Secure Message Authentication in the Presence of Leakage and Faults

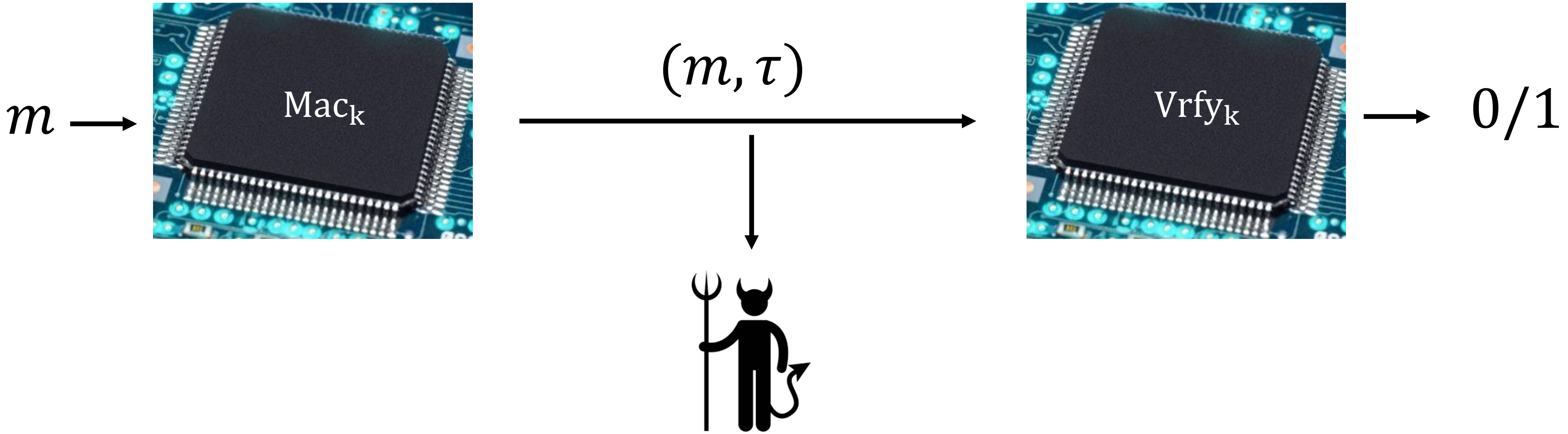
Francesco Berti, Chun Guo, Thomas Peters
Yaobin Shen, François-Xavier Standaert

March 22, FSE 2023@Beijing, China

Outline

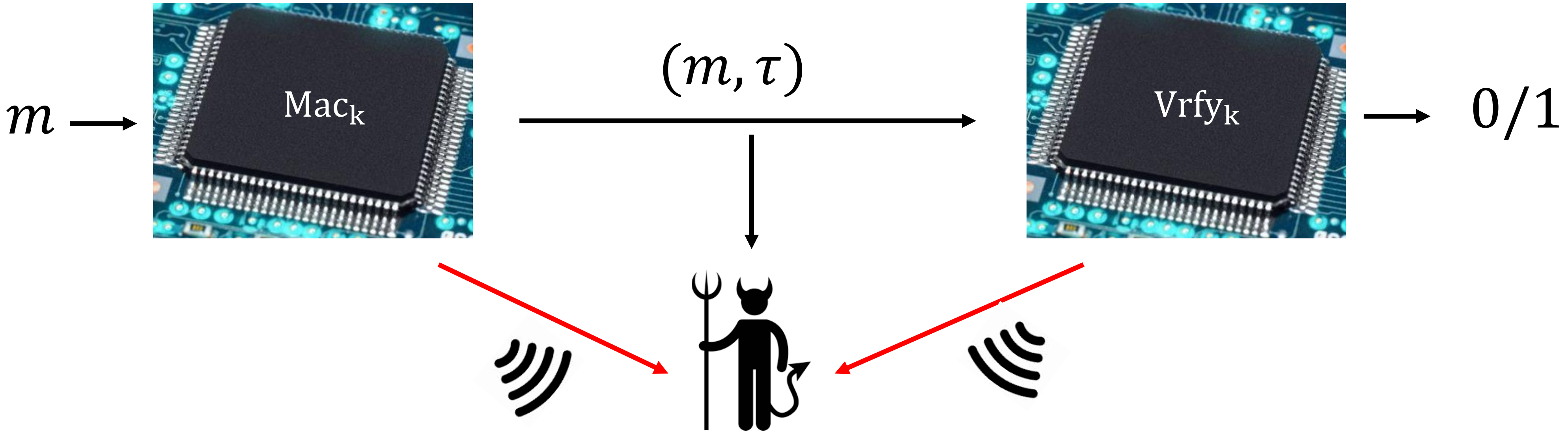
- Motivation
- Contribution
- Conclusion

Message Authentication Codes (MACs)



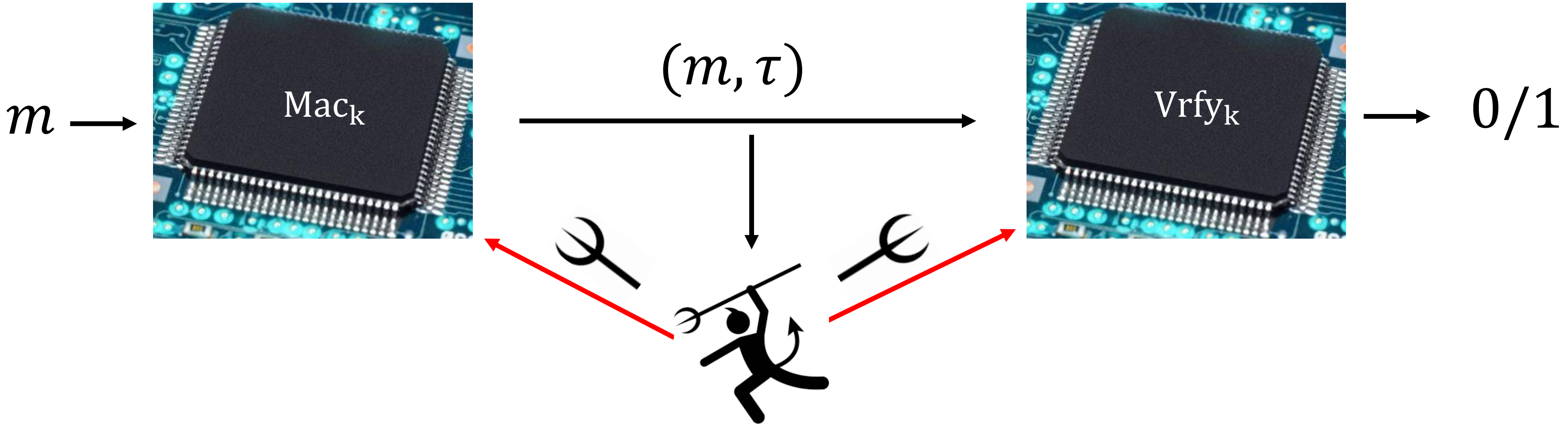
- Black-box secure message authentication codes to ensure integrity
  - attacker knows algorithm and only sees inputs/outputs
  - the key is kept secret
  - internal states are secret

MACs against Side-Channel Attacks (SCA)



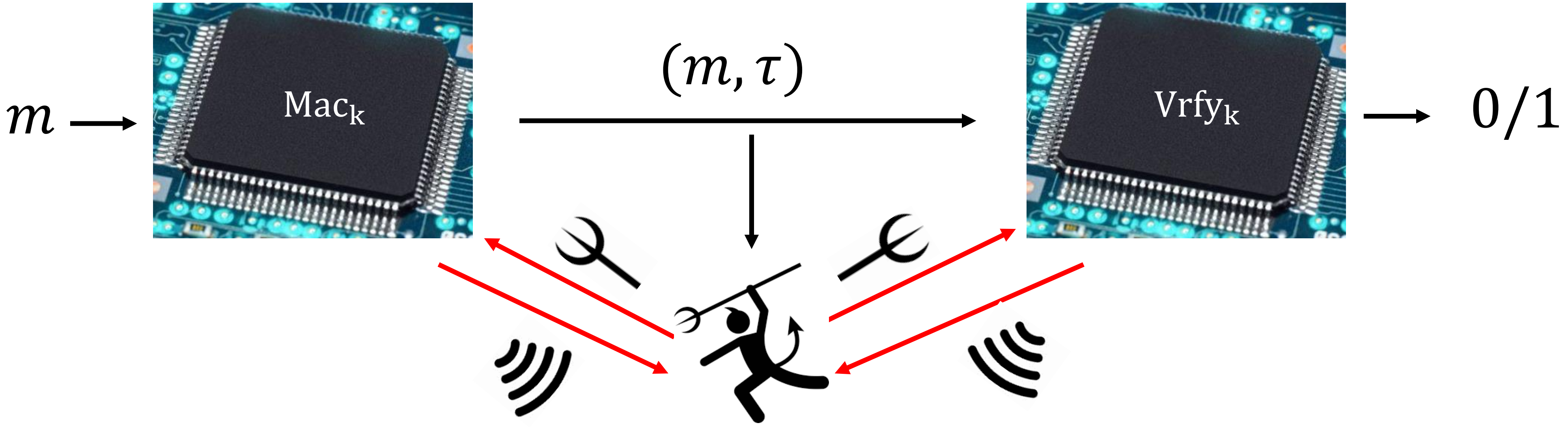
- Side-channel attacks (time, power consumption, Electromagnetic radiation)
  - the information of key may be leaked
  - the internal values may be leaked

MACs against Faults Attacks (FA)



- Faults attacks (voltage glitch, electromagnetic pulse, LASER,…)
  - the key may be influenced
  - the internal values may be influenced

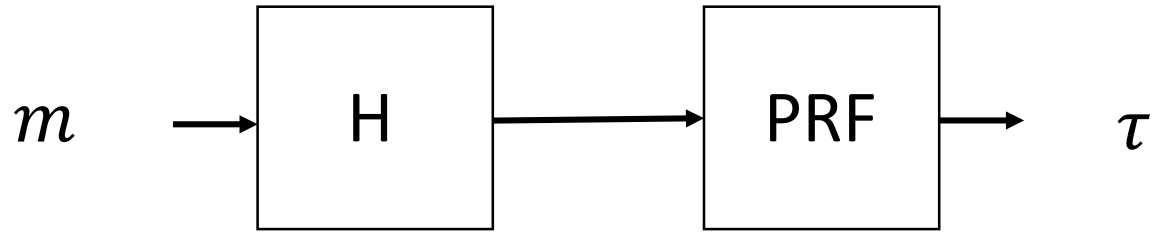
MACs against both SCA and FA



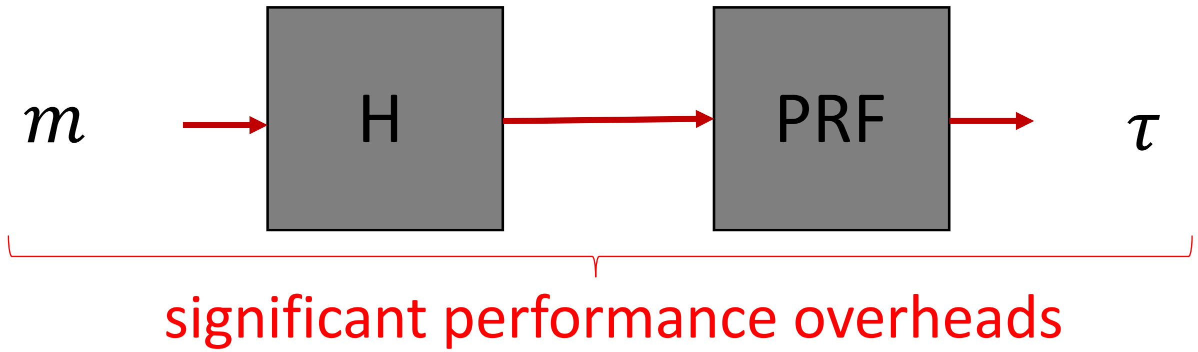
- Combined attacks: side-channel and faults attacks
  - the key may be leaked and influenced
  - the internal values may be leaked and influenced

How to Protect against Leakage and Faults

- Hash-then-PRF: a popular way to design a MAC

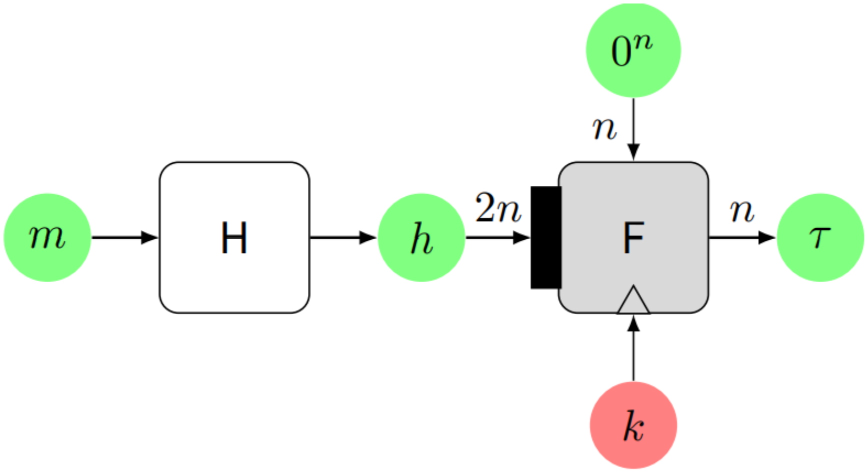


- Protection against side-channel and faults, e.g., masking + redundancy



How to Improve the Performance

- Leveled implementation [PSV15]
  - avoid equally protecting all parts of an implementation
  - identify the protection level of each part (performance gains)
- LR-MAC1 [BGPS21] : unbounded leakage for hash + DPA-protected TBC
  - can lead to substantial performance gains



- Can we use leveled implementation for combined attacks?
- We initiate a mode-level study of MACs against side-channel and faults attacks in leveled implemetation

Outline

- Motivation
- Contribution
- Conclusion

Our Contribution: Overview

- A model to capture both leakage and faults
  - assume some atomic components that out of control of the adversary

- Show that LR-MAC1[BGPS21] is secure if only the verification is faulted
  - attack when tag generation is faulted

- Propose two MACs that are both fault-resilience and leakage-resilience
  - LR-MACd can resist one fault injection
  - LR-MACr can resist multiple fault injections with an additional randomness

	Faults Vrfy	Faults Mac	Fault types	#protected TBCs
LR-MAC1	✓	✗	SaF&DF, multiple	1
LR-MACd	✓	✓	SaF&DF, 1	2
LR-MACr	✓	✓	DF, multiple	1

SaF: Stuck-at-Faults, DF: Differential Faults

Modeling Faults (1/2)

- For a algorithm $\mathrm{y} = \mathrm{Algo_k}(\mathrm{x})$ with implementation $(f_1, \ldots, f_m)$
  - use **dependency matrix** to define this implementation
  - each item of dependency matrix may be faulted

$$
\begin{aligned}
f_1(x_1, x_2, \ldots, x_n) &= y_1, \\
f_2(x_1, x_2, \ldots, x_n, y_1) &= y_2, \\
&\vdots \\
f_m(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_{m-1}) &= y_m,
\end{aligned}
$$

Implementation

transform →

$$
\begin{pmatrix}
\tilde{x}_{11} & \tilde{x}_{12} & \cdots & \tilde{x}_{1n} & \varepsilon & \varepsilon & \cdots & \varepsilon \\
\tilde{x}_{21} & \tilde{x}_{22} & \cdots & \tilde{x}_{2n} & \tilde{y}_{21} & \varepsilon & \cdots & \varepsilon \\
\vdots & & & \vdots & & & \ddots & \vdots \\
\tilde{x}_{m1} & \tilde{x}_{m2} & \cdots & \tilde{x}_{mn} & \tilde{y}_{m1} & \tilde{y}_{m2} & \cdots & \tilde{y}_{m\,m-1}
\end{pmatrix}
$$

Dependency matrix

- Example: implementation (f_1, f_2, f_3), input (x_1, x_2)
  - f_1 takes x_1 as input
  - f_2 takes x_2 as input
  - f_3 takes x_1, y_1, y_2 as input

$$
\begin{pmatrix}
x_1 & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & x_2 & \varepsilon & \varepsilon \\
x_1 & \varepsilon & y_1 & y_2
\end{pmatrix}
$$

Dependency matrix

Modeling Faults (2/2)

- Faulty matrix to capture injected faults
  - faulted values: $x_1 \to x_1', y_2 \to y_2'$
  - non-faulted values are represented by the dot "$\cdot$ “
  - symbol $\perp$ means this value is protected against faults

$$\begin{pmatrix} x_1 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & x_2 & \varepsilon & \varepsilon \\ x_1 & \varepsilon & y_1 & y_2 \end{pmatrix}$$

Dependency matrix

inject faults

$$\begin{pmatrix} x_1' & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \cdot & \varepsilon & \varepsilon \\ \cdot & \varepsilon & \perp & y_2' \end{pmatrix}$$

Faulty matrix

- Two faults considered in our work
  - stuck-at faults: can replace the bits of x by any value
  - differential faults: can xor Δ to the value x

Modeling Leakage

- For a algorithm $\mathrm{y} = \mathrm{Algo_k(x)}$ with implementation $(f_1, \dots, f_m)$
  - associate a leakage function $\mathrm{L_i}$ for each f_i, and $\mathrm{L_{Algo}} = (\mathrm{L_1}, \dots, \mathrm{L_m})$
  - write $\mathrm{LAlgo_k(x)}$ for the leaky algorithm $\approx \mathrm{Algo_k(x)}$ + the output of $\mathrm{L_{Algo}}$
- Naturally, define faulty leaky algorithm as $\mathrm{LAlgo_k(x, z)}$ where z is the faulty tuple
- Example: $z = (x_1', \cdot, \cdot, y_2')$ in the reading direction
  - then $\mathrm{LAlgo_k(x, z)}$ is the faulty leaky algorithm

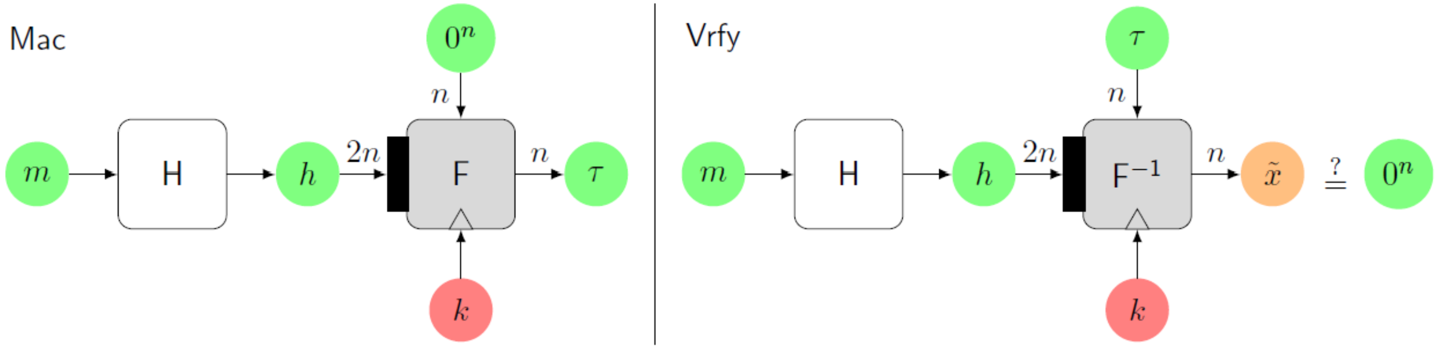
$$\begin{pmatrix} x_1' & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \cdot & \varepsilon & \varepsilon \\ \cdot & \varepsilon & \perp & y_2' \end{pmatrix}$$

Faulty matrix

- Some assumptions
  - the key is fault-immune
  - each f_i is regarded as a atomic component
  - Fault-then-leak model
  - unbounded faults and ℓ-bounded faults

LR-MAC1 against Leakage and Faults

- LR-MAC1 [BGPS21]
  - hash function H is ϵ_{CR}-collision resistant
  - tweakable block cipher F is ϵ_{SUP-L2}-strong unpredictable with leakage



- Advantage for stuck-at and differential fault-then-leak attacks in verification

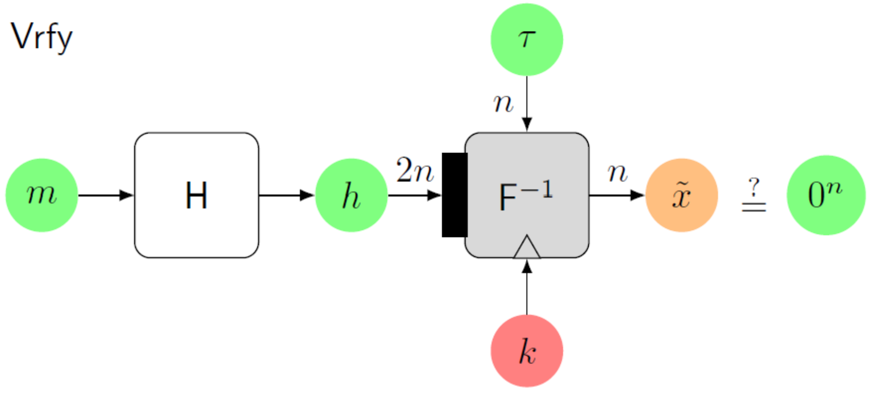
$$\epsilon \leq \epsilon_{\mathsf{CR}} + (q_V + 1)\epsilon_{\mathsf{SUP\text{-}L2}}.$$

q_V: #verification queries

- To find a valid forgery (m, τ), the adversary needs to
  - either find a collision against the hash function H
  - or find a valid tuple against the $\mathrm{SUP-L2}$ security of TBC F

Model Leakage and Faults for LR-MAC1

- LR-MAC1 resists faults in verification
  - atomic implementation $f_1 = H(\cdot), f_2 = F_k^{-1}(\cdot,\cdot)$
  - for input $(x_1, x_2) = (m, \tau)$, $y_1 = H(x_1), y_2 = F_k^{-1}(y_1, \tau)$



$$\begin{pmatrix} x_1 & \varepsilon & \varepsilon \\ \varepsilon & x_2 & y_1 \end{pmatrix}$$

Dependency matrix

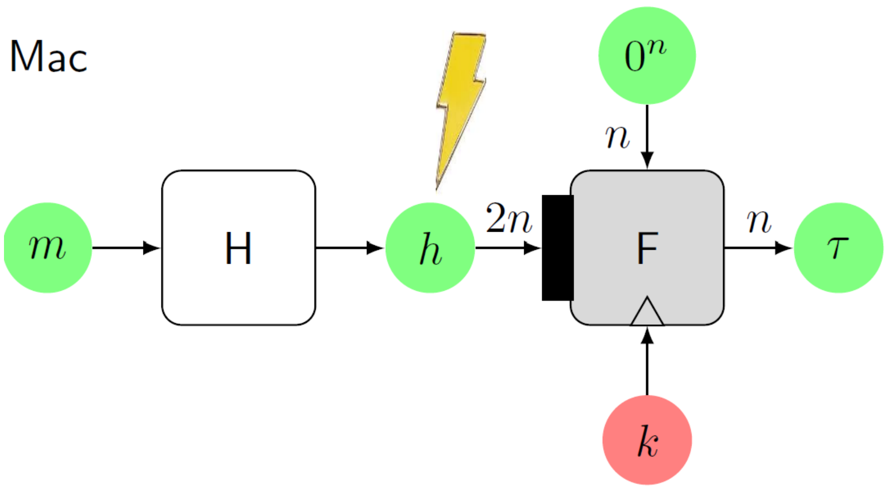
$$\mathcal{F}_{\mathsf{Vrfy}} = \begin{pmatrix} z_1 & \varepsilon, & \varepsilon \\ \varepsilon & z_2 & z_3 \end{pmatrix}$$

Faulty matrix

  - thus, a faulty leaky verification query is captured by $\mathbf{FLVrfy_k}(m, \tau, (z_1, z_2, z_3))$
- A leaky tag generation query is captured by $\mathrm{LMac_k}(m)$

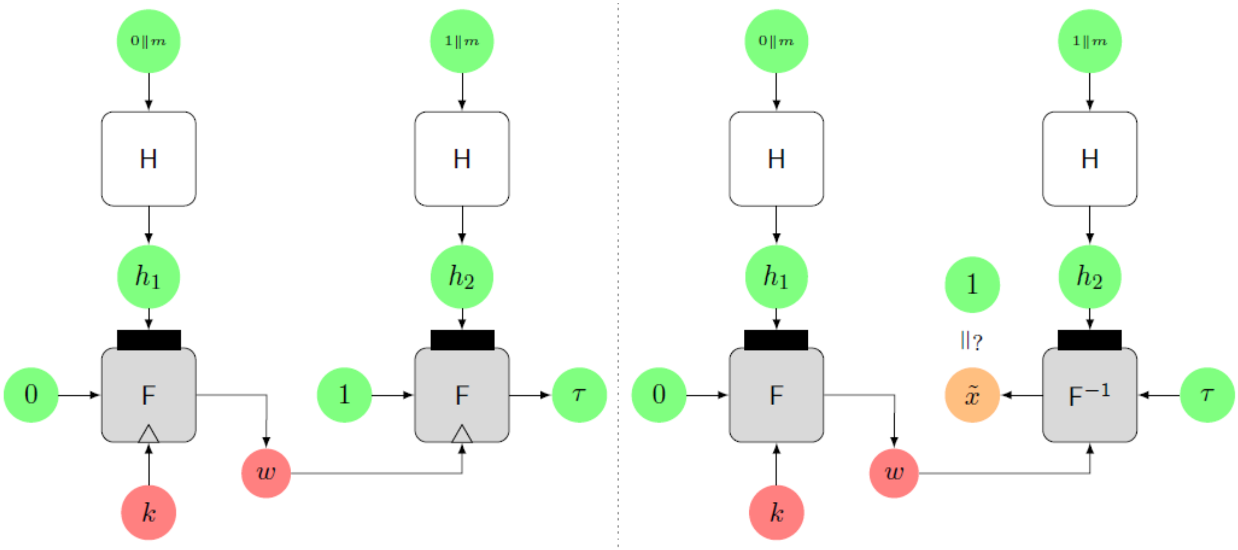
Attacks against LR-MAC1 and others

- Insecure tag generation of LR-MAC1
  - computes $h = H(m)$ and $h' = H(m')$, $\Delta = h \oplus h'$
  - queries m and injects differential fault Δ into h to obtain τ
  - (m', τ) is a valid forgery



LR-MACd: Improved Security by Iteration

- LR-MACd
  - two ϵ_{CR}-collision resistant hashes
  - two ϵ_{SPU-L2}-self-preserving unpredictable TBCs
  - the ephemeral key w of the second TBC should be protected



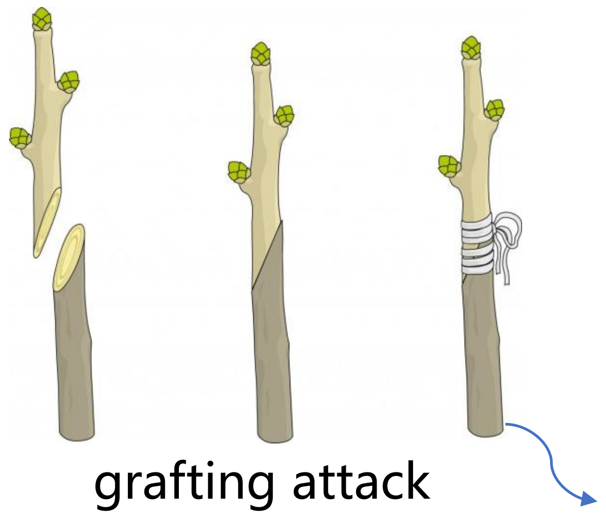
- Forge advantage for stuck-at and differential 1-bounded fault-then-leak attacks in tag generation and verification:

$$\epsilon \leq \epsilon_{\mathsf{CR}} + (q_V + 1)\epsilon_{\mathsf{SPU\text{-}L2}}.$$

q_V: #verification queries

Grating Attack on Iterative Schemes

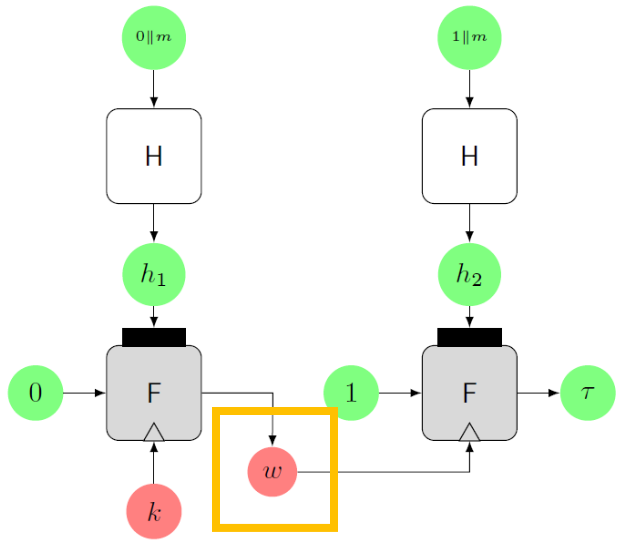
- For any iterative scheme $S(m) = F \circ H(m)$
  - queries m_1 to S and injects faulted value h^* to replace $h_1 = H(m_1)$
  - queries m_2 to S and injects faulted value h_1 to replace $h_2 = H(m_2)$, and obtain τ_2
  - (m_1, τ_2) is a valid forgery



grafting attack

forgery

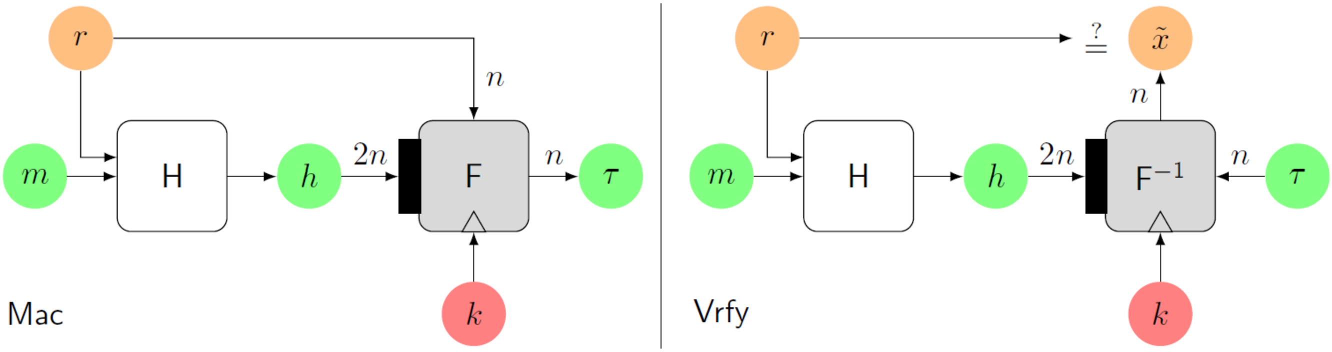
- The protection of w is necessary in LR-MACd
- By iterating, it can resist more faults



LR-MACd

LR-MACr: Improved Security with Randomness

- LR-MACr
  - H is ϵ_{CR}-collision resistant and ϵ_{PRC}-preimage resistant after computation
  - F is ϵ_{SUP-L2}-strong unpredictable with leakage
  - randomness $r \in \{0,1\}^n$ is selected for each tag generation



- Forge advantage for unbounded differential fault-then-leak attacks in tag generation and verification

$$\epsilon \leq \epsilon_{\mathsf{CR}} + (q_V + 1)\epsilon_{\mathsf{SUP\text{-}L2}} + \epsilon_{\mathsf{PRC}} + \frac{q_M^2}{2^{n+1}} + \frac{q_M}{2^n}$$

q_V: #verification queries, q_M: #generation queries

Outline

- Motivation
- Contribution
- Conclusion

Conclusion

- A model to capture both leakage and faults
- Show that LR-MAC1 is secure if only the tag verification is faulted
- Propose two MACs that are fault-resilience and leakage-resilience
  - LR-MACd
  - LR-MACr

- More in paper
  - Fault-resilience vs Fault-resistance
  - Sub-atomic faults
  - Model discussion and proof details

Thanks

yaobin.shen@uclouvain.be?