Full Disk Encryption and Beyond

Monday, 15 July 2019

Louiza Khati

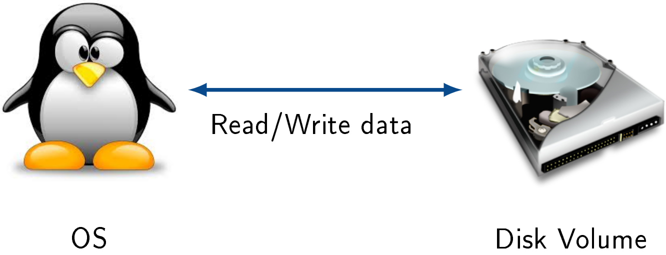
Outline

- Part 1. From disk storage to security models
- Part 2. Key dependent-message security of Even-Mansour ciphers
- Part 3. Incremental MACs

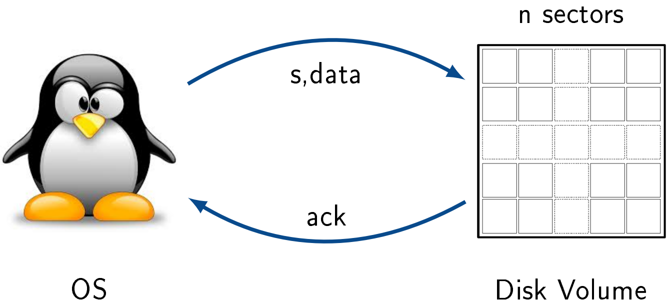
Part 1.

From Disk Storage to Security Models

Disk Storage

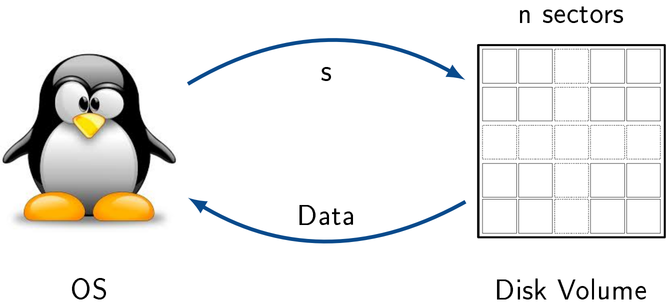


Disk Storage: Write



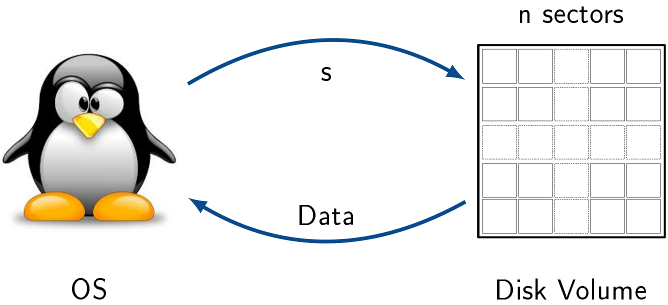
- Write data in disk
- Store data

Disk Storage: Read



- Read data in disk
- Give back data

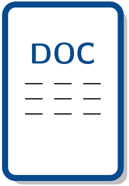
Disk Storage: Performance



- Read/write speed is a priority (optimized)
- Competitive aspect for manufacturers

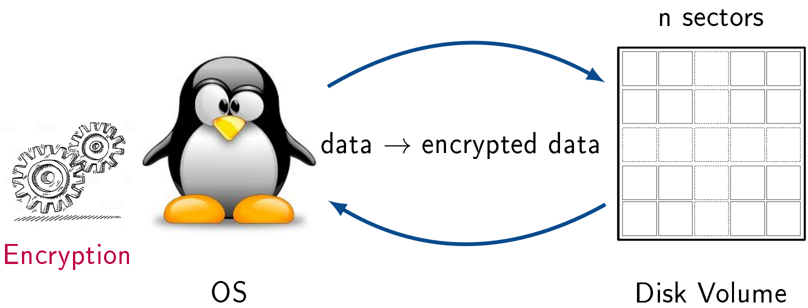
Full Disk Encryption VS File Encryption

- File encryption
  - File content is encrypted
    - Title, file size encrypted?
  - User action
    - Ask to encrypt a specific file
  - Space for metadata
    - Better security using IV
    - Integrity



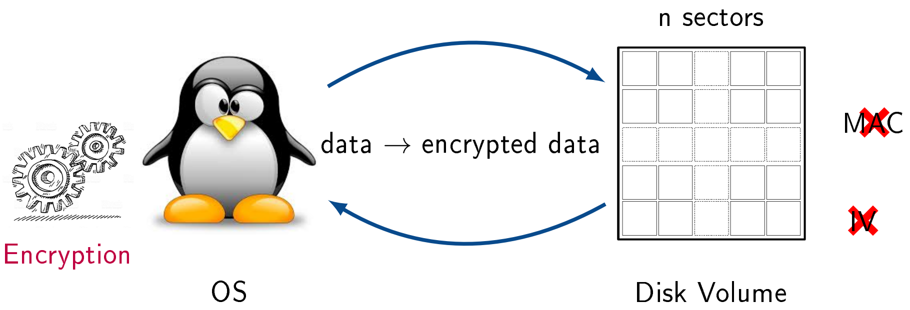
- Full Disk Encryption
  - All the data are encrypted
    - Sector-based encryption
  - Transparent for the user
    - Automatic
  - No space for metadata
    - No IVs
    - No Integrity

Full Disk Encryption (FDE)

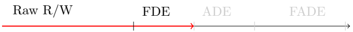


- Read and write: atomic operations
  - A sector is encrypted independently from the others

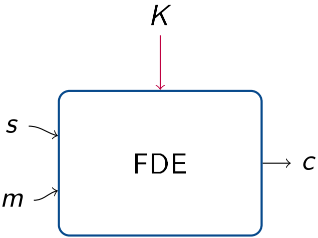
Full Disk Encryption (FDE)



- Length preserving encryption (no metadata)
- Deterministic encryption

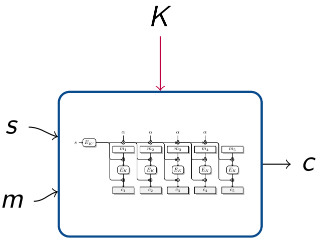


FDE schemes



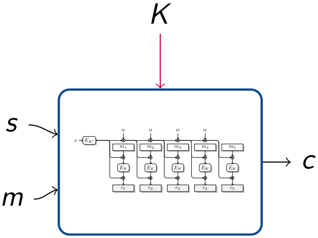
- Symmetric encryption (speed)
  - Blockciphers (AES)
  - Sector size > blockcipher input size

FDE schemes



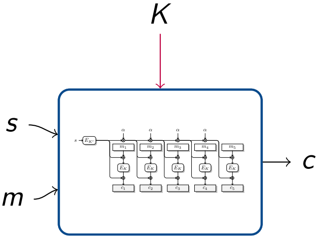
- Symmetric encryption (speed)
  - Blockciphers (AES)
  - Sector size > blockcipher input size
- FDE Modes of operation
  - Length preserving modes
  - Tweak s used to enhance security

FDE schemes



- Symmetric encryption (speed)
  - Blockciphers (AES)
  - Sector size > blockcipher input size
- FDE Modes of operation
  - Length preserving modes
  - Tweak s used to enhance security
- Security proofs [K., Mouha, Vergnaud]
  - Reduction to blockcipher security
  - Different security notions

FDE schemes



- Symmetric encryption (speed)
  - Blockciphers (AES)
  - Sector size > blockcipher input size
- FDE Modes of operation
  - Length preserving modes
  - Tweak s used to enhance security
- Security proofs [K., Mouha, Vergnaud]
  - Reduction to blockcipher security
  - Different security notions
- Examples (`dm-crypt`)
  - CBC-ESSIV
  - XTS (based on XEX)
  - Adiantum (new)

FDE tools: no control of what is stored!

Full Disk Encryption and KDM security

- Atypical scenario can happen
  - The key can be stored in the disk
  - A (weird) function of the key can be stored

↓

Key-Dependent Message security Model

- Security analysis with an adversary that can ask to encrypt the key
- Key-Alternating Feistel ciphers [Farshim, K., Seurin, Vergnaud]
- Even-Mansour ciphers [Farshim, K., Vergnaud]

Part 2.

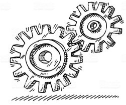
Incremental MACs and "FDE"

Integrity $\rightarrow$ outside "FDE" Model!
How to get integrity with a minimal impact on performance?

- Authenticated Disk Encryption (ADE)
  - Ensures sector content integrity
  - MAC for each sector (a local tag/sector)
  - `dm-integrity` (Linux Kernel)

Authenticated Disk Encryption

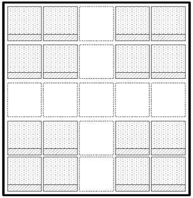
Authenticated Decryption



OS

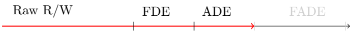
- Read a sector in disk

n sectors



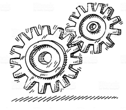
Disk Volume

- Give back sector content



Authenticated Disk Encryption

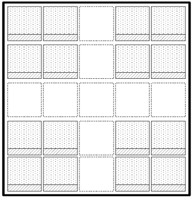
Authenticated Encryption



OS

- Write a sector in disk

n sectors

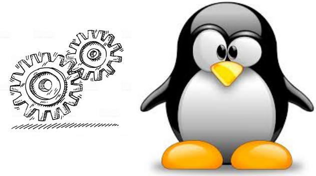


Disk Volume

- Store sector content

Raw R/W FDE ADE FADE

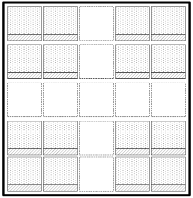
Authenticated Disk Encryption



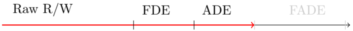
OS

Confidentiality
+
Integrity

n sectors



Disk Volume



Incremental MACs and “FDE”

Integrity $\rightarrow$ outside “FDE” Model!
How to get integrity with a minimal impact on performance?

- Authenticated Disk Encryption (ADE)
  - Ensures sector content integrity,
  - MAC for each sector (a local tag/sector)
  - dm-integrity (Linux Kernel)

Does not prevent replay-attacks!

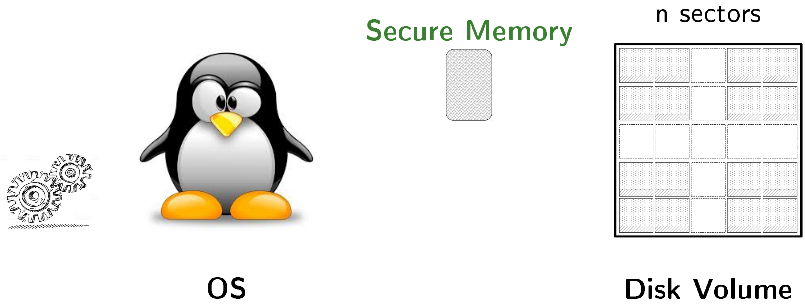
Incremental MACs and “FDE”

Integrity $\rightarrow$ outside “FDE” Model!
How to get integrity with a minimal impact on performance?

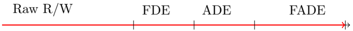
- Authenticated Disk Encryption (ADE)
  - Ensures sector content integrity,
  - MAC for each sector (a local tag/sector)
  - dm-integrity (Linux Kernel)

  Does not prevent replay-attacks!
- Fully Authenticated Disk Encryption (FADE)
  - Prevent replay-attacks
  - Ensures local tags integrity
  - MAC over all the local tags (global tag/disk)

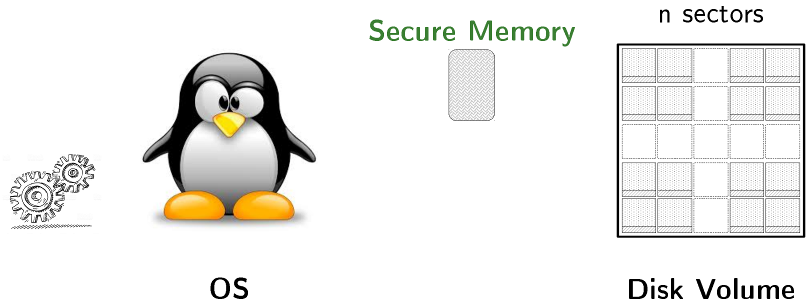
Fully Authenticated Disk Encryption



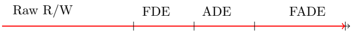
- Global tag = MAC over local tags
- Global tag in Secure memory (small)
- MAC is too expensive



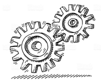
Fully Authenticated Disk Encryption



- Global tag = MAC over local tags
- Global tag in Secure memory (small)
- ~~MAC is too expensive~~ Incremental MACs



Fully Authenticated Disk Encryption

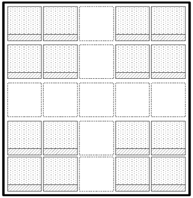


OS

Secure Memory

Confidentiality
+
Integrity
+
Anti-replay

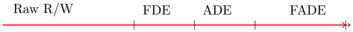
n sectors



Disk Volume

- Global tag = MAC over local tags
- Global tag in Secure memory (small)
- ~~MAC is too expensive~~ Incremental MACs

Part 3.



Part 2.

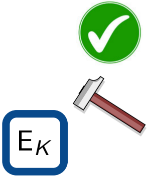
Key-Dependent Message (KDM) Security
Even-Mansour Ciphers

Security Analysis

- Robustness against an arbitrary adversary?

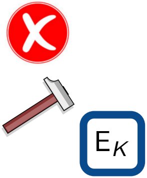
Security Analysis

- Robustness against an arbitrary adversary?
- Robustness against specific attacks?
  - Specific to a blockcipher and not enough



Security Analysis

- Robustness against an arbitrary adversary?
- Robustness against specific attacks?
  - Specific to a blockcipher and not enough



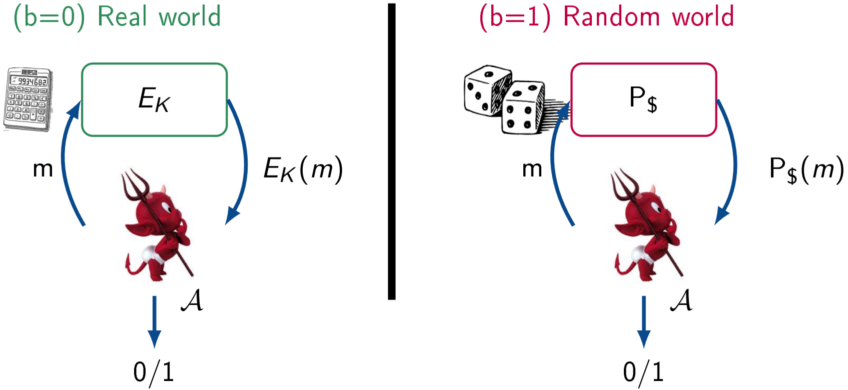
Security Analysis

- Robustness against an arbitrary adversary?
- Robustness against specific attacks?
  - Specific to a blockcipher and not enough
- Robustness against generic attacks?
  - Feasible: Internal primitives idealized

Security Analysis

- Robustness against an arbitrary adversary?
- Robustness against specific attacks?
  - Specific to a blockcipher and not enough
- Robustness against generic attacks?
  - Feasible: Internal primitives idealized
- Security Proof
  - Modeled by a game: adversary/challenger
  - Adversary model (power)

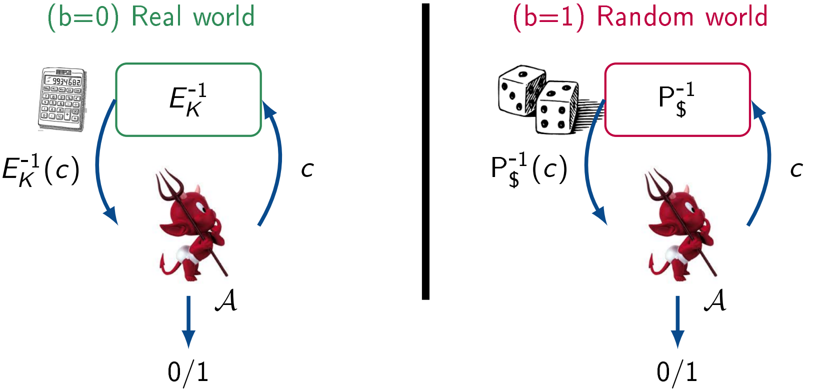
Indistinguishability game



Chosen Plaintext Attack (CPA) adversary

$$\textbf{Adv} = |\ \Pr[\mathcal{A} \rightarrow 1 | \text{Real}]\ \text{-}\ \Pr[\mathcal{A} \rightarrow 1 | \text{Random}]\ |$$

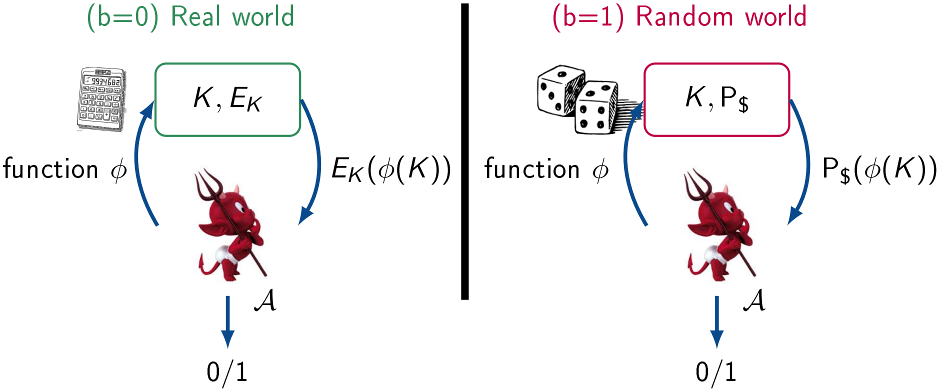
Indistinguishability game



Chosen Ciphertext Attack (CCA) adversary

$$\mathbf{Adv} = |\ \Pr[\mathcal{A} \rightarrow 1 | \text{Real}] - \Pr[\mathcal{A} \rightarrow 1 | \text{Random}]\ |$$

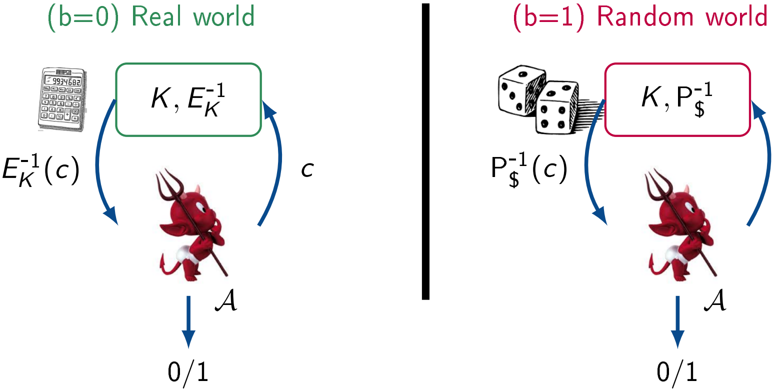
KDM security: Indistinguishability game



KDM-CPA adversary

$$\textbf{Adv} = |\ \Pr[\mathcal{A} \rightarrow 1 | \text{Real}] - \Pr[\mathcal{A} \rightarrow 1 | \text{Random}]\ |$$

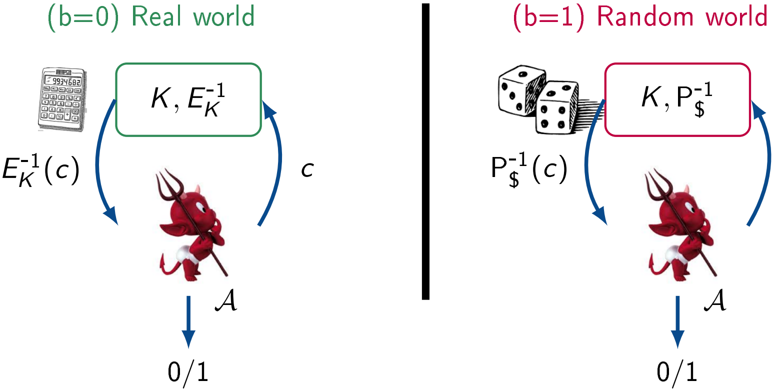
KDM security: Indistinguishability game



KDM-CCA adversary ("Standard" decryption)

$$\textbf{Adv} = |\ \Pr[\mathcal{A} \to 1 | \text{Real}] - \Pr[\mathcal{A} \to 1 | \text{Random}]\ |$$

KDM security: Indistinguishability game



Forbidden queries: Repeat queries, Enc/Dec oracle's answers

Key Dependent Message Security analysis

- Find the largest set Φ of functions ϕ such that Adv is small
  - Including constant functions
- What if Φ is not restricted?

Key Dependent Message Security analysis

- Find the largest set Φ of functions ϕ such that Adv is small
  - Including constant functions
- What if Φ is not restricted?

Example: Projections
$\phi_1(K) = (K \;\; \& \;\; 0...01)$
If $K =?? \; 1$ then $\phi_1(K) = 0...01$
If $K =?? \; 0$ then $\phi_1(K) = 0...00$

Key Dependent Message Security analysis

- Find the largest set Φ of functions ϕ such that Adv is small
  - Including constant functions
- What if Φ is not restricted?

Example: Projections
$\phi_1(K) = (K \ \ \& \ \ 0...01)$
If K =?? 1 then $\phi_1(K) = 0...01$
If K =?? 0 then $\phi_1(K) = 0...00$

Using ϕ_2 and ϕ_3 such that:
$\phi_2(K) = 0...01$
$\phi_3(K) = 0...00$

Key Dependent Message Security analysis

- Find the largest set Φ of functions ϕ such that Adv is small
  - Including constant functions
- What if Φ is not restricted?

Example: Projections
$\phi_1(K) = (K \,\&\, 0...01) \rightarrow c_1$
If $K = ??1$ then $\phi_1(K) = 0...01$
If $K = ??0$ then $\phi_1(K) = 0...00$

Using ϕ_2 and ϕ_3 such that:
$\phi_2(K) = 0...01 \rightarrow c_2$
$\phi_3(K) = 0...00 \rightarrow c_3$

Key Dependent Message Security analysis

- Find the largest set Φ of functions ϕ such that Adv is small
  - Including constant functions
- What if Φ is not restricted?

Example: Projections
$\phi_1(K) = (K \ \& \ 0...01) \rightarrow c_1$
If $K = ??1$ then $\phi_1(K) = 0...01$
If $K = ??0$ then $\phi_1(K) = 0...00$

Using ϕ_2 and ϕ_3 such that:
$\phi_2(K) = 0...01 \rightarrow c_2$
$\phi_3(K) = 0...00 \rightarrow c_3$

If $c_1 = c_2$ then
$K = ??1$ otherwise $K = ??0$

Last bit recovered!!

Key Dependent Message Security analysis

- Find the largest set Φ of functions ϕ such that Adv is small
  - Including constant functions
- What if Φ is not restricted?

Example: Projections
$\phi_1(K) = (K \ \& \ 0...01) \rightarrow c_1$
If $K = ??1$ then $\phi_1(K) = 0...01$
If $K = ??0$ then $\phi_1(K) = 0...00$

If $c_1 = c_2$ then
$K = ??1$ otherwise $K = ??0$

Last bit recovered!!

Using ϕ_2 and ϕ_3 such that:
$\phi_2(K) = 0...01 \rightarrow c_2$
$\phi_3(K) = 0...00 \rightarrow c_3$

Key bits can be recovered one by one!

KDM set Φ has to be restricted.

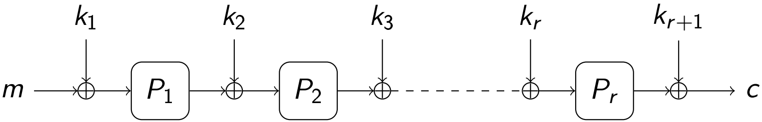
KDM set restriction: Claw-freeness

Claw-freeness of a set Φ: $\forall\ \phi_1 \neq \phi_2$, $\Pr[\phi_1(K) = \phi_2(K)]$ is small.

KDM security:

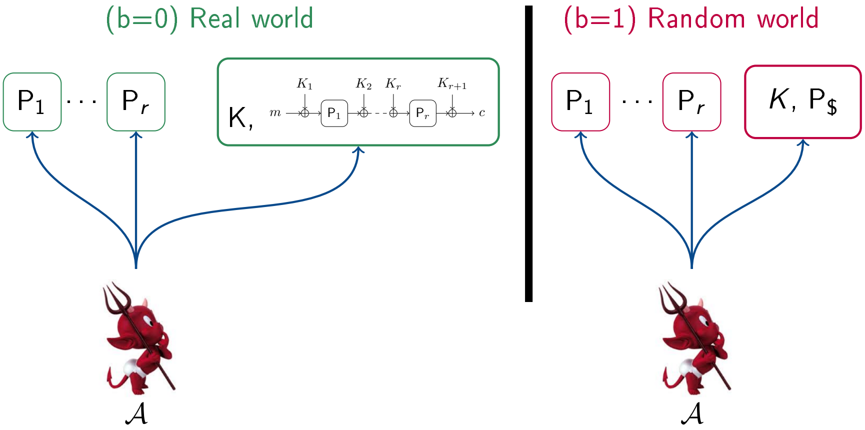
- Ideal-Cipher KDM-secure under claw-free sets
  - [Farshim, K., Vergnaud].
- What about Even-Mansour ciphers?

Even-Mansour ciphers



- Configuration:
  - r rounds
  - r permutations ($=$ or $\neq$)
  - Key schedule:
    - $r+1$ keys ($=$ or $\neq$)
    - $r+1$ keys derivated from a master key
- Previous security analysis
  - Indistinguishability
  - Related-key attack
  - Indifferentiability
- Examples:
  - AES, SERPENT, PRESENT ...

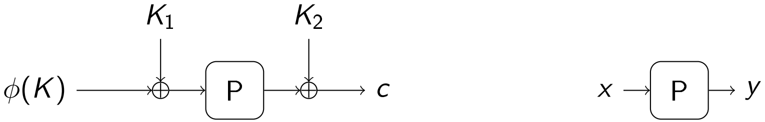
Securiy analysis: Random permutation model



P_i uniformly random permutations,
KDM functions are oracle-independent ($\phi^{P_i} \notin$ KDM set Φ)

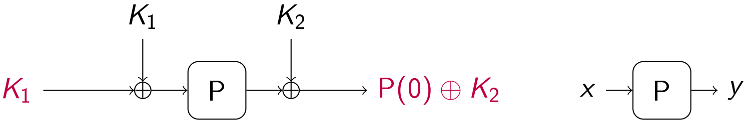
KDM attack: 1-round Even-Mansour

A claw-free set Φ not always enough...



KDM attack: 1-round Even-Mansour

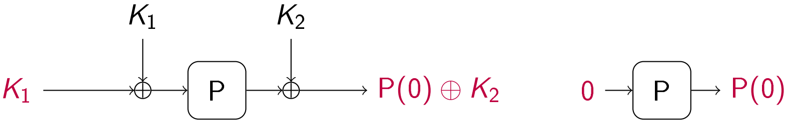
A claw-free set Φ not always enough...



- Step 1. Challenge query $\phi(K_1||K_2) = K_1 \rightarrow c = \mathsf{P}(0) \oplus K_2$

KDM attack: 1-round Even-Mansour

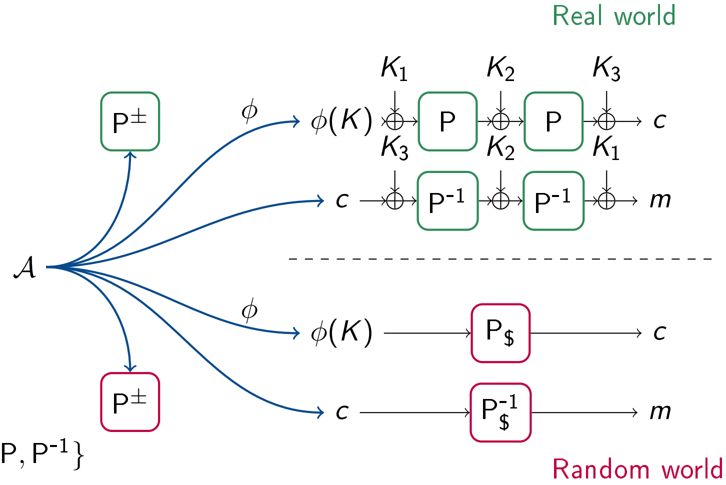
A claw-free set Φ not always enough...



- Step 1. Challenge query $\phi(K_1||K_2) = K_1 \to c = \mathsf{P}(0) \oplus K_2$
- Step 2. Direct query to P $x = 0 \to y = \mathsf{P}(0)$
- Step 3. $\mathcal{A}$ computes $K_2 = c \oplus y$

Key extraction attack by a KDM adversary.

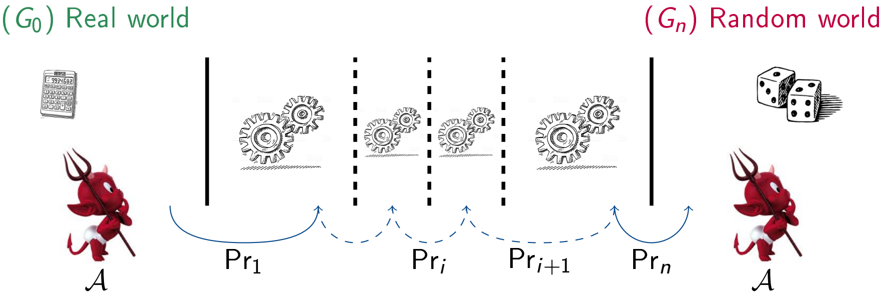
KDM security analysis: 2-round Even-Mansour



$P^{\pm} = \{P, P^{-1}\}$

Restrictions on KDM set Φ to have KDM security?

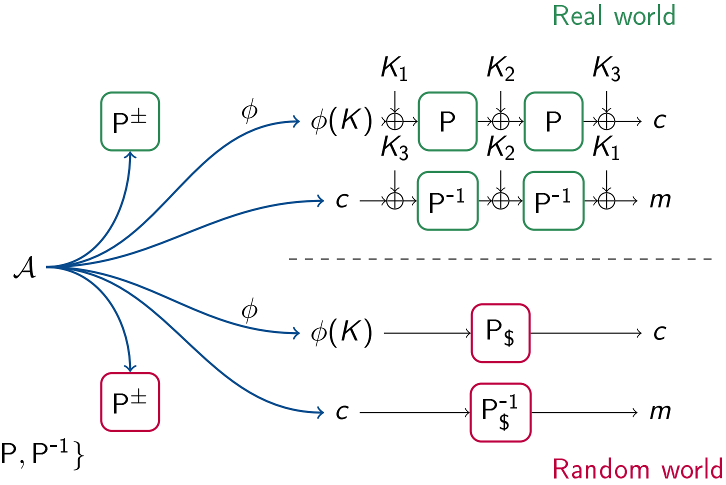
KDM Security Analysis: Game playing



$\Pr_i = \Pr[\mathcal{A}$ sets **bad** in $G_i]$

- Adversary goal:
  - Trig bad events: distinguish real world from random world
- Fundamental lemma of game playing: Adv $\leq \sum \Pr_i$

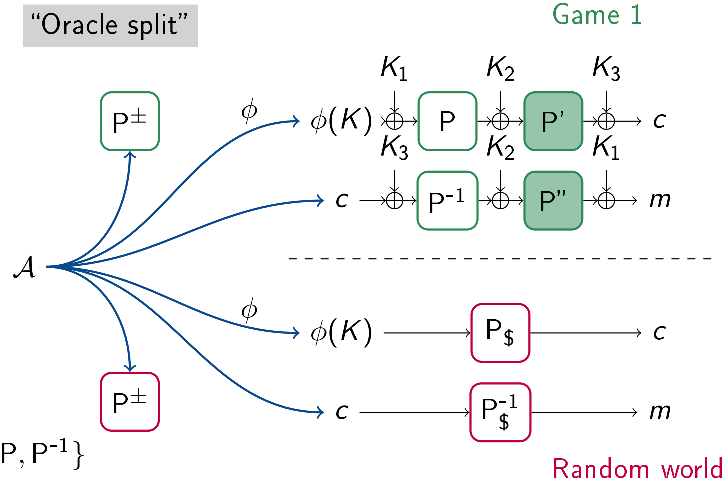
KDM security analysis: Splitting and forgetting technique



$P^{\pm} = \{P, P^{-1}\}$

Application to 2r-EM same permutations, independent keys.

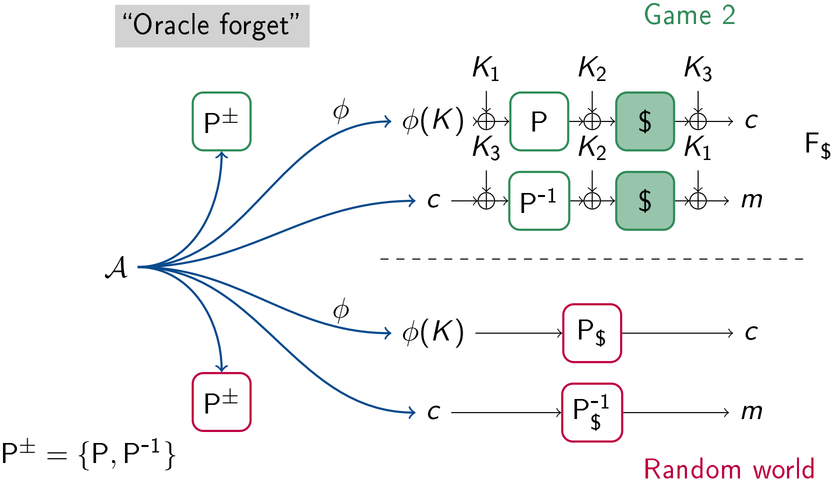
KDM security analysis: Splitting and forgetting technique



$\mathsf{P}^{\pm} = \{\mathsf{P}, \mathsf{P}^{-1}\}$

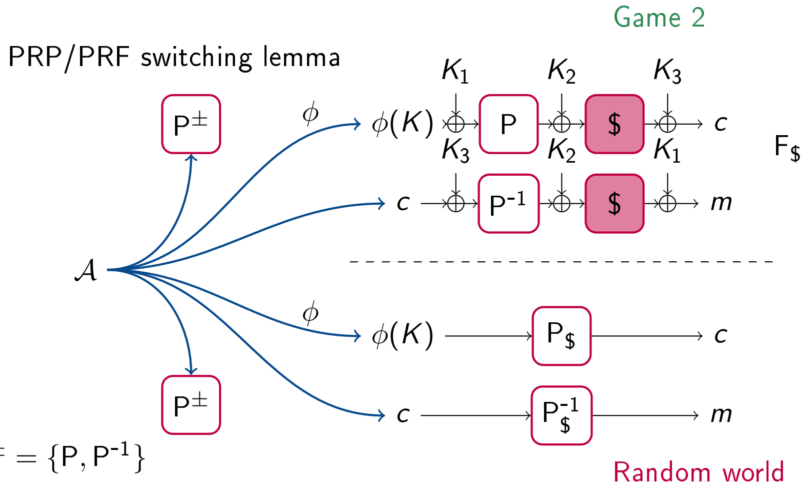
Game 1: Replace last P, P^{-1} with independent random permutations

KDM security analysis: Splitting and forgetting technique



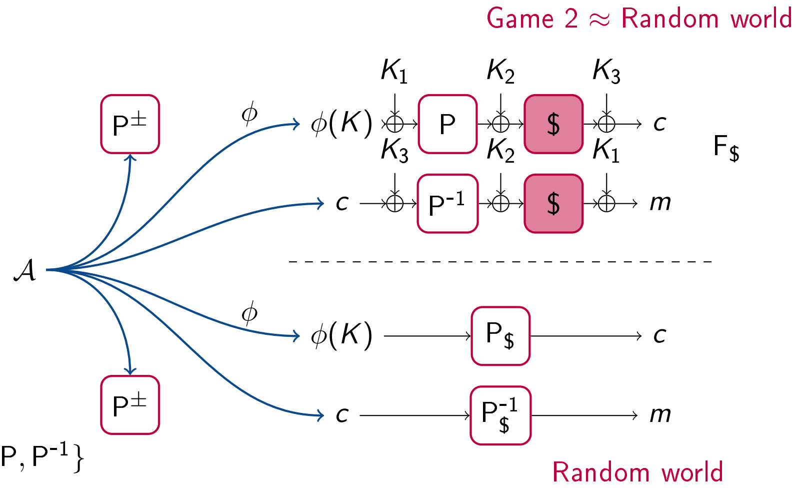
Game 2: Replace last P', P" with forgetful random oracles $

KDM security analysis: Splitting and forgetting technique



$P^{\pm} = \{P, P^{-1}\}$

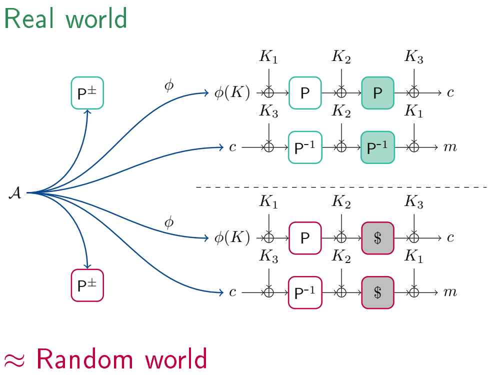
KDM security analysis: Splitting and forgetting technique

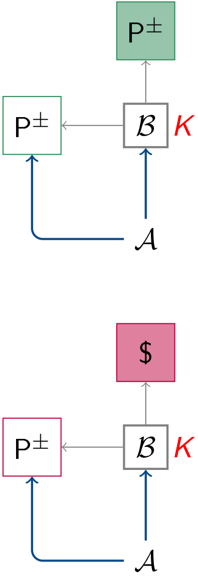


$P^{\pm} = \{P, P^{-1}\}$

Analysis of real world/≈ random world?

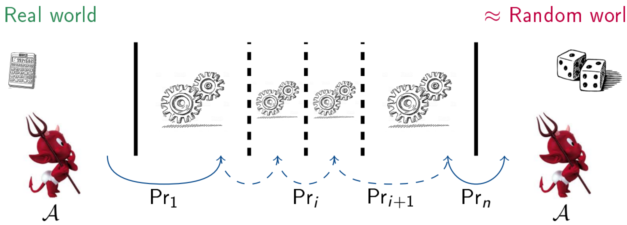
Splitting and forgetting technique



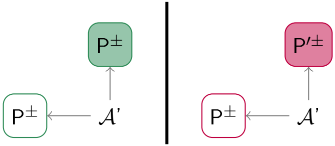


Simulator $\mathcal{B}$ for challenge queries

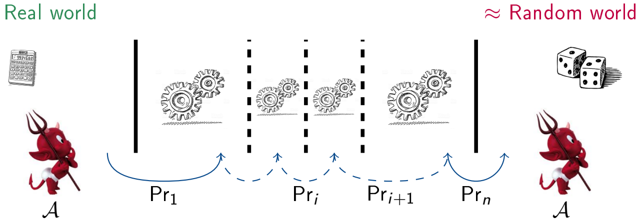
KDM security analysis: Splitting and forgetting technique



- Bad events between real world and ≈ random world:
  - Reduction to adv "splitting game"

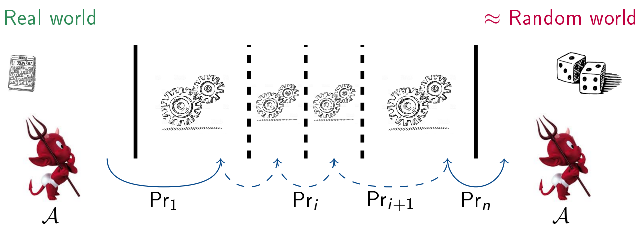


KDM security analysis: Splitting and forgetting technique

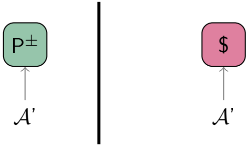


- Bad events between real world and ≈ random world:
  - Reduction to adv "splitting game"
  - Pr[sp] (splitting events type 1 and 2)

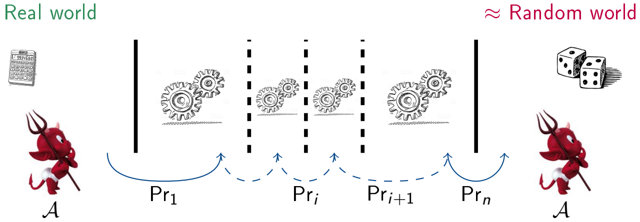
KDM security analysis: Splitting and forgetting technique



- Bad events between real world and ≈ random world:
  - Reduction to adv "splitting game"
  - Pr[sp] (splitting events type 1 and 2)
  - Reduction to adv "forgetful switching game"



KDM security analysis: Splitting and forgetting technique

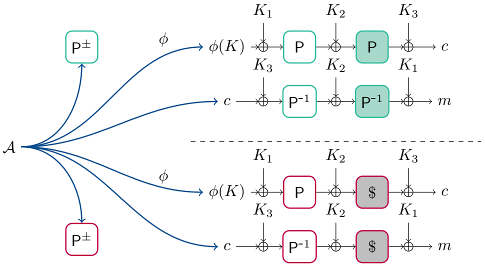


- Bad events between real world and ≈ random world:
  - Reduction to adv "splitting game"
  - Pr[sp] (splitting events type 1 and 2)
  - Reduction to adv "forgetful switching game"
  - Pr[fg] (forgetful events)

Splitting and forgetting technique

Real world

≈ Random world



$$\mathsf{Adv}(\mathcal{A}) \leq 18q^2/2^n + q^2(2 \cdot \mathbf{Adv}^{cf}(\mathcal{A}_1) + \mathbf{Adv}^{ox}(\mathcal{A}_2))$$

when $q_p = q$

offset-xor function: $\phi(K) = K_i \oplus K_j \oplus \Delta$

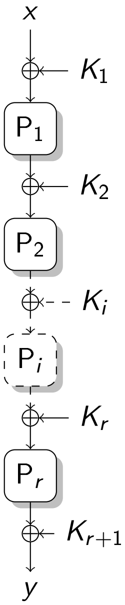
Results: Even-Mansour

Rounds	Permutations	Key schedule	KDM set
1	P	$K_i =$	$\mathrm{cf} \wedge \mathrm{offset}$
2	$\mathrm{P}_i \neq$	$K_i =$	cf
2	$\mathrm{P}_i =$	$K_i \neq$	$\mathrm{cf} \wedge \mathrm{ox}$
2	$\mathrm{P}_i =$	$K_i =$	$\mathrm{cf} \wedge \mathrm{offset}?$
3	$\mathrm{P}_i =$	$K_i =$	$\mathrm{cf} \wedge \mathrm{offset}?$
3	$\mathrm{P}_i =$	$K_i \neq$	cf

Security proofs [Farshim, K., Vergnaud]

On going work

Previous example



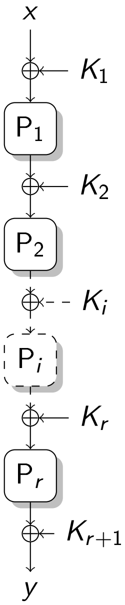
Results: Even-Mansour

Rounds	Permutations	Key schedule	KDM set
1	P	$K_i =$	cf $\wedge$ offset
2	$P_i \neq$	$K_i =$	cf
2	$P_i =$	$K_i \neq$	cf $\wedge$ ox
2	$P_i =$	$K_i =$	cf $\wedge$ offset?
3	$P_i =$	$K_i =$	cf $\wedge$ offset?
3	$P_i =$	$K_i \neq$	cf

Security proofs [Farshim, K., Vergnaud]

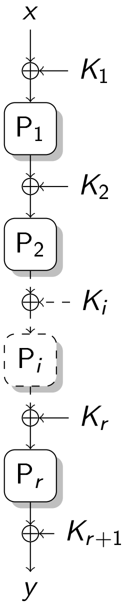
On going work

IC KDM security level



Results: Even-Mansour

Rounds	Permutations	Key schedule	KDM set
1	P	$K_i =$	cf $\wedge$ offset
2	$\mathsf{P}_i \neq$	$K_i =$	cf
2	$\mathsf{P}_i =$	$K_i \neq$	cf $\wedge$ ox
2	$\mathsf{P}_i =$	$K_i =$	cf $\wedge$ offset?
3	$\mathsf{P}_i =$	$K_i =$	cf $\wedge$ offset?
3	$\mathsf{P}_i =$	$K_i \neq$	cf



Security proofs [Farshim, K., Vergnaud]

On going work

Sliding attacks: $\mathsf{P} =$ and $K =$

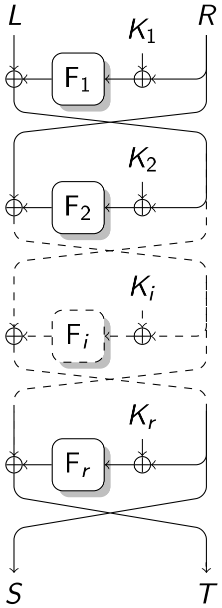
Results: Key-Alternating Feistel

Rounds	Functions	Keys schedule	KDM set
4	F $=$	$K_1, 0, 0, K_2$	$\mathrm{cf} \wedge \mathrm{offset} \wedge \mathrm{ox}$
4	$\mathsf{F}_i \neq$	$K_i \neq$	$\mathrm{cf} \wedge \mathrm{offset}?$
5	$\mathsf{F}_i =$	$K_i \neq$	$\mathrm{cf} \wedge \mathrm{offset}?$
?	$\mathsf{F}_i =$	$K_i \neq$	cf

Security proof based on H-coefficient technique [Farshim, K., Seurin and Vergnaud]

Conjectures.

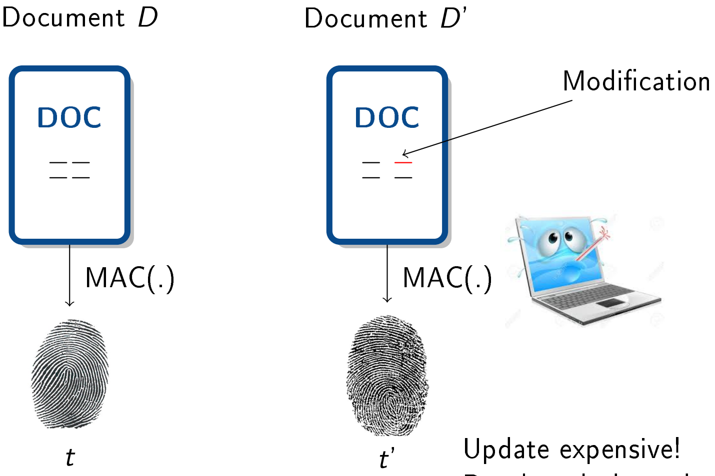
Open question: How many rounds with the same function needed to have KDM-security for a cf-set?



Part 3.

Incremental MACs

Classical MAC algorithm



Update expensive!
Doc length dependent

Incremental Cryptography: MAC

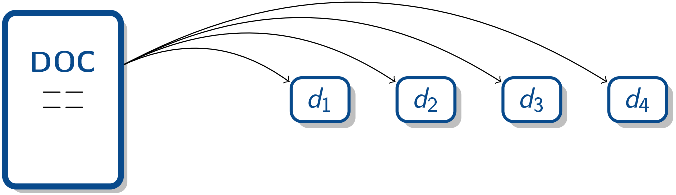
- Generate a tag t of a document D,
- For each edition, the tag t is *updated*
  - Update in time dependent of modification size
  - Update time $<$ MAC time

Incremental MACs: the idea

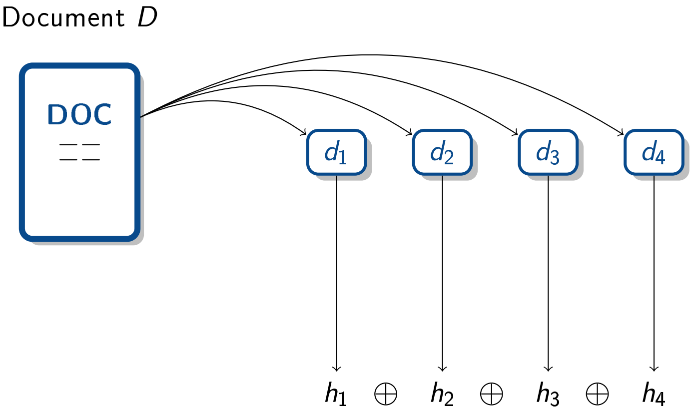
Document D

Incremental MACs: the idea

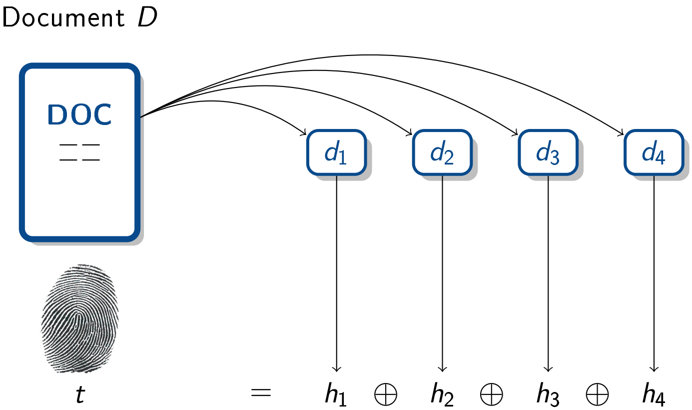
Document D



Incremental MACs: the idea

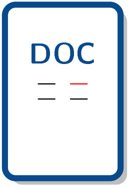


Incremental MACs: the idea



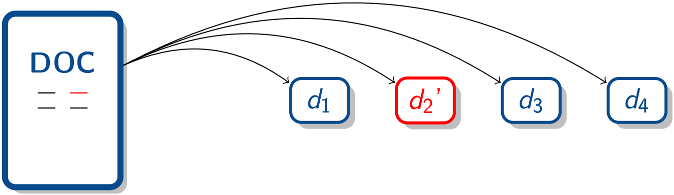
Incremental Cryptography: MAC

Document *D*'

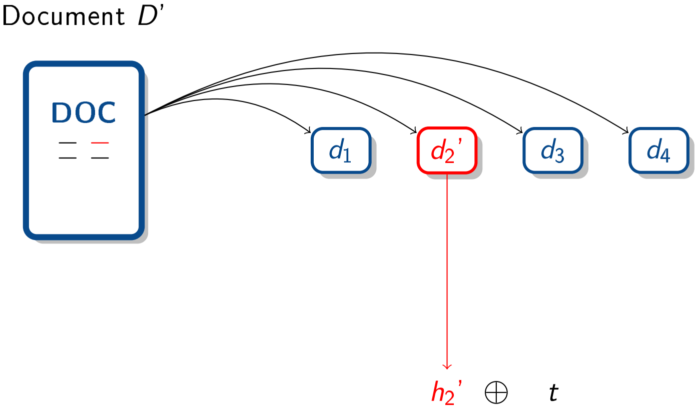


Incremental Cryptography: MAC

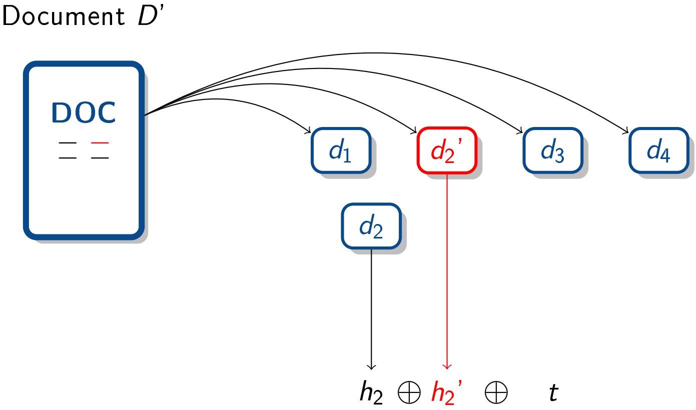
Document *D*'



Incremental Cryptography: MAC

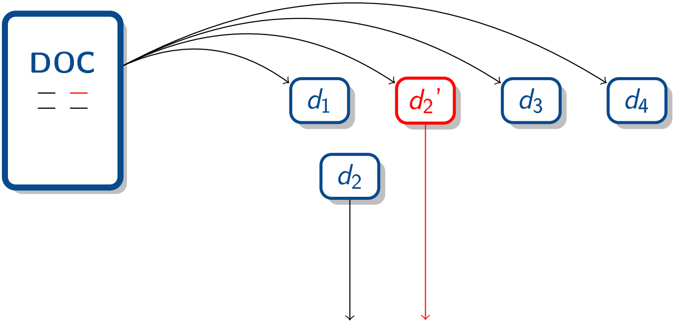


Incremental Cryptography: MAC



Incremental Cryptography: MAC

Document D'



$$t' = h_2 \oplus h_2' \oplus t$$

Tag independent from block order!

Incremental MAC

An algorithm is incremental regarding specific *update* operations.

- Insert
- Delete
- Replace (possible using the previous operations)

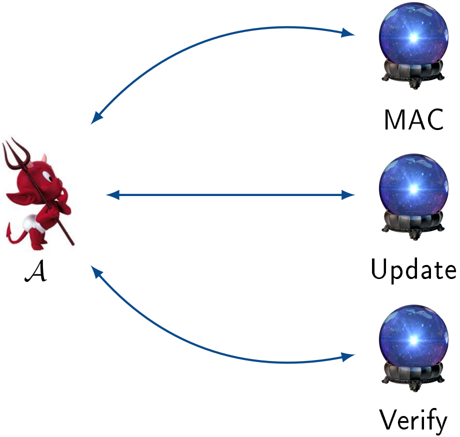
An update operation must be cheaper than recomputing a tag from scratch.

[BGG] Incremental Cryptography and Application to virus protection, Bellare, Goldreich, Goldwasser (1995):

- Security notions: basic security and tamper-proof security
- Chained Xor-Scheme (basic secure)

Security Notion 1: Basic Security Model [BGG]

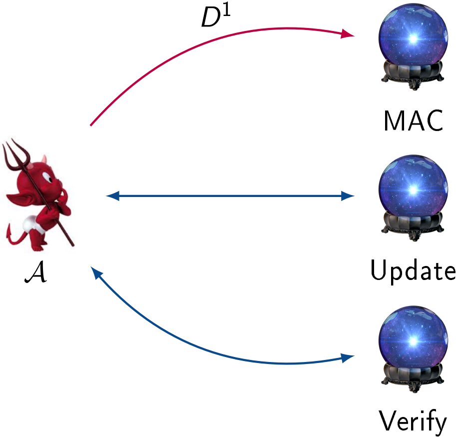
$\mathcal{L} := \{\}$



Notations: D^i (i-th document) and D_j (j-th document block)

Security Notion 1: Basic Security Model [BGG]

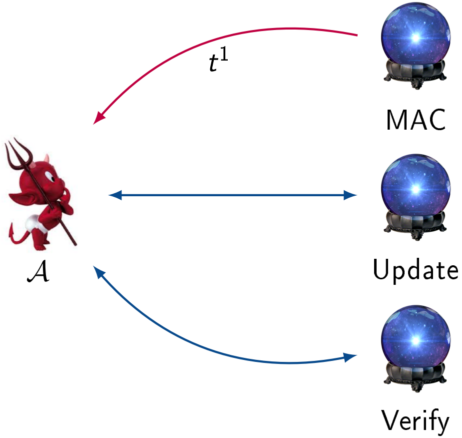
$\mathcal{L} := \{\}$



Notations: D^i (i-th document) and D_j (j-th document block)

Security Notion 1: Basic Security Model [BGG]

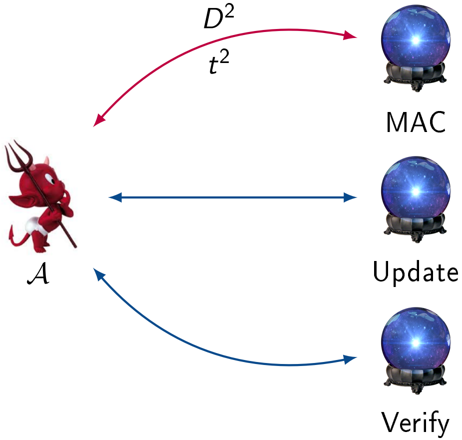
$\mathcal{L} := \{D^1\}$



Notations: D^i (i-th document) and D_j (j-th document block)

Security Notion 1: Basic Security Model [BGG]

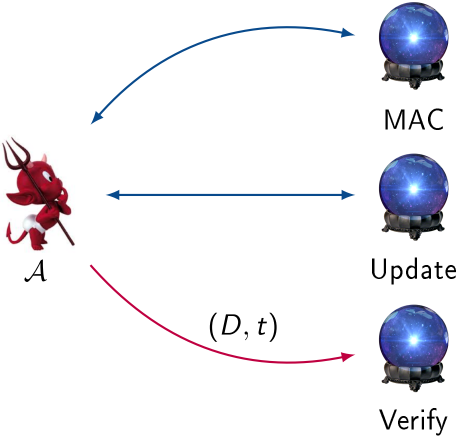
$\mathcal{L} := \{D^1, D^2\}$



Notations: D^i (i-th document) and D_j (j-th document block)

Security Notion 1: Basic Security Model [BGG]

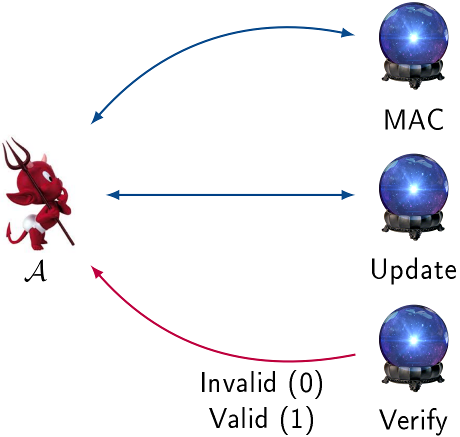
$\mathcal{L} := \{D^1, D^2\}$



Notations: D^i (i-th document) and D_j (j-th document block)

Security Notion 1: Basic Security Model [BGG]

$\mathcal{L} := \{D^1, D^2\}$

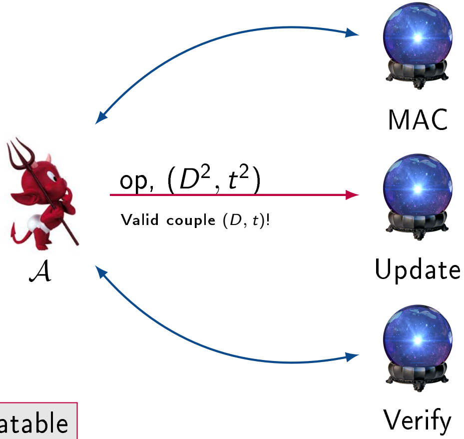


Notations: D^i (i-th document) and D_j (j-th document block)

Security Notion 1: Basic Security Model [BGG]

$\mathcal{L} := \{D^1, D^2\}$

op = Delete first block



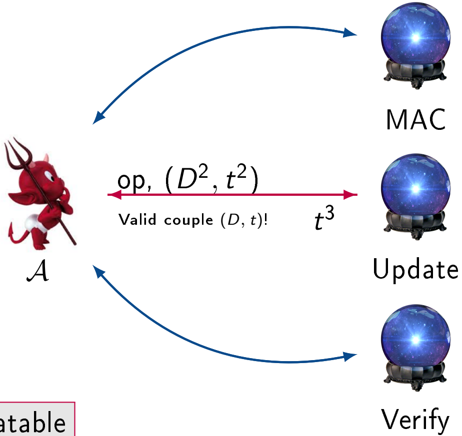
Only valid couples are updatable

Notations: D^i (i-th document) and D_j (j-th document block)

Security Notion 1: Basic Security Model [BGG]

$\mathcal{L} := \{D^1, D^2, D^3\}$

op $=$ Delete first block
$D^3 = \text{op}(D^2)$

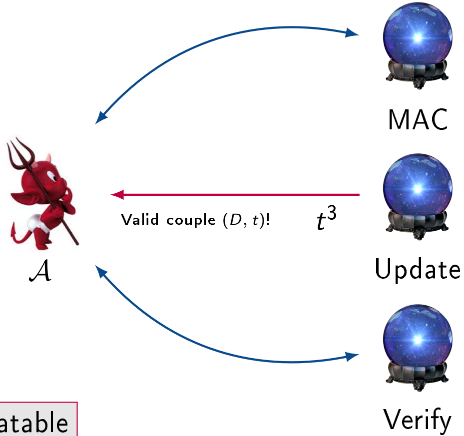


Only valid couples are updatable

Notations: D^i (i-th document) and D_j (j-th document block)

Security Notion 1: Basic Security Model [BGG]

$\mathcal{L} := \{D^1, D^2, D^3, \ldots, D^q\}$



Winning conditions:

$\mathcal{A} \rightarrow (D^*, t^*)$ such that:

- Verify(D^*, t^*) returns 1,
- $D^* \notin \mathcal{L}$.

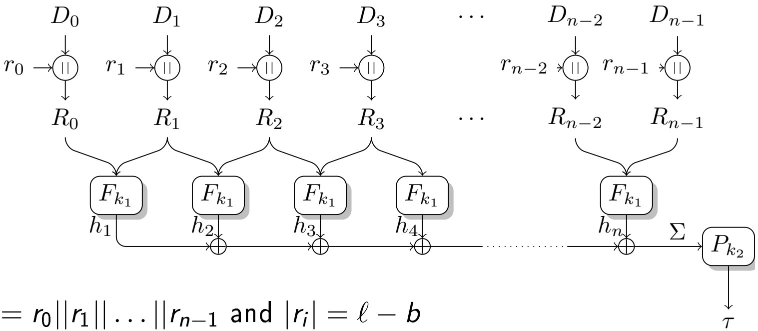
Only valid couples are updatable

Notations: D^i (i-th document) and D_j (j-th document block)

Chained Xor-Scheme [BGG]

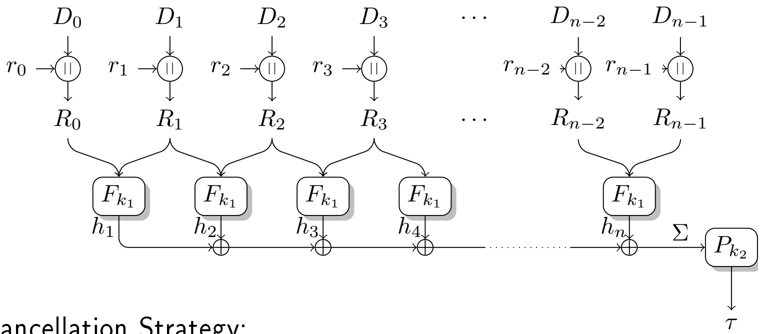
- Pair block chaining algorithm
  - $\mathsf{F} : \mathcal{K}_\mathsf{F} \times \{0,1\}^{2\ell} \rightarrow \{0,1\}^L$
  - $\mathsf{P} : \mathcal{K}_\mathsf{P} \times \{0,1\}^{L} \rightarrow \{0,1\}^L$

- In: Document D (n blocks D_i)
- Out: Tag t such that $t = (r, \tau)$



$r = r_0||r_1||\ldots||r_{n-1}$ and $|r_i| = \ell - b$

Simple forgery strategy



Cancellation Strategy:

- $\mathcal{A}$ asks a MAC on any document D and receives $t = (r, \tau)$
- Goal: Play with D to build D^* such that $\Sigma = \Sigma^*$

[K. and Vergnaud]

Example: 3-block document D

$D = D_0||D_1||D_2$

$t = (r, \tau)$ such that $r := r_0||r_1||r_2$ $\qquad (R_i = D_i||r_i)$

$$
\begin{array}{ccccc}
(R_0, R_1) & & (R_1, R_2) & & \\
\downarrow & & \downarrow & & \\
h_1 & \oplus & h_2 & = & \Sigma
\end{array}
$$

Example: 3-block document D

$D = D_0||D_1||D_2$

$t = (r, \tau)$ such that $r := r_0||r_1||r_2$ $\qquad (R_i = D_i||r_i)$

$$\begin{array}{ccccc} (R_0, R_1) & & (R_1, R_2) & & \\ \downarrow & & \downarrow & & \\ h_1 & \oplus & h_2 & = & \Sigma \end{array}$$

Build D^* and r^* such that :

$$\begin{array}{ccccccccc} (R_0, R_1) & & (.,.) & & (.,.) & & (R_1, R_2) & & \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \\ h_1 & \oplus & \underbrace{\ldots & \oplus & \ldots}_{=0} & \oplus & h_2 & = & \Sigma \end{array}$$

Attack Example: 3-block document D

$D = D_0||D_1||D_2$ and $R_i = D_i||r_i$
$t = (r, \tau)$ such that $r := r_0||r_1||r_2$

Build D^* and r^* such that:

$$\begin{array}{cccccccc} (R_0, R_1) & (R_1, R_2) & (R_2, R_1) & (R_1, R_2) & (R_2, R_1) & (R_1, R_2) & & \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & \\ h_1 & \cancel{h_2} & \cancel{h_2}' & \cancel{h_2} & \cancel{h_2}' & h_2 & = & \Sigma^* \end{array}$$

$$\underbrace{\cancel{h_2} \quad \cancel{h_2}' \quad \cancel{h_2} \quad \cancel{h_2}'}_{=0}$$

$D^* = D_0||D_1||D_2||D_1||D_2||D_1||D_2$ $\tau^* = \tau$ and $t^* = (r^*, \tau^*)$
$r^* = r_0||r_1||r_2||r_1||r_2||r_1||r_2$ $(D^*, t^*) \neq (D, t)$

Attack Example: 3-block document D

$D = D_0||D_1||D_2$ and $R_i = D_i||r_i$
$t = (r, \tau)$ such that $r := r_0||r_1||r_2$

Build D^* and r^* such that:

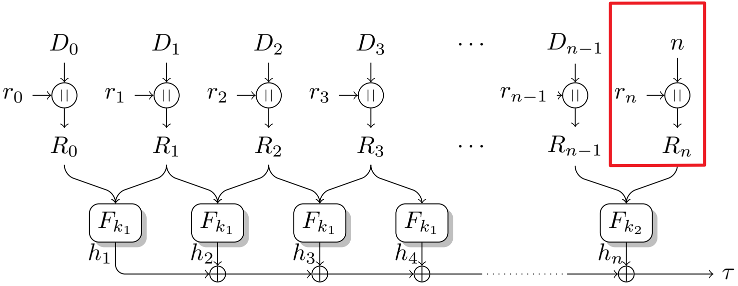
$$\begin{array}{cccccccc}
(R_0, R_1) & (R_1, R_2) & (R_2, R_1) & (R_1, R_2) & (R_2, R_1) & (R_1, R_2) & & \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & \\
h_1 & \cancel{h_2} & \cancel{h_2}' & \cancel{h_2} & \cancel{h_2}' & h_2 & = & \Sigma^*
\end{array}$$

$$\underbrace{\cancel{h_2} \quad \cancel{h_2}' \quad \cancel{h_2} \quad \cancel{h_2}'}_{= 0}$$

$D^* = D_0||D_1||D_2||D_1||D_2||D_1||D_2$ $\quad$ $\tau^* = \tau$ and $t^* = (r^*, \tau^*)$
$r^* = r_0||r_1||r_2||r_1||r_2||r_1||r_2$ $\quad$ $(D^*, t^*) \neq (D, t)$

More attacks in the thesis.

Modified Xor-Scheme 2

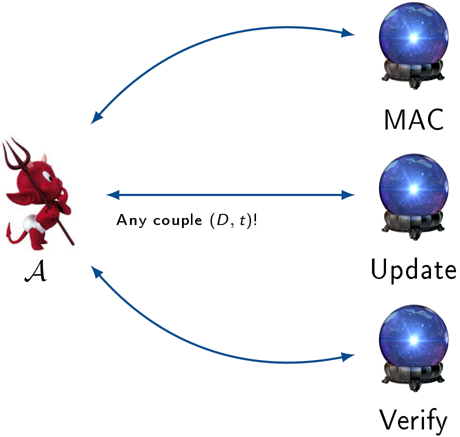


- A **fresh** value r_n for each update operation
- The random value r_n is necessary!

Basic secure scheme

Security Notion 2: Tamper-proof Security Model [BGG]

$\mathcal{L} := \{\}$



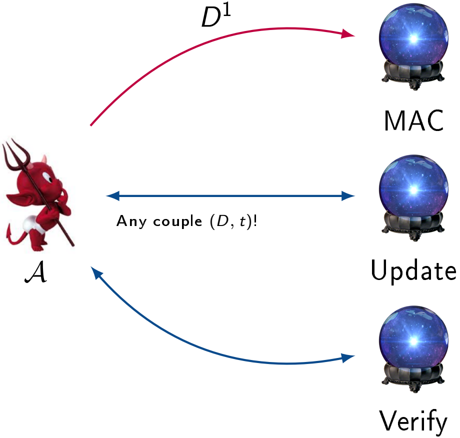
Winning conditions:

$\mathcal{A} \rightarrow (D^*, t^*)$ such that:

- Verify(D^*, t^*) returns 1
- $D^* \notin \mathcal{L}$

Security Notion 2: Tamper-proof Security Model [BGG]

$\mathcal{L} := \{\}$



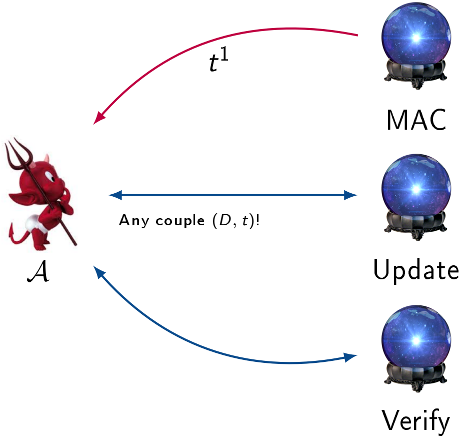
Winning conditions:

$\mathcal{A} \rightarrow (D^*, t^*)$ such that:

- Verify(D^*, t^*) returns 1
- $D^* \notin \mathcal{L}$

Security Notion 2: Tamper-proof Security Model [BGG]

$\mathcal{L} := \{D^1\}$



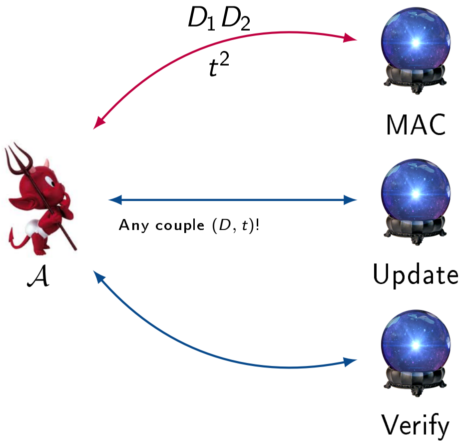
Winning conditions:

$\mathcal{A} \rightarrow (D^*, t^*)$ such that:

- Verify(D^*, t^*) returns 1
- $D^* \notin \mathcal{L}$

Security Notion 2: Tamper-proof Security Model [BGG]

$\mathcal{L} := \{D^1,\ D_1 D_2\}$



Winning conditions:

$\mathcal{A} \rightarrow (D^*, t^*)$ such that:

- Verify(D^*, t^*) returns 1
- $D^* \notin \mathcal{L}$

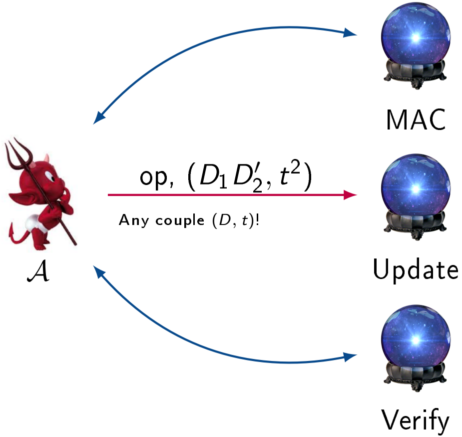
Security Notion 2: Tamper-proof Security Model [BGG]

$\mathcal{L} := \{D^1,\ D_1 D_2\}$

op = Replace block 1 by D_1'
$D^3 = \text{op}(?)$

<u>Winning conditions:</u>
$\mathcal{A} \rightarrow (D^*, t^*)$ such that:

- Verify(D^*, t^*) returns 1
- $D^* \notin \mathcal{L}$



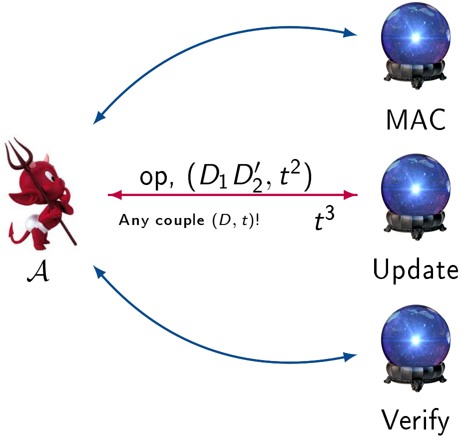
Security Notion 2: Tamper-proof Security Model [BGG]

$\mathcal{L} := \{D^1,\ D_1 D_2,\ D_1' D_2'?\}$

op = Replace block 1 by D_1'
$D^3 = \text{op}(?)$

Winning conditions:
$\mathcal{A} \to (D^*, t^*)$ such that:

- Verify(D^*, t^*) returns 1
- $D^* \notin \mathcal{L}$



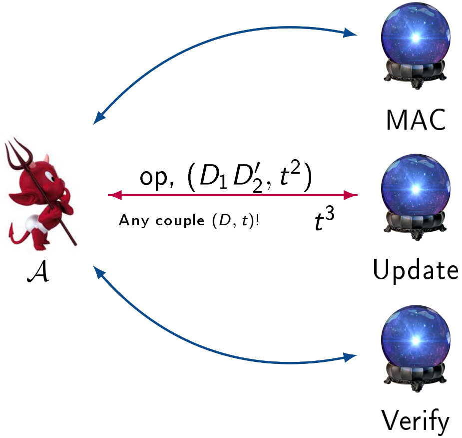
Security Notion 2: Tamper-proof Security Model [BGG]

$\mathcal{L} := \{D^1,\ D_1 D_2,\ D_1' D_2'?\}$
or $D_1' D_2$? or both?$\}$

op = Replace block 1 by D_1'
$D^3 = \text{op}(?)$

Winning conditions:
$\mathcal{A} \rightarrow (D^*, t^*)$ such that:

- Verify(D^*, t^*) returns 1
- $D^* \notin \mathcal{L}$



Security Notion 2: Tamper-proof Security Model [BGG]

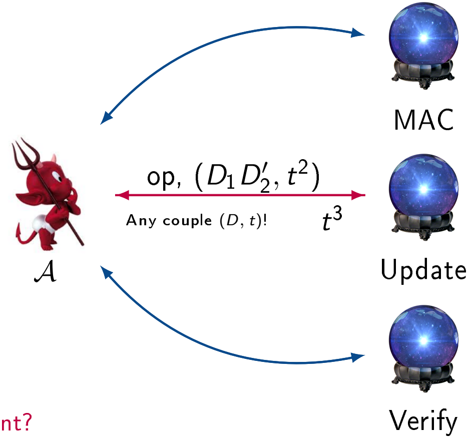
$\mathcal{L} := \{D^1,\ D_1D_2,\ D_1'D_2'?\}$
or $D_1'D_2$? or both?$\}$

op = Replace block 1 by D_1'
$D^3 = \text{op}(?)$

Winning conditions:
$\mathcal{A} \rightarrow (D^*, t^*)$ such that:

- Verify(D^*, t^*) returns 1
- $D^* \notin \mathcal{L}$

But how to build $\mathcal{L}$?
How can we track each document?
No game definition...

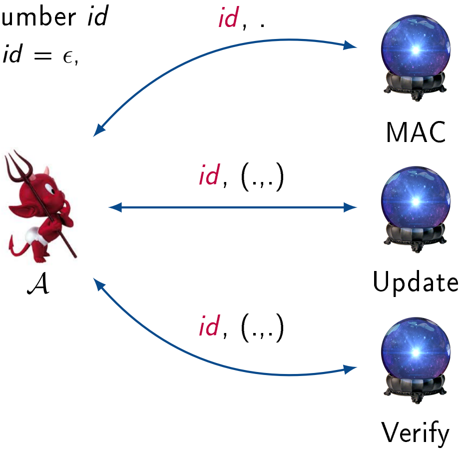


$\Longrightarrow$ Introduction of the document identification number *id*

New Framework for iMAC

- Document identification number *id*
  - Single-document (SD) $id = \epsilon$,
  - Multi-document (MD).

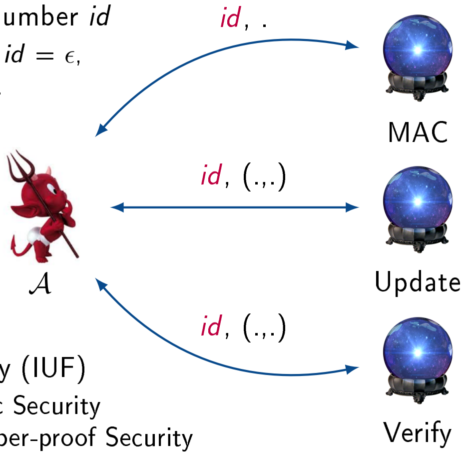
  One MAC call/*id*



New Framework for iMAC

- Document identification number *id*
  - Single-document (SD) $id = \epsilon$,
  - Multi-document (MD).

  One MAC call/*id*



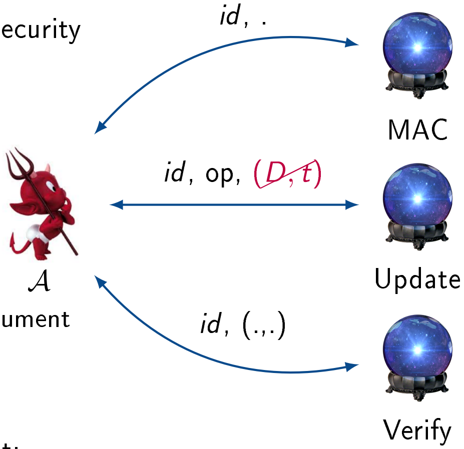
- Incremental UnForgeability (IUF)
  - Type 1 (IUF1) $\approx$ Basic Security
  - Type 2 (IUF2) $\approx$ Tamper-proof Security

Security game IUF1

- Definition close to Basic security
- List $\mathcal{L}$

1	D^1	D^2
2	D^1	D^2
3	D^1	

- For each *id*
  - Last version of the document updated
- Winning conditions:
  $\mathcal{A} \rightarrow (id, D^*, t^*)$ such that:
  - Verify (id, D^*, t^*) returns 1,
  - $(id, D^*) \notin \mathcal{L}$.

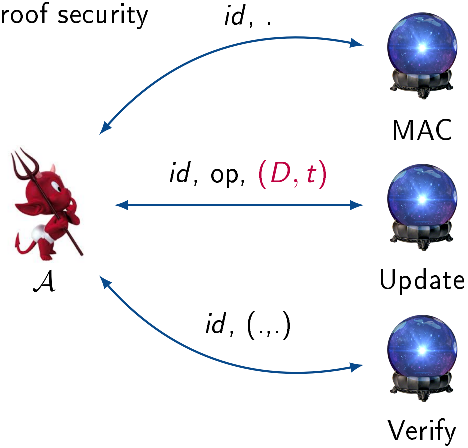


Security game IUF2

- Definition close to Tamper-proof security
- List $\mathcal{L}$

  1 $D^1 \rightarrow D^2$
  2 $D^1 \rightarrow D^2$
  3 D^1

- For each *id*
  - tag: Computed with D
  - List: Filled with $\mathrm{op}(D_{id})$
- Winning conditions:
  $\mathcal{A} \rightarrow (id, D^*, t^*)$ such that:
  - Verify (id, D^*, t^*) returns 1,
  - $(id, D^*) \notin \mathcal{L}$.

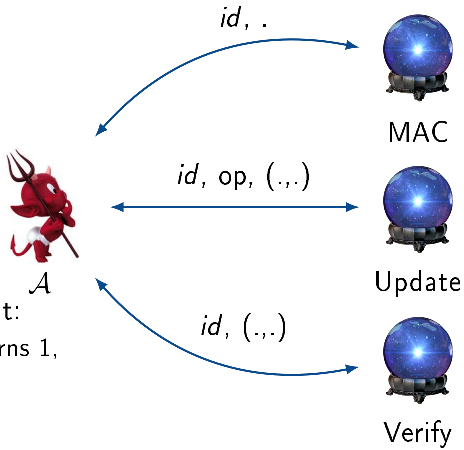


Security game IUFR ("Replay")

- IUF1R or IUF2R
- List $\mathcal{L}$ (last doc/ id)

1	D^7
2	D^4
3	D^1

- Winning conditions:
  $\mathcal{A} \rightarrow (id, D^*, t^*)$ such that:
  - Verify (id, D^*, t^*) returns 1,
  - $(id, D^*) \notin \mathcal{L}$.



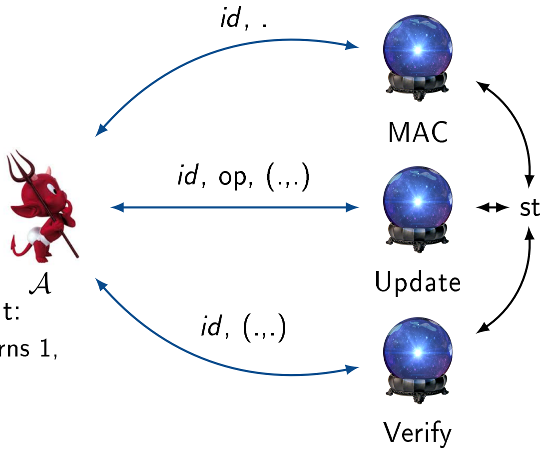
FADE mechanism

Security game IUFR ("Replay")

- IUF1R or IUF2R
- List $\mathcal{L}$ (last doc/ *id*)

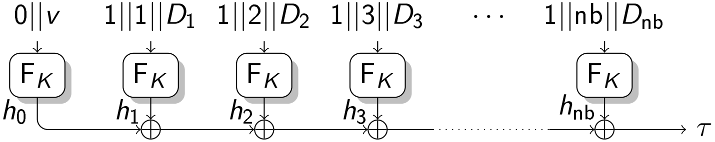
1	D^7
2	D^4
3	D^1

- <u>Winning conditions:</u>
  $\mathcal{A} \rightarrow (id, D^*, t^*)$ such that:
  - Verify (id, D^*, t^*) returns 1,
  - $(id, D^*) \notin \mathcal{L}$.
- Stateful Schemes only
  - (Secure memory)



FADE mechanism

Results: Constructions

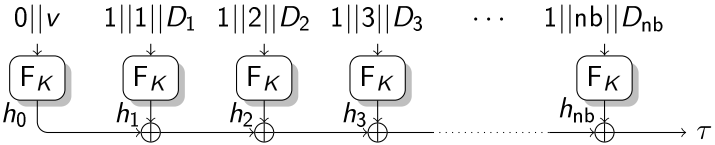


From a basic secure Xor-MAC to a IUF1R-MD construction.

- Xor-MAC is basic secure
  - "Xor-MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions", Bellare, Guérin, Rogaway.
- Basic security $\Longrightarrow$ IUF1-SD
- A construction IUF1R-MD
  - Generic construct.: SD to MD
  - Generic construct.: IUFx to IUFxR

[Arte, Bellare, K. and Vergnaud]

Results: Constructions



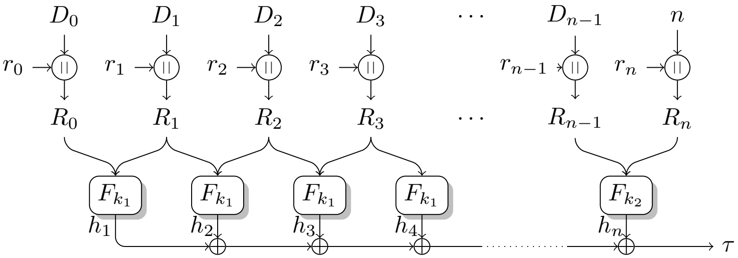
From a basic secure Xor-MAC to a IUF1R-MD construction.

- Xor-MAC is basic secure
  - "Xor-MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions", Bellare, Guérin, Rogaway.
- Basic security $\implies$ IUF1-SD
- A construction IUF1R-MD
  - Generic construct.: SD to MD
  - Generic construct.: IUFx to IUFxR

Not IUF2

[Arte, Bellare, K. and Vergnaud]

Results: Constructions

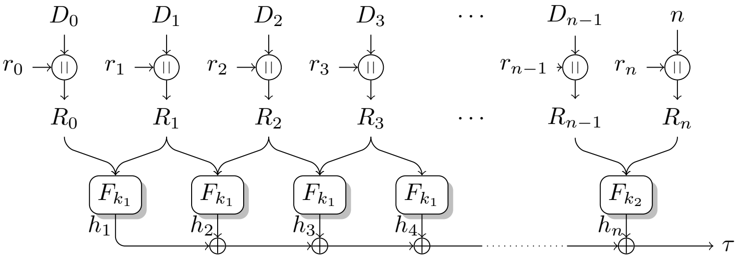


From an IUF2-SD secure Xor-Scheme to IUF2R-MD secure construction.

- Xor-Scheme proved IUF2-SD
- A IUF2R-MD secure construction
  - Generic construct.: SD to MD,
  - Generic construct.: IUFx to IUFxR.

[Arte, Bellare, K. and Vergnaud].

Results: Constructions



From an IUF2-SD secure Xor-Scheme to IUF2R-MD secure construction.

- Xor-Scheme proved IUF2-SD
- A IUF2R-MD secure construction
  - Generic construct.: SD to MD,
  - Generic construct.: IUFx to IUFxR.

Strongest security notion

[Arte, Bellare, K. and Vergnaud].

Conclusion

- KDM security:
  - Forgetting and splitting (application EM)
  - H-coefficient technique (application KAF)

Conclusion

- KDM security:
  - Forgetting and splitting (application EM)
  - H-coefficient technique (application KAF)
  - Minimal KAF configuration KDM secure under a claw-free set
  - Application to other schemes?

Conclusion

- KDM security:
  - Forgetting and splitting (application EM)
  - H-coefficient technique (application KAF)
  - Minimal KAF configuration KDM secure under a claw-free set
  - Application to other schemes?
- Incremental MACs
  - Security notions and Relations among security notions,
  - Generic constructions,
  - An IUF2R-MD secure construction
    - Tag too large,
    - Greedy in randomness.

Conclusion

- KDM security:
  - Forgetting and splitting (application EM)
  - H-coefficient technique (application KAF)
  - Minimal KAF configuration KDM secure under a claw-free set
  - Application to other schemes?
- Incremental MACs
  - Security notions and Relations among security notions,
  - Generic constructions,
  - An IUF2R-MD secure construction
    - Tag too large,
    - Greedy in randomness.
  - More efficient schemes (time/storage)?
    - Can we build such a scheme ?
  - What about implementation?

Thank you for your attention!

Full Disk Encryption and Beyond

Monday, 15 July 2019

ENS ÉCOLE NORMALE SUPÉRIEURE

PSL RESEARCH UNIVERSITY PARIS

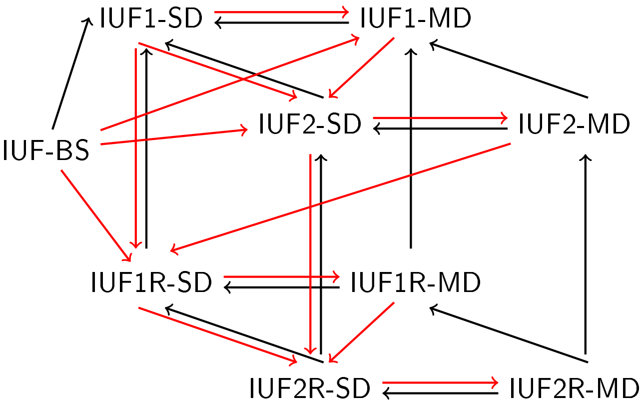
ANSSI

Louiza Khati

Contributions

- FDE: Bridging theory and practice, RSA 2017
  - K., Mouha and Vergnaud.
- Even-Mansour cipher under KDM security, FSE 2018
  - Farshim, K. and Vergnaud
- KDM-Security of Key-Alternating Feistel Ciphers
  - Farshim, K., Seurin and Vergnaud
- Analysis and improvement of an incremental scheme, SAC 2018
  - K. and Vergnaud
- Incremental MACs
  - Arte, Bellare, K. and Vergnaud.

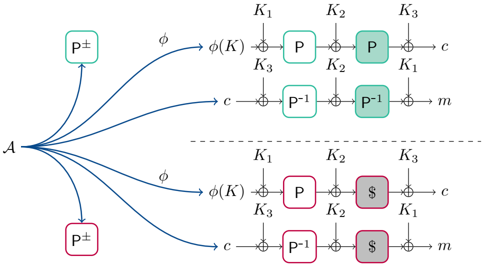
Relations among security notions



Splitting and forgetting technique

Real world

≈ Random world

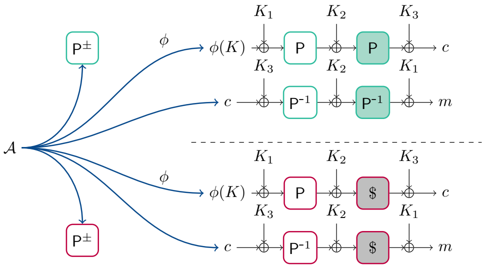


$\mathsf{Adv}(\mathcal{A}) \leq 18q^2/2^n + q^2(2 \cdot \mathbf{Adv}^{cf}(\mathcal{A}_1) + \mathbf{Adv}^{ox}(\mathcal{A}_2))$
when $q_p = q$

offset-xor function: $\phi(K) = K_i \oplus K_j \oplus \Delta$

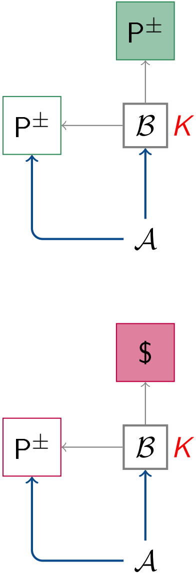
Splitting and forgetting technique

Real world



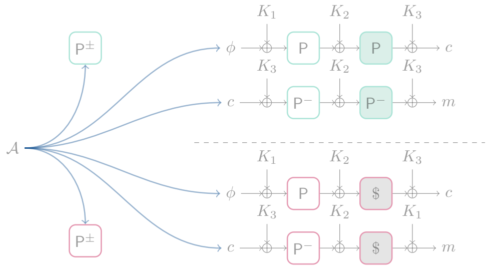
≈ Random world

2-round Even-Mansour

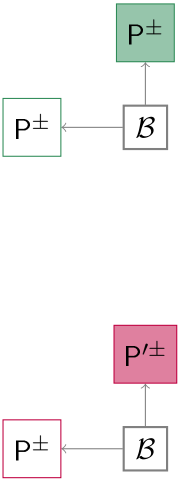


Splitting and forgetting technique

Real world



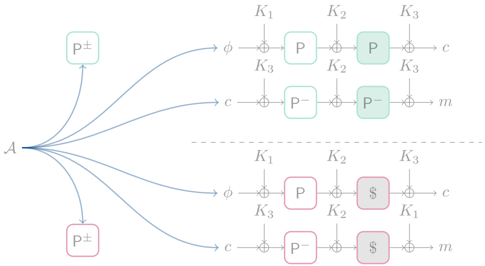
≈ Random world



Splitting game

Splitting and forgetting technique

Real world

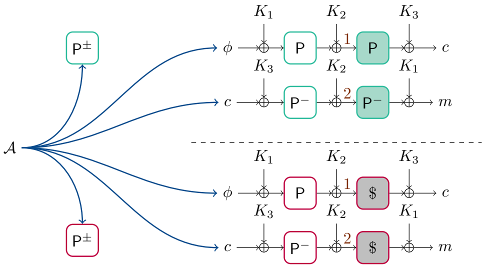


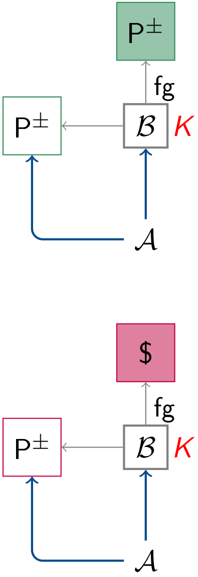
≈ Random world

Forgetful switching game

Splitting and forgetting technique

Real world



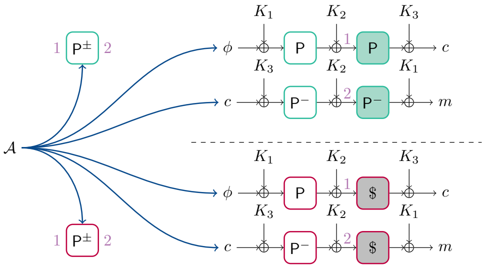


≈ Random world

Forgetful events

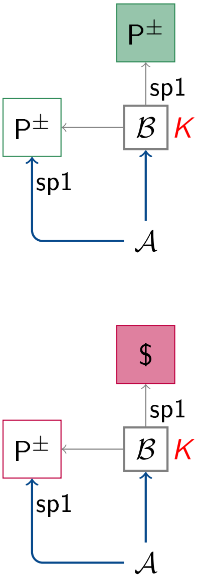
Splitting and forgetting technique

Real world



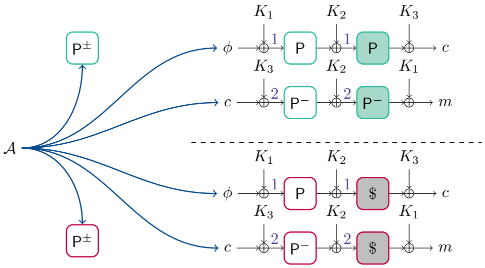
≈ Random world

Splitting events 1

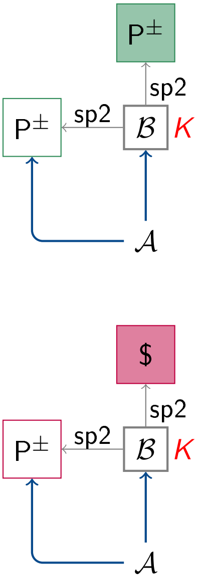


Splitting and forgetting technique

Real world



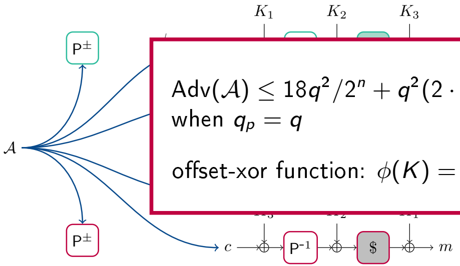
≈ Random world



Splitting events 2

Splitting and forgetting technique

Real world



Adv($\mathcal{A}$) $\leq 18q^2/2^n + q^2(2 \cdot \mathbf{Adv}^{cf}(\mathcal{A}_1) + \mathbf{Adv}^{ox}(\mathcal{A}_2))$
when $q_p = q$

offset-xor function: $\phi(K) = K_i \oplus K_j \oplus \Delta$

$\approx$ Random world