Radboud University

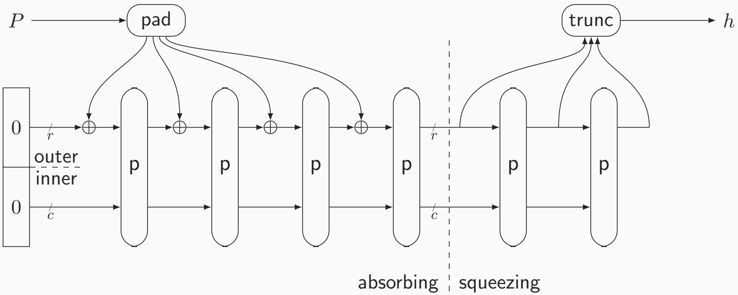
Understanding the Duplex and Its Security

Bart Mennink
Radboud University (The Netherlands)
Permutation-Based Crypto 2023, Lyon
April 23, 2023

ESCADA

History on Sponges and Duplexes

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, ...
- Behaves as RO up to query complexity $\approx 2^{c/2}$ [BDPV08]

Keyed Sponge

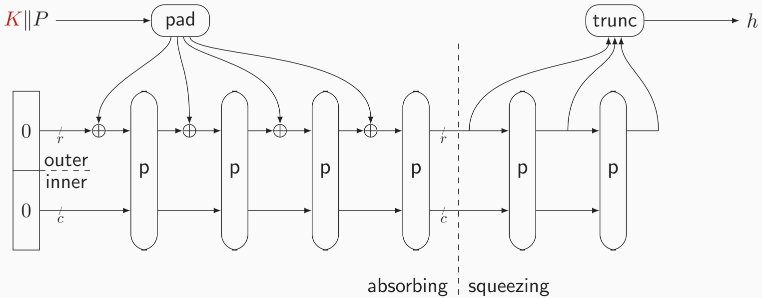
- $\text{PRF}(K, P) = \text{sponge}(K \| P)$
- Message authentication with tag size t: $\text{MAC}(K, P, t) = \text{sponge}(K \| P, t)$
- Keystream generation of length ℓ: $\text{SC}(K, D, \ell) = \text{sponge}(K \| D, \ell)$
- (All assuming K is fixed-length)

Keyed Sponge

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

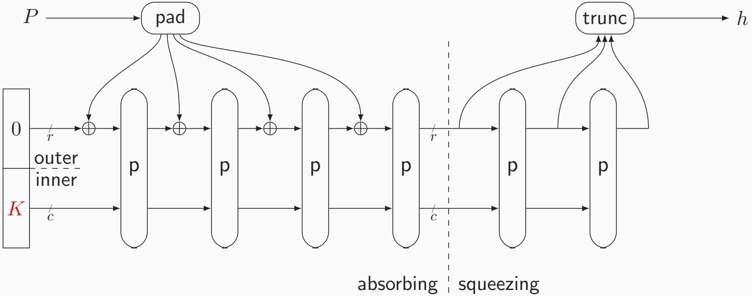
Keyed Duplex

- Authenticated encryption
- Multiple CAESAR and NIST LWC submissions



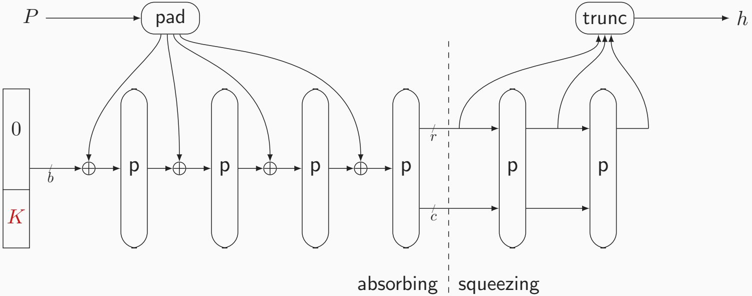
- Outer-Keyed Sponge [BDPV11b, ADMV15, NY16, Men18]

Evolution of Keyed Sponges



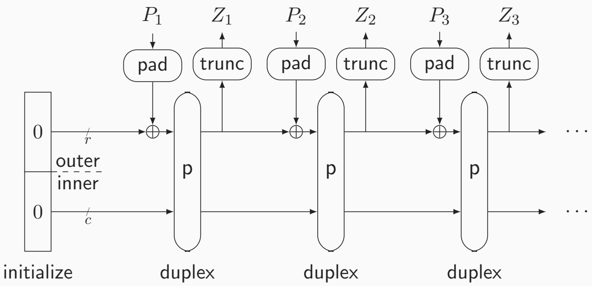
- Outer-Keyed Sponge [BDPV11b, ADMV15, NY16, Men18]
- Inner-Keyed Sponge [CDH+12, ADMV15, NY16]

Evolution of Keyed Sponges



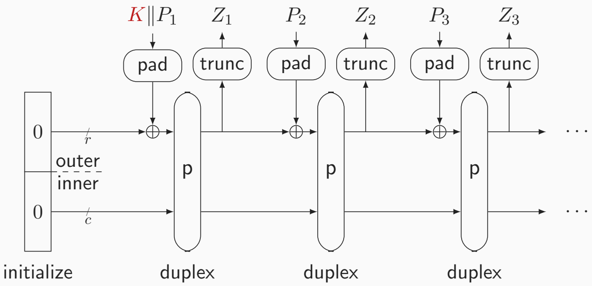
- Outer-Keyed Sponge [BDPV11b, ADMV15, NY16, Men18]
- Inner-Keyed Sponge [CDH^{+}12, ADMV15, NY16]
- Full-Keyed Sponge [BDPV12, GPT15, MRV15]

Evolution of Keyed Duplexes



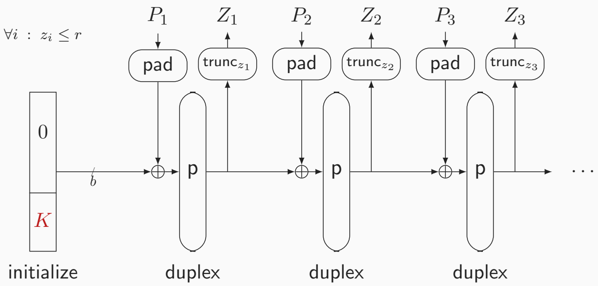
- Unkeyed Duplex [BDPV11a]

Evolution of Keyed Duplexes



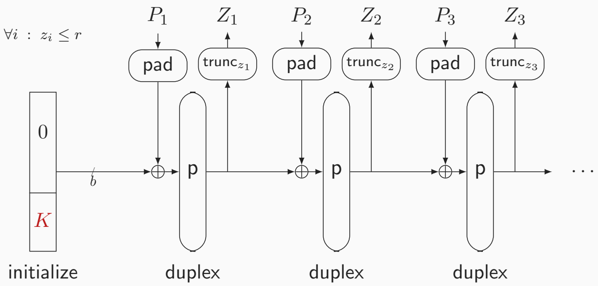
- Unkeyed Duplex [BDPV11a]
- Outer-Keyed Duplex [BDPV11a]

Evolution of Keyed Duplexes



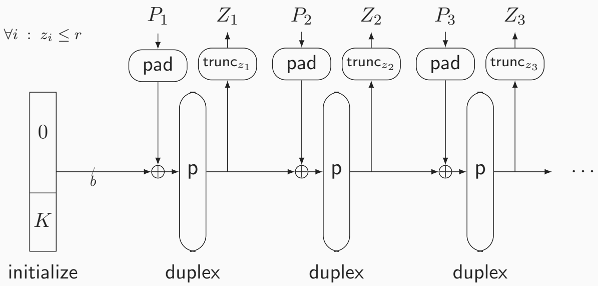
- Unkeyed Duplex [BDPV11a]
- Outer-Keyed Duplex [BDPV11a]
- Full-Keyed Duplex [MRV15, DMV17, DM19]

Evolution of Keyed Duplexes

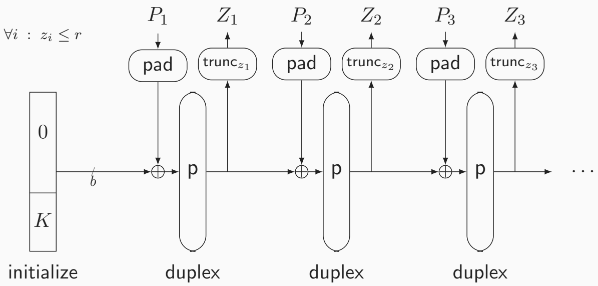


- Unkeyed Duplex [BDPV11a]
- Outer-Keyed Duplex [BDPV11a]
- Full-Keyed Duplex [MRV15, DMV17, DM19]

Full-Keyed Duplex of [MRV15] (1)



Full-Keyed Duplex of [MRV15] (1)

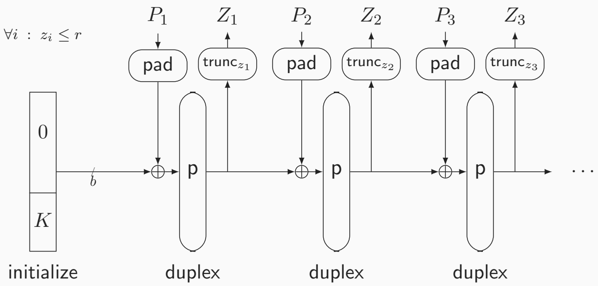


- M: data complexity (calls to construction)
- N: time complexity (calls to primitive)
- $\mu \leq 2M$: multiplicity ("maximum outer collision of p")

Simplified Security Bound

$$\frac{\mu N}{2^k} + \frac{M^2}{2^c}$$

Full-Keyed Duplex of [MRV15] (1)



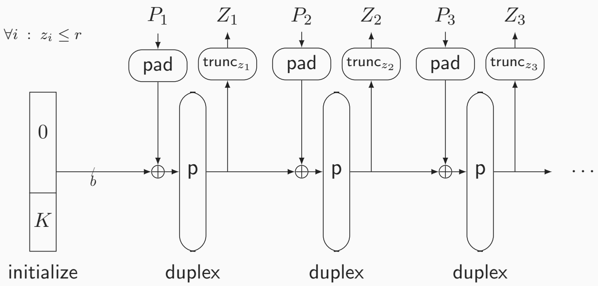
- M: data complexity (calls to construction)
- N: time complexity (calls to primitive)
- $\mu \leq 2M$: multiplicity ("maximum outer collision of p")

Simplified Security Bound

$$\frac{\mu N}{2^k} + \frac{M^2}{2^c}$$

scheme behaves "randomly" as long as this term is $\ll 1$

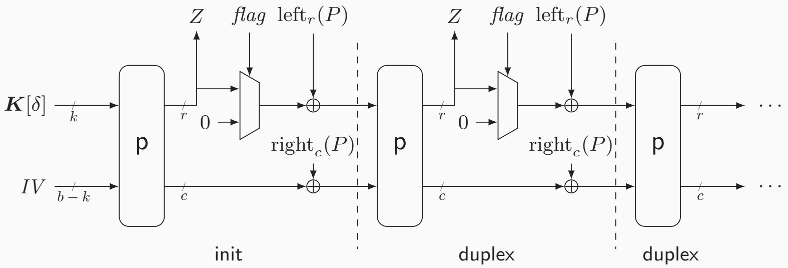
Full-Keyed Duplex of [MRV15] (2)

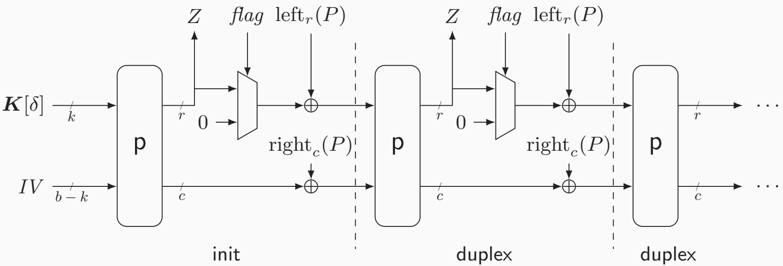


Limitations

- Multiplicity μ only known a posteriori
- Dominating term $\mu N/2^k$ rather than $\mu N/2^c$
- Limited flexibility in modeling adversarial power
  (multi-user security, blockwise adaptive behavior, nonces, ...)

Full-Keyed Duplex of [DMV17] (1)

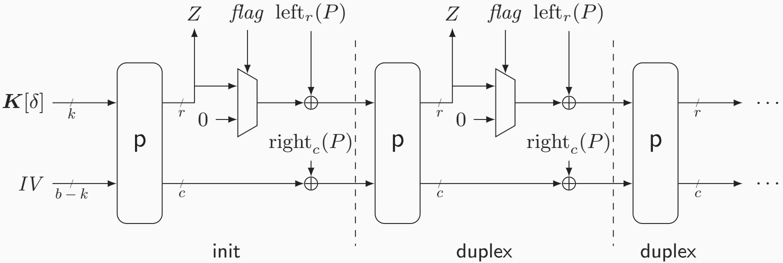




Features

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

Full-Keyed Duplex of [DMV17] (2)

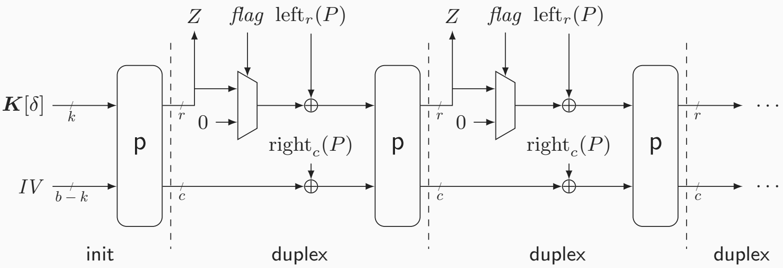


- M: data complexity (calls to construction)
- N: time complexity (calls to primitive)
- Q: number of init calls
- Q_{IV}: max # init calls for single IV
- L: # queries with repeated path (e.g., nonce-violation)
- Ω: # queries with overwriting outer part (e.g., RUP)
- $\nu_{r,c}^M$: some multicollision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

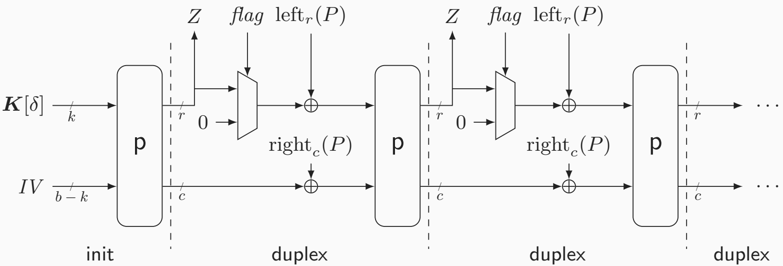
Simplified Security Bound

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

Full-Keyed Duplex of [DM19] (1)



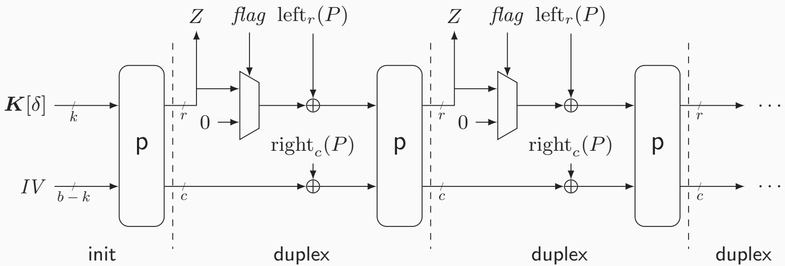
Full-Keyed Duplex of [DM19] (1)



Features

- Initialization can be rotated (not depicted)
- Another rephasing: $Z, P,$ p instead of p, Z, P instead of $P,$ p, Z
- Security analysis in leaky setting
- Even further refined adversarial strength
- Comparable bound

Full-Keyed Duplex of [DM19] (2)

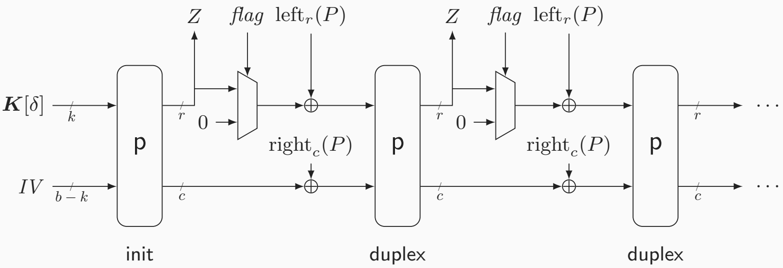


- M: data complexity (calls to construction)
- N: time complexity (calls to primitive)
- Q: number of init calls
- Q_{IV}: max # init calls for single IV
- Q_δ: maximum # init calls for single δ
- L: # queries with repeated path (e.g., nonce-violation)
- Ω: # queries with overwriting outer part (e.g.,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

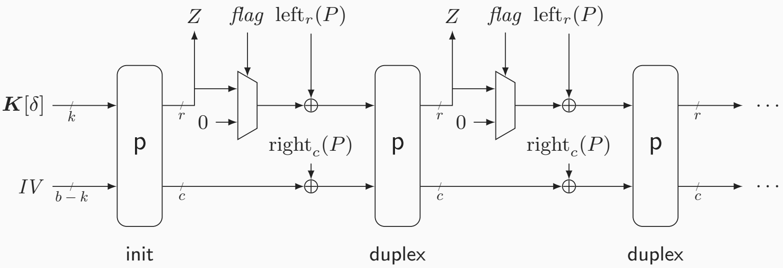
Simplified Security Bound

$$\frac{Q_{IV}N}{2^{k-Q_\delta\lambda}} + \frac{(L+\Omega+\nu_{r,c}^M)N}{2^{c-(R+1)\lambda}}$$

State of Affairs

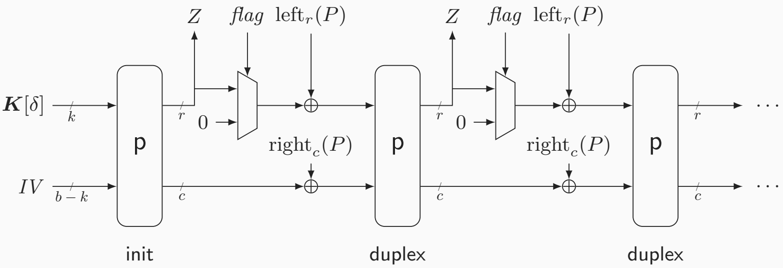


State of Affairs



Scheme: versatile but complex

- What about these rephasings?
- What about the flag?

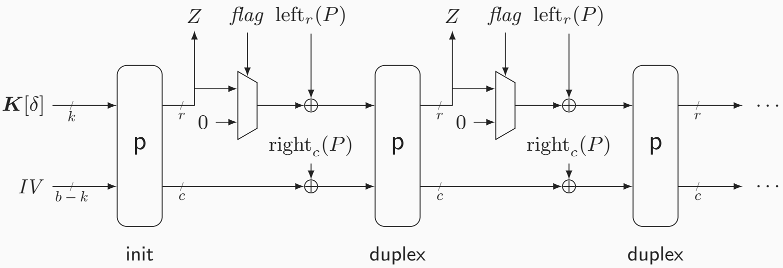


Scheme: versatile but complex

- What about these rephasings?
- What about the flag?

Security bounds: strong but complex

- Security parameters hard to understand
- Bound quickly misunderstood
- Unclear how use case affects bound



Scheme: versatile but complex

- What about these rephasings?
- What about the flag?

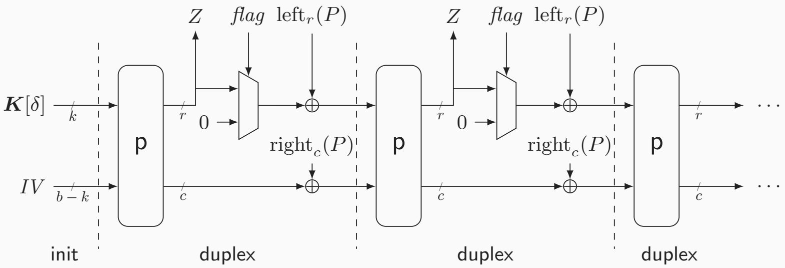
Security bounds: strong but complex

- Security parameters hard to understand
- Bound quickly misunderstood
- Unclear how use case affects bound

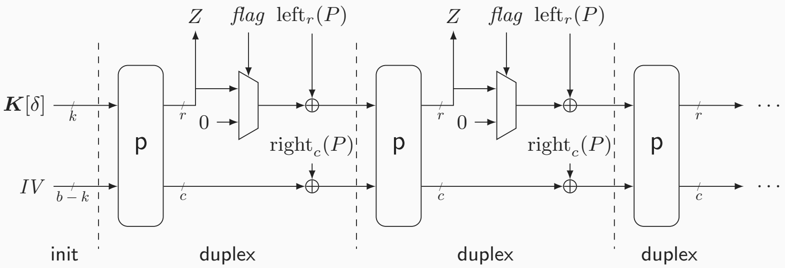
This work: explanation of the duplex, its security, and some applications

Understanding the Duplex

Generalized Keyed Duplex



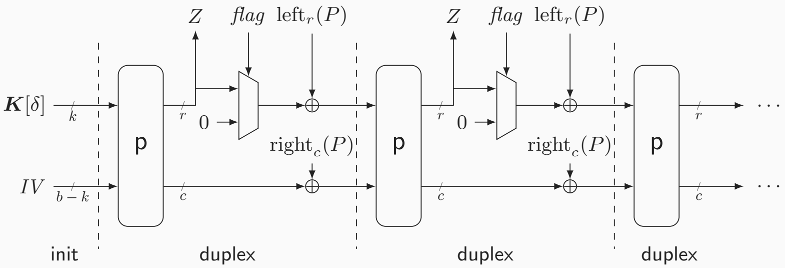
Generalized Keyed Duplex



Features

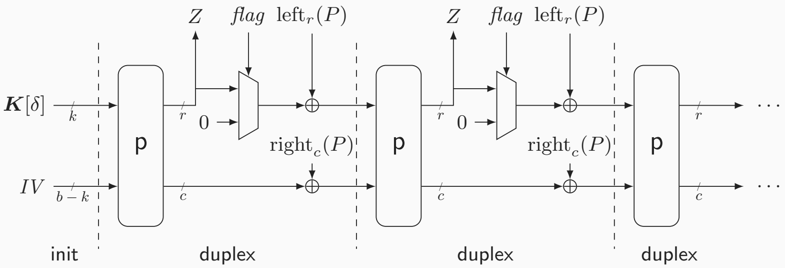
- Basically the scheme of [DMV17] and [DM19], but:
  - including possible initial state rotation (not depicted)
  - yet another rephasing

Generalized Keyed Duplex



Features

- Basically the scheme of [DMV17] and [DM19], but:
    - including possible initial state rotation (not depicted)
    - yet another rephasing
- Security results of [DMV17] and [DM19] carry over



Features

- Basically the scheme of [DMV17] and [DM19], but:
  - including possible initial state rotation (not depicted)
  - yet another rephasing
- Security results of [DMV17] and [DM19] carry over
- First: understanding phasing and flagging

Generalized Keyed Duplex: Phasing

	A	P	S	A	P	S	A	P	S	A	$\cdots$

Generalized Keyed Duplex: Phasing

	A	P	S	A	P	S	A	P	S	A	...
[BDPV11a]	init			duplex			duplex			...	

- [BDPV11a]: duplex security reduced to sponge indifferentiability

Generalized Keyed Duplex: Phasing

	A	P	S	A	P	S	A	P	S	A	⋯
[BDPV11a]	init			duplex			duplex			⋯	
[MRV15]	init			duplex			duplex			⋯	

- [BDPV11a]: duplex security reduced to sponge indifferentiability
- [MRV15]: same structure but tighter bound

Generalized Keyed Duplex: Phasing

<table>
<tr><th></th><th>A</th><th>P</th><th>S</th><th>A</th><th>P</th><th>S</th><th>A</th><th>P</th><th>S</th><th>A</th><th>···</th></tr>
<tr><td>[BDPV11a]</td><td colspan="3">init</td><td colspan="3">duplex</td><td colspan="3">duplex</td><td colspan="2">···</td></tr>
<tr><td>[MRV15]</td><td colspan="3">init</td><td colspan="3">duplex</td><td colspan="3">duplex</td><td colspan="2">···</td></tr>
<tr><td>[DMV17]</td><td colspan="4">init</td><td colspan="3">duplex</td><td colspan="3">duplex</td><td>···</td></tr>
</table>

- [BDPV11a]: duplex security reduced to sponge indifferentiability
- [MRV15]: same structure but tighter bound
- [DMV17]: improved bound by re-structuring, but *flag* needed

Generalized Keyed Duplex: Phasing

<table>
<tr><th></th><th>A</th><th>P</th><th>S</th><th>A</th><th>P</th><th>S</th><th>A</th><th>P</th><th>S</th><th>A</th><th>$\cdots$</th></tr>
<tr><td>[BDPV11a]</td><td colspan="3">init</td><td colspan="3">duplex</td><td colspan="3">duplex</td><td colspan="2">$\cdots$</td></tr>
<tr><td>[MRV15]</td><td colspan="3">init</td><td colspan="3">duplex</td><td colspan="3">duplex</td><td colspan="2">$\cdots$</td></tr>
<tr><td>[DMV17]</td><td colspan="4">init</td><td colspan="3">duplex</td><td colspan="3">duplex</td><td>$\cdots$</td></tr>
<tr><td>[DM19]</td><td colspan="2">init</td><td colspan="3">duplex</td><td colspan="3">duplex</td><td colspan="3">$\cdots$</td></tr>
</table>

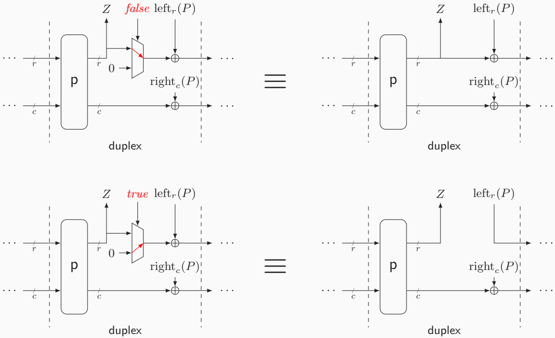
- [BDPV11a]: duplex security reduced to sponge indifferentiability
- [MRV15]: same structure but tighter bound
- [DMV17]: improved bound by re-structuring, but *flag* needed
- [DM19]: security analysis in leaky setting, include upcoming p

Generalized Keyed Duplex: Phasing

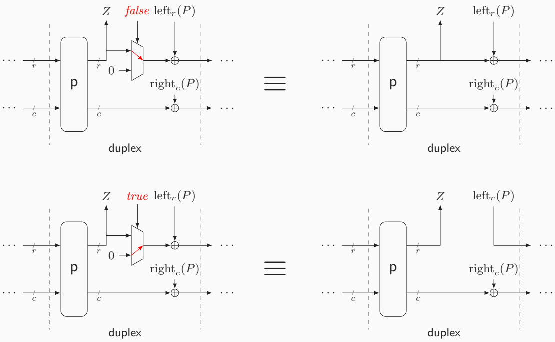
<table>
<tr><th></th><th>A</th><th>P</th><th>S</th><th>A</th><th>P</th><th>S</th><th>A</th><th>P</th><th>S</th><th>A</th><th>⋯</th></tr>
<tr><td>[BDPV11a]</td><td colspan="3">init</td><td colspan="3">duplex</td><td colspan="3">duplex</td><td colspan="2">⋯</td></tr>
<tr><td>[MRV15]</td><td colspan="3">init</td><td colspan="3">duplex</td><td colspan="3">duplex</td><td colspan="2">⋯</td></tr>
<tr><td>[DMV17]</td><td colspan="4">init</td><td colspan="3">duplex</td><td colspan="3">duplex</td><td>⋯</td></tr>
<tr><td>[DM19]</td><td colspan="2">init</td><td colspan="3">duplex</td><td colspan="3">duplex</td><td colspan="3">⋯</td></tr>
<tr><td>now</td><td>init</td><td colspan="3">duplex</td><td colspan="3">duplex</td><td colspan="3">duplex</td><td>⋯</td></tr>
</table>

- [BDPV11a]: duplex security reduced to sponge indifferentiability
- [MRV15]: same structure but tighter bound
- [DMV17]: improved bound by re-structuring, but *flag* needed
- [DM19]: security analysis in leaky setting, include upcoming p
- now: seemingly most useful phasing

Generalized Keyed Duplex: Flag (1)

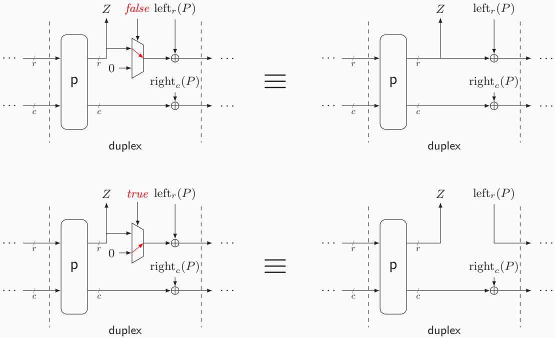


Generalized Keyed Duplex: Flag (1)



- 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 $flag = true$
- Earlier P, p, Z phasing allowed outer part overwriting by default

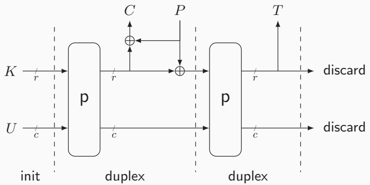
Generalized Keyed Duplex: Flag (2)

- 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

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

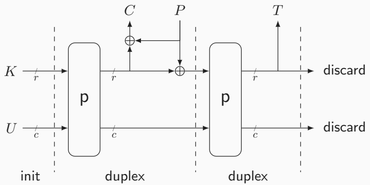
Encryption



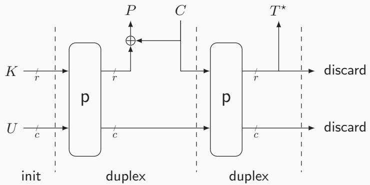
Generalized Keyed Duplex: Flag (2)

- 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



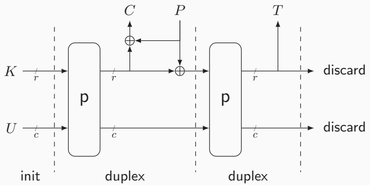
Decryption



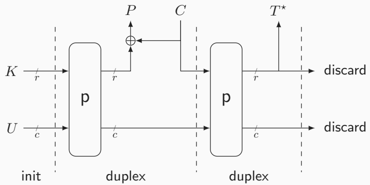
Generalized Keyed Duplex: Flag (2)

- 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



Decryption



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

Understanding Duplex Security

Security Model ([DMV17, DM19], with updated initial rotation and rephasing)

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

Security Model ([DMV17, DM19], with updated initial rotation and rephasing)

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: $(flag, P) \in \{true, false\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$
  $S \leftarrow \mathsf{p}(S)$
  $Z \leftarrow \mathrm{left}_r(S)$
  $S \leftarrow S \oplus [flag] \cdot (Z \| 0^{b-r}) \oplus P$
  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: $\varnothing$
  $path \leftarrow \mathrm{encode}[\delta] \parallel IV$
  return $\varnothing$

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

Security Model ([DMV17, DM19], with updated initial rotation and rephasing)

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, DM19], with updated initial rotation and rephasing)

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: $Z \in \{0,1\}^r$
$S \leftarrow \mathsf{p}(S)$
$Z \leftarrow \mathsf{ro}(\mathit{path}, r)$
$\mathit{path} \leftarrow \mathit{path} \parallel ([\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P)$
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

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, DM19], with updated initial rotation and rephasing)

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
- Bound on adversarial resources is in turn determined by use case!

Security Bounds From [DMV17] and [DM19]

- 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 [DM19]

- 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]:

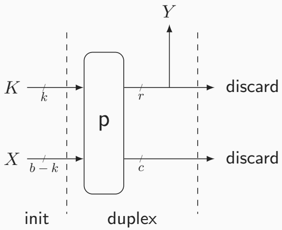
$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) \le \frac{(L+\Omega)N}{2^c} + \frac{2\nu_{r,c}^{2(M-L)}(N+1)}{2^c} + \frac{\binom{L+\Omega+1}{2}}{2^c} + \frac{(M-L-Q)Q}{2^b-Q} + \frac{M(M-L-1)}{2^b} + \frac{Q(M-L-Q)}{2^{\min\{c+k,\max\{b-\alpha,c\}\}}} + \frac{Q_{IV}N}{2^k} + \frac{\binom{\mu}{2}}{2^k}$$

- [DM19] (with one simplification):

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

Intermezzo: Multicollision Coefficient (Skipped)

Use Case 1: Truncated Permutation

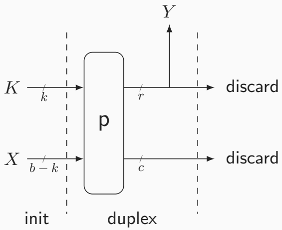


Algorithm Truncated permutation TP[p]

Input: $(K, X) \in \{0,1\}^k \times \{0,1\}^{b-k}$
Output: $Y \in \{0,1\}^r$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  KD.init$(1, X)$
  $Y \leftarrow$ KD.duplex$(\mathit{false}, 0^b)$
  return Y

- PRP-to-PRF conversion: SoP/EDM/EDMD/truncation/STH/...

Truncated Permutation

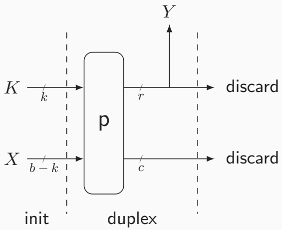


Algorithm Truncated permutation TP[p]

Input: $(K, X) \in \{0,1\}^k \times \{0,1\}^{b-k}$
Output: $Y \in \{0,1\}^r$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  KD.init$(1, X)$
  $Y \leftarrow$ KD.duplex$(false, 0^b)$
  return Y

- PRP-to-PRF conversion: SoP/EDM/EDMD/truncation/STH/...
- Trend towards RP-to-PRF conversion:
  - Sum of externally keyed permutations [CLM19]
  - Permutation-based EDM [DNT21]

Truncated Permutation



Algorithm Truncated permutation TP[p]

Input: $(K, X) \in \{0,1\}^k \times \{0,1\}^{b-k}$
Output: $Y \in \{0,1\}^r$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  KD.init$(1, X)$
  $Y \leftarrow$ KD.duplex$(\mathit{false}, 0^b)$
  return Y

- PRP-to-PRF conversion: SoP/EDM/EDMD/truncation/STH/...
- Trend towards RP-to-PRF conversion:
  - Sum of externally keyed permutations [CLM19]
  - Permutation-based EDM [DNT21]
- Truncation of externally keyed permutation can be described using duplex

Truncated Permutation: Security (1)

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

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

- D can make q construction queries $+$ N primitive queries

Truncated Permutation: Security (1)

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

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

- D can make q construction queries + N primitive queries
- $\mathsf{TP}[\mathsf{p}]_K$ is basically just $\mathsf{TP}[\mathsf{KD}[\mathsf{p}]_K]$

Truncated Permutation: Security (1)

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

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

- D can make q construction queries + N primitive queries
- $\mathsf{TP}[\mathsf{p}]_K$ is basically just $\mathsf{TP}[\mathsf{KD}[\mathsf{p}]_K]$
- Triangle inequality:

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

Truncated Permutation: Security (1)

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

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

- D can make q construction queries + N primitive queries
- $\mathsf{TP}[\mathsf{p}]_K$ is basically just $\mathsf{TP}[\mathsf{KD}[\mathsf{p}]_K]$
- Triangle inequality:

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

Truncated Permutation: Security (1)

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

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

- D can make q construction queries + N primitive queries
- $\mathsf{TP}[\mathsf{p}]_K$ is basically just $\mathsf{TP}[\mathsf{KD}[\mathsf{p}]_K]$
- Triangle inequality:

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

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

Truncated Permutation: Security (1)

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

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

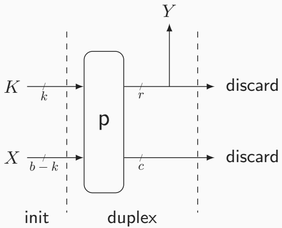
- D can make q construction queries $+$ N primitive queries
- $\mathsf{TP}[\mathsf{p}]_K$ is basically just $\mathsf{TP}[\mathsf{KD}[\mathsf{p}]_K]$
- Triangle inequality:

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

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

- What are the resources of D′?

Truncated Permutation: Security (2)

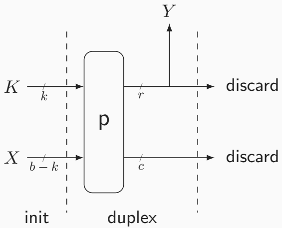


Algorithm Truncated permutation TP[p]

Input: $(K, X) \in \{0,1\}^k \times \{0,1\}^{b-k}$
Output: $Y \in \{0,1\}^r$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  KD.init$(1, X)$
  $Y \leftarrow$ KD.duplex$(\mathit{false}, 0^b)$
  return Y

---

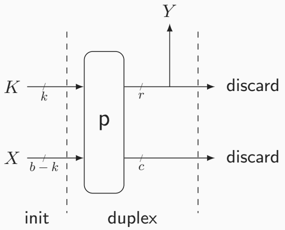
resources of D′	in terms of	resources of D
M: data complexity (calls to construction)		
N: time complexity (calls to primitive)		
Q: number of init calls		
Q_{IV}: max # init calls for single IV		
L: # queries with repeated path		
Ω: # queries with overwriting outer part		



Algorithm Truncated permutation TP[p]

Input: $(K, X) \in \{0,1\}^k \times \{0,1\}^{b-k}$
Output: $Y \in \{0,1\}^r$
Underlying keyed duplex: KD[p]$_{(K)}$
KD.init$(1, X)$
$Y \leftarrow$ KD.duplex$(\mathit{false}, 0^b)$
return Y

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		

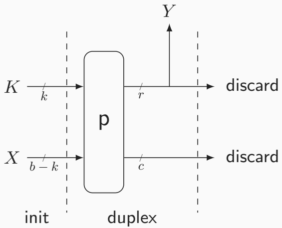


Algorithm Truncated permutation TP[p]

Input: $(K, X) \in \{0,1\}^k \times \{0,1\}^{b-k}$
Output: $Y \in \{0,1\}^r$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  $\mathsf{KD.init}(1, X)$
  $Y \leftarrow \mathsf{KD.duplex}(\mathit{false}, 0^b)$
  return Y

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	q
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		

Truncated Permutation: Security (2)

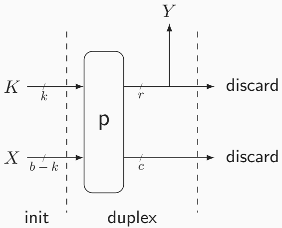


Algorithm Truncated permutation TP[p]

Input: $(K, X) \in \{0,1\}^k \times \{0,1\}^{b-k}$
Output: $Y \in \{0,1\}^r$
Underlying keyed duplex: $\mathrm{KD[p]}_{(K)}$
  KD.init$(1, X)$
  $Y \leftarrow$ KD.duplex$(\mathit{false}, 0^b)$
  return Y

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	q
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		

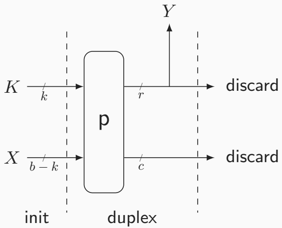
Truncated Permutation: Security (2)



Algorithm Truncated permutation TP[p]

Input: $(K, X) \in \{0,1\}^k \times \{0,1\}^{b-k}$
Output: $Y \in \{0,1\}^r$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  $\mathsf{KD.init}(1, X)$
  $Y \leftarrow \mathsf{KD.duplex}(\mathit{false}, 0^b)$
  return Y

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	q
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		

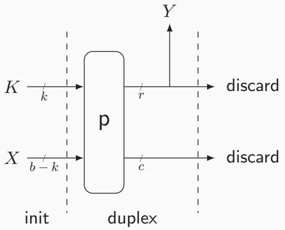


Algorithm Truncated permutation TP[p]

Input: $(K, X) \in \{0,1\}^k \times \{0,1\}^{b-k}$
Output: $Y \in \{0,1\}^r$
Underlying keyed duplex: $\mathsf{KD[p]}_{(K)}$
KD.init$(1, X)$
$Y \leftarrow$ KD.duplex$(\mathit{false}, 0^b)$
return Y

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	q
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

Truncated Permutation: Security (2)



Algorithm Truncated permutation TP[p]

Input: $(K, X) \in \{0,1\}^k \times \{0,1\}^{b-k}$
Output: $Y \in \{0,1\}^r$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  $\mathsf{KD.init}(1, X)$
  $Y \leftarrow \mathsf{KD.duplex}(\mathit{false}, 0^b)$
  return Y

---

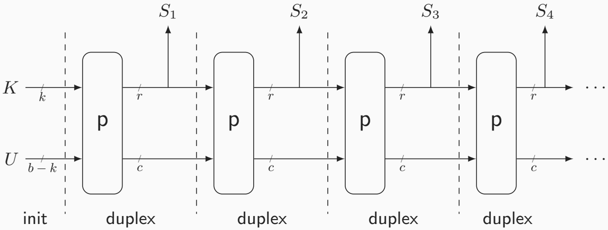
resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	q
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	q
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path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

From [DMV17] (in single-user setting): $\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}') \leq \frac{2\nu_{r,c}^{2q}(N+1)}{2^c} + \frac{2\binom{q}{2}}{2^b} + \frac{N}{2^k}$

Use Case 2: Parallel Keystream Generation (Skipped)

Use Case 3: Sequential Keystream Generation

Sequential Keystream Generation



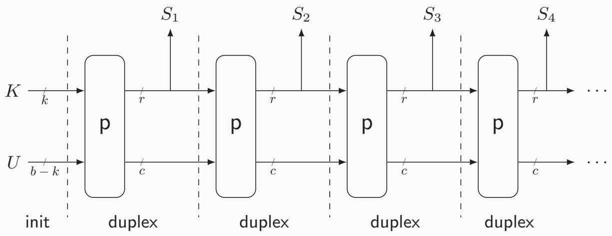
- Input: key K, nonce U
- Output: keystream S of requested length

Algorithm Sequential keystream generation S-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 $\mathrm{left}_\ell(S)$

Sequential Keystream Generation



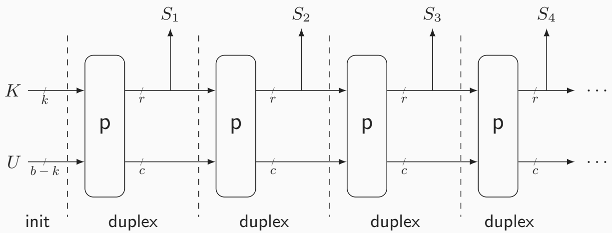
- Input: key K, nonce U
- Output: keystream S of requested length
- PRF security of S-SC[p]:
  - Comparable analysis as for TP[p]

Algorithm Sequential keystream generation S-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 $\mathrm{left}_\ell(S)$

Sequential Keystream Generation



- Input: key K, nonce U
- Output: keystream S of requested length
- PRF security of S-SC[p]:
  - Comparable analysis as for TP[p]
  - Resources of D′ slightly differ

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

Sequential Keystream Generation: Security

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

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

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

Sequential Keystream Generation: Security

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

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

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

Sequential Keystream Generation: Security

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

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

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

Sequential Keystream Generation: Security

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

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

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

Sequential Keystream Generation: Security

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

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

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

Sequential Keystream Generation: Security

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

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

- D can make q construction queries (total σ blocks) $+$ N primitive queries
- Triangle inequality: $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{S\text{-}SC}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$
- 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		
L: # queries with repeated path		
Ω: # queries with overwriting outer part		

Sequential Keystream Generation: Security

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

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

- D can make q construction queries (total σ blocks) $+$ N primitive queries
- Triangle inequality: $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{S\text{-}SC}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$
- 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	$\longrightarrow$	0
Ω: # queries with overwriting outer part	$\longrightarrow$	0

Sequential Keystream Generation: Security

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

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

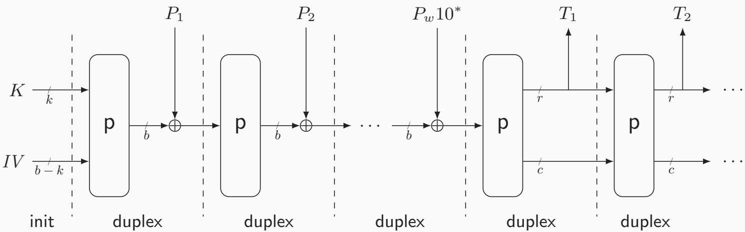
- D can make q construction queries (total σ blocks) $+$ N primitive queries
- Triangle inequality: $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{S\text{-}SC}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm} \ ; \ \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right)$
- What 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	$\longrightarrow$	0
Ω: # queries with overwriting outer part	$\longrightarrow$	0

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

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

Use Case 4: Message Authentication



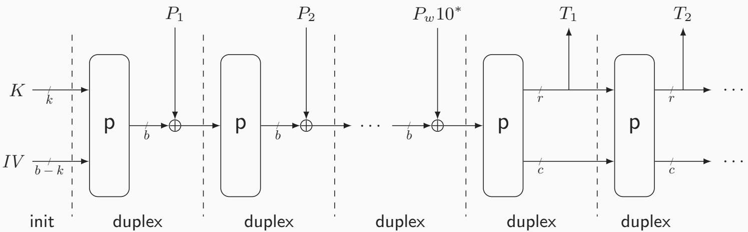
- Input: key K, initial value IV, message P
- Output: tag T

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**
  $T \leftarrow T \parallel$ KD.duplex$(\mathit{false}, 0^b)$
return $\mathrm{left}_t(T)$

Full-State Keyed Sponge [BDPV12]

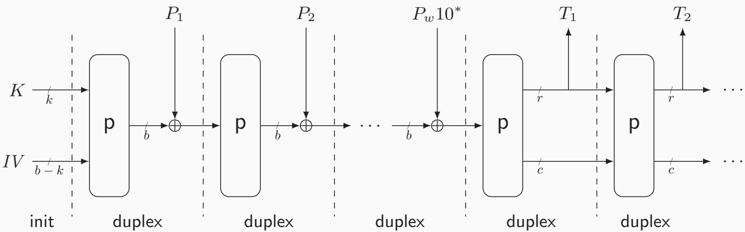


- Input: key K, initial value IV, message P
- Output: tag T
- Analysis of [MRV15] applies

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]



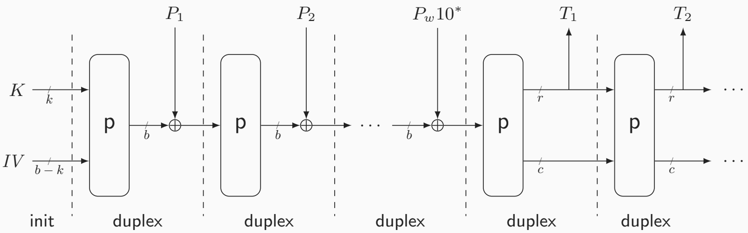
- Input: key K, initial value IV, message P
- Output: tag T
- Analysis of [MRV15] applies
- PRF security of FSKS[p]:
  - Comparable analysis as for S-SC[p]

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**
  $T \leftarrow T \,\|\,$ KD.duplex$(\mathit{false}, 0^b)$
return $\mathrm{left}_t(T)$

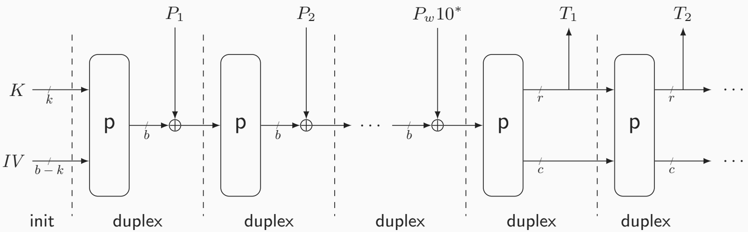
Full-State Keyed Sponge [BDPV12]



- Input: key K, initial value IV, message P
- Output: tag T
- Analysis of [MRV15] applies
- PRF security of FSKS[p]:
  - Comparable analysis as for S-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 $\mathrm{left}_t(T)$



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

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

- 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):

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

- 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):

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

influence of L

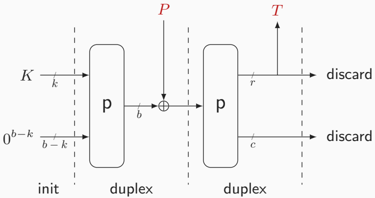
- Repeated paths (i.e., large L) can seriously affect security

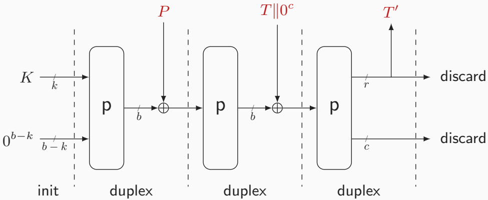
Full-State Keyed Sponge: Adversarial Power in Influencing Outer Part

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

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

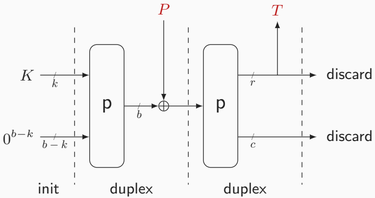
- Repeated paths (i.e., large L) can seriously affect security
- Consider simplified FSKS[p]: no IV, no padding, r-bit tag
- Distinguisher makes two queries: $P \mapsto T$ and $P\|T\|0^c \mapsto T'$

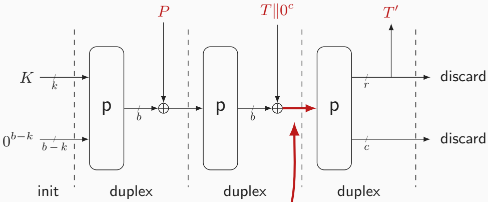




Full-State Keyed Sponge: Adversarial Power in Influencing Outer Part

- 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 $P\|T\|0^c \mapsto T'$

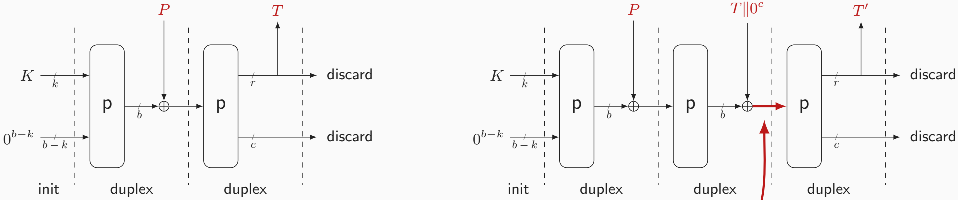




- State of second query before squeezing equals $0^r\|*^c$

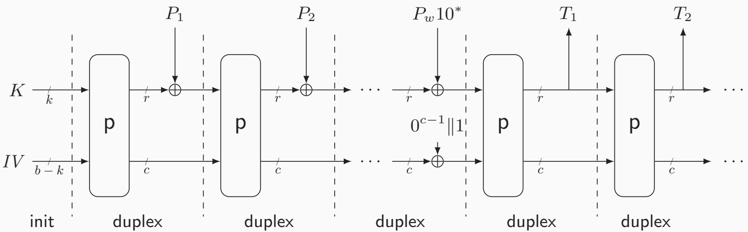
Full-State Keyed Sponge: Adversarial Power in Influencing Outer Part

- 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]

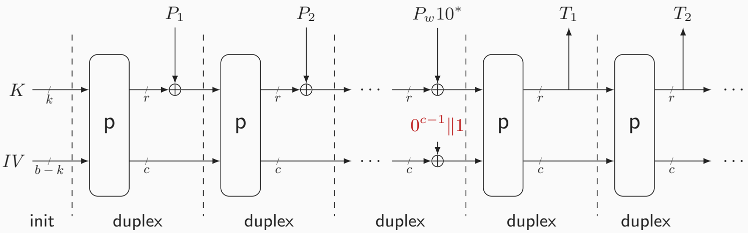


- Input: key K, initial value IV, message P
- Output: tag T

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, ..., 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)
```



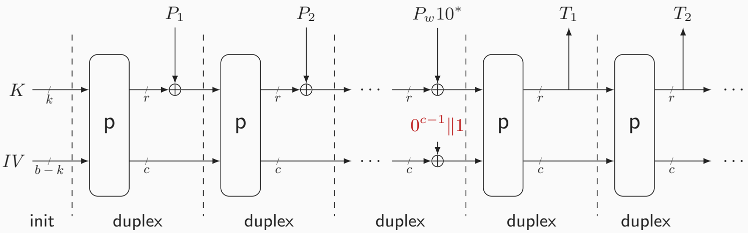
- Input: key K, initial value IV, message P
- Output: tag T
- Domain separation solves problem of repeated paths

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 $\mathrm{left}_t(T)$

Ascon-PRF [DEMS21]



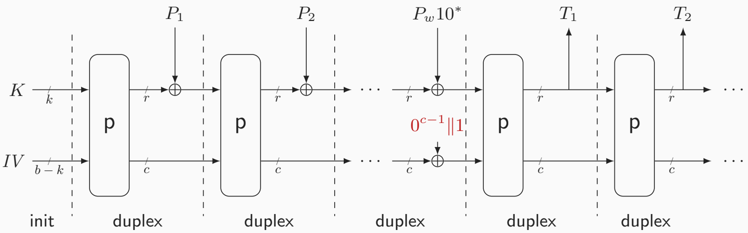
- Input: key K, initial value IV, message P
- Output: tag T
- Domain separation solves problem of repeated paths
  - Repeated paths may still occur...

Algorithm Ascon-PRF[p]

Input: $(K, IV, P) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
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**
  $T \leftarrow T \parallel$ 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
  - 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**
  $T \leftarrow T \parallel$ 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 [DM19]:
  - 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]

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

Ascon-PRF: Implication

Multi-user bound from [DMV17]

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

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

Multi-user bound from [DMV17]

$$
\begin{array}{rccccccccccc}
\mathbf{Adv}^{\mu\text{-prf}}_{\textsf{Ascon-PRF}}(\mathsf{D}) \leq & \frac{2\nu^{2\sigma}_{r,c}(N+1)}{2^c} & + & \frac{(\sigma-q)q}{2^b-q} & + & \frac{2\binom{\sigma}{2}}{2^b} & + & \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} & + & \frac{\mu N}{2^k} & + & \frac{\binom{\mu}{2}}{2^k} \\
& \Big\downarrow \leq & & \Big\downarrow \leq & & \Big\downarrow \leq & & \Big\downarrow \leq & & \Big\downarrow \leq & & \Big\downarrow \leq \\
& \frac{10(N+1)}{2^{192}} & + & \frac{2^{128}}{2^{320}} & + & \frac{2^{128}}{2^{320}} & + & \frac{2^{128}}{2^{320}} & + & \frac{\mu N}{2^{128}} & + & \frac{\binom{\mu}{2}}{2^{128}}
\end{array}
$$

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]

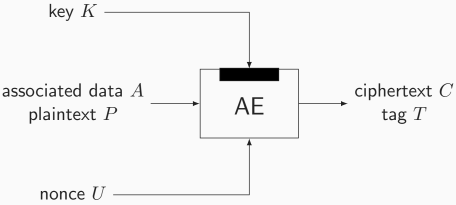
$$
\begin{array}{rcccccccccccc}
\mathbf{Adv}^{\mu\text{-prf}}_{\text{Ascon-PRF}}(\mathsf{D}) \leq & \frac{2\nu_{r,c}^{2\sigma}(N+1)}{2^c} & + & \frac{(\sigma-q)q}{2^b-q} & + & \frac{2\binom{\sigma}{2}}{2^b} & + & \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} & + & \frac{\mu N}{2^k} & + & \frac{\binom{\mu}{2}}{2^k} \\
& \Big\downarrow \leq & & \Big\downarrow \leq & & \Big\downarrow \leq & & \Big\downarrow \leq & & \Big\downarrow \leq & & \Big\downarrow \leq \\
& \frac{10(N+1)}{2^{192}} & + & \frac{2^{128}}{2^{320}} & + & \frac{2^{128}}{2^{320}} & + & \frac{2^{128}}{2^{320}} & + & \frac{\mu N}{2^{128}} & + & \frac{\binom{\mu}{2}}{2^{128}}
\end{array}
$$

Application to Ascon-PRF Parameters

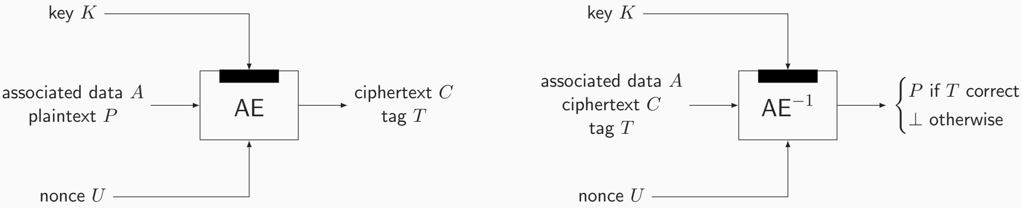
- $(k, b, c, r) = (128, 320, 192, 128)$
- Assume online complexity of $q, \sigma \ll 2^{64}$ (could be taken higher)
- The multicollision term $\nu_{128,192}^{2^{65}}$ is at most 5
- Generic security as long as $N \ll 2^{128}/\mu$

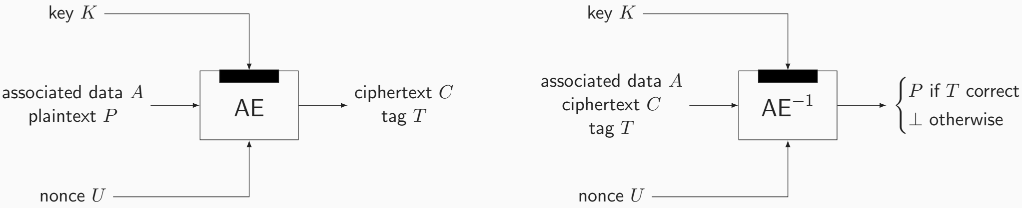
Use Case 5: Authenticated Encryption (Shortened)

Authenticated Encryption



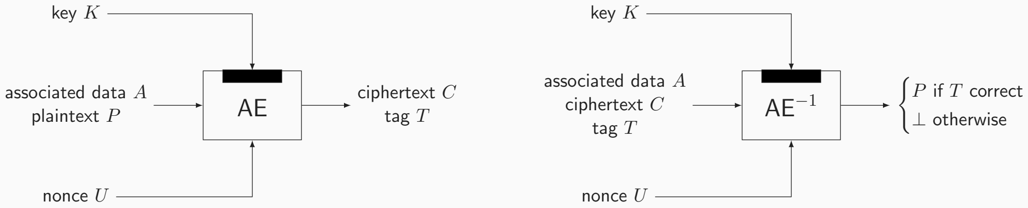
Authenticated Encryption





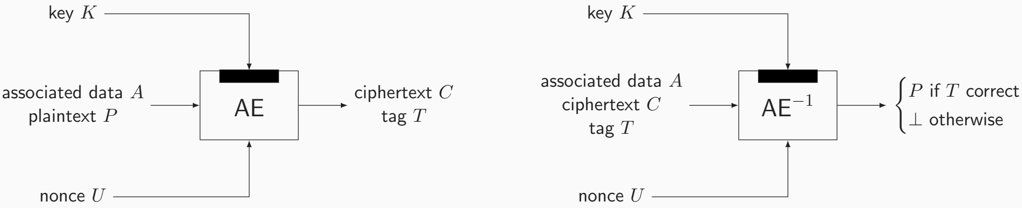
Role of Duplex

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



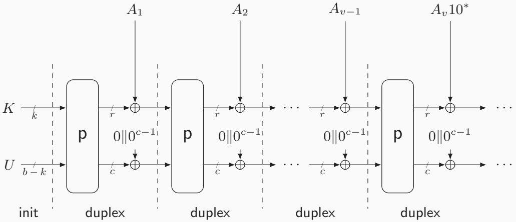
Role of Duplex

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

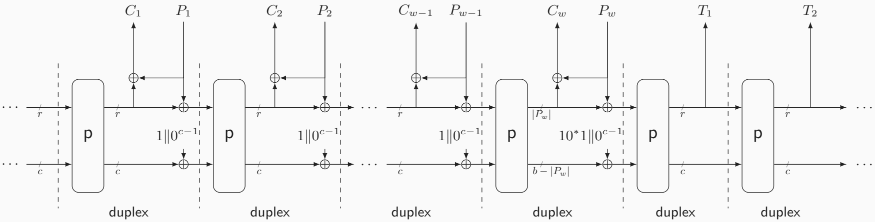


Role of Duplex

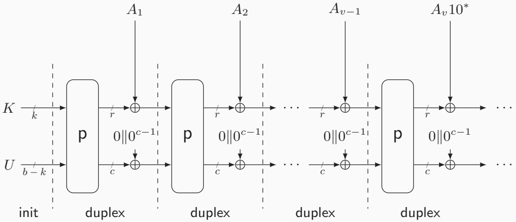
- Blockwise construction allows for processing different types of in-/output
- Usage of flag makes duplex-style encryption decryptable
  (Although the flag is not a necessity for this)

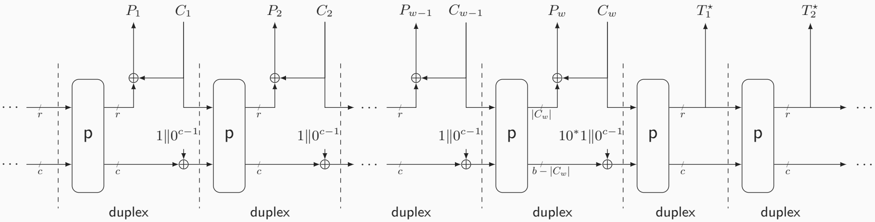


- State initialized using key and nonce
- Cleaned-up and synchronized domain separation
- Spill-over into inner part
- Used in Xoodyak and Gimli (a.o.)



MonkeySpongeWrap: Decryption





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

MonkeySpongeWrap: Security (1)

- Consider distinguisher D against AE security of MonkeySpongeWrap[p]

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MonkeySpongeWrap}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{ENC}[\mathsf{p}]_K, \mathsf{DEC}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{ae}}, \perp, \mathsf{p}^{\pm}\right)$$

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries

MonkeySpongeWrap: Security (1)

- Consider distinguisher D against AE security of MonkeySpongeWrap[p]

$$\mathbf{Adv}^{\text{ae}}_{\mathsf{MonkeySpongeWrap}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{ENC}[\mathsf{p}]_K, \mathsf{DEC}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\text{ae}}, \perp, \mathsf{p}^{\pm}\right)$$

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

MonkeySpongeWrap: Security (1)

- Consider distinguisher D against AE security of MonkeySpongeWrap[p]

$$\mathbf{Adv}^{\text{ae}}_{\text{MonkeySpongeWrap}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{ENC}[\mathsf{p}]_K, \mathsf{DEC}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\text{ae}}, \perp, \mathsf{p}^{\pm}\right)$$

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*
- Triangle inequality derivation slightly more involved than before:

$$\mathbf{Adv}^{\text{ae}}_{\text{MonkeySpongeWrap}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right) + \frac{q_d}{2^t}$$

- Consider distinguisher D against AE security of MonkeySpongeWrap[p]

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MonkeySpongeWrap}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{ENC}[\mathsf{p}]_K, \mathsf{DEC}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\mathrm{ae}}, \perp, \mathsf{p}^{\pm}\right)$$

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*
- Triangle inequality derivation slightly more involved than before:

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MonkeySpongeWrap}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right) + \frac{q_d}{2^t}$$

- What are the resources of D′?

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

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		

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

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

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

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

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

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

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	$\sigma_e + \sigma_d$
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	$q_e + q_d$
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q_d$
Ω: # queries with overwriting outer part	$\longrightarrow$	$\leq \sigma_d - 2q_d$

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	$\sigma_e + \sigma_d$
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	$q_e + q_d$
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q_d$
Ω: # queries with overwriting outer part	$\longrightarrow$	$\leq \sigma_d - 2q_d$

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

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

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	$\sigma_e + \sigma_d$
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	$q_e + q_d$
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q_d$
Ω: # queries with overwriting outer part	$\longrightarrow$	$\leq \sigma_d - 2q_d$

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

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

attack of Gilbert et al. [GBKR23] “operates” here

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	$\sigma_e + \sigma_d$
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	$q_e + q_d$
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q_d$
Ω: # queries with overwriting outer part	$\longrightarrow$	$\leq \sigma_d - 2q_d$

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

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

attack of Gilbert et al. [GBKR23] "operates" here, with $\sigma_d, N \approx 2^{3c/4}$

Conclusion

Conclusion

Generalized Keyed Duplex

- Versatile construction but application not always clear
- Five representative use cases
- Further use cases: PRNG, PBKDF, ...
- Generic security of ISAP v2 follows from duplex and SuKS [DEM+20]
- Caution: all presented results only hold in random permutation model

Conclusion

Generalized Keyed Duplex

- Versatile construction but application not always clear
- Five representative use cases
- Further use cases: PRNG, PBKDF, ...
- Generic security of ISAP v2 follows from duplex and SuKS [DEM$^+$20]
- Caution: all presented results only hold in random permutation model

Much More in Paper

- More detailed explanation on duplex, multicollisions, applications, ...
- Application of bounds of both [DMV17] and [DM19] to use cases
- Multi-user security

Generalized Keyed Duplex

- Versatile construction but application not always clear
- Five representative use cases
- Further use cases: PRNG, PBKDF, ...
- Generic security of ISAP v2 follows from duplex and SuKS [DEM+20]
- Caution: all presented results only hold in random permutation model

Much More in Paper

- More detailed explanation on duplex, multicollisions, applications, ...
- Application of bounds of both [DMV17] and [DM19] to use cases
- Multi-user security

Thank you for your attention!

References i

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.

Yu Long Chen, Eran Lambooij, and Bart Mennink.
How to Build Pseudorandom Functions from Public Random Permutations.
In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 266–293. Springer, 2019.

Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer.
Isap v2.0.
IACR Trans. Symmetric Cryptol., 2020(S1):390–416, 2020.

- 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.

- 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.

Avijit Dutta, Mridul Nandi, and Suprita Talnikar.
Permutation Based EDM: An Inverse Free BBB Secure PRF.
IACR Trans. Symmetric Cryptol., 2021(2):31–70, 2021.

Henri Gilbert, Rachelle Heim Boissier, Louiza Khati, and Yann Rotella.
Generic Attack on Duplex-Based AEAD Modes Using Random Function Statistics.
In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part IV*, volume 14007 of *LNCS*, pages 348–378. Springer, 2023.

Peter Gaži, Krzysztof Pietrzak, and Stefano Tessaro.
The Exact PRF Security of Truncation: Tight Bounds for Keyed Sponges and Truncated CBC.
In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 368–387. Springer, 2015.

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

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.

Supporting Slides

Intermezzo: Multicollision Coefficient

Intermezzo: Multicollision Coefficient $\nu_{r,c}^M$ (1)

Definition

- M balls, 2^r bins
- $\nu_{r,c}^M$ is smallest x such that $\Pr\left(|\text{fullest bin}| > x\right) \leq \frac{x}{2^c}$

Intermezzo: Multicollision Coefficient $\nu_{r,c}^{M}$ (1)

Definition

- M balls, 2^r bins
- $\nu_{r,c}^{M}$ is smallest x such that $\Pr\left(|\text{fullest bin}| > x\right) \leq \frac{x}{2^c}$

Intuition Behind Definition

- We often need upper bound on the maximum multicollision in outer part
- Denote this maximum by ν

Intermezzo: Multicollision Coefficient $\nu_{r,c}^{M}$ (1)

Definition

- M balls, 2^r bins
- $\nu_{r,c}^{M}$ is smallest x such that $\Pr\left(|\text{fullest bin}| > x\right) \leq \frac{x}{2^c}$

Intuition Behind Definition

- We often need upper bound on the maximum multicollision in outer part
- Denote this maximum by ν
- The value ν appears in the security bound in a term of the form $\frac{\nu \cdot N}{2^c}$

Intermezzo: Multicollision Coefficient $\nu_{r,c}^M$ (1)

Definition

- M balls, 2^r bins
- $\nu_{r,c}^M$ is smallest x such that $\Pr\left(|\text{fullest bin}| > x\right) \leq \frac{x}{2^c}$

Intuition Behind Definition

- We often need upper bound on the maximum multicollision in outer part
- Denote this maximum by ν
- The value ν appears in the security bound in a term of the form $\frac{\nu \cdot N}{2^c}$
- We could be unlucky: there could be a $> \nu$-multicollision

Intermezzo: Multicollision Coefficient $\nu_{r,c}^{M}$ (1)

Definition

- M balls, 2^r bins
- $\nu_{r,c}^{M}$ is smallest x such that $\Pr\left(|\text{fullest bin}| > x\right) \leq \frac{x}{2^c}$

Intuition Behind Definition

- We often need upper bound on the maximum multicollision in outer part
- Denote this maximum by ν
- The value ν appears in the security bound in a term of the form $\frac{\nu \cdot N}{2^c}$
- We could be unlucky: there could be a $> \nu$-multicollision
- However, if we take $\nu = \nu_{r,c}^{M}$, this happens with probability at most $\frac{\nu}{2^c}$

Intermezzo: Multicollision Coefficient $\nu_{r,c}^{M}$ (1)

Definition

- M balls, 2^r bins
- $\nu_{r,c}^{M}$ is smallest x such that $\Pr\left(|\text{fullest bin}| > x\right) \leq \frac{x}{2^c}$

Intuition Behind Definition

- We often need upper bound on the maximum multicollision in outer part
- Denote this maximum by ν
- The value ν appears in the security bound in a term of the form $\frac{\nu \cdot N}{2^c}$
- We could be unlucky: there could be a $> \nu$-multicollision
- However, if we take $\nu = \nu_{r,c}^{M}$, this happens with probability at most $\frac{\nu}{2^c}$
- This term is negligible compared to the main probability bound

Intermezzo: Multicollision Coefficient $\nu_{r,c}^{M}$ (2)

Definition

- M balls, 2^r bins
- $\nu_{r,c}^{M}$ is smallest x such that $\Pr\left(|\text{fullest bin}| > x\right) \leq \frac{x}{2^c}$

Intuition of Behavior

Intermezzo: Multicollision Coefficient $\nu_{r,c}^{M}$ (2)

Definition

- M balls, 2^r bins
- $\nu_{r,c}^{M}$ is smallest x such that $\Pr\left(|\text{fullest bin}| > x\right) \leq \frac{x}{2^c}$

Intuition of Behavior

- If $M \ll 2^r$, all bins will likely be “reasonably” empty

Intermezzo: Multicollision Coefficient $\nu_{r,c}^{M}$ (2)

Definition

- M balls, 2^r bins
- $\nu_{r,c}^{M}$ is smallest x such that $\Pr\left(|\text{fullest bin}| > x\right) \leq \frac{x}{2^c}$

Intuition of Behavior

- If $M \ll 2^r$, all bins will likely be “reasonably” empty
- If $M \gg 2^r$, there will likely be a bin with around $linear(b) \cdot \frac{M}{2^r}$ balls

Intermezzo: Multicollision Coefficient $\nu_{r,c}^{M}$ (2)

Definition

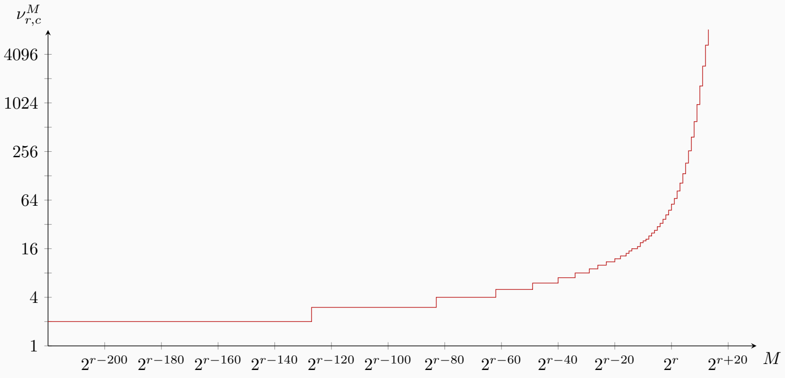
- M balls, 2^r bins
- $\nu_{r,c}^{M}$ is smallest x such that $\Pr\left(|\text{fullest bin}| > x\right) \leq \frac{x}{2^c}$

Intuition of Behavior

- If $M \ll 2^r$, all bins will likely be "reasonably" empty
- If $M \gg 2^r$, there will likely be a bin with around $linear(b) \cdot \frac{M}{2^r}$ balls
- Formula for $\nu_{r,c}^{M}$, and upper bounds in above 2 cases, derived in [DMV17]
- $\nu_{r,c}^{M}$ is (at most) smallest x that satisfies

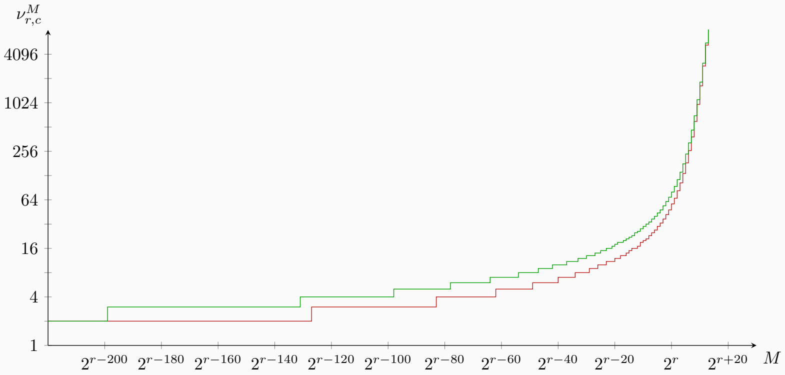
$$\frac{2^b e^{-M/2^r} (M/2^r)^x}{(x - M/2^r) x!} \leq 1$$

Stairway to Heaven for $b = 256$



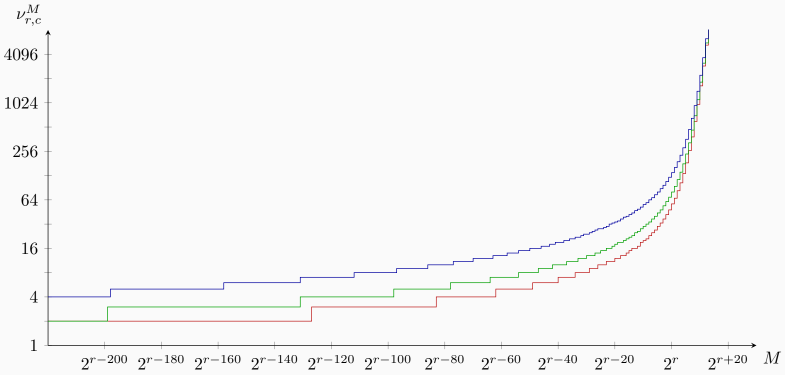
$M/2^r$	$\nu_{r,c}^{M}$
2^{-256}	—
2^{-128}	2
2^{-64}	4
2^{-32}	8
2^{-16}	14
2^{-8}	23
2^{0}	57
2^{8}	601
2^{16}	70205
2^{19}	537313

Stairway to Heaven for $b = 256$, $b = 400$



$M/2^r$	$\nu_{r,c}^{M}$	$\nu_{r,c}^{M}$
2^{-256}	—	2
2^{-128}	2	4
2^{-64}	4	7
2^{-32}	8	12
2^{-16}	14	21
2^{-8}	23	34
2^{0}	57	80
2^{8}	601	707
2^{16}	70205	71484
2^{19}	537313	540887

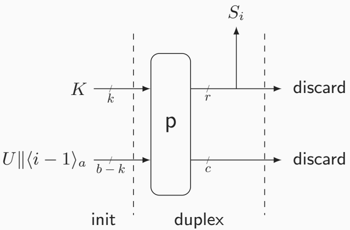
Stairway to Heaven for $b = 256$, $b = 400$, $b = 800$



$M/2^r$	$\nu_{r,c}^M$	$\nu_{r,c}^M$	$\nu_{r,c}^M$
2^{-256}	—	2	4
2^{-128}	2	4	7
2^{-64}	4	7	12
2^{-32}	8	12	23
2^{-16}	14	21	40
2^{-8}	23	34	64
2^{0}	57	80	139
2^{8}	601	707	944
2^{16}	70205	71484	74119
2^{19}	537313	540887	548194

Use Case 2: Parallel Keystream Generation

Parallel Keystream Generation



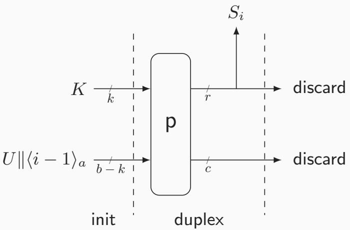
Algorithm Parallel keystream generation P-SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k-a} \times \{0,\ldots,r2^a\}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: $\mathsf{KD[p]}_{(K)}$

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

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

Parallel Keystream Generation



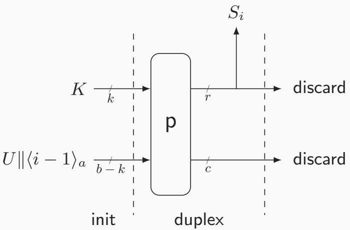
Algorithm Parallel keystream generation P-SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k-a} \times \{0, \dots, r2^a\}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$

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

- Input: key K, nonce U
- Output: keystream S of requested length
- P-SC[p] can be seen as TP[p] in counter mode

Parallel Keystream Generation



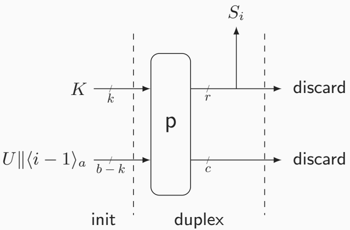
Algorithm Parallel keystream generation P-SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k-a} \times \{0,\ldots,r2^a\}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$

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

- Input: key K, nonce U
- Output: keystream S of requested length
- P-SC[p] can be seen as TP[p] in counter mode
- PRF security of P-SC[p] easily follows:

Parallel Keystream Generation



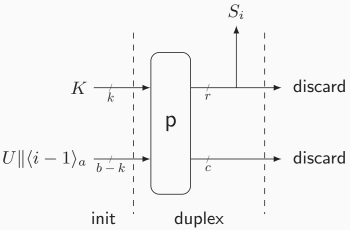
Algorithm Parallel keystream generation P-SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k-a} \times \{0, \ldots, r2^a\}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$

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

- Input: key K, nonce U
- Output: keystream S of requested length
- P-SC[p] can be seen as TP[p] in counter mode
- PRF security of P-SC[p] easily follows:
  - TP[p] behaves like a PRF (up to good bound)

Parallel Keystream Generation



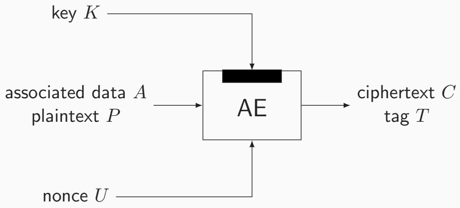
Algorithm Parallel keystream generation P-SC[p]

Input: $(K, U, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k-a} \times \{0, \dots, r2^a\}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
$S \leftarrow \varnothing$
for $i = 1, \dots, \lceil \ell/r \rceil$ **do**
  KD.init$(1, U\|\langle i-1\rangle_a)$
  $S \leftarrow S \,\|\,$ KD.duplex$(\mathit{false}, 0^b)$
return $\mathrm{left}_\ell(S)$

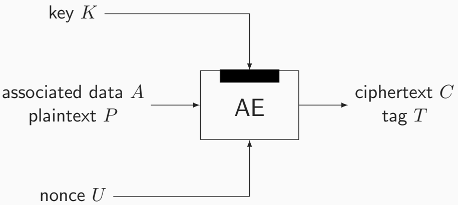
- Input: key K, nonce U
- Output: keystream S of requested length
- P-SC[p] can be seen as TP[p] in counter mode
- PRF security of P-SC[p] easily follows:
  - TP[p] behaves like a PRF (up to good bound)
  - Counter mode with a PRF generates uniform random keystream (provided nonce/counter never repeats)

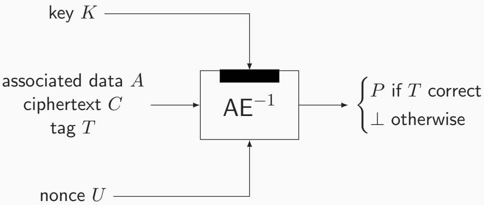
Use Case 5: Authenticated Encryption

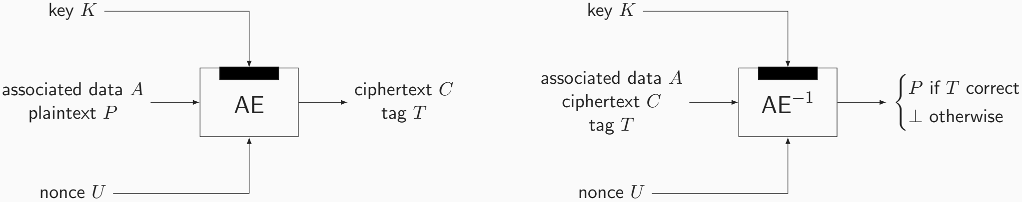
Authenticated Encryption



Authenticated Encryption



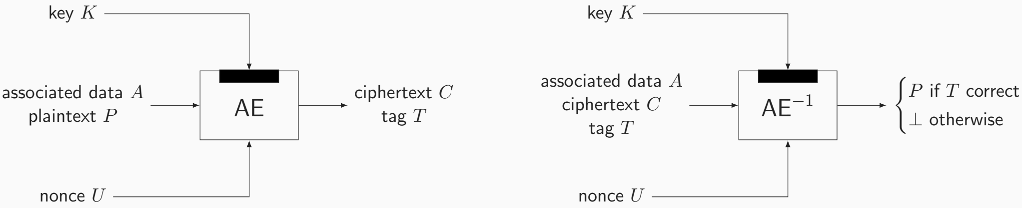




Role of Duplex

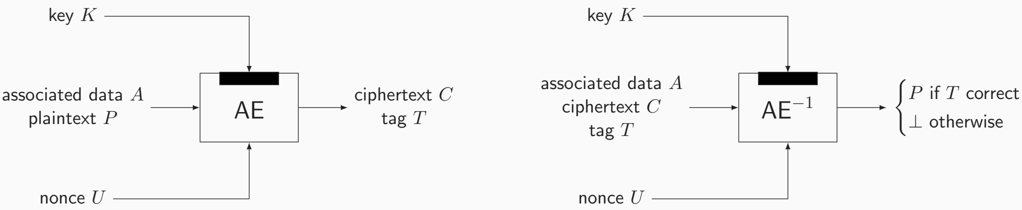
- Blockwise construction allows for processing different types of in-/output

Authenticated Encryption



Role of Duplex

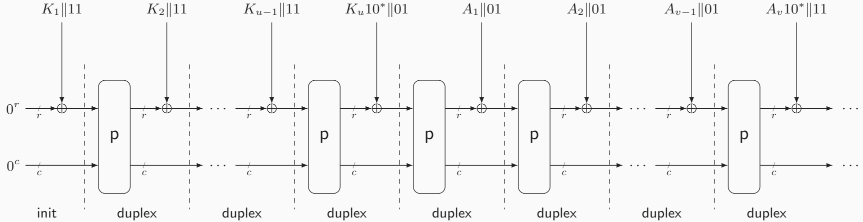
- Blockwise construction allows for processing different types of in-/output
- Usage of flag makes duplex-style encryption decryptable

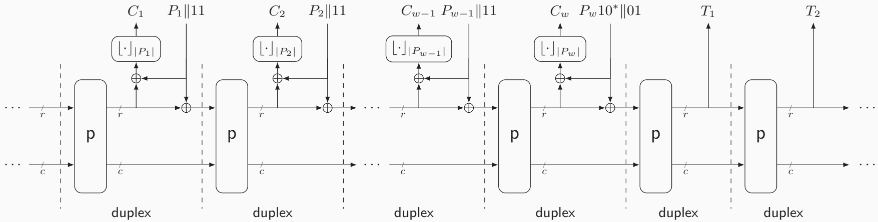


Role of Duplex

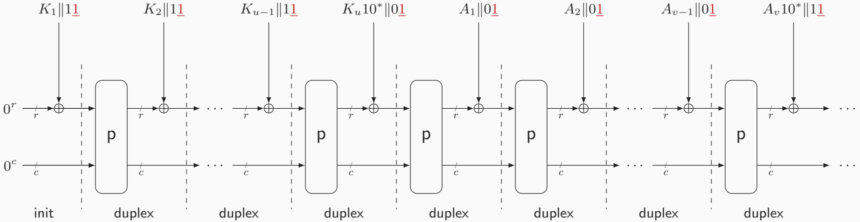
- Blockwise construction allows for processing different types of in-/output
- Usage of flag makes duplex-style encryption decryptable
  (Although the flag is not a necessity for this)

SpongeWrap [BDPV11a]

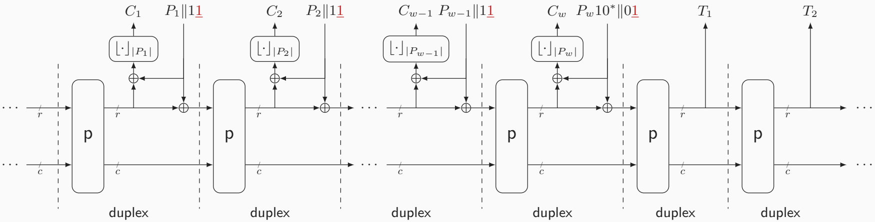




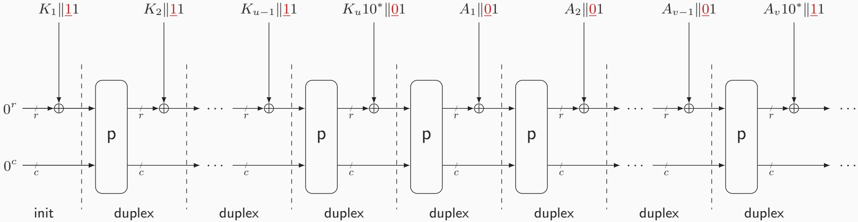
Issues with SpongeWrap



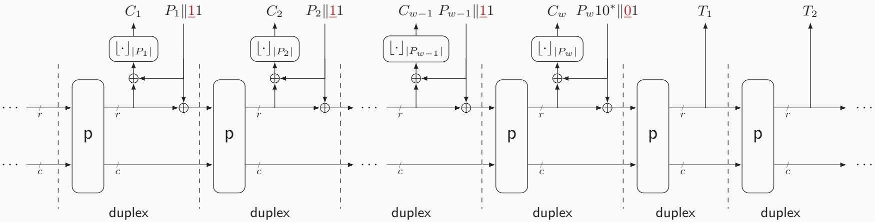
? always 1-padding



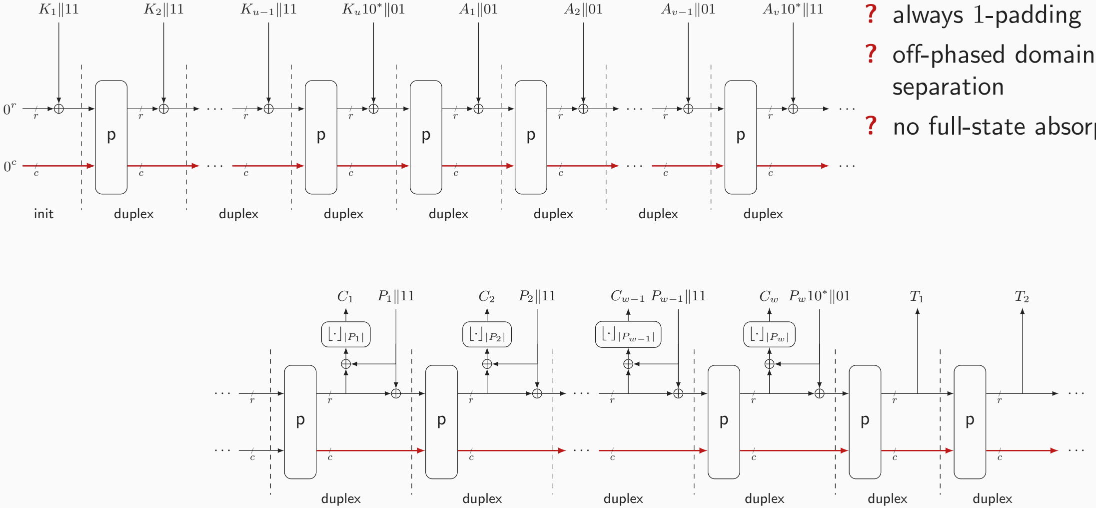
Issues with SpongeWrap



- ? always 1-padding
- ? off-phased domain separation

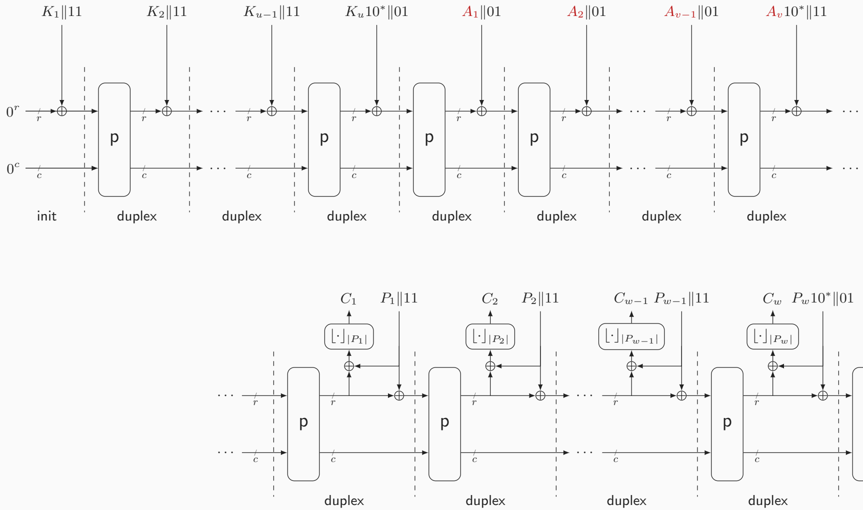


Issues with SpongeWrap



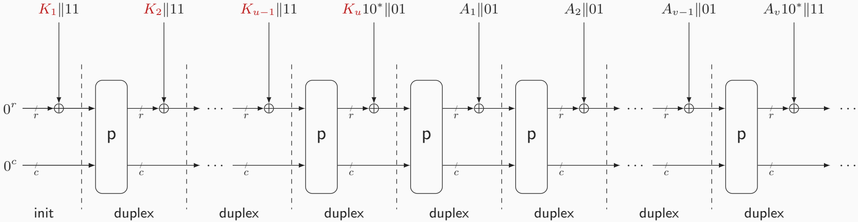
- ? always 1-padding
- ? off-phased domain separation
- ? no full-state absorption

Issues with SpongeWrap

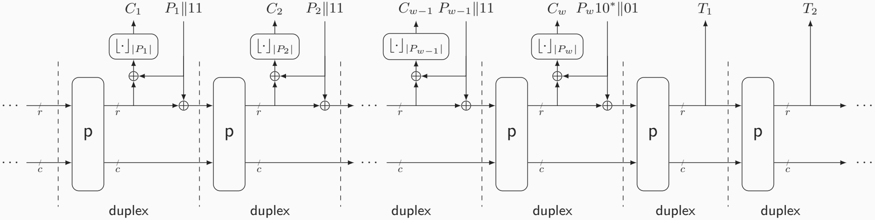


- ? always 1-padding
- ? off-phased domain separation
- ? no full-state absorption
- ? no explicit nonce

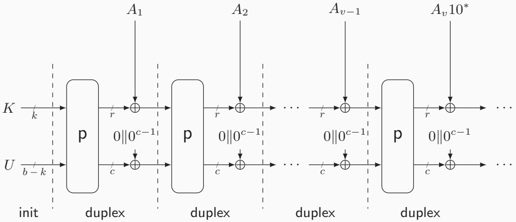
Issues with SpongeWrap



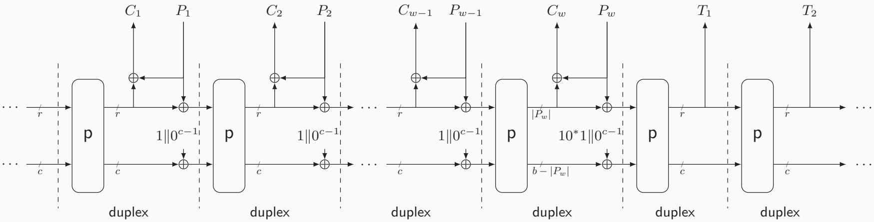
- ? always 1-padding
- ? off-phased domain separation
- ? no full-state absorption
- ? no explicit nonce
- ? blockwise key



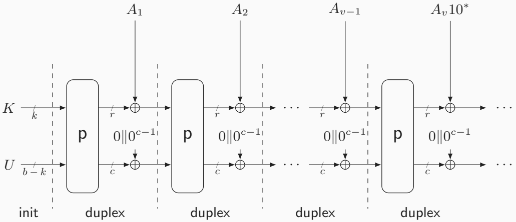
MonkeySpongeWrap: Encryption



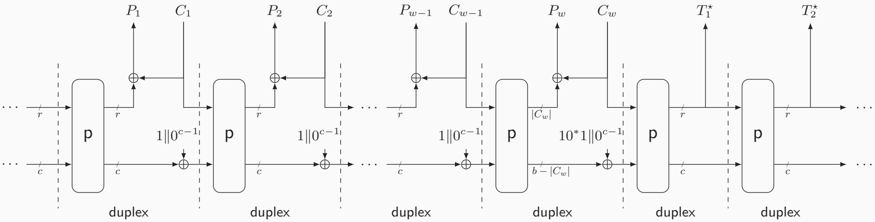
- State initialized using key and nonce
- Cleaned-up and synchronized domain separation
- Spill-over into inner part
- Used in Xoodyak and Gimli (a.o.)



MonkeySpongeWrap: Decryption



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



MonkeySpongeWrap: Algorithm

Algorithm MonkeySpongeWrap[p]: ENC

Input: $(K, U, A, P) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \{0,1\}^* \times \{0,1\}^*$
Output: $(C, T) \in \{0,1\}^{|P|} \times \{0,1\}^t$
Underlying keyed duplex: KD[p]$_{(K)}$

$(A_1, A_2, \ldots, A_v) \leftarrow \mathrm{pad}_r^{10^*}(A)$
$(P_1, P_2, \ldots, P_w) \leftarrow \mathrm{pad}_r^{10^*}(P)$
$C \leftarrow \varnothing$
$T \leftarrow \varnothing$
KD.init$(1, U)$
for $i = 1, \ldots, v$ **do**
  KD.duplex$(\mathit{false}, A_i \| 0 \| 0^{c-1})$ ▷ discard output
for $i = 1, \ldots, w$ **do**
  $C \leftarrow C \,\|\,$ KD.duplex$(\mathit{false}, P_i \| 1 \| 0^{c-1}) \oplus P_i$
for $i = 1, \ldots, \lceil t/r \rceil$ **do**
  $T \leftarrow T \,\|\,$ KD.duplex$(\mathit{false}, 0^b)$
return $(\mathrm{left}_{|P|}(C), \mathrm{left}_t(T))$

Algorithm MonkeySpongeWrap[p]: DEC

Input: $(K, U, A, C, T) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^t$
Output: $P \in \{0,1\}^{|C|}$ or $\perp$
Underlying keyed duplex: KD[p]$_{(K)}$

$(A_1, A_2, \ldots, A_v) \leftarrow \mathrm{pad}_r^{10^*}(A)$
$(C_1, C_2, \ldots, C_w) \leftarrow \mathrm{pad}_r^{10^*}(C)$
$P \leftarrow \varnothing$
$T^\star \leftarrow \varnothing$
KD.init$(1, U)$
for $i = 1, \ldots, v$ **do**
  KD.duplex$(\mathit{false}, A_i \| 0 \| 0^{c-1})$ ▷ discard output
for $i = 1, \ldots, w$ **do**
  $P \leftarrow P \,\|\,$ KD.duplex$(\mathit{true}, C_i \| 1 \| 0^{c-1}) \oplus C_i$
for $i = 1, \ldots, \lceil t/r \rceil$ **do**
  $T^\star \leftarrow T^\star \,\|\,$ KD.duplex$(\mathit{false}, 0^b)$
return $\mathrm{left}_t(T) = \mathrm{left}_t(T^\star)\ ?\ \mathrm{left}_{|C|}(P) : \perp$

MonkeySpongeWrap: Security (1)

- Consider distinguisher D against AE security of MonkeySpongeWrap[p]

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MonkeySpongeWrap}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{ENC}[\mathsf{p}]_K, \mathsf{DEC}[\mathsf{p}]_K, \mathsf{p}^{\pm} \ ; \ \mathsf{R}^{\mathrm{ae}}, \perp, \mathsf{p}^{\pm}\right)$$

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries

MonkeySpongeWrap: Security (1)

- Consider distinguisher D against AE security of MonkeySpongeWrap[p]

$$\mathbf{Adv}^{\text{ae}}_{\text{MonkeySpongeWrap}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{ENC}[\mathsf{p}]_K, \mathsf{DEC}[\mathsf{p}]_K, \mathsf{p}^{\pm}\ ;\ \mathsf{R}^{\text{ae}}, \perp, \mathsf{p}^{\pm}\right)$$

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

MonkeySpongeWrap: Security (1)

- Consider distinguisher D against AE security of MonkeySpongeWrap[p]

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MonkeySpongeWrap}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{ENC}[\mathsf{p}]_K, \mathsf{DEC}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{R}^{\mathrm{ae}}, \perp, \mathsf{p}^{\pm}\right)$$

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*
- Triangle inequality derivation slightly more involved than before:

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MonkeySpongeWrap}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right) + \frac{q_d}{2^t}$$

- Consider distinguisher D against AE security of MonkeySpongeWrap[p]

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MonkeySpongeWrap}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{ENC}[\mathsf{p}]_K, \mathsf{DEC}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{R}^{\mathrm{ae}}, \perp, \mathsf{p}^{\pm}\right)$$

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*
- Triangle inequality derivation slightly more involved than before:

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MonkeySpongeWrap}}(\mathsf{D}) \leq \Delta_{\mathsf{D}'}\left(\mathsf{KD}[\mathsf{p}]_K, \mathsf{p}^{\pm} \;;\; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm}\right) + \frac{q_d}{2^t}$$

- What are the resources of D′?

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, flag always false
- Decryption calls: nonce may repeat, flag may be true

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

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		

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

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

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

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

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

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

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	$\sigma_e + \sigma_d$
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	$q_e + q_d$
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q_d$
Ω: # queries with overwriting outer part	$\longrightarrow$	$\leq \sigma_d - 2q_d$

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	$\sigma_e + \sigma_d$
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	$q_e + q_d$
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q_d$
Ω: # queries with overwriting outer part	$\longrightarrow$	$\leq \sigma_d - 2q_d$

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

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

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	$\sigma_e + \sigma_d$
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	$q_e + q_d$
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q_d$
Ω: # queries with overwriting outer part	$\longrightarrow$	$\leq \sigma_d - 2q_d$

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

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

attack of Gilbert et al. [GBKR23] “operates” here

MonkeySpongeWrap: Security (2)

- D can make: q_e encryption queries (total σ_e blocks),
  q_d decryption queries (total σ_d blocks),
  N primitive queries
- Encryption calls: unique nonce, *flag* always *false*
- Decryption calls: nonce may repeat, *flag* may be *true*

resources of D′	in terms of	resources of D
M: data complexity (calls to construction)	$\longrightarrow$	$\sigma_e + \sigma_d$
N: time complexity (calls to primitive)	$\longrightarrow$	N
Q: number of init calls	$\longrightarrow$	$q_e + q_d$
Q_{IV}: max # init calls for single IV	$\longrightarrow$	1
L: # queries with repeated path	$\longrightarrow$	$\leq q_d$
Ω: # queries with overwriting outer part	$\longrightarrow$	$\leq \sigma_d - 2q_d$

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

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

attack of Gilbert et al. [GBKR23] “operates” here, with $\sigma_d, N \approx 2^{3c/4}$