Radbound University

# Keyed Sponges

## Applied Cryptography – Spring 2024

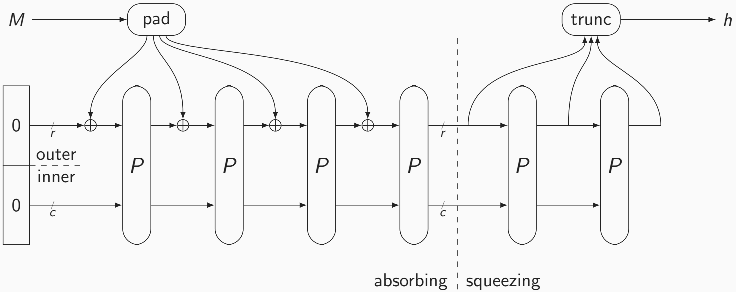
Bart Mennink

March 4, 2024

Institute for Computing and Information Sciences
Radboud University

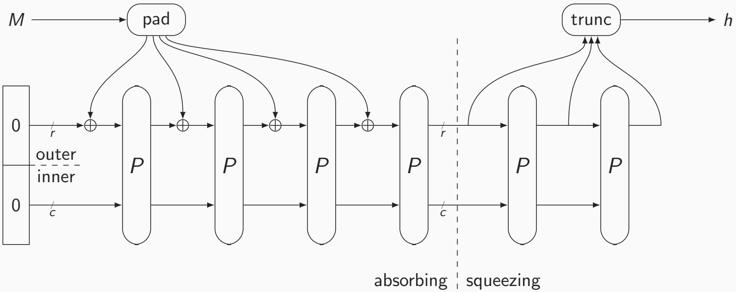
# Last Lectures

- Permutations appear to be powerful primitives in cryptography
- Most importantly, sponge hash functions



# Last Lectures

- Permutations appear to be powerful primitives in cryptography
- Most importantly, sponge hash functions




- This lecture will be fully devoted to keyed versions of the sponge
    - Stream encryption
    - Authentication
    - ...

# Outline

Keying Sponges

Authenticated Encryption Using the Duplex

Provable Security of Full-Keyed Sponge Construction

Suffix Keyed Sponge

Conclusion

# Keying Sponges

# Keying Sponges

## Keyed Sponge

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

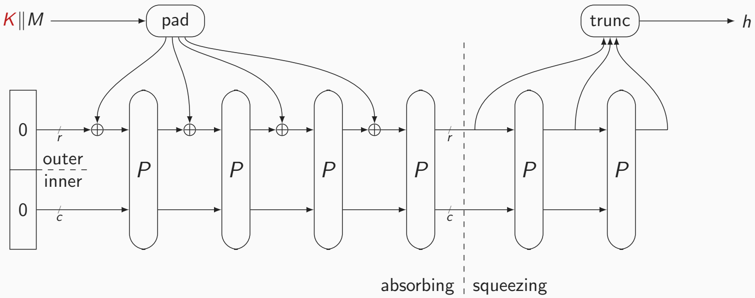
## Keyed Sponge

- $\mathrm{PRF}(K, M) = \mathrm{Sponge}(K \parallel M)$
- Message authentication with tag size $t$: $\mathrm{MAC}(K, M, t) = \mathrm{Sponge}(K \parallel M, t)$
- Keystream generation of length $\ell$: $\mathrm{Stream}(K, D, \ell) = \mathrm{Sponge}(K \parallel D, \ell)$
- (All assuming $K$ is fixed-length)

## Keyed Duplex

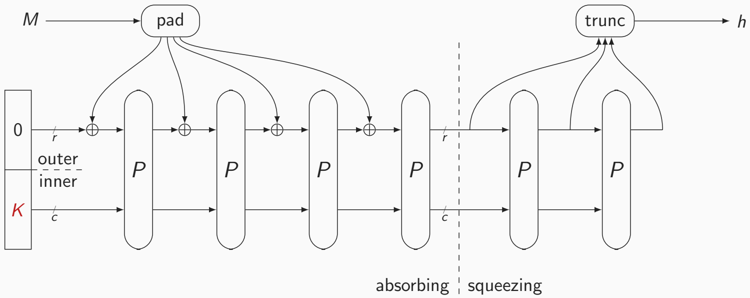
- Authenticated encryption
- Multiple CAESAR and NIST LWC submissions

# Evolution of Keyed Sponges



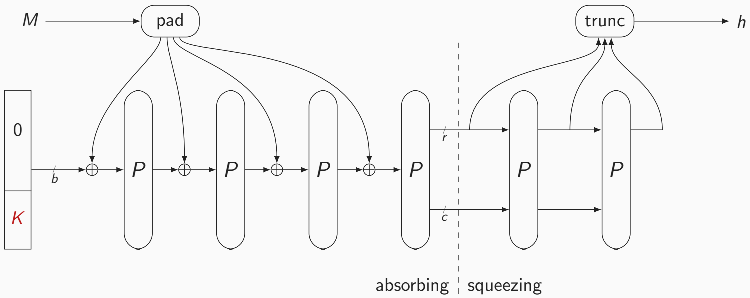

- Outer-Keyed Sponge [BDPV11,ADMV15,NY16,Men18]

# Evolution of Keyed Sponges



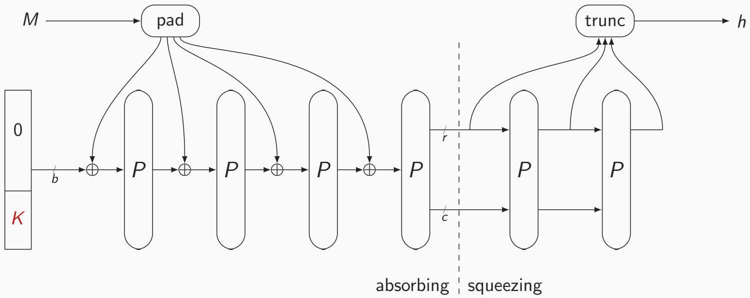

- Outer-Keyed Sponge [BDPV11,ADMV15,NY16,Men18]
- Inner-Keyed Sponge [CDHKN12,ADMV15,NY16]

# Evolution of Keyed Sponges



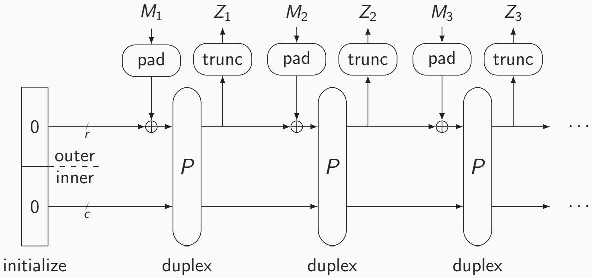

- Outer-Keyed Sponge [BDPV11,ADMV15,NY16,Men18]
- Inner-Keyed Sponge [CDHKN12,ADMV15,NY16]
- Full-Keyed Sponge [BDPV12,GPT15,MRV15]

# Evolution of Keyed Sponges



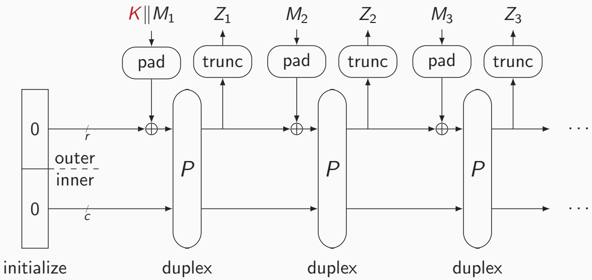

- Outer-Keyed Sponge [BDPV11,ADMV15,NY16,Men18]
- Inner-Keyed Sponge [CDHKN12,ADMV15,NY16]
- Full-Keyed Sponge [BDPV12,GPT15,MRV15]
- Generic security does not degrade: all can be used for PRF or MAC

# Evolution of Keyed Duplexes



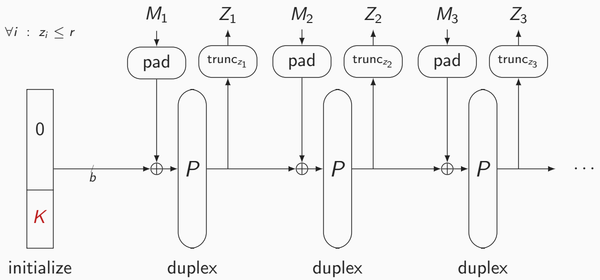

- Unkeyed Duplex [BDPV11]

## Evolution of Keyed Duplexes



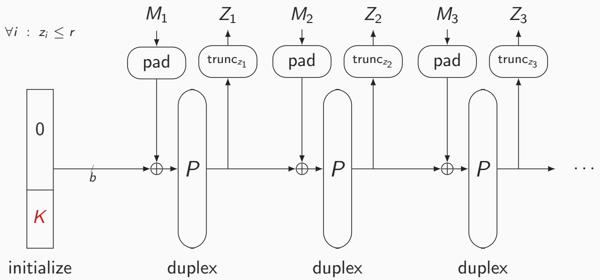

- Unkeyed Duplex [BDPV11]
- Outer-Keyed Duplex [BDPV11]

# Evolution of Keyed Duplexes



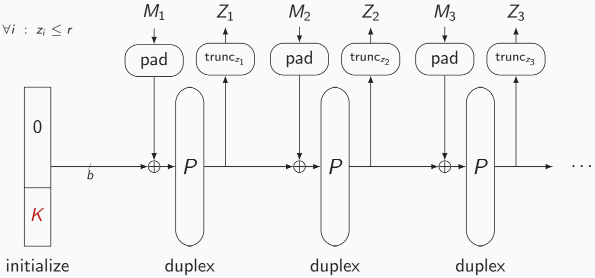

- Unkeyed Duplex [BDPV11]
- Outer-Keyed Duplex [BDPV11]
- Full-Keyed Duplex [MRV15,DMV17,DM19]

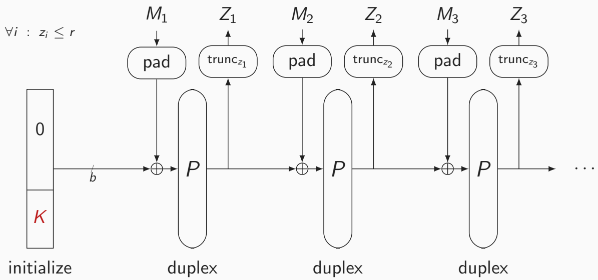
# Evolution of Keyed Duplexes




- Unkeyed Duplex [BDPV11]
- Outer-Keyed Duplex [BDPV11]
- Full-Keyed Duplex [MRV15,DMV17,DM19]
- Generic security does not degrade: both can be used for authenticated encryption

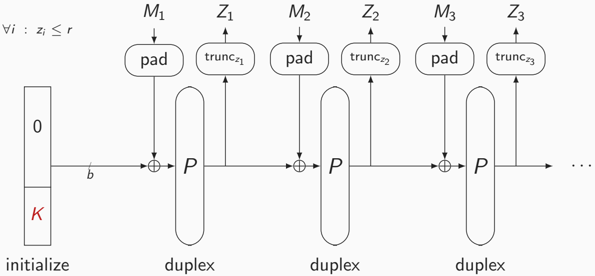
# Security Model for Duplexes (1/2)





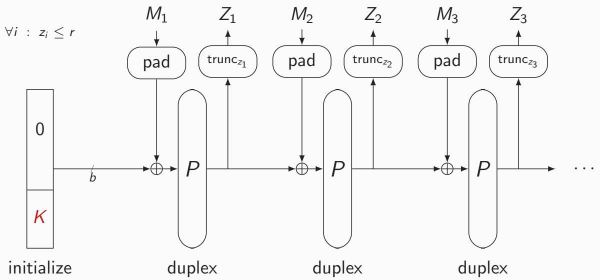

- For each duplex call, we can specify a path
  - In above picture, $\text{path}_1 = M_1$, $\text{path}_2 = M_1 \| M_2$, $\text{path}_3 = M_1 \| M_2 \| M_3$

# Security Model for Duplexes (1/2)

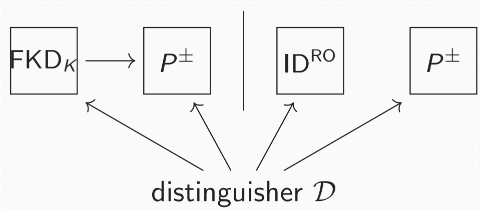



- For each duplex call, we can specify a path
  - In above picture, $\text{path}_1 = M_1$, $\text{path}_2 = M_1 \| M_2$, $\text{path}_3 = M_1 \| M_2 \| M_3$
- Ideally, output is random for each path: $Z_i \overset{?}{\sim} \mathcal{RO}(\text{path}_i, z_i)$

# Security Model for Duplexes (1/2)

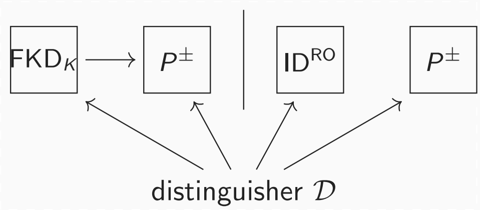



- For each duplex call, we can specify a path
  - In above picture, $\text{path}_1 = M_1$, $\text{path}_2 = M_1 \| M_2$, $\text{path}_3 = M_1 \| M_2 \| M_3$
- Ideally, output is random for each path: $Z_i \overset{?}{\sim} \mathcal{RO}(\text{path}_i, z_i)$
- Call such a function an ideal duplex (ID)




- Two oracles: $\mathrm{FKD}_K$ (for secret key $K$) and $\mathrm{ID}^{\mathrm{RO}}$ (for secret RO)
  - Both with initialize and duplex interface
- Distinguisher $\mathcal{D}$ has query access to one of these, plus the random permutation $P$

# Security Model for Duplexes (2/2)

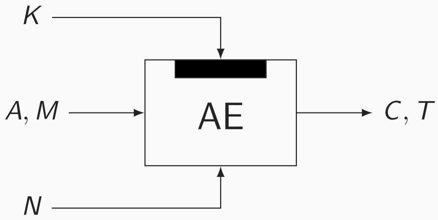



- Two oracles: $\mathsf{FKD}_K$ (for secret key $K$) and $\mathsf{ID}^{\mathsf{RO}}$ (for secret RO)
  - Both with initialize and duplex interface
- Distinguisher $\mathcal{D}$ has query access to one of these, plus the random permutation $P$
- $\mathcal{D}$ tries to determine which oracle it communicates with
- Its advantage is defined as:

$$\mathbf{Adv}^{\mathrm{duplex}}_{\mathsf{FKD}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\mathsf{FKD}_K, P^{\pm}\ ;\ \mathsf{ID}^{\mathsf{RO}}, P^{\pm}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{FKD}_K, P^{\pm}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{ID}^{\mathsf{RO}}, P^{\pm}} = 1\right)\right|$$

# Authenticated Encryption Using the Duplex

# Recap: Authenticated Encryption



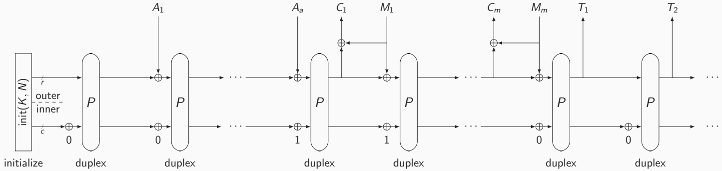

- Using key $K$:
  - Message $M$ is encrypted in ciphertext $C$
  - Associated data $A$ and message $M$ are authenticated using $T$
- Nonce $N$ randomizes the scheme
- Key, nonce, and tag are typically of fixed size
- Associated data, message, and ciphertext could be arbitrary length

# SpongeWrap [BDPV11]

- Now: slight variant of original design

# SpongeWrap [BDPV11]

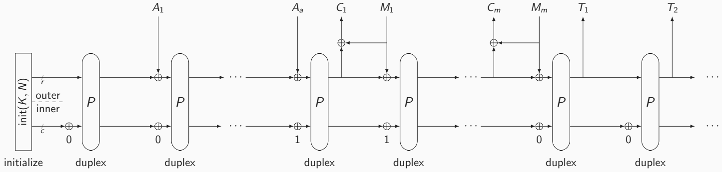
## Authenticated Encryption



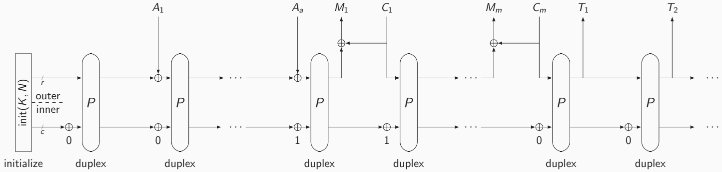

- Now: slight variant of original design
- SpongeWrap embeds generalization of duplex
- Note the domain separation (why?)

# SpongeWrap [BDPV11]

## Authenticated Encryption

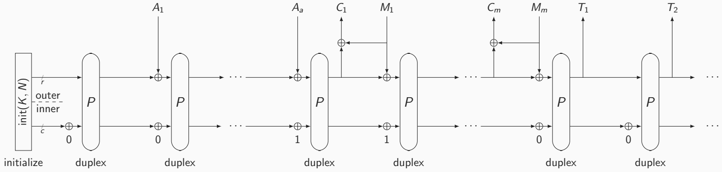



## Authenticated Decryption

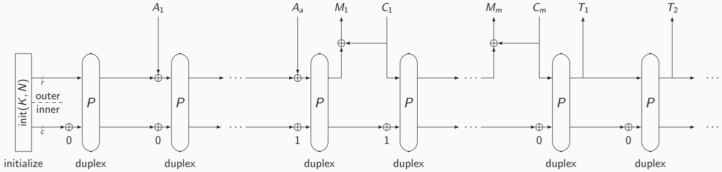



- Now: slight variant of original design
- SpongeWrap embeds generalization of duplex
- Note the domain separation (why?)
- Decryption similar

## Authenticated Encryption



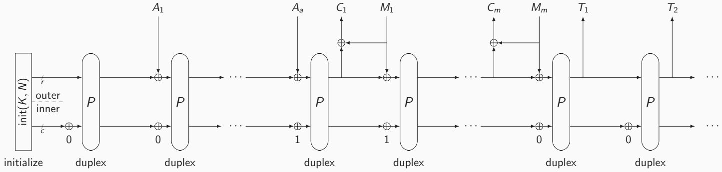

## Authenticated Decryption



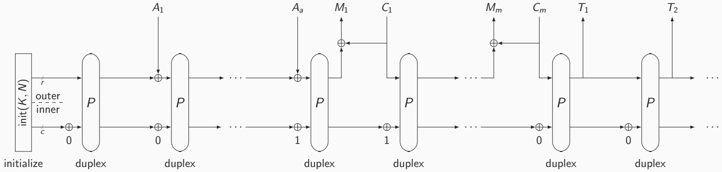

- Now: slight variant of original design
- SpongeWrap embeds generalization of duplex
- Note the domain separation (why?)
- Decryption similar
- Variations:
  - Absorb $A$ full-state or alongside $M$?
  - Intermediate tags?
  - Misuse resistance?

# SpongeWrap [BDPV11]

## Authenticated Encryption

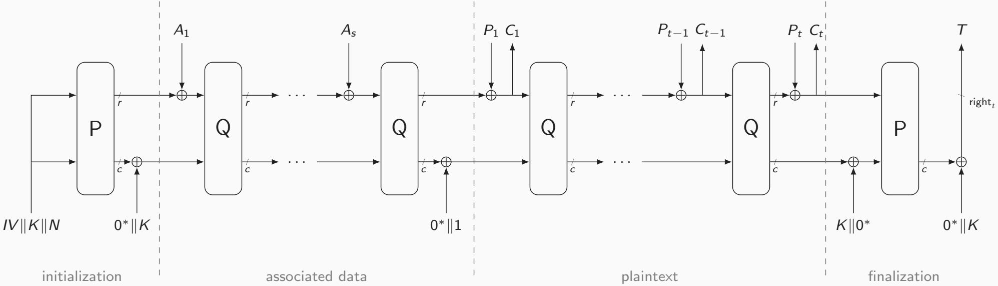



## Authenticated Decryption




- Now: slight variant of original design
- SpongeWrap embeds generalization of duplex
- Note the domain separation (why?)
- Decryption similar
- Variations:
  - Absorb $A$ full-state or alongside $M$?
  - Intermediate tags?
  - Misuse resistance?
- Popular approach (a.o. 3 NIST LWC finalists)

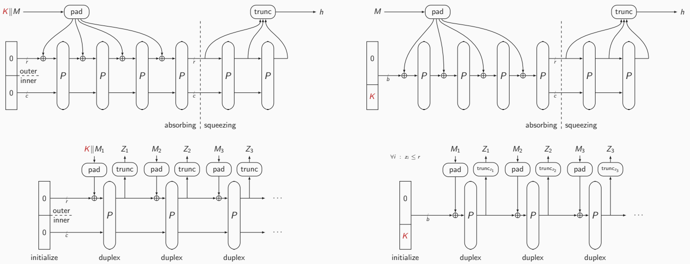
# Ascon Authenticated Encryption




- Winner of the NIST Lightweight Cryptography competition in 2023
- Inspired by SpongeWrap but with some changes:
  - Key blinding: extra robustness against state recovery
  - Different permutations: outer ones are stronger than inner ones

# Provable Security of Full-Keyed Sponge Construction

# Simplified Security Bound

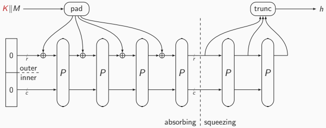
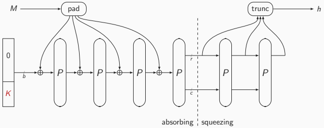
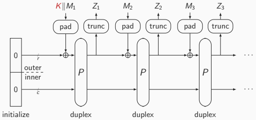
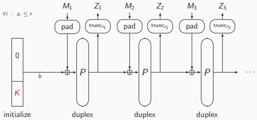



- $F \in \{\mathrm{OKS}, \mathrm{FKS}, \mathrm{OKD}, \mathrm{FKD}\}$
- $M$: data (construction) complexity
- $N$: time (primitive) complexity

**Drastically Simplified Security Bound**

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

# Simplified Security Bound

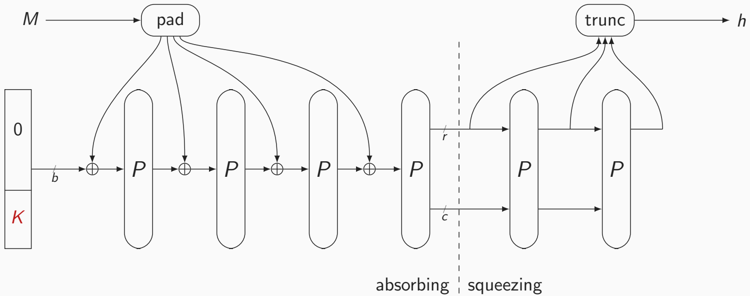
- $F \in \{\mathsf{OKS}, \mathsf{FKS}, \mathsf{OKD}, \mathsf{FKD}\}$
- $M$: data (construction) complexity
- $N$: time (primitive) complexity

**Drastically Simplified Security Bound**

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

**Now: rough idea how security of FKS is argued**

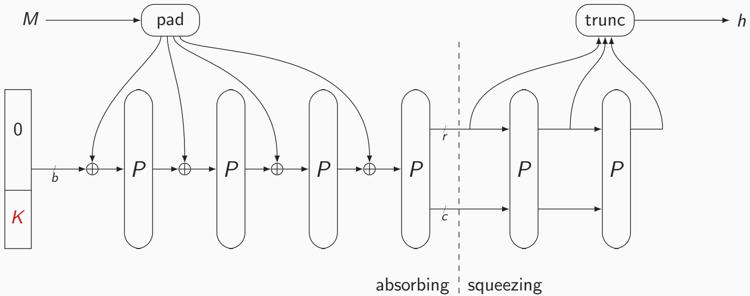
# Full-Keyed Sponge Construction




## Setting

- Assume random $b$-bit permutation $P$
- Key size $k$; rate $r$ and capacity $c$ with $b = r + c$

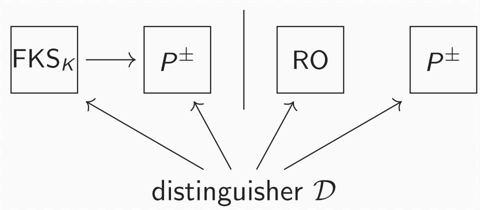
# Full-Keyed Sponge Construction




## Setting

- Assume random $b$-bit permutation $P$
- Key size $k$; rate $r$ and capacity $c$ with $b = r + c$
- FKS should behave like a random oracle

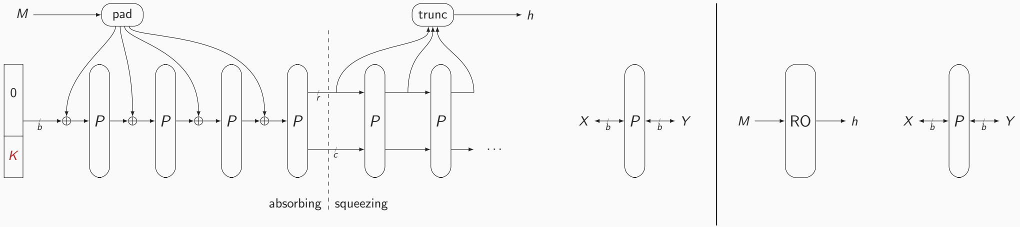
# PRF Security



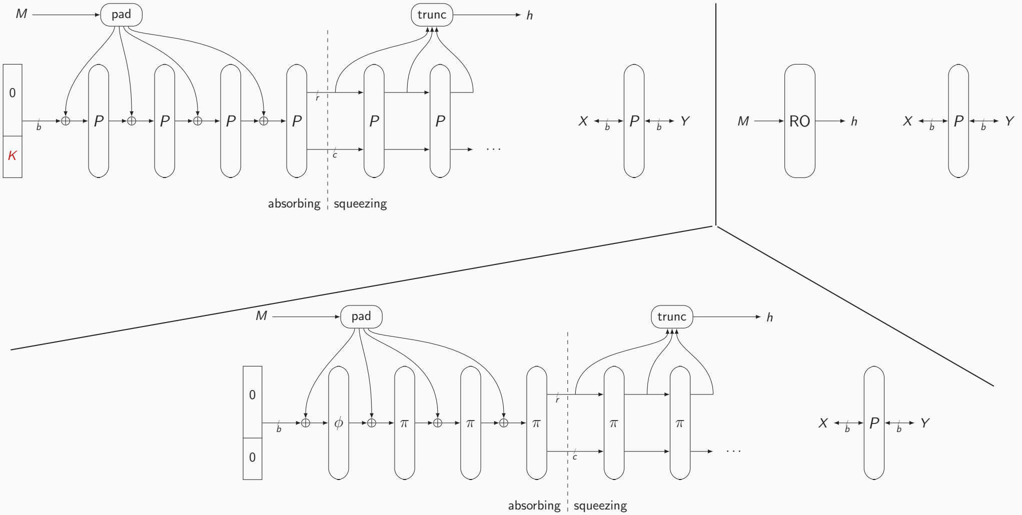

- Two oracles: $\mathsf{FKS}_K$ (for secret key $K$) and RO (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these, plus the random permutation $P$
- $\mathcal{D}$ tries to determine which oracle it communicates with
- Its advantage is defined as:

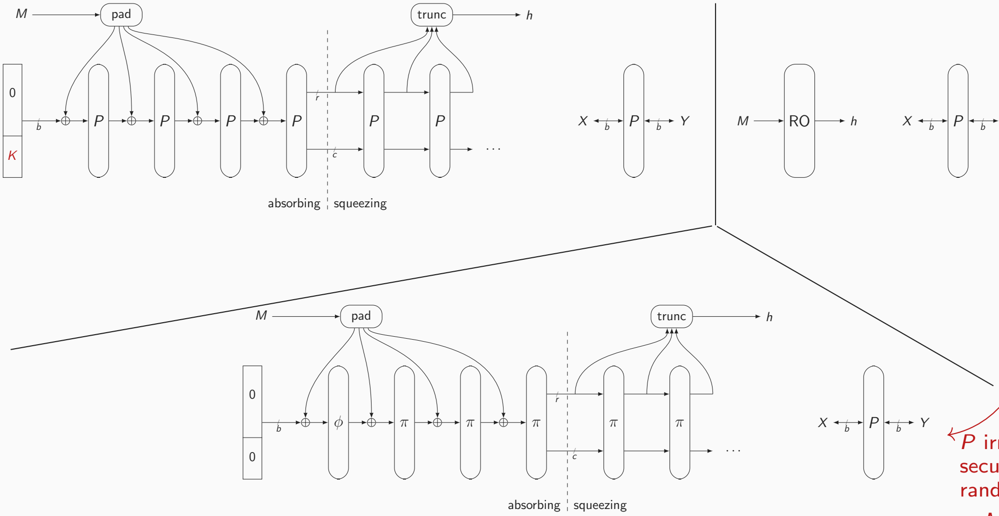
$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{FKS}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\mathsf{FKS}_K, P^{\pm}\ ;\ \mathsf{RO}, P^{\pm}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{FKS}_K, P^{\pm}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{RO}, P^{\pm}} = 1\right)\right|$$

# Hybrid Argument



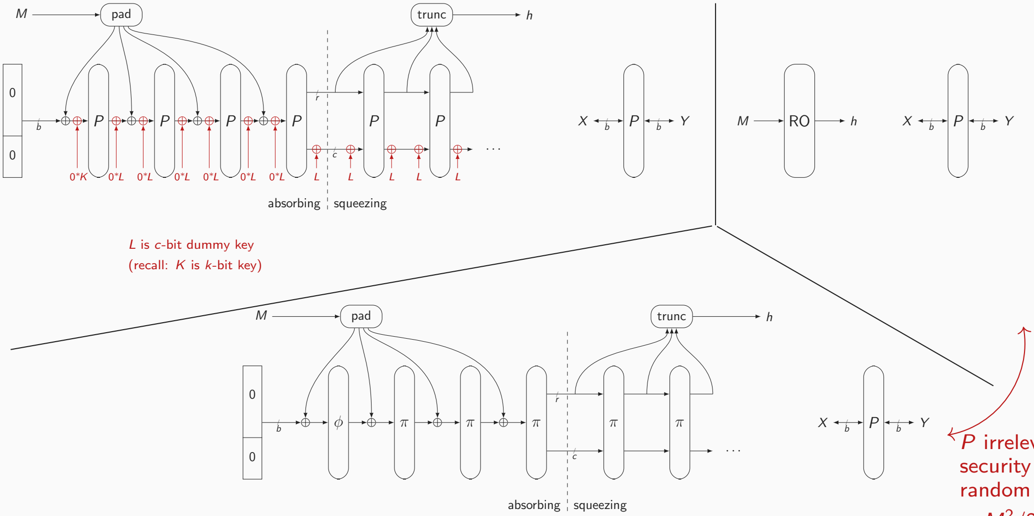
# Hybrid Argument






$P$ irrelevant;
security of
random duplex;
$\approx M^2/2^c$

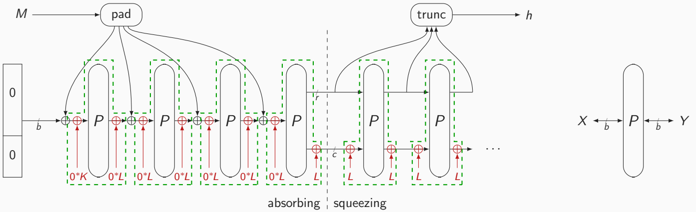
# Hybrid Argument

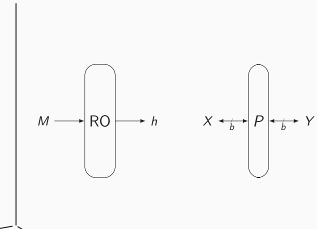



$L$ is $c$-bit dummy key
(recall: $K$ is $k$-bit key)

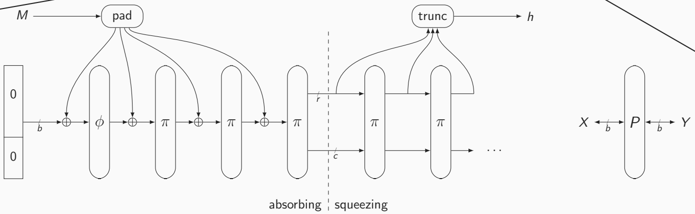
$P$ irrelevant; security of random duplex; $\approx M^2/2^c$

# Hybrid Argument



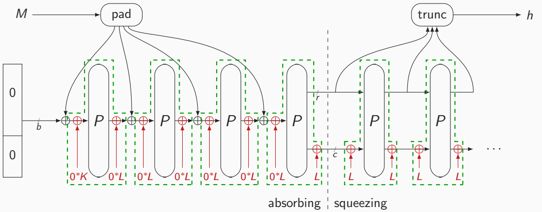



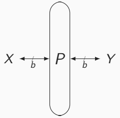

L is c-bit dummy key
(recall: K is k-bit key)

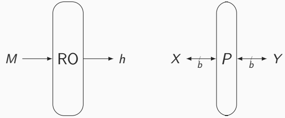



$P$ irrelevant;
security of
random duplex;
$\approx M^2/2^c$

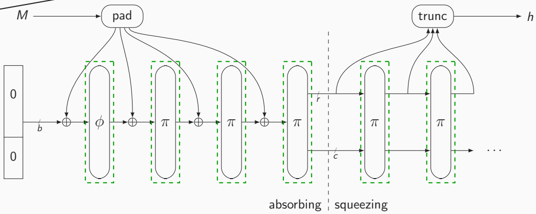
# Hybrid Argument

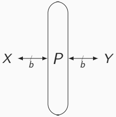



$L$ is $c$-bit dummy key
(recall: $K$ is $k$-bit key)



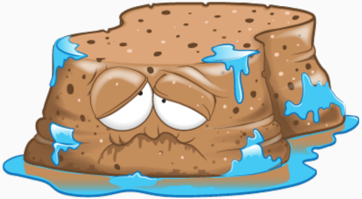



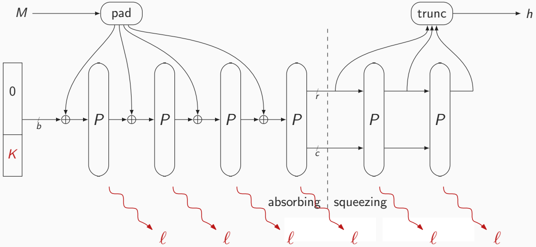

security of two
$P$-based Even-Mansours;
$\approx MN/2^c + N/2^k$

$P$ irrelevant;
security of
random duplex;
$\approx M^2/2^c$

# Suffix Keyed Sponge

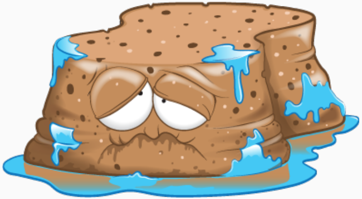
# Leakage Resilience of Keyed Sponges

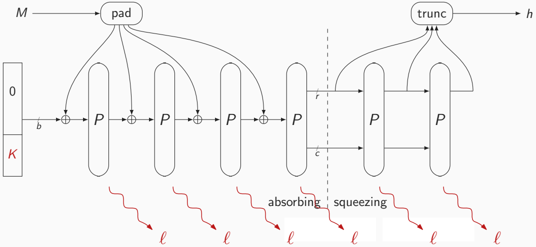





- Permutation $P$ repeatedly evaluated on secret state
- Any evaluation of $P$ may leak information

# Leakage Resilience of Keyed Sponges

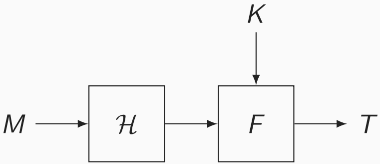





- Permutation $P$ repeatedly evaluated on secret state
- Any evaluation of $P$ may leak information

Minimizing leakage of keyed sponge?

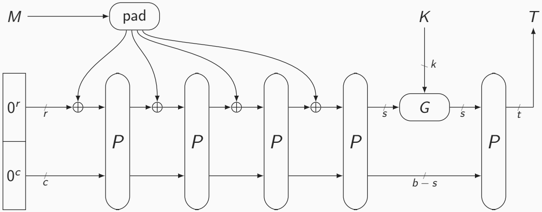
# Hash-then-MAC

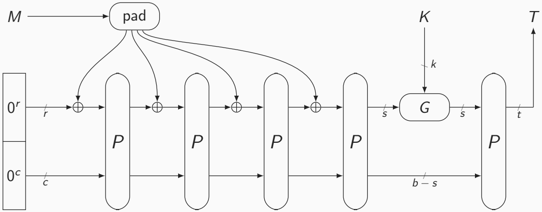



**Typical Approach**

- Hash function is unkeyed $\rightarrow$ nothing to be protected
- Keyed function $F$ applied to fixed-size input
- Hash output (hence $F$ input) must be at least $2k$ bits for $k$-bit security

# Suffix Keyed Sponge

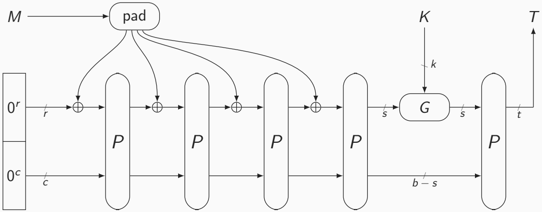





### SuKS versus Full-Keyed Sponge

- No full-state absorption
- Side-channel leakage limited
- $s, t$ arbitrary (typical: $s = t = c/2$)

# Suffix Keyed Sponge




## SuKS versus Full-Keyed Sponge

- No full-state absorption
- Side-channel leakage limited
- $s, t$ arbitrary (typical: $s = t = c/2$)

## SuKS versus Hash-then-MAC

- State of keyed function half as large
- $G$ need not be cryptographically strong (a XOR suffices)
- Single cryptographic primitive needed

# Conclusion

# Overall Conclusion

## Main Take-Away

- Unconditional security impossible!
- Security proofs are for modes and assume a strong building block
- Cryptanalysis, which is about investigating the actual strength of such building blocks, completes the picture
- Both provable security and cryptanalysis are active research areas

# Overall Conclusion

## Main Take-Away

- Unconditional security impossible!
- Security proofs are for modes and assume a strong building block
- Cryptanalysis, which is about investigating the actual strength of such building blocks, completes the picture
- Both provable security and cryptanalysis are active research areas

## Concluding Remarks

- This last lecture concludes the symmetric cryptography part
- If you have any question on the lectures, or symmetric cryptography in general, you are always free to contact me