Security of Permutation-Based Modes and Its Application to Ascon

Bart Mennink
Radboud University (The Netherlands)
NIST Lightweight Cryptography Workshop 2023
June 22, 2023
Sponges and Ascon-Hash Mode
• $p$ is a $b$-bit permutation, with $b = r + c$
  • $r$ is the rate
  • $c$ is the capacity (security parameter)
• SHA-3, XOFs, lightweight hashing, ...
• Assume that $p$ is a random permutation
Indifferentiability of the Sponge [BDPV08]

• Assume that \( p \) is a random permutation
• Sponge indifferentiable from random oracle:

\[
\Delta_D(\text{sponge}, p; \text{ro, sim}) \leq N^2/2^{c+1}
\]

• \( N \) 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:
  \[
  \Delta_D(\text{sponge}, p; \text{ro, sim}) \leq \frac{N^2}{2^{c+1}}
  \]
  
  - $N$ 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: $\frac{N^2}{2^{c+1}} + \frac{N^2}{2^{n+1}}$
  Second preimage resistance: $\frac{N^2}{2^{c+1}} + \frac{N}{2^n}$
  Preimage resistance: $\frac{N^2}{2^{c+1}} + \frac{N}{2^n}$
Indifferentiability of the Sponge [BDPV08]

• Assume that \( p \) is a random permutation
• Sponge indifferentiable from random oracle:
  \[
  \Delta_D(\text{sponge, } p; \text{ro, sim}) \leq N^2/2^{c+1}
  \]

• \( N \) 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: \( N^2/2^{c+1} + N/2^n \)

  \[ \text{distance from sponge to RO} \quad \text{classical attacks against RO} \]
  \[ (N \text{ is } \# \text{ primitive evaluations}) \quad (N \text{ is } \# \text{ oracle evaluations}) \]
Indifferentiability of the Sponge [BDPV08]

• Assume that \( p \) is a random permutation

• Sponge indifferentiable from random oracle:

\[
\Delta_D(\text{sponge, } p;\text{ ro, sim}) \leq N^2/2^{c+1}
\]

• \( N \) 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: \( N^2/2^{c+1} + N/2^n \)

\( \uparrow \) distance from sponge to RO

\( \uparrow \) classical attacks against RO

\( (N \text{ is } \# \text{ primitive evaluations}) \)

\( (N \text{ is } \# \text{ oracle evaluations}) \)
Indifferentiability of the Sponge [BDPV08]

• Assume that \( p \) is a random permutation
• Sponge indifferentiable from random oracle:
  \[ \Delta_D(\text{sponge}, p; \text{ro}, \text{sim}) \leq N^2/2^{c+1} \]
  
  • \( N \) 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:
  \[ N^2/2^{c+1} + N^2/2^{n+1} \leftarrow \text{attack in } \min\{2^{c/2}, 2^n/2\} \]

  Second preimage resistance:
  \[ N^2/2^{c+1} + N/2^n \leftarrow \text{attack in } \min\{2^{c/2}, 2^n\} \]

  Preimage resistance:
  \[ N^2/2^{c+1} + N/2^n \leftarrow \text{attack in } \min\{2^{n-r} + 2^{c/2}, 2^n\} \]

  \[ \text{distance from sponge to RO} \]
  \[ \text{(} N \text{ is } \# \text{ primitive evaluations)} \]
  \[ \text{classical attacks against RO} \]
  \[ \text{(} N \text{ is } \# \text{ oracle evaluations)} \]
Tight Preimage Resistance

- Security proven up to $\approx \min \{2^{c/2}, 2^n\}$ 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 $\approx \min \{2^{c/2}, 2^n\}$ 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

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

- Security proven up to $\approx \min \{2^{c/2}, 2^n\}$ 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

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

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
Keying Sponges

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 $\ell$: SC($K, D, \ell$) = sponge($K \| D, \ell$)
- (All assuming $K$ is fixed-length)
Keying Sponges

Keyed Sponge

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

Keyed Duplex

- Authenticated encryption
- Multiple CAESAR and NIST LWC submissions
Evolution of Keyed Sponges

- 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, GPT15, MRV15]
• Unkeyed Duplex [BDPV11a]
• 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]

∀i : zi ≤ r
Understanding the Duplex
Generalized Keyed Duplex ([DMV17, DM19a, Men23])

Features
- Multi-user by design: index $\delta$ specifies key in array
- Initial state: concatenation of $K[\delta]$ and IV
- Full-state absorption, no padding
- Refined adversarial strength
Generalized Keyed Duplex ([DMV17, DM19a, Men23])

Features

• Multi-user by design: index \( \delta \) specifies key in array
• Initial state: concatenation of \( K[\delta] \) and \( IV \)
• Full-state absorption, no padding
• Refined adversarial strength
Generalized Keyed Duplex: Flag (1)

Typical use case: authenticated encryption using duplex

Security decreases for increasing number of calls with flag $= \text{true}$
Generalized Keyed Duplex: Flag (1)

- Typical use case: authenticated encryption using duplex
• Typical use case: authenticated encryption using duplex
• Security decreases for increasing number of calls with flag = 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
• 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
Generalized Keyed Duplex: Flag (2)

- 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**

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

**Decryption**
Algorithm Keyed duplex construction \( \text{KD}[p]_K \)

**Interface:** KD.init

**Input:** \((\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{I}\mathcal{V}\)

**Output:** ∅

\[\begin{align*}
S &\leftarrow \text{rot}_\alpha(K[\delta] \parallel IV) \\
\text{return} &\varnothing
\end{align*}\]

**Interface:** KD.duplex

**Input:** \((\text{flag}, P) \in \{\text{true, false}\} \times \{0, 1\}^b\)

**Output:** \(Z \in \{0, 1\}^r\)

\[\begin{align*}
S &\leftarrow p(S) \\
Z &\leftarrow \text{left}_r(S) \\
S &\leftarrow S \oplus [\text{flag}] \cdot (Z \parallel 0^{b-r}) \oplus P \\
\text{return} &\varnothing
\end{align*}\]
Algorithm Keyed duplex construction $\text{KD}[p]_K$

**Interface:** KD.init

**Input:** $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{I}V$

**Output:** $\varnothing$

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

**Interface:** KD.duplex

**Input:** $(\text{flag}, P) \in \{\text{true, false}\} \times \{0, 1\}^b$

**Output:** $Z \in \{0, 1\}^r$

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

---

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

**Interface:** IXIF.init

**Input:** $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{I}V$

**Output:** $\varnothing$

\[
\text{path} \leftarrow \text{encode}[\delta] \parallel IV \\
\text{return } \varnothing
\]

**Interface:** IXIF.duplex

**Input:** $(\text{flag}, P) \in \{\text{true, false}\} \times \{0, 1\}^b$

**Output:** $Z \in \{0, 1\}^r$

\[
S \leftarrow p(S) \\
Z \leftarrow \text{ro}(\text{path}, r) \\
\text{path} \leftarrow \text{path} || ([\text{flag}] \cdot (Z\|0^{b-r}) \oplus P) \\
\text{return } Z
\]
Security Model ([DMV17, DM19a, Men23])

**Algorithm** Keyed duplex construction \(KD[p]_K\)

**Interface:** \(KD\).\text{init}

**Input:** \((\delta, IV) \in \{1, \ldots, \mu\} \times IV\)

**Output:** \(\emptyset\)
- \(S \leftarrow \text{rot}_\alpha(K[\delta] \parallel IV)\)
- \(\text{return } \emptyset\)

**Interface:** \(KD\).\text{duplex}

**Input:** \((\text{flag}, P) \in \{\text{true}, \text{false}\} \times \{0, 1\}^b\)

**Output:** \(Z \in \{0, 1\}^\tau\)
- \(S \leftarrow p(S)\)
- \(Z \leftarrow \text{left}_r(S)\)
- \(S \leftarrow S \oplus [\text{flag}] \cdot (Z_0^{b-r}) \oplus P\)
- \(\text{return } Z\)

\[
\text{Adv}_{KD}(D) = \Delta_D \left( KD[p]_K, p^\pm ; \text{IXIF}[ro], p^\pm \right)
\]
Security Model ([DMV17, DM19a, Men23])

Algorithm Keyed duplex construction $KD[p]_K$

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

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

Algorithm Ideal extendable input function $IXIF[ro]$

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

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

$\text{Adv}_{KD}(D) = \Delta_D (KD[p]_K, p^\pm ; IXIF[ro], p^\pm)$

- $IXIF[ro]$ is basically random oracle in disguise
Security Model ([DMV17, DM19a, Men23])

**Algorithm** Keyed duplex construction $\text{KD}[p]_K$

**Interface**: $\text{KD}.\text{init}$
**Input**: $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{I}_V$
**Output**: $\emptyset$

$S \leftarrow \text{rot}_\alpha(K[\delta] \parallel IV)$
return $\emptyset$

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

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

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

**Interface**: $\text{IXIF}.\text{init}$
**Input**: $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{I}_V$
**Output**: $\emptyset$

$\text{path} \leftarrow \text{encode}[\delta] \parallel IV$
return $\emptyset$

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

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

$$\text{Adv}_{\text{KD}}(D) = \Delta_D (\text{KD}[p]_K, p^\pm ; \text{IXIF}[ro], p^\pm)$$

- $\text{IXIF}[ro]$ is basically random oracle in disguise
- If $\text{KD}[p]_K$ is hard to distinguish from $\text{IXIF}[ro]$ for certain bound on adversarial resources, $\text{KD}[p]_K$ roughly “behaves like” random oracle
Security Model ([DMV17, DM19a, Men23])

Algorithm Keyed duplex construction $\text{KD}[p]_K$

**Interface:** \(\text{KD}.\text{init}\)

**Input:** \((δ, IV) \in \{1, \ldots, µ\} \times TV\)

**Output:** \(\emptyset\)

\[
S \leftarrow \text{rot}_α(K[δ] || IV)
\]

return \(\emptyset\)

**Interface:** \(\text{KD}.\text{duplex}\)

**Input:** \((\text{flag}, P) \in \{\text{true}, \text{false}\} \times \{0,1\}^b\)

**Output:** \(Z \in \{0,1\}^r\)

\[
S \leftarrow p(S)
\]

\[
Z \leftarrow \text{left}_r(S)
\]

\[
S \leftarrow S \oplus [\text{flag}] \cdot (Z \| 0^{b-r}) \oplus P
\]

return \(Z\)

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

**Interface:** \(\text{IXIF}.\text{init}\)

**Input:** \((δ, IV) \in \{1, \ldots, µ\} \times TV\)

**Output:** \(\emptyset\)

\[
\text{path} \leftarrow \text{encode}[δ] || IV
\]

return \(\emptyset\)

**Interface:** \(\text{IXIF}.\text{duplex}\)

**Input:** \((\text{flag}, P) \in \{\text{true}, \text{false}\} \times \{0,1\}^b\)

**Output:** \(Z \in \{0,1\}^r\)

\[
Z \leftarrow \text{ro(path, r)}
\]

\[
\text{path} \leftarrow \text{path} \| ([\text{flag}] \cdot (Z \| 0^{b-r}) \oplus P)
\]

return \(Z\)

\[
\text{Adv}_{\text{KD}}(D) = ∆_D (\text{KD}[p]_K, p^\pm ; \text{IXIF}[ro], p^\pm)
\]

- \(\text{IXIF}[ro]\) is basically random oracle in disguise
- If \(\text{KD}[p]_K\) is hard to distinguish from \(\text{IXIF}[ro]\) for certain bound on adversarial resources, \(\text{KD}[p]_K\) 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)
- $\Omega$: # 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)
- $\Omega$: # 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]:

$$\text{Adv}_{KD}^N(D) \leq \frac{(L + \Omega)N}{2^c} + \frac{2\nu_{r,c}^M(M-L)(N+1)}{2^c} + \frac{(L+\Omega+1)}{2^c} + \frac{(M - L - Q)Q}{2^b - Q} + \frac{M(M - L - 1)}{2^b} + \frac{Q(M - L - Q)}{2^c - Q} + \frac{Q_{IV}N}{2^k} + \frac{(\nu_{r,c}^M)^2}{2^k}$$

- [DM19a] (with one simplification):

$$\text{Adv}_{KD}^N(D) \leq \frac{(L + \Omega)N}{2^c} + \frac{2\nu_{r,c}^M(N+1)}{2^c} + \frac{\nu_{r,c}^M(L + \Omega)}{2^c} + \frac{(M - L - Q)}{2^b} + \frac{(M - L - Q)(L + \Omega)}{2^b} + \frac{(M + N)}{2^b} + \frac{(\nu_{r,c}^M)^2}{2^c - Q} + \frac{Q(M - Q)}{2^c - Q} + \frac{Q_{IV}N}{2^k} + \frac{(\nu_{r,c}^M)^2}{2^k}$$
Duplex Application: Keystream Generation
Keystream Generation

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

Algorithm Keystream generation $\text{SC}[p]$

| 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: $\text{KD}[p](K)$ |
| $S \leftarrow \emptyset$ |
| $\text{KD}.\text{init}(1, U)$ |
| for $i = 1, \ldots, \lceil \ell/r \rceil$ do |
| $S \leftarrow S \| \text{KD}..\text{duplex}(\text{false}, 0^b)$ |
| return $\text{left}_\ell(S)$ |
Keystream Generation

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

**Algorithm** Keystream generation $SC[p]$

<table>
<thead>
<tr>
<th>Input: $(K, U, \ell) \in {0, 1}^k \times {0, 1}^{b-k} \times N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: $S \in {0, 1}^\ell$</td>
</tr>
<tr>
<td>Underlying keyed duplex: $KD[p]_{(K)}$</td>
</tr>
<tr>
<td>$S \leftarrow \emptyset$</td>
</tr>
<tr>
<td>$KD.init(1, U)$</td>
</tr>
<tr>
<td>for $i = 1, \ldots, \lceil \ell/r \rceil$ do</td>
</tr>
<tr>
<td>$S \leftarrow S \parallel KD.duplex(false, 0^b)$</td>
</tr>
<tr>
<td>return $\text{left}_\ell(S)$</td>
</tr>
</tbody>
</table>
Keystream Generation: Security (1)

- Consider distinguisher $D$ against PRF security of $\text{SC}[p]$
  \[
  \text{Adv}^{\text{prf}}_{\text{SC}}(D) = \Delta_D \left( \text{SC}[p]_K, p^\pm; R^{\text{prf}}, p^\pm \right)
  \]

- $D$ can make $q$ construction queries (total $\sigma$ blocks) + $N$ primitive queries
• Consider distinguisher $D$ against PRF security of $SC[p]$

$$Adv^\text{prf}_{SC}(D) = \Delta_D \left( SC[p]_K, p^\pm ; R^\text{prf}, p^\pm \right)$$

• $D$ can make $q$ construction queries (total $\sigma$ blocks) + $N$ primitive queries

• $SC[p]_K$ is basically just $SC[\text{KD}[p]_K]$
• Consider distinguisher D against PRF security of SC[p]

\[ \text{Adv}_{\text{SC}}^{\text{prf}}(D) = \Delta_D \left( \text{SC}[p]_K, p^\pm ; \ R^{\text{prf}}, p^\pm \right) \]

• D can make \( q \) construction queries (total \( \sigma \) blocks) + \( N \) primitive queries

• SC[p]_K is basically just SC[KD[p]_K]

• Triangle inequality:

\[ \text{Adv}_{\text{SC}}^{\text{prf}}(D) = \Delta_D \left( \text{SC}[p]_K, p^\pm ; \ R^{\text{prf}}, p^\pm \right) \]
\[ = \Delta_D \left( \text{SC}[KD[p]_K], p^\pm ; \ R^{\text{prf}}, p^\pm \right) \]
\[ \leq \Delta_D \left( \text{SC}[KD[p]_K], p^\pm ; \ SC[\text{IXIF}[\text{ro}]], p^\pm \right) + \Delta_D \left( \text{SC}[\text{IXIF}[\text{ro}]], p^\pm ; \ R^{\text{prf}}, p^\pm \right) \]
• Consider distinguisher $D$ against PRF security of $SC[p]$

$$\text{Adv}^{\text{prf}}_{SC}(D) = \Delta_D \left( SC[p]_K, p^\pm ; R^{\text{prf}}, p^\pm \right)$$

• $D$ can make $q$ construction queries (total $\sigma$ blocks) + $N$ primitive queries

• $SC[p]_K$ is basically just $SC[KD[p]_K]$

• Triangle inequality:

$$\text{Adv}^{\text{prf}}_{SC}(D) = \Delta_D \left( SC[p]_K, p^\pm ; R^{\text{prf}}, p^\pm \right)$$

$$= \Delta_D \left( SC[KD[p]_K], p^\pm ; R^{\text{prf}}, p^\pm \right)$$

$$\leq \Delta_D \left( SC[KD[p]_K], p^\pm ; SC[IxIF[ro]], p^\pm \right) + \Delta_D \left( SC[IxIF[ro]], p^\pm ; R^{\text{prf}}, p^\pm \right)$$

$$= 0$$
Keystream Generation: Security (1)

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

$$\text{Adv}_{SC}^{\text{prf}}(D) = \Delta_D \left( SC[p^K], p^\pm ; R^{\text{prf}}, p^\pm \right)$$

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

- Triangle inequality:

$$\text{Adv}_{SC}^{\text{prf}}(D) = \Delta_D \left( SC[p^K], p^\pm ; R^{\text{prf}}, p^\pm \right)$$

$$= \Delta_D \left( SC[KD[p]^K], p^\pm ; R^{\text{prf}}, p^\pm \right)$$

$$\leq \Delta_D \left( SC[KD[p]^K], p^\pm ; SC[IXIF[ro]], p^\pm \right) + \Delta_D \left( SC[IXIF[ro]], p^\pm ; R^{\text{prf}}, p^\pm \right)$$

$$\leq \Delta_{D'} \left( KD[p]^K, p^\pm ; IXIF[ro], p^\pm \right) = 0$$
Keystream Generation: Security (1)

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

\[
\text{Adv}_{SC}^{\text{prf}}(D) = \Delta_D \left( SC[p], p^\pm ; R^{\text{prf}}, p^\pm \right)
\]

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

\[
\text{Adv}_{SC}^{\text{prf}}(D) = \Delta_D \left( SC[p]_K, p^\pm ; R^{\text{prf}}, p^\pm \right)
\]

\[
= \Delta_D \left( SC[KD[p]_K], p^\pm ; R^{\text{prf}}, p^\pm \right)
\]

\[
\leq \Delta_D \left( SC[KD[p]_K], p^\pm ; SC[IXIF[ro]], p^\pm \right) + \Delta_D \left( SC[IXIF[ro]], p^\pm ; R^{\text{prf}}, p^\pm \right)
\]

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

- What are the resources of $D'$?
Keystream Generation: Security (2)

Algorithm Keystream generation $\text{SC}[p]$

- **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:** $\text{KD}[p](K)$

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

<table>
<thead>
<tr>
<th>resources of $D'$</th>
<th>in terms of resources of $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$: data complexity (calls to construction)</td>
<td></td>
</tr>
<tr>
<td>$N$: time complexity (calls to primitive)</td>
<td></td>
</tr>
<tr>
<td>$Q$: number of init calls</td>
<td></td>
</tr>
<tr>
<td>$Q_{IV}$: max # init calls for single IV</td>
<td></td>
</tr>
<tr>
<td>$L$: # queries with repeated path</td>
<td></td>
</tr>
<tr>
<td>$\Omega$: # queries with overwriting outer part</td>
<td></td>
</tr>
</tbody>
</table>
Algorithm Keystream generation SC[p]

Input: $(K, U, ℓ) \in \{0, 1\}^k \times \{0, 1\}^{b - k} \times \mathbb{N}$
Output: $S \in \{0, 1\}^ℓ$

Underlying keyed duplex: $KD[p](K)$

\[
S \leftarrow \emptyset \\
KD.init(1, U) \\
for i = 1, \ldots, \lceil ℓ/r \rceil \ do \\
\quad S \leftarrow S \parallel KD.duplex(false, 0^b) \\
return left_ℓ(S)
\]

resources of D' in terms of resources of D

- $M$: data complexity (calls to construction)
- $N$: time complexity (calls to primitive) $\rightarrow N$
- $Q$: number of init calls
- $Q_{IV}$: max # init calls for single IV
- $L$: # queries with repeated path
- $Ω$: # queries with overwriting outer part
Algorithm Keystream generation $\text{SC}[p]$

Input: $(K, U, \ell) \in \{0, 1\}^k \times \{0, 1\}^{b-k} \times N$

Output: $S \in \{0, 1\}^\ell$

Underlying keyed duplex: $\text{KD}[p](K)$

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

<table>
<thead>
<tr>
<th>resources of D'</th>
<th>in terms of</th>
<th>resources of D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$: data complexity (calls to construction)</td>
<td>$\longrightarrow$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>$N$: time complexity (calls to primitive)</td>
<td>$\longrightarrow$</td>
<td>$N$</td>
</tr>
<tr>
<td>$Q$: number of init calls</td>
<td>$\longrightarrow$</td>
<td>$q$</td>
</tr>
<tr>
<td>$Q_{IV}$: max # init calls for single IV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$L$: # queries with repeated path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Omega$: # queries with overwriting outer part</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Keystream Generation: Security (2)

Algorithm Keystream generation \( SC[p]\)

**Input:** \((K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k} \times N\)

**Output:** \( S \in \{0,1\}^{\ell} \)

**Underlying keyed duplex:** \( KD[p](K)\)

\[
S \leftarrow \emptyset
\]

\[
KD\.init(1, U)
\]

\[
\text{for } i = 1, \ldots, \lceil \ell/r \rceil \text{ do }
\]

\[
S \leftarrow S \parallel KD\.duplex(false, 0^b)
\]

\[
\text{return } \text{left}_\ell(S)
\]

<table>
<thead>
<tr>
<th>resources of D'</th>
<th>in terms of</th>
<th>resources of D</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M ): data complexity (calls to construction)</td>
<td>( \rightarrow )</td>
<td>( \sigma )</td>
</tr>
<tr>
<td>( N ): time complexity (calls to primitive)</td>
<td>( \rightarrow )</td>
<td>( N )</td>
</tr>
<tr>
<td>( Q ): number of init calls</td>
<td>( \rightarrow )</td>
<td>( q )</td>
</tr>
<tr>
<td>( Q_{IV} ): max # init calls for single IV</td>
<td>( \rightarrow )</td>
<td>( 1 )</td>
</tr>
<tr>
<td>( L ): # queries with repeated path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \Omega ): # queries with overwriting outer part</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Algorithm Keystream generation $SC[p]$

Input: $(K, U, \ell) \in \{0, 1\}^k \times \{0, 1\}^{b-k} \times N$
Output: $S \in \{0, 1\}^{\ell}$

Underlying keyed duplex: $KD[p](K)$

$S \leftarrow \emptyset$
$KD$.init$(1, U)$

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

<table>
<thead>
<tr>
<th>resources of $D'$</th>
<th>in terms of</th>
<th>resources of $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$: data complexity (calls to construction)</td>
<td>$\rightarrow$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>$N$: time complexity (calls to primitive)</td>
<td>$\rightarrow$</td>
<td>$N$</td>
</tr>
<tr>
<td>$Q$: number of init calls</td>
<td>$\rightarrow$</td>
<td>$q$</td>
</tr>
<tr>
<td>$Q_{IV}$: max # init calls for single $IV$</td>
<td>$\rightarrow$</td>
<td>$1$</td>
</tr>
<tr>
<td>$L$: # queries with repeated path</td>
<td>$\rightarrow$</td>
<td>$0$</td>
</tr>
<tr>
<td>$\Omega$: # queries with overwriting outer part</td>
<td>$\rightarrow$</td>
<td>$0$</td>
</tr>
</tbody>
</table>
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: $KD[p](K)$

$S \leftarrow \emptyset$

$KD$.$init(1, U)$

\begin{align*}
  &\text{for } i = 1, \ldots, \lceil \ell/r \rceil \text{ do} \\
  &\quad S \leftarrow S \parallel KD$.$duplex(false, 0^b)$
\end{align*}

\begin{align*}
  &\text{return } left_{\ell}(S)
\end{align*}

<table>
<thead>
<tr>
<th>resources of $D'$</th>
<th>in terms of</th>
<th>resources of $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$: data complexity (calls to construction)</td>
<td>$\longrightarrow\sigma$</td>
<td></td>
</tr>
<tr>
<td>$N$: time complexity (calls to primitive)</td>
<td>$\longrightarrow N$</td>
<td></td>
</tr>
<tr>
<td>$Q$: number of init calls</td>
<td>$\longrightarrow q$</td>
<td></td>
</tr>
<tr>
<td>$Q_{IV}$: max # init calls for single $IV$</td>
<td>$\longrightarrow 1$</td>
<td></td>
</tr>
<tr>
<td>$L$: # queries with repeated path</td>
<td>$\longrightarrow 0$</td>
<td></td>
</tr>
<tr>
<td>$\Omega$: # queries with overwriting outer part</td>
<td>$\longrightarrow 0$</td>
<td></td>
</tr>
</tbody>
</table>
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: \(KD[p](K)\)

\[ S \leftarrow \emptyset \]
KD.init(1, U)

for \(i = 1, \ldots, \left\lceil \ell/r \right\rceil\) do

\[ S \leftarrow S \parallel KD.duplex(false, 0^b) \]

return \(\text{left}_\ell(S)\)

\[
\text{Adv}_{KD}(D') \leq \frac{2^{2\sigma} \sigma^2 (N+1)}{2^c} + \frac{(\sigma-q)q}{2^b} + \frac{2(\sigma^2)}{2^b} + \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{N}{2^k}
\]

resources of D' in terms of resources of D

| M: data complexity (calls to construction) | \(\sigma\) |
| N: time complexity (calls to primitive) | \(N\) |
| Q: number of init calls | \(q\) |
| \(Q_{IV}\): max \# init calls for single IV | 1 |
| L: \# queries with repeated path | 0 |
| \(\Omega\): \# queries with overwriting outer part | 0 |

From [DMV17] (in single-user setting):
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: $KD[p](K)$ |
| $(P_1, P_2, \ldots, P_w) \leftarrow \text{pad}_{10^*}^b(P)$ |
| $T \leftarrow \emptyset$ |
| $KD$.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$(false, 0^b)$ |
| return $\text{left}_t(T)$ |
**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: $(K, IV, P) \in \{0, 1\}^k \times TV \times \{0, 1\}^*$ |
| Output: $T \in \{0, 1\}^t$ |

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

$(P_1, P_2, \ldots, P_w) \leftarrow \text{pad}_{10^*}(P)$
$T \leftarrow \emptyset$
KD.init(1, IV)
for $i = 1, \ldots, w$ do
  KD.duplex($false, P_i$) \hspace{1cm} ▷ discard output
for $i = 1, \ldots, \lceil t/r \rceil$ do
  $T \leftarrow T \parallel$ KD.duplex($false, 0^b$)
return 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 $\text{FSKS}[p]$:  
  - Comparable analysis as for $\text{SC}[p]$
  - ... but distinguisher can repeat paths

Algorithm Full-state keyed sponge $\text{FSKS}[p]$

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

$(P_1, P_2, \ldots, P_w) \leftarrow \text{pad}_{b^*}(P)$

$T \leftarrow \emptyset$

$\text{KD}.\text{init}(1, IV)$

for $i = 1, \ldots, w$ do
  $\text{KD}.\text{duplex}(false, P_i)$  \hspace{1cm} $\triangleright$ discard output

for $i = 1, \ldots, \lceil t/r \rceil$ do
  $T \leftarrow T \parallel \text{KD}.\text{duplex}(false, 0^b)$

return $\text{left}_t(T)$
• 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
  • Impacts resources of $D'$

```
Algorithm Full-state keyed sponge FSKS$[p]$

Input: $(K, IV, P) \in \{0,1\}^k \times IV \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
Underlying keyed duplex: $KD[p](K)$
$(P_1, P_2, \ldots, P_w) \leftarrow \text{pad}_{10^*}(P)$
$T \leftarrow \emptyset$
$KD$.init$(1, IV)$
for $i = 1, \ldots, w$ do
  $KD$.duplex$(false, P_i)$ \quad \triangleright \text{discard output}$
for $i = 1, \ldots, \lceil t/r \rceil$ do
  $T \leftarrow T \parallel KD$.duplex$(false, 0^b)$
return $\left.t(T)\right.$
```
• Consider distinguisher $D$ against PRF security of $\text{FSKS}[p]$

$$\text{Adv}^{\text{prf}}_{\text{FSKS}}(D) = \Delta_D \left( \text{FSKS}[p]_K, p^\pm ; R^{\text{prf}}, p^\pm \right)$$

• $D$ can make $q$ construction queries (total $\sigma$ blocks) + $N$ primitive queries
• Consider distinguisher $D$ against PRF security of $\text{FSKS}[p]$

$$
\text{Adv}^\text{prf}_{\text{FSKS}}(D) = \Delta_D \left( \text{FSKS}[p]_K, p^\pm ; R^\text{prf}, p^\pm \right)
$$

• $D$ can make $q$ construction queries (total $\sigma$ blocks) + $N$ primitive queries

• Triangle inequality: $\text{Adv}^\text{prf}_{\text{FSKS}}(D) \leq \Delta_D' \left( \text{KD}[p]_K, p^\pm ; \text{IXIF}[\rho], p^\pm \right)$
• Consider distinguisher $D$ against PRF security of $\text{FSKS}[p]$

$$\operatorname{Adv}_{\text{FSKS}}^{\text{prf}}(D) = \Delta_D \left( \text{FSKS}[p]_K, p^\pm ; R^{\text{prf}}, p^\pm \right)$$

• $D$ can make $q$ construction queries (total $\sigma$ blocks) + $N$ primitive queries

• Triangle inequality: $\operatorname{Adv}_{\text{FSKS}}^{\text{prf}}(D) \leq \Delta_{D'} \left( \text{KD}[p]_K, p^\pm ; \text{IXIF}[ro], p^\pm \right)$

• What are the resources of $D'$?
• Consider distinguisher $D$ against PRF security of $\text{FSKS}[p]$

$$\text{Adv}^\text{prf}_{\text{FSKS}}(D) = \Delta_D \left( \text{FSKS}[p]_K, p^\pm ; \text{R}^\text{prf}, p^\pm \right)$$

• $D$ can make $q$ construction queries (total $\sigma$ blocks) + $N$ primitive queries

• Triangle inequality: $\text{Adv}^\text{prf}_{\text{FSKS}}(D) \leq \Delta_{D'} (\text{KD}[p]_K, p^\pm ; \text{IXIF}[\sigma], p^\pm)$

• What are the resources of $D'$?

<table>
<thead>
<tr>
<th>resources of $D'$</th>
<th>in terms of</th>
<th>resources of $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$: data complexity (calls to construction)</td>
<td>$\Delta D'$</td>
<td>$\Delta D$</td>
</tr>
<tr>
<td>$N$: time complexity (calls to primitive)</td>
<td>$\Delta D'$</td>
<td>$\Delta D$, $\Delta D$, $\Delta D$, $\Delta D$</td>
</tr>
<tr>
<td>$Q$: number of init calls</td>
<td>$\Delta D'$</td>
<td>$\Delta D$, $\Delta D$, $\Delta D$, $\Delta D$</td>
</tr>
<tr>
<td>$Q_{IV}$: max # init calls for single IV</td>
<td>$\Delta D'$</td>
<td>$\Delta D$, $\Delta D$, $\Delta D$, $\Delta D$</td>
</tr>
<tr>
<td>$L$: # queries with repeated path</td>
<td>$\Delta D'$</td>
<td>$\Delta D$, $\Delta D$, $\Delta D$, $\Delta D$</td>
</tr>
<tr>
<td>$\Omega$: # queries with overwriting outer part</td>
<td>$\Delta D'$</td>
<td>$\Delta D$, $\Delta D$, $\Delta D$, $\Delta D$</td>
</tr>
</tbody>
</table>
• Consider distinguisher $D$ against PRF security of $\text{FSKS}[p]$

$$\text{Adv}^{\text{prf}}_{\text{FSKS}}(D) = \Delta_D \left( \text{FSKS}[p]_K, p^\pm ; R^{\text{prf}}, p^\pm \right)$$

• $D$ can make $q$ construction queries (total $\sigma$ blocks) + $N$ primitive queries

• Triangle inequality: $\text{Adv}^{\text{prf}}_{\text{FSKS}}(D) \leq \Delta_{D'} \left( \text{KD}[p]_K, p^\pm ; \text{IXIF}[\text{ro}], p^\pm \right)$

• What are the resources of $D'$?

<table>
<thead>
<tr>
<th>resources of $D'$</th>
<th>in terms of</th>
<th>resources of $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$: data complexity (calls to construction)</td>
<td>$\longrightarrow$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>$N$: time complexity (calls to primitive)</td>
<td>$\longrightarrow$</td>
<td>$N$</td>
</tr>
<tr>
<td>$Q$: number of init calls</td>
<td>$\longrightarrow$</td>
<td>$q$</td>
</tr>
<tr>
<td>$Q_{IV}$: max # init calls for single $IV$</td>
<td>$\longrightarrow$</td>
<td>1</td>
</tr>
<tr>
<td>$L$: # queries with repeated path</td>
<td>$\longrightarrow$</td>
<td></td>
</tr>
<tr>
<td>$\Omega$: # queries with overwriting outer part</td>
<td>$\longrightarrow$</td>
<td></td>
</tr>
</tbody>
</table>
• Consider distinguisher $D$ against PRF security of $\text{FSKS}[p]$

\[
\text{Adv}_{\text{FSKS}}^{\text{prf}}(D) = \Delta_D \left( \text{FSKS}[p]_K, p^\pm ; R^{\text{prf}}, p^\pm \right)
\]

• $D$ can make $q$ construction queries (total $\sigma$ blocks) + $N$ primitive queries

• Triangle inequality: $\text{Adv}_{\text{FSKS}}^{\text{prf}}(D) \leq \Delta_{D'} (\text{KD}[p]_K, p^\pm ; \text{IXIF}[\rho], p^\pm)$

• What are the resources of $D'$?

<table>
<thead>
<tr>
<th>resources of $D'$</th>
<th>in terms of</th>
<th>resources of $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$: data complexity (calls to construction)</td>
<td>$\rightarrow$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>$N$: time complexity (calls to primitive)</td>
<td>$\rightarrow$</td>
<td>$N$</td>
</tr>
<tr>
<td>$Q$: number of init calls</td>
<td>$\rightarrow$</td>
<td>$q$</td>
</tr>
<tr>
<td>$Q_{IV}$: max # init calls for single $IV$</td>
<td>$\rightarrow$</td>
<td>1</td>
</tr>
<tr>
<td>$L$: # queries with repeated path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Omega$: # queries with overwriting outer part</td>
<td>$\rightarrow$</td>
<td>0</td>
</tr>
</tbody>
</table>
• Consider distinguisher $D$ against PRF security of $\text{FSKS}[p]$

$$\text{Adv}_{\text{FSKS}}^{\text{prf}}(D) = \Delta_D \left( \text{FSKS}[p]^K, p^\pm ; R^{\text{prf}}, p^\pm \right)$$

• $D$ can make $q$ construction queries (total $\sigma$ blocks) + $N$ primitive queries

• Triangle inequality: $\text{Adv}_{\text{FSKS}}^{\text{prf}}(D) \leq \Delta_{D'} \left( \text{KD}[p]^K, p^\pm ; \text{IXIF}[\text{ro}], p^\pm \right)$

• What are the resources of $D'$?

<table>
<thead>
<tr>
<th>resources of $D'$</th>
<th>in terms of</th>
<th>resources of $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$: data complexity (calls to construction)</td>
<td>$\rightarrow$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>$N$: time complexity (calls to primitive)</td>
<td>$\rightarrow$</td>
<td>$N$</td>
</tr>
<tr>
<td>$Q$: number of init calls</td>
<td>$\rightarrow$</td>
<td>$q$</td>
</tr>
<tr>
<td>$Q_{IV}$: max # init calls for single $IV$</td>
<td>$\rightarrow$</td>
<td>$1$</td>
</tr>
<tr>
<td>$L$: # queries with repeated path</td>
<td>$\rightarrow$</td>
<td>$\leq q - 1$</td>
</tr>
<tr>
<td>$\Omega$: # queries with overwriting outer part</td>
<td>$\rightarrow$</td>
<td>$0$</td>
</tr>
</tbody>
</table>
Full-State Keyed Sponge: Security

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

\[ \text{Adv}^{\text{prf}}_{\text{FSKS}}(D) = \Delta_D \left( \text{FSKS}[p]^K, p^+; R^{\text{prf}}, p^\pm \right) \]

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

- Triangle inequality: \( \text{Adv}^{\text{prf}}_{\text{FSKS}}(D) \leq \Delta_{D'} \left( \text{KD}[p]^K, p^+; \text{IXIF}[\text{ro}], p^\pm \right) \)

- What are the resources of D’?

<table>
<thead>
<tr>
<th>resources of D’</th>
<th>in terms of</th>
<th>resources of D</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M ): data complexity (calls to construction)</td>
<td>( \rightarrow )</td>
<td>σ</td>
</tr>
<tr>
<td>( N ): time complexity (calls to primitive)</td>
<td>( \rightarrow )</td>
<td>N</td>
</tr>
<tr>
<td>( Q ): number of init calls</td>
<td>( \rightarrow )</td>
<td>q</td>
</tr>
<tr>
<td>( Q_{IV} ): max # init calls for single IV</td>
<td>( \rightarrow )</td>
<td>1</td>
</tr>
<tr>
<td>( L ): # queries with repeated path</td>
<td>( \rightarrow )</td>
<td>( \leq q - 1 )</td>
</tr>
<tr>
<td>( \Omega ): # queries with overwriting outer part</td>
<td>( \rightarrow )</td>
<td>0</td>
</tr>
</tbody>
</table>

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

\[ \text{Adv}^{\text{KD}}_{\text{D}'}(D') \leq \frac{2^{\nu(\sigma+1)k}}{2^c} + \frac{(q-1)N+q}{2^c} + \frac{(\sigma-q)q}{2^b} + \frac{2(\sigma^2)}{2^b} + \frac{q(\sigma-q)}{2^k} + \frac{N}{2^k} \]
Consider distinguisher $D$ against PRF security of $FSKS[p]$

$$\text{Adv}_{FSKS}^{prf}(D) = \Delta_D \left( FSKS[p]_K, p^\pm ; R^{prf}, p^\pm \right)$$

$D$ can make $q$ construction queries (total $\sigma$ blocks) + $N$ primitive queries

Triangle inequality: $\text{Adv}_{FSKS}^{prf}(D) \leq \Delta_{D'} \left( KD[p]_K, p^\pm ; IXIF[ro], p^\pm \right)$

What are the resources of $D'$?

<table>
<thead>
<tr>
<th>resources of $D'$</th>
<th>in terms of</th>
<th>resources of $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$: data complexity (calls to construction)</td>
<td>$\longrightarrow$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>$N$: time complexity (calls to primitive)</td>
<td>$\longrightarrow$</td>
<td>$N$</td>
</tr>
<tr>
<td>$Q$: number of init calls</td>
<td>$\longrightarrow$</td>
<td>$q$</td>
</tr>
<tr>
<td>$Q_{IV}$: max # init calls for single IV</td>
<td>$\longrightarrow$</td>
<td>$1$</td>
</tr>
<tr>
<td>$L$: # queries with repeated path</td>
<td>$\longrightarrow$</td>
<td>$\leq q - 1$</td>
</tr>
<tr>
<td>$\Omega$: # queries with overwriting outer part</td>
<td>$\longrightarrow$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

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

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

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

• State of second query before squeezing equals $0^r || 0^c$
• 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 $P \| T \| 0^c \mapsto T'$

\[
\begin{align*}
K & \quad \rightarrow \\
\downarrow & & \downarrow \\
p & & p \\
\leftarrow & & \leftarrow \\
\frac{0^{b-k}}{b-k} & & \frac{0^{b-k}}{b-k} \\
\text{init} & & \text{init} \\
\text{duplex} & & \text{duplex} \\
\text{duplex} & & \text{duplex} \\
\end{align*}
\]

• State of second query before squeezing equals $0^r \| \star^c$
• Key recovery attack:
  • Make $q$ twin queries as above and $N$ primitive queries of form $0^r \| \star^c$
  • Construction-primitive collision (likely if $\frac{q \cdot N}{2^c} \approx 1$) $\longrightarrow$ derive $K$
• Input: key $K$, initial value $IV$, message $P$
• Output: tag $T$

Algorithm Ascon-PRF[$p$]

Input: $(K, IV, P) \in \{0, 1\}^k \times IV \times \{0, 1\}^*$
Output: $T \in \{0, 1\}^t$
Underlying keyed duplex: $KD[p](K)$
$(P_1, P_2, \ldots, P_w) \leftarrow \text{pad}_r^{10^*}(P)$
$T \leftarrow \emptyset$
$KD$.init($1, IV$)
for $i = 1, \ldots, w - 1$ do
  $KD$.duplex($false, P_i$) ▷ discard output
$KD$.duplex($false, P_w || 0^{c-1}$)
for $i = 1, \ldots, \lceil t/r \rceil$ do
  $T \leftarrow T \parallel KD$.duplex($false, 0^b$)
return left$_t(T)$
Input: key $K$, initial value $IV$, message $P$
Output: tag $T$

Domain separation solves problem of repeated paths

**Algorithm** Ascon-PRF[$p$]

**Input:** $(K, IV, P) \in \{0,1\}^k \times TV \times \{0,1\}^*$
**Output:** $T \in \{0,1\}^t$

Underlying keyed duplex: $KD[p](K)$

$(P_1, P_2, \ldots, P_w) \leftarrow \text{pad}_r^{10^*}(P)$

$T \leftarrow \emptyset$

$KD$.init$(1, IV)$

for $i = 1, \ldots, w - 1$ do

KD.duplex($false, P_i$) ▷ discard output

KD.duplex($false, P_w \parallel 0^{c-1}1$)

for $i = 1, \ldots, \lceil t/r \rceil$ do

$T \leftarrow T \parallel KD$.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...

---

**Algorithm** Ascon-PRF[$p$]

- **Input**: $(K, IV, P) \in \{0,1\}^k \times \mathcal{T} \times \{0,1\}^*$
- **Output**: $T \in \{0,1\}^t$
- **Underlying keyed duplex**: $KD[p](K)$
  - $(P_1, P_2, \ldots, P_w) \leftarrow \text{pad}_{10^*}(P)$
  - $T \leftarrow \varnothing$
  - $KD.\text{init}(1, IV)$
  - for $i = 1, \ldots, w - 1$ do
    - $KD.\text{duplex}(false, P_i)$ ▷ discard output
  - $KD.\text{duplex}(false, P_w || 0^{c-1}1)$
  - for $i = 1, \ldots, [t/r]$ do
    - $T \leftarrow T \parallel KD.\text{duplex}(false, 0^b)$
  - return $\text{left}_t(T)$
• **Input:** key $K$, initial value $IV$, message $P$
• **Output:** tag $T$

**Domain separation** solves problem of repeated paths
• Repeated paths may still occur...
• ...but adversary cannot exploit them

---

**Algorithm Ascon-PRF**

**Input:** $(K, IV, P) \in \{0, 1\}^k \times TV \times \{0, 1\}^*$

**Output:** $T \in \{0, 1\}^t$

**Underlying keyed duplex:** $KD[p](K)$

$(P_1, P_2, \ldots, P_w) \leftarrow pad^{10^*}(P)$

$T \leftarrow \emptyset$

$KD.init(1, IV)$

for $i = 1, \ldots, w - 1$ do

$KD.duplex(false, P_i)$  \hspace{1cm} \text{\texttt{discard output}}

$KD.duplex(false, P_w || 0^{c-1}1)$

for $i = 1, \ldots, \lceil t/r \rceil$ do

$T \leftarrow T \parallel KD.duplex(false, 0^{b})$

return left$_t(T)$
• Unfortunately, (bounds on) the resources of D’ do not change:
Unfortunately, (bounds on) the resources of $D'$ do not change:

<table>
<thead>
<tr>
<th>resources of $D'$</th>
<th>in terms of</th>
<th>resources of $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$: data complexity (calls to construction)</td>
<td>$\rightarrow \sigma$</td>
<td></td>
</tr>
<tr>
<td>$N$: time complexity (calls to primitive)</td>
<td>$\rightarrow N$</td>
<td></td>
</tr>
<tr>
<td>$Q$: number of init calls</td>
<td>$\rightarrow q$</td>
<td></td>
</tr>
<tr>
<td>$Q_{IV}$: max # init calls for single $IV$</td>
<td>$\rightarrow 1$</td>
<td></td>
</tr>
<tr>
<td>$L$: # queries with repeated path</td>
<td>$\rightarrow \leq q - 1$</td>
<td></td>
</tr>
<tr>
<td>$\Omega$: # queries with overwriting outer part</td>
<td>$\rightarrow 0$</td>
<td></td>
</tr>
</tbody>
</table>
Ascon-PRF: Security

• Unfortunately, (bounds on) the resources of D’ **do not change**:

<table>
<thead>
<tr>
<th>resources of D’</th>
<th>in terms of</th>
<th>resources of D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$: data complexity (calls to construction)</td>
<td>$\rightarrow$ $\sigma$</td>
<td>$N$: time complexity (calls to primitive)</td>
</tr>
<tr>
<td>$N$: time complexity (calls to primitive)</td>
<td>$\rightarrow$ $N$</td>
<td>$Q$: number of init calls</td>
</tr>
<tr>
<td>$Q$: number of init calls</td>
<td>$\rightarrow$ $q$</td>
<td>$Q_{IV}$: max # init calls for single IV</td>
</tr>
<tr>
<td>$Q_{IV}$: max # init calls for single IV</td>
<td>$\rightarrow$ $1$</td>
<td>$L$: # queries with repeated path</td>
</tr>
<tr>
<td>$L$: # queries with repeated path</td>
<td>$\rightarrow$ $\leq q - 1$</td>
<td>$\Omega$: # queries with overwriting outer part</td>
</tr>
<tr>
<td>$\Omega$: # queries with overwriting outer part</td>
<td>$\rightarrow$ $0$</td>
<td></td>
</tr>
</tbody>
</table>

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

• Unfortunately, (bounds on) the resources of $D'$ **do not change**:

<table>
<thead>
<tr>
<th>resources of $D'$</th>
<th>in terms of</th>
<th>resources of $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$: data complexity (calls to construction)</td>
<td>$\rightarrow$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>$N$: time complexity (calls to primitive)</td>
<td>$\rightarrow$</td>
<td>$N$</td>
</tr>
<tr>
<td>$Q$: number of init calls</td>
<td>$\rightarrow$</td>
<td>$q$</td>
</tr>
<tr>
<td>$Q_{IV}$: max # init calls for single IV</td>
<td>$\rightarrow$</td>
<td>$1$</td>
</tr>
<tr>
<td>$L$: # queries with repeated path</td>
<td>$\rightarrow$</td>
<td>$\leq q - 1$</td>
</tr>
<tr>
<td>$\Omega$: # queries with overwriting outer part</td>
<td>$\rightarrow$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

• Improved bound from [DMV17]:
  • Loose bounding in original proof
  • Resolving this loose bounding makes $\frac{(q-1)N+(\frac{q}{2})}{2^c}$ 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 $\frac{(q-1)N+(\frac{q}{2})}{2^c}$ never appears in the first place
Ascon-PRF: Implication

Multi-user bound from [DMV17]

\[ \text{Adv}_{\text{Ascon-PRF}}^{\mu-\text{prf}}(D) \leq \frac{2\nu \sigma (N+1)}{2^c} + \frac{(\sigma-q)q}{2^b} + \frac{2\sigma}{2^b} + \frac{q(\sigma-q)}{2\min\{c+k,b\}} + \frac{\mu N}{2^k} + \frac{(\mu^2}{2^k} \]
Ascon-PRF: Implication

Multi-user bound from [DMV17]

\[
\text{Adv}^\mu_{\text{Ascon-PRF}}(D) \leq \frac{2\nu \sigma^2 (N+1)}{2^c} + \frac{(\sigma-q)q}{2^b-q} + \frac{2^{(\frac{\sigma}{2})}}{2^b} + \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{(\frac{\mu}{2})}{2^k}
\]

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 \(\nu^2_{128,192}\) is at most 5
Ascon-PRF: Implication

Multi-user bound from [DMV17]

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

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 \(\nu_{128, 192}^{2^{65}}\) is at most 5
Ascon-PRF: Implication

Multi-user bound from [DMV17]

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

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 \(\nu_{128,192}^{2^{65}}\) is at most 5
• Generic security as long as \(N \ll \frac{2^{128}}{\mu}\)
Duplex Application: MonkeySpongeWrap
Authenticated Encryption

key $K$

associated data $A$
plaintext $P$

nonce $U$

AE

ciphertext $C$
tag $T$

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)
Authenticated Encryption

- **Key** $K$
- **Plaintext** $P$
- **Associated Data** $A$
- **Nonce** $U$
- **Ciphertext** $C$
- **Tag** $T$

**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.)

$$AE^{-1} \begin{cases} P & \text{if } T \text{ correct} \\ \bot & \text{otherwise} \end{cases}$$
Authenticated Encryption

Role of Duplex

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

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

$$AE$$

- key $K$
- associated data $A$
- plaintext $P$
- nonce $U$

$$AE^{-1}$$

- key $K$
- associated data $A$
- ciphertext $C$
- tag $T$
- nonce $U$

$P$ if $T$ correct
$\bot$ otherwise
Authenticated Encryption

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 $\text{flag} = \text{true}$
• 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 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
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

<table>
<thead>
<tr>
<th>property</th>
<th>setting</th>
<th>security as long as (highly simplified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>confidentiality</td>
<td>nonce-respecting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nonce-misuse</td>
<td></td>
</tr>
<tr>
<td>authenticity</td>
<td>nonce-respecting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nonce-misuse</td>
<td></td>
</tr>
</tbody>
</table>
## Multi-User Security Under Typical Models

<table>
<thead>
<tr>
<th>property</th>
<th>setting</th>
<th>security as long as (highly simplified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>confidentiality</td>
<td>nonce-respecting</td>
<td>(N \ll \min{2^k/\mu, 2^{b/2}, 2^c})</td>
</tr>
<tr>
<td></td>
<td>nonce-misuse</td>
<td>—</td>
</tr>
<tr>
<td>authenticity</td>
<td>nonce-respecting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nonce-misuse</td>
<td></td>
</tr>
</tbody>
</table>
## Multi-User Security Under Typical Models

<table>
<thead>
<tr>
<th>property</th>
<th>setting</th>
<th>security as long as (highly simplified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>confidentiality</td>
<td>nonce-respecting</td>
<td>$N \ll \min{2^k/\mu, 2^{b/2}, 2^c}$</td>
</tr>
<tr>
<td></td>
<td>nonce-misuse</td>
<td>—</td>
</tr>
<tr>
<td>authenticity</td>
<td>nonce-respecting</td>
<td>$N \ll \min{2^k/\mu, 2^b/\sigma_E, 2^c/\sigma_D}$</td>
</tr>
<tr>
<td></td>
<td>nonce-misuse</td>
<td>$N \ll \min{2^k/\mu, 2^c/\left(\sigma_E + \sigma_D\right)}$</td>
</tr>
</tbody>
</table>

- Application to Ascon-AEAD Parameters
  - $(k, b, c, r, t) = (128, 320, 256, 64, 128)$ for Ascon-128
  - $(128, 320, 192, 128, 128)$ for Ascon-128a
  - $(160, 320, 256, 64, 128)$ for Ascon-80pq

- Assume online complexity of $q, \sigma_E < 2^{64}$ (could be taken higher)
## Multi-User Security Under Typical Models

<table>
<thead>
<tr>
<th>Property</th>
<th>Setting</th>
<th>Security as long as (highly simplified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidentiality</td>
<td>Nonce-respecting</td>
<td>$N \ll \min{2^k/\mu, 2^{b/2}, 2^c}$</td>
</tr>
<tr>
<td></td>
<td>Nonce-misuse</td>
<td>—</td>
</tr>
<tr>
<td>Authenticity</td>
<td>Nonce-respecting</td>
<td>$N \ll \min{2^k/\mu, 2^b/\sigma_E, 2^c/\sigma_D}$</td>
</tr>
<tr>
<td></td>
<td>Nonce-misuse</td>
<td>$N \ll \min{2^k/\mu, 2^c/\sigma_E + \sigma_D}$</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>property</th>
<th>setting</th>
<th>security as long as (highly simplified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>confidentiality</td>
<td>nonce-respecting</td>
<td>( N \ll \min{2^k/\mu, 2^b/2, 2^c} )</td>
</tr>
<tr>
<td></td>
<td>nonce-misuse</td>
<td>—</td>
</tr>
<tr>
<td>authenticity</td>
<td>nonce-respecting</td>
<td>( N \ll \min{2^k/\mu, 2^b/\sigma_E, 2^c/\sigma_D} )</td>
</tr>
<tr>
<td></td>
<td>nonce-misuse</td>
<td>( N \ll \min{2^k/\mu, 2^c/(\sigma_E + \sigma_D)} )</td>
</tr>
</tbody>
</table>

**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
Authenticity Under State Recovery (1)

**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
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 $\frac{\mu N}{2^k} + \frac{(\mu^2)}{2^{2k}}$
  - Assumes that attacker has full control over $IV$

\[ K[\delta] \xrightarrow{\init} p \xrightarrow{\duplex} \cdots \]

\[ IV \xrightarrow{\init} p \xrightarrow{\duplex} \cdots \]
On the Power of Initialization

- Plain initialization: incurs term $\frac{\mu N}{2^k} + \frac{\left(\frac{\mu}{2}\right)}{2^k}$
  - 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
### Different Initializations

<table>
<thead>
<tr>
<th>case</th>
<th>initL($K, \delta, i$)</th>
<th>initR($K, \delta, i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>$K[\delta]$</td>
<td>encode$_{b-k}[\hat{i}]$</td>
</tr>
<tr>
<td>global $IV$</td>
<td>$K[\delta]$</td>
<td>encode$_{b-k}[(\delta, i)]$</td>
</tr>
<tr>
<td>random $IV$</td>
<td>$K[\delta]$</td>
<td>$RIV \parallel 0^{b-k-n}$</td>
</tr>
<tr>
<td>quasi-random $IV$</td>
<td>$K[\delta]$</td>
<td>$(RIV_\delta \oplus encode_n[i]) \parallel 0^{b-k-n}$</td>
</tr>
<tr>
<td>$IV$ on key</td>
<td>$K[\delta] \oplus encode_k[i]$</td>
<td>$0^{b-k}$</td>
</tr>
<tr>
<td>global $IV$ on key</td>
<td>$K[\delta] \oplus encode_k[i]$</td>
<td>encode$_{b-k}[\delta]$</td>
</tr>
</tbody>
</table>

- Different types of initialization (see paper for side-conditions)
- $RIV$ stands for random $IV$, $RIV_\delta$ unique random $IV$ per user
### Different Initializations

<table>
<thead>
<tr>
<th>Case</th>
<th>initL((K, \delta, i))</th>
<th>initR((K, \delta, i))</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>(K[\delta])</td>
<td>encode_{b-k}[i]</td>
</tr>
<tr>
<td>global IV</td>
<td>(K[\delta])</td>
<td>encode_{b-k}[(\delta, i)]</td>
</tr>
<tr>
<td>random IV</td>
<td>(K[\delta])</td>
<td>RIV \Vert 0^{b-k-n}</td>
</tr>
<tr>
<td>quasi-random IV</td>
<td>(K[\delta])</td>
<td>(RIV_{\delta} \oplus encode_n[i]) \Vert 0^{b-k-n}</td>
</tr>
<tr>
<td>IV on key</td>
<td>(K[\delta] \oplus encode_k[i])</td>
<td>0^{b-k}</td>
</tr>
<tr>
<td>global IV on key</td>
<td>(K[\delta] \oplus encode_k[i])</td>
<td>encode_{b-k}[\delta]</td>
</tr>
</tbody>
</table>

- Different types of initialization (see paper for side-conditions)
- \(RIV\) stands for random IV, \(RIV_{\delta}\) unique random IV per user
- **Improved security bound** for optimized initialization
Different Initializations

<table>
<thead>
<tr>
<th>case</th>
<th>initL($K, \delta, i$)</th>
<th>initR($K, \delta, i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>$K[\delta]$</td>
<td>encode_{b-k}[i]</td>
</tr>
<tr>
<td>global IV</td>
<td>$K[\delta]$</td>
<td>encode_{b-k}[(\delta, i)]</td>
</tr>
<tr>
<td>random IV</td>
<td>$K[\delta]$</td>
<td>$RIV | 0^{b-k-n}$</td>
</tr>
<tr>
<td>quasi-random IV</td>
<td>$K[\delta]$</td>
<td>$(RIV_\delta \oplus encode_n[i])|0^{b-k-n}$</td>
</tr>
<tr>
<td>IV on key</td>
<td>$K[\delta] \oplus encode_k[i]$</td>
<td>$0^{b-k}$</td>
</tr>
<tr>
<td>global IV on key</td>
<td>$K[\delta] \oplus encode_k[i]$</td>
<td>encode_{b-k}[\delta]</td>
</tr>
</tbody>
</table>

- Different types of initialization (see paper for side-conditions)
- $RIV$ stands for random $IV$, $RIV_\delta$ unique random $IV$ per user
- **Improved security bound** for optimized initialization
- Application to keystream and authenticated encryption
Application to Keystream Generation (Randomized IV in Paper)

\[ \text{initL}(K, \delta, i) \]
\[ \text{initR}(K, \delta, i) \]

\[ \text{init} \quad \text{duplex} \quad \text{duplex} \quad \text{duplex} \quad \text{duplex} \]

\[ S_1 \quad S_2 \quad S_3 \quad S_4 \]

\[ k \quad r \quad c \quad r \quad c \quad r \quad c \quad r \]

33 / 34
Application to Keystream Generation (Randomized IV in Paper)

![Diagram of keystream generation](image)

<table>
<thead>
<tr>
<th>case</th>
<th>initL($K, \delta, i$)</th>
<th>initR($K, \delta, i$)</th>
<th>initialization term (simplified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>$K[\delta]$</td>
<td>encode$_{b-k}[i]$</td>
<td>$\frac{\mu N}{2^k} + \frac{(\mu^2)}{2^k}$</td>
</tr>
<tr>
<td>global IV</td>
<td>$K[\delta]$</td>
<td>encode$_{b-k}[(\delta, i)]$</td>
<td>$\frac{N}{2^k}$</td>
</tr>
<tr>
<td>IV on key</td>
<td>$K[\delta] \oplus$ encode$_k[i]$</td>
<td>$0^{b-k}$</td>
<td>$\frac{QN}{2^k} + \frac{(Q^2)}{2^k}$</td>
</tr>
<tr>
<td>global IV on key</td>
<td>$K[\delta] \oplus$ encode$_k[i]$</td>
<td>encode$_{b-k}[\delta]$</td>
<td>$\frac{Q\delta N}{2^k} + \frac{\mu(Q\delta^2)}{2^k}$</td>
</tr>
</tbody>
</table>

$Q$ stands for $\#$ initializations, $Q\delta$ initializations per user
Conclusion
Main Takeaways

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

Acknowledgments

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

Thank you for your attention!
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
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 Dobranig [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.*

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. 
*Sponge functions.*

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. 
*On the Indifferentiability of the Sponge Construction.*

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.


Christoph Dobraunig and Bart Mennink.  
**Leakage Resilience of the Duplex Construction.**  

Christoph Dobraunig and Bart Mennink.  
**Security of the Suffix Keyed Sponge.**  

Christoph Dobraunig and Bart Mennink.  
**Generalized Initialization of the Duplex Construction.**  
Joan Daemen, Bart Mennink, and Gilles Van Assche.  
**Full-State Keyed Duplex with Built-In Multi-user Support.**  

Peter Gaži, Krzysztof Pietrzak, and Stefano Tessaro.  
**The Exact PRF Security of Truncation: Tight Bounds for Keyed Sponges and Truncated CBC.**  

Charlotte Lefevre and Bart Mennink.  
**Tight Preimage Resistance of the Sponge Construction.**  
Charlotte Lefevre and Bart Mennink.  
**Generic Security of the Ascon Mode: On the Power of Key Blinding.**  

Bart Mennink.  
**Key Prediction Security of Keyed Sponges.**  

Bart Mennink.  
**Understanding the Duplex and Its Security.**  
to appear.