Maastricht University

The Theory of Permutation-Based Modes

Bart Mennink

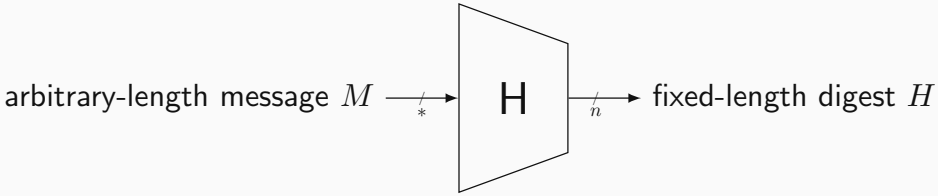
ProTeCS 2026 – Rome

May 9, 2026

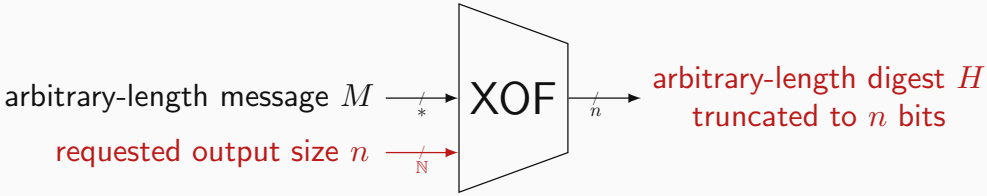
Outline

Introduction

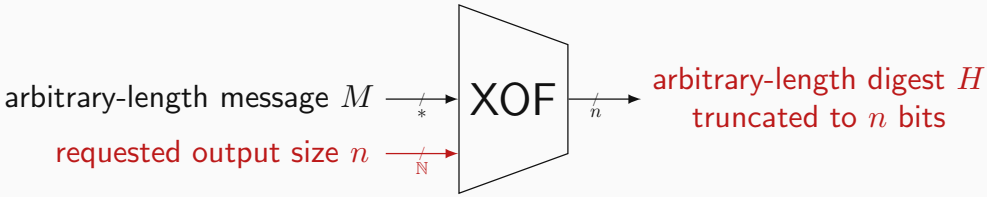
Cryptographic Hash Function



eXtendable Output Function

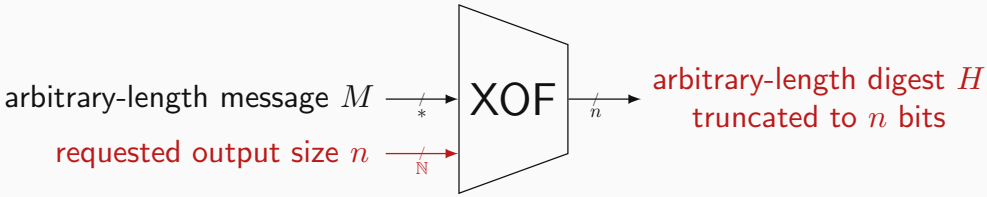


eXtendable Output Function



- Ideally behaves like a random oracle RO [BR93]

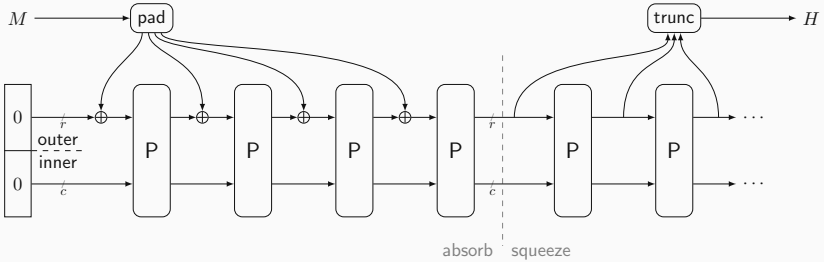
eXtendable Output Function



- Ideally behaves like a random oracle RO [BR93]
  - Different messages $\longrightarrow$ independent random-looking digests
  - Same message but different output sizes $\longrightarrow$ prefixed digests

Sponges [BDPV07]

Sponges [BDPV07]



- P is a b-bit permutation, with $b = r + c$
  - r is the rate
  - c is the capacity (security parameter)

Generic Security of the Sponge

- Assume that P is a random permutation
- Sponge behaves like a random oracle RO up to bound $\mathcal{N}^2/2^c$ [BDPV08]

Generic Security of the Sponge

- Assume that P is a random permutation
- Sponge behaves like a random oracle RO up to bound $\mathcal{N}^2/2^c$ [BDPV08]

- Security of Sponge truncated to n bits against classical attacks [AMP10]:

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

distance from Sponge to RO ($\mathcal{N}$ is # primitive evaluations)

classical attacks against RO ($\mathcal{N}$ is # oracle evaluations)

Generic Security of the Sponge

- Assume that P is a random permutation
- Sponge behaves like a random oracle RO up to bound $\mathcal{N}^2/2^c$ [BDPV08]

- Security of Sponge truncated to n bits against classical attacks [AMP10]:

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

distance from Sponge to RO ($\mathcal{N}$ is # primitive evaluations)

classical attacks against RO ($\mathcal{N}$ is # oracle evaluations)

- Generic security of Sponge truncated to n bits:
  - Collision resistance: $\min\{c/2, n/2\}$
  - First and second preimage resistance: $\min\{c/2, n\}$

Implications for Use Cases

Sponge Behaves Like a Random Oracle

- Message authentication with tag size t: $\mathsf{MAC}(K, M, t) = \mathsf{Sponge}(K \parallel M, t)$
- Keystream generation of length ℓ: $\mathsf{Stream}(K, D, \ell) = \mathsf{Sponge}(K \parallel D, \ell)$

Implications for Use Cases

Sponge Behaves Like a Random Oracle

- Message authentication with tag size t: $\mathsf{MAC}(K, M, t) = \mathsf{Sponge}(K \parallel M, t)$
- Keystream generation of length ℓ: $\mathsf{Stream}(K, D, \ell) = \mathsf{Sponge}(K \parallel D, \ell)$
- One could combine these for authenticated encryption

Implications for Use Cases

Sponge Behaves Like a Random Oracle

- Message authentication with tag size t: $\mathsf{MAC}(K, M, t) = \mathsf{Sponge}(K \parallel M, t)$
- Keystream generation of length ℓ: $\mathsf{Stream}(K, D, \ell) = \mathsf{Sponge}(K \parallel D, \ell)$
- One could combine these for authenticated encryption
- Key commitment somewhat comes for free [DHM+25]

Implications for Use Cases

Sponge Behaves Like a Random Oracle

- Message authentication with tag size t: $\mathsf{MAC}(K, M, t) = \mathsf{Sponge}(K \parallel M, t)$
- Keystream generation of length ℓ: $\mathsf{Stream}(K, D, \ell) = \mathsf{Sponge}(K \parallel D, \ell)$
- One could combine these for authenticated encryption
- Key commitment somewhat comes for free [DHM+25]

However...

Implications for Use Cases

Sponge Behaves Like a Random Oracle

- Message authentication with tag size t: $\mathsf{MAC}(K, M, t) = \mathsf{Sponge}(K \parallel M, t)$
- Keystream generation of length ℓ: $\mathsf{Stream}(K, D, \ell) = \mathsf{Sponge}(K \parallel D, \ell)$
- One could combine these for authenticated encryption
- Key commitment somewhat comes for free [DHM+25]

However...

- ... indifferentiability bound is generic:
  - it does not distinguish between different types of complexities
  - dedicated analysis is often tighter

Implications for Use Cases

Sponge Behaves Like a Random Oracle

- Message authentication with tag size t: $\mathsf{MAC}(K, M, t) = \mathsf{Sponge}(K \parallel M, t)$
- Keystream generation of length ℓ: $\mathsf{Stream}(K, D, \ell) = \mathsf{Sponge}(K \parallel D, \ell)$
- One could combine these for authenticated encryption
- Key commitment somewhat comes for free [DHM+25]

However...

- ... indifferentiability bound is generic:
  - it does not distinguish between different types of complexities
  - dedicated analysis is often tighter
- ... adjustments to the mode sometimes lead to efficiency/security improvements

What Happened Next?

History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

Keyless

Keyed

2010 JH+proof [Wu09, BMN10]

2011 PHOTON [GPP11]

2012 Parazoa [AMP12]

2014 PHOTON's proof [NO14]

2015 *SHA-3 standard [Nat15]*

2022 Tight preimage resistance [LM22]

2023 Restricted indiff. [Lef23]

2024 Double Sponge [LM24b]

2025 Sponge-DM/ Sponge-EDM [SLZ+25]

Hash

History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

Keyless

Keyed

2010 JH+proof [Wu09, BMN10]

2011 PHOTON [GPP11]

2012 Parazoa [AMP12]

2014 PHOTON's proof [NO14]

2015 *SHA-3 standard [Nat15]*

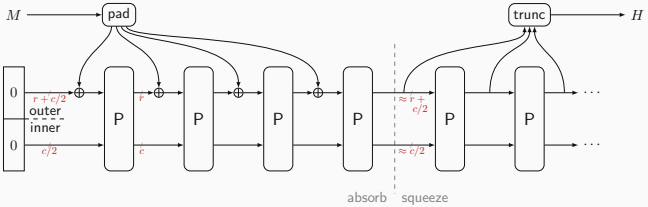
2022 Tight preimage resistance [LM22]

2023 Restricted indiff. [Lef23]

2024 Double Sponge [LM24b]

2025 Sponge-DM/ Sponge-EDM [SLZ+25]

Hash



History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

Keyless

Keyed

- 2010 JH+proof [Wu09, BMN10]
- 2011 PHOTON [GPP11]
- 2012 Parazoa [AMP12]
- 2014 PHOTON's proof [NO14]
- 2015 *SHA-3 standard [Nat15]*
- 2022 Tight preimage resistance [LM22]
- 2023 Restricted indiff. [Lef23]
- 2024 Double Sponge [LM24b]
- 2025 Sponge-DM/ Sponge-EDM [SLZ+25]

Hash

FIPS PUB 202

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions

CATEGORY: COMPUTER SECURITY SUBCATEGORY: CRYPTOGRAPHY

Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8900

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.FIPS.202

August 2015

History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

Keyless

Keyed

2010 JH+proof [Wu09, BMN10]

2011 PHOTON [GPP11]

2012 Parazoa [AMP12]

2014 PHOTON's proof [NO14]

2015 *SHA-3 standard [Nat15]*

2022 Tight preimage resistance [LM22]

2023 Restricted indiff. [Lef23]

2024 Double Sponge [LM24b]

2025 Sponge-DM/ Sponge-EDM [SLZ+25]

Hash

History of Sponge Modes – Not Exhausted (But I Did Get Exhausted)

Keyless

Keyed

2010 JH+proof [Wu09, BMN10]

2011 PHOTON [GPP11]

2012 Parazoa [AMP12]

2014 PHOTON's proof [NO14]

2015 *SHA-3 standard [Nat15]*

2022 Tight preimage resistance [LM22]

2023 SAFE [AKMQ23]

2023 SAFE's proof [KMM23]

2023 Restricted indiff. [Lef23]

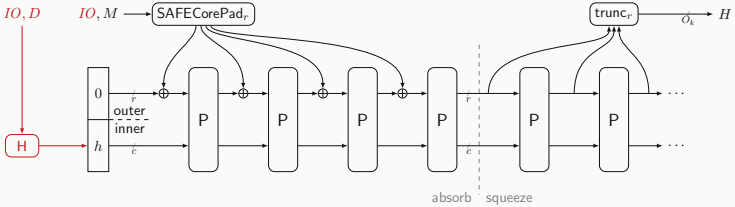
2024 Double Sponge [LM24b]

2025 Padding-free hashing [LMM25]

2025 Sponge-DM/ Sponge-EDM [SLZ+25]

AO-Specific

Hash



History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

Keyless

Keyed

2010 • JH+proof [Wu09, BMN10]

2011 • PHOTON [GPP11]

2012 • Parazoa [AMP12]

2014 • PHOTON's proof [NO14]

2014 • Sufficient conditions [BDPV14]

2015 • *SHA-3 standard [Nat15]*

2018 • **SoK**: sound tree hashing [DMV18]

2020 • Corrigendum [GDM20]

2022 • Tight preimage resistance [LM22]

2023 • SAFE [AKMQ23]

2023 • SAFE's proof [KMM23]

2023 • Restricted indiff. [Lef23]

2024 • Double Sponge [LM24b]

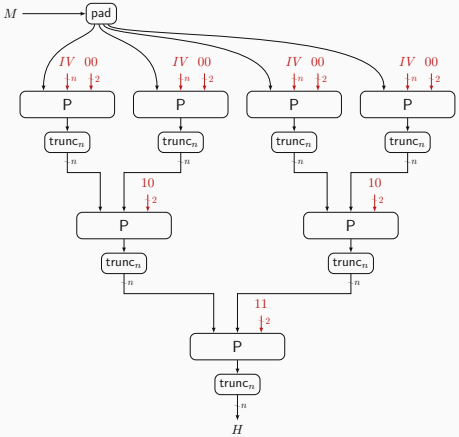
2025 • Padding-free hashing [LMM25]

2025 • Sponge-DM/ Sponge-EDM [SLZ+25]

AO-Specific

Hash

Tree Hash



History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

Keyless

Keyed

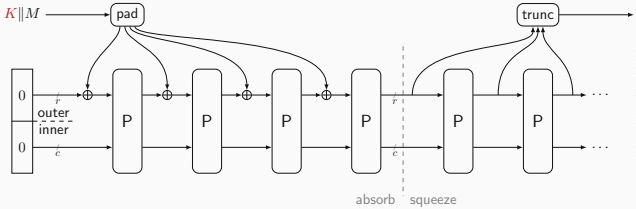
- 2011 Outer-Keyed Sponge [BDPV11b]
- 2012 Inner-Keyed Sponge [CDH+12]
- 2015 OKS/IKS analysis [ADMV15]
- 2015 Full-Keyed Sponge [MRV15]
- 2016 impr. OKS/IKS analysis [NY16]
- 2017 *Farfalle [BDH+17]*
- 2018 Key prediction [Men18]
- 2022 Asakey [DMP22]
- 2024 AsconPRF [DEMS24]
- 2025 MacaKey [LM25a]

Keyed Sponge

History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

Keyless

Keyed



2011 Outer-Keyed Sponge [BDPV11b]

2012 Inner-Keyed Sponge [CDH+12]

2015 OKS/IKS analysis [ADMV15]

2015 Full-Keyed Sponge [MRV15]

2016 impr. OKS/IKS analysis [NY16]

2017 *Farfalle [BDH+17]*

2018 Key prediction [Men18]

2022 Asakey [DMP22]

2024 AsconPRF [DEMS24]

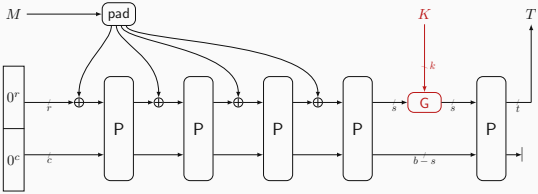
2025 MacaKey [LM25a]

Keyed Sponge

History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

Keyless

Keyed



2011 Outer-Keyed Sponge [BDPV11b]

2012 Inner-Keyed Sponge [CDH+12]

2015 OKS/IKS analysis [ADMV15]

2015 Full-Keyed Sponge [MRV15]

2016 impr. OKS/IKS analysis [NY16]

2017 *Farfalle [BDH+17]*

2018 Key prediction [Men18]

2019 SuKS [DM19b]

2020 Analysis of SuKS [DM20]

2022 Asakey [DMP22]

2024 Leakage resilience of SuKS [BM24]

2024 AsconPRF [DEMS24]

2025 MacaKey [LM25a]

Suffix-Specific

Keyed Sponge

History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

Keyless

Keyed

Keyed Sponge

- 2011 Outer-Keyed Sponge [BDPV11b]
- 2012 Inner-Keyed Sponge [CDH+12]
- 2015 OKS/IKS analysis [ADMV15]
- 2015 Full-Keyed Sponge [MRV15]
- 2016 impr. OKS/IKS analysis [NY16]
- 2017 *Farfalle [BDH+17]*
- 2018 Key prediction [Men18]
- 2022 Asakey [DMP22]
- 2024 AsconPRF [DEMS24]
- 2025 MacaKey [LM25a]

Keyed Duplex

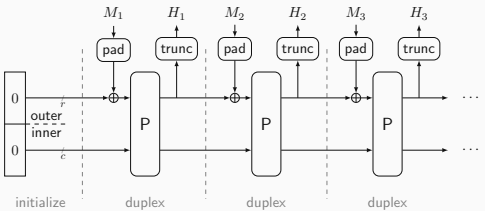
- 2011 Duplex [BDPV11a]
- 2015 Full-Keyed Duplex [MRV15]
- 2017 Modern Duplex [DMV17]
- 2019 Leakage resilience [DM19a]
- 2023 **SoK**: Duplex guide [Men23]
- 2024 Initialization of Duplex [DM24]

Suffix-Specific

- 2019 SuKS [DM19b]
- 2020 Analysis of SuKS [DM20]
- 2024 Leakage resilience of SuKS [BM24]

History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

Keyless



Keyed

- 2011 Outer-Keyed Sponge [BDPV11b]
- 2012 Inner-Keyed Sponge [CDH+12]
- 2015 OKS/IKS analysis [ADMV15]
- 2015 Full-Keyed Sponge [MRV15]
- 2016 impr. OKS/IKS analysis [NY16]
- 2017 *Farfalle [BDH+17]*
- 2018 Key prediction [Men18]
- 2022 Asakey [DMP22]
- 2024 AsconPRF [DEMS24]
- 2025 MacaKey [LM25a]

Keyed Sponge

- 2011 Duplex [BDPV11a]
- 2015 Full-Keyed Duplex [MRV15]
- 2017 Modern Duplex [DMV17]
- 2019 Leakage resilience [DM19a]
- 2023 **SoK**: Duplex guide [Men23]
- 2024 Initialization of Duplex [DM24]

Keyed Duplex

- 2019 SuKS [DM19b]
- 2020 Analysis of SuKS [DM20]
- 2024 Leakage resilience of SuKS [BM24]

Suffix-Specific

History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

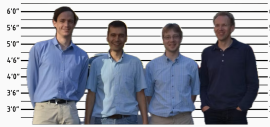
Keyed Duplex

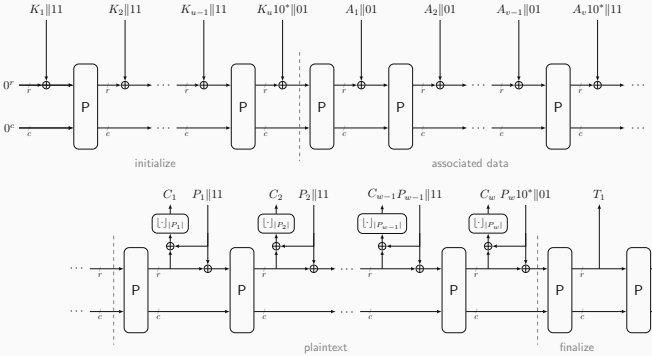
- 2011 Duplex [BDPV11a]
- 2015 Full-Keyed Duplex [MRV15]
- 2017 Modern Duplex [DMV17]
- 2019 Leakage resilience [DM19a]
- 2023 **SoK**: Duplex guide [Men23]
- 2024 Initialization of Duplex [DM24]

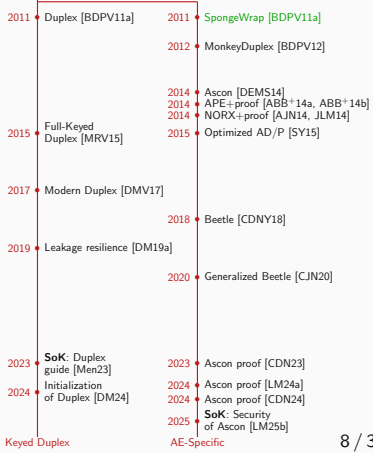
AE-Specific

- 2011 SpongeWrap [BDPV11a]
- 2012 MonkeyDuplex [BDPV12]
- 2014 Ascon [DEMS14]
- 2014 APE+proof [ABB+14a, ABB+14b]
- 2014 NORX+proof [AJN14, JLM14]
- 2015 Optimized AD/P [SY15]
- 2018 Beetle [CDNY18]
- 2020 Generalized Beetle [CJN20]
- 2023 Ascon proof [CDN23]
- 2024 Ascon proof [LM24a]
- 2024 Ascon proof [CDN24]
- 2025 **SoK**: Security of Ascon [LM25b]

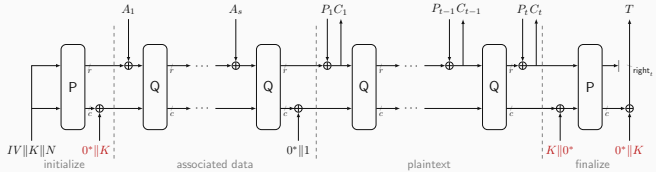
History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)





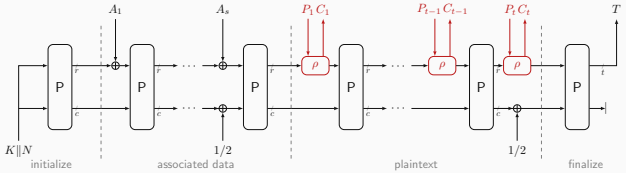


History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)



Keyed Duplex		AE-Specific	
2011	Duplex [BDPV11a]	2011	SpongeWrap [BDPV11a]
		2012	MonkeyDuplex [BDPV12]
		2014	Ascon [DEMS14]
		2014	APE+proof [ABB+14a, ABB+14b]
		2014	NORX+proof [AJN14, JLM14]
2015	Full-Keyed Duplex [MRV15]	2015	Optimized AD/P [SY15]
2017	Modern Duplex [DMV17]		
		2018	Beetle [CDNY18]
2019	Leakage resilience [DM19a]		
		2020	Generalized Beetle [CJN20]
2023	**SoK**: Duplex guide [Men23]	2023	Ascon proof [CDN23]
2024	Initialization of Duplex [DM24]	2024	Ascon proof [LM24a]
		2024	Ascon proof [CDN24]
		2025	**SoK**: Security of Ascon [LM25b]

History of Sponge Modes – Not Exhausted (But I Did Get Exhausted)



Keyed Duplex		AE-Specific	
2011	Duplex [BDPV11a]	2011	SpongeWrap [BDPV11a]
		2012	MonkeyDuplex [BDPV12]
		2014	Ascon [DEMS14]
		2014	APE+proof [ABB+14a, ABB+14b]
		2014	NORX+proof [AJN14, JLM14]
2015	Full-Keyed Duplex [MRV15]	2015	Optimized AD/P [SY15]
2017	Modern Duplex [DMV17]		
		2018	Beetle [CDNY18]
2019	Leakage resilience [DM19a]		
		2020	Generalized Beetle [CJN20]
2023	**SoK**: Duplex guide [Men23]	2023	Ascon proof [CDN23]
2024	Initialization of Duplex [DM24]	2024	Ascon proof [LM24a]
		2024	Ascon proof [CDN24]
		2025	**SoK**: Security of Ascon [LM25b]

History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

Keyless

Keyed

2010 Sponge PRNG [BDPV10]

2016 Robust PRNG [GT16]

2016 Reverie [Hut16]

PRNGs

2011 Duplex [BDPV11a]

2015 Full-Keyed Duplex [MRV15]

2017 Modern Duplex [DMV17]

2019 Leakage resilience [DM19a]

2023 **SoK**: Duplex guide [Men23]

2024 Initialization of Duplex [DM24]

Keyed Duplex

2011 SpongeWrap [BDPV11a]

2012 MonkeyDuplex [BDPV12]

2014 Ascon [DEMS14]

2014 APE+proof [ABB+14a, ABB+14b]

2014 NORX+proof [AJN14, JLM14]

2015 Optimized AD/P [SY15]

2018 Beetle [CDNY18]

2020 Generalized Beetle [CJN20]

2023 Ascon proof [CDN23]

2024 Ascon proof [LM24a]

2024 Ascon proof [CDN24]

2025 **SoK**: Security of Ascon [LM25b]

AE-Specific

History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

Keyless

Keyed

2010 JH+proof [Wu09, BMN10]

2011 PHOTON [GPP11]

2012 Parazoa [AMP12]

2014 PHOTON's proof [NO14]

2015 *SHA-3 standard [Nat15]*

2014 Sufficient conditions [BDPV14]

2018 **SoK**: sound tree hashing [DMV18]

2020 Corrigendum [GDM20]

2022 Tight preimage resistance [LM22]

2023 SAFE [AKMQ23]

2023 SAFE's proof [KMM23]

2025 Padding-free hashing [LMM25]

AO-Specific

Hash

Tree Hash

2010 Sponge PRNG [BDPV10]

2016 Robust PRNG [GT16]

2016 Reverie [Hut16]

PRNGs

2019 SuKS [DM19b]

2020 Analysis of SuKS [DM20]

2023 Restricted indiff. [Lef23]

2024 Double Sponge [LM24b]

2025 Sponge-DM/ Sponge-EDM [SLZ+25]

2024 Leakage resilience of SuKS [BM24]

Suffix-Specific

2011 Outer-Keyed Sponge [BDPV11b]

2012 Inner-Keyed Sponge [CDH+12]

2015 OKS/IKS analysis [ADMV15]

2015 Full-Keyed Sponge [MRV15]

2016 impr. OKS/IKS analysis [NY16]

2017 *Farfalle [BDH+17]*

2018 Key prediction [Men18]

2022 Asakey [DMP22]

2024 AsconPRF [DEMS24]

2025 MacaKey [LM25a]

Keyed Sponge

2011 Duplex [BDPV11a]

Full-Keyed Duplex

2015 Full-Keyed Duplex [MRV15]

2017 Modern Duplex [DMV17]

2019 Leakage resilience [DM19a]

2023 **SoK**: Duplex guide [Men23]

2024 Initialization of Duplex [DM24]

Keyed Duplex

2011 SpongeWrap [BDPV11a]

2012 MonkeyDuplex [BDPV12]

2014 Ascon [DEMS14]

2014 APE+proof [ABB+14a, ABB+14b]

2014 NORX+proof [AJN14, JLM14]

2015 Optimized AD/P [SY15]

2018 Beetle [CDNY18]

2020 Generalized Beetle [CJN20]

2023 Ascon proof [CDN23]

2024 Ascon proof [LM24a]

2024 Ascon proof [CDN24]

2025 **SoK**: Security of Ascon [LM25b]

AE-Specific

History of Sponge Modes – Not Exhaustive (But I Did Get Exhausted)

Keyless

Keyed

2010 JH+proof [Wu09, BMN10]

2011 PHOTON [GPP11]

2012 Parazoa [AMP12]

2014 PHOTON's proof [NO14]

2015 *SHA-3 standard [Nat15]*

2014 Sufficient conditions [BDPV14]

2018 **SoK**: sound tree hashing [DMV18]

2020 Corrigendum [GDM20]

2022 Tight preimage resistance [LM22]

2023 SAFE [AKMQ23]

2023 SAFE's proof [KMM23]

2025 Padding-free hashing [LMM25]

AO-Specific

Hash

Tree Hash

2010 Sponge PRNG [BDPV10]

2016 Robust PRNG [GT16]

2016 Reverie [Hut16]

PRNGs

2019 SuKS [DM19b]

2020 Analysis of SuKS [DM20]

2023 Restricted indiff. [Lef23]

2024 Double Sponge [LM24b]

2024 Leakage resilience of SuKS [BM24]

2025 Sponge-DM/ Sponge-EDM [SLZ+25]

Suffix-Specific

2011 Outer-Keyed Sponge [BDPV11b]

2012 Inner-Keyed Sponge [CDH+12]

2015 OKS/IKS analysis [ADMV15]

2015 Full-Keyed Sponge [MRV15]

2016 impr. OKS/IKS analysis [NY16]

2017 *Farfalle [BDH+17]*

2018 Key prediction [Men18]

2022 Asakey [DMP22]

2024 AsconPRF [DEMS24]

2025 MacaKey [LM25a]

Keyed Sponge

2011 Duplex [BDPV11a]

Full-Keyed Duplex

2015 Full-Keyed Duplex [MRV15]

2017 Modern Duplex [DMV17]

2019 Leakage resilience [DM19a]

2023 **SoK**: Duplex guide [Men23]

2024 Initialization of Duplex [DM24]

Keyed Duplex

2011 SpongeWrap [BDPV11a]

2012 MonkeyDuplex [BDPV12]

2014 Ascon [DEMS14]

2014 APE+proof [ABB+14a, ABB+14b]

2014 NORX+proof [AJN14, JLM14]

2015 Optimized AD/P [SY15]

2018 Beetle [CDNY18]

2020 Generalized Beetle [CJN20]

2023 Ascon proof [CDN23]

2024 Ascon proof [LM24a]

2024 Ascon proof [CDN24]

2025 **SoK**: Security of Ascon [LM25b]

AE-Specific

Hunt For Preimage Resistance

Generic Security of the Sponge (Cont.)

- Assume that P is a random permutation
- Sponge behaves like a random oracle RO up to bound $\mathcal{N}^2/2^c$ [BDPV08]

- Security of Sponge truncated to n bits against classical attacks [AMP10]:

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

distance from Sponge to RO ($\mathcal{N}$ is # primitive evaluations)

classical attacks against RO ($\mathcal{N}$ is # oracle evaluations)

Generic Security of the Sponge (Cont.)

- Assume that P is a random permutation
- Sponge behaves like a random oracle RO up to bound $\mathcal{N}^2/2^c$ [BDPV08]

- Security of Sponge truncated to n bits against classical attacks [AMP10]:

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

distance from Sponge to RO ($\mathcal{N}$ is # primitive evaluations)

classical attacks against RO ($\mathcal{N}$ is # oracle evaluations)

- Attacks already described in [BDPV07]

Generic Security of the Sponge (Cont.)

- Assume that P is a random permutation
- Sponge behaves like a random oracle RO up to bound $\mathcal{N}^2/2^c$ [BDPV08]

- Security of Sponge truncated to n bits against classical attacks [AMP10]:

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

distance from Sponge to RO ($\mathcal{N}$ is # primitive evaluations)

classical attacks against RO ($\mathcal{N}$ is # oracle evaluations)

- Attacks already described in [BDPV07]

Generic Security of the Sponge (Cont.)

- Assume that P is a random permutation
- Sponge behaves like a random oracle RO up to bound $\mathcal{N}^2/2^c$ [BDPV08]

- Security of Sponge truncated to n bits against classical attacks [AMP10]:

Collision resistance: $\mathcal{N}^2/2^c + \mathcal{N}^2/2^{n+1}$ ← attack in $\min\{2^{c/2}, 2^{n/2}\}$

Second preimage resistance: $\mathcal{N}^2/2^c + \mathcal{N}/2^n$ ← attack in $\min\{2^{c/2}, 2^n\}$

~~Preimage resistance: $\mathcal{N}^2/2^c + \mathcal{N}/2^n$ ← attack in $\min\{2^{n-r} + 2^{c/2}, 2^n\}$~~

distance from Sponge to RO ($\mathcal{N}$ is # primitive evaluations)

classical attacks against RO ($\mathcal{N}$ is # oracle evaluations)

- Attacks already described in [BDPV07]
- Tightened preimage resistance bound by Lefevre and Mennink [LM22]:

Preimage resistance: $\min\left\{\mathcal{N}/2^{n-r}, \mathcal{N}^2/2^c\right\} + \mathcal{N}/2^n$ ← attack in $\min\{2^{n-r} + 2^{c/2}, 2^n\}$

Application to Ascon-Hash and Ascon-(C)XOF Parameters [SMK+25]

- $(b, c, r, n) = \begin{cases} (320, 256, 64, 256) \text{ for Ascon-Hash} \\ (320, 256, 64, \infty) \text{ for Ascon-XOF} \\ (320, 256, 64, \infty) \text{ for Ascon-CXOF} \end{cases}$

Application to Ascon-Hash and Ascon-(C)XOF Parameters [SMK+25]

- $(b, c, r, n) = \begin{cases} (320, 256, 64, 256) \text{ for Ascon-Hash} \\ (320, 256, 64, \infty) \text{ for Ascon-XOF} \\ (320, 256, 64, \infty) \text{ for Ascon-CXOF} \end{cases}$
- Generic collision resistance as long as $\mathcal{N} \ll \min\{2^{128}, 2^{n/2}\}$

Application to Ascon-Hash and Ascon-(C)XOF Parameters [SMK+25]

- $(b, c, r, n) = \begin{cases} (320, 256, 64, 256) \text{ for Ascon-Hash} \\ (320, 256, 64, \infty) \text{ for Ascon-XOF} \\ (320, 256, 64, \infty) \text{ for Ascon-CXOF} \end{cases}$
- Generic collision resistance as long as $\mathcal{N} \ll \min\{2^{128}, 2^{n/2}\}$
- Generic second preimage resistance as long as $\mathcal{N} \ll \min\{2^{128}, 2^{n}\}$

Application to Ascon-Hash and Ascon-(C)XOF Parameters [SMK+25]

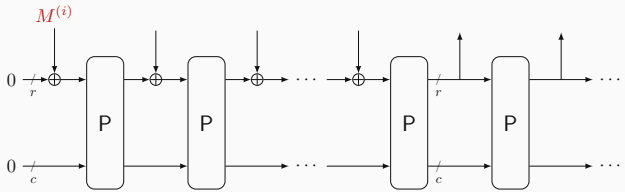
- $(b, c, r, n) = \begin{cases} (320, 256, 64, 256) \text{ for Ascon-Hash} \\ (320, 256, 64, \infty) \text{ for Ascon-XOF} \\ (320, 256, 64, \infty) \text{ for Ascon-CXOF} \end{cases}$
- Generic collision resistance as long as $\mathcal{N} \ll \min\{2^{128}, 2^{n/2}\}$
- Generic second preimage resistance as long as $\mathcal{N} \ll \min\{2^{128}, 2^{n}\}$
- Generic preimage resistance as long as $\mathcal{N} \ll \min\{\cancel{2^{128}} 2^{192}, 2^{n}\}$

Application to Ascon-Hash and Ascon-(C)XOF Parameters [SMK+25]

- $(b, c, r, n) = \begin{cases} (320, 256, 64, 256) \text{ for Ascon-Hash} \\ (320, 256, 64, \infty) \text{ for Ascon-XOF} \\ (320, 256, 64, \infty) \text{ for Ascon-CXOF} \end{cases}$
- Generic collision resistance as long as $\mathcal{N} \ll \min\{2^{128}, 2^{n/2}\}$
- Generic second preimage resistance as long as $\mathcal{N} \ll \min\{2^{128}, 2^{n}\}$
- Generic preimage resistance as long as $\mathcal{N} \ll \min\{\cancel{2^{128}} 2^{192}, 2^{n}\}$

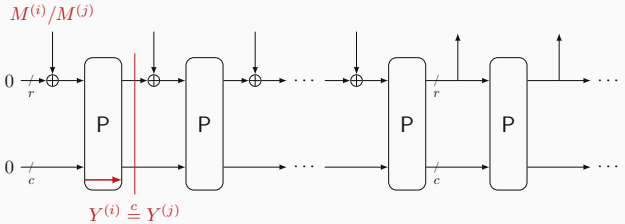
Can We Beat First and Second Preimage Bounds?

Intermezzo: Collision Attack [BDPV07]



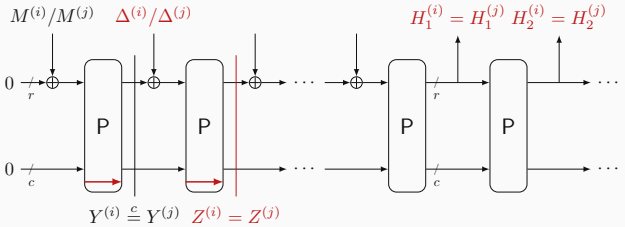
- Absorb $2^{c/2}$ different messages $M^{(i)}$

Intermezzo: Collision Attack [BDPV07]



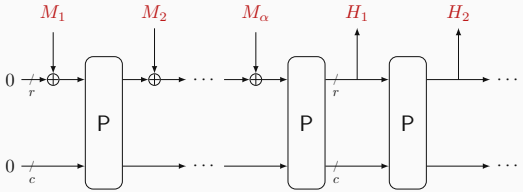
- Absorb $2^{c/2}$ different messages $M^{(i)}$
- With high probability, there exist $Y^{(i)} \neq Y^{(j)}$ that collide on their inner part

Intermezzo: Collision Attack [BDPV07]



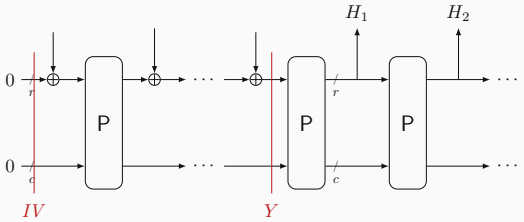
- Absorb $2^{c/2}$ different messages $M^{(i)}$
- With high probability, there exist $Y^{(i)} \neq Y^{(j)}$ that collide on their inner part
- Compensate the difference in next absorb call
- Triggers a full state collision and hence digest collision

Intermezzo: Second Preimage Attack [BDPV07]



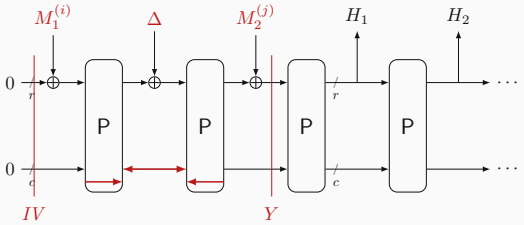
- Let $M_1 \| M_2 \| \cdots \| M_\alpha$ be the first preimage

Intermezzo: Second Preimage Attack [BDPV07]



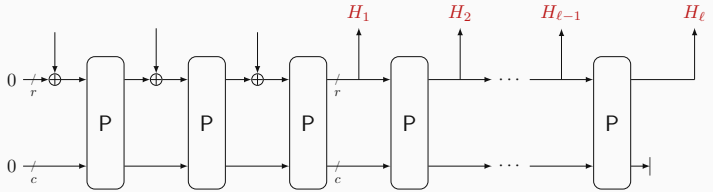
- Let $M_1 \| M_2 \| \cdots \| M_\alpha$ be the first preimage
- Retrieve Y, the state right before squeezing

Intermezzo: Second Preimage Attack [BDPV07]



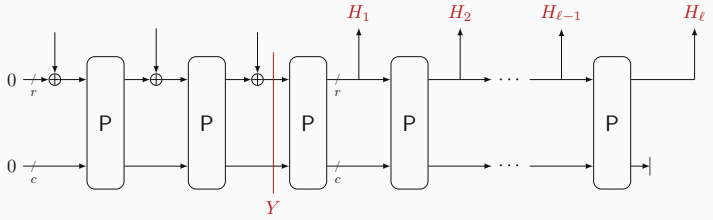
- Let $M_1 \| M_2 \| \cdots \| M_\alpha$ be the first preimage
- Retrieve Y, the state right before squeezing
- Connect the IV and Y with an inner collision
- Attack in $\approx 2^{c/2}$ queries

Intermezzo: First Preimage Attack [BDPV07]



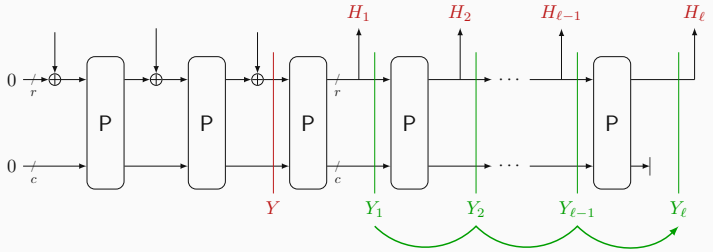
- Let $H_1 \| H_2 \| \cdots \| H_\ell$ be the target image

Intermezzo: First Preimage Attack [BDPV07]



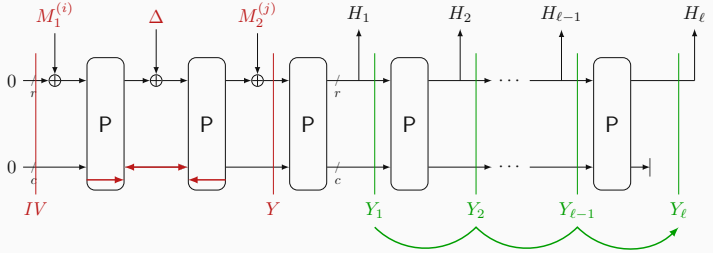
- Let $H_1 \| H_2 \| \cdots \| H_\ell$ be the target image
- No intermediate state Y is given: we first need to find it

Intermezzo: First Preimage Attack [BDPV07]



- Let $H_1 \| H_2 \| \cdots \| H_\ell$ be the target image
- No intermediate state Y is given: we first need to find it
  - Guess any Y_1 and verify follow-up $\ell - 1$ squeezes (success prob. $\approx \frac{1}{2^{n-r}}$)

Intermezzo: First Preimage Attack [BDPV07]



- Let $H_1 \| H_2 \| \cdots \| H_\ell$ be the target image
- No intermediate state Y is given: we first need to find it
  - Guess any Y_1 and verify follow-up $\ell - 1$ squeezes (success prob. $\approx \frac{1}{2^{n-r}}$)
- Connect the IV and Y with an inner collision
- Attack in $\approx 2^{n-r} + 2^{c/2}$ queries

Beating First and Second Preimage Bounds?

Crux Observation

- All attacks rely on finding inner collisions in $\approx 2^{c/2}$ queries

Beating First and Second Preimage Bounds?

Crux Observation

- All attacks rely on finding inner collisions in $\approx 2^{c/2}$ queries
  - Collision attack only evaluates P in forward direction
  - First and second preimage attacks evaluate P in both directions

Beating First and Second Preimage Bounds?

Crux Observation

- All attacks rely on finding inner collisions in $\approx 2^{c/2}$ queries
  - Collision attack only evaluates P in forward direction
  - First and second preimage attacks evaluate P in both directions
- What if the compression is hard to invert?

Beating First and Second Preimage Bounds?

Crux Observation

- All attacks rely on finding inner collisions in $\approx 2^{c/2}$ queries
  - Collision attack only evaluates P in forward direction
  - First and second preimage attacks evaluate P in both directions
- What if the compression is hard to invert?

Sponge With Transformation

- Analyzed by our Foekens in 2023 [Foe23]
  - Preimage resistance up to $\approx 2^n$ queries
  - Second preimage resistance up to $\approx \min\{2^n, 2^c/\alpha\}$ queries (for first preimage of α blocks)

Beating First and Second Preimage Bounds?

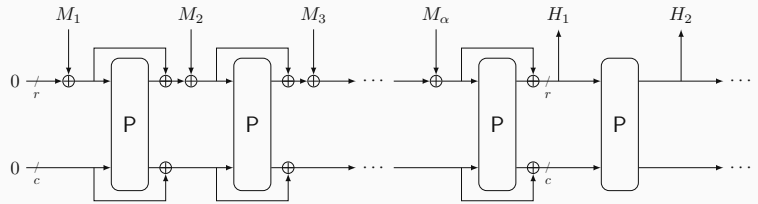
Crux Observation

- All attacks rely on finding inner collisions in $\approx 2^{c/2}$ queries
  - Collision attack only evaluates P in forward direction
  - First and second preimage attacks evaluate P in both directions
- What if the compression is hard to invert?

Sponge With Transformation

- Analyzed by our Foekens in 2023 [Foe23]
  - Preimage resistance up to $\approx 2^n$ queries
  - Second preimage resistance up to $\approx \min\{2^n, 2^c/\alpha\}$ queries (for first preimage of α blocks)
- Core inspiration for Sponge-DM and Sponge-EDM [SLZ+25]

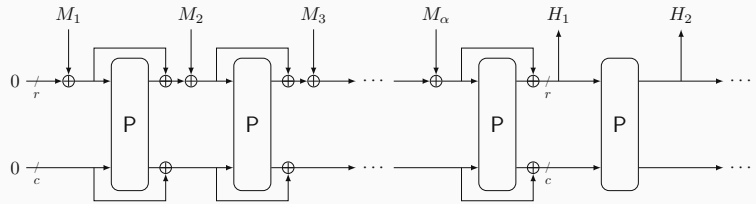
Sponge-DM [SLZ+25]



- Sponge-DM adds a feed-forward during absorption*

* Similar approach later also adopted by Guo et al. [GHJ+25]

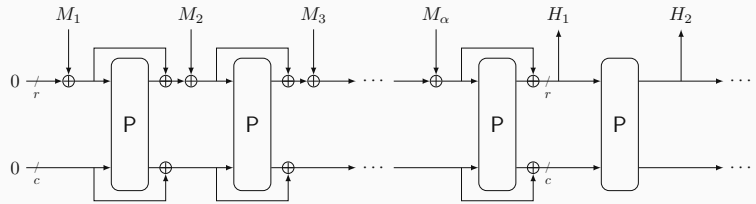
Sponge-DM [SLZ+25]



- Sponge-DM adds a feed-forward during absorption*
  - Preimage resistance up to $\approx 2^n$ queries
  - Second preimage resistance up to $\approx \min\{2^n, 2^c/\alpha\}$ queries
  - Indifferentiability up to original bound

* Similar approach later also adopted by Guo et al. [GHJ+25]

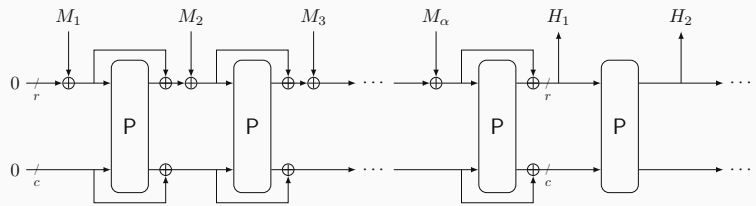
Sponge-DM [SLZ+25]



- Sponge-DM adds a feed-forward during absorption*
  - Preimage resistance up to $\approx 2^n$ queries
  - Second preimage resistance up to $\approx \min\{2^n, 2^c/\alpha\}$ queries
  - Indifferentiability up to original bound
- Relevant for applications that require high (second) preimage resistance

* Similar approach later also adopted by Guo et al. [GHJ+25]

Sponge-DM [SLZ+25]



- Sponge-DM adds a feed-forward during absorption*
  - Preimage resistance up to $\approx 2^n$ queries
  - Second preimage resistance up to $\approx \min\{2^n, 2^c/\alpha\}$ queries
  - Indifferentiability up to original bound
- Relevant for applications that require high (second) preimage resistance

Can We Reduce Feed-Forward Size?

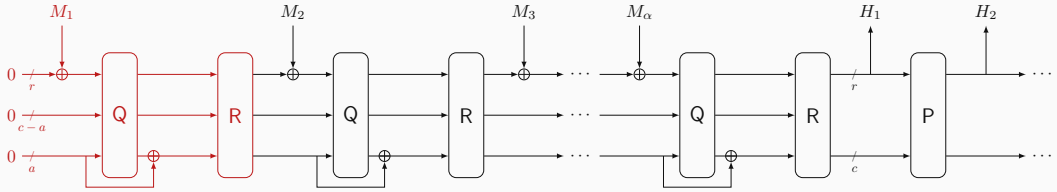
* Similar approach later also adopted by Guo et al. [GHJ+25]

Sponge-EDM^a for $a \in \{0, \ldots, b\}$ [SLZ$^+$25]

- Just feed-forwarding the inner part does not work

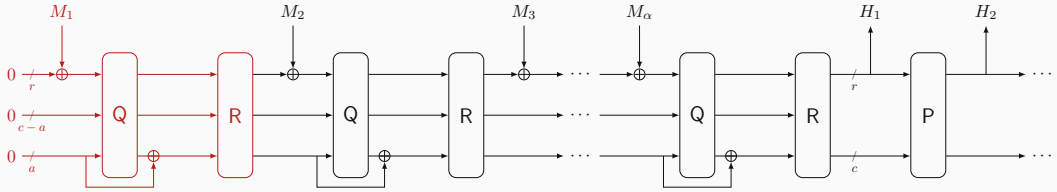
Sponge-EDM^a for $a \in \{0, \ldots, b\}$ [SLZ+25]

- Just feed-forwarding the inner part does not work
- Augmenting absorption with extra permutation call resolves this
  - Q and R may be round-reduced permutations



Sponge-EDM^a for $a \in \{0, \ldots, b\}$ [SLZ+25]

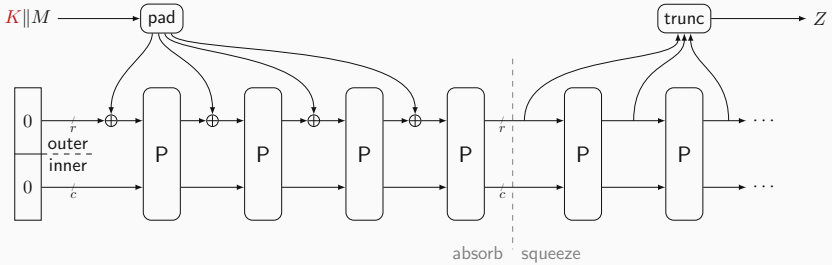
- Just feed-forwarding the inner part does not work
- Augmenting absorption with extra permutation call resolves this
  - Q and R may be round-reduced permutations



- Security ranges from that of Sponge-DM (for $a = b$) to Sponge (for $a = 0$)

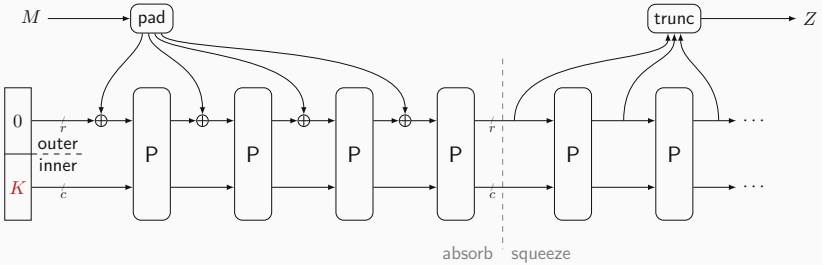
From Outer- to Full-Keyed Sponge

Evolution of Keyed Sponges



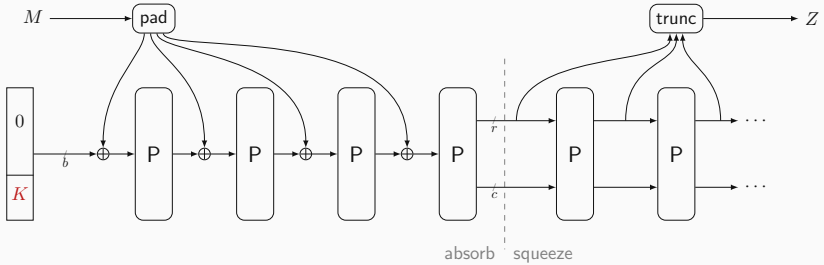
- Outer-Keyed Sponge [BDPV11b, ADMV15, NY16, Men18]

Evolution of Keyed Sponges



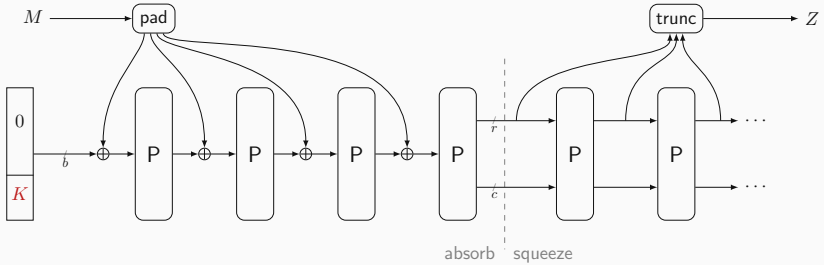
- Outer-Keyed Sponge [BDPV11b, ADMV15, NY16, Men18]
- Inner-Keyed Sponge [CDH+12, ADMV15, NY16]

Evolution of Keyed Sponges



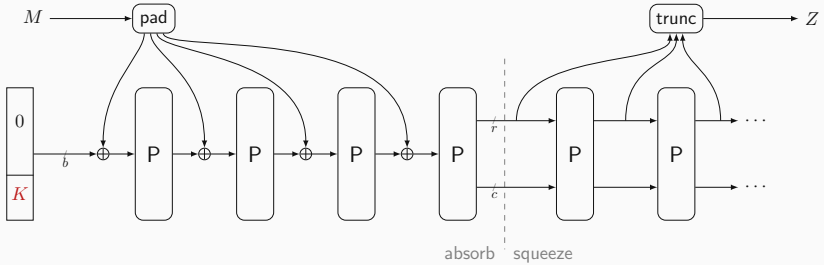
- Outer-Keyed Sponge [BDPV11b, ADMV15, NY16, Men18]
- Inner-Keyed Sponge [CDH+12, ADMV15, NY16]
- Full-Keyed Sponge [BDPV12, GPT15, MRV15]

Evolution of Keyed Sponges



- Outer-Keyed Sponge [BDPV11b, ADMV15, NY16, Men18]
- Inner-Keyed Sponge [CDH+12, ADMV15, NY16]
- Full-Keyed Sponge [BDPV12, GPT15, MRV15]
- Generic security does not degrade: all can be used for PRF or MAC

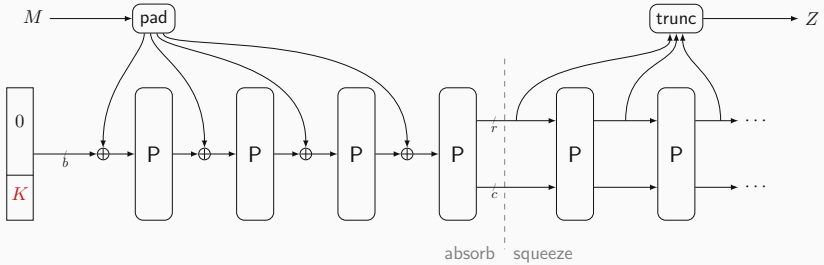
Security of Full-Keyed Sponge (1/3)



Setting

- Assume random b-bit permutation P
- Rate r and capacity c with $b = r + c$; key size $k \leq c$

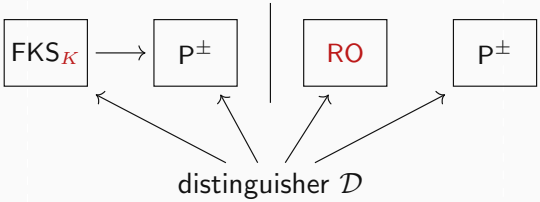
Security of Full-Keyed Sponge (1/3)



Setting

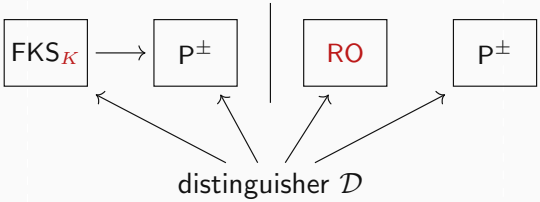
- Assume random b-bit permutation P
- Rate r and capacity c with $b = r + c$; key size $k \leq c$
- FKS_K should behave like a random oracle RO

Security of Full-Keyed Sponge (2/3)



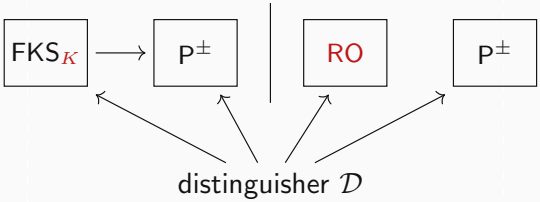
- Two oracles: FKS$_K$ (for secret key K) and RO (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these, plus the random permutation P

Security of Full-Keyed Sponge (2/3)



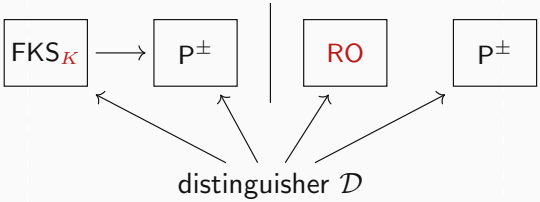
- Two oracles: FKS$_K$ (for secret key K) and RO (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these, plus the random permutation P
- $\mathcal{D}$ tries to determine which oracle it communicates with

Security of Full-Keyed Sponge (2/3)



- Two oracles: FKS$_K$ (for secret key K) and RO (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these, plus the random permutation P
- $\mathcal{D}$ tries to determine which oracle it communicates with
- It has construction query budget $\mathcal{M}$ and primitive query budget $\mathcal{N}$

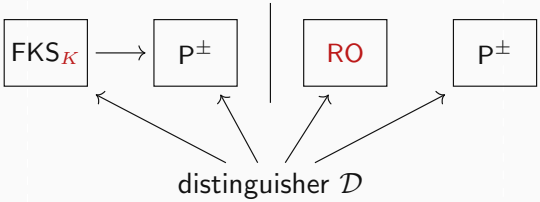
Security of Full-Keyed Sponge (2/3)



- Two oracles: FKS$_K$ (for secret key K) and RO (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these, plus the random permutation P
- $\mathcal{D}$ tries to determine which oracle it communicates with
- It has construction query budget $\mathcal{M}$ and primitive query budget $\mathcal{N}$
- Its advantage is defined as:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FKS}}(\mathcal{D}) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{FKS}_K,\mathsf{P}^{\pm}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{RO},\mathsf{P}^{\pm}} = 1\right)\right|$$

Security of Full-Keyed Sponge (2/3)

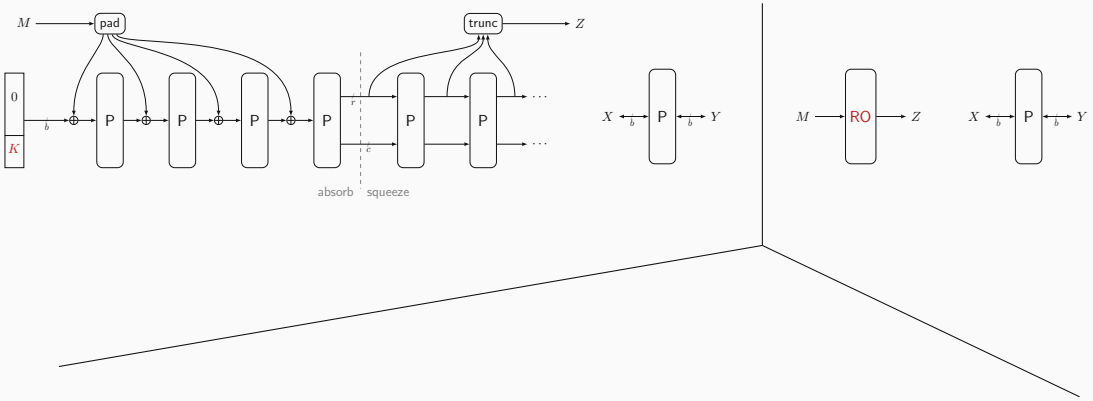


- Two oracles: FKS$_K$ (for secret key K) and RO (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these, plus the random permutation P
- $\mathcal{D}$ tries to determine which oracle it communicates with
- It has construction query budget $\mathcal{M}$ and primitive query budget $\mathcal{N}$
- Its advantage is defined as:

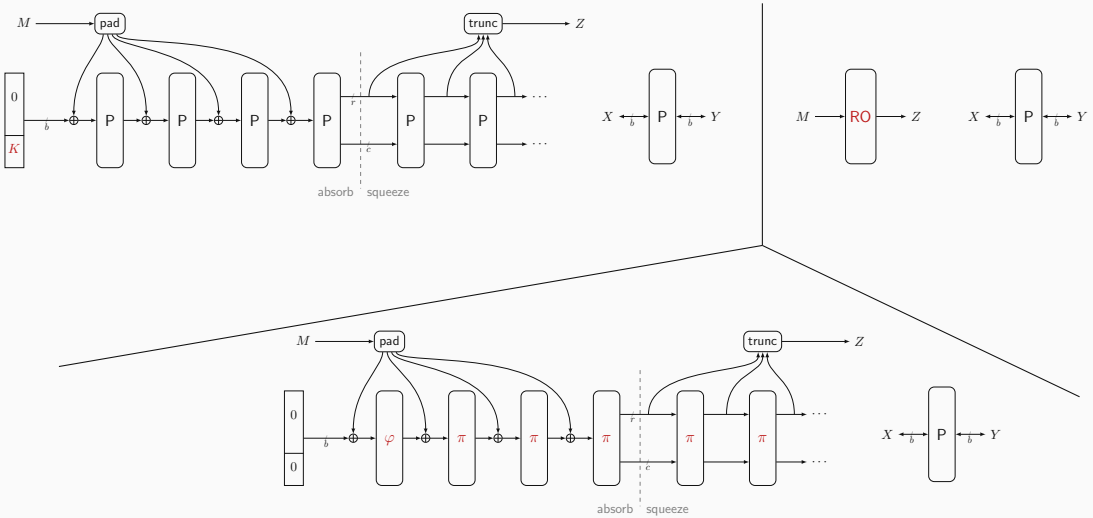
scheme behaves randomly as long as this distance is $\ll 1$

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FKS}}(\mathcal{D}) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{FKS}_K,\mathsf{P}^{\pm}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{RO},\mathsf{P}^{\pm}} = 1\right)\right|$$

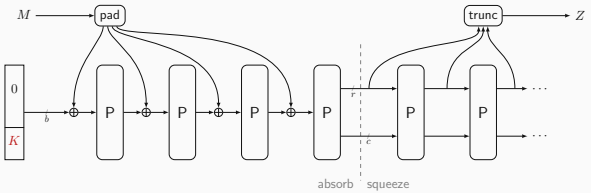
Security of Full-Keyed Sponge (3/3)

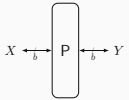


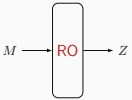
Security of Full-Keyed Sponge (3/3)

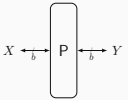


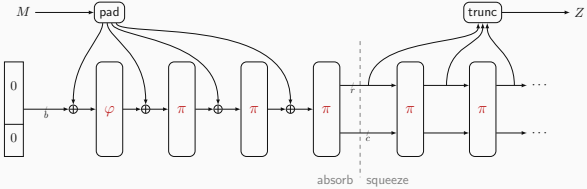
Security of Full-Keyed Sponge (3/3)

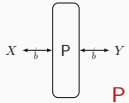






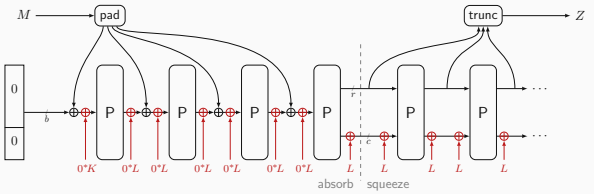




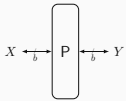


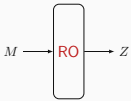
P irrelevant; security of random Sponge; $\approx \mathcal{M}^2/2^c$

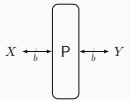
Security of Full-Keyed Sponge (3/3)

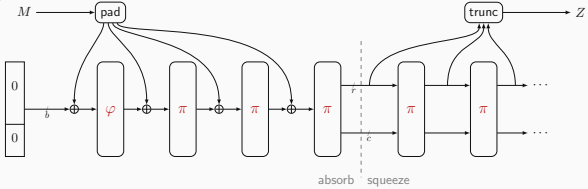


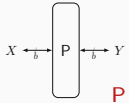
L is c-bit dummy key
(recall: K is k-bit key)





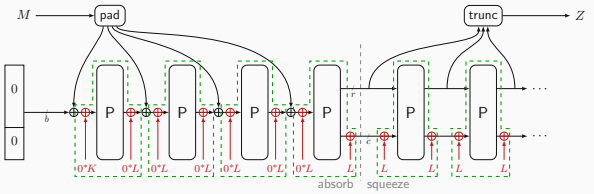




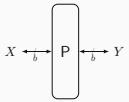


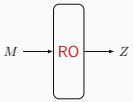
P irrelevant;
security of
random Sponge;
$\approx \mathcal{M}^2/2^c$

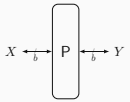
Security of Full-Keyed Sponge (3/3)

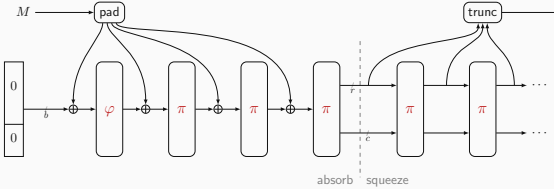


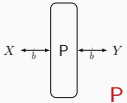
L is c-bit dummy key
(recall: K is k-bit key)





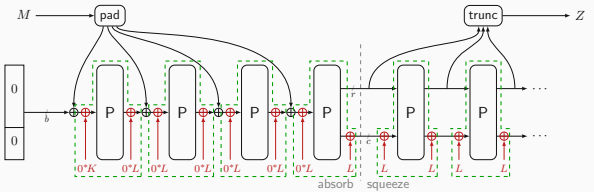




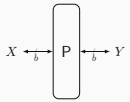


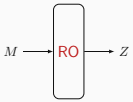
P irrelevant;
security of
random Sponge;
$\approx \mathcal{M}^2/2^c$

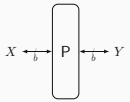
Security of Full-Keyed Sponge (3/3)



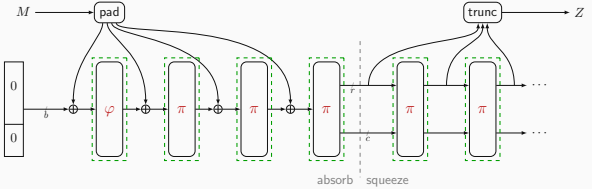
L is c-bit dummy key
(recall: K is k-bit key)

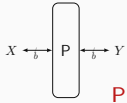






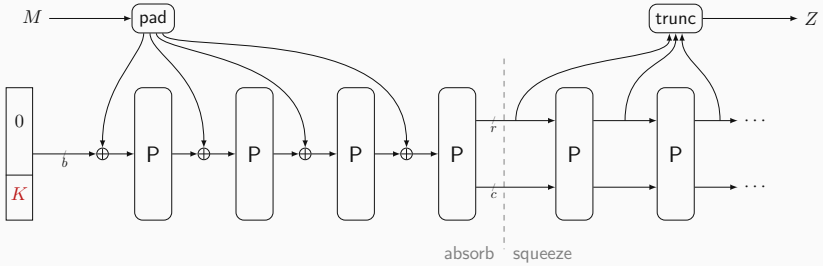
security of two
P-based Even-Mansours;
$\approx \mathcal{MN}/2^c + \mathcal{N}/2^k$



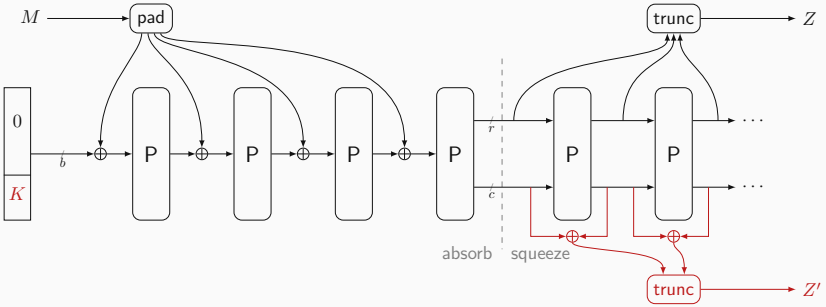


P irrelevant;
security of
random Sponge;
$\approx \mathcal{M}^2/2^c$

We Optimized Absorbing, Can We Optimize Squeezing?



We Optimized Absorbing, Can We Optimize Squeezing?

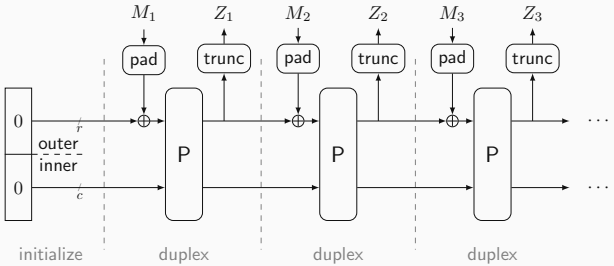


MacaKey-Style Squeezing [LM25a]

- Elegant combination of Sponge with summation-truncation hybrid [GM20]
  - Follow-up work by Bhaumik et al. [BCD26]
- Generic security does not degrade by b-bit squeezing

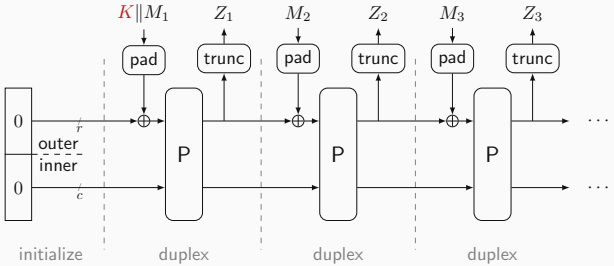
From Original to Full-Keyed Duplex

Evolution of Keyed Duplexes



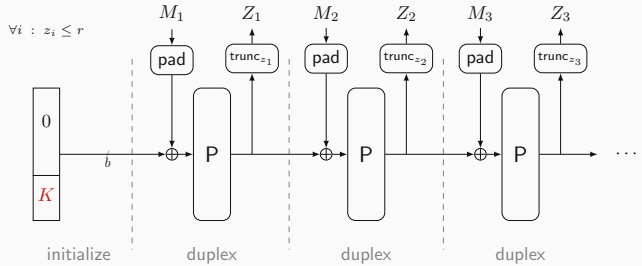
- Unkeyed Duplex [BDPV11a]

Evolution of Keyed Duplexes



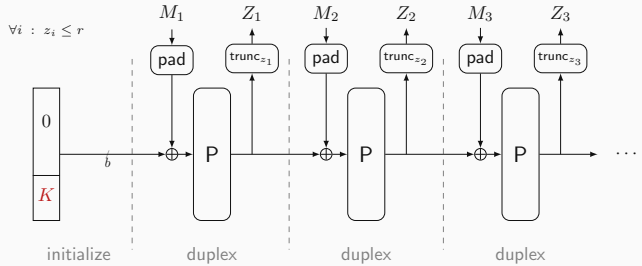
- Unkeyed Duplex [BDPV11a]
- Outer-Keyed Duplex [BDPV11a]

Evolution of Keyed Duplexes



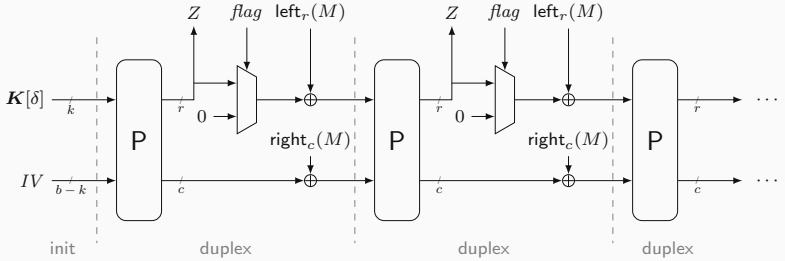
- Unkeyed Duplex [BDPV11a]
- Outer-Keyed Duplex [BDPV11a]
- Full-Keyed Duplex [MRV15, DMV17, DM19b, Men23]

Evolution of Keyed Duplexes

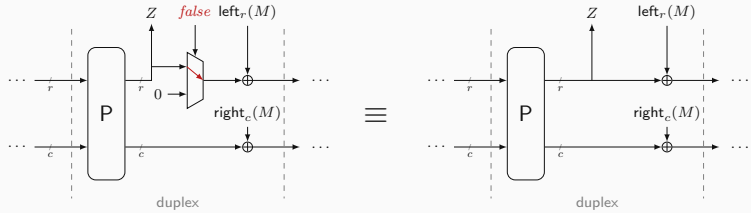


- Unkeyed Duplex [BDPV11a]
- Outer-Keyed Duplex [BDPV11a]
- Full-Keyed Duplex [MRV15, DMV17, DM19b, Men23]

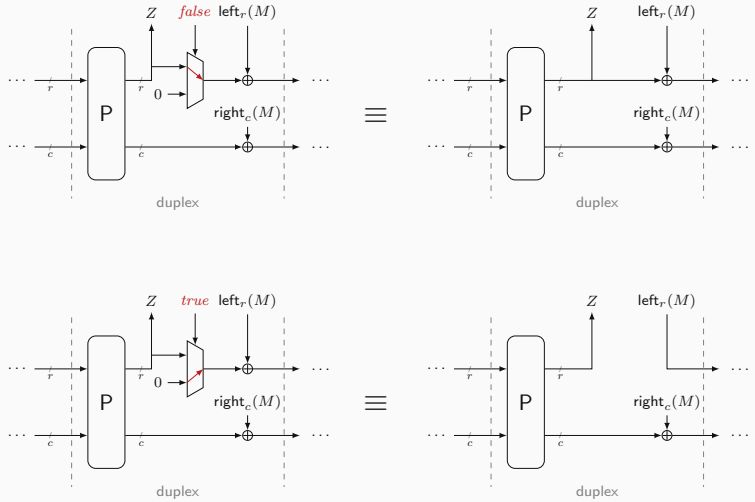
Generalized Keyed Duplex [DMV17, DM19a, Men23]



Generalized Keyed Duplex: Flag (1/2)



Generalized Keyed Duplex: Flag (1/2)



Generalized Keyed Duplex: Flag (2/2)

Extremely Simplified SpongeWrap

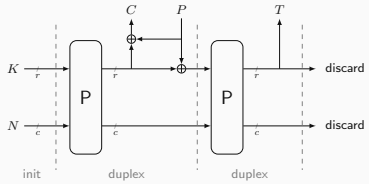
- Key K, plaintext P, ciphertext C, and tag T all r bits; nonce N c bits

Generalized Keyed Duplex: Flag (2/2)

Extremely Simplified SpongeWrap

- Key K, plaintext P, ciphertext C, and tag T all r bits; nonce N c bits

Encryption

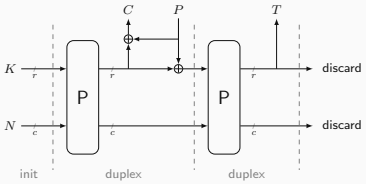


Generalized Keyed Duplex: Flag (2/2)

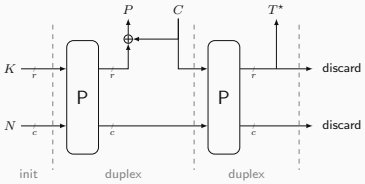
Extremely Simplified SpongeWrap

- Key K, plaintext P, ciphertext C, and tag T all r bits; nonce N c bits

Encryption



Decryption

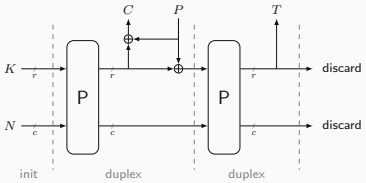


Generalized Keyed Duplex: Flag (2/2)

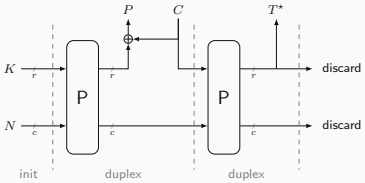
Extremely Simplified SpongeWrap

- Key K, plaintext P, ciphertext C, and tag T all r bits; nonce N c bits

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

Generalized Keyed Duplex: Security Model

Algorithm Keyed Duplex construction $\mathsf{KD}[\mathsf{P}]_{\boldsymbol{K}}$

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

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

Generalized Keyed Duplex: Security Model

Algorithm Keyed Duplex construction $\mathsf{KD}[\mathsf{P}]_{\boldsymbol{K}}$

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

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

Algorithm Ideal extendable input function IXIF[RO]

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

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

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

Generalized Keyed Duplex: Security Model

Algorithm Keyed Duplex construction $\mathsf{KD}[\mathsf{P}]_{\boldsymbol{K}}$

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

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

Algorithm Ideal extendable input function IXIF[RO]

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

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

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

$$\mathbf{Adv}_{\mathsf{KD}}(\mathcal{D}) = \left| \mathbf{Pr}\left(\mathcal{D}^{\mathsf{KD}[\mathsf{P}]_{\boldsymbol{K}}, \mathsf{P}^{\pm}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{IXIF}[\mathsf{RO}], \mathsf{P}^{\pm}} = 1\right) \right|$$

Generalized Keyed Duplex: Security Model

Algorithm Keyed Duplex construction $\mathsf{KD}[\mathsf{P}]_{\boldsymbol{K}}$

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

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

Algorithm Ideal extendable input function IXIF[RO]

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

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

$$\mathbf{Adv}_{\mathsf{KD}}(\mathcal{D}) = \left| \mathbf{Pr}\left(\mathcal{D}^{\mathsf{KD}[\mathsf{P}]_{\boldsymbol{K}}, \mathsf{P}^{\pm}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{IXIF}[\mathsf{RO}], \mathsf{P}^{\pm}} = 1\right) \right|$$

- IXIF[RO] is basically random oracle in disguise

Generalized Keyed Duplex: Security Model

Algorithm Keyed Duplex construction $\mathsf{KD}[\mathsf{P}]_{\boldsymbol{K}}$

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

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

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

$$\mathbf{Adv}_{\mathsf{KD}}(\mathcal{D}) = \left| \mathbf{Pr}\left(\mathcal{D}^{\mathsf{KD}[\mathsf{P}]_{\boldsymbol{K}}, \mathsf{P}^{\pm}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{IXIF}[\mathsf{RO}], \mathsf{P}^{\pm}} = 1\right) \right|$$

- IXIF[RO] is basically random oracle in disguise
- We prove that $\mathsf{KD}[\mathsf{P}]_{\boldsymbol{K}}$ behaves like IXIF[RO] for certain bound on adversarial resources

Generalized Keyed Duplex: Security Model

Algorithm Keyed Duplex construction $\mathsf{KD}[\mathsf{P}]_{\boldsymbol{K}}$

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

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

Algorithm Ideal extendable input function IXIF[RO]

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

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

$$\mathbf{Adv}_{\mathsf{KD}}(\mathcal{D}) = \left| \mathbf{Pr}\left(\mathcal{D}^{\mathsf{KD}[\mathsf{P}]_{\boldsymbol{K}}, \mathsf{P}^{\pm}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{IXIF}[\mathsf{RO}], \mathsf{P}^{\pm}} = 1\right) \right|$$

- IXIF[RO] is basically random oracle in disguise
- We prove that $\mathsf{KD}[\mathsf{P}]_{\boldsymbol{K}}$ behaves like IXIF[RO] for certain bound on adversarial resources
  - Bound on adversarial resources is in turn determined by use case!

Generalized Keyed Duplex: Security Bounds From [DMV17, DM19a]

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

Simplified Security Bound

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

Generalized Keyed Duplex: Security Bounds From [DMV17, DM19a]

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

Simplified Security Bound

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

Actual Security Bounds

$$\mathbf{Adv}_{\mathsf{KD}}(\mathcal{D}) \leq \frac{(\mathcal{L}+\Omega)\mathcal{N}}{2^c} + \frac{2\nu_{r,c}^{2(\mathcal{M}-\mathcal{L})}(\mathcal{N}+1)}{2^c} + \frac{\binom{\mathcal{L}+\Omega+1}{2}}{2^c} + \frac{(\mathcal{M}-\mathcal{L}-\mathcal{Q})\mathcal{Q}}{2^b-\mathcal{Q}} + \frac{\mathcal{M}(\mathcal{M}-\mathcal{L}-1)}{2^b} + \frac{\mathcal{Q}(\mathcal{M}-\mathcal{L}-\mathcal{Q})}{2^{\min\{c+k,\max\{b-\alpha,c\}\}}} + \frac{\mathcal{Q}_{IV}\mathcal{N}}{2^k} + \frac{\binom{\mu}{2}}{2^k} \quad \text{[DMV17]}$$

$$\mathbf{Adv}_{\mathsf{KD}}(\mathcal{D}) \leq \frac{(\mathcal{L}+\Omega)\mathcal{N}}{2^c} + \frac{2\nu_{r,c}^{\mathcal{M}}(\mathcal{N}+1)}{2^c} + \frac{\nu_{r,c}^{\mathcal{M}}(\mathcal{L}+\Omega) + \binom{\mathcal{L}+\Omega}{2}}{2^c} + \frac{\binom{\mathcal{M}-\mathcal{L}-\mathcal{Q}}{2} + (\mathcal{M}-\mathcal{L}-\mathcal{Q})(\mathcal{L}+\Omega)}{2^b} + \frac{\binom{\mathcal{M}+\mathcal{N}}{2} + \binom{\mathcal{N}}{2}}{2^b} + \frac{\mathcal{Q}(\mathcal{M}-\mathcal{Q})}{2^{\min\{c+k,\max\{b-\alpha,c\}\}}} + \frac{\mathcal{Q}_{IV}\mathcal{N}}{2^k} + \frac{\binom{\mu}{2}}{2^k} \quad \text{[DM19b]}$$

Generalized Keyed Duplex: Security Bounds From [DMV17, DM19a]

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

Simplified Security Bound

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

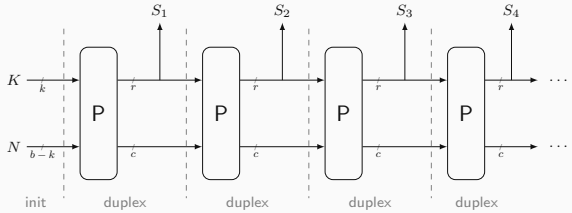
Actual Security Bounds

$$\mathbf{Adv}_{\mathsf{KD}}(\mathcal{D}) \leq \frac{(\mathcal{L}+\Omega)\mathcal{N}}{2^c} + \frac{2\nu_{r,c}^{2(\mathcal{M}-\mathcal{L})}(\mathcal{N}+1)}{2^c} + \frac{\binom{\mathcal{L}+\Omega+1}{2}}{2^c} + \frac{(\mathcal{M}-\mathcal{L}-\mathcal{Q})\mathcal{Q}}{2^b-\mathcal{Q}} + \frac{\mathcal{M}(\mathcal{M}-\mathcal{L}-1)}{2^b} + \frac{\mathcal{Q}(\mathcal{M}-\mathcal{L}-\mathcal{Q})}{2^{\min\{c+k,\max\{b-\alpha,c\}\}}} + \frac{\mathcal{Q}_{IV}\mathcal{N}}{2^k} + \frac{\binom{\mu}{2}}{2^k} \quad \text{[DMV17]}$$

$$\mathbf{Adv}_{\mathsf{KD}}(\mathcal{D}) \leq \frac{(\mathcal{L}+\Omega)\mathcal{N}}{2^c} + \frac{2\nu_{r,c}^{\mathcal{M}}(\mathcal{N}+1)}{2^c} + \frac{\nu_{r,c}^{\mathcal{M}}(\mathcal{L}+\Omega) + \binom{\mathcal{L}+\Omega}{2}}{2^c} + \frac{\binom{\mathcal{M}-\mathcal{L}-\mathcal{Q}}{2} + (\mathcal{M}-\mathcal{L}-\mathcal{Q})(\mathcal{L}+\Omega)}{2^b} + \frac{\binom{\mathcal{M}+\mathcal{N}}{2} + \binom{\mathcal{N}}{2}}{2^b} + \frac{\mathcal{Q}(\mathcal{M}-\mathcal{Q})}{2^{\min\{c+k,\max\{b-\alpha,c\}\}}} + \frac{\mathcal{Q}_{IV}\mathcal{N}}{2^k} + \frac{\binom{\mu}{2}}{2^k} \quad \text{[DM19b]}$$

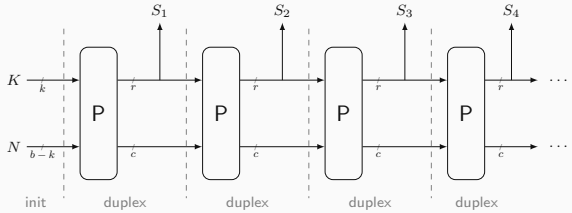
Bounds Simplify For Particular Applications!

Application to Keystream Generation



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

Application to Keystream Generation

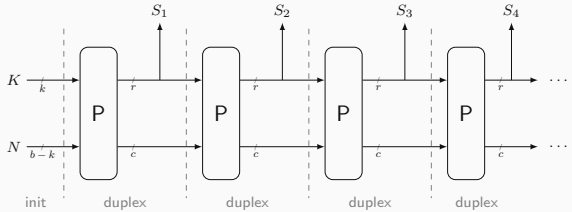


- Input: key K, nonce N
- Output: keystream S of requested length
- Can be described using Duplex

Algorithm Keystream generation SC[P]

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

Application to Keystream Generation



- Input: key K, nonce N
- Output: keystream S of requested length
- Can be described using Duplex

- Strong simplified bound immediately follows:

Algorithm Keystream generation SC[P]

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

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(\mathcal{D}) \leq \frac{2\nu_{r,c}^{2\mathcal{M}}(\mathcal{N}+1)}{2^c} + \frac{(\mathcal{M}-\mathcal{Q})\mathcal{Q}}{2^b-\mathcal{Q}} + \frac{2\binom{\mathcal{M}}{2}}{2^b} + \frac{\mathcal{Q}(\mathcal{M}-\mathcal{Q})}{2^{\min\{c+k,b\}}} + \frac{\mathcal{N}}{2^k}$$

Application to Keystream Generation: Security (1/2)

- Consider distinguisher $\mathcal{D}$ against PRF security of SC[P]

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

- $\mathcal{D}$ can make $\mathcal{Q}$ construction queries (total $\mathcal{M}$ blocks) and $\mathcal{N}$ primitive queries

Application to Keystream Generation: Security (1/2)

- Consider distinguisher $\mathcal{D}$ against PRF security of SC[P]

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

- $\mathcal{D}$ can make $\mathcal{Q}$ construction queries (total $\mathcal{M}$ blocks) and $\mathcal{N}$ primitive queries
- $\mathsf{SC}[\mathsf{P}]_K$ is basically just $\mathsf{SC}[\mathsf{KD}[\mathsf{P}]_K]$

Application to Keystream Generation: Security (1/2)

- Consider distinguisher $\mathcal{D}$ against PRF security of SC[P]

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

- $\mathcal{D}$ can make $\mathcal{Q}$ construction queries (total $\mathcal{M}$ blocks) and $\mathcal{N}$ primitive queries
- $\mathsf{SC}[\mathsf{P}]_K$ is basically just $\mathsf{SC}[\mathsf{KD}[\mathsf{P}]_K]$
- Triangle inequality:

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

Application to Keystream Generation: Security (1/2)

- Consider distinguisher $\mathcal{D}$ against PRF security of SC[P]

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

- $\mathcal{D}$ can make $\mathcal{Q}$ construction queries (total $\mathcal{M}$ blocks) and $\mathcal{N}$ primitive queries
- $\mathsf{SC}[\mathsf{P}]_K$ is basically just $\mathsf{SC}[\mathsf{KD}[\mathsf{P}]_K]$
- Triangle inequality:

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

Application to Keystream Generation: Security (1/2)

- Consider distinguisher $\mathcal{D}$ against PRF security of SC[P]

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

- $\mathcal{D}$ can make $\mathcal{Q}$ construction queries (total $\mathcal{M}$ blocks) and $\mathcal{N}$ primitive queries
- $\mathsf{SC}[\mathsf{P}]_K$ is basically just $\mathsf{SC}[\mathsf{KD}[\mathsf{P}]_K]$
- Triangle inequality:

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

$\leq \Delta_{\mathcal{D}'}\left(\mathsf{KD}[\mathsf{P}]_K, \mathsf{P}^{\pm}\ ;\ \mathsf{IXIF}[\mathsf{RO}], \mathsf{P}^{\pm}\right)$ $\quad = 0$

Application to Keystream Generation: Security (1/2)

- Consider distinguisher $\mathcal{D}$ against PRF security of $\mathsf{SC}[\mathsf{P}]$

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

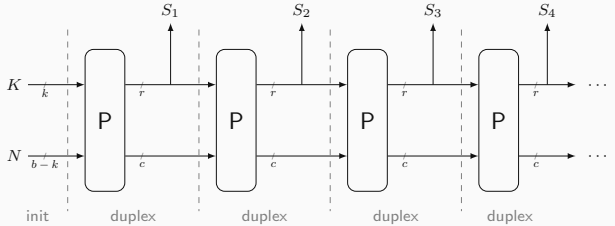
- $\mathcal{D}$ can make $\mathcal{Q}$ construction queries (total $\mathcal{M}$ blocks) and $\mathcal{N}$ primitive queries
- $\mathsf{SC}[\mathsf{P}]_K$ is basically just $\mathsf{SC}[\mathsf{KD}[\mathsf{P}]_K]$
- Triangle inequality:

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

$\leq \Delta_{\mathcal{D}'}\left(\mathsf{KD}[\mathsf{P}]_K, \mathsf{P}^{\pm}\ ;\ \mathsf{IXIF}[\mathsf{RO}], \mathsf{P}^{\pm}\right)$ $\quad = 0$

What Are the Resources of $\mathcal{D}'$?

Application to Keystream Generation: Security (2/2)

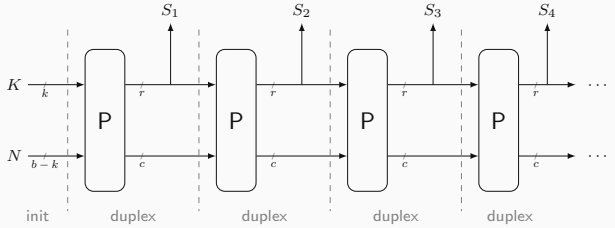


Algorithm Keystream generation SC[P]

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

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

Application to Keystream Generation: Security (2/2)

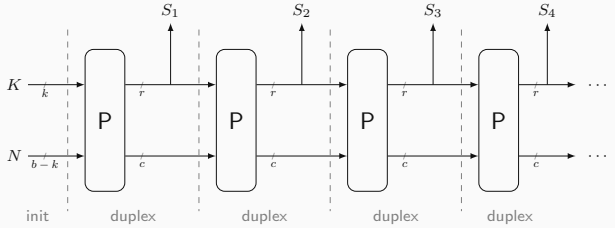


Algorithm Keystream generation SC[P]

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

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

Application to Keystream Generation: Security (2/2)

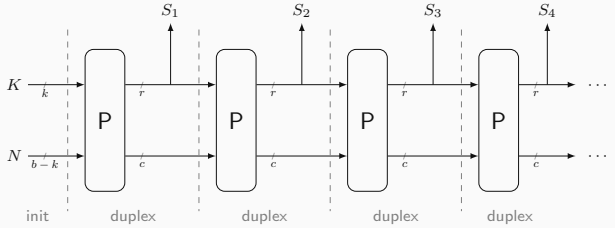


Algorithm Keystream generation SC[P]

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

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

Application to Keystream Generation: Security (2/2)

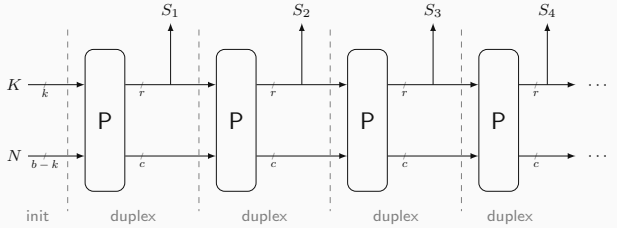


Algorithm Keystream generation SC[P]

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

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

Application to Keystream Generation: Security (2/2)

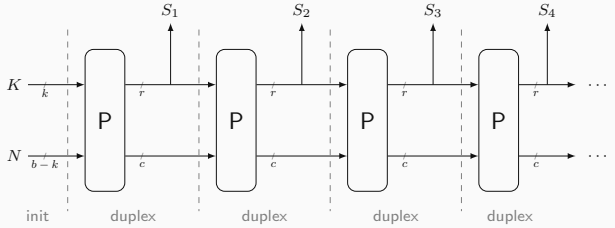


Algorithm Keystream generation SC[P]

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

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

Application to Keystream Generation: Security (2/2)

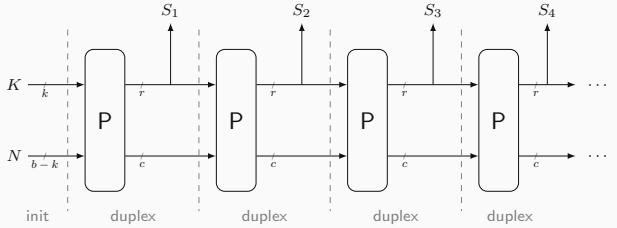


Algorithm Keystream generation SC[P]

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

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

Application to Keystream Generation: Security (2/2)



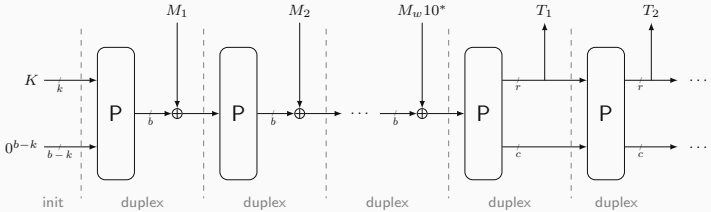
Algorithm Keystream generation SC[P]

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

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

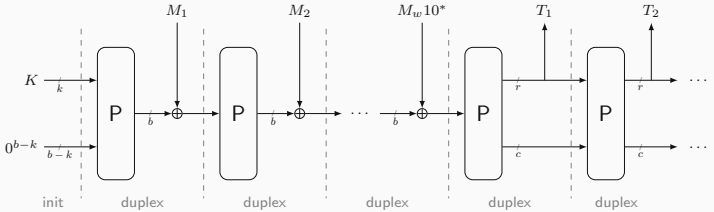
From [DMV17] (single-user setting): $\mathbf{Adv}_{\mathsf{KD}}(\mathcal{D}') \leq \frac{2\nu_{r,c}^{2\mathcal{M}}(\mathcal{N}+1)}{2^c} + \frac{(\mathcal{M}-\mathcal{Q})\mathcal{Q}}{2^b-\mathcal{Q}} + \frac{2\binom{\mathcal{M}}{2}}{2^b} + \frac{\mathcal{Q}(\mathcal{M}-\mathcal{Q})}{2^{\min\{c+k,b\}}}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

Application to Full-Keyed Sponge [BDPV12, GPT15, MRV15]



- Input: key K, message M
- Output: tag T

Application to Full-Keyed Sponge [BDPV12, GPT15, MRV15]

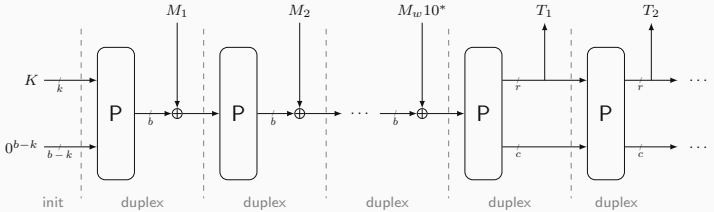


- Input: key K, message M
- Output: tag T
- Can be described using Duplex

Algorithm Full-Keyed Sponge FKS[P]

Input: $(K, M) \in \{0,1\}^k \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
  $(M_1, M_2, \ldots, M_w) \leftarrow \mathrm{pad}_b^{10^*}(M)$
  $T \leftarrow \varnothing$
  KD.init$(1, 0^{b-k})$
  for $i = 1, \ldots, w$ **do**
    KD.duplex(false, M_i) ▷ discard output
  for $i = 1, \ldots, \lceil t/r \rceil$ **do**
    $T \leftarrow T \parallel$ KD.duplex$(\mathit{false}, 0^b)$
  return left$_t(T)$

Application to Full-Keyed Sponge [BDPV12, GPT15, MRV15]



- Input: key K, message M
- Output: tag T
- Can be described using Duplex

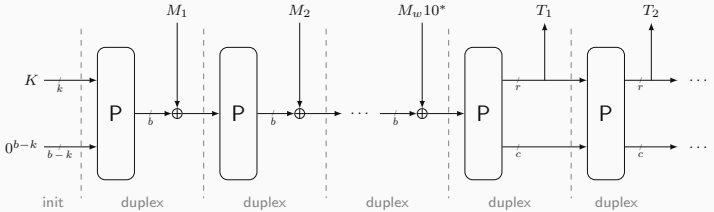
- Analysis of [MRV15] applies...
- ... but better bound follows from Duplex:

Algorithm Full-Keyed Sponge FKS[P]

Input: $(K, M) \in \{0,1\}^k \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
$(M_1, M_2, \ldots, M_w) \leftarrow \mathrm{pad}_b^{10^*}(M)$
$T \leftarrow \varnothing$
KD.init$(1, 0^{b-k})$
for $i = 1, \ldots, w$ **do**
  KD.duplex(false, M_i) ▷ discard output
for $i = 1, \ldots, \lceil t/r \rceil$ **do**
  $T \leftarrow T \,\|\, \mathrm{KD.duplex}(\mathit{false}, 0^b)$
return $\mathrm{left}_t(T)$

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FKS}}(\mathcal{D}) \leq \frac{2\nu_{r,c}^{2\mathcal{M}}(\mathcal{N}+1)}{2^c} + \frac{(\mathcal{Q}-1)\mathcal{N}+\binom{\mathcal{Q}}{2}}{2^c} + \frac{(\mathcal{M}-\mathcal{Q})\mathcal{Q}}{2^b-\mathcal{Q}} + \frac{2\binom{\mathcal{M}}{2}}{2^b} + \frac{\mathcal{Q}(\mathcal{M}-\mathcal{Q})}{2^{\min\{c+k,b\}}} + \frac{\mathcal{N}}{2^k}$$

Application to Full-Keyed Sponge [BDPV12, GPT15, MRV15]



- Input: key K, message M
- Output: tag T
- Can be described using Duplex

Algorithm Full-Keyed Sponge FKS[P]

Input: $(K, M) \in \{0,1\}^k \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
$(M_1, M_2, \ldots, M_w) \leftarrow \mathrm{pad}_b^{10^*}(M)$
$T \leftarrow \varnothing$
KD.init$(1, 0^{b-k})$
for $i = 1, \ldots, w$ **do**
  KD.duplex(false, M_i) ▷ discard output
for $i = 1, \ldots, \lceil t/r \rceil$ **do**
  $T \leftarrow T \parallel$ KD.duplex$(\mathit{false}, 0^b)$
return left$_t(T)$

- Analysis of [MRV15] applies...
- ... but better bound follows from Duplex:

influence of repeated paths

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FKS}}(\mathcal{D}) \leq \frac{2\nu_{r,c}^{2\mathcal{M}}(\mathcal{N}+1)}{2^c} + \frac{(\mathcal{Q}-1)\mathcal{N}+\binom{\mathcal{Q}}{2}}{2^c} + \frac{(\mathcal{M}-\mathcal{Q})\mathcal{Q}}{2^b-\mathcal{Q}} + \frac{2\binom{\mathcal{M}}{2}}{2^b} + \frac{\mathcal{Q}(\mathcal{M}-\mathcal{Q})}{2^{\min\{c+k,b\}}} + \frac{\mathcal{N}}{2^k}$$

Application to Full-Keyed Sponge: Power of Influencing Outer Part

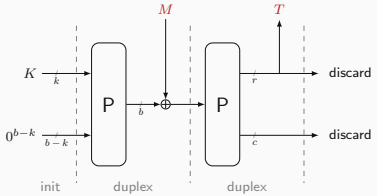
- Repeated paths can seriously affect security

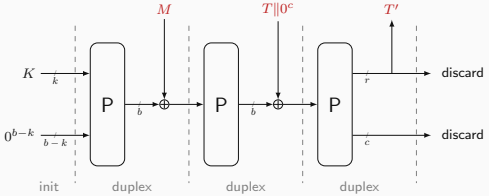
Application to Full-Keyed Sponge: Power of Influencing Outer Part

- Repeated paths can seriously affect security
- Consider simplified FKS[P]: no padding, r-bit tag

Application to Full-Keyed Sponge: Power of Influencing Outer Part

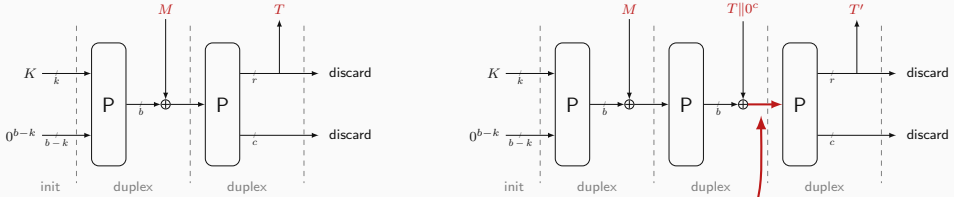
- Repeated paths can seriously affect security
- Consider simplified FKS[P]: no padding, r-bit tag
- Distinguisher makes two queries: $M \mapsto T$ and $M \| T \| 0^c \mapsto T'$





Application to Full-Keyed Sponge: Power of Influencing Outer Part

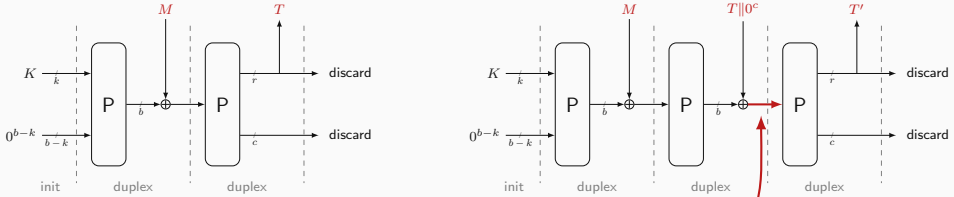
- Repeated paths can seriously affect security
- Consider simplified FKS[P]: no padding, r-bit tag
- Distinguisher makes two queries: $M \mapsto T$ and $M \| T \| 0^c \mapsto T'$



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

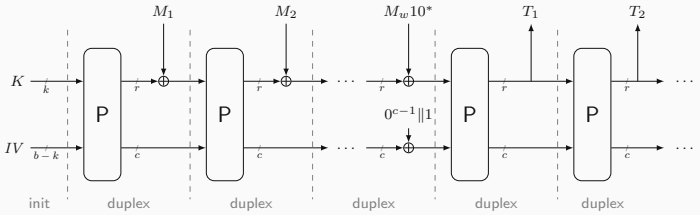
Application to Full-Keyed Sponge: Power of Influencing Outer Part

- Repeated paths can seriously affect security
- Consider simplified FKS[P]: no padding, r-bit tag
- Distinguisher makes two queries: $M \mapsto T$ and $M\|T\|0^c \mapsto T'$



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

Application to Ascon-PRF [DEMS24]

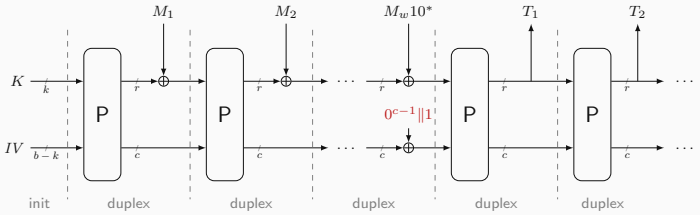


- Input: key K, initial value IV, message M
- Output: tag T

Algorithm Ascon-PRF[P]

Input: $(K, IV, M) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
$(M_1, M_2, \ldots, M_w) \leftarrow \mathrm{pad}_r^{10^*}(M)$
$T \leftarrow \varnothing$
KD.init$(1, IV)$
for $i = 1, \ldots, w-1$ **do**
  KD.duplex(false, M_i) ▷ discard output
KD.duplex$(\mathit{false}, M_w\|0^{c-1}1)$
for $i = 1, \ldots, \lceil t/r \rceil$ **do**
  $T \leftarrow T \,\|\, \text{KD.duplex}(\mathit{false}, 0^b)$
return $\mathsf{left}_t(T)$

Application to Ascon-PRF [DEMS24]



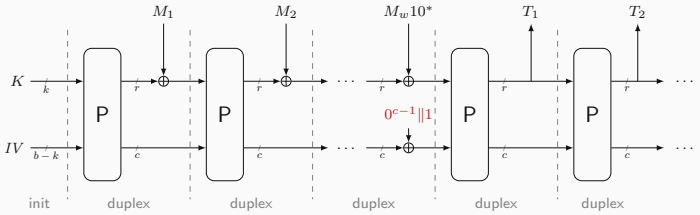
- Input: key K, initial value IV, message M
- Output: tag T

- Domain separation solves problem of repeated paths

Algorithm Ascon-PRF[P]

Input: $(K, IV, M) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
$(M_1, M_2, \ldots, M_w) \leftarrow \mathrm{pad}_r^{10^*}(M)$
$T \leftarrow \varnothing$
KD.init$(1, IV)$
for $i = 1, \ldots, w-1$ **do**
  KD.duplex(false, M_i) ▷ discard output
KD.duplex$(\mathit{false}, M_w\|0^{c-1}1)$
for $i = 1, \ldots, \lceil t/r \rceil$ **do**
  $T \leftarrow T \,\|\, \text{KD.duplex}(\mathit{false}, 0^b)$
return $\mathrm{left}_t(T)$

Application to Ascon-PRF [DEMS24]



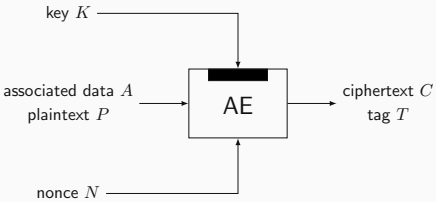
- Input: key K, initial value IV, message M
- Output: tag T

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

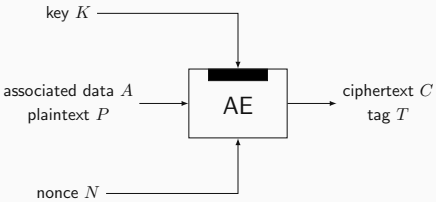
Algorithm Ascon-PRF[P]

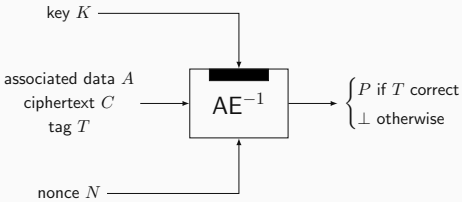
Input: $(K, IV, M) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
$(M_1, M_2, \ldots, M_w) \leftarrow \mathrm{pad}_r^{10^*}(M)$
$T \leftarrow \varnothing$
KD.init$(1, IV)$
for $i = 1, \ldots, w-1$ **do**
  KD.duplex(false, M_i) ▷ discard output
KD.duplex$(\mathit{false}, M_w\|0^{c-1}1)$
for $i = 1, \ldots, \lceil t/r \rceil$ **do**
  $T \leftarrow T \,\|\, \text{KD.duplex}(\mathit{false}, 0^b)$
return $\mathrm{left}_t(T)$

Application to AE: Model

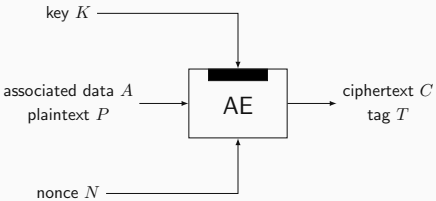


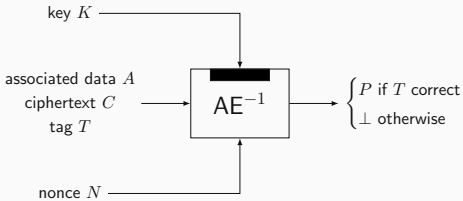
Application to AE: Model





Application to AE: Model

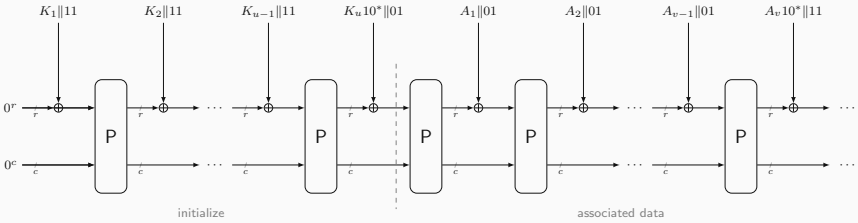


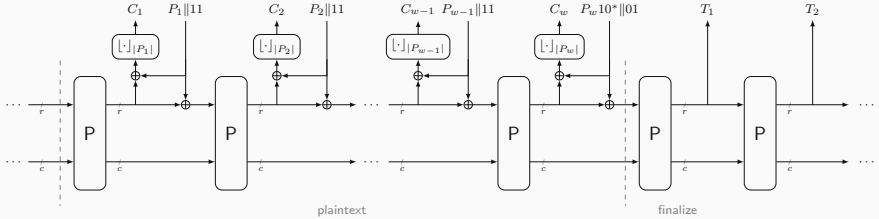


Role of Duplex

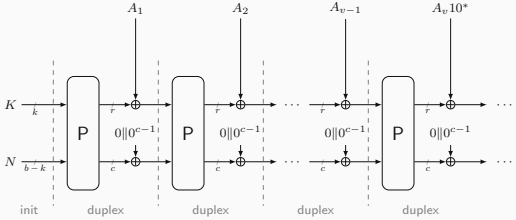
- Blockwise construction allows for processing different types of in-/output
- Usage of flag makes Duplex-style encryption decryptable

Application to AE: SpongeWrap [BDPV11a]

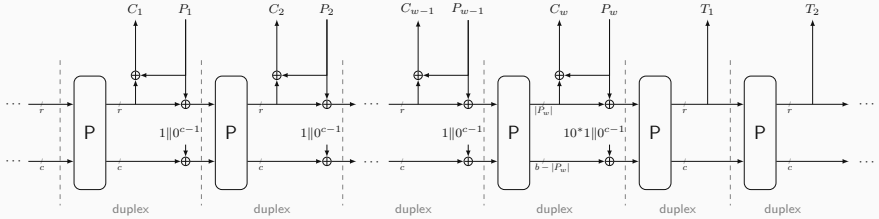




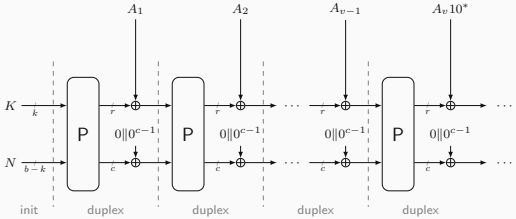
Application to AE: MonkeySpongeWrap: Encryption [Men23]



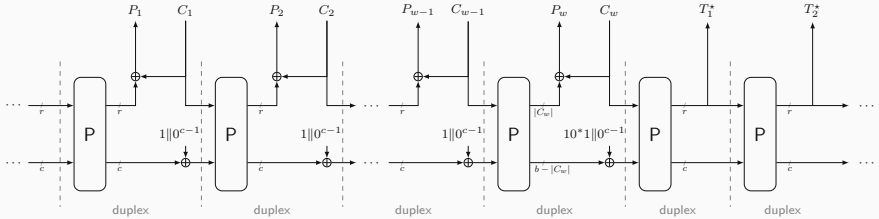
- Evolved version of SpongeWrap [BDPV11a]
  - State initialized using key and nonce
  - No block-wise padding
  - Domain separation spilled over



Application to AE: MonkeySpongeWrap: Decryption [Men23]



- Decryption similar to encryption
- Notable difference:
  - Processing of C
  - Duplexing with $flag = true$
- Impacts security bound



Application to AE: MonkeySpongeWrap: Security [Men23]

- Can be described using Duplex

Algorithm MonkeySpongeWrap[P]: ENC

Input: $(K, N, A, P) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \{0,1\}^* \times \{0,1\}^*$
Output: $(C, T) \in \{0,1\}^{|P|} \times \{0,1\}^t$

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

Algorithm MonkeySpongeWrap[P]: DEC

Input: $(K, N, A, C, T) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^t$
Output: $P \in \{0,1\}^{|C|}$ or $\perp$

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

Application to AE: MonkeySpongeWrap: Security [Men23]

- Can be described using Duplex

Algorithm MonkeySpongeWrap[P]: ENC

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

Algorithm MonkeySpongeWrap[P]: DEC

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

- Security bound for MonkeySpongeWrap immediately follows:

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MonkeySpongeWrap}}(\mathcal{D}) \leq \frac{2\nu_{r,c}^{2\mathcal{M}}(\mathcal{N}+1)}{2^c} + \frac{\mathcal{M}_d\mathcal{N} + \binom{\mathcal{M}_d}{2}}{2^c} + \frac{(\mathcal{M}-\mathcal{Q})\mathcal{Q}}{2^b - \mathcal{Q}} + \frac{2\binom{\mathcal{M}}{2}}{2^b} + \frac{\mathcal{Q}(\mathcal{M}-\mathcal{Q})}{2^{\min\{c+k,b\}}} + \frac{\mathcal{N}}{2^k} + \frac{\mathcal{Q}_d}{2^t}$$

Conclusion

Conclusion

Disclaimer: Results Only Hold in Random Permutation Model

Conclusion

Disclaimer: Results Only Hold in Random Permutation Model

- May hide security issues or give false sense of security
- Permutation needs to be sufficiently strong to be used in the proven modes

Conclusion

Disclaimer: Results Only Hold in Random Permutation Model

- May hide security issues or give false sense of security
- Permutation needs to be sufficiently strong to be used in the proven modes
- Still, generic security results have meaning and impact

Conclusion

Disclaimer: Results Only Hold in Random Permutation Model

- May hide security issues or give false sense of security
- Permutation needs to be sufficiently strong to be used in the proven modes
- Still, generic security results have meaning and impact

Generic Security of Permutation-Based Modes

- Lots of research activity... and I did not even cover everything:
  - Parallel modes such as Minalpher [STA+15], Motorist [BDP+15], and Elephant [BCDM20]
  - Sum of public permutations such as SoEM [CLM19] and sirX [Men25]
  - Post-quantum security [CHS19, Hos25, LLMT25, ACMT25]
  - ...

Conclusion

Disclaimer: Results Only Hold in Random Permutation Model

- May hide security issues or give false sense of security
- Permutation needs to be sufficiently strong to be used in the proven modes
- Still, generic security results have meaning and impact

Generic Security of Permutation-Based Modes

- Lots of research activity... and I did not even cover everything:
  - Parallel modes such as Minalpher [STA+15], Motorist [BDP+15], and Elephant [BCDM20]
  - Sum of public permutations such as SoEM [CLM19] and sirX [Men25]
  - Post-quantum security [CHS19, Hos25, LLMT25, ACMT25]
  - ...

Thank you!

References i

Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Florian Mendel, Bart Mennink, Nicky Mouha, Qingju Wang, and Kan Yasuda.
PRIMATEs v1, 2014.
Submission to CAESAR competition.

Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda.
APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography.
In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*, pages 168–186. Springer, 2014.

References ii


Gorjan Alagic, Joseph Carolan, Christian Majenz, and Saliha Tokat.
The Sponge Is Quantum Indifferentiable.
In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2025, Sydney, Australia, December 14-17, 2025*, pages 2591–2630. IEEE, 2025.


Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche.
Security of Keyed Sponge Constructions Using a Modular Proof Approach.
In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 364–384. Springer, 2015.


Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves.
NORX v1.
Submission to CAESAR competition, 2014.

References iii


Jean-Philippe Aumasson, Dmitry Khovratovich, Bart Mennink, and Porçu Quine.
SAFE: Sponge API for Field Elements.
Cryptology ePrint Archive, Paper 2023/522, 2023.


Elena Andreeva, Bart Mennink, and Bart Preneel.
Security Reductions of the Second Round SHA-3 Candidates.
In Mike Burmester, Gene Tsudik, Spyros S. Magliveras, and Ivana Ilic, editors, *Information Security - 13th International Conference, ISC 2010, Boca Raton, FL, USA, October 25-28, 2010, Revised Selected Papers*, volume 6531 of *Lecture Notes in Computer Science*, pages 39–53. Springer, 2010.


Elena Andreeva, Bart Mennink, and Bart Preneel.
The parazoa family: generalizing the sponge hash functions.
Int. J. Inf. Sec., 11(3):149–165, 2012.

References iv


Ritam Bhaumik, Bishwajit Chakraborty, and Chandranan Dhar.
Breaking and Fixing MacaKey.
IACR Trans. Symmetric Cryptol., 2026(1):76–94, 2026.


Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink.
Dumbo, Jumbo, and Delirium: Parallel Authenticated Encryption for the Lightweight Circus.
IACR Trans. Symmetric Cryptol., 2020(S1):5–30, 2020.


Guido Bertoni, Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer.
Farfalle: parallel permutation-based cryptography.
IACR Trans. Symmetric Cryptol., 2017(4):1–38, 2017.


Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer.
Keyak v2, 2015.
Submission to CAESAR competition.


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, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.


Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
Sponge-Based Pseudo-Random Number Generators.
In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2010.


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, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*, 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.


Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
Sufficient conditions for sound tree and sequential hashing modes.
Int. J. Inf. Sec., 13(4):335–353, 2014.


Henk Berendsen and Bart Mennink.
Tightening Leakage Resilience of the Suffix Keyed Sponge.
IACR Trans. Symmetric Cryptol., 2024(1):459–496, 2024.


Rishiraj Bhattacharyya, Avradip Mandal, and Mridul Nandi.
Security Analysis of the Mode of JH Hash Function.
In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers*, volume 6147 of *Lecture Notes in Computer Science*, pages 168–191. Springer, 2010.


Mihir Bellare and Phillip Rogaway.
Random Oracles are Practical: A Paradigm for Designing Efficient Protocols.
In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, pages 62–73. ACM, 1993.


Donghoon Chang, Morris Dworkin, Seokhie Hong, John Kelsey, and Mridul Nandi.
A Keyed Sponge Construction with Pseudorandomness in the Standard Model.
NIST SHA–3 Workshop, March 2012.


Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi.
Exact Security Analysis of ASCON.
In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part III*, volume 14440 of *Lecture Notes in Computer Science*, pages 346–369. Springer, 2023.


Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi.
Tight Multi-user Security of Ascon and Its Large Key Extension.
In Tianqing Zhu and Yannan Li, editors, *Information Security and Privacy - 29th Australasian Conference, ACISP 2024, Sydney, NSW, Australia, July 15-17, 2024, Proceedings, Part I*, volume 14895 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2024.

References x


Avik Chakraborti, Nilanjan Datta, Mridul Nandi, and Kan Yasuda.
Beetle Family of Lightweight and Secure Authenticated Encryption Ciphers.
IACR Trans. Cryptogr. Hardw. Embed. Syst., 2018(2):218–241, 2018.


Jan Czajkowski, Andreas Hülsing, and Christian Schaffner.
Quantum Indistinguishability of Random Sponges.
In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 296–325. Springer, 2019.


Bishwajit Chakraborty, Ashwin Jha, and Mridul Nandi.
On the Security of Sponge-type Authenticated Encryption Modes.
IACR Trans. Symmetric Cryptol., 2020(2):93–119, 2020.


Yu Long Chen, Eran Lambooij, and Bart Mennink.
How to Build Pseudorandom Functions from Public Random Permutations.
In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, Lecture Notes in Computer Science, pages 266–293. Springer, 2019.


Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer.
Ascon v1.
Submission to CAESAR competition, 2014.


Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer.
Ascon MAC, PRF, and Short-Input PRF - Lightweight, Fast, and Efficient Pseudorandom Functions.
In Elisabeth Oswald, editor, *Topics in Cryptology - CT-RSA 2024 - Cryptographers' Track at the RSA Conference 2024, San Francisco, CA, USA, May 6-9, 2024, Proceedings*, volume 14643 of *Lecture Notes in Computer Science*, pages 381–403. Springer, 2024.


Joan Daemen, Seth Hoffert, Silvia Mella, Gilles Van Assche, and Ronny Van Keer.
Shaking up authenticated encryption.
In *10th IEEE European Symposium on Security and Privacy, EuroS&P 2025, Venice, Italy, June 30 - July 4, 2025*, pages 861–882. IEEE, 2025.


Christoph Dobraunig and Bart Mennink.
Leakage Resilience of the Duplex Construction.
In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 225–255. Springer, 2019.


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


Christoph Dobraunig and Bart Mennink.
Tightness of the Suffix Keyed Sponge Bound.
IACR Trans. Symmetric Cryptol., 2020(4):195–212, 2020.


Christoph Dobraunig and Bart Mennink.
Generalized Initialization of the Duplex Construction.
In Christina Pöpper and Lejla Batina, editors, *Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part II*, volume 14584 of *Lecture Notes in Computer Science*, pages 460–484. Springer, 2024.


Christoph Dobraunig, Bart Mennink, and Robert Primas.
Leakage and Tamper Resilient Permutation-Based Cryptography.
In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 859–873. ACM, 2022.


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, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 606–637. Springer, 2017.


Joan Daemen, Bart Mennink, and Gilles Van Assche.
Sound Hashing Modes of Arbitrary Functions, Permutations, and Block Ciphers.
IACR Trans. Symmetric Cryptol., 2018(4):197–228, 2018.


Robin Foekens.
Security of the Sponge Construction with a Random Transformation.
Bachelor's thesis, Radboud University, Nijmegen, 2023.


Aldo Gunsing, Joan Daemen, and Bart Mennink.
Errata to Sound Hashing Modes of Arbitrary Functions, Permutations, and Block Ciphers.
IACR Trans. Symmetric Cryptol., 2020(3):362–366, 2020.


Chun Guo, Kai Hu, Shuntian Jiang, Yanhong Fan, Yong Fu, Bart Preneel, and Meiqin Wang.
Permutation-Based Hash from Non-Idealized Assumptions: Adding Feed-Forward to Sponge.
Cryptology ePrint Archive, Report 2025/1006, 2025.
http://eprint.iacr.org/2025/1006.


Aldo Gunsing and Bart Mennink.
The Summation-Truncation Hybrid: Reusing Discarded Bits for Free.
In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 187–217. Springer, 2020.


Jian Guo, Thomas Peyrin, and Axel Poschmann.
The PHOTON Family of Lightweight Hash Functions.
In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 222–239. Springer, 2011.


Peter Gazi, 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, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 368–387. Springer, 2015.


Peter Gazi and Stefano Tessaro.
Provably Robust Sponge-Based PRNGs and KDFs.
In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 87–116. Springer, 2016.


Akinori Hosoyamada.
Post-Quantum Security of Keyed Sponge-Based Constructions Through a Modular Approach.
In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part I*, volume 16245 of *Lecture Notes in Computer Science*, pages 411–445. Springer, 2025.


Daniel Hutchinson.
A Robust and Sponge-Like PRNG with Improved Efficiency.
In Roberto Avanzi and Howard M. Heys, editors, *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, volume 10532 of *Lecture Notes in Computer Science*, pages 381–398. Springer, 2016.

References xx

Philipp Jovanovic, Atul Luykx, and Bart Mennink.
Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes.
In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 85–104. Springer, 2014.

Dmitry Khovratovich, Mario Marhuenda Beltrán, and Bart Mennink.
Generic Security of the SAFE API and Its Applications.
In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part VIII*, volume 14445 of *Lecture Notes in Computer Science*, pages 301–327. Springer, 2023.


Charlotte Lefevre.
Indifferentiability of the Sponge Construction with a Restricted Number of Message Blocks.
IACR Trans. Symmetric Cryptol., 2023(1):224–243, 2023.


Nathalie Lang, Stefan Lucks, Bart Mennink, and Suprita Talnikar.
Security of the Ascon Authenticated Encryption Mode in the Presence of Quantum Adversaries.
Cryptology ePrint Archive, Report 2025/411, 2025.
`http://eprint.iacr.org/2025/411`.


Charlotte Lefevre and Bart Mennink.
Tight Preimage Resistance of the Sponge Construction.
In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 185–204. Springer, 2022.


Charlotte Lefevre and Bart Mennink.
Generic Security of the Ascon Mode: On the Power of Key Blinding.
In Maria Eichlseder and Sébastien Gambs, editors, *Selected Areas in Cryptography - SAC 2024 - 31st International Conference, Montreal, QC, Canada, August 28-30, 2024, Revised Selected Papers, Part II*, volume 15517 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2024.


Charlotte Lefevre and Bart Mennink.
Permutation-Based Hashing Beyond the Birthday Bound.
IACR Trans. Symmetric Cryptol., 2024(1):71–113, 2024.


Charlotte Lefevre and Mario Marhuenda Beltrán.
MacaKey: Full-State Keyed Sponge Meets the Summation-Truncation Hybrid.
Cryptology ePrint Archive, Report 2025/963, 2025.
`http://eprint.iacr.org/2025/893`.


Charlotte Lefevre and Bart Mennink.
IACR Trans. Symmetric Cryptol., 2025(1):138–210, 2025.


Charlotte Lefevre, Mario Marhuenda Beltrán, and Bart Mennink.
To Pad or Not to Pad? Padding-Free Arithmetization-Oriented Sponges.
IACR Trans. Symmetric Cryptol., 2025(1):97–137, 2025.


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


Bart Mennink.
Understanding the Duplex and Its Security.
IACR Trans. Symmetric Cryptol., 2023(2):1–46, 2023.


Bart Mennink.
Minimized PRFs from Public Permutations.
IACR Trans. Symmetric Cryptol., 2025(3):230–260, 2025.


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, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 465–489. Springer, 2015.


National Institute of Standards and Technology.
FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, August 2015.


Yusuke Naito and Kazuo Ohta.
Improved Indifferentiable Security Analysis of PHOTON.
In Michel Abdalla and Roberto De Prisco, editors, *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, volume 8642 of *Lecture Notes in Computer Science*, pages 340–357. Springer, 2014.


Yusuke Naito and Kan Yasuda.
New Bounds for Keyed Sponges with Extendable Output: Independence Between Capacity and Message Length.
In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2016.


Siwei Sun, Shun Li, Zhiyu Zhang, Charlotte Lefevre, Bart Mennink, Zhen Qin, and Dengguo Feng.
Permutation-Based Hashing With Stronger (Second) Preimage Resistance.
Cryptology ePrint Archive, Report 2025/963, 2025.
http://eprint.iacr.org/2025/963.


Meltem Sönmez Turan, Kerry A. McKay, Jinkeon Kang, John Kelsey, and Donghoon Chang.
Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions.
NIST SP 800-232, August 2025.
https://csrc.nist.gov/pubs/sp/800/232/final.

Yu Sasaki, Yosuke Todo, Kazumaro Aoki, Yusuke Naito, Takeshi Sugawara, Yumiko Murakami, Mitsuru Matsui, and Shoichi Hirose.
Minalpher v1.1.
Submission to the CAESAR competition, 2015.

Yu Sasaki and Kan Yasuda.
How to Incorporate Associated Data in Sponge-Based Authenticated Encryption.
In Kaisa Nyberg, editor, *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, volume 9048 of *Lecture Notes in Computer Science*, pages 353–370. Springer, 2015.

Hongjun Wu.
The Hash Function JH, 2009.
Submission to NIST's SHA-3 competition.