Analysis and Improvement of an Authentication Scheme in Incremental Cryptography

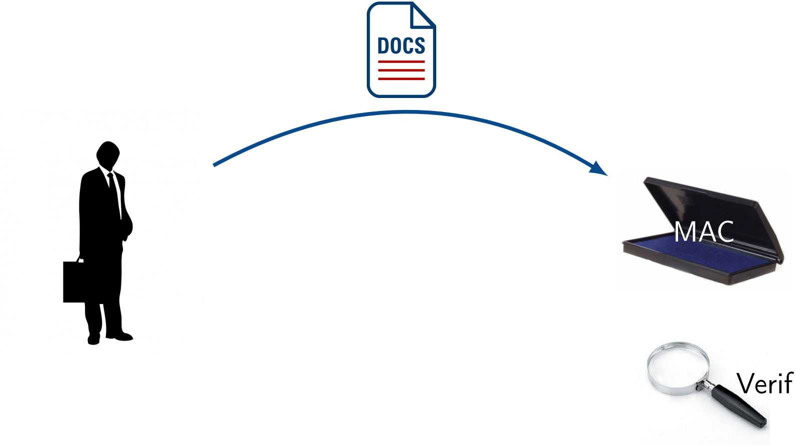
Louiza Khati[1,2], Damien Vergnaud[2,3]

1 ANSSI
2 ENS
3 LIP6

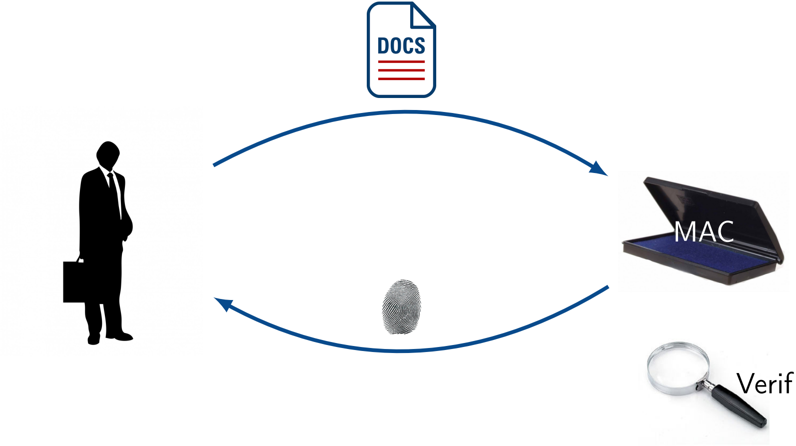
Last update:

Friday, August 17th, 2018

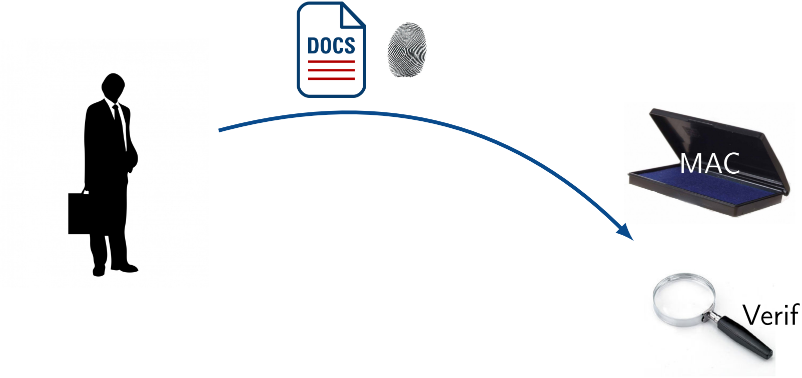
MAC Algorithm



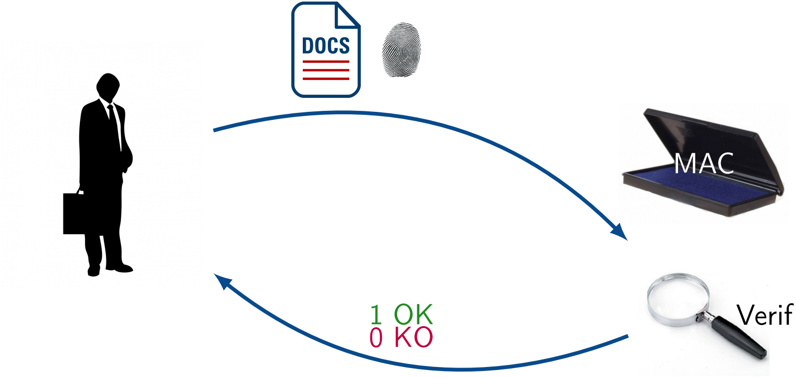
MAC Algorithm



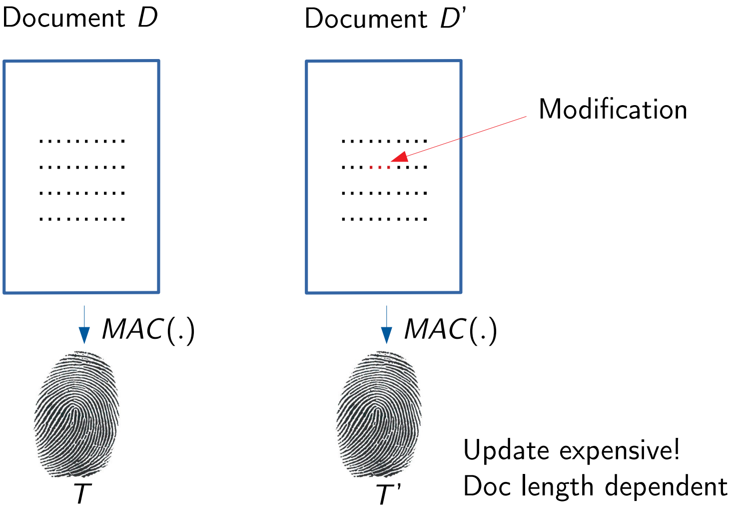
MAC Algorithm



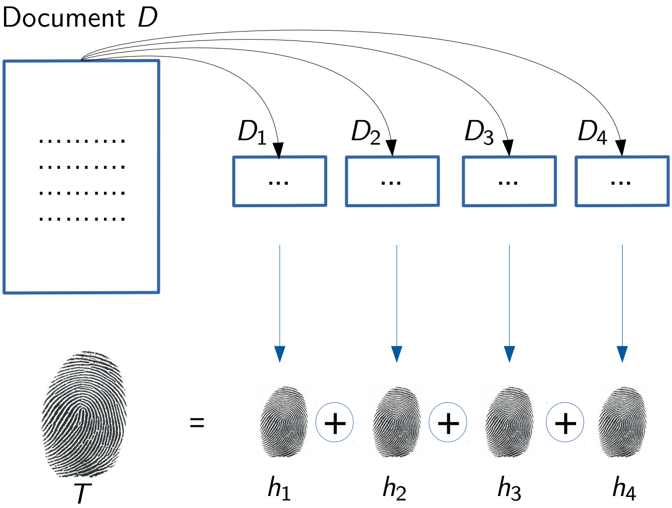
MAC Algorithm



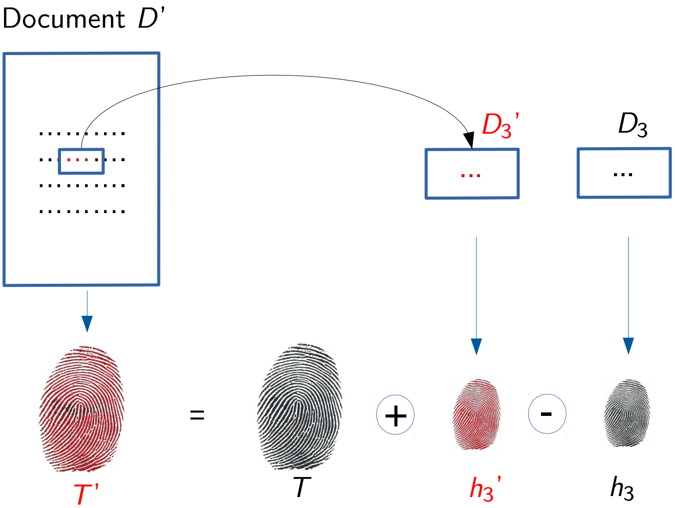
MAC algorithm



Incremental Cryptography: MAC



Incremental Cryptography: MAC



Incremental MAC

An algorithm is incremental regarding specific *update* operations.

- Insert *n* blocks
- Delete *n* blocks
- *Replace* *n* blocks at any position

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

Incremental MAC

An algorithm is incremental regarding specific *update* operations.

- Insert n blocks at any position*
- Delete n blocks at any position*
- *Replace* n blocks at any position

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

*Strongly Incremental

Previous Works

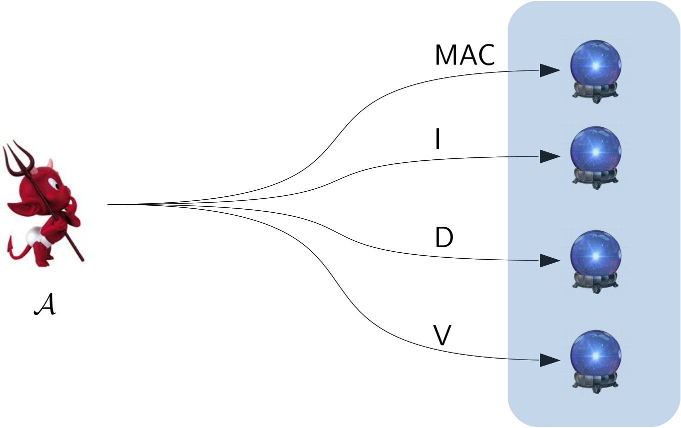
- Seminal paper by Bellare, Goldreich and Goldwasser (1994)
  - Introduction of incremental cryptography,
  - Security notions,
  - Pairwise chaining XOR-Scheme (Strongly Incremental).
- *XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions* (1995).
  - XOR-Scheme (different from the chaining algo).
- *A new mode of operation for incremental authenticated encryption with associated data* by Sasaki and Yasuda (2016)
  - Replace and (Insert, Delete) at the **last** position
- Many other papers on various primitives.

Advantages and Use cases

Incremental cryptography is useful to solve some challenges:

- Computation on Big Data,
- Limited energy consumption (ex. mobile phones),
- Sensors (non stop recording data),
- etc. . .

Adversary Model

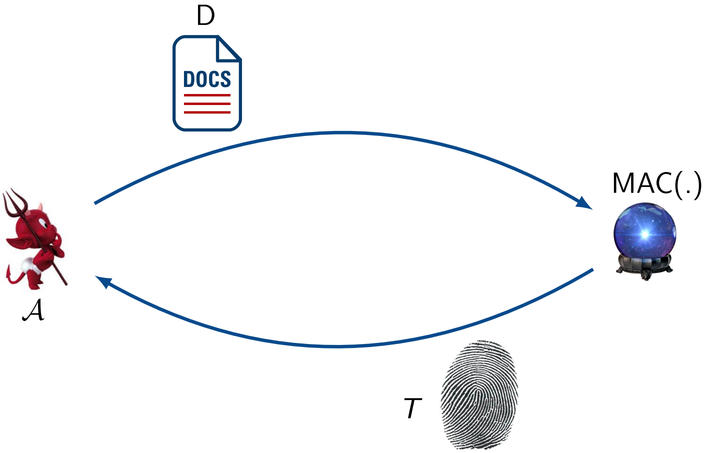


Insert blocks

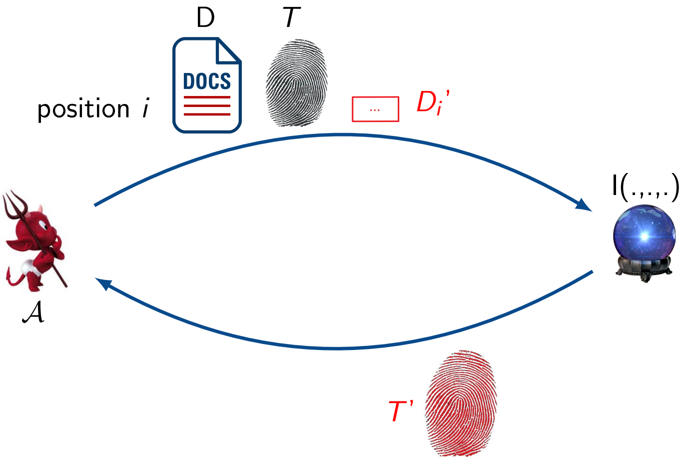
Delete blocks

Verification

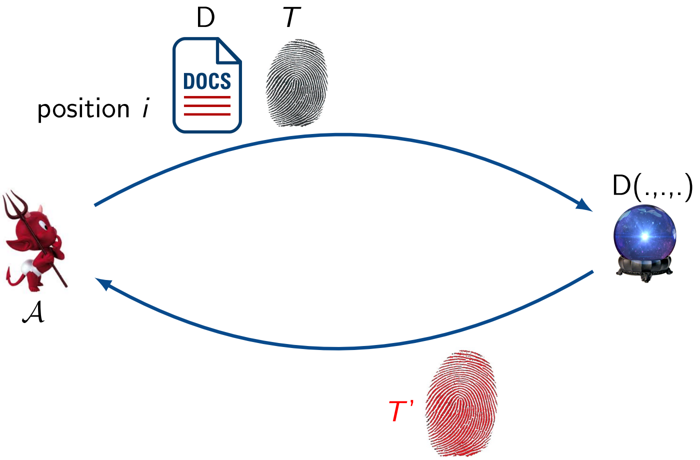
MAC algorithm



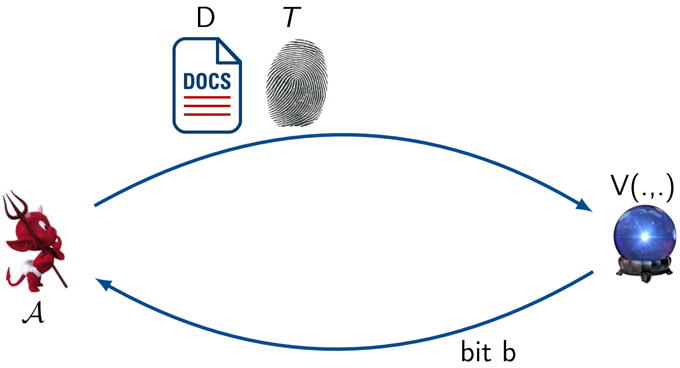
Insert algorithm I(.,.,.)



Delete algorithm D(.,.,.)



Verification algorithm V(., .)

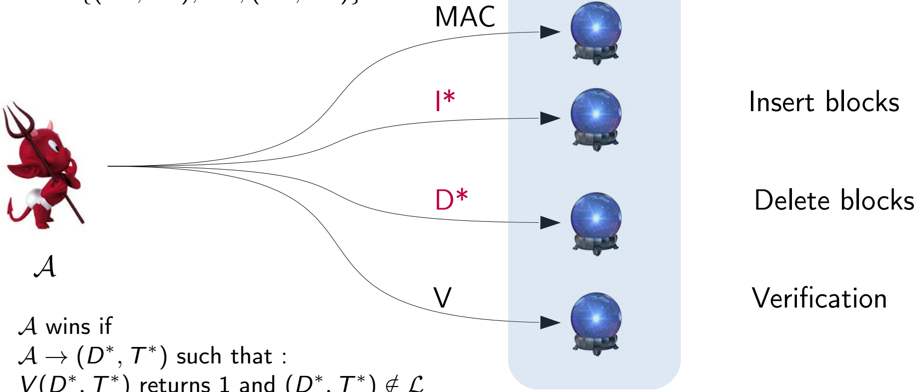


b = 0: verification fails,

b = 1: verification succeeds.

Security Notion 1: Basic Security Model

$\mathcal{L} := \{(D^1, T^1), \ldots, (D^q, T^q)\}$



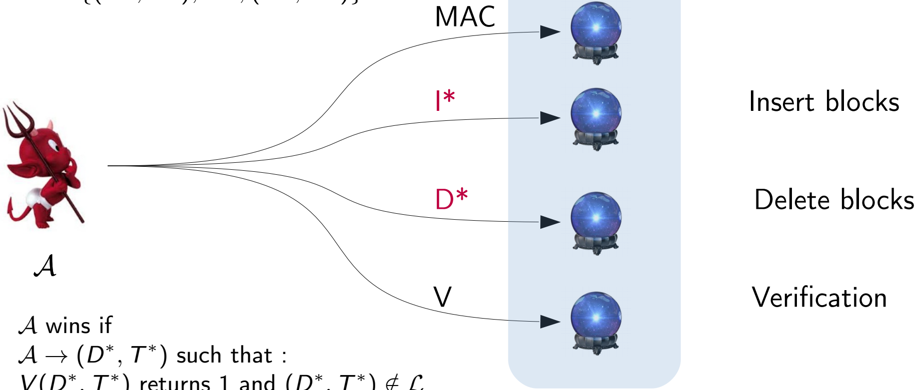
$\mathcal{A}$

$\mathcal{A}$ wins if
$\mathcal{A} \rightarrow (D^*, T^*)$ such that :
$V(D^*, T^*)$ returns 1 and $(D^*, T^*) \notin \mathcal{L}$

$*(D^i, T^i) \in \mathcal{L}$!

Security Notion 2: Tamper-proof Security Model

$\mathcal{L} := \{(D^1, T^1), \ldots, (D^q, T^q)\}$

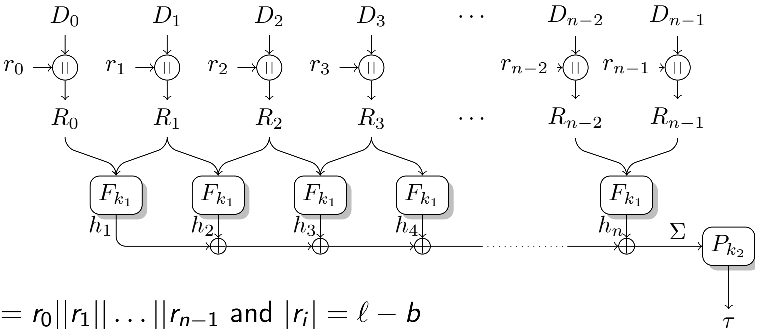


$\mathcal{A}$ wins if
$\mathcal{A} \rightarrow (D^*, T^*)$ such that :
$V(D^*, T^*)$ returns 1 and $(D^*, T^*) \notin \mathcal{L}$

*~~$(D^i, T^i) \in \mathcal{L}$~~!

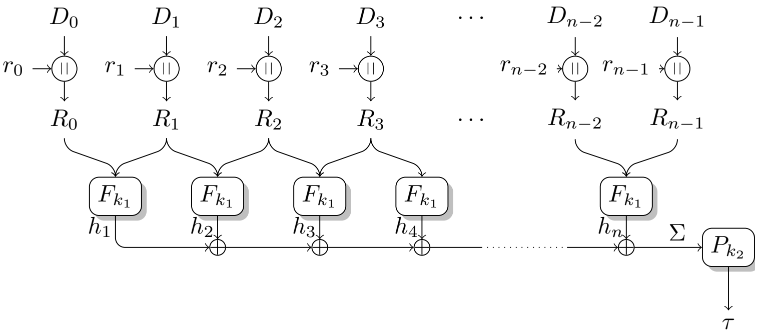
Chained Xor-Scheme ('94)

- Pair block chaining algorithm
  - $F : \mathcal{K}_F \times \{0,1\}^{2\ell} \to \{0,1\}^L$
  - $P : \mathcal{K}_P \times \{0,1\}^L \to \{0,1\}^L$
- In: Document D (n blocks D_i)
- Out: Tag T such that $T := (r, \tau)$



$r = r_0||r_1||\dots||r_{n-1}$ 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 $\Sigma = \Sigma^*$

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:

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

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

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

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:

$$\begin{array}{cccccccc} (R_0, R_1) & (R_1, R_2) & (R_2, R_1) & (R_1, R_2) & (R_2, R_1) & (R_1, R_2) & & \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & \\ h_1 & \not{h_2} & \not{h_2}' & \not{h_2} & \not{h_2}' & h_2 & = & \Sigma^* \end{array}$$

$$\underbrace{\not{h_2} \quad \not{h_2}' \quad \not{h_2} \quad \not{h_2}'}_{=0}$$

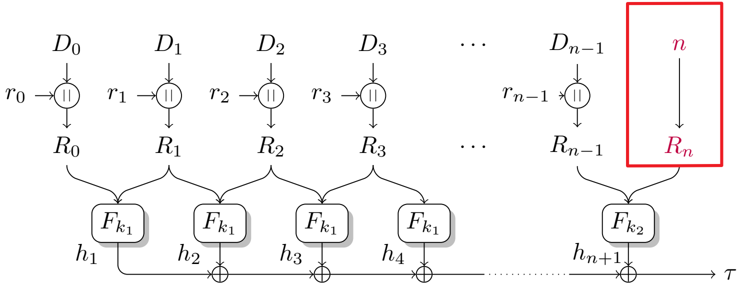
$D^* = D_0||D_1||D_2||D_1||D_2||D_1||D_2$
$r^* = r_0||r_1||r_2||r_1||r_2||r_1||r_2$

$\tau^* = \tau$ and $T^* = (r^*, \tau^*)$
$(D^*, T^*) \neq (D, T)$

More attacks in the paper...

Modified Xor-Scheme 1

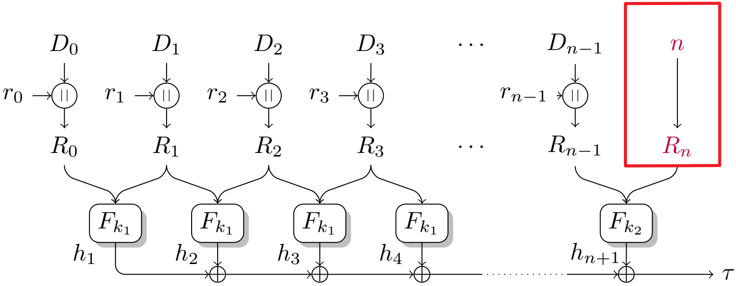
Remark: If τ depends on the doc length, previous attacks fail!
Idea: Add a block with block number *n*.



- Remove the permutation,
- Use of a different key k_2 for the last F call (Domain separation).

Modified Xor-Scheme 1

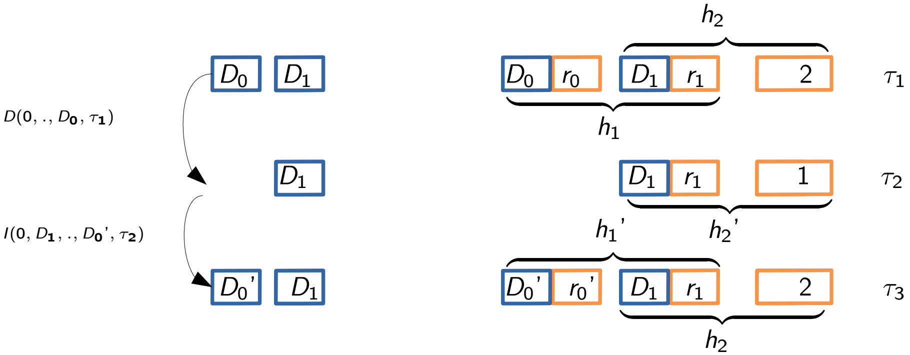
Remark: If τ depends on the doc length, previous attacks fail!
Idea: Add a block with block number *n*.



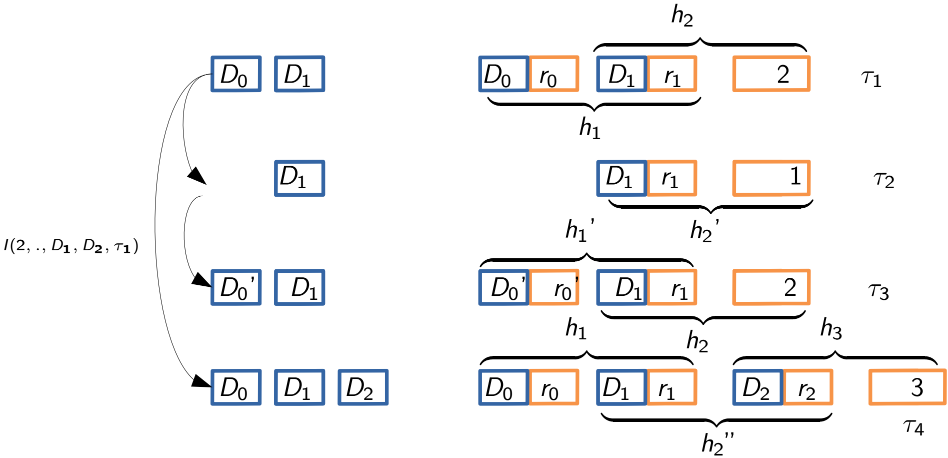
- Remove the permutation,
- Use of a different key k_2 for the last F call (Domain separation).

Still vulnerable...

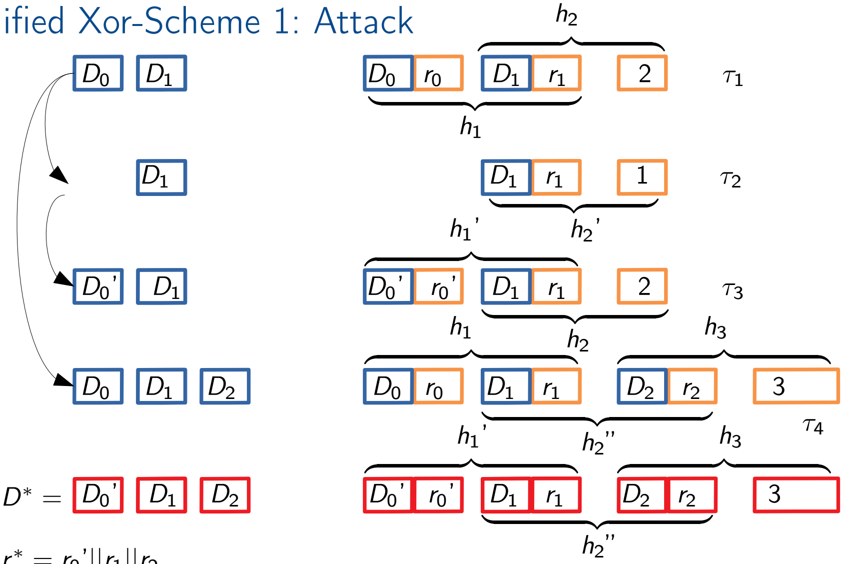
Modified Xor-Scheme 1: Attack



Modified Xor-Scheme 1: Attack



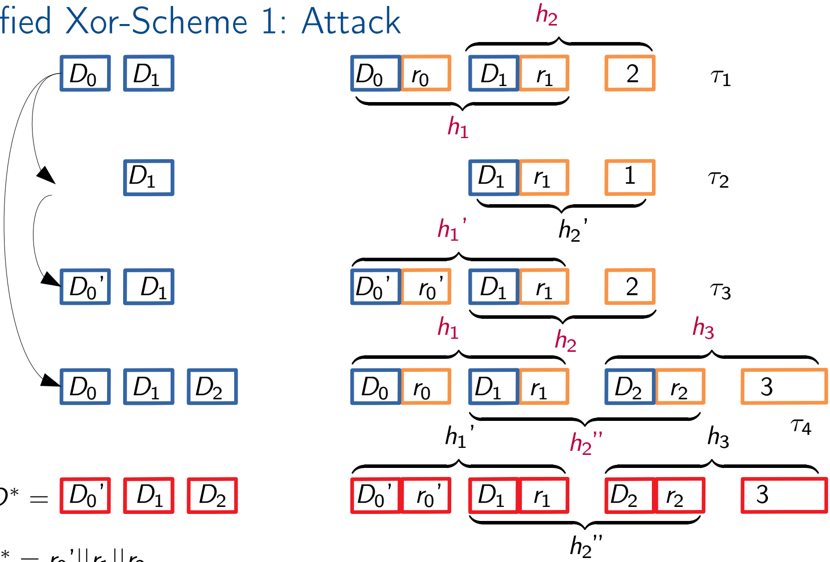
Modified Xor-Scheme 1: Attack



$$r^* = r_0'||r_1||r_2$$

$$\tau^* = \tau_1 \oplus \tau_3 \oplus \tau_4$$

Modified Xor-Scheme 1: Attack

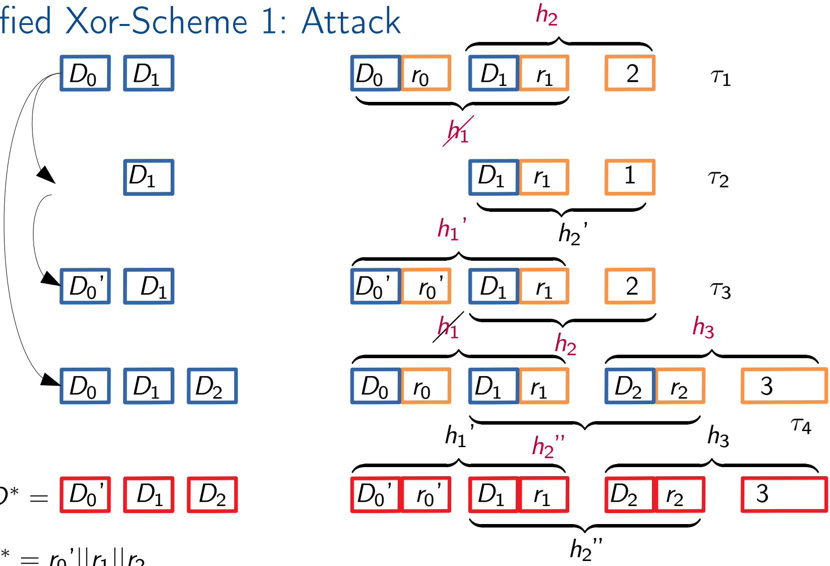


$D^* =$ D_0' D_1 D_2

$r^* = r_0' || r_1 || r_2$

$\tau^* = \tau_1 \oplus \tau_3 \oplus \tau_4$

Modified Xor-Scheme 1: Attack

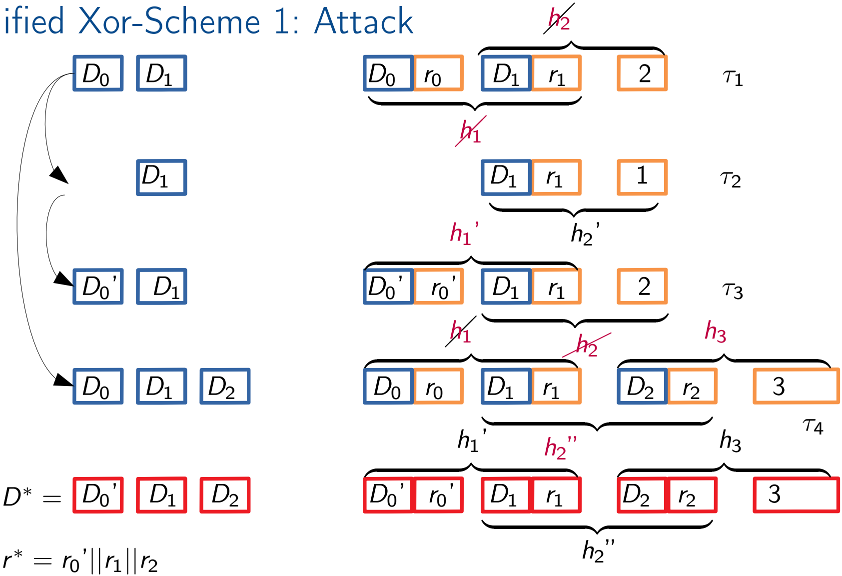


$D^* = $ D_0' D_1 D_2

$r^* = r_0'||r_1||r_2$

$\tau^* = \tau_1 \oplus \tau_3 \oplus \tau_4$

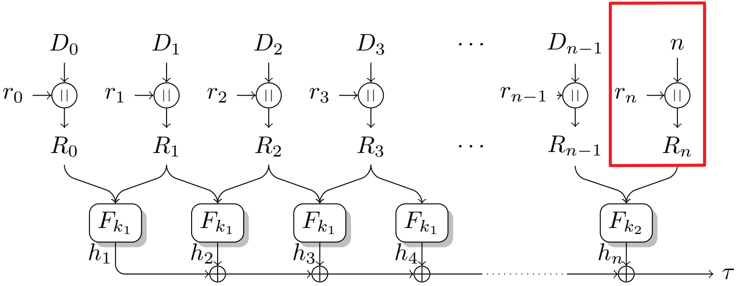
Modified Xor-Scheme 1: Attack



$r^* = r_0' || r_1 || r_2$

$\tau^* = \tau_1 \oplus \tau_3 \oplus \tau_4$

Modified Xor-Scheme 2



- A **fresh** value r_n for each update operation.

Modified XOR-Scheme 2: Security Proof

- $\mathcal{A}$ makes q (mac and inc) queries and 1 verify query

$$
\left.\begin{array}{llll}
1 & (D_0^1, D_1^1, \ldots, D_{t_1-1}^1) & (r_0^1, r_1^1, \ldots, r_{t_1}^1) & \tau_1 \\
2 & (D_0^2, D_1^2, \ldots, D_{t_2-1}^2) & (r_0^2, r_1^2, \ldots, r_{t_2}^2) & \tau_2 \\
 & \ldots & \ldots & \ldots \\
i & (D_0^i, D_1^i, \ldots, D_{t_i-1}^i) & (r_0^i, r_1^i, \ldots, r_{t_i}^i) & \tau_i \\
j & (D_0^j, D_1^j, \ldots, D_{t_j-1}^j) & (r_0^j, r_1^j, \ldots, r_{t_j}^j) & \tau_j \\
 & \ldots & \ldots & \ldots \\
q & (D_0^q, D_1^q, \ldots, D_{t_q-1}^q) & (r_0^q, r_1^q, \ldots, r_{t_q}^q) & \tau_q
\end{array}\right\} \text{mac/inc queries}
$$

$$
1 \quad (D_0^*, D_1^*, \ldots, D_{t-1}^*) \quad (r_0^*, r_1^*, \ldots, r_t^*) \quad \tau^* \; \} \text{ Verif query}
$$

Follow the security proof strategy of "XOR MACs" paper.

Modified XOR-Scheme 2: Security Proof

$$\left.\begin{array}{llll}
1 & (D_0^1, D_1^1, \ldots, D_{t_1-1}^1) & (r_0^1, r_1^1, \ldots, r_{t_1}^1) & \tau_1 \\
2 & (D_0^2, D_1^2, \ldots, D_{t_2-1}^2) & (r_0^2, r_1^2, \ldots, r_{t_2}^2) & \tau_2 \\
 & \ldots & \ldots & \ldots \\
i & (D_0^i, D_1^i, \ldots, D_{t_i-1}^i) & (r_0^i, r_1^i, \ldots, {\color{red} r_{t_i}^i}) & \tau_i \\
j & (D_0^j, D_1^j, \ldots, D_{t_j-1}^j) & (r_0^j, r_1^j, \ldots, {\color{red} r_{t_j}^j}) & \tau_j \\
 & \ldots & \ldots & \ldots \\
q & (D_0^q, D_1^q, \ldots, D_{t_q-1}^q) & (r_0^q, r_1^q, \ldots, r_{t_q}^q) & \tau_q
\end{array}\right\} \text{mac/inc queries}$$

$$1 \quad (D_0^*, D_1^*, \ldots, D_{t-1}^*) \quad (r_0^*, r_1^*, \ldots, r_t^*) \quad \tau^* \; \} \; \text{Verif query}$$

- Case 1: $\exists(i, j)$ st ${\color{red} r_{t_i}^i = r_{t_j}^j} \rightarrow$ Birthday Bound

Modified XOR-Scheme 2: Security Proof

All different

1	$(D_0^1, D_1^1, \ldots, D_{t_1-1}^1)$	$(r_0^1, r_1^1, \ldots, r_{t_1}^1)$	τ_1	mac/inc queries
2	$(D_0^2, D_1^2, \ldots, D_{t_2-1}^2)$	$(r_0^2, r_1^2, \ldots, r_{t_2}^2)$	τ_2	
	$\ldots$	$\ldots$	$\ldots$	
i	$(D_0^i, D_1^i, \ldots, D_{t_i-1}^i)$	$(r_0^i, r_1^i, \ldots, r_{t_i}^i)$	τ_i	
j	$(D_0^j, D_1^j, \ldots, D_{t_j-1}^j)$	$(r_0^j, r_1^j, \ldots, r_{t_j}^j)$	τ_j	
	$\ldots$	$\ldots$	$\ldots$	
q	$(D_0^q, D_1^q, \ldots, D_{t_q-1}^q)$	$(r_0^q, r_1^q, \ldots, r_{t_q}^q)$	τ_q	
1	$(D_0^*, D_1^*, \ldots, D_{t-1}^*)$	$(r_0^*, r_1^*, \ldots, r_t^*)$	τ^*	Verif query

- Case 1: $\exists(i,j)$ st $r_{t_i}^i = r_{t_j}^j \rightarrow$ Birthday Bound
- Case 2: $\forall(i,j)$ $r_{t_i}^i \neq r_{t_j}^j$

Modified XOR-Scheme 2: Security Proof

All different

1	$(D_0^1, D_1^1, \ldots, D_{t_1-1}^1)$	$(r_0^1, r_1^1, \ldots, r_{t_1}^1)$	τ_1	mac/inc queries
2	$(D_0^2, D_1^2, \ldots, D_{t_2-1}^2)$	$(r_0^2, r_1^2, \ldots, r_{t_2}^2)$	τ_2	
	$\ldots$	$\ldots$	$\ldots$	
i	$(D_0^i, D_1^i, \ldots, D_{t_i-1}^i)$	$(r_0^i, r_1^i, \ldots, r_{t_i}^i)$	τ_i	
j	$(D_0^j, D_1^j, \ldots, D_{t_j-1}^j)$	$(r_0^j, r_1^j, \ldots, r_{t_j}^j)$	τ_j	
	$\ldots$	$\ldots$	$\ldots$	
q	$(D_0^q, D_1^q, \ldots, D_{t_q-1}^q)$	$(r_0^q, r_1^q, \ldots, r_{t_q}^q)$	τ_q	
1	$(D_0^*, D_1^*, \ldots, D_{t-1}^*)$	$(r_0^*, r_1^*, \ldots, r_t^*)$	τ^*	Verif query

- Case 2: $\forall (i,j)\ r_{t_i}^i \neq r_{t_j}^j$
  - Case a: $\forall i \in \{0, \ldots, q\}$, $(t^*, r_t^*) \neq (t_i, r_{t_i}^i)$
    $\rightarrow F_{k_2}(D_{t-1}^* || r_{t-1}^* || t || r_t^*)$ impredictible

Modified XOR-Scheme 2: Security Proof

All different

1	$(D_0^1, D_1^1, \ldots, D_{t_1-1}^1)$	$(r_0^1, r_1^1, \ldots, r_{t_1}^1)$	τ_1	mac/inc queries
2	$(D_0^2, D_1^2, \ldots, D_{t_2-1}^2)$	$(r_0^2, r_1^2, \ldots, r_{t_2}^2)$	τ_2	
	$\ldots$	$\ldots$	$\ldots$	
i	$(D_0^i, D_1^i, \ldots, D_{t_i-1}^i)$	$(r_0^i, r_1^i, \ldots, r_{t_i}^i)$	τ_i	
j	$(D_0^j, D_1^j, \ldots, D_{t_j-1}^j)$	$(r_0^j, r_1^j, \ldots, r_{t_j}^j)$	τ_j	
	$\ldots$	$\ldots$	$\ldots$	
q	$(D_0^q, D_1^q, \ldots, D_{t_q-1}^q)$	$(r_0^q, r_1^q, \ldots, r_{t_q}^q)$	τ_q	
1	$(D_0^*, D_1^*, \ldots, D_{t-1}^*)$	$(r_0^*, r_1^*, \ldots, r_t^*)$	τ^*	Verif query

- Case 2: $\forall (i,j)\ r_{t_i}^i \neq r_{t_j}^j$
  - Case b: $\exists i \in \{0, \ldots, q\}$ st $(t^*, r_t^*) = (t_i, r_{t_i}^i)$
    Inputs of F_{k_1}: $\exists$ at least one block $(D_n^* || r_n^* || D_{n+1}^* || r_{n+1}^*)$
    $\neq$ all input blocks $(D_n^i || r_n^i || D_{n+1}^i || r_{n+1}^i)$

Conclusion

- Modified-Xor-Scheme strongly incremental and secure,
- Not suitable in practice,
- Is it possible to have a strongly incremental MAC that is practical?

Conclusion

- Modified-Xor-Scheme strongly incremental and secure,
- Not suitable in practice,
- Is it possible to have a strongly incremental MAC that is practical?
  - Yes? Which design
  - No? Impossibility result

Thank you for your attention!

Questions?

Analysis and Improvement of an Authentication Scheme in Incremental Cryptography

Louiza Khati[1,2], Damien Vergnaud[2,3]

1 ANSSI
2 ENS
3 LIP6

Last update:

Friday, August 17th, 2018