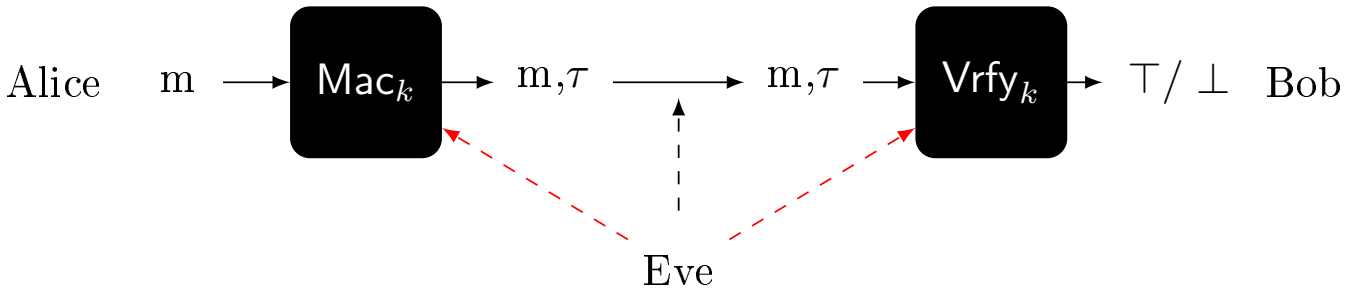
Efficient Leakage-Resilient MACs Without Idealized Assumptions

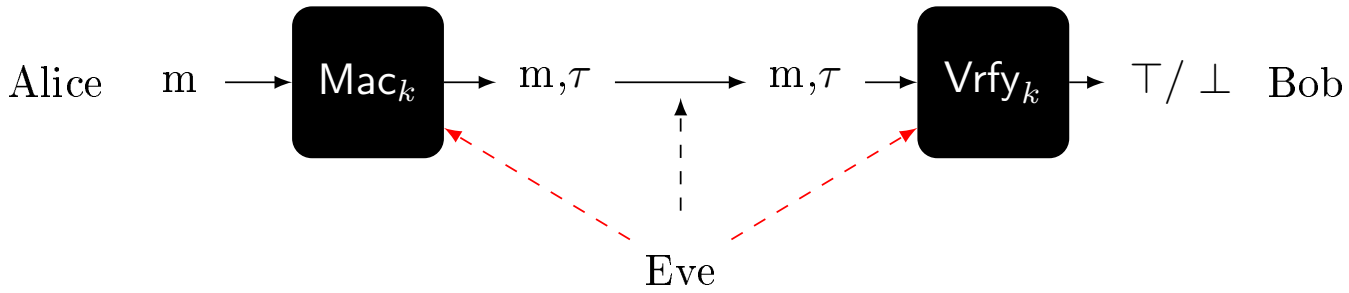
Francesco Berti, Chun Guo, Thomas Peters, François-Xavier Standaert

07-12-2021

Integrity and Leakage



Integrity and Leakage

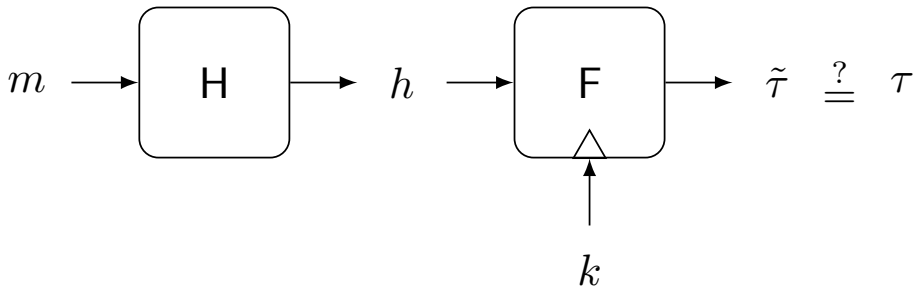


With access to the physical devices, measure

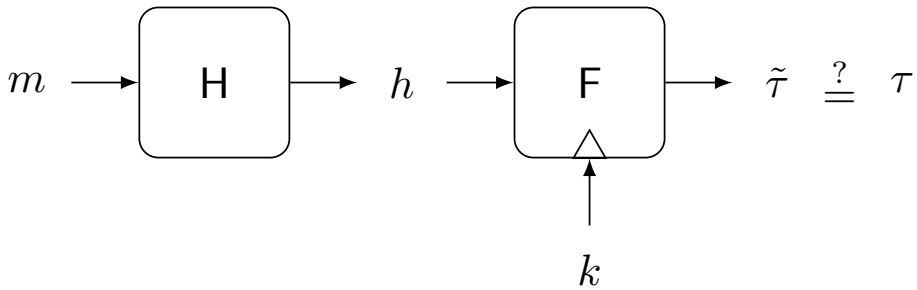
- power consumption, time, EM radiation [side-channels]

These physical observations *leak* critical information (even the full key).

Side-channel vs integrity - ex: Hash-then-BC

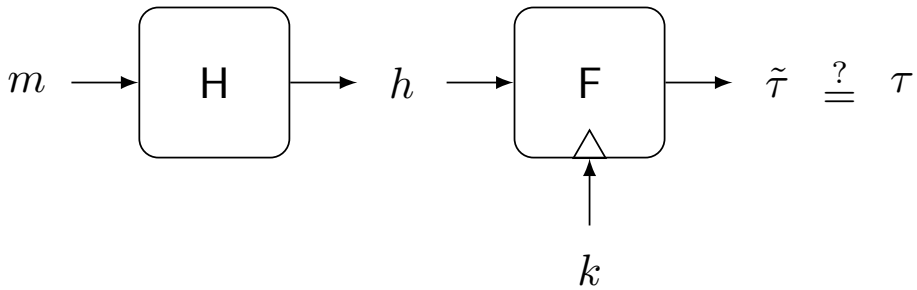


Side-channel vs integrity - ex: Hash-then-BC



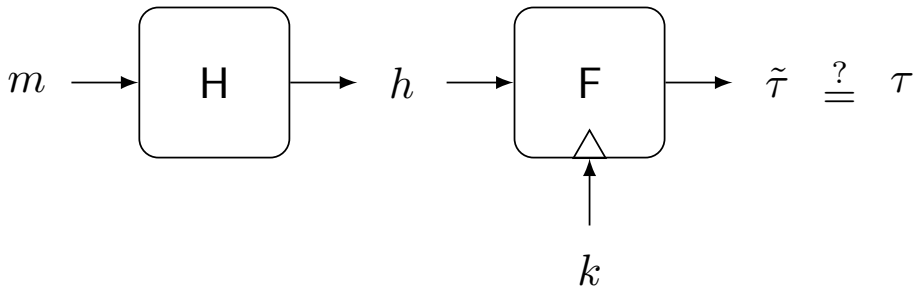
Forgery possible from k

Side-channel vs integrity - ex: Hash-then-BC



Forgery possible from k but also $\tilde{\tau}$.

Side-channel vs integrity - ex: Hash-then-BC



Forgery possible from k but also $\tilde{\tau}$.
Cryptographic device *are not* blackbox oracles.

Leakage-resilient Cryptography - Tradeoffs

- **Goal:** Security in presence of leakage.

Leakage-resilient Cryptography - Tradeoffs

- **Goal:** Security in presence of leakage.
- **Problem:** Best tradeoff
  - weak physical assumptions
  - efficient constructions

Leakage-resilient Cryptography - Tradeoffs

- **Goal:** Security in presence of leakage.
- **Problem:** Best tradeoff
  - weak physical assumptions
  - efficient constructions

Idealized assumption are not good with leakage.
Example for a TBC (tweakable blockcipher)

- **leak-free:** (ideal hypothesis)

Leakage-resilient Cryptography - Tradeoffs

- **Goal:** Security in presence of leakage.
- **Problem:** Best tradeoff
  - weak physical assumptions
  - efficient constructions

Idealized assumption are not good with leakage.
Example for a TBC (tweakable blockcipher)

- **leak-free:** (ideal hypothesis)

Problems

- leakage of an ideal object
- verifiability
- security bounds

Leakage-resilient Cryptography - Tradeoffs

- **Goal:** Security in presence of leakage.
- **Problem:** Best tradeoff
  - weak physical assumptions
  - efficient constructions

Idealized assumption are not good with leakage.
Example for a TBC (tweakable blockcipher)

- **leak-free:** (ideal hypothesis)

Problems

- leakage of an ideal object
- verifiability
- security bounds

Anyway, it gives : what to protect + efficient constructions.

Leakage-resilient Cryptography - Tradeoffs

- **Goal:** Security in presence of leakage.
- **Problem:** Best tradeoff
  - weak physical assumptions
  - efficient constructions

Idealized assumption are not good with leakage.
Example for a TBC (tweakable blockcipher)

- **leak-free:** (ideal hypothesis)

Problems

- leakage of an ideal object
- verifiability
- security bounds

Anyway, it gives : what to protect + efficient constructions.
Thus, we use *strong unpredictability with leakage* (sUL)

Results: integrity with leakage

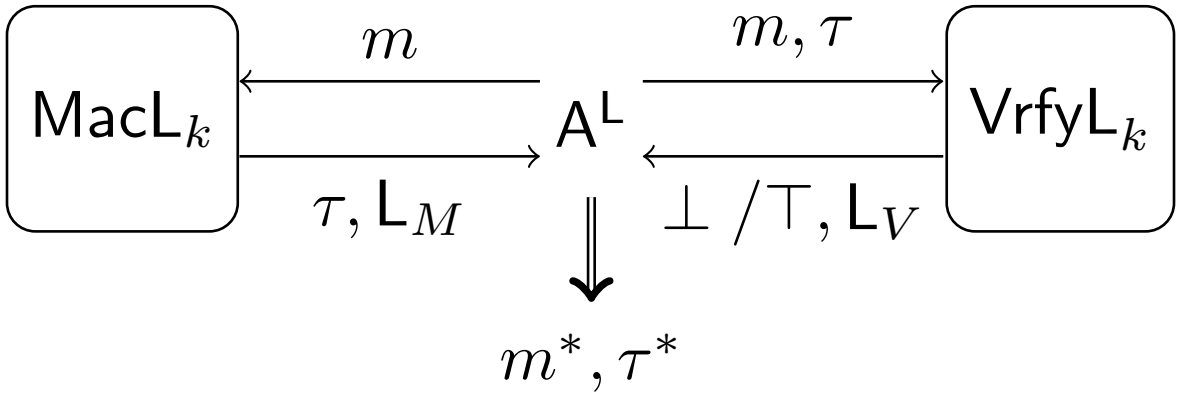
- LR–MAC1
  - 1 TBC call and a collision-resistant hash
    - Beyond birthday secure (BBB) only with long tweaks (2n)
- LR–MAC2
  - 2 TBC calls and a collision-resistant hash
    - BBB with half collision resistant hash function
- HTBC
  - 1 TBC call and a collision-resistant hash
    - security with a stronger (not idealized) assumption for the hash

Summary

1. Background
   1. Unforgeability for MAC in presence of leakage
   2. Modeling leakage - unbounded leakage model
   3. sUL - Security definition for TBC in presence of leakage
   4. HTBC - [BGPPS19]
2. LR–MAC1
3. LR–MAC2
4. HTBC

suf–L2 - Security for MACs with leakage [BGPPS19]

Strong unforgeability with Leakage

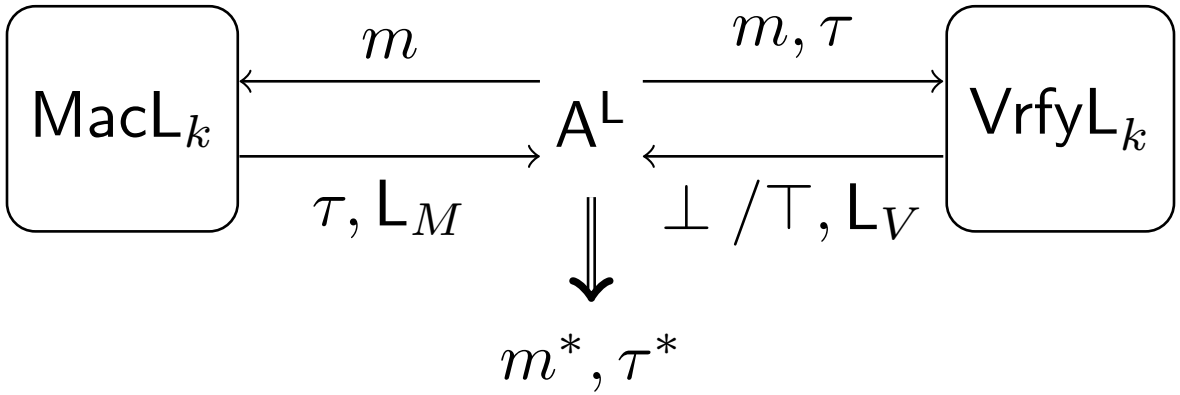


Hard to forge even if A

- model the leakage [A^{L}]
- receives L_M, leakage Mac queries
- receives L_V, leakage Vrfy queries

suf–L2 - Security for MACs with leakage [BGPPS19]

Strong unforgeability with Leakage

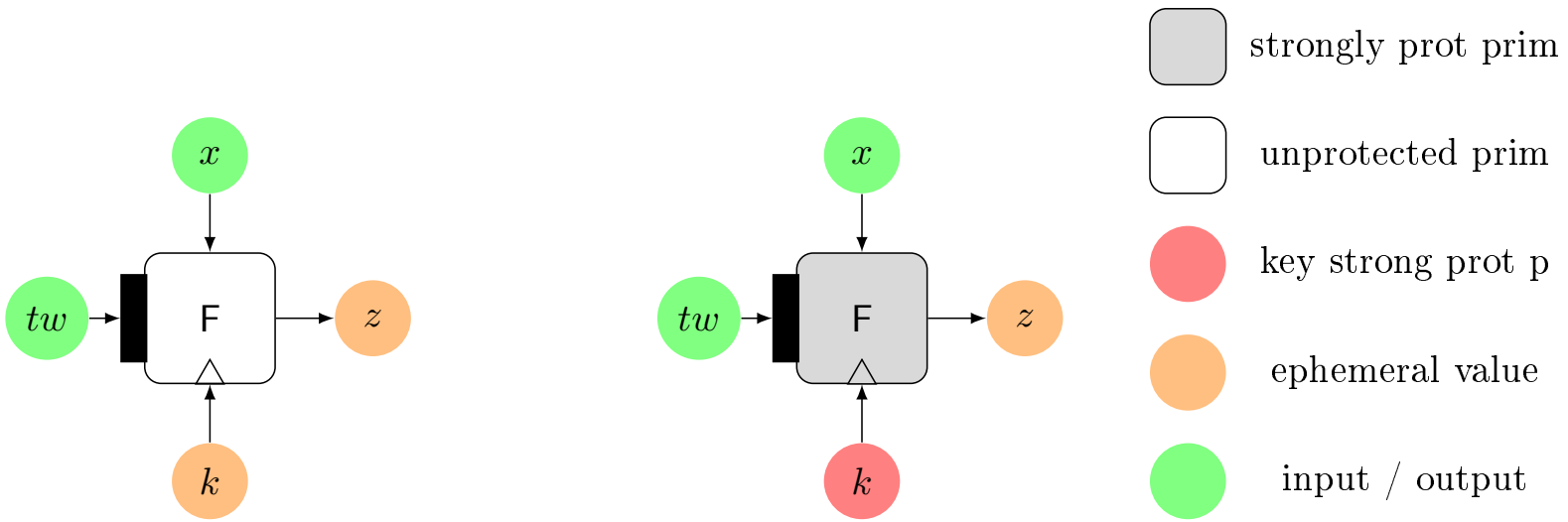


Hard to forge even if A

- model the leakage [A^{L}]
- receives L_M, leakage Mac queries
- receives L_V, leakage Vrfy queries

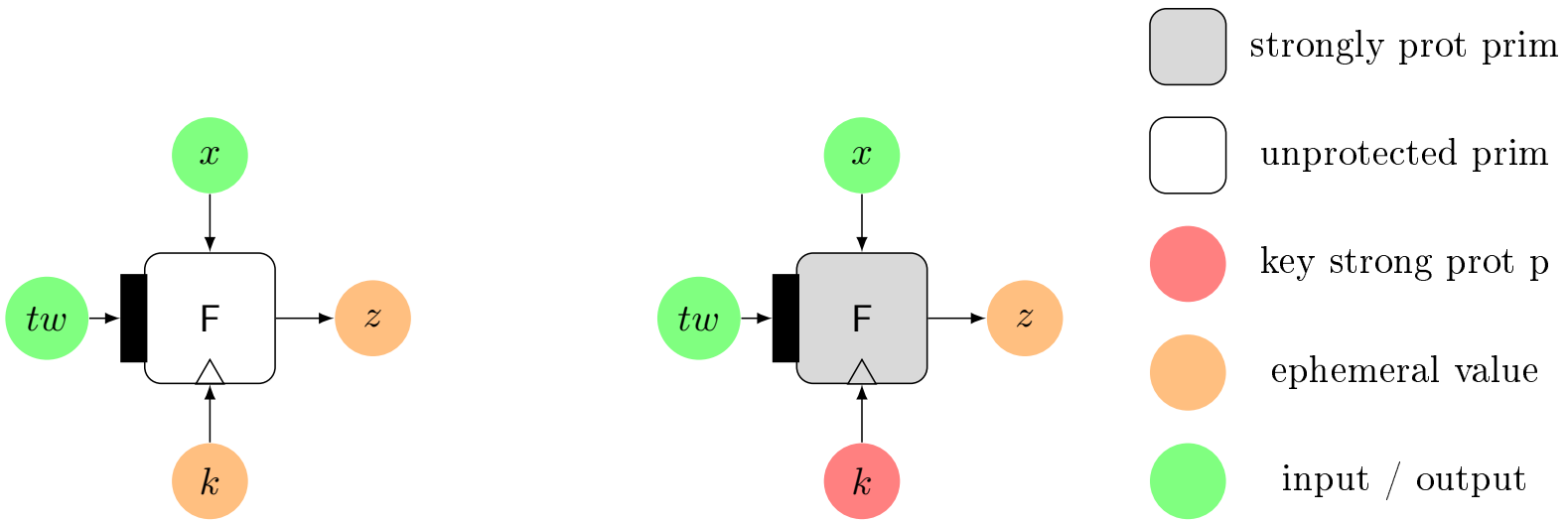
Problem: Model the leakage (L)?

Modeling leakage - Unbounded leakage model [BPPS17]



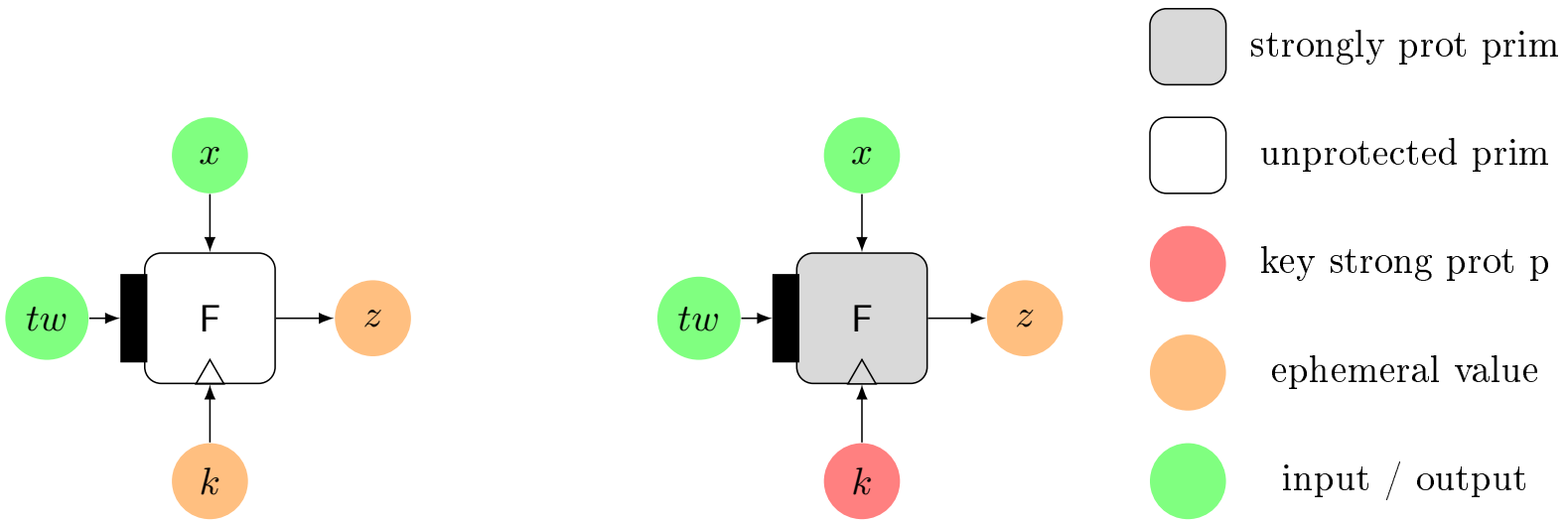
- **Unprotected primitives:** All inputs and outputs leak

Modeling leakage - Unbounded leakage model [BPPS17]



- **Unprotected primitives:** All inputs and outputs leak
- **Strongly protected primitives:** All *but not the secret ones* inputs and outputs leak

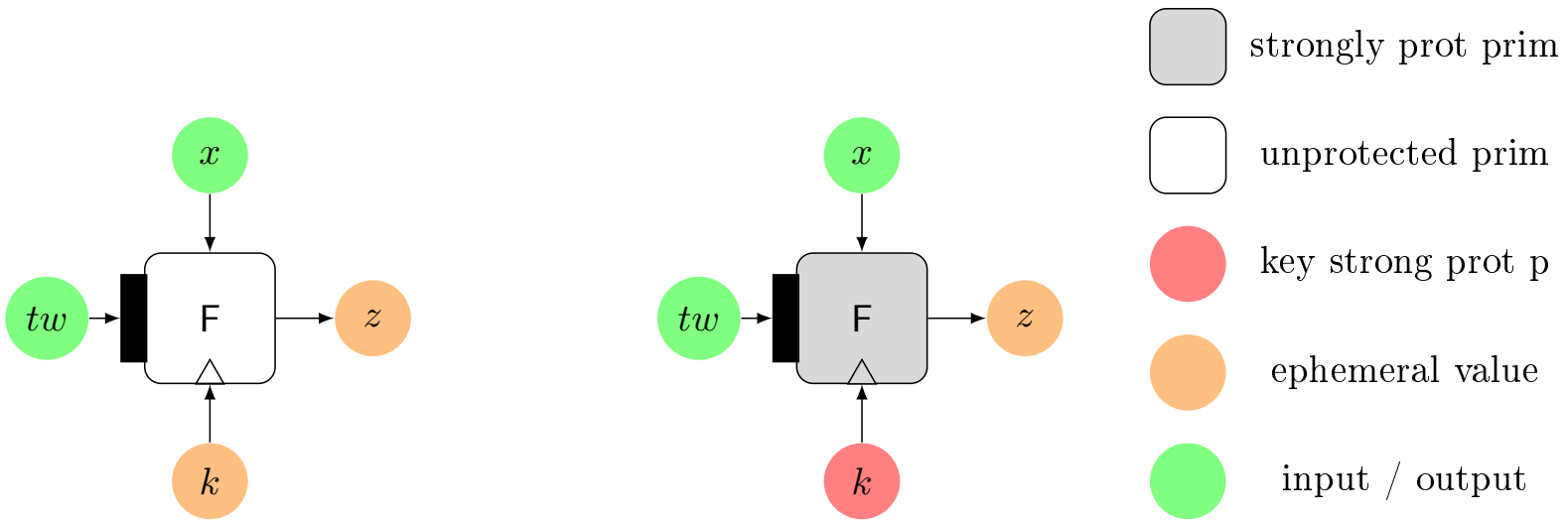
Modeling leakage - Unbounded leakage model [BPPS17]



- **Unprotected primitives:** All inputs and outputs leak
- **Strongly protected primitives:** All *but not the secret ones* inputs and outputs leak

Unbounded leakage model: orange + green.

Modeling leakage - Unbounded leakage model [BPPS17]

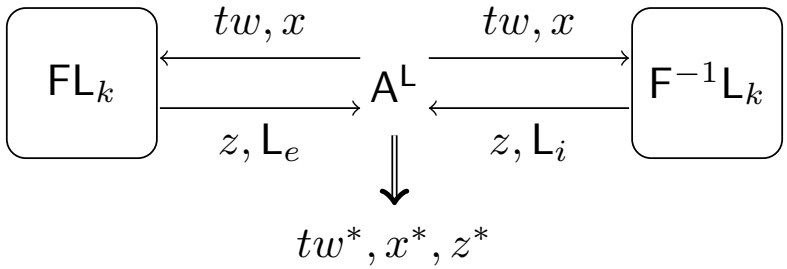


- **Unprotected primitives:** All inputs and outputs leak
- **Strongly protected primitives:** All *but not the secret ones* inputs and outputs leak

Unbounded leakage model: orange + green.
Problem: Verifiable security of the grey TBC.

sUL - Security in presence of leakage for strongly protected TBC [BGPPS19]

Strong Unpredictability with Leakage

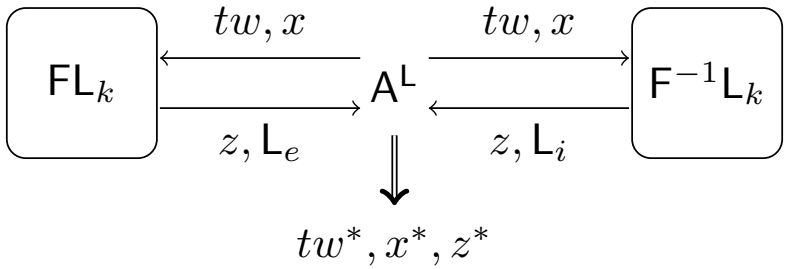


Hard for A to find a correct and fresh triple even if A

- model the leakage [A^{L}]
- receives the leakage of F queries [L_e]
- receives the leakage of F^{-1} queries F_k^{-1} [L_i]

sUL - Security in presence of leakage for strongly protected TBC [BGPPS19]

Strong Unpredictability with Leakage

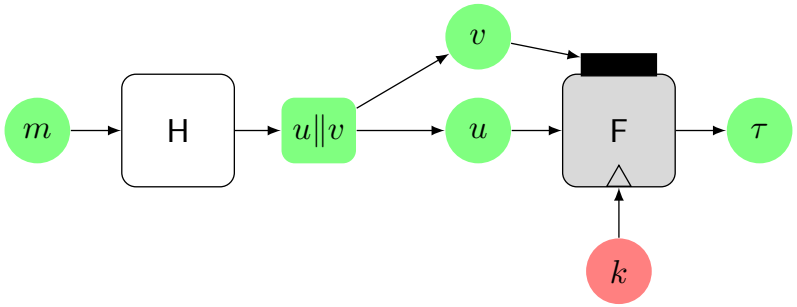


Hard for A to find a correct and fresh triple even if A

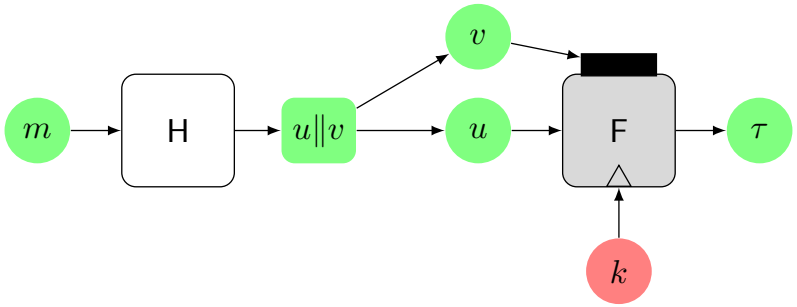
- model the leakage [A^{L}]
- receives the leakage of F queries [L_e]
- receives the leakage of F^{-1} queries F_k^{-1} [L_i]

Verifiabile by a laboratory.

HTBC - F sUL + H Random Oracle [BGPPS19]

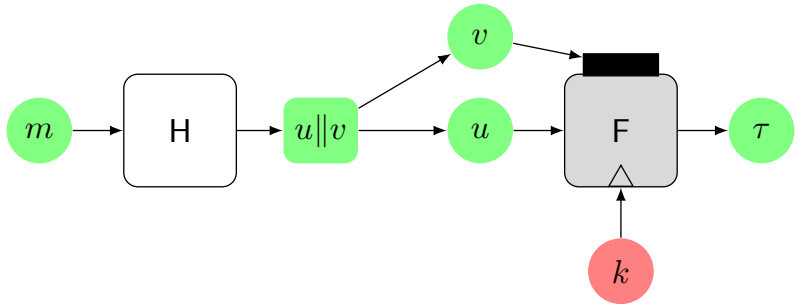


HTBC - F sUL + H Random Oracle [BGPPS19]



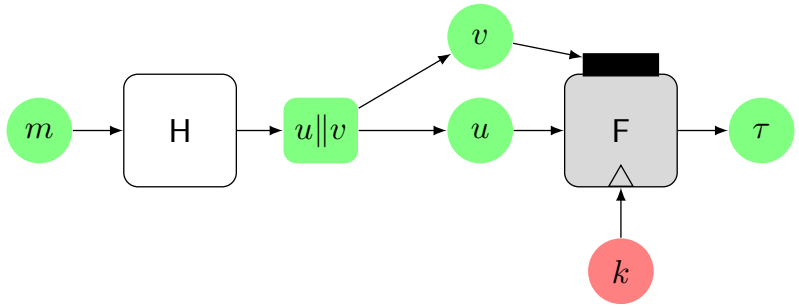
Not recomputing τ in Vrfy.

HTBC - F sUL + H Random Oracle [BGPPS19]

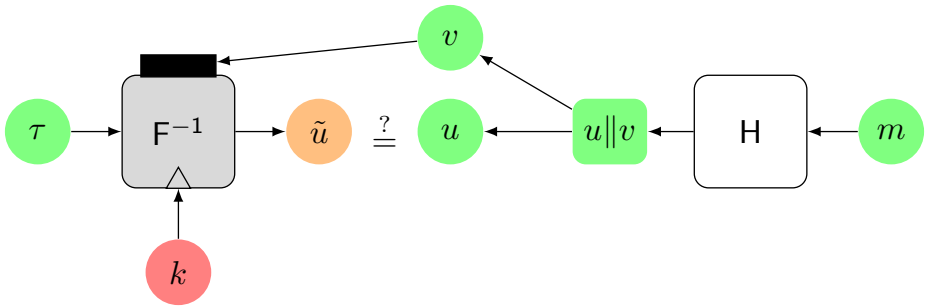


Not recomputing τ in Vrfy. **Idea:** using F^{-1}.

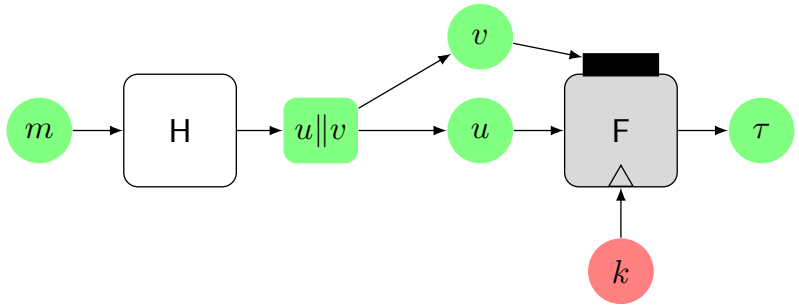
HTBC - F sUL + H Random Oracle [BGPPS19]



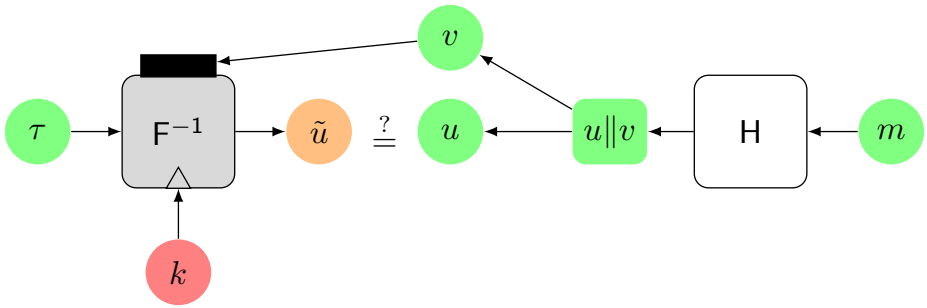
Not recomputing τ in Vrfy. **Idea:** using F^{-1}.



HTBC - F sUL + H Random Oracle [BGPPS19]

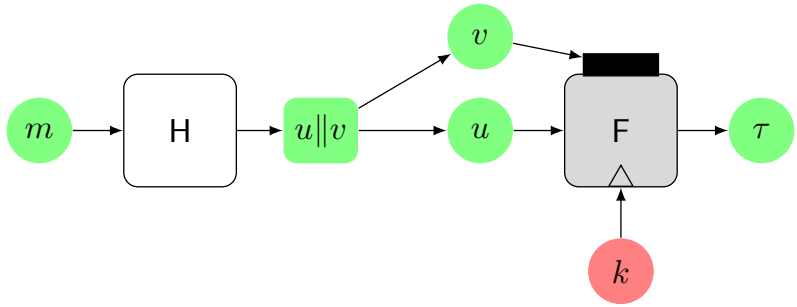


Not recomputing τ in Vrfy. **Idea:** using F^{-1}.

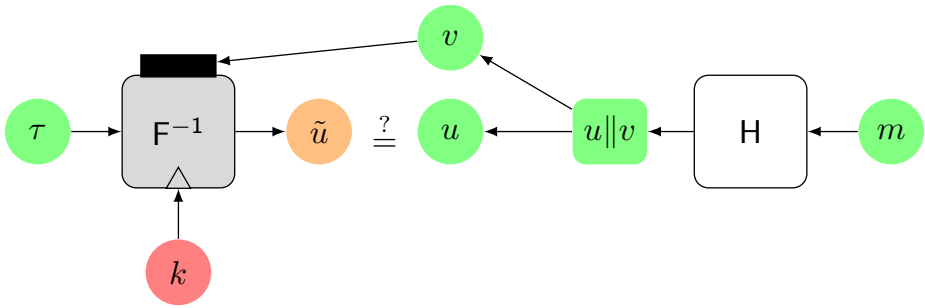


[BGPPS19] If F is sUL, problematic interactions between F and H.

HTBC - F sUL + H Random Oracle [BGPPS19]



Not recomputing τ in Vrfy. **Idea:** using F^{-1}.



[BGPPS19] If F is sUL, problematic interactions between F and H.

- They used the random oracle to bound the probability that the adversary computes $u\|v = \mathsf{H}(m)$ and then obtains $u = \mathsf{F}_k^{-1,v}(\tau)$.

Summary

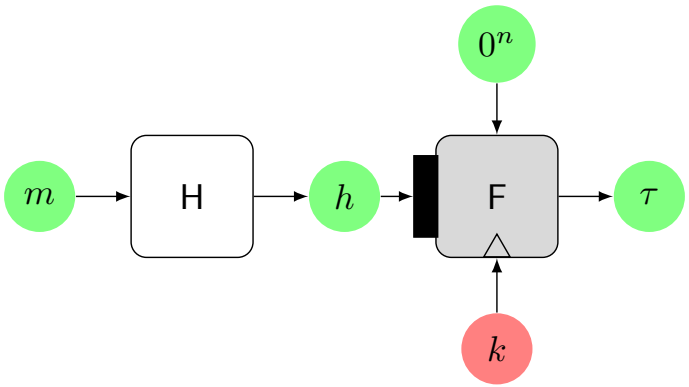
1. Background
   1. Unforgeability for MAC in presence of leakage
   2. Modeling leakage - unbounded leakage model
   3. sUL - Security definition for TBC in presence of leakage
   4. HTBC - [BGPPS19]
2. LR–MAC1
3. LR–MAC2
4. HTBC

LR-MAC1

Problem: Preventing bad interactions between H and F with leakage.

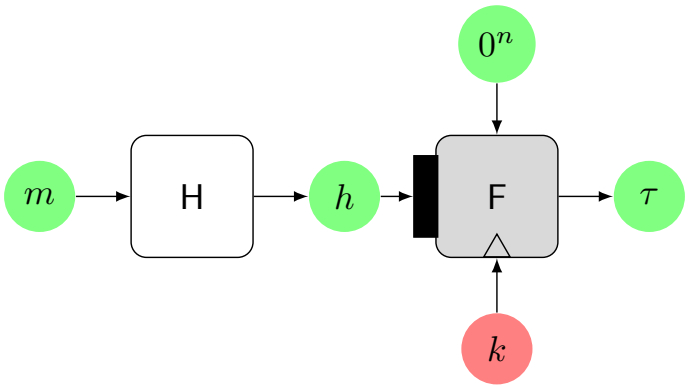
LR-MAC1

Problem: Preventing bad interactions between H and F with leakage.



LR-MAC1

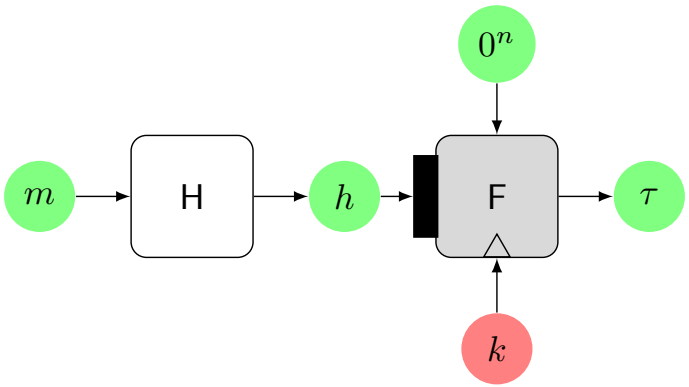
Problem: Preventing bad interactions between H and F with leakage.



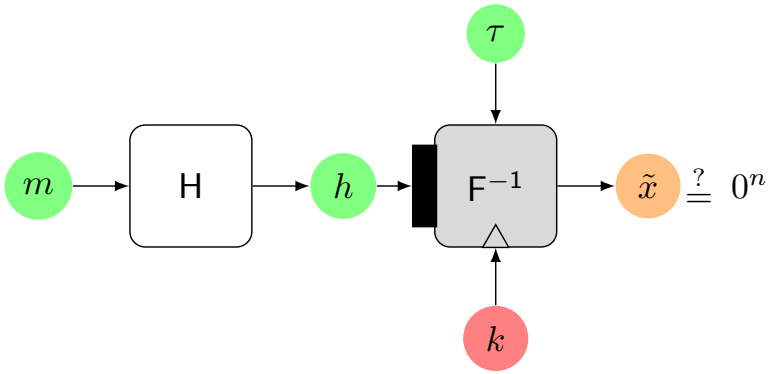
h only as a tweak + fixed input for F

LR-MAC1

Problem: Preventing bad interactions between H and F with leakage.

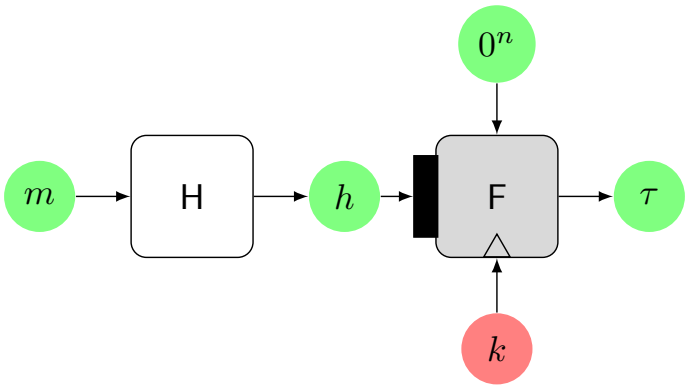


h only as a tweak + fixed input for F

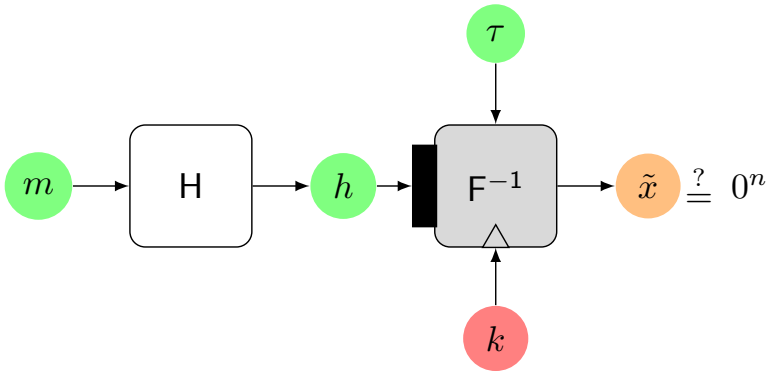


LR-MAC1

Problem: Preventing bad interactions between H and F with leakage.



h only as a tweak + fixed input for F



$\tilde{x}$ (if $\neq 0^n$) is useless

LR–MAC1 - suf–L2-security

LR–MAC1 is (q_M, q_V, ϵ)-suf–L2 secure with

$$\epsilon \leq \epsilon_{\mathsf{CR}} + (q_V + 1)\epsilon_{\mathsf{sUL}}, \text{ with}$$

LR–MAC1 - suf–L2-security

LR–MAC1 is (q_M, q_V, ϵ)-suf–L2 secure with

$$\epsilon \leq \epsilon_{\mathsf{CR}} + (q_V + 1)\epsilon_{\mathsf{sUL}}, \text{ with}$$

- F be a $(q_{\mathsf{L}}, q_M, q_V, \epsilon_{\mathsf{sUL}})$-strongly unpredictable with leakage TBC.
- H be a ϵ_{CR}-collision resistant hash function,

LR–MAC1 - suf–L2-security

LR–MAC1 is (q_M, q_V, ϵ)-suf–L2 secure with

$$\epsilon \leq \epsilon_{\mathsf{CR}} + (q_V + 1)\epsilon_{\mathsf{sUL}}, \text{ with}$$

- F be a $(q_{\mathsf{L}}, q_M, q_V, \epsilon_{\mathsf{sUL}})$-strongly unpredictable with leakage TBC.
- H be a ϵ_{CR}-collision resistant hash function,

Concrete security:

- For a good H $\epsilon_{\mathsf{CR}} \approx \frac{q^2}{2^{|\mathcal{TW}|}}$
- ϵ_{sUL} depends on the side-channel security of the implementation of F

LR–MAC1 - suf–L2-security

LR–MAC1 is (q_M, q_V, ϵ)-suf–L2 secure with

$$\epsilon \leq \epsilon_{\mathsf{CR}} + (q_V + 1)\epsilon_{\mathsf{sUL}}, \text{ with}$$

- F be a $(q_{\mathsf{L}}, q_M, q_V, \epsilon_{\mathsf{sUL}})$-strongly unpredictable with leakage TBC.
- H be a ϵ_{CR}-collision resistant hash function,

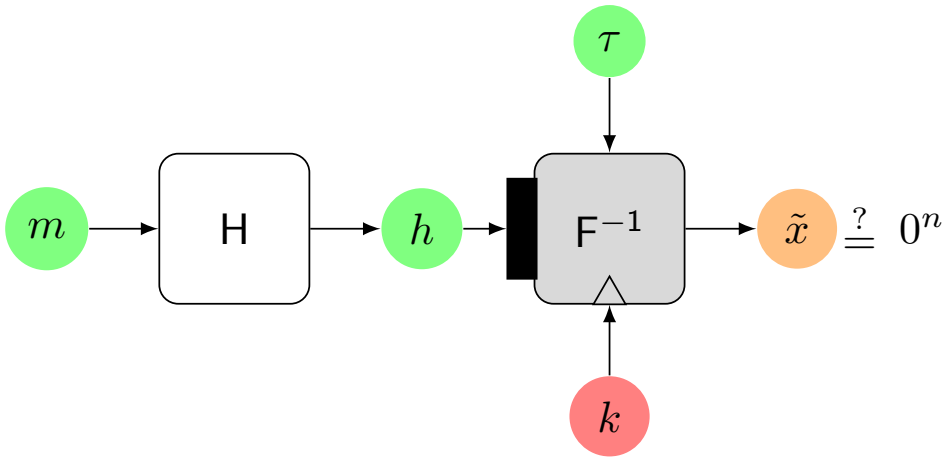
Concrete security:

- For a good H $\epsilon_{\mathsf{CR}} \approx \frac{q^2}{2^{|\mathcal{TW}|}}$
- ϵ_{sUL} depends on the side-channel security of the implementation of F

Proposal implementation: F = Deoxys–384 and H is SHA256.

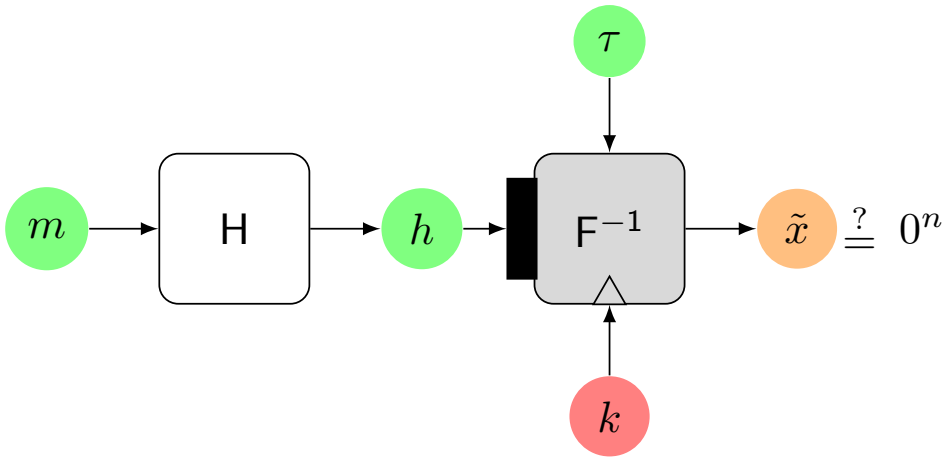
- Deoxys [128-bit block and 256-bit tweak]

Construction 1 - Idea of the proof



Consider a fresh and valid verification query (m^i, τ^i). Let $h^i = \mathsf{H}_s(m^i)$.

Construction 1 - Idea of the proof

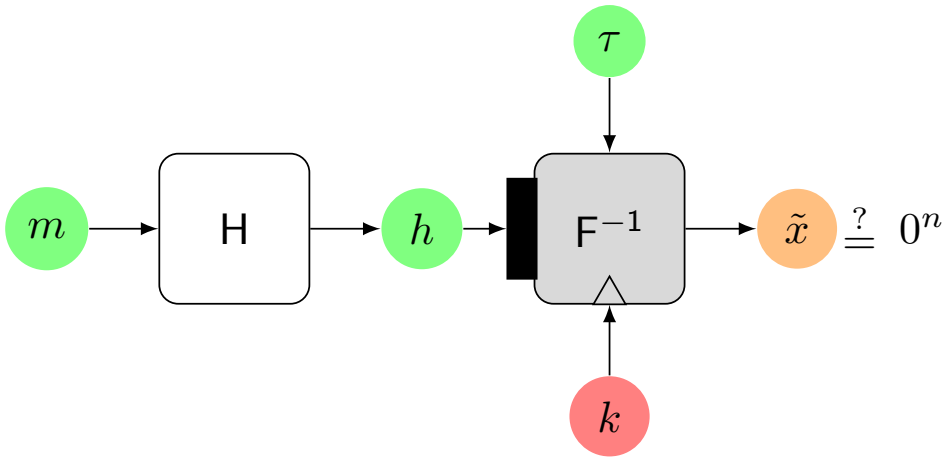


Consider a fresh and valid verification query (m^i, τ^i). Let $h^i = \mathsf{H}_s(m^i)$.

F_1: There exists a tag-generation query m^j with s.t. $\mathsf{H}_s(m^j) = h^j = h^i$.

F_2: h^i has never been obtained as hash of a message used in a tag-generation query.

Construction 1 - Idea of the proof



Consider a fresh and valid verification query (m^i, τ^i). Let $h^i = \mathsf{H}_s(m^i)$.

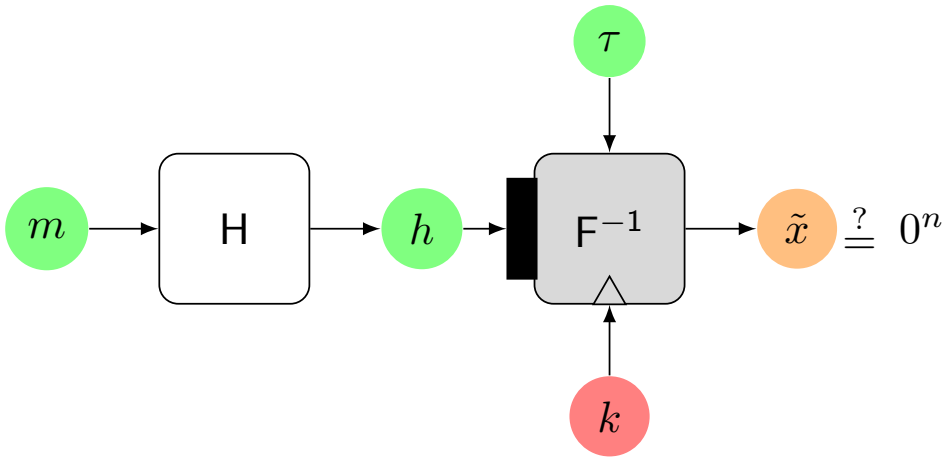
F_1: There exists a tag-generation query m^j with s.t. $\mathsf{H}_s(m^j) = h^j = h^i$.

F_2: h^i has never been obtained as hash of a message used in a tag-generation query.

Now, we obtain easily the security result

F_1: We have found a collision for the hash function.

Construction 1 - Idea of the proof



Consider a fresh and valid verification query (m^i, τ^i). Let $h^i = \mathsf{H}_s(m^i)$.

F_1: There exists a tag-generation query m^j with s.t. $\mathsf{H}_s(m^j) = h^j = h^i$.

F_2: h^i has never been obtained as hash of a message used in a tag-generation query.

Now, we obtain easily the security result

F_1: We have found a collision for the hash function.

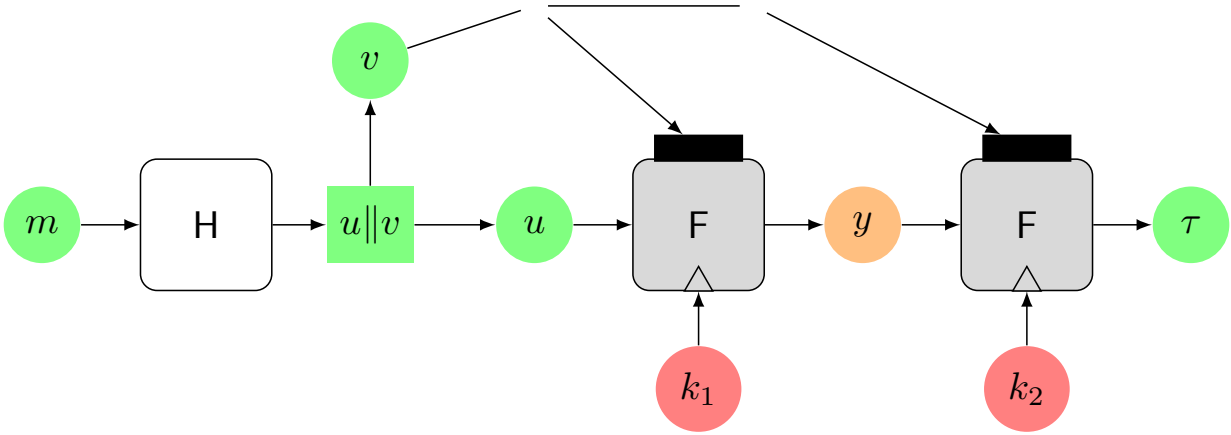
F_2: $(0^n, h^i, \tau^i)$ is a valid triple for F_k which has never been computed before. Thus, we are able to predict the output of F_k.

LR-MAC2 - Achieving BBB security without large tweaks.

Problem: If we have only TBC with n-bit tweaks, achieve BBB security.

LR-MAC2 - Achieving BBB security without large tweaks.

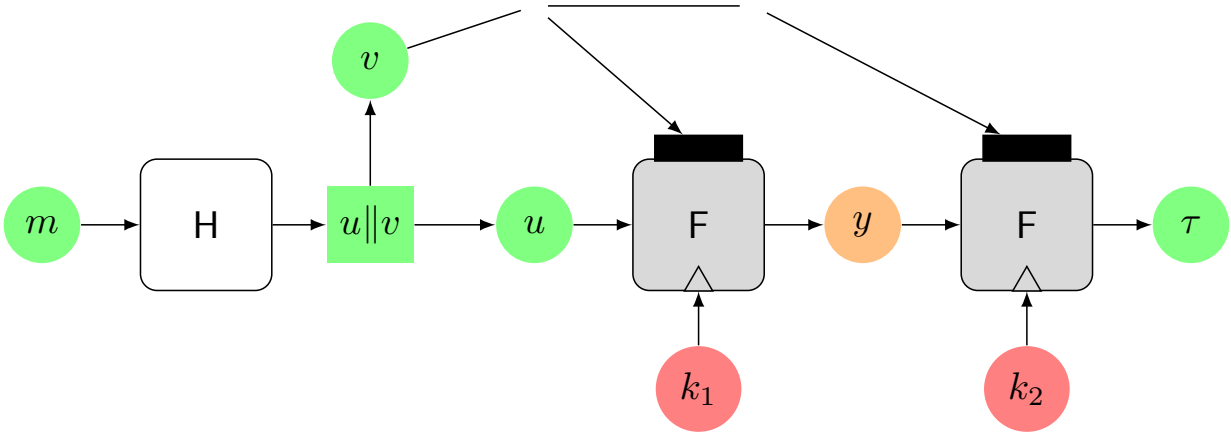
Problem: If we have only TBC with n-bit tweaks, achieve BBB security.



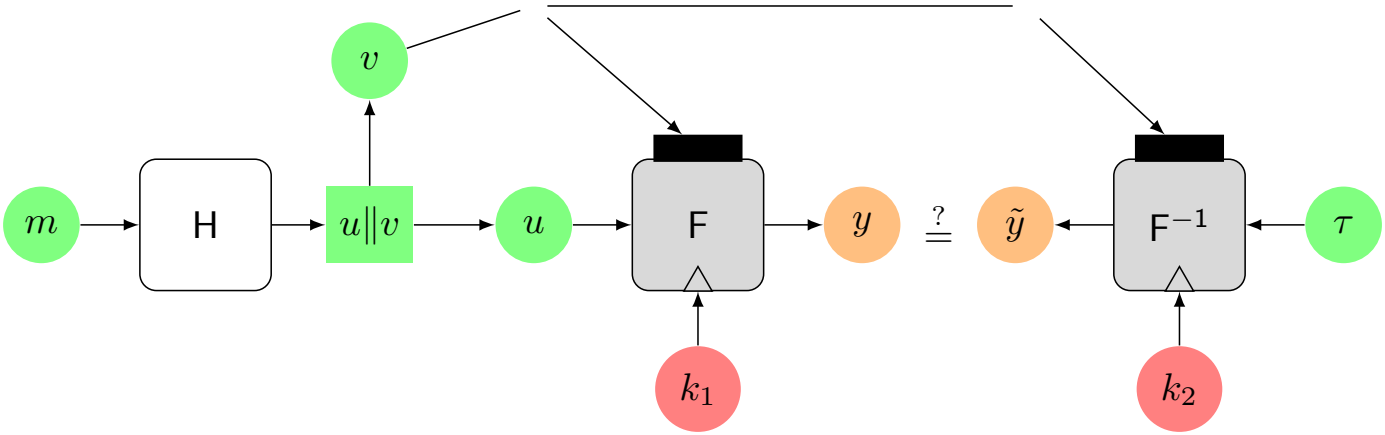
Same tweak for the 2 F calls.

LR-MAC2 - Achieving BBB security without large tweaks.

Problem: If we have only TBC with n-bit tweaks, achieve BBB security.

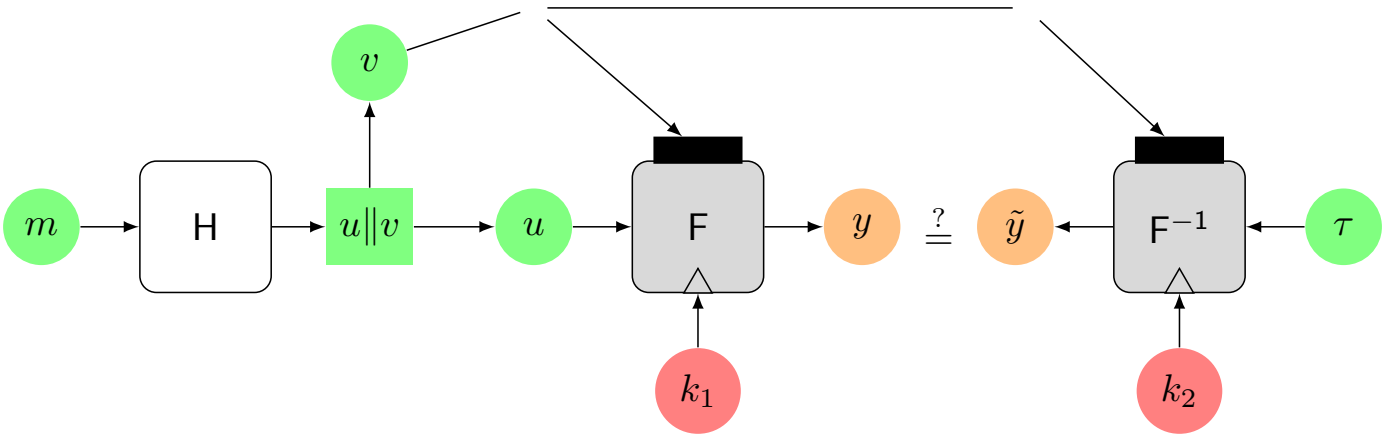


Same tweak for the 2 F calls.



Verify if $y = \tilde{y}$. Use F^{-1}.

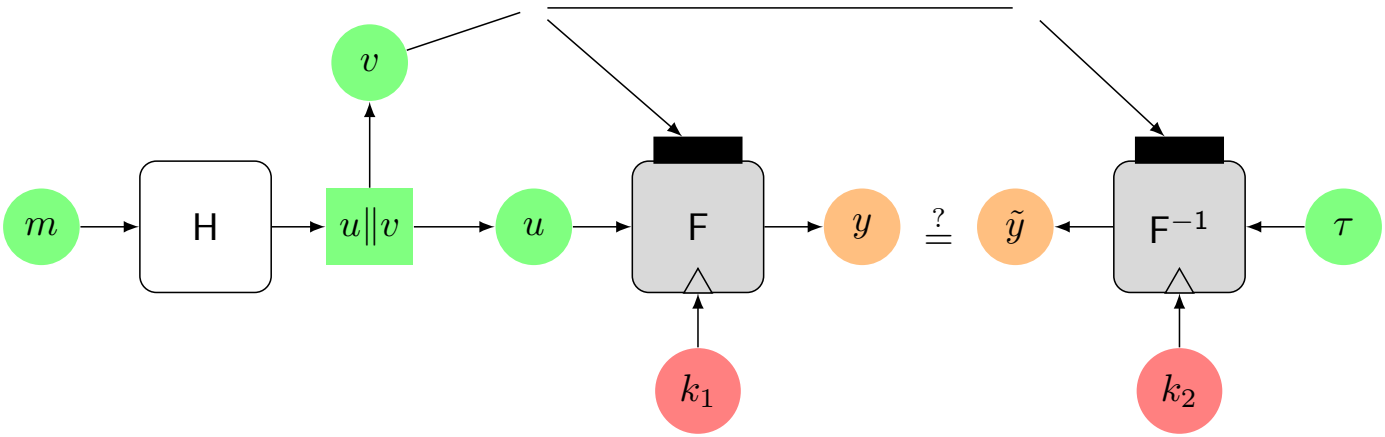
LR–MAC2 - suf–L2-security



LR–MAC2 is (q_M, q_V, ϵ)-suf–L2 secure with

$$\epsilon \leq \epsilon_{\mathsf{CR}(2n)} + \epsilon_{\mu-\mathsf{CR}(n)} + 2\mu q_V \epsilon_{\mathsf{sUL}}, \text{ with}$$

LR–MAC2 - suf–L2-security

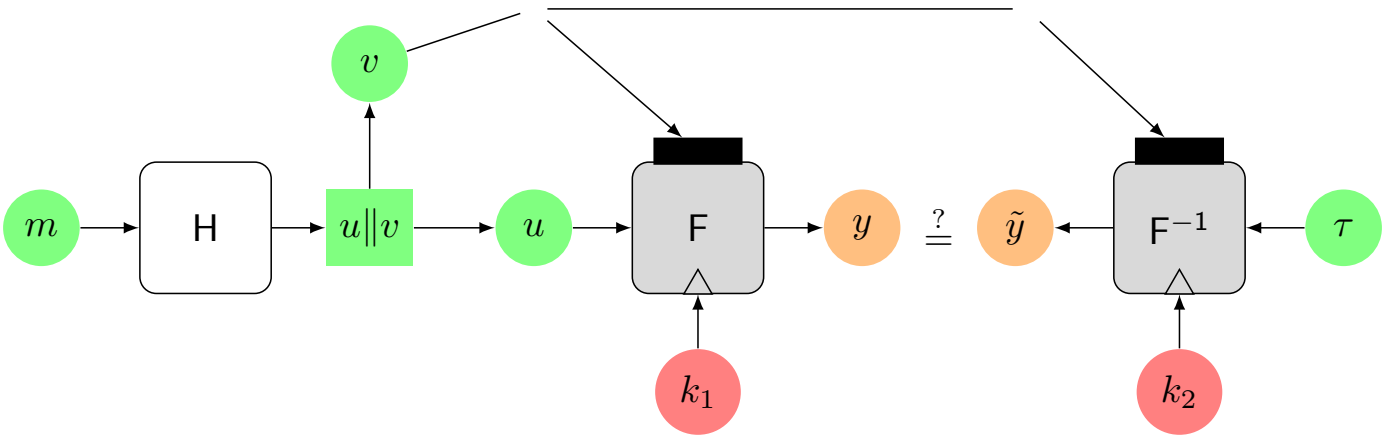


LR–MAC2 is (q_M, q_V, ϵ)-suf–L2 secure with

$$\epsilon \leq \epsilon_{\mathsf{CR}(2n)} + \epsilon_{\mu-\mathsf{CR}(n)} + 2\mu q_V \epsilon_{\mathsf{sUL}}, \text{ with}$$

- F be a $(q_{\mathsf{L}}, q_M, q_V, \epsilon_{\mathsf{sUL}})$-strongly unpredictable with leakage TBC.
- H be a ϵ_{CR}-collision resistant hash function,

LR–MAC2 - suf–L2-security

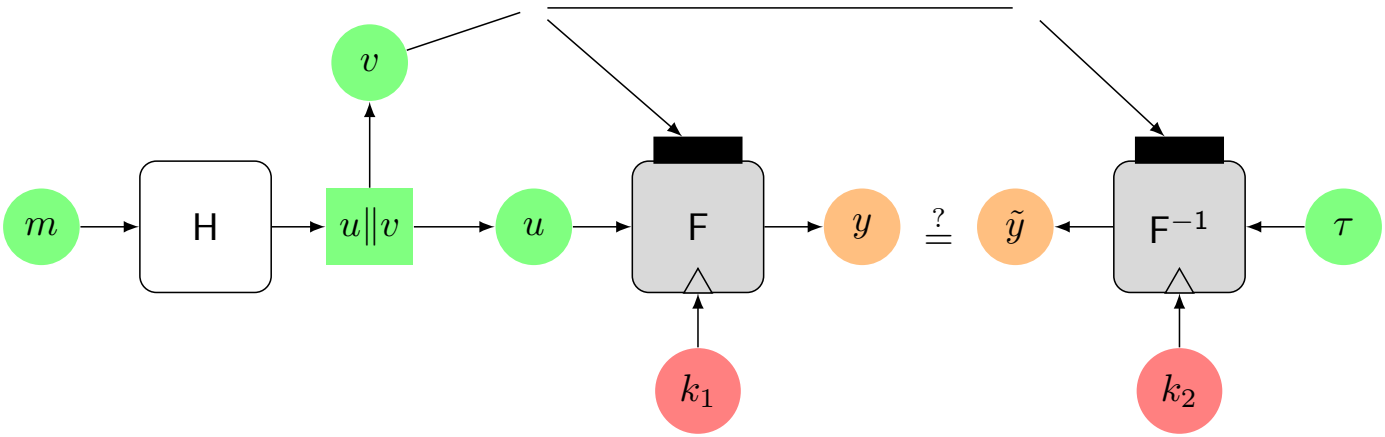


LR–MAC2 is (q_M, q_V, ϵ)-suf–L2 secure with

$$\epsilon \leq \epsilon_{\mathsf{CR}(2n)} + \epsilon_{\mu-\mathsf{CR}(n)} + 2\mu q_V \epsilon_{\mathsf{sUL}}, \text{ with}$$

- F be a $(q_{\mathsf{L}}, q_M, q_V, \epsilon_{\mathsf{sUL}})$-strongly unpredictable with leakage TBC.
- H be a ϵ_{CR}-collision resistant hash function,
- H be a $\epsilon_{\mu-\mathsf{CR}(n)}$-lower half-collision resistant,

LR–MAC2 - suf–L2-security



LR–MAC2 is (q_M, q_V, ϵ)-suf–L2 secure with

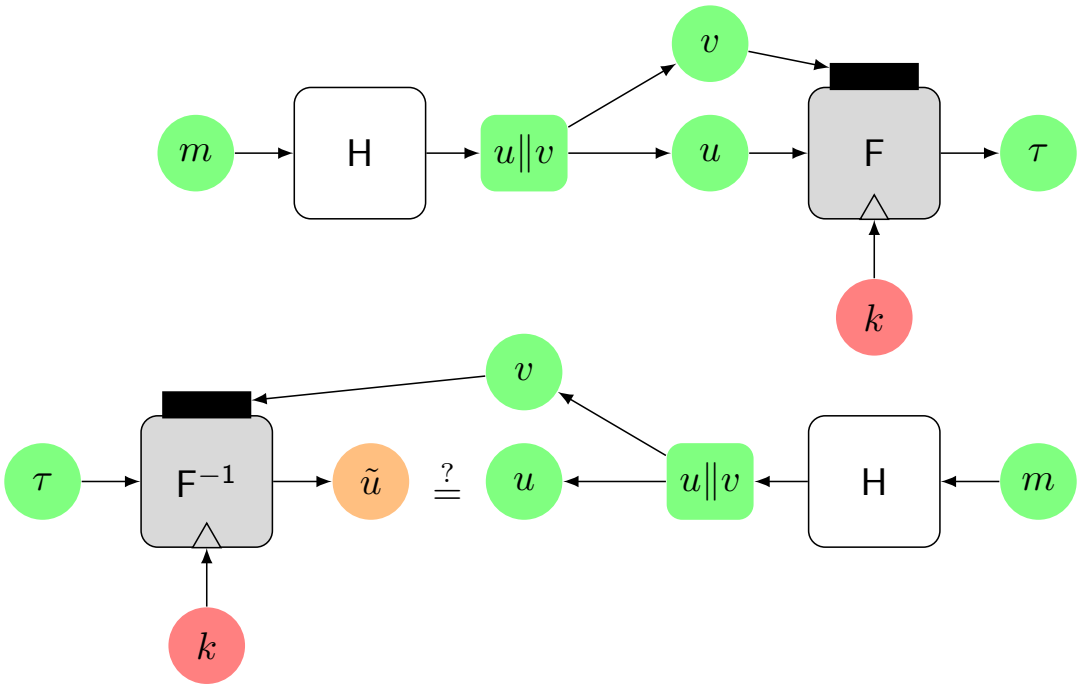
$$\epsilon \leq \epsilon_{\mathsf{CR}(2n)} + \epsilon_{\mu-\mathsf{CR}(n)} + 2\mu q_V \epsilon_{\mathsf{sUL}}, \text{ with}$$

- F be a $(q_{\mathsf{L}}, q_M, q_V, \epsilon_{\mathsf{sUL}})$-strongly unpredictable with leakage TBC.
- H be a ϵ_{CR}-collision resistant hash function,
- H be a $\epsilon_{\mu-\mathsf{CR}(n)}$-lower half-collision resistant,

Lower-half-collision resistance: Find $m_1, ..., m_\mu$ s.t. $\mathsf{H}_s(m_i) = ...\|h_2$.

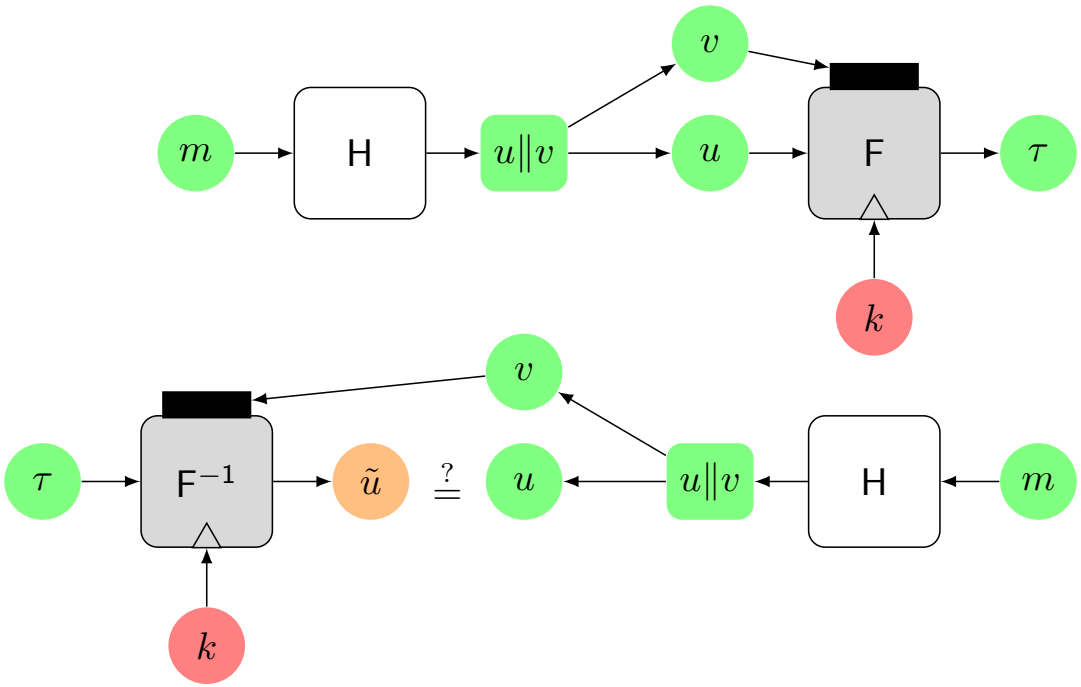
HTBC - New security results

Strong hypothesis for H.



HTBC - New security results

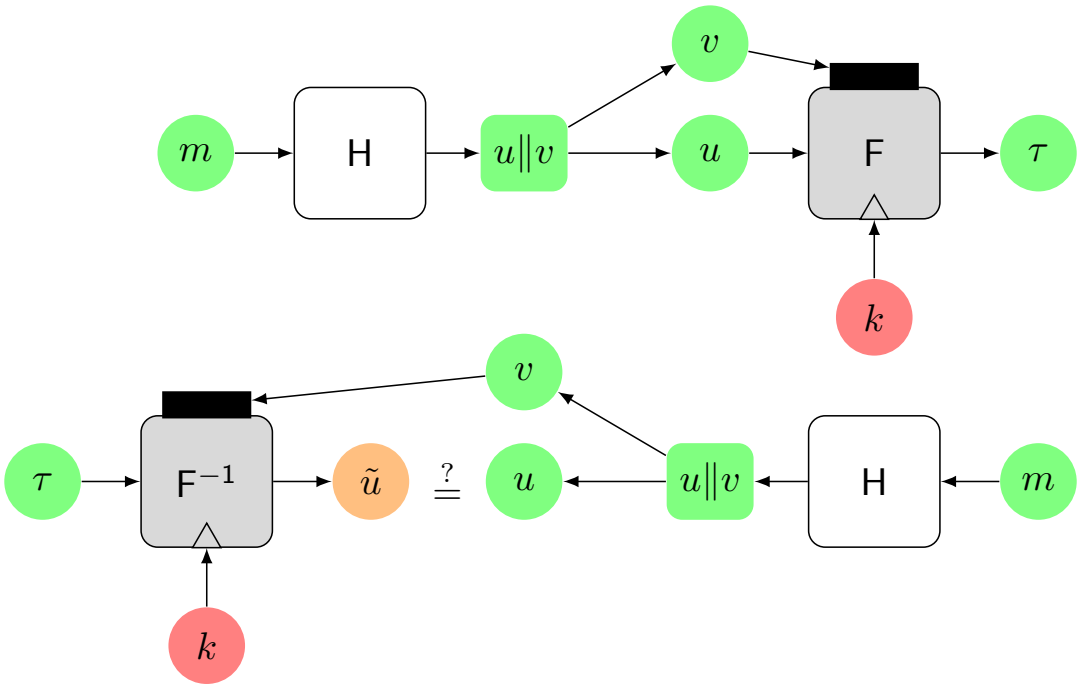
Strong hypothesis for H.



There is a set of weak points in the codomain s.t. their pre-image is easily computable. An hash function is *weak-set computable* if there exists a PPT adversary that outputs the previous set.

HTBC - New security results

Strong hypothesis for H.

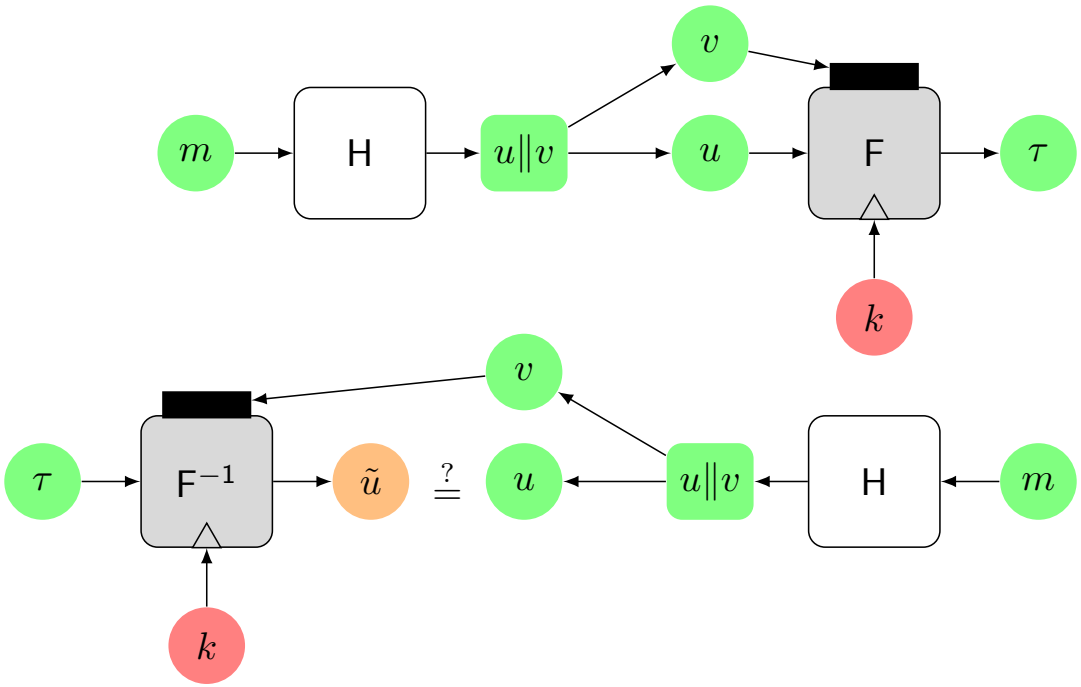


There is a set of weak points in the codomain s.t. their pre-image is easily computable. An hash function is *weak-set computable* if there exists a PPT adversary that outputs the previous set.

- Not ideal, but very strong

HTBC - New security results

Strong hypothesis for H.



There is a set of weak points in the codomain s.t. their pre-image is easily computable. An hash function is *weak-set computable* if there exists a PPT adversary that outputs the previous set.

- Not ideal, but very strong
- HTBC very efficient

Conclusion

- **New:** 3 leakage resilient MACs. They range from pragmatic to theoretically more involved

Conclusion

- **New:** 3 leakage resilient MACs. They range from pragmatic to theoretically more involved
- Security based on
  - minimum physical assumptions (sUL)
  - black-box (non idealized) assumptions for H

Conclusion

- **New:** 3 leakage resilient MACs. They range from pragmatic to theoretically more involved
- Security based on
  - minimum physical assumptions (sUL)
  - black-box (non idealized) assumptions for H
- These MACs have
  - security proofs and bounds in the standard model
  - concrete requirements for implementers

Conclusion

- **New:** 3 leakage resilient MACs. They range from pragmatic to theoretically more involved
- Security based on
  - minimum physical assumptions (sUL)
  - black-box (non idealized) assumptions for H
- These MACs have
  - security proofs and bounds in the standard model
  - concrete requirements for implementers
- Future works
  - Instantiation + comparing performances
  - Extension to authenticated encryption (AE)

Thank you

- francesco.berti@tu-darmstadt.de
- chun.guo.sc@gmail.com
- thomas.peters@uclouvain.be
- fstandae@uclouvain.be