.)

- Does not completely capture Ascon-AEAD
  - Additional key blindings at initialization and finalization
  - Outer and inner permutations p and q differ (minor)



Security of Ascon-AEAD Mode

Security of Ascon-AEAD Mode



Two New Complementary Results on Ascon-AEAD

- Chakraborty et al. [CDN23]: tight bound on nonce-respecting confidentiality and authenticity in case p = q (next talk)
- Lefevre and Mennink [LM23]: general confidentiality and authenticity with main focus on role of key blindings (now)

Multi-User Security Under Typical Models

property	setting	security as long as (highly simplified)
confidentiality	nonce-respecting	
	nonce-misuse	
authenticity	nonce-respecting	
	nonce-misuse	

Multi-User Security Under Typical Models

property	setting	security as long as (highly simplified)
confidentiality	nonce-respecting	$N \ll \min\{2^k/\mu, 2^{b/2}, 2^c\}$
	nonce-misuse	—
authenticity	nonce-respecting	
	nonce-misuse	

Multi-User Security Under Typical Models

property	setting	security as long as (highly simplified)
confidentiality	nonce-respecting	$N \ll \min\{2^k/\mu, 2^{b/2}, 2^c\}$
	nonce-misuse	—
authenticity	nonce-respecting	$N \ll \min\{2^k/\mu, 2^b/\sigma_{\mathcal{E}}, 2^c/\sigma_{\mathcal{D}}\}$
	nonce-misuse	$N \ll \min\{2^k/\mu, 2^c/(\sigma_{\mathcal{E}} + \sigma_{\mathcal{D}})\}$

Multi-User Security Under Typical Models

property	setting	security as long as (highly simplified)
confidentiality	nonce-respecting	$N \ll \min\{2^k/\mu, 2^{b/2}, 2^c\}$
	nonce-misuse	—
authenticity	nonce-respecting	$N \ll \min\{2^k/\mu, 2^b/\sigma_{\mathcal{E}}, 2^c/\sigma_{\mathcal{D}}\}$
	nonce-misuse	$N \ll \min\{2^k/\mu, 2^c/(\sigma_{\mathcal{E}} + \sigma_{\mathcal{D}})\}$

Application to Ascon-AEAD Parameters

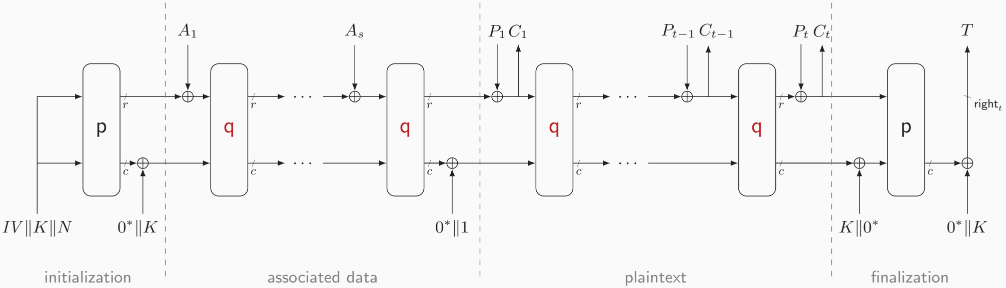
- $(k, b, c, r, t) = \begin{cases} (128, 320, 256, 64, 128) \text{ for Ascon-128} \\ (128, 320, 192, 128, 128) \text{ for Ascon-128a} \\ (160, 320, 256, 64, 128) \text{ for Ascon-80pq} \end{cases}$
- Assume online complexity of $q, \sigma \ll 2^{64}$ (could be taken higher)

Multi-User Security Under Typical Models

property	setting	security as long as (highly simplified)
confidentiality	nonce-respecting	$N \ll \min\{2^k/\mu, 2^{b/2}, 2^c\}$
	nonce-misuse	—
authenticity	nonce-respecting	$N \ll \min\{2^k/\mu, 2^b/\sigma_{\mathcal{E}}, 2^c/\sigma_{\mathcal{D}}\}$
	nonce-misuse	$N \ll \min\{2^k/\mu, 2^c/(\sigma_{\mathcal{E}} + \sigma_{\mathcal{D}})\}$

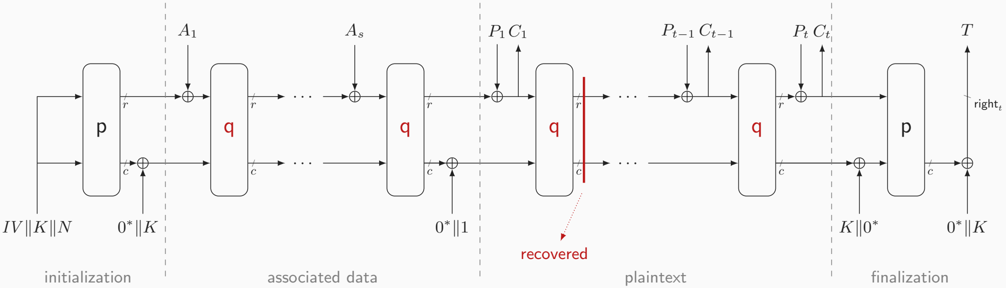
Application to Ascon-AEAD Parameters

- $(k, b, c, r, t) = \begin{cases} (128, 320, 256, 64, 128) & \text{for Ascon-128} \\ (128, 320, 192, 128, 128) & \text{for Ascon-128a} \\ (160, 320, 256, 64, 128) & \text{for Ascon-80pq} \end{cases}$
- Assume online complexity of $q, \sigma \ll 2^{64}$ (could be taken higher)
- Generic security as long as $N \ll 2^{128}/\mu$ (or $N \ll 2^{160}/\mu$ for Ascon-80pq)



Attack Setting

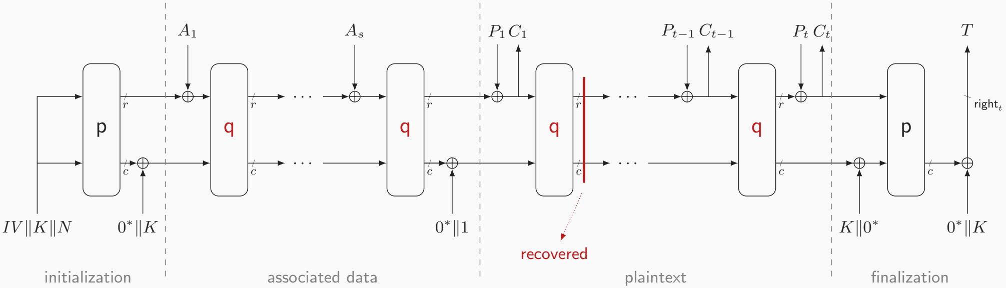
- Inner permutation q may get weaker protection than outer permutation



Attack Setting

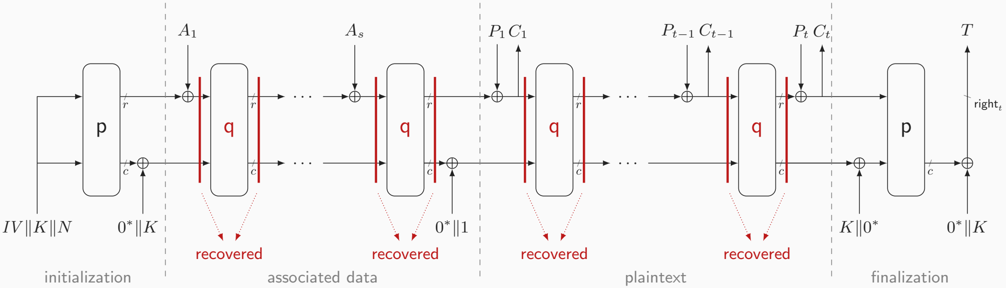
- Inner permutation q may get weaker protection than outer permutation
- Adversary may somehow recover any inner state

Authenticity Under State Recovery (1)



Attack Setting

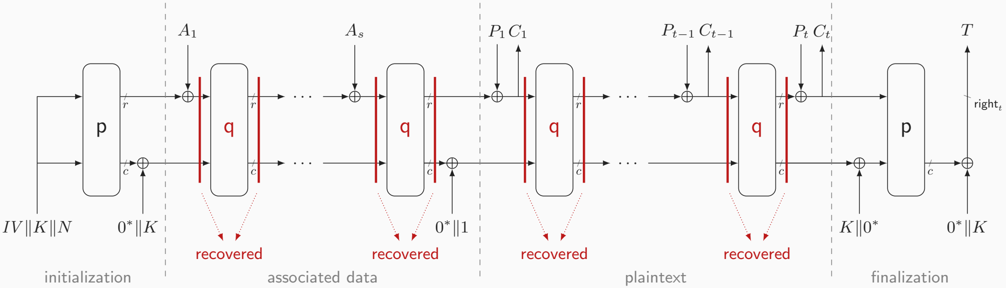
- Inner permutation q may get weaker protection than outer permutation
- Adversary may somehow recover any inner state
- Ascon-AEAD designed to still achieve authenticity in this setting



Model

- Without loss of generality: all evaluations of inner permutation q leak

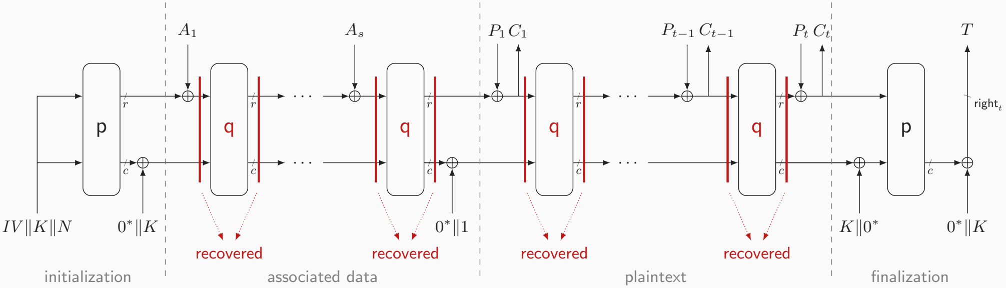
Authenticity Under State Recovery (2)



Model

- Without loss of generality: all evaluations of inner permutation q leak
- Model inspired by permutation-based leakage resilience [DM19a, DM19b]
- Adversary wins if it forges tag even under inner state recovery

Authenticity Under State Recovery (3)

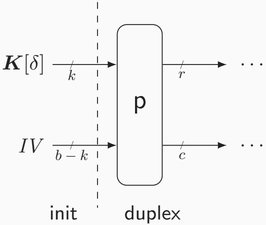


Results

- MonkeySpongeWrap-style AEAD does not achieve this property
- Ascon-AEAD mode achieves security as long as $N \ll \min\{2^k/\mu, 2^{c/2}\}$
- For Ascon-AEAD parameters: generic security as long as $N \ll 2^{128}/\mu$

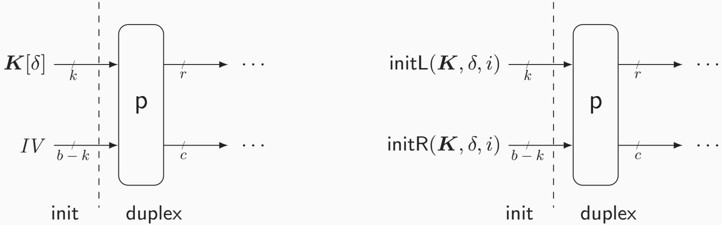
Generalized Duplex Initialization

On the Power of Initialization



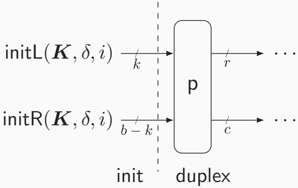
- Plain initialization: incurs term $\frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}$
  - Assumes that attacker has full control over IV

On the Power of Initialization



- Plain initialization: incurs term $\frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}$
  - Assumes that attacker has full control over IV
- Dobraunig and Mennink [DM23]: generalized analysis of initialization
  - Both inner and outer part may be keyed or depend on IV
  - i serves role of IV but also allows to formally capture random IV's

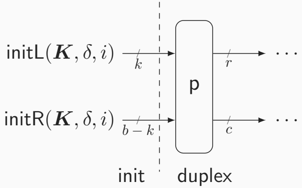
case	initL$(\boldsymbol{K}, \delta, i)$	initR$(\boldsymbol{K}, \delta, i)$
baseline	$\boldsymbol{K}[\delta]$	$\mathrm{encode}_{b-k}[i]$
global IV	$\boldsymbol{K}[\delta]$	$\mathrm{encode}_{b-k}[(\delta, i)]$
random IV	$\boldsymbol{K}[\delta]$	$RIV \Vert 0^{b-k-n}$
quasi-random IV	$\boldsymbol{K}[\delta]$	$(RIV_\delta \oplus \mathrm{encode}_n[i]) \Vert 0^{b-k-n}$
IV on key	$\boldsymbol{K}[\delta] \oplus \mathrm{encode}_k[i]$	0^{b-k}
global IV on key	$\boldsymbol{K}[\delta] \oplus \mathrm{encode}_k[i]$	$\mathrm{encode}_{b-k}[\delta]$



- Different types of initialization (see paper for side-conditions)
- RIV stands for random IV, RIV_δ unique random IV per user

Different Initializations

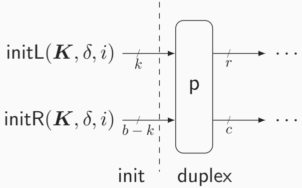
case	$\mathsf{initL}(\boldsymbol{K}, \delta, i)$	$\mathsf{initR}(\boldsymbol{K}, \delta, i)$
baseline	$\boldsymbol{K}[\delta]$	$\text{encode}_{b-k}[i]$
global IV	$\boldsymbol{K}[\delta]$	$\text{encode}_{b-k}[(\delta, i)]$
random IV	$\boldsymbol{K}[\delta]$	$RIV \| 0^{b-k-n}$
quasi-random IV	$\boldsymbol{K}[\delta]$	$(RIV_\delta \oplus \text{encode}_n[i]) \| 0^{b-k-n}$
IV on key	$\boldsymbol{K}[\delta] \oplus \text{encode}_k[i]$	0^{b-k}
global IV on key	$\boldsymbol{K}[\delta] \oplus \text{encode}_k[i]$	$\text{encode}_{b-k}[\delta]$



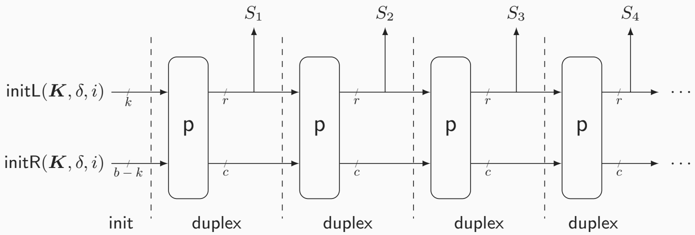
- Different types of initialization (see paper for side-conditions)
- RIV stands for random IV, RIV_δ unique random IV per user
- Improved security bound for optimized initialization

Different Initializations

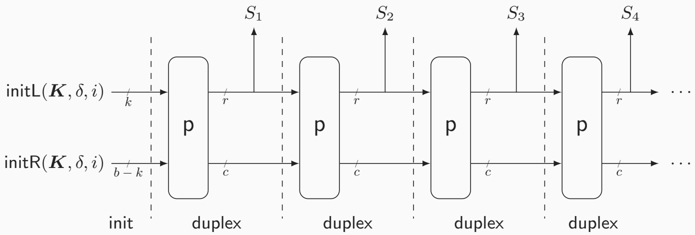
case	initL$(\boldsymbol{K}, \delta, i)$	initR$(\boldsymbol{K}, \delta, i)$
baseline	$\boldsymbol{K}[\delta]$	$\mathrm{encode}_{b-k}[i]$
global IV	$\boldsymbol{K}[\delta]$	$\mathrm{encode}_{b-k}[(\delta, i)]$
random IV	$\boldsymbol{K}[\delta]$	$RIV \| 0^{b-k-n}$
quasi-random IV	$\boldsymbol{K}[\delta]$	$(RIV_\delta \oplus \mathrm{encode}_n[i]) \| 0^{b-k-n}$
IV on key	$\boldsymbol{K}[\delta] \oplus \mathrm{encode}_k[i]$	0^{b-k}
global IV on key	$\boldsymbol{K}[\delta] \oplus \mathrm{encode}_k[i]$	$\mathrm{encode}_{b-k}[\delta]$



- Different types of initialization (see paper for side-conditions)
- RIV stands for random IV, RIV_δ unique random IV per user
- Improved security bound for optimized initialization
- Application to keystream and authenticated encryption


S_1
S_2
S_3
S_4
initL$(\boldsymbol{K}, \delta, i)$
initR$(\boldsymbol{K}, \delta, i)$
k
$b-k$
r
c
p
p
p
p
init
duplex
duplex
duplex
duplex

Application to Keystream Generation (Randomized IV in Paper)



case	initL$(\boldsymbol{K}, \delta, i)$	initR$(\boldsymbol{K}, \delta, i)$	initialization term (simplified)
baseline	$\boldsymbol{K}[\delta]$	$\mathrm{encode}_{b-k}[i]$	$\frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}$
global IV	$\boldsymbol{K}[\delta]$	$\mathrm{encode}_{b-k}[(\delta, i)]$	$\frac{N}{2^k}$
IV on key	$\boldsymbol{K}[\delta] \oplus \mathrm{encode}_k[i]$	0^{b-k}	$\frac{QN}{2^k} + \frac{\binom{Q}{2}}{2^k}$
global IV on key	$\boldsymbol{K}[\delta] \oplus \mathrm{encode}_k[i]$	$\mathrm{encode}_{b-k}[\delta]$	$\frac{Q_\delta N}{2^k} + \frac{\mu\binom{Q_\delta}{2}}{2^k}$

Q stands for # initializations, Q_δ initializations per user

Conclusion

Main Takeaways

- Keyed duplex
  - Versatile construction but application not always clear
  - Dedicated analysis sometimes more suited

Conclusion

Main Takeaways

- Keyed duplex
  - Versatile construction but application not always clear
  - Dedicated analysis sometimes more suited
- Additional key blindings at initialization and finalization improve security

Conclusion

Main Takeaways

- Keyed duplex
  - Versatile construction but application not always clear
  - Dedicated analysis sometimes more suited
- Additional key blindings at initialization and finalization improve security
- Gains in multi-user setting by specific initialization

Conclusion

Main Takeaways

- Keyed duplex
  - Versatile construction but application not always clear
  - Dedicated analysis sometimes more suited
- Additional key blindings at initialization and finalization improve security
- Gains in multi-user setting by specific initialization
- Caution: all presented results only hold in random permutation model

Conclusion

Main Takeaways

- Keyed duplex
  - Versatile construction but application not always clear
  - Dedicated analysis sometimes more suited
- Additional key blindings at initialization and finalization improve security
- Gains in multi-user setting by specific initialization
- Caution: all presented results only hold in random permutation model

Acknowledgments

- Parts of the presentation come from recent collaborations with Christoph Dobraunig [DM23] and Charlotte Lefevre [LM22, LM23]

Main Takeaways

- Keyed duplex
  - Versatile construction but application not always clear
  - Dedicated analysis sometimes more suited
- Additional key blindings at initialization and finalization improve security
- Gains in multi-user setting by specific initialization
- Caution: all presented results only hold in random permutation model

Acknowledgments

- Parts of the presentation come from recent collaborations with Christoph Dobraunig [DM23] and Charlotte Lefevre [LM22, LM23]

Thank you for your attention!

Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche.
Security of Keyed Sponge Constructions Using a Modular Proof Approach.
In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 364–384. Springer, 2015.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
Sponge functions.
Ecrypt Hash Workshop 2007, May 2007.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
On the Indifferentiability of the Sponge Construction.
In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, 2008.

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, *SAC 2011*, volume 7118 of *LNCS*, pages 320–337. Springer, 2011.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
On the security of the keyed sponge construction.
Symmetric Key Encryption Workshop, February 2011.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
Permutation-based encryption, authentication and authenticated encryption.
Directions in Authenticated Ciphers, July 2012.

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.

Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi.
Exact Security Analysis of ASCON.
Cryptology ePrint Archive, Report 2023/775, 2023.

Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer.
Ascon PRF, MAC, and Short-Input MAC.
Cryptology ePrint Archive, Report 2021/1574, 2021.

Christoph Dobraunig and Bart Mennink.
Leakage Resilience of the Duplex Construction.
In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 225–255. Springer, 2019.

Christoph Dobraunig and Bart Mennink.
Security of the Suffix Keyed Sponge.
IACR Trans. Symmetric Cryptol., 2019(4):223–248, 2019.

Christoph Dobraunig and Bart Mennink.
Generalized Initialization of the Duplex Construction.
Cryptology ePrint Archive, Report 2023/924, 2023.

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, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 606–637. Springer, 2017.

Peter Gazi and Stefano Tessaro.
Provably Robust Sponge-Based PRNGs and KDFs.
In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 87–116. Springer, 2016.

Charlotte Lefevre and Bart Mennink.
Tight Preimage Resistance of the Sponge Construction.
In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 185–204. Springer, 2022.

Charlotte Lefevre and Bart Mennink.
Generic Security of the Ascon Mode: On the Power of Key Blinding.
Cryptology ePrint Archive, Report 2023/796, 2023.

Bart Mennink.
Key Prediction Security of Keyed Sponges.
IACR Trans. Symmetric Cryptol., 2018(4):128–149, 2018.

Bart Mennink.
Understanding the Duplex and Its Security.
IACR Trans. Symmetric Cryptol., 2023(2), 2023.
to appear.

Bart Mennink, Reza Reyhanitabar, and Damian Vizár.
Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption.
In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 465–489. Springer, 2015.

Yusuke Naito and Kan Yasuda.
New Bounds for Keyed Sponges with Extendable Output: Independence Between Capacity and Message Length.
In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 3–22. Springer, 2016.