Introduction to Provable Security

Bart Mennink
KU Leuven (Belgium)

IACR School on Design and Security of Cryptographic Algorithms and Devices

October 20, 2015

Introduction

What is Provable Security?

Introduction

What is Provable Security?

absolute security

reductionist security

Introduction

What is Provable Security?

absolute security
(e.g., lower bound on the number of active S-boxes)

reductionist security

Introduction

What is Provable Security?

absolute security
(e.g., lower bound on the number of active S-boxes)

reductionist security

Introduction

What is Provable Security?

absolute security
(e.g., lower bound on the
number of active S-boxes)

reductionist security

based on
crypto primitive

based on
math problem

Introduction

What is Provable Security?

absolute security
(e.g., lower bound on the number of active S-boxes)

reductionist security

based on crypto primitive

based on math problem

Introduction

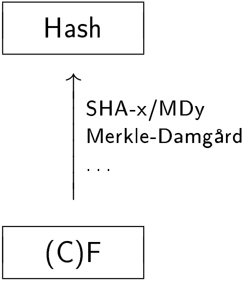
What is Provable Security?

- absolute security
  (e.g., lower bound on the number of active S-boxes)
- reductionist security
  - based on crypto primitive
    - keyed
    - keyless
  - based on math problem

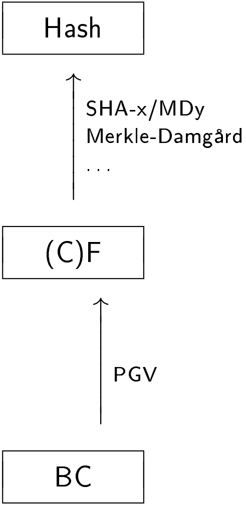
Introduction

$\boxed{\text{Hash}}$

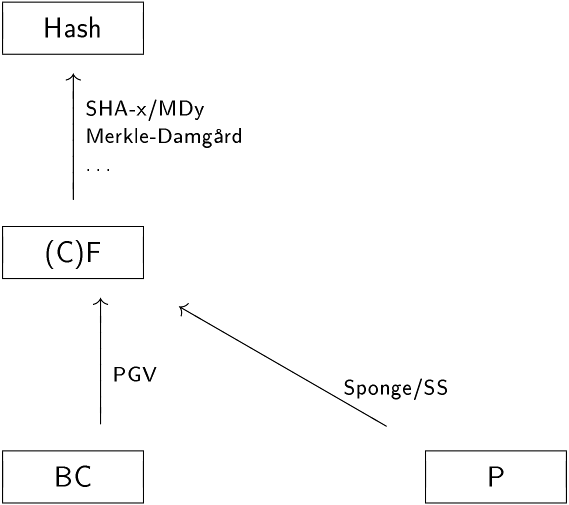
Introduction



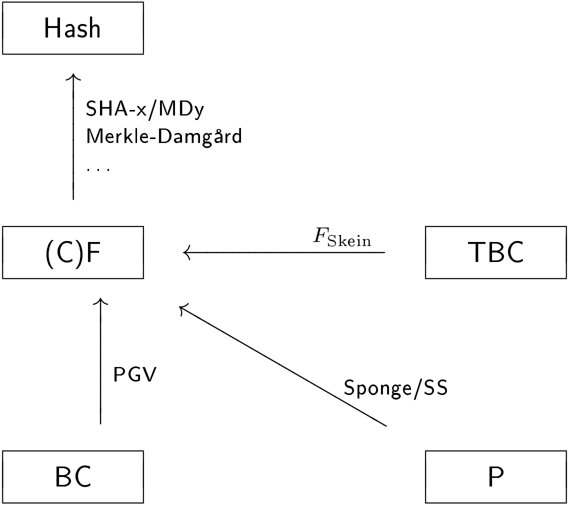
Introduction



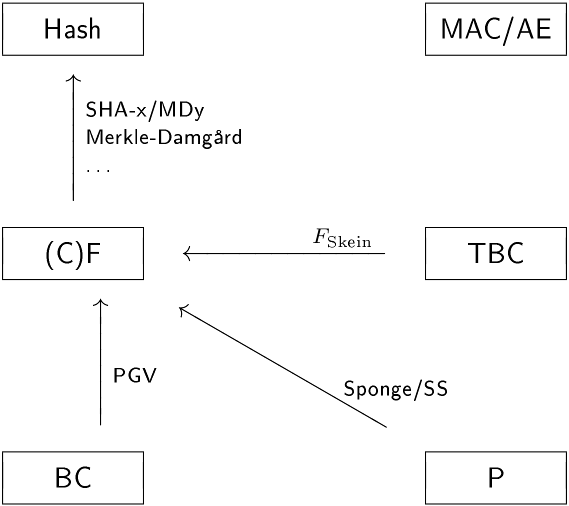
Introduction



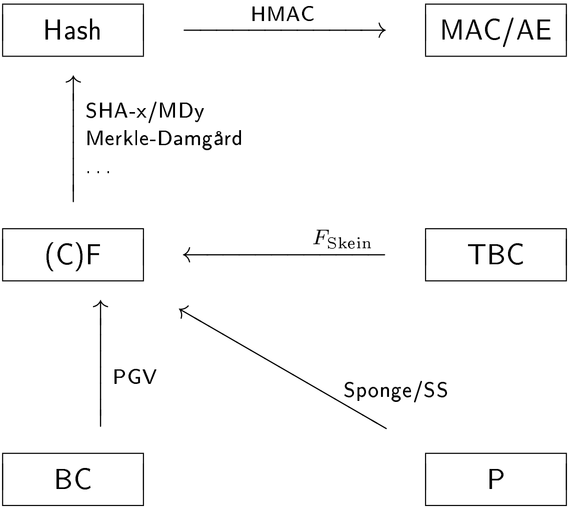
Introduction



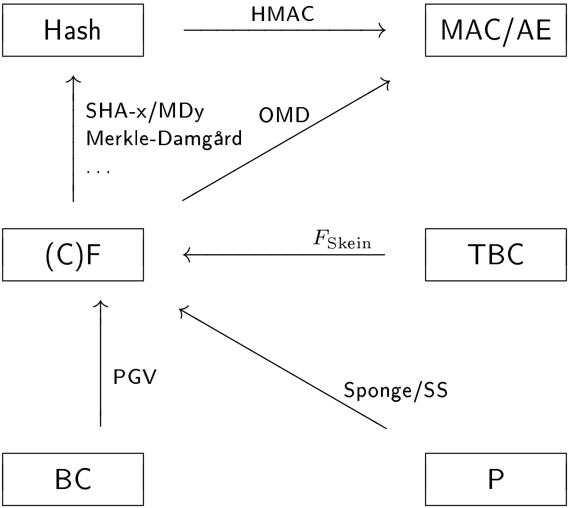
Introduction



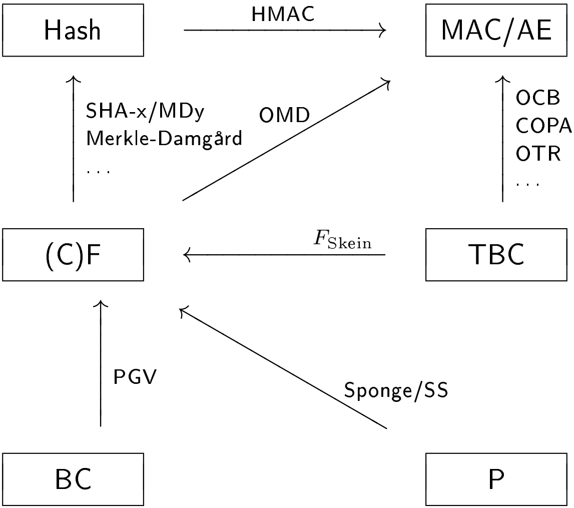
Introduction



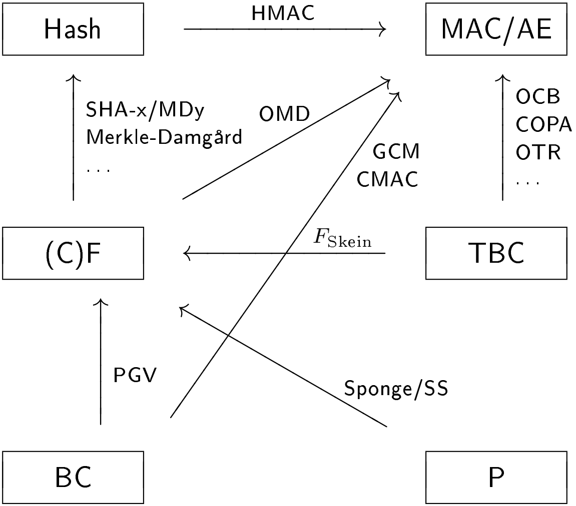
Introduction



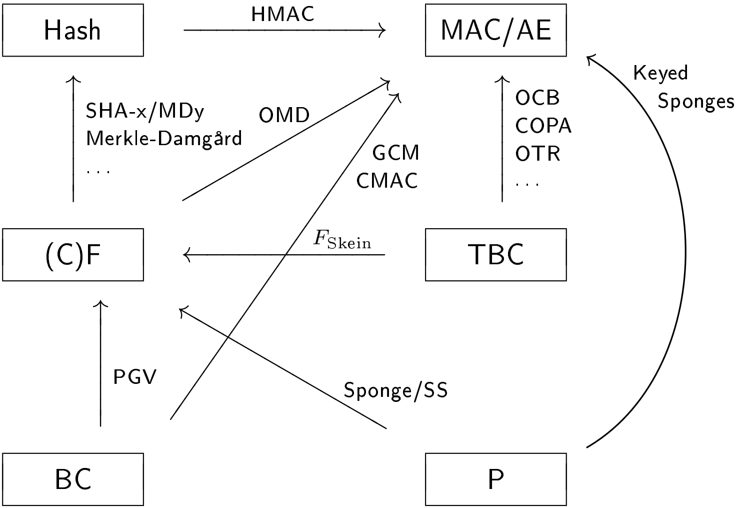
Introduction



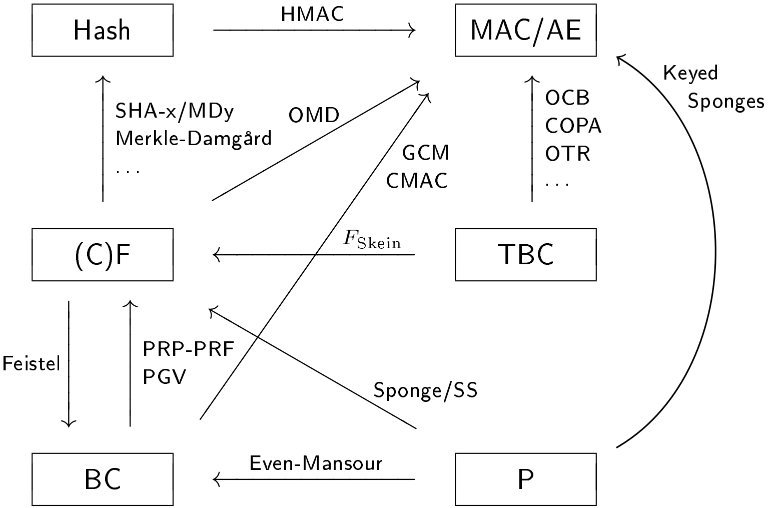
Introduction



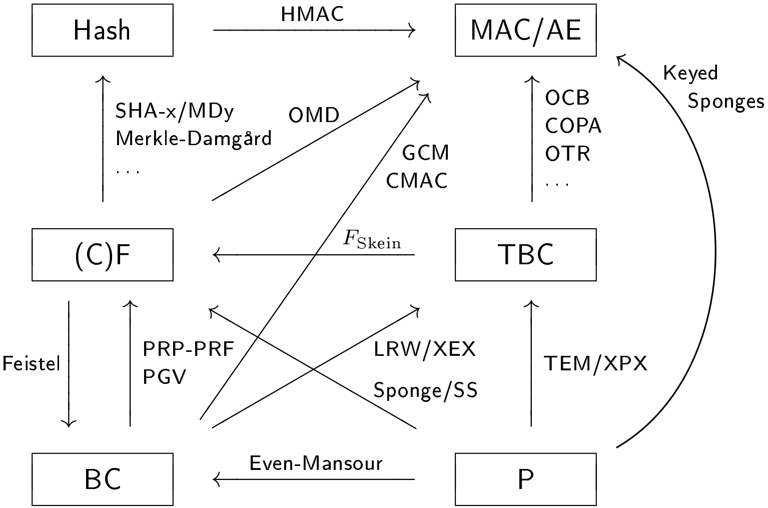
Introduction



Introduction

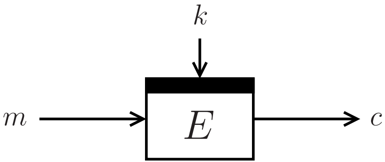


Introduction



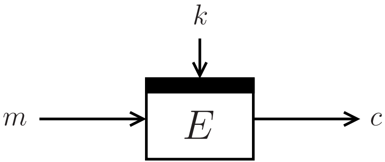
Keyed Constructions

Keyed Blockcipher



- Blockcipher: a family of permutations indexed by key

Keyed Blockcipher



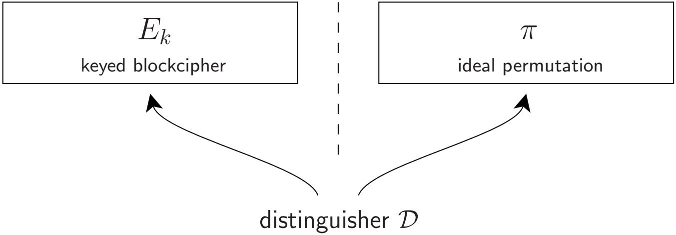
- Blockcipher: a family of permutations indexed by key
- E_k for secret k should behave like permutation π

Keyed Blockcipher: (S)PRP Security

E_k keyed blockcipher	π ideal permutation

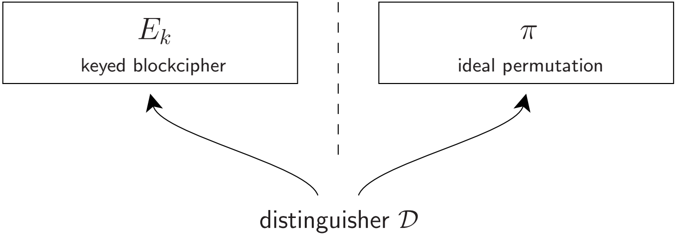
- Two oracles: E_k (for secret random key k) and π

Keyed Blockcipher: (S)PRP Security



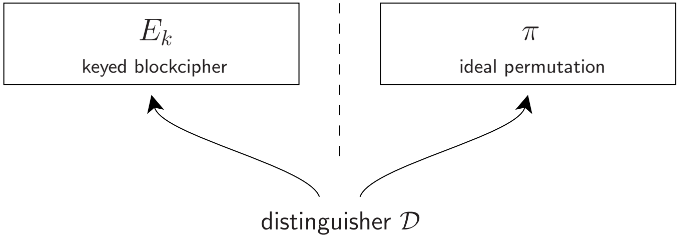
- Two oracles: E_k (for secret random key k) and π
- Distinguisher $\mathcal{D}$ has query access to either E_k or π

Keyed Blockcipher: (S)PRP Security



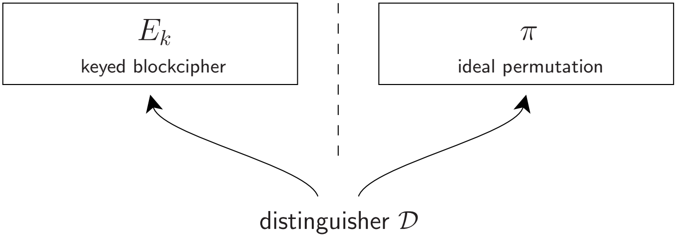
- Two oracles: E_k (for secret random key k) and π
- Distinguisher $\mathcal{D}$ has query access to either E_k or π
  - Forward query: $m \longrightarrow E_k(m)$ or $\pi(m)$
  - Inverse query: $c \longrightarrow E_k^{-1}(c)$ or $\pi^{-1}(c)$

Keyed Blockcipher: (S)PRP Security



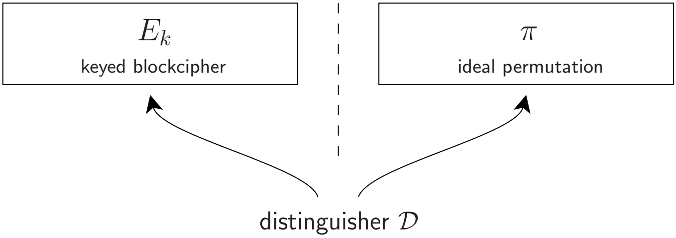
- Two oracles: E_k (for secret random key k) and π
- Distinguisher $\mathcal{D}$ has query access to either E_k or π
  - Forward query: $m \quad \longrightarrow \quad E_k(m)$ or $\pi(m)$
  - Inverse query: $c \quad \longrightarrow \quad E_k^{-1}(c)$ or $\pi^{-1}(c)$
- $\mathcal{D}$ can also make "offline" evaluations of E

Keyed Blockcipher: (S)PRP Security



- Two oracles: E_k (for secret random key k) and π
- Distinguisher $\mathcal{D}$ has query access to either E_k or π
  - Forward query: $m \quad \longrightarrow \quad E_k(m)$ or $\pi(m)$
  - Inverse query: $c \quad \longrightarrow \quad E_k^{-1}(c)$ or $\pi^{-1}(c)$
- $\mathcal{D}$ can also make "offline" evaluations of E
- $\mathcal{D}$ tries to determine which oracle it communicates with

Keyed Blockcipher: (S)PRP Security



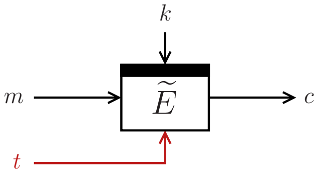
Definition

E is a (strong) pseudorandom permutation if

$$\mathbf{Adv}_E^{\mathrm{(s)prp}}(\mathcal{D}) = \left| \mathbf{Pr}\left[\mathcal{D}^{E_k^{(\pm)}} = 1\right] - \mathbf{Pr}\left[\mathcal{D}^{\pi^{(\pm)}} = 1\right] \right|$$

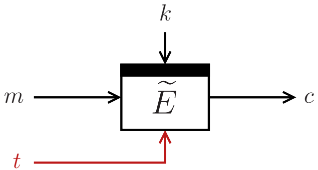
is small. $\mathcal{D}$ is parametrized by:
- Q online queries
- T offline evaluations

Keyed Tweakable Blockcipher



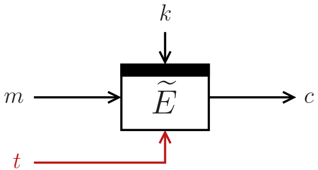
- Tweak: flexibility to the cipher
- Each key and tweak gives a different permutation

Keyed Tweakable Blockcipher



- Tweak: flexibility to the cipher
- Each key and tweak gives a different permutation
- $\widetilde{E}_k$ for secret k should behave like tweakable permutation $\widetilde{\pi}$

Keyed Tweakable Blockcipher



- Tweak: flexibility to the cipher
- Each key and tweak gives a different permutation
- $\widetilde{E}_k$ for secret k should behave like tweakable permutation $\widetilde{\pi}$

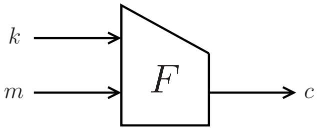
Definition

$\widetilde{E}$ is a (strong) tweakable pseudorandom permutation if

$$\mathbf{Adv}_{\widetilde{E}}^{\mathrm{(s)tprp}}(\mathcal{D}) = \left|\mathbf{Pr}\left[\mathcal{D}^{\widetilde{E}_k^{(\pm)}} = 1\right] - \mathbf{Pr}\left[\mathcal{D}^{\widetilde{\pi}^{(\pm)}} = 1\right]\right|$$

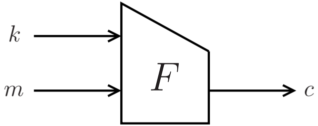
is small. $\mathcal{D}$ is parametrized by:
- Q online queries
- T offline evaluations

Keyed One-Way Function



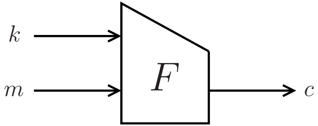
- Keyed one-way function (could be compressing)

Keyed One-Way Function



- Keyed one-way function (could be compressing)
- F_k for secret k should behave like random function \$

Keyed One-Way Function



- Keyed one-way function (could be compressing)
- F_k for secret k should behave like random function $\$$

Definition

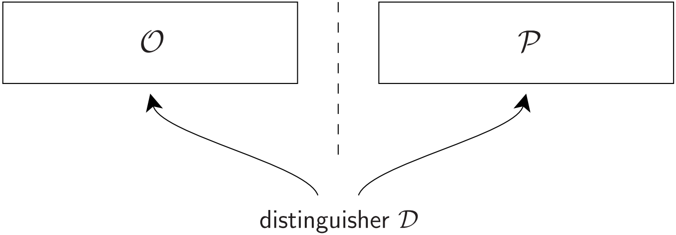
F is a pseudorandom function if

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{D}) = \left|\mathbf{Pr}\left[\mathcal{D}^{F_k} = 1\right] - \mathbf{Pr}\left[\mathcal{D}^{\$} = 1\right]\right|$$

is small. $\mathcal{D}$ is parametrized by:
- Q online queries of length ℓ
- T offline evaluations

Indistinguishability

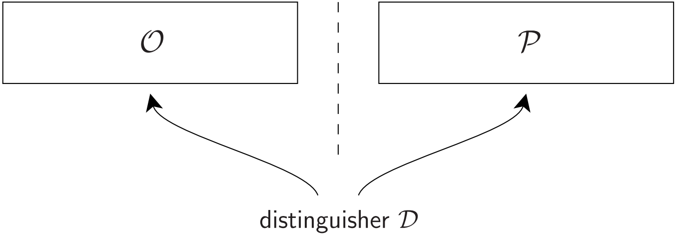
Generalization: Indistinguishability of Random Systems



$$\mathbf{Adv}^{\mathrm{ind}}(\mathcal{D}) = \left|\mathbf{Pr}\left[\mathcal{D}^{\mathcal{O}} = 1\right] - \mathbf{Pr}\left[\mathcal{D}^{\mathcal{P}} = 1\right]\right| = \Delta_{\mathcal{D}}(\mathcal{O}\ ;\ \mathcal{P})$$

Indistinguishability

Generalization: Indistinguishability of Random Systems

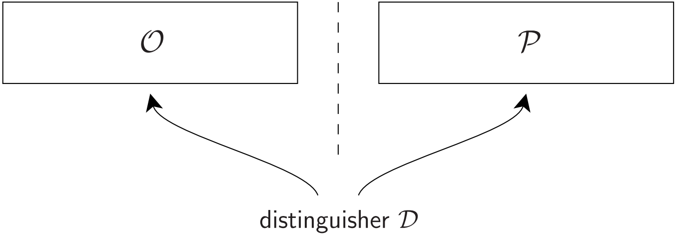


$$\mathbf{Adv}^{\mathrm{ind}}(\mathcal{D}) = \left|\mathbf{Pr}\left[\mathcal{D}^{\mathcal{O}} = 1\right] - \mathbf{Pr}\left[\mathcal{D}^{\mathcal{P}} = 1\right]\right| = \Delta_{\mathcal{D}}(\mathcal{O}\ ;\ \mathcal{P})$$

How to Prove that $\mathbf{Adv}^{\mathrm{ind}}(\mathcal{D})$ is Small?

Indistinguishability

Generalization: Indistinguishability of Random Systems



$$\mathbf{Adv}^{\mathrm{ind}}(\mathcal{D}) = \left|\mathbf{Pr}\left[\mathcal{D}^{\mathcal{O}} = 1\right] - \mathbf{Pr}\left[\mathcal{D}^{\mathcal{P}} = 1\right]\right| = \Delta_{\mathcal{D}}(\mathcal{O}\ ;\ \mathcal{P})$$

How to Prove that $\mathbf{Adv}^{\mathrm{ind}}(\mathcal{D})$ is Small?

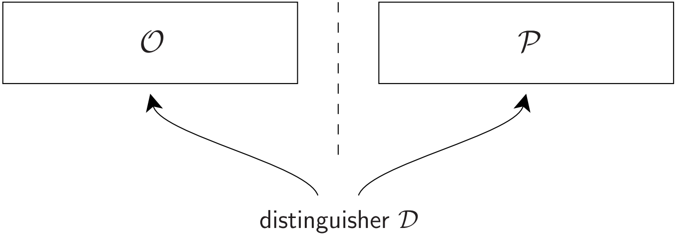
- Game-playing technique
- H-coefficient technique

Game-Playing Technique

- Bellare and Rogaway [BR06]
- Similar to Maurer's methodology [Mau02]

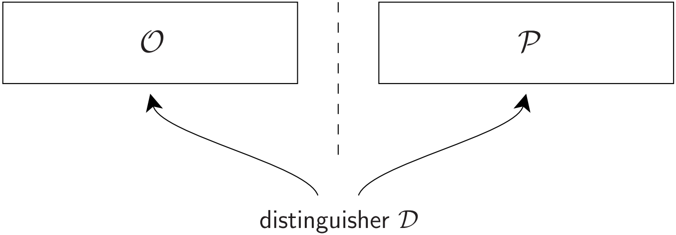
Game-Playing Technique

- Bellare and Rogaway [BR06]
- Similar to Maurer's methodology [Mau02]



Game-Playing Technique

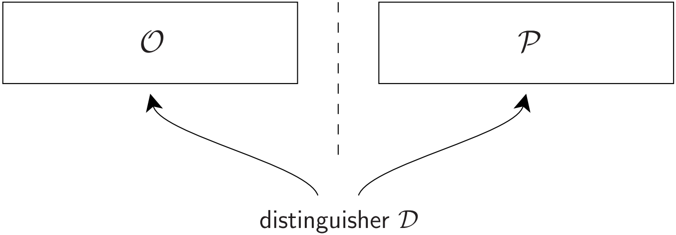
- Bellare and Rogaway [BR06]
- Similar to Maurer's methodology [Mau02]



- Basic idea:
  - From $\mathcal{O}$ to $\mathcal{P}$ in small steps

Game-Playing Technique

- Bellare and Rogaway [BR06]
- Similar to Maurer's methodology [Mau02]



- Basic idea:
  - From $\mathcal{O}$ to $\mathcal{P}$ in small steps
  - Intermediate steps (presumably) easy to analyze

Game-Playing Technique

Triangle Inequality

Fundamental Lemma

Game-Playing Technique

Triangle Inequality

$$\Delta(\mathcal{O};\mathcal{P}) \leq \Delta(\mathcal{O};\mathcal{R}) + \Delta(\mathcal{R};\mathcal{P})$$

Fundamental Lemma

Game-Playing Technique

Triangle Inequality

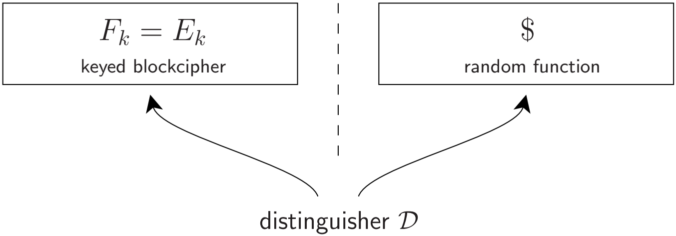
$$\Delta(\mathcal{O};\mathcal{P}) \leq \Delta(\mathcal{O};\mathcal{R}) + \Delta(\mathcal{R};\mathcal{P})$$

Fundamental Lemma

If $\mathcal{O}$ and $\mathcal{P}$ are identical until bad, then:

$$\Delta(\mathcal{O};\mathcal{P}) \leq \mathbf{Pr}\left[\mathcal{P} \text{ sets bad}\right]$$

Example: PRP-PRF Switch (1/4)



Theorem

For any distinguisher $\mathcal{D}$ making Q queries to $E_k/\$$ and T offline evaluations

$$\Delta_{\mathcal{D}}(E_k; \$) \leq \mathbf{Adv}_E^{\mathrm{prp}}(\mathcal{D}) + \frac{\binom{Q}{2}}{2^n}$$

Example: PRP-PRF Switch (2/4)

$$\Delta_{\mathcal{D}}(E_k; \$)$$

Example: PRP-PRF Switch (2/4)

Step 1. "Replace" E_k by Random Permutation π

$$\Delta_{\mathcal{D}}(E_k; \$)$$

Example: PRP-PRF Switch (2/4)

Step 1. "Replace" E_k by Random Permutation π

- Triangle inequality:

$$\Delta_{\mathcal{D}}(E_k; \$) \leq \Delta_{\mathcal{D}}(E_k; \pi) + \Delta_{\mathcal{D}}(\pi; \$)$$

Example: PRP-PRF Switch (2/4)

Step 1. "Replace" E_k by Random Permutation π

- Triangle inequality:

$$\Delta_{\mathcal{D}}(E_k; \$) \leq \Delta_{\mathcal{D}}(E_k; \pi) + \Delta_{\mathcal{D}}(\pi; \$)$$

- $\Delta_{\mathcal{D}}(E_k; \pi) = \mathbf{Adv}_E^{\mathrm{prp}}(\mathcal{D})$ by definition

Example: PRP-PRF Switch (2/4)

Step 1. "Replace" E_k by Random Permutation π

- Triangle inequality:

$$\Delta_{\mathcal{D}}(E_k;\$) \leq \Delta_{\mathcal{D}}(E_k;\pi) + \Delta_{\mathcal{D}}(\pi;\$)$$

- $\Delta_{\mathcal{D}}(E_k;\pi) = \mathbf{Adv}_E^{\mathrm{prp}}(\mathcal{D})$ by definition
- $\Delta_{\mathcal{D}}(\pi;\$)$
  - $\mathcal{D}$ is parametrized by Q queries to $\pi/\$$

Example: PRP-PRF Switch (3/4)

Step 2. Random Permutation to Random Function

- Consider lazily sampled π and $\$$
  - Initially empty list of responses $\mathcal{L}$
  - Randomly generated response for every new query

Example: PRP-PRF Switch (3/4)

Step 2. Random Permutation to Random Function

- Consider lazily sampled π and $\$$
  - Initially empty list of responses $\mathcal{L}$
  - Randomly generated response for every new query

Oracle π
$y \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{L}$ $\mathcal{L} \xleftarrow{\cup} y$ return y

Example: PRP-PRF Switch (3/4)

Step 2. Random Permutation to Random Function

- Consider lazily sampled π and $\$$
  - Initially empty list of responses $\mathcal{L}$
  - Randomly generated response for every new query

Oracle π	Oracle $\$$
$y \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{L}$	$y \xleftarrow{\$} \{0,1\}^n$
$\mathcal{L} \xleftarrow{\cup} y$	
return y	return y

Example: PRP-PRF Switch (3/4)

Step 2. Random Permutation to Random Function

- Consider lazily sampled π and $\$$
  - Initially empty list of responses $\mathcal{L}$
  - Randomly generated response for every new query

Oracle π	Oracle π'	Oracle $\$$

$y \xleftarrow{\$} \{0,1\}^n \setminus \mathcal{L}$	$y \xleftarrow{\$} \{0,1\}^n$
	if $y \in \mathcal{L}$
	    $y \xleftarrow{\$} \{0,1\}^n \setminus \mathcal{L}$
	    bad
		$y \xleftarrow{\$} \{0,1\}^n$

$\mathcal{L} \xleftarrow{\cup} y$ return y	$\mathcal{L} \xleftarrow{\cup} y$ return y	return y

Example: PRP-PRF Switch (4/4)

Oracle π	Oracle π'	Oracle $\$$

$y \xleftarrow{\$} \{0,1\}^n \setminus \mathcal{L}$	$y \xleftarrow{\$} \{0,1\}^n$
	if $y \in \mathcal{L}$
	    $y \xleftarrow{\$} \{0,1\}^n \setminus \mathcal{L}$
	    bad
		$y \xleftarrow{\$} \{0,1\}^n$

$\mathcal{L} \xleftarrow{\cup} y$ return y	$\mathcal{L} \xleftarrow{\cup} y$ return y	return y

$$\Delta_{\mathcal{D}}(\pi; \$)$$

Example: PRP-PRF Switch (4/4)

Oracle π	Oracle π'	Oracle $\$$

$y \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{L}$	$y \xleftarrow{\$} \{0,1\}^n$
	if $y \in \mathcal{L}$
	    $y \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{L}$
	    bad
		$y \xleftarrow{\$} \{0,1\}^n$

$\mathcal{L} \xleftarrow{\cup} y$ return y	$\mathcal{L} \xleftarrow{\cup} y$ return y	return y

- Triangle inequality:

$$\Delta_{\mathcal{D}}(\pi; \$) \leq \Delta_{\mathcal{D}}(\pi; \pi') + \Delta_{\mathcal{D}}(\pi'; \$)$$

Example: PRP-PRF Switch (4/4)

Oracle π

$y \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{L}$

$\mathcal{L} \xleftarrow{\cup} y$
return y

Oracle π'

$y \xleftarrow{\$} \{0,1\}^n$
if $y \in \mathcal{L}$
  $y \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{L}$
  bad
$\mathcal{L} \xleftarrow{\cup} y$
return y

Oracle $\$$

$y \xleftarrow{\$} \{0,1\}^n$

return y

identical

- Triangle inequality:

$$\Delta_{\mathcal{D}}(\pi;\$) \leq \Delta_{\mathcal{D}}(\pi;\pi') + \Delta_{\mathcal{D}}(\pi';\$)$$

$$\leq \quad 0 \quad +$$

Example: PRP-PRF Switch (4/4)

Oracle π	Oracle π'	Oracle \$

$y \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{L}$	$y \xleftarrow{\$} \{0,1\}^n$
	if $y \in \mathcal{L}$
	    $y \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{L}$
	    bad
		$y \xleftarrow{\$} \{0,1\}^n$

$\mathcal{L} \xleftarrow{\cup} y$ return y	$\mathcal{L} \xleftarrow{\cup} y$ return y	return y

identical   identical until bad

- Triangle inequality:

$$\begin{aligned}\Delta_{\mathcal{D}}(\pi;\$) &\leq \Delta_{\mathcal{D}}(\pi;\pi') + \Delta_{\mathcal{D}}(\pi';\$)\\ &\leq \quad 0 \quad + \mathbf{Pr}\left[\pi' \text{ sets bad}\right]\end{aligned}$$

Example: PRP-PRF Switch (4/4)

Oracle π	Oracle π'	Oracle \$
$y \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{L}$	$y \xleftarrow{\$} \{0,1\}^n$	$y \xleftarrow{\$} \{0,1\}^n$
	if $y \in \mathcal{L}$	
	$\quad y \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{L}$	
	$\quad$ bad	
$\mathcal{L} \xleftarrow{\cup} y$	$\mathcal{L} \xleftarrow{\cup} y$	
return y	return y	return y

identical

identical until bad

- Triangle inequality:

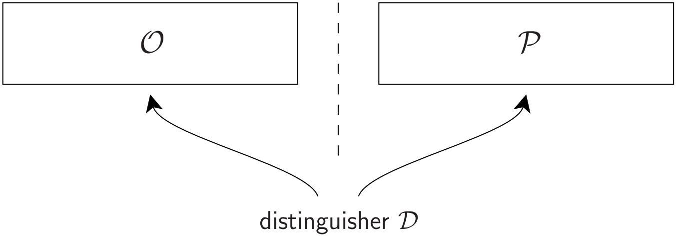
$$\begin{aligned} \Delta_{\mathcal{D}}(\pi;\$) &\leq \Delta_{\mathcal{D}}(\pi;\pi') + \Delta_{\mathcal{D}}(\pi';\$) \\ &\leq \quad 0 \quad + \mathbf{Pr}\left[\pi' \text{ sets bad}\right] \leq \frac{\binom{Q}{2}}{2^n} \end{aligned}$$

H-Coefficient Technique

- Patarin [Pat91,Pat08]
- Popularized by Chen and Steinberger [CS14]
- Similar to “Strong Interpolation Technique” [Ber05]

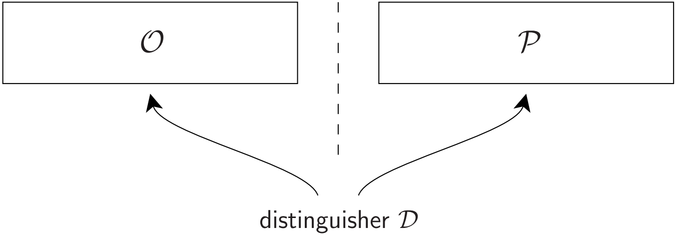
H-Coefficient Technique

- Patarin [Pat91,Pat08]
- Popularized by Chen and Steinberger [CS14]
- Similar to “Strong Interpolation Technique” [Ber05]



H-Coefficient Technique

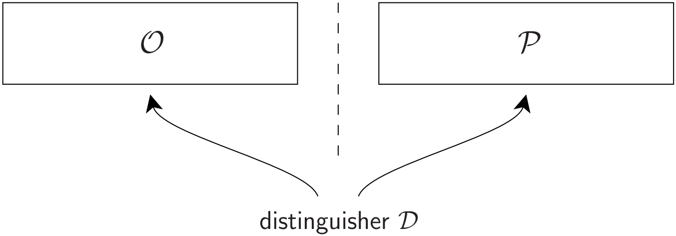
- Patarin [Pat91,Pat08]
- Popularized by Chen and Steinberger [CS14]
- Similar to "Strong Interpolation Technique" [Ber05]



- Basic idea:
  - Each conversation defines a transcript τ

H-Coefficient Technique

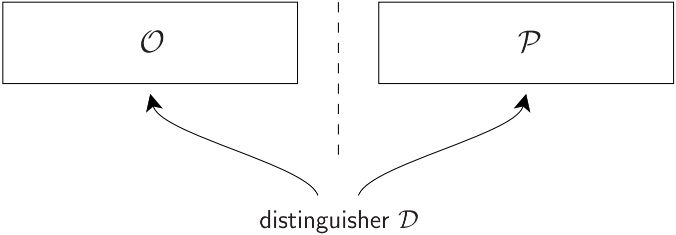
- Patarin [Pat91,Pat08]
- Popularized by Chen and Steinberger [CS14]
- Similar to "Strong Interpolation Technique" [Ber05]



- Basic idea:
  - Each conversation defines a transcript τ
  - $\mathcal{O} \approx \mathcal{P}$ for most of the transcripts

H-Coefficient Technique

- Patarin [Pat91,Pat08]
- Popularized by Chen and Steinberger [CS14]
- Similar to "Strong Interpolation Technique" [Ber05]



- Basic idea:
  - Each conversation defines a transcript τ
  - $\mathcal{O} \approx \mathcal{P}$ for most of the transcripts
  - Remaining transcripts occur with small probability

H-Coefficient Technique

- $\mathcal{D}$ is computationally unbounded and deterministic
- Each conversation defines a transcript τ

H-Coefficient Technique

- $\mathcal{D}$ is computationally unbounded and deterministic
- Each conversation defines a transcript τ
- Consider good and bad transcripts

H-Coefficient Technique

- $\mathcal{D}$ is computationally unbounded and deterministic
- Each conversation defines a transcript τ
- Consider good and bad transcripts

Lemma

Let $\varepsilon > 0$ be such that for all good transcripts τ:

$$\frac{\mathbf{Pr}\left[\mathcal{O} \text{ gives } \tau\right]}{\mathbf{Pr}\left[\mathcal{P} \text{ gives } \tau\right]} \geq 1 - \varepsilon$$

Then, $\Delta_{\mathcal{D}}(\mathcal{O}; P) \leq \varepsilon + \mathbf{Pr}\left[\text{bad transcript for } \mathcal{P}\right]$

H-Coefficient Technique

- $\mathcal{D}$ is computationally unbounded and deterministic
- Each conversation defines a transcript τ
- Consider good and bad transcripts

Lemma

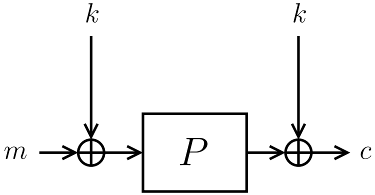
Let $\varepsilon > 0$ be such that for all good transcripts τ:

$$\frac{\mathbf{Pr}\left[\mathcal{O} \text{ gives } \tau\right]}{\mathbf{Pr}\left[\mathcal{P} \text{ gives } \tau\right]} \geq 1 - \varepsilon$$

Then, $\Delta_{\mathcal{D}}(\mathcal{O}; P) \leq \varepsilon + \mathbf{Pr}\left[\text{bad transcript for } \mathcal{P}\right]$

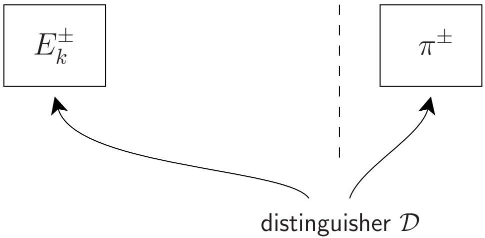
Trade-off: define bad transcripts smartly!

Example: Even-Mansour (1/10)



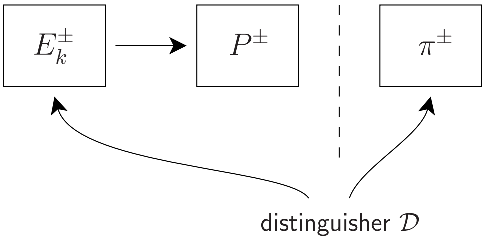
$$E_k(m) = P(m \oplus k) \oplus k$$

Example: Even-Mansour (2/10)



Slightly Different Security Model

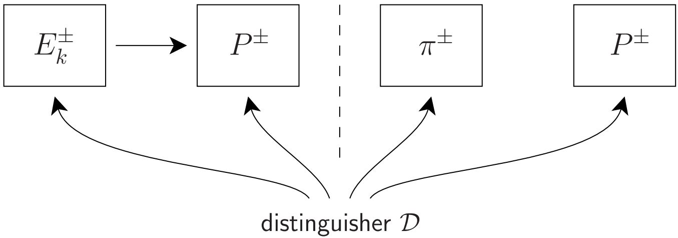
Example: Even-Mansour (2/10)



Slightly Different Security Model

- Underlying permutation

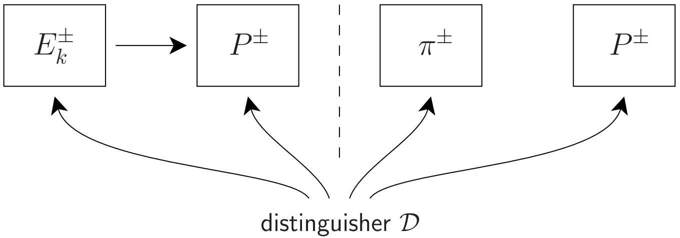
Example: Even-Mansour (2/10)



Slightly Different Security Model

- Underlying permutation randomized
- Information-theoretic distinguisher $\mathcal{D}$
  - Q construction queries
  - T offline evaluations $\approx T$ primitive queries

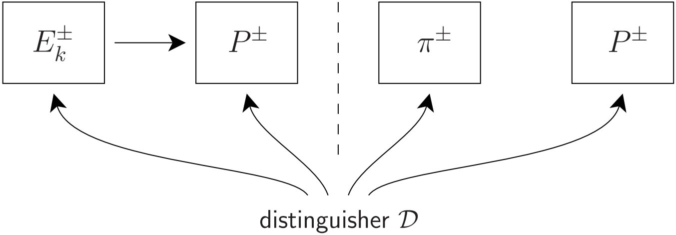
Example: Even-Mansour (2/10)



Slightly Different Security Model

- Underlying permutation randomized
- Information-theoretic distinguisher $\mathcal{D}$
  - Q construction queries
  - T offline evaluations $\approx T$ primitive queries
  - Unbounded computational power

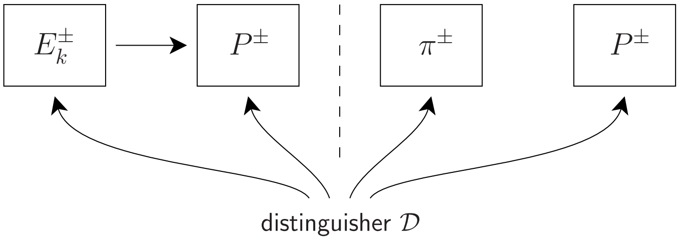
Example: Even-Mansour (3/10)



Slightly Different Security Model

- Without loss of generality, $\mathcal{D}$ is deterministic
  - No random choices

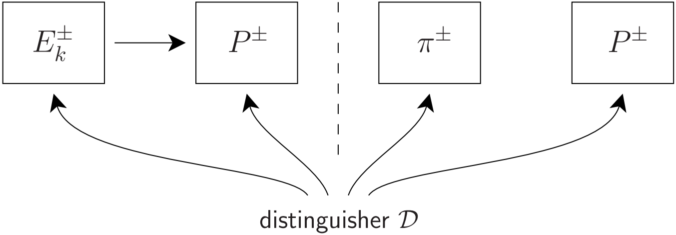
Example: Even-Mansour (3/10)



Slightly Different Security Model

- Without loss of generality, $\mathcal{D}$ is deterministic
  - No random choices
- Reason: at the end we maximize over all distinguishers

Example: Even-Mansour (4/10)



Theorem

For any deterministic distinguisher $\mathcal{D}$ making Q queries to $E_k/\$$ and T primitive queries

$$\mathbf{Adv}_E^{\mathrm{sprp}}(\mathcal{D}) = \Delta_{\mathcal{D}}(E_k^{\pm}, P^{\pm}; \pi^{\pm}, P^{\pm}) \leq \frac{2QT}{2^n}$$

Example: Even-Mansour (5/10)

Step 1. Define how transcripts look like

Step 2. Define good and bad transcripts

Step 3. Upper bound $\mathbf{Pr}\left[\text{bad transcript for } (\pi^{\pm}, P^{\pm})\right]$

Step 4. Lower bound $\frac{\mathbf{Pr}\left[(E_k^{\pm}, P^{\pm}) \text{ gives } \tau\right]}{\mathbf{Pr}\left[(\pi^{\pm}, P^{\pm}) \text{ gives } \tau\right]} \geq 1 - \varepsilon$ ($\forall$ good τ)

Example: Even-Mansour (6/10)

1. **Define how transcripts look like**
   - Construction queries:
     $$\tau_E = \{(m_1, c_1), \ldots, (m_Q, c_Q)\}$$
   - Primitive queries:
     $$\tau_P = \{(x_1, y_1), \ldots, (x_T, y_T)\}$$

Example: Even-Mansour (6/10)

1. Define how transcripts look like

- Construction queries:

$$\tau_E = \{(m_1, c_1), \ldots, (m_Q, c_Q)\}$$

- Primitive queries:

$$\tau_P = \{(x_1, y_1), \ldots, (x_T, y_T)\}$$

- Unordered lists (ordering not needed in current proof)
- 1-to-1 correspondence between any $\mathcal{D}$ and any (τ_E, τ_P)

Example: Even-Mansour (6/10)

1. **Define how transcripts look like**
   - Construction queries:
   $$\tau_E = \{(m_1, c_1), \ldots, (m_Q, c_Q)\}$$
   - Primitive queries:
   $$\tau_P = \{(x_1, y_1), \ldots, (x_T, y_T)\}$$
   - Unordered lists (ordering not needed in current proof)
   - 1-to-1 correspondence between any $\mathcal{D}$ and any (τ_E, τ_P)

   - Bonus information!
     - After interaction of $\mathcal{D}$ with oracles: reveal the key

Example: Even-Mansour (6/10)

1. **Define how transcripts look like**
   - Construction queries:
     $$\tau_E = \{(m_1, c_1), \ldots, (m_Q, c_Q)\}$$
   - Primitive queries:
     $$\tau_P = \{(x_1, y_1), \ldots, (x_T, y_T)\}$$
   - Unordered lists (ordering not needed in current proof)
   - 1-to-1 correspondence between any $\mathcal{D}$ and any (τ_E, τ_P)
   - Bonus information!
     - After interaction of $\mathcal{D}$ with oracles: reveal the key
     - Real world $(E_k^{\pm}, P^{\pm})$: key used for encryption

Example: Even-Mansour (6/10)

1. Define how transcripts look like

- Construction queries:

$$\tau_E = \{(m_1, c_1), \ldots, (m_Q, c_Q)\}$$

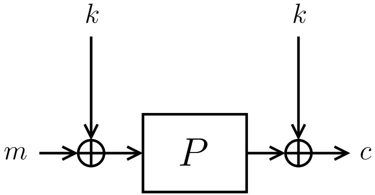
- Primitive queries:

$$\tau_P = \{(x_1, y_1), \ldots, (x_T, y_T)\}$$

- Unordered lists (ordering not needed in current proof)
- 1-to-1 correspondence between any $\mathcal{D}$ and any (τ_E, τ_P)

- Bonus information!
  - After interaction of $\mathcal{D}$ with oracles: reveal the key
  - Real world $(E_k^{\pm}, P^{\pm})$: key used for encryption
  - Ideal world $(\pi^{\pm}, P^{\pm})$: dummy key $k \xleftarrow{\$} \{0,1\}^n$

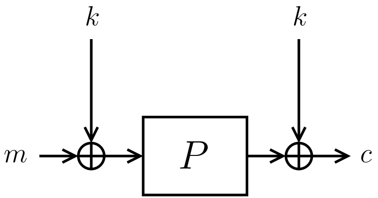
Example: Even-Mansour (7/10)



2. Define good and bad transcripts

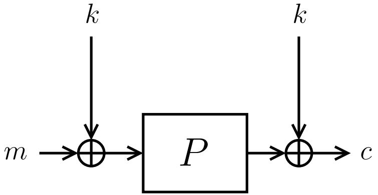
- Intuition:

Example: Even-Mansour (7/10)



2. **Define good and bad transcripts**
   - Intuition:
     - $(m, c) \in \tau_E$ "defines" P-query $(m \oplus k, c \oplus k)$

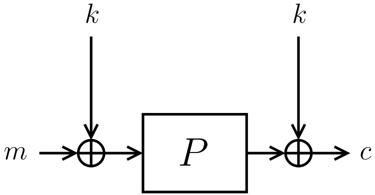
Example: Even-Mansour (7/10)



2. Define good and bad transcripts

- Intuition:
  - $(m, c) \in \tau_E$ "defines" P-query $(m \oplus k, c \oplus k)$
  - Should not collide with any $(x, y) \in \tau_P$

Example: Even-Mansour (7/10)

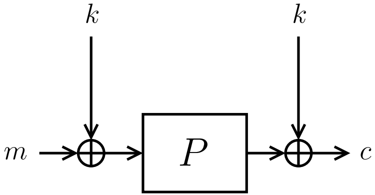


2. Define good and bad transcripts

- Intuition:
  - $(m, c) \in \tau_E$ "defines" P-query $(m \oplus k, c \oplus k)$
  - Should not collide with any $(x, y) \in \tau_P$
- Transcript $\tau = (\tau_E, \tau_P, k)$ is bad if

$\exists (m, c) \in \tau_E, (x, y) \in \tau_P$ such that $m \oplus k = x$ or $c \oplus k = y$

Example: Even-Mansour (7/10)



2. Define good and bad transcripts

- Intuition:
  - $(m, c) \in \tau_E$ "defines" P-query $(m \oplus k, c \oplus k)$
  - Should not collide with any $(x, y) \in \tau_P$
- Transcript $\tau = (\tau_E, \tau_P, k)$ is bad if

$$\exists (m, c) \in \tau_E, (x, y) \in \tau_P \text{ such that } m \oplus k = x \text{ or } c \oplus k = y$$

- Note: no internal collisions in τ_E and τ_P

Example: Even-Mansour (8/10)

3. Upper bound $\mathbf{Pr}\big[$bad transcript for $(\pi^{\pm}, P^{\pm})\big]$

- Transcript $\tau = (\tau_E, \tau_P, k)$ is bad if

$\exists (m, c) \in \tau_E, (x, y) \in \tau_P$ such that $m \oplus k = x$ or $c \oplus k = y$

3. Upper bound $\mathbf{Pr}\big[$bad transcript for $(\boldsymbol{\pi}^{\pm}, \boldsymbol{P}^{\pm})\big]$

- Transcript $\tau = (\tau_E, \tau_P, k)$ is bad if

$\exists (m, c) \in \tau_E, (x, y) \in \tau_P$ such that $m \oplus k = x$ or $c \oplus k = y$

$$\Updownarrow$$

$$k \in \{m \oplus x, c \oplus y \mid (m, c) \in \tau_E, (x, y) \in \tau_P\}$$

Example: Even-Mansour (8/10)

3. Upper bound $\mathbf{Pr}\big[$bad transcript for $(\pi^{\pm}, P^{\pm})\big]$

- Transcript $\tau = (\tau_E, \tau_P, k)$ is bad if

$\exists (m,c) \in \tau_E, (x,y) \in \tau_P$ such that $m \oplus k = x$ or $c \oplus k = y$

$$\Updownarrow$$

$$k \in \underbrace{\{m \oplus x, c \oplus y \mid (m,c) \in \tau_E, (x,y) \in \tau_P\}}_{\text{of size } \leq 2QT}$$

Example: Even-Mansour (8/10)

3. Upper bound $\mathbf{Pr}\left[\text{bad transcript for } (\pi^{\pm}, P^{\pm})\right]$

- Transcript $\tau = (\tau_E, \tau_P, k)$ is bad if

$$\exists (m, c) \in \tau_E, (x, y) \in \tau_P \text{ such that } m \oplus k = x \text{ or } c \oplus k = y$$

$$\Updownarrow$$

$$k \in \underbrace{\{m \oplus x, c \oplus y \mid (m, c) \in \tau_E, (x, y) \in \tau_P\}}_{\text{of size } \leq 2QT}$$

$\uparrow$

independently generated n-bit dummy key

Example: Even-Mansour (8/10)

3. Upper bound $\mathbf{Pr}\left[\text{bad transcript for } (\pi^{\pm}, P^{\pm})\right]$

- Transcript $\tau = (\tau_E, \tau_P, k)$ is bad if

$\exists (m,c) \in \tau_E, (x,y) \in \tau_P$ such that $m \oplus k = x$ or $c \oplus k = y$

$$\Updownarrow$$

$$\underset{\substack{\uparrow \\ \text{independently generated } n\text{-bit dummy key}}}{k} \in \underbrace{\{m \oplus x, c \oplus y \mid (m,c) \in \tau_E, (x,y) \in \tau_P\}}_{\text{of size } \leq 2QT}$$

$$\mathbf{Pr}\left[\text{bad transcript for } (\pi^{\pm}, P^{\pm})\right] \leq \frac{2QT}{2^n}$$

Example: Even-Mansour (9/10)

4. Lower bound $\dfrac{\mathbf{Pr}\left[(E_k^{\pm}, P^{\pm}) \text{ gives } \tau\right]}{\mathbf{Pr}\left[(\pi^{\pm}, P^{\pm}) \text{ gives } \tau\right]} \geq 1 - \varepsilon \ (\forall \text{ good } \tau)$

Example: Even-Mansour (9/10)

4. **Lower bound** $\dfrac{\mathbf{Pr}\left[(E_k^{\pm}, P^{\pm}) \text{ gives } \tau\right]}{\mathbf{Pr}\left[(\pi^{\pm}, P^{\pm}) \text{ gives } \tau\right]} \geq 1 - \varepsilon$ **($\forall$ good τ)**

- Counting "compatible" oracles (modulo details):

$$\mathbf{Pr}\left[\mathcal{O} \text{ gives } \tau\right] = \frac{\left|\text{oracles } \mathcal{O} \text{ that could give } \tau\right|}{\left|\text{oracles } \mathcal{O}\right|}$$

Example: Even-Mansour (9/10)

4. Lower bound $\dfrac{\mathbf{Pr}\left[(E_k^{\pm}, P^{\pm}) \text{ gives } \tau\right]}{\mathbf{Pr}\left[(\pi^{\pm}, P^{\pm}) \text{ gives } \tau\right]} \geq 1 - \varepsilon \ (\forall \text{ good } \tau)$

- Counting "compatible" oracles (modulo details):

$$\mathbf{Pr}\left[\mathcal{O} \text{ gives } \tau\right] = \frac{\left|\text{oracles } \mathcal{O} \text{ that could give } \tau\right|}{\left|\text{oracles } \mathcal{O}\right|}$$

- For real world $(E_k^{\pm}, P^{\pm})$:

$$\mathbf{Pr}\left[(E_k^{\pm}, P^{\pm}) \text{ gives } \tau\right] = \overline{}$$

Example: Even-Mansour (9/10)

4. Lower bound $\dfrac{\mathbf{Pr}\left[(E_k^{\pm}, P^{\pm}) \text{ gives } \tau\right]}{\mathbf{Pr}\left[(\pi^{\pm}, P^{\pm}) \text{ gives } \tau\right]} \geq 1 - \varepsilon$ **($\forall$ good τ)**

- Counting "compatible" oracles (modulo details):

$$\mathbf{Pr}\left[\mathcal{O} \text{ gives } \tau\right] = \frac{\left|\text{oracles } \mathcal{O} \text{ that could give } \tau\right|}{\left|\text{oracles } \mathcal{O}\right|}$$

- For real world $(E_k^{\pm}, P^{\pm})$:

$$\mathbf{Pr}\left[(E_k^{\pm}, P^{\pm}) \text{ gives } \tau\right] = \frac{}{2^n \cdot 2^n!}$$

Example: Even-Mansour (9/10)

4. Lower bound $\dfrac{\mathbf{Pr}\left[(E_k^{\pm}, P^{\pm}) \text{ gives } \tau\right]}{\mathbf{Pr}\left[(\pi^{\pm}, P^{\pm}) \text{ gives } \tau\right]} \geq 1 - \varepsilon$ **($\forall$ good τ)**

- Counting "compatible" oracles (modulo details):

$$\mathbf{Pr}\left[\mathcal{O} \text{ gives } \tau\right] = \frac{\left|\text{oracles } \mathcal{O} \text{ that could give } \tau\right|}{\left|\text{oracles } \mathcal{O}\right|}$$

- For real world $(E_k^{\pm}, P^{\pm})$:

$$\mathbf{Pr}\left[(E_k^{\pm}, P^{\pm}) \text{ gives } \tau\right] = \frac{(2^n - Q - T)!}{2^n \cdot 2^n!}$$

Example: Even-Mansour (9/10)

4. Lower bound $\boldsymbol{\frac{\mathbf{Pr}\left[(E_k^\pm, P^\pm) \text{ gives } \tau\right]}{\mathbf{Pr}\left[(\pi^\pm, P^\pm) \text{ gives } \tau\right]} \geq 1 - \varepsilon \; (\forall \text{ good } \tau)}$

- Counting "compatible" oracles (modulo details):

$$\mathbf{Pr}\left[\mathcal{O} \text{ gives } \tau\right] = \frac{\left|\text{oracles } \mathcal{O} \text{ that could give } \tau\right|}{\left|\text{oracles } \mathcal{O}\right|}$$

- For real world $(E_k^\pm, P^\pm)$:

$$\mathbf{Pr}\left[(E_k^\pm, P^\pm) \text{ gives } \tau\right] = \frac{(2^n - Q - T)!}{2^n \cdot 2^n!}$$

- For ideal world $(\pi^\pm, P^\pm)$:

$$\mathbf{Pr}\left[(\pi^\pm, P^\pm) \text{ gives } \tau\right] = \frac{(2^n - Q)!(2^n - T)!}{2^n \cdot (2^n!)^2}$$

Example: Even-Mansour (10/10)

4. Lower bound $\frac{\mathbf{Pr}\left[(E_k^{\pm},P^{\pm})\text{ gives }\tau\right]}{\mathbf{Pr}\left[(\pi^{\pm},P^{\pm})\text{ gives }\tau\right]} \geq 1-\varepsilon$ **($\forall$ good τ)**

- Putting things together:

$$\frac{\mathbf{Pr}\left[(E_k^{\pm},P^{\pm})\text{ gives }\tau\right]}{\mathbf{Pr}\left[(\pi^{\pm},P^{\pm})\text{ gives }\tau\right]} = \frac{\frac{(2^n-Q-T)!}{2^n\cdot 2^n!}}{\frac{(2^n-Q)!(2^n-T)!}{2^n\cdot(2^n!)^2}} = \frac{(2^n-Q-T)!2^n!}{(2^n-Q)!(2^n-T)!}$$

Example: Even-Mansour (10/10)

4. Lower bound $\dfrac{\mathbf{Pr}\left[(E_k^{\pm},P^{\pm}) \text{ gives } \tau\right]}{\mathbf{Pr}\left[(\pi^{\pm},P^{\pm}) \text{ gives } \tau\right]} \geq 1-\varepsilon$ **($\forall$ good τ)**

- Putting things together:

$$\frac{\mathbf{Pr}\left[(E_k^{\pm},P^{\pm}) \text{ gives } \tau\right]}{\mathbf{Pr}\left[(\pi^{\pm},P^{\pm}) \text{ gives } \tau\right]} = \frac{\frac{(2^n-Q-T)!}{2^n\cdot 2^n!}}{\frac{(2^n-Q)!(2^n-T)!}{2^n\cdot(2^n!)^2}}$$

$$= \frac{(2^n-Q-T)!2^n!}{(2^n-Q)!(2^n-T)!}$$

$$\geq 1$$

Example: Even-Mansour (10/10)

4. Lower bound $\dfrac{\mathbf{Pr}\left[(E_k^{\pm}, P^{\pm}) \text{ gives } \tau\right]}{\mathbf{Pr}\left[(\pi^{\pm}, P^{\pm}) \text{ gives } \tau\right]} \geq 1 - \varepsilon$ **($\forall$ good τ)**

- Putting things together:

$$\frac{\mathbf{Pr}\left[(E_k^{\pm}, P^{\pm}) \text{ gives } \tau\right]}{\mathbf{Pr}\left[(\pi^{\pm}, P^{\pm}) \text{ gives } \tau\right]} = \frac{\frac{(2^n-Q-T)!}{2^n \cdot 2^n!}}{\frac{(2^n-Q)!(2^n-T)!}{2^n \cdot (2^n!)^2}} = \frac{(2^n - Q - T)! 2^n!}{(2^n - Q)!(2^n - T)!} \geq 1$$

- We put $\varepsilon = 0$

Example: Even-Mansour (10/10)

4. Lower bound $\frac{\mathbf{Pr}\left[(E_k^{\pm}, P^{\pm}) \text{ gives } \tau\right]}{\mathbf{Pr}\left[(\pi^{\pm}, P^{\pm}) \text{ gives } \tau\right]} \geq 1 - \varepsilon$ **($\forall$ good τ)**

- Putting things together:

$$\frac{\mathbf{Pr}\left[(E_k^{\pm}, P^{\pm}) \text{ gives } \tau\right]}{\mathbf{Pr}\left[(\pi^{\pm}, P^{\pm}) \text{ gives } \tau\right]} = \frac{\frac{(2^n-Q-T)!}{2^n \cdot 2^n!}}{\frac{(2^n-Q)!(2^n-T)!}{2^n \cdot (2^n!)^2}}$$

$$= \frac{(2^n - Q - T)!2^n!}{(2^n - Q)!(2^n - T)!}$$

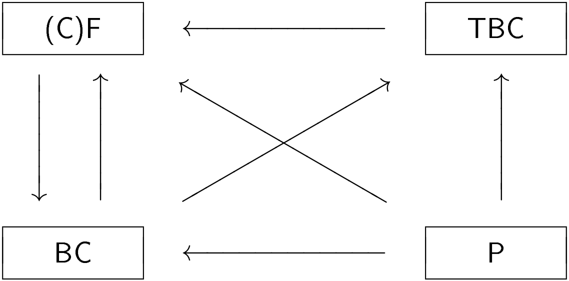
$$\geq 1$$

- We put $\varepsilon = 0$

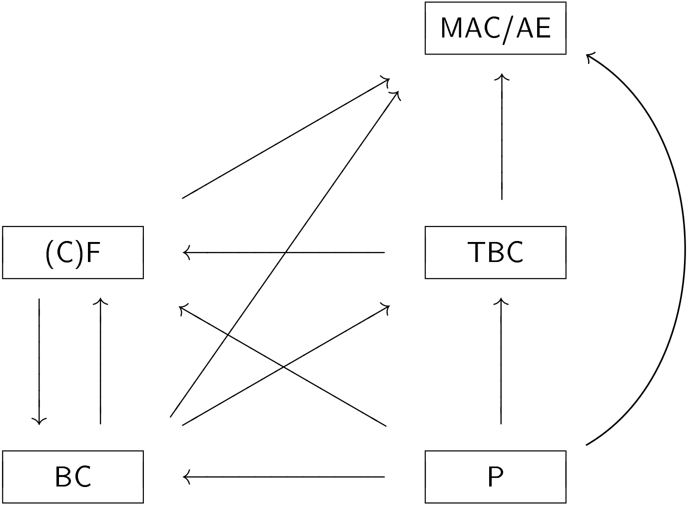
- Conclusion:

$$\mathbf{Adv}_E^{\text{sprp}}(\mathcal{D}) = \Delta_{\mathcal{D}}(E_k^{\pm}, P^{\pm}; \pi^{\pm}, P^{\pm}) \leq \frac{2QT}{2^n} + 0$$

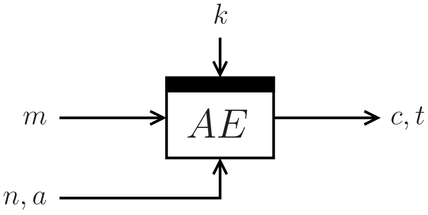
Keyed Authenticated Encryption



Keyed Authenticated Encryption



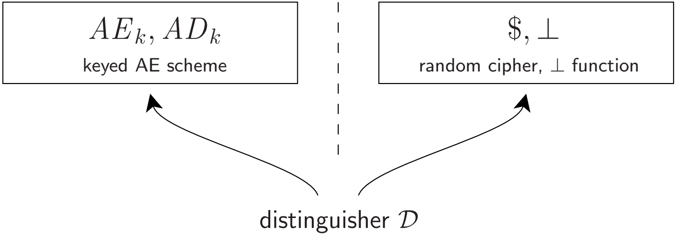
Keyed Authenticated Encryption



- Confidentiality: c should always look random
- Authenticity: t is "hard to forge"

- (AE_k, AD_k) for secret k should behave like $(\$, \perp)$

Keyed Authenticated Encryption: AE Security



Definition

(AE, AD) is a secure authenticated encryption scheme if[1]

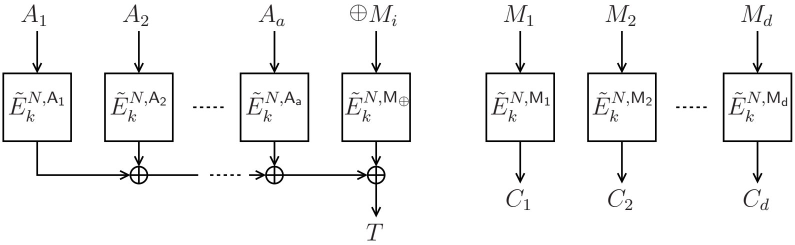
$$\mathbf{Adv}_{AE}^{\mathrm{ae}}(\mathcal{D}) = \left| \mathbf{Pr}\left[\mathcal{D}^{AE_k, AD_k} = 1\right] - \mathbf{Pr}\left[\mathcal{D}^{\$,\perp} = 1\right] \right|$$

is small. $\mathcal{D}$ is parametrized by:

- Q online queries of length ℓ (nonce-respecting/misusing)
- T offline evaluations

[1] Also known as CCA3 security

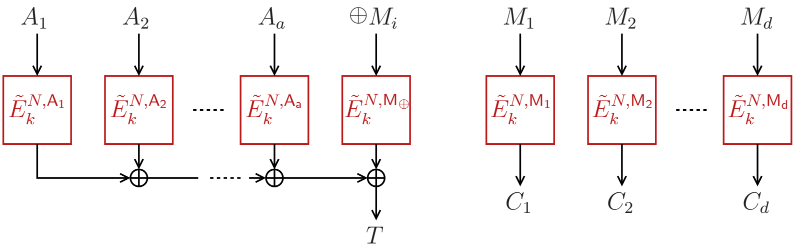
Example: OCBx (1/2)



- Generalized OCB by Rogaway et al. [RBBK01,Rog04,KR11]

$$\Delta_{\mathcal{D}}(AE_k \;, AD_k \;; \$, \perp)$$

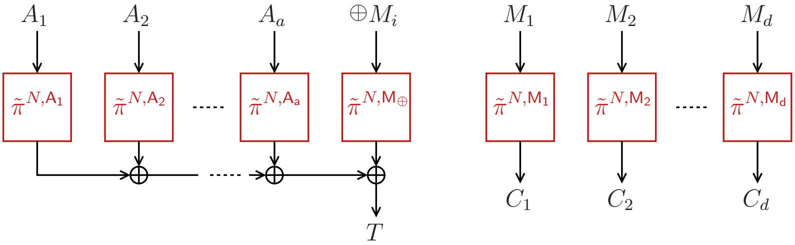
Example: OCBx (1/2)



- Generalized OCB by Rogaway et al. [RBBK01,Rog04,KR11]
- Internally based on tweakable blockcipher $\widetilde{E}$
  - Tweak (N, tweak) is unique for every evaluation

$$\Delta_{\mathcal{D}}(AE_k^{\widetilde{E}_k}, AD_k^{\widetilde{E}_k}; \$, \perp)$$

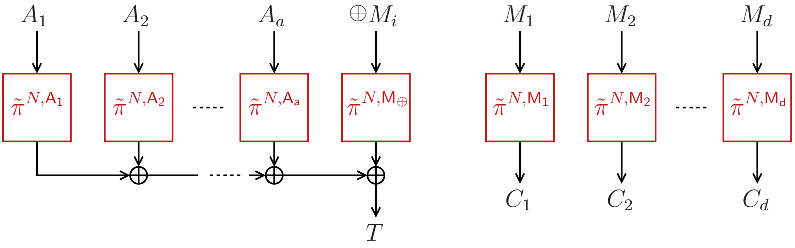
Example: OCBx (1/2)



- Generalized OCB by Rogaway et al. [RBBK01,Rog04,KR11]
- Internally based on tweakable blockcipher $\widetilde{E}$
  - Tweak (N, tweak) is unique for every evaluation
- Triangle inequality:

$$\Delta_{\mathcal{D}}(AE_k^{\widetilde{E}_k}, AD_k^{\widetilde{E}_k}; \$, \perp) \leq \Delta_{\mathcal{D}}(AE^{\widetilde{\pi}}, AD^{\widetilde{\pi}}; \$, \perp) + \Delta_{\mathcal{D}'}(\widetilde{E}_k^{\pm}; \widetilde{\pi}^{\pm})$$

Example: OCBx (1/2)

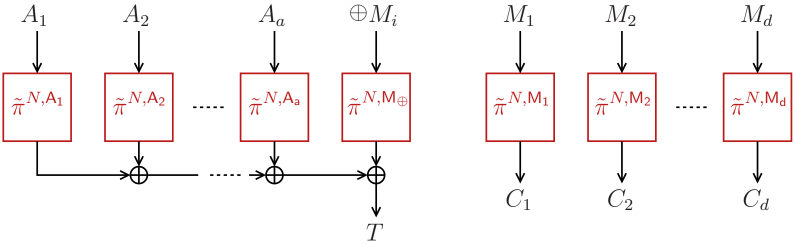


- Generalized OCB by Rogaway et al. [RBBK01,Rog04,KR11]
- Internally based on tweakable blockcipher $\widetilde{E}$
  - Tweak (N, tweak) is unique for every evaluation
- Triangle inequality:

$$\Delta_{\mathcal{D}}(AE_k^{\widetilde{E}_k}, AD_k^{\widetilde{E}_k}; \$, \perp) \leq \Delta_{\mathcal{D}}(AE^{\widetilde{\pi}}, AD^{\widetilde{\pi}}; \$, \perp) + \Delta_{\mathcal{D}'}(\widetilde{E}_k^{\pm}; \widetilde{\pi}^{\pm})$$

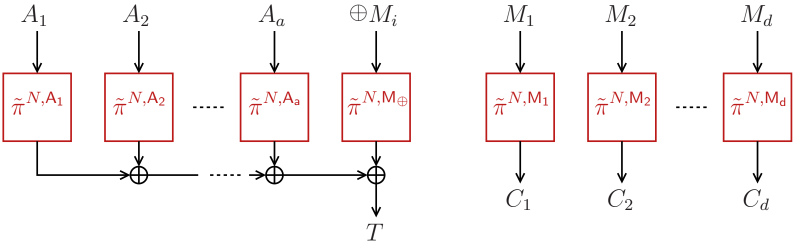
STPRP security of $\widetilde{E}$

Example: OCBx (2/2)



$$\Delta_{\mathcal{D}}(AE^{\widetilde{\pi}}, AD^{\widetilde{\pi}}; \$, \perp)$$

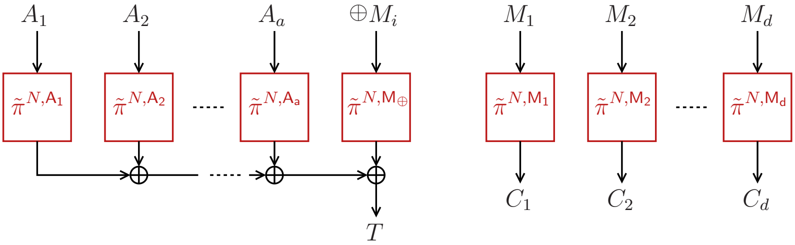
Example: OCBx (2/2)



- Nonce uniqueness $\Rightarrow$ tweak uniqueness

$$\Delta_{\mathcal{D}}(AE^{\widetilde{\pi}}, AD^{\widetilde{\pi}}; \$, \perp)$$

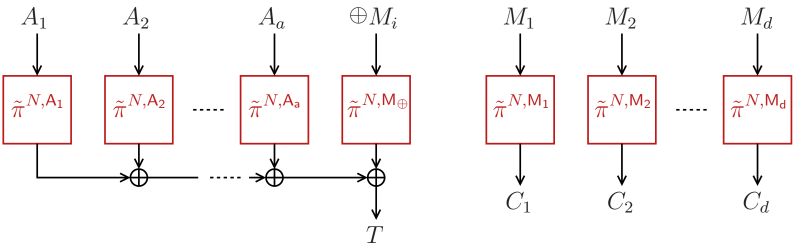
Example: OCBx (2/2)



- Nonce uniqueness $\Rightarrow$ tweak uniqueness
- Encryption calls behave like random functions: $AE^{\widetilde{\pi}} = \$$

$$\Delta_{\mathcal{D}}(AE^{\widetilde{\pi}}, AD^{\widetilde{\pi}}; \$, \perp)$$

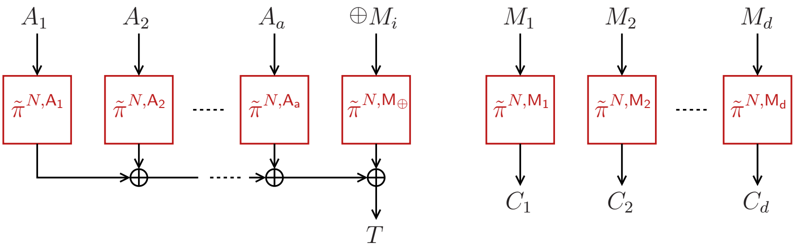
Example: OCBx (2/2)



- Nonce uniqueness $\Rightarrow$ tweak uniqueness
- Encryption calls behave like random functions: $AE^{\widetilde{\pi}} = \$$
- Authentication behaves like random function

$$\Delta_{\mathcal{D}}(AE^{\widetilde{\pi}}, AD^{\widetilde{\pi}}; \$, \perp)$$

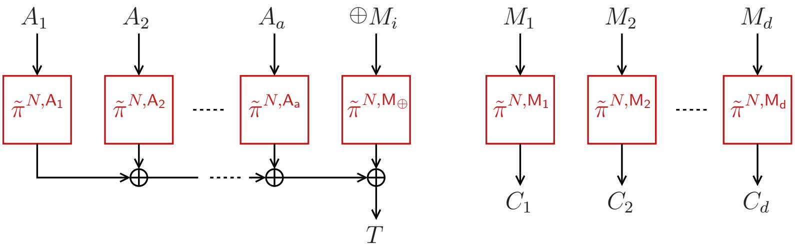
Example: OCBx (2/2)



- Nonce uniqueness $\Rightarrow$ tweak uniqueness
- Encryption calls behave like random functions: $AE^{\widetilde{\pi}} = \$$
- Authentication behaves like random function
  - Tag forged with probability at most $1/(2^n - 1)$

$$\Delta_{\mathcal{D}}(AE^{\widetilde{\pi}}, AD^{\widetilde{\pi}}; \$, \perp)$$

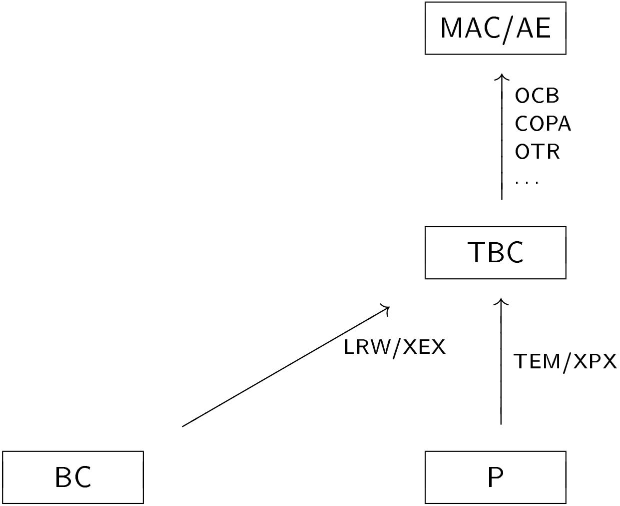
Example: OCBx (2/2)



- Nonce uniqueness $\Rightarrow$ tweak uniqueness
- Encryption calls behave like random functions: $AE^{\widetilde{\pi}} = \$$
- Authentication behaves like random function
  - Tag forged with probability at most $1/(2^n - 1)$

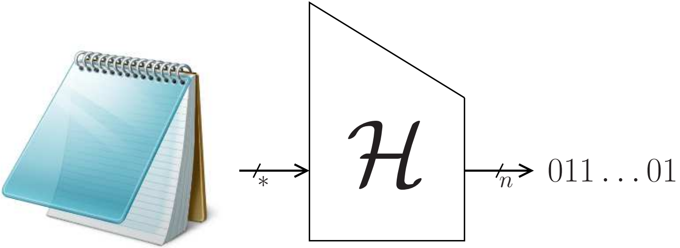
$$\Delta_{\mathcal{D}}(AE^{\widetilde{\pi}}, AD^{\widetilde{\pi}}; \$, \perp) \leq 1/(2^n - 1)$$

Tomorrow's Talk



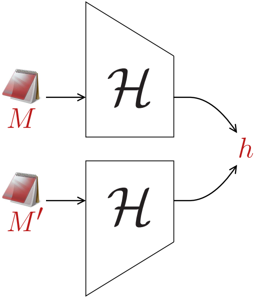
Keyless Constructions

Hash Functions



Hash Functions: Classical Security Requirements

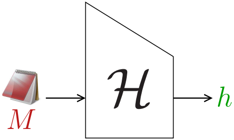
Collision



Find $M \neq M'$

Application:
2012 Flame virus

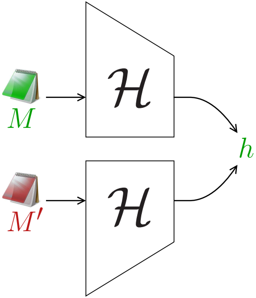
Preimage



Given h, find M

Application:
passphrase protection

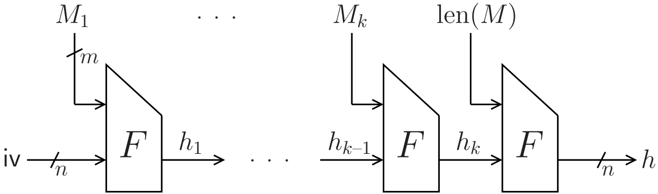
Second Preimage



Given M, find $M' \neq M$

Application:
data integrity

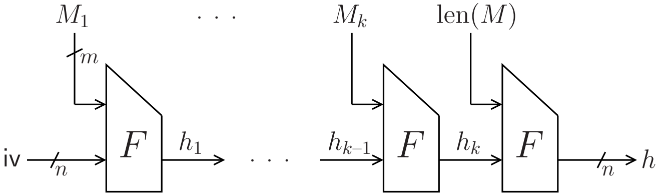
Hash Functions from Compression Functions



Merkle-Damgård with Strengthening

- Damgård [Dam89] and Merkle [Mer89]
- Consecutive evaluation of compression function F
- Length encoding at the end

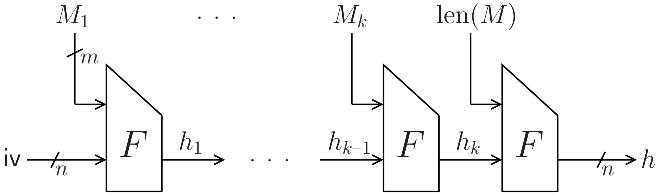
Hash Functions from Compression Functions



Security of Merkle-Damgård

- $\mathcal{H}$ and F have same security models

Hash Functions from Compression Functions



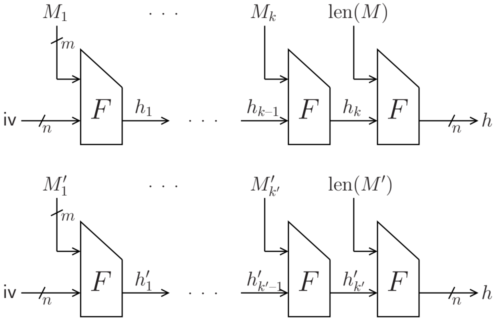
Security of Merkle-Damgård

- $\mathcal{H}$ and F have same security models
- Ideally, we want:

$$F \text{ is col/sec/pre secure} \implies \mathcal{H} \text{ is col/sec/pre secure}$$

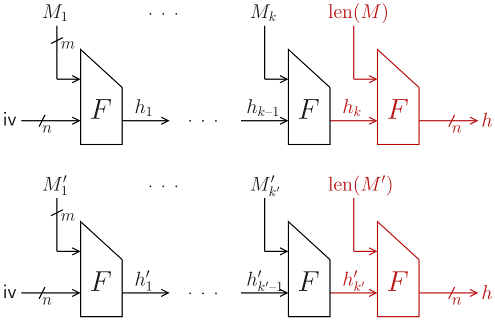
Collision Security of Merkle-Damgård

- Suppose we are given a collision $\mathcal{H}(M) = \mathcal{H}(M')$



Collision Security of Merkle-Damgård

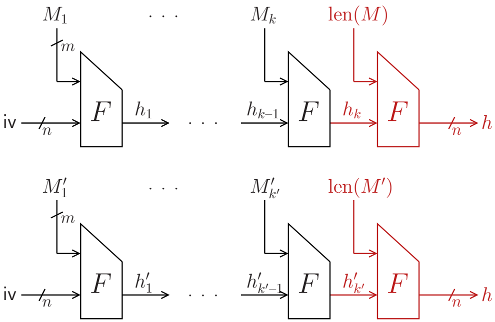
- Suppose we are given a collision $\mathcal{H}(M) = \mathcal{H}(M')$



- Then, $F(h_k, \mathrm{len}(M)) = F(h'_{k'}, \mathrm{len}(M'))$

Collision Security of Merkle-Damgård

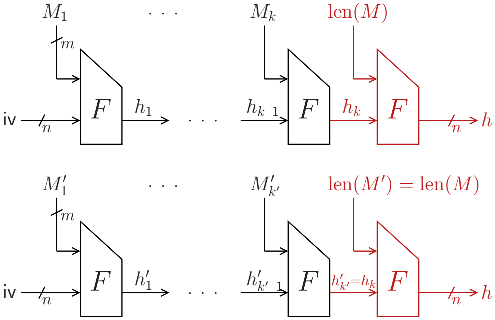
- Suppose we are given a collision $\mathcal{H}(M) = \mathcal{H}(M')$



- Then, $F(h_k, \mathrm{len}(M)) = F(h'_{k'}, \mathrm{len}(M'))$
  - If $(h_k, \mathrm{len}(M)) \neq (h'_{k'}, \mathrm{len}(M'))$: this is an F-collision

Collision Security of Merkle-Damgård

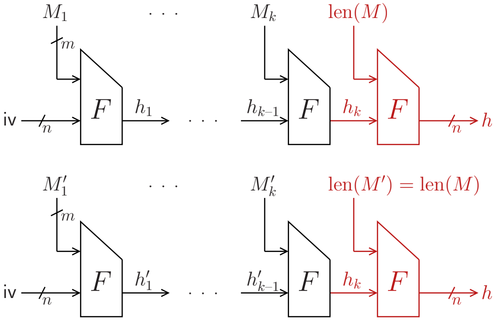
- Suppose we are given a collision $\mathcal{H}(M) = \mathcal{H}(M')$



- Then, $F(h_k, \mathrm{len}(M)) = F(h'_{k'}, \mathrm{len}(M'))$
  - If $(h_k, \mathrm{len}(M)) \neq (h'_{k'}, \mathrm{len}(M'))$: this is an F-collision
  - Else,

Collision Security of Merkle-Damgård

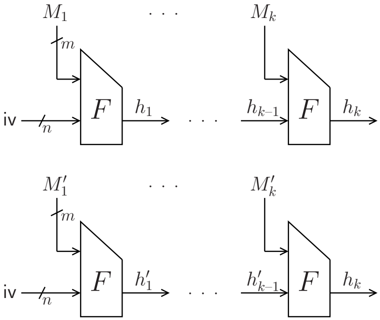
- Suppose we are given a collision $\mathcal{H}(M) = \mathcal{H}(M')$



- Then, $F(h_k, \mathrm{len}(M)) = F(h'_{k'}, \mathrm{len}(M'))$
  - If $(h_k, \mathrm{len}(M)) \neq (h'_{k'}, \mathrm{len}(M'))$: this is an F-collision
  - Else,

Collision Security of Merkle-Damgård

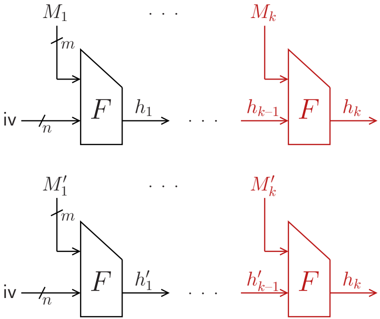
- Suppose we are given a collision $\mathcal{H}(M) = \mathcal{H}(M')$



- Then, $F(h_k, \mathrm{len}(M)) = F(h'_{k'}, \mathrm{len}(M'))$
  - If $(h_k, \mathrm{len}(M)) \neq (h'_{k'}, \mathrm{len}(M'))$: this is an F-collision
  - Else,

Collision Security of Merkle-Damgård

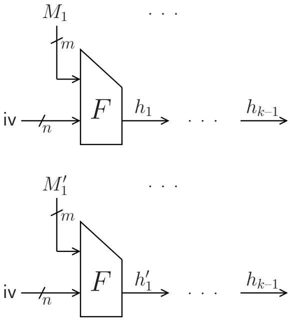
- Suppose we are given a collision $\mathcal{H}(M) = \mathcal{H}(M')$



- Then, $F(h_k, \mathrm{len}(M)) = F(h'_{k'}, \mathrm{len}(M'))$
  - If $(h_k, \mathrm{len}(M)) \neq (h'_{k'}, \mathrm{len}(M'))$: this is an F-collision
  - Else, $F(h_{k-1}, M_k) = F(h'_{k-1}, M'_k)$

Collision Security of Merkle-Damgård

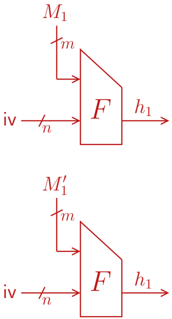
- Suppose we are given a collision $\mathcal{H}(M) = \mathcal{H}(M')$



- Then, $F(h_k, \mathrm{len}(M)) = F(h'_{k'}, \mathrm{len}(M'))$
  - If $(h_k, \mathrm{len}(M)) \neq (h'_{k'}, \mathrm{len}(M'))$: this is an F-collision
  - Else, $F(h_{k-1}, M_k) = F(h'_{k-1}, M'_k)$
    - $\cdots$

Collision Security of Merkle-Damgård

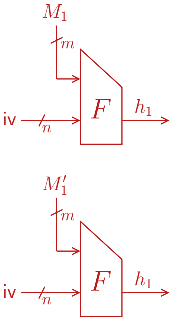
- Suppose we are given a collision $\mathcal{H}(M) = \mathcal{H}(M')$



- Then, $F(h_k, \mathrm{len}(M)) = F(h'_{k'}, \mathrm{len}(M'))$
  - If $(h_k, \mathrm{len}(M)) \neq (h'_{k'}, \mathrm{len}(M'))$: this is an F-collision
  - Else, $F(h_{k-1}, M_k) = F(h'_{k-1}, M'_k)$
    - $\ldots$

Collision Security of Merkle-Damgård

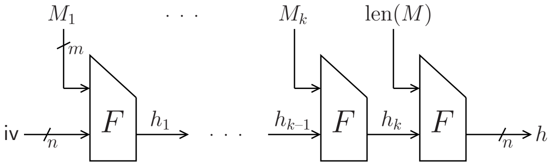
- Suppose we are given a collision $\mathcal{H}(M) = \mathcal{H}(M')$



- Then, $F(h_k, \mathrm{len}(M)) = F(h'_{k'}, \mathrm{len}(M'))$
  - If $(h_k, \mathrm{len}(M)) \neq (h'_{k'}, \mathrm{len}(M'))$: this is an F-collision
  - Else, $F(h_{k-1}, M_k) = F(h'_{k-1}, M'_k)$
    - $\dots$
- We can find F-collision

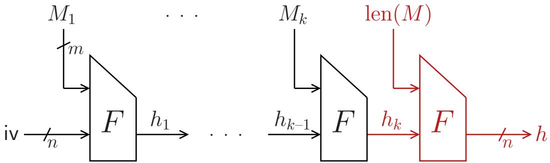
Preimage Security of Merkle-Damgård

- Let h be any range value
- Suppose we are given a preimage $\mathcal{H}(M) = h$



Preimage Security of Merkle-Damgård

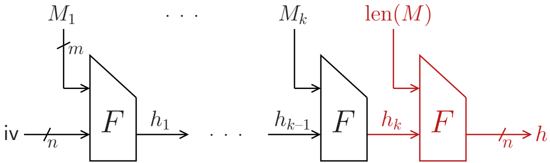
- Let h be any range value
- Suppose we are given a preimage $\mathcal{H}(M) = h$



- Then, $F(h_k, \mathrm{len}(M)) = h$

Preimage Security of Merkle-Damgård

- Let h be any range value
- Suppose we are given a preimage $\mathcal{H}(M) = h$



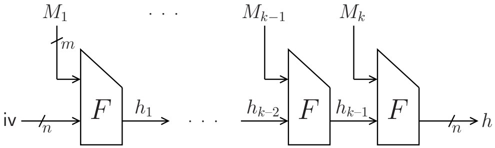
- Then, $F(h_k, \mathrm{len}(M)) = h$
- We can find F-preimage for h

Second Preimage Security of Merkle-Damgård

- Second preimage for $\mathcal{H}$: easier than for F
- Kelsey and Schneier [KS05]

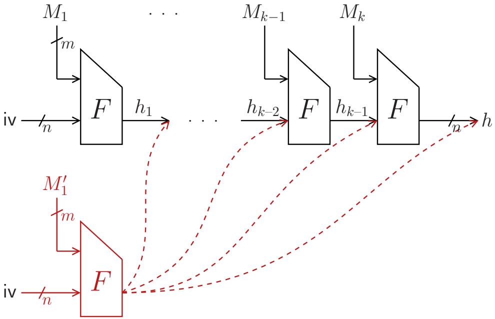
Second Preimage Security of Merkle-Damgård

- Second preimage for $\mathcal{H}$: easier than for F
- Kelsey and Schneier [KS05]
  - Assume F is an ideal function
  - Let M be any preimage for $\mathcal{H}$ (forget about $\mathrm{len}(M)$)



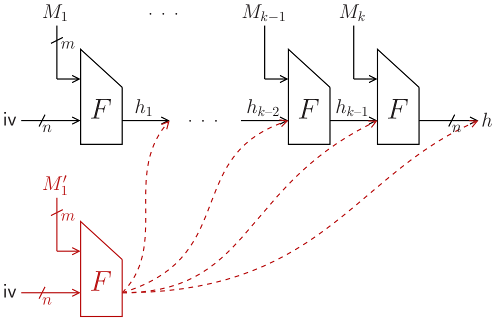
Second Preimage Security of Merkle-Damgård

- Second preimage for $\mathcal{H}$: easier than for F
- Kelsey and Schneier [KS05]
  - Assume F is an ideal function
  - Let M be any preimage for $\mathcal{H}$ (forget about $\text{len}(M)$)
  - Second preimage attack complexity $\approx 2^n/k$



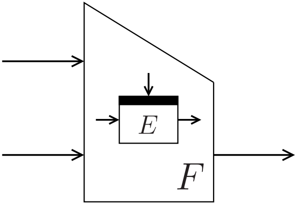
Second Preimage Security of Merkle-Damgård

- Second preimage for $\mathcal{H}$: easier than for F
- Kelsey and Schneier [KS05]
  - Assume F is an ideal function
  - Let M be any preimage for $\mathcal{H}$ (forget about $\mathrm{len}(M)$)
  - Second preimage attack complexity $\approx 2^n/k$
  - Work-around trick if $\mathrm{len}(M)$ is included

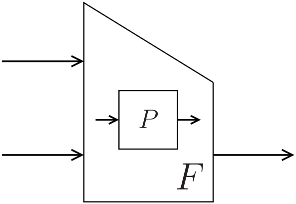


Compression Functions from Blockciphers/Permutations

Blockcipher Based

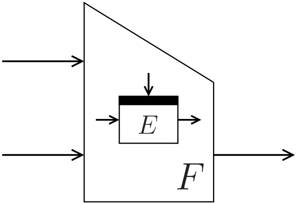


Permutation Based

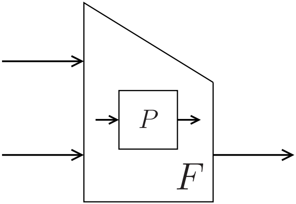


Compression Functions from Blockciphers/Permutations

Blockcipher Based



Permutation Based



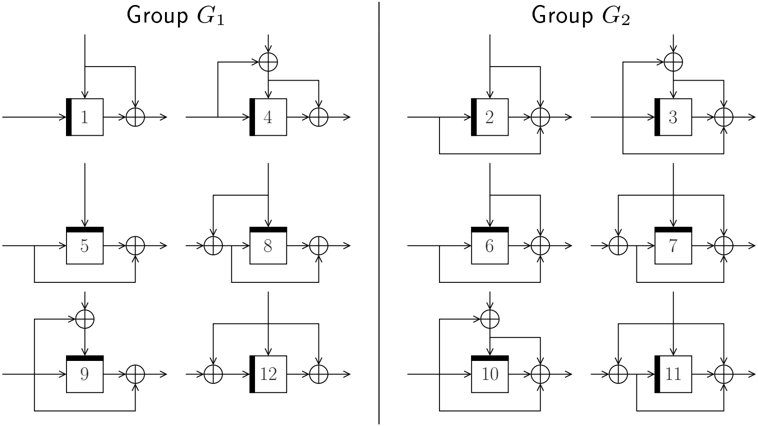
- Proofs in ideal model
  - E or P assumed to be uniformly random primitives
  - Proof via combinatorics and probability theory

Compression Functions from Blockciphers

- Classical approach
- PGV compression functions [PGV93]
- Used in MD5, SHA-1/2, ...

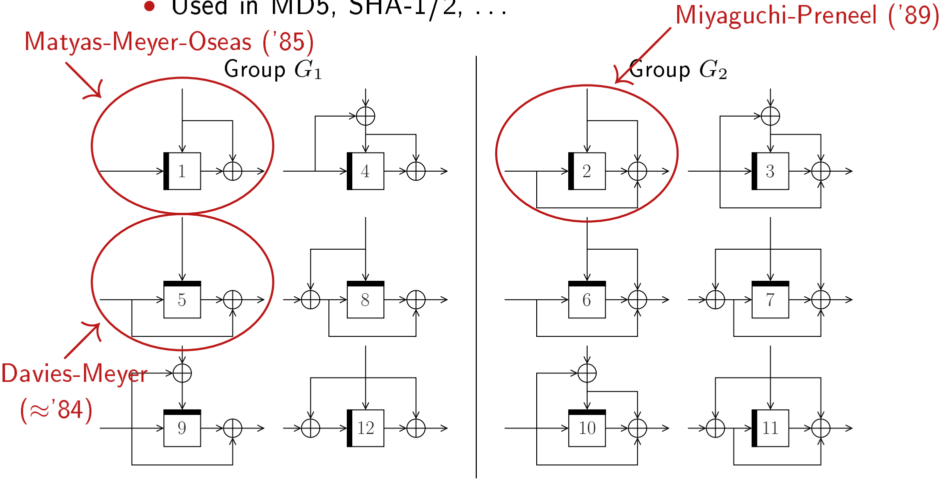
Compression Functions from Blockciphers

- Classical approach
- PGV compression functions [PGV93]
- Used in MD5, SHA-1/2, ...

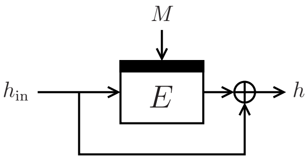


Compression Functions from Blockciphers

- Classical approach
- PGV compression functions [PGV93]
- Used in MD5, SHA-1/2, ...

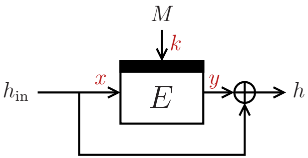


Security of Davies-Meyer Compression Function



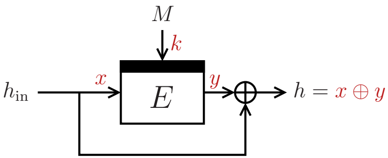
- Black et al. [BRS02]
- Assume that E is an ideal cipher

Security of Davies-Meyer Compression Function



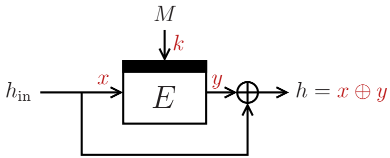
- Black et al. [BRS02]
- Assume that E is an ideal cipher
- Adversary $\mathcal{A}$ makes Q queries to E
  - Forward query $(k, x) \to y$
  - Inverse query $(k, y) \to x$

Security of Davies-Meyer Compression Function



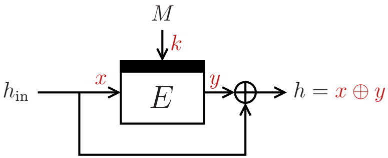
- Black et al. [BRS02]
- Assume that E is an ideal cipher
- Adversary $\mathcal{A}$ makes Q queries to E
  - Forward query $(k, x) \to y$
  - Inverse query $(k, y) \to x$
- Query (k, x, y) corresponds to $\mathsf{DM}(x, k) = x \oplus y$

Security of Davies-Meyer Compression Function



- Black et al. [BRS02]
- Assume that E is an ideal cipher
- Adversary $\mathcal{A}$ makes Q queries to E
  - Forward query $(k, x) \to y$
  - Inverse query $(k, y) \to x$
- Query (k, x, y) corresponds to $\mathsf{DM}(x, k) = x \oplus y$
- $\mathcal{A}$ must make the required queries for the attack

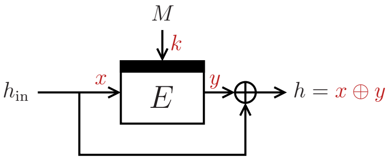
Security of Davies-Meyer Compression Function



Collision Resistance

- Consider ith query (k, x, y) (forward or inverse)

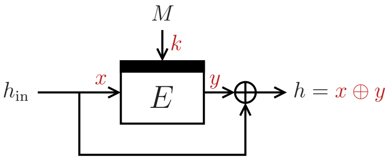
Security of Davies-Meyer Compression Function



Collision Resistance

- Consider ith query (k, x, y) (forward or inverse)
  - Response is random from set of size at least $2^n - (i-1)$

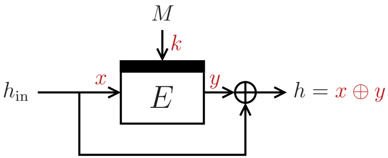
Security of Davies-Meyer Compression Function



Collision Resistance

- Consider ith query (k, x, y) (forward or inverse)
  - Response is random from set of size at least $2^n - (i-1)$
  - Renders collision if $x \oplus y = x' \oplus y'$ for any earlier (k', x', y')

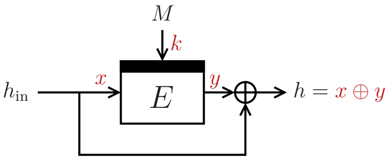
Security of Davies-Meyer Compression Function



Collision Resistance

- Consider ith query (k, x, y) (forward or inverse)
  - Response is random from set of size at least $2^n - (i - 1)$
  - Renders collision if $x \oplus y = x' \oplus y'$ for any earlier (k', x', y')
  - There are $i - 1$ earlier queries (k', x', y')

Security of Davies-Meyer Compression Function

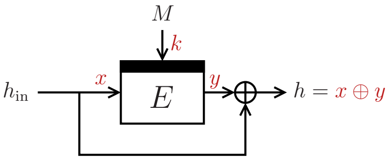


Collision Resistance

- Consider ith query (k, x, y) (forward or inverse)
  - Response is random from set of size at least $2^n - (i-1)$
  - Renders collision if $x \oplus y = x' \oplus y'$ for any earlier (k', x', y')
  - There are $i-1$ earlier queries (k', x', y')

$$\mathbf{Pr}\left[\text{collision for DM}\right] \leq \sum_{i=1}^{Q} \frac{i-1}{2^n - (i-1)}$$

Security of Davies-Meyer Compression Function

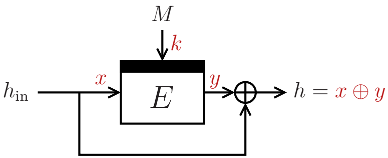


Collision Resistance

- Consider ith query (k, x, y) (forward or inverse)
  - Response is random from set of size at least $2^n - (i-1)$
  - Renders collision if $x \oplus y = x' \oplus y'$ for any earlier (k', x', y')
  - There are $i-1$ earlier queries (k', x', y')

$$\mathbf{Pr}\left[\text{collision for DM}\right] \leq \sum_{i=1}^{Q} \frac{i-1}{2^n - (i-1)} \leq \frac{\binom{Q}{2}}{2^n - Q}$$

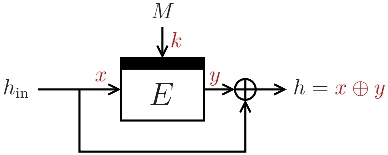
Security of Davies-Meyer Compression Function



Preimage Resistance

- Consider any fixed target value h
- Consider ith query (k, x, y) (forward or inverse)

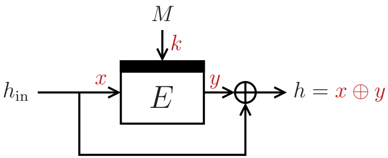
Security of Davies-Meyer Compression Function



Preimage Resistance

- Consider any fixed target value h
- Consider ith query (k, x, y) (forward or inverse)
  - Response is random from set of size at least $2^n - (i-1)$
  - Renders preimage if $x \oplus y = h$

Security of Davies-Meyer Compression Function

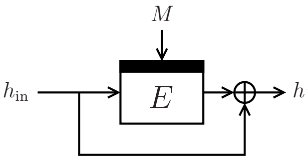


Preimage Resistance

- Consider any fixed target value h
- Consider ith query (k, x, y) (forward or inverse)
  - Response is random from set of size at least $2^n - (i-1)$
  - Renders preimage if $x \oplus y = h$

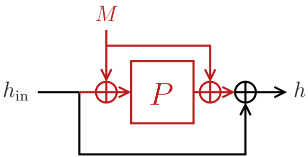
$$\mathbf{Pr}\,[\text{preimage for DM}] \leq \sum_{i=1}^{Q} \frac{1}{2^n - (i-1)} \leq \frac{Q}{2^n - Q}$$

Security of Davies-Meyer Compression Function



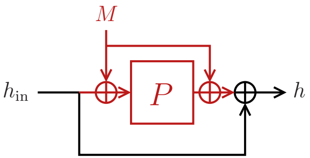
- Can we prove security if E is not ideal?
- What if E is SPRP?

Security of Davies-Meyer Compression Function



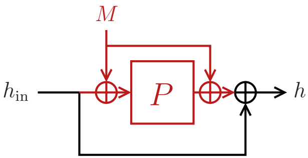
- Can we prove security if E is not ideal?
- What if E is SPRP?
  - Even-Mansour is SPRP

Security of Davies-Meyer Compression Function



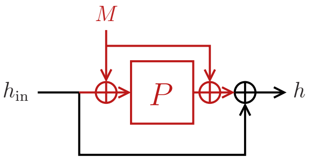
- Can we prove security if E is not ideal?
- What if E is SPRP?
  - Even-Mansour is SPRP
  - But $\mathsf{DM}^{\mathsf{EM}}(h_{\text{in}}, M) = \mathsf{DM}^{\mathsf{EM}}(h_{\text{in}} \oplus \delta, M \oplus \delta)$

Security of Davies-Meyer Compression Function



- Can we prove security if E is not ideal?
- What if E is SPRP?
  - Even-Mansour is SPRP
  - But $\mathsf{DM}^{\mathsf{EM}}(h_{\text{in}}, M) = \mathsf{DM}^{\mathsf{EM}}(h_{\text{in}} \oplus \delta, M \oplus \delta)$
  - E is used as keyless primitive

Security of Davies-Meyer Compression Function



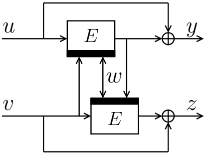
- Can we prove security if E is not ideal?
- What if E is SPRP?
  - Even-Mansour is SPRP
  - But $\mathsf{DM}^{\mathsf{EM}}(h_{\text{in}}, M) = \mathsf{DM}^{\mathsf{EM}}(h_{\text{in}} \oplus \delta, M \oplus \delta)$
  - E is used as keyless primitive
- Bottom line:
  - Be careful with choice of security model
  - Be careful with instantiation of the construction

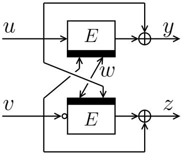
Compression Functions from Blockciphers

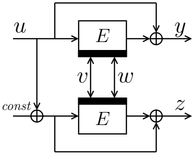
- Similar ideas for other PGVs

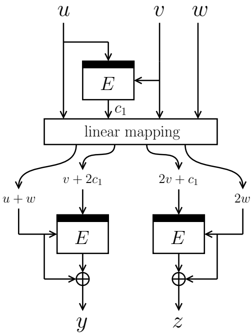
Compression Functions from Blockciphers

- Similar ideas for other PGVs
- Additional proof tricks for double length functions



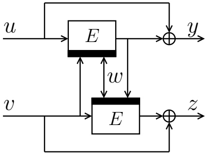


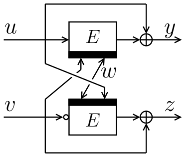


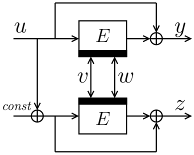


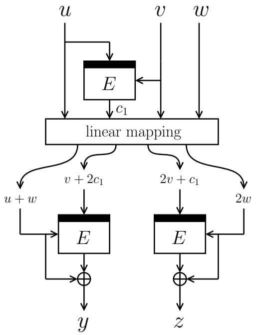
Compression Functions from Blockciphers

- Similar ideas for other PGVs
- Additional proof tricks for double length functions
  - "Wish lists" [LSS10]
  - "Free super queries" [LSS11]
  - Many case distinctions (up to ≈ 40 in worst case)









Compression Functions from Permutations

- Blockcipher calls require re-keying

Compression Functions from Permutations

- Blockcipher calls require re-keying
- Instead use fixed-key blockciphers, or permutations

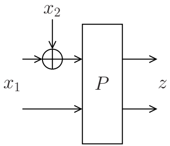
Compression Functions from Permutations

- Blockcipher calls require re-keying
- Instead use fixed-key blockciphers, or permutations
  - Formalized by Black et al. [BCS05]

Compression Functions from Permutations

- Blockcipher calls require re-keying
- Instead use fixed-key blockciphers, or permutations
  - Formalized by Black et al. [BCS05]

Sponge ('07)

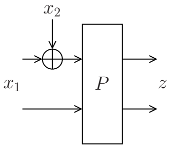


collision: 1 query
preimage: 1 query

Compression Functions from Permutations

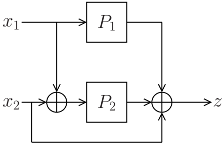
- Blockcipher calls require re-keying
- Instead use fixed-key blockciphers, or permutations
  - Formalized by Black et al. [BCS05]

Sponge ('07)



collision: 1 query
preimage: 1 query

Grøstl ('09)

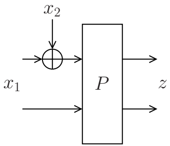


collision: $2^{n/4}$ queries
preimage: $2^{n/2}$ queries

Compression Functions from Permutations

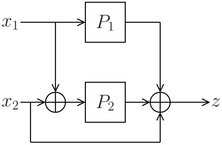
- Blockcipher calls require re-keying
- Instead use fixed-key blockciphers, or permutations
  - Formalized by Black et al. [BCS05]

Sponge ('07)



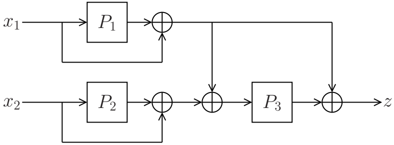
collision: 1 query
preimage: 1 query

Grøstl ('09)



collision: $2^{n/4}$ queries
preimage: $2^{n/2}$ queries

Shrimpton-Stam ('08)

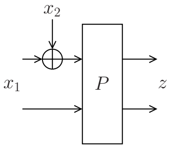


collision: $2^{n/2}$ queries
preimage: $2^{2n/3}$ queries

Compression Functions from Permutations

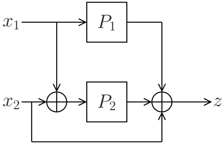
- Blockcipher calls require re-keying
- Instead use fixed-key blockciphers, or permutations
  - Formalized by Black et al. [BCS05]
  - More primitive calls needed
  - Similar proof techniques but more cases

Sponge ('07)



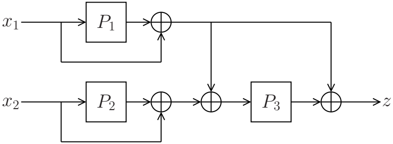
collision: 1 query
preimage: 1 query

Grøstl ('09)



collision: $2^{n/4}$ queries
preimage: $2^{n/2}$ queries

Shrimpton-Stam ('08)

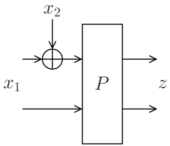


collision: $2^{n/2}$ queries
preimage: $2^{2n/3}$ queries

Compression Functions from Permutations

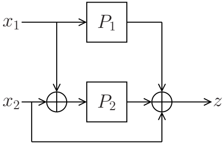
- Blockcipher calls require re-keying
- Instead use fixed-key blockciphers, or permutations
  - Formalized by Black et al. [BCS05]
  - More primitive calls needed
  - Similar proof techniques but more cases

Sponge ('07)



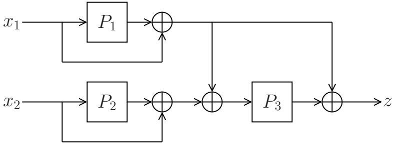
collision: 1 query
preimage: 1 query

Grøstl ('09)



collision: $2^{n/4}$ queries
preimage: $2^{n/2}$ queries

Shrimpton-Stam ('08)



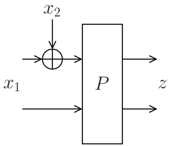
collision: $2^{n/2}$ queries
preimage: $2^{2n/3}$ queries

are the Sponges insecure?

Compression Functions from Permutations

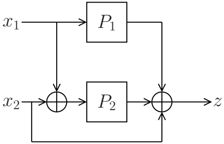
- Blockcipher calls require re-keying
- Instead use fixed-key blockciphers, or permutations
  - Formalized by Black et al. [BCS05]
  - More primitive calls needed
  - Similar proof techniques but more cases

Sponge ('07)



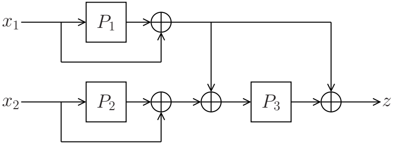
collision: 1 query
preimage: 1 query

Grøstl ('09)



collision: $2^{n/4}$ queries
preimage: $2^{n/2}$ queries

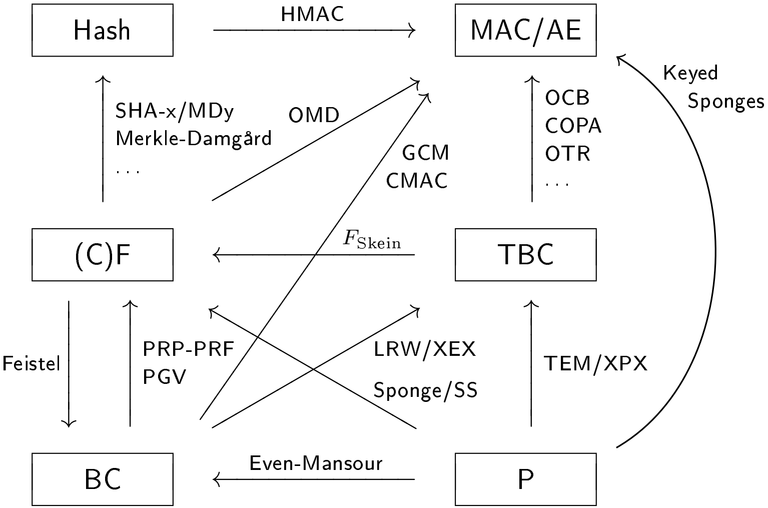
Shrimpton-Stam ('08)



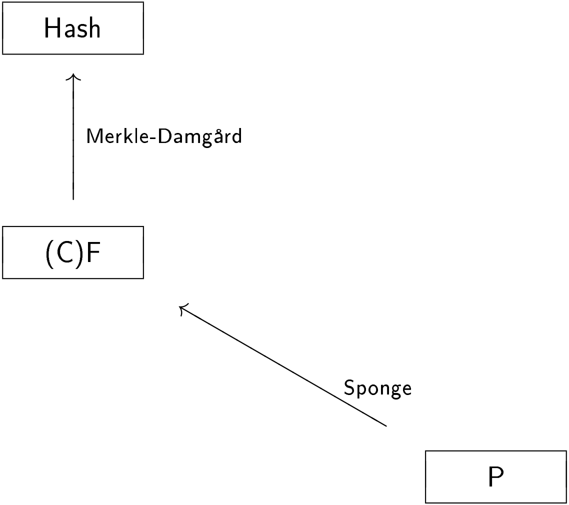
collision: $2^{n/2}$ queries
preimage: $2^{2n/3}$ queries

are the Sponges insecure? No ☺!

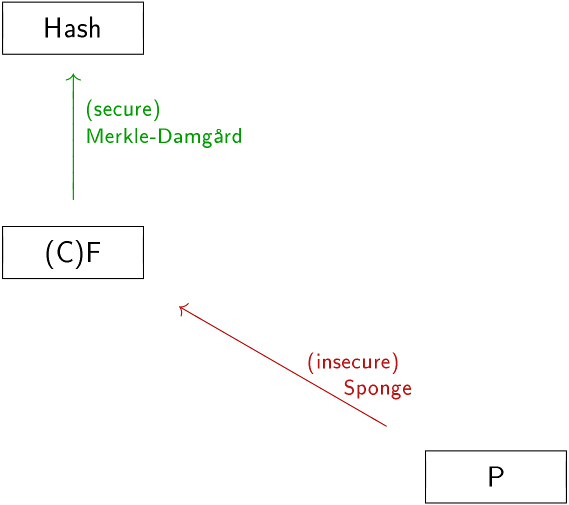
Security of Sponge



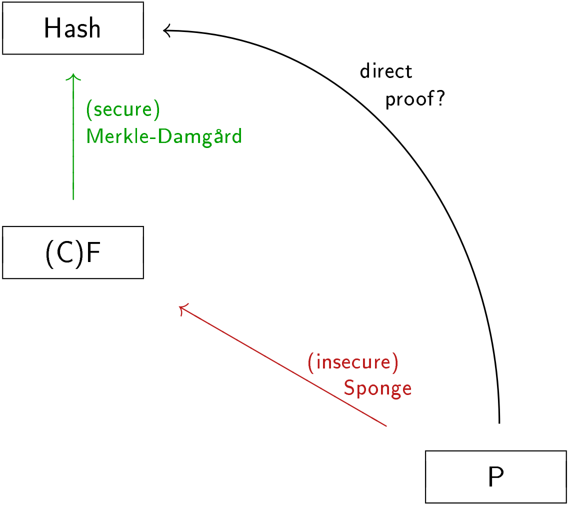
Security of Sponge



Security of Sponge



Security of Sponge

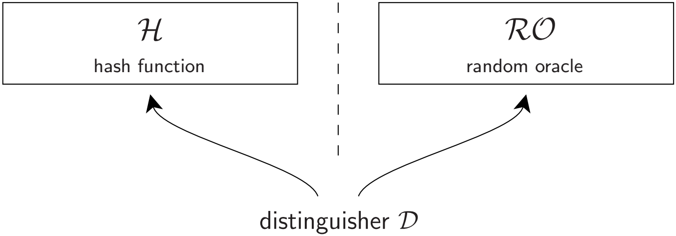


Indistinguishability of Hash Functions

- $\mathcal{H}$ should behave like random oracle $\mathcal{RO}$

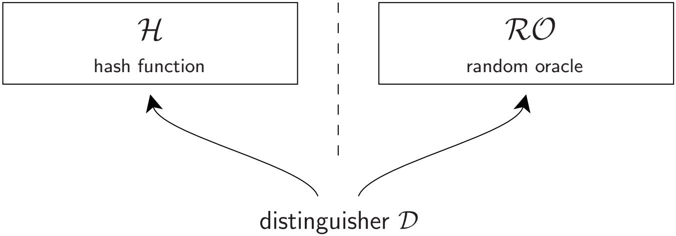
Indistinguishability of Hash Functions

- $\mathcal{H}$ should behave like random oracle $\mathcal{RO}$
- Indistinguishability of random systems:



Indistinguishability of Hash Functions

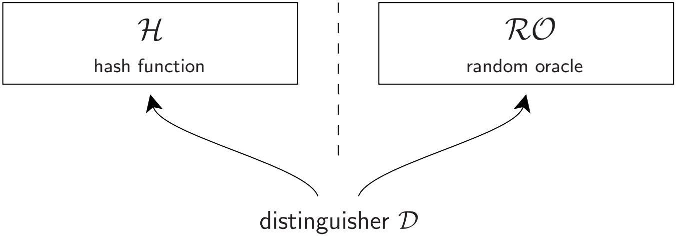
- $\mathcal{H}$ should behave like random oracle $\mathcal{RO}$
- Indistinguishability of random systems:



- But $\mathcal{H}$ is not a random system
- Distinguisher succeeds with probability 1

Indistinguishability of Hash Functions

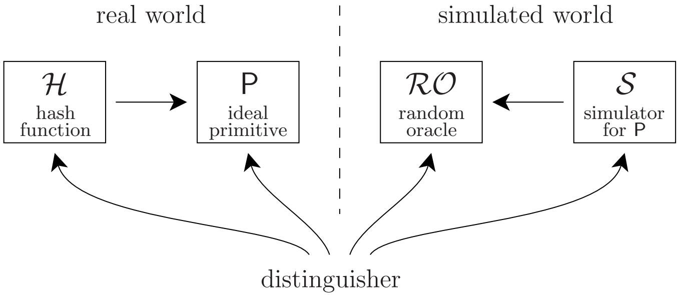
- $\mathcal{H}$ should behave like random oracle $\mathcal{RO}$
- Indistinguishability of random systems:



- But $\mathcal{H}$ is not a random system
- Distinguisher succeeds with probability 1

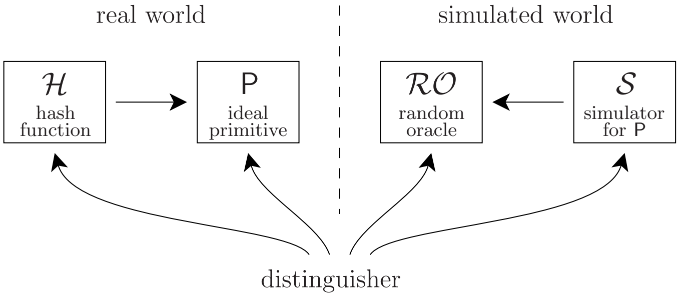
- Solution: indifferentiability

Indifferentiability of Hash Functions



- Maurer et al. [MRH04]

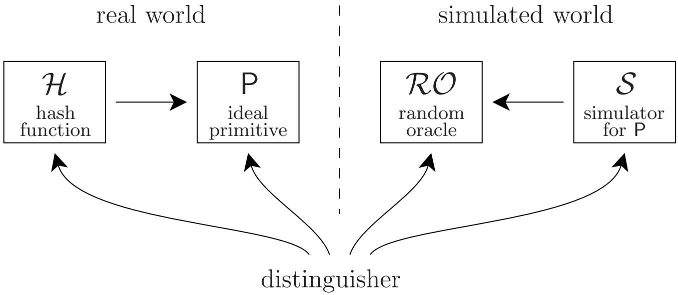
Indifferentiability of Hash Functions



- Maurer et al. [MRH04]
- $\mathcal{H}$ is indifferentiable from $\mathcal{RO}$ if for some simulator $\mathcal{S}$:

$$\Delta_{\mathcal{D}}(\mathcal{H}, \mathsf{P}; \mathcal{RO}, \mathcal{S}) \text{ is small}$$

Indifferentiability of Hash Functions



- Maurer et al. [MRH04]
- $\mathcal{H}$ is indifferentiable from $\mathcal{RO}$ if for some simulator $\mathcal{S}$:

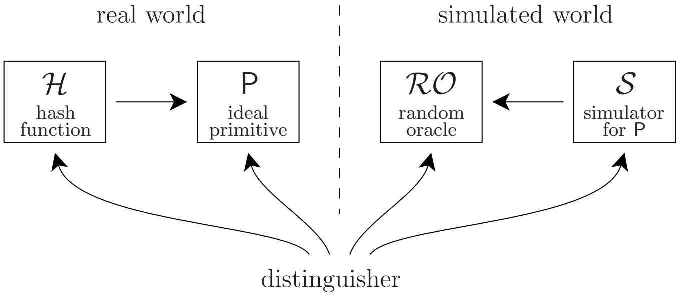
$$\Delta_{\mathcal{D}}(\mathcal{H}, \mathsf{P}; \mathcal{RO}, \mathcal{S}) \text{ is small}$$

- Proof idea:

  Step 1. Construct a clever simulator $\mathcal{S}$

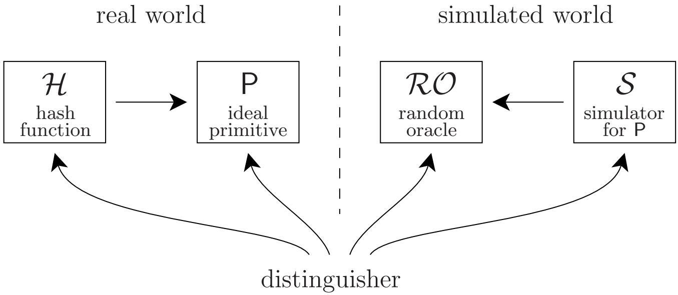
  Step 2. Use game-playing or H-coefficient technique

Indifferentiability of Hash Functions



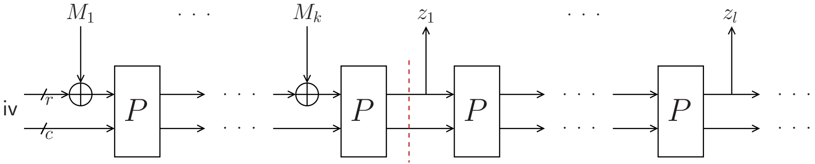
- $\mathcal{H}$ can replace $\mathcal{RO}$ in certain scenarios
    - Single-stage only, Ristenpart et al. [RSS11]

Indifferentiability of Hash Functions



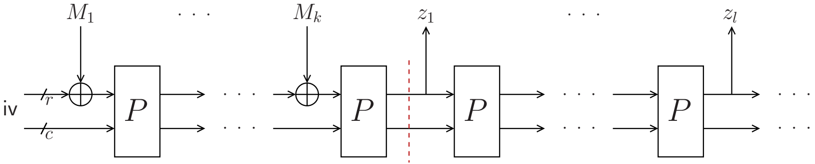
- $\mathcal{H}$ can replace $\mathcal{RO}$ in certain scenarios
  - Single-stage only, Ristenpart et al. [RSS11]
- Indifferentiability $\Longrightarrow$ coll/pre/sec security

Indifferentiability of Sponge



- Bertoni et al. [BDPA07]

Indifferentiability of Sponge

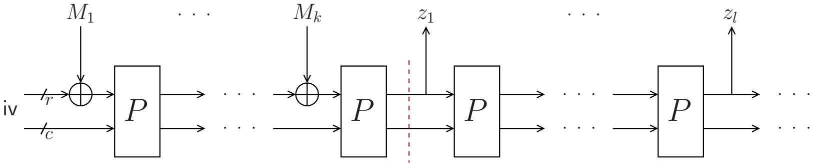


- Bertoni et al. [BDPA07]
- Sponge indifferentiable from random oracle

$$\Delta_{\mathcal{D}}(\text{Sponge}, P; \mathcal{RO}, \mathcal{S}) \leq \frac{Q^2}{2^c}$$

(with Q the total complexity)

Indifferentiability of Sponge

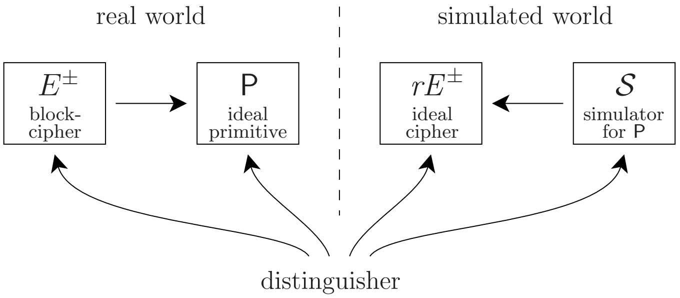


- Bertoni et al. [BDPA07]
- Sponge indifferentiable from random oracle

$$\Delta_{\mathcal{D}}(\text{Sponge}, P; \mathcal{RO}, \mathcal{S}) \leq \frac{Q^2}{2^c}$$

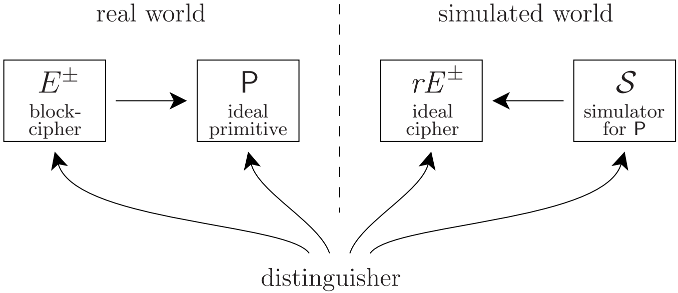
  (with Q the total complexity)
- Optimal collision, preimage, second preimage security (if $c \geq 2n$)

Indifferentiability of Blockciphers



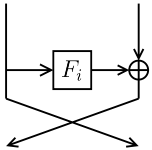
- Similar model for keyless blockciphers
- Blockcipher behaves like ideal cipher
- Can be plugged into PGV compression functions

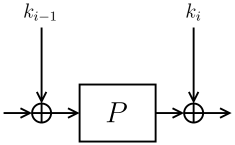
Indifferentiability of Blockciphers



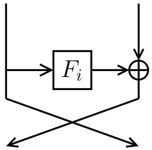
- Similar model for keyless blockciphers
- Blockcipher behaves like ideal cipher
- Can be plugged into PGV compression functions
- Much (!!) harder to prove

Indifferentiability of Blockciphers

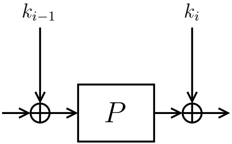




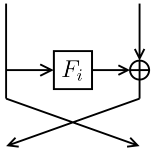
Indifferentiability of Blockciphers



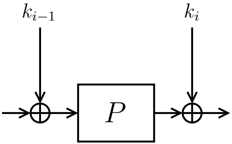
Feistel	bound	remark
Coron et al. '08	$2^{18}\ q^{8}\ /2^n$	6 rnd (flawed)
Holenstein et al. '10	$2^{66}\ q^{10}/2^n$	14 rnd
Guo and Lin '15	$2^{222}q^{30}/2^n$	21 rnd (alter. key)
Dachman-Soled et al. '15	$2^{51}\ q^{12}/2^n$	10 rnd
Dai and Steinberger '15	$2^{23}\ q^{8}\ /2^n$	10 rnd



Indifferentiability of Blockciphers

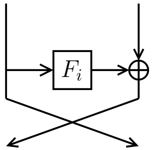


Feistel	bound	remark
Coron et al. '08	$2^{18}q^{8}/2^n$	6 rnd (flawed)
Holenstein et al. '10	$2^{66}q^{10}/2^n$	14 rnd
Guo and Lin '15	$2^{222}q^{30}/2^n$	21 rnd (alter. key)
Dachman-Soled et al. '15	$2^{51}q^{12}/2^n$	10 rnd
Dai and Steinberger '15	$2^{23}q^{8}/2^n$	10 rnd

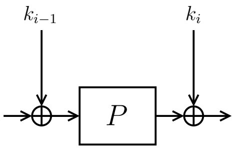


Even-Mansour	bound	remark
Andreeva et al. '13	$2^{34}q^{10}/2^n$	5 rnd (random kdf)
Lampe and Seurin '13	$2^{91}q^{12}/2^n$	12 rnd
Guo and Lin '15	$2^{11}q^{8}/2^n$	15 rnd (alter. key)

Indifferentiability of Blockciphers



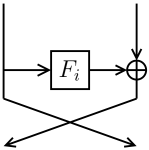
Feistel	bound	remark
Coron et al. '08	$2^{18}q^8/2^n$	6 rnd (flawed)
Holenstein et al. '10	$2^{66}q^{10}/2^n$	14 rnd
Guo and Lin '15	$2^{222}q^{30}/2^n$	21 rnd (alter. key)
Dachman-Soled et al. '15	$2^{51}q^{12}/2^n$	10 rnd
Dai and Steinberger '15	$2^{23}q^8/2^n$	10 rnd



Even-Mansour	bound	remark
Andreeva et al. '13	$2^{34}q^{10}/2^n$	5 rnd (random kdf)
Lampe and Seurin '13	$2^{91}q^{12}/2^n$	12 rnd
Guo and Lin '15	$2^{11}q^8/2^n$	15 rnd (alter. key)

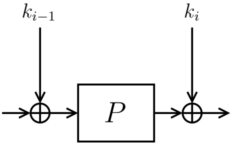
Extremely hard research question!

Indifferentiability of Blockciphers



Feistel	bound	remark
Coron et al. '08	$2^{18} q^{8} / 2^{n}$	6 rnd (flawed)
Holenstein et al. '10	$2^{66} q^{10} / 2^{n}$	14 rnd
Guo and Lin '15	$2^{222} q^{30} / 2^{n}$	21 rnd (alter. key)
Dachman-Soled et al. '15	$2^{51} q^{12} / 2^{n}$	10 rnd
Dai and Steinberger '15	$2^{23} q^{8} / 2^{n}$	10 rnd

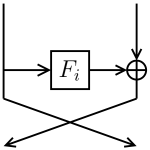
pointless for $n = 128$; security up to $q \lesssim 2$ for $n = 256$



Even-Mansour	bound	remark
Andreeva et al. '13	$2^{34} q^{10} / 2^{n}$	5 rnd (random kdf)
Lampe and Seurin '13	$2^{91} q^{12} / 2^{n}$	12 rnd
Guo and Lin '15	$2^{11} q^{8} / 2^{n}$	15 rnd (alter. key)

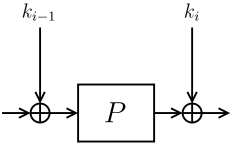
Extremely hard research question!

Indifferentiability of Blockciphers



Feistel	bound	remark
Coron et al. '08	$2^{18}\ q^{8}\ /2^n$	6 rnd (flawed)
Holenstein et al. '10	$2^{66}\ q^{10}/2^n$	14 rnd
Guo and Lin '15	$2^{222}q^{30}/2^n$	21 rnd (alter. key)
Dachman-Soled et al. '15	$2^{51}\ q^{12}/2^n$	10 rnd
Dai and Steinberger '15	$2^{23}\ q^{8}\ /2^n$	10 rnd

pointless for $n = 128$; security up to $q \lesssim 2$ for $n = 256$



Even-Mansour	bound	remark
Andreeva et al. '13	$2^{34}q^{10}/2^n$	5 rnd (random kdf)
Lampe and Seurin '13	$2^{91}q^{12}/2^n$	12 rnd
Guo and Lin '15	$2^{11}q^{8}\ /2^n$	15 rnd (alter. key)

security up to $q \lesssim 8$ for $n = 128$

Extremely hard research question!

Conclusion

Recipe

- Model:
  - Keyed: usually indistinguishability
  - Keyless: indifferentiability or dedicated security model
- Technique: game-playing, H-coefficient,
  explicit reduction, combinatorics, ...

Conclusion

Recipe

- Model:
  - Keyed: usually indistinguishability
  - Keyless: indifferentiability or dedicated security model
- Technique: game-playing, H-coefficient,
  explicit reduction, combinatorics, ...
- Approach depends on specific scheme and application

Conclusion

Recipe

- Model:
  - Keyed: usually indistinguishability
  - Keyless: indifferentiability or dedicated security model
- Technique: game-playing, H-coefficient,
  explicit reduction, combinatorics, ...
- Approach depends on specific scheme and application

Thank you for your attention!