Advances on CIML2

Francesco Berti

Objectives

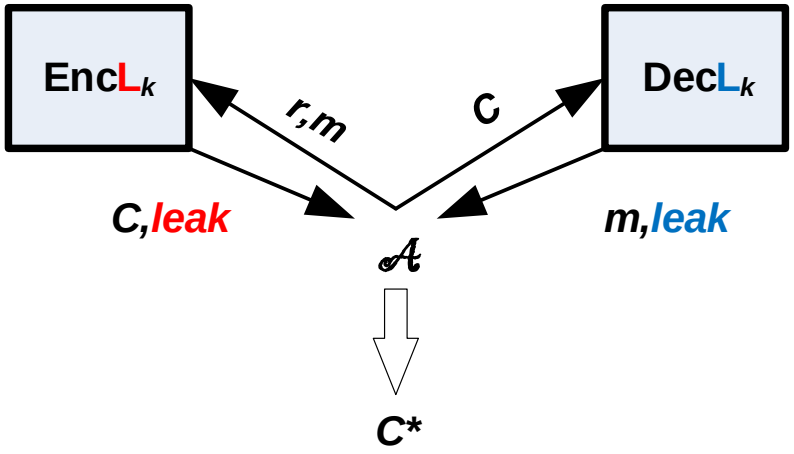
Limitation of the leak free implementation model

- No well defined security game
- Countermeasures aims to hide the key
- Hard to have security up to 2^{128} measurements
- Expensive (even 1000x)

- Replace the leak free assumption keeping CIML2
  - New hypothesis: strong unpredictability with leakage
- A CIML2-secure AE scheme using *only once* the leak free implemented TBC

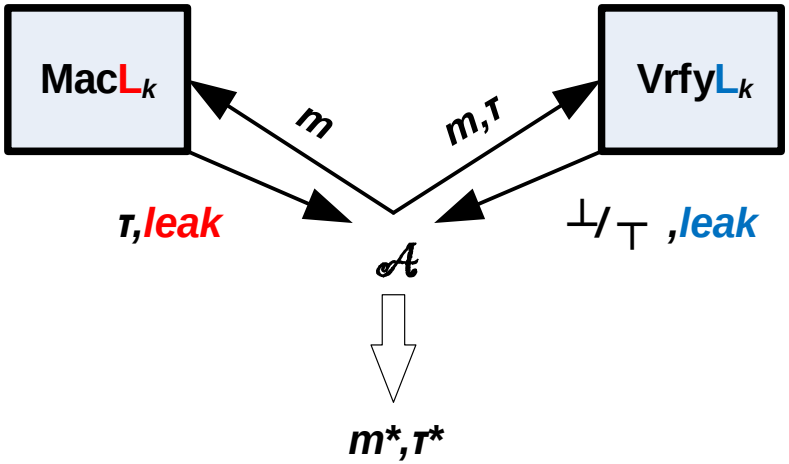
CIML2 & eufL2

- CIML2 for AE



- eufL2 for MAC

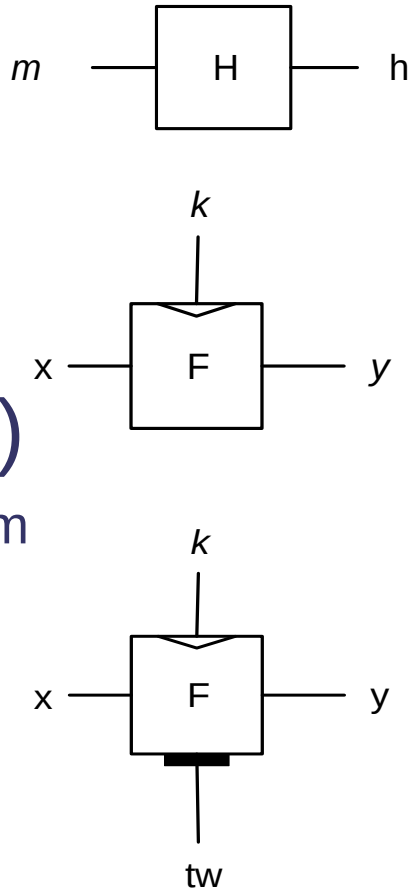
(existentially unforgeability with leakage in both tag-generation and verification)



Components

- Hash functions
  - Collision resistance:
    hard find $m, m's.t.\ \mathrm{H}(m) = \mathrm{H}(m')$
  - Range-oriented pre-image resistance:
    given a random y, hard find $m\ s.t.\ \mathrm{H}(m) = h$
- (Tweakable) block cipher ((T)BC)
  - Pseudorandom: hard to distinguish its output from random ones
    - (strong) even having access to its inverse

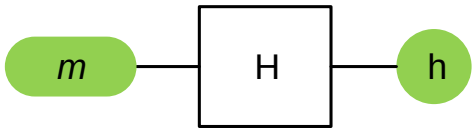
block = n-bit string

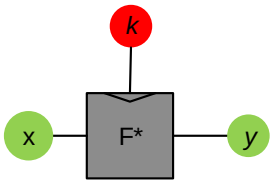


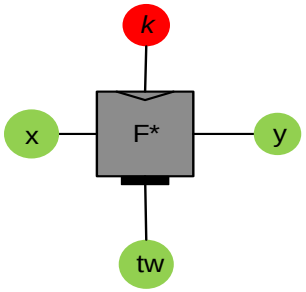
Components with leakage

Leveled implementation:

- Hash function
  - No protection
- BC/TBC:
  - Well protected (leak free):
    - key perfectly hided, outputs and inputs known and random



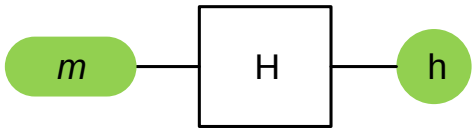


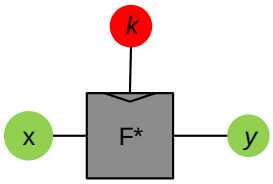


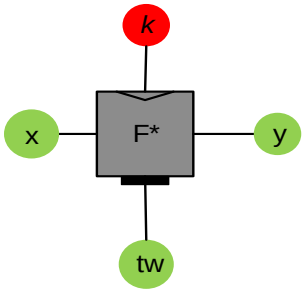
Components with leakage

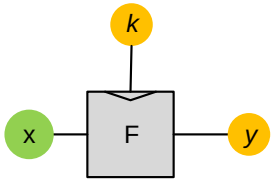
Leveled implementation:

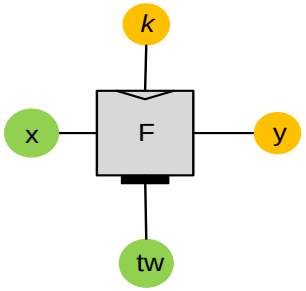
- Hash function
  - No protection
- BC/TBC:
  - Well protected (leak free):
    - key perfectly hided, outputs and inputs known and random
  - Weakly protected:
    - No protection for CIML2, some for confidentiality







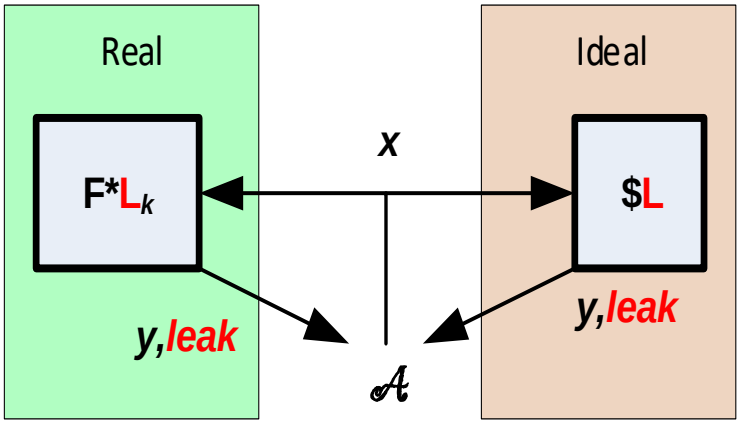




Problem of leak-free

The leak free model has many advantages, but:

- No well-defined security game
  - Hard to simulate ideal leakage



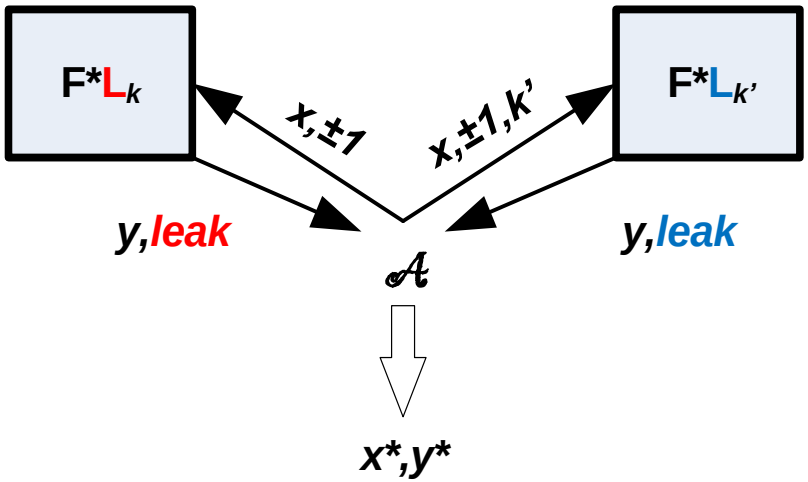
Problem of leak-free

The leak free model has many advantages, but:

- No well-defined security game
  - Hard to simulate ideal leakage
- Leak free implementations hides the key
- Hard to protect up to 2^{128} measurements
- Expensive (even 1000x)

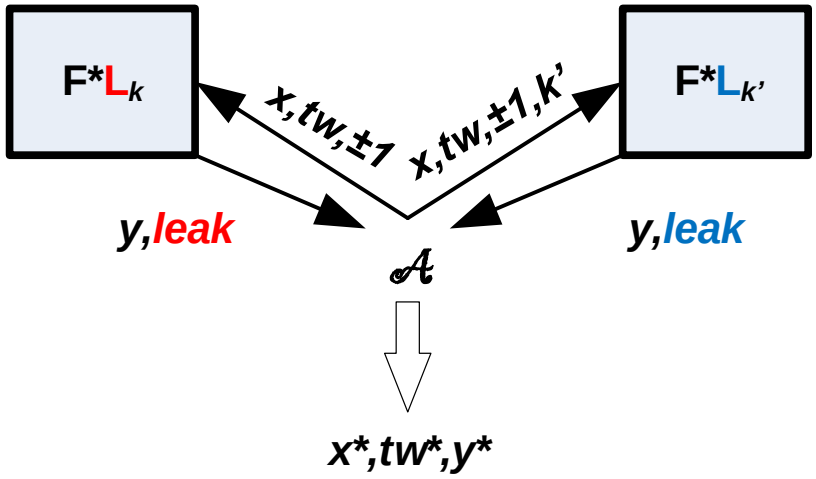
Strong unpredictability with leakage

For BC



Hard provide fresh $(x*, y*)$ s.t. $y^* = \mathrm{F}_k^*(x^*)$.

For TBC



Hard provide fresh $(x*, tw*, y*)$ s.t. $y^* = \mathrm{F}_k^*(x^*, tw^*)$.

Advantages of sUnpL

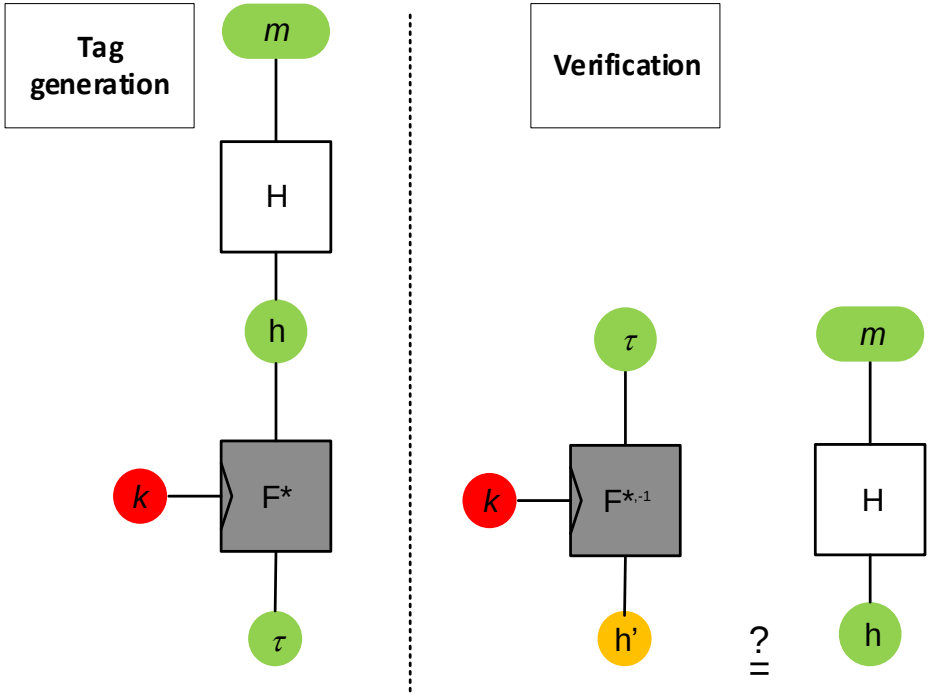
sUnpL:

- Well defined security game
- Reduced round (T)BC may be sUnpL
- Verifiable in laboratories
- Less demanding

Leak free:

- Easier to manipulate
- More versatile
  - Easier to use for confidentiality

Hash-then-BC (EDT)



$$\epsilon_{\mathrm{eufL2}} = \epsilon_{\mathrm{cr}} + (q_{\mathrm{V}} + 1)\epsilon_{\mathrm{sUnpL}} + q_{\mathrm{V}}\epsilon_{\mathrm{pr}}\epsilon_{\mathrm{sUnpL}}2^{n}$$

Tight bound for Hash-then-BC

The term $\epsilon_{\mathrm{pr}}\epsilon_{\mathrm{sUnpL}}2^n$ is tight.

Consider the following hash function H and the following BC:

Hash function

$$\mathrm{H} = \begin{cases} x||0^{\frac{n}{2}} & \text{if } |m| = n \text{ and } m = 0^{\frac{n}{2}}||x \\ f(m)||1 & \text{otherwise} \end{cases}$$

$$\epsilon_{\mathrm{pr}} = 2^{\frac{-n}{2}}$$

sUnpL of the BC

$$\Pr[y = x'] = \begin{cases} 2^{\frac{-n}{2}} & \text{if } y = w||0^{\frac{n}{2}} \\ 0 & \text{otherwise} \end{cases}$$

$$\epsilon_{\mathrm{sUnpL}} = 2^{\frac{-n}{2}}$$

Their composition is insecure

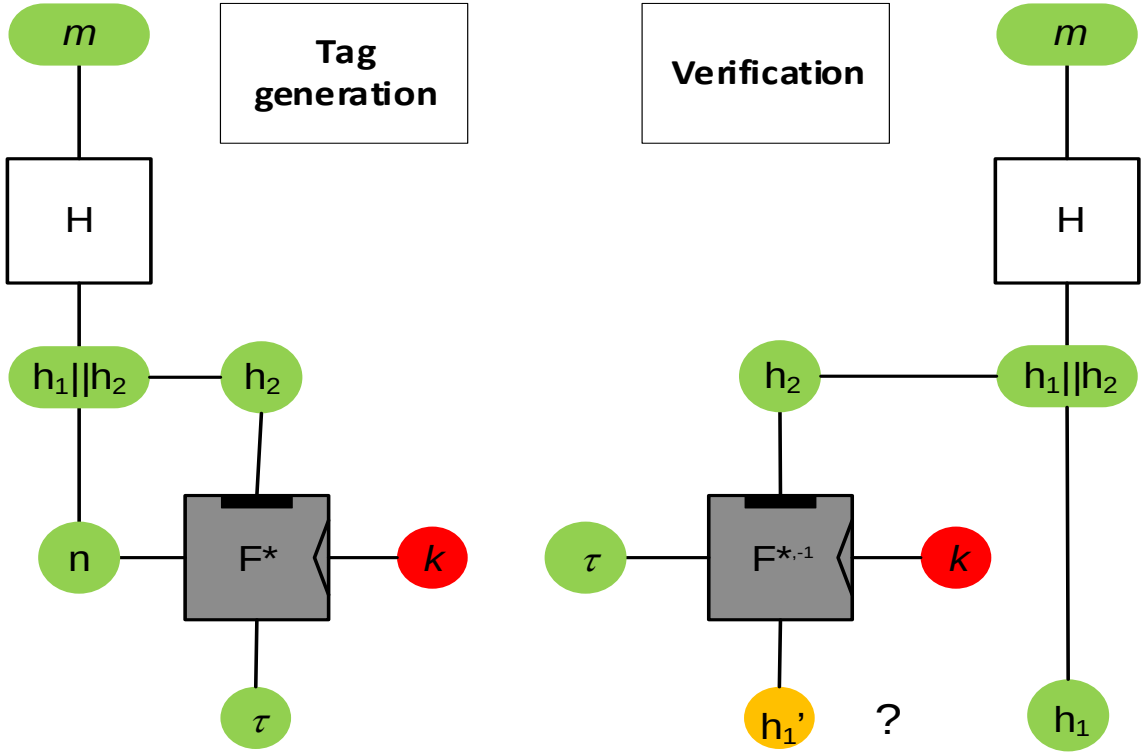
Tight bound for Hash-then-BC

The term $\epsilon_{\mathrm{pr}}\epsilon_{\mathrm{sUnpL}}2^n$ is tight.

Consider the following hash function H and the following BC:

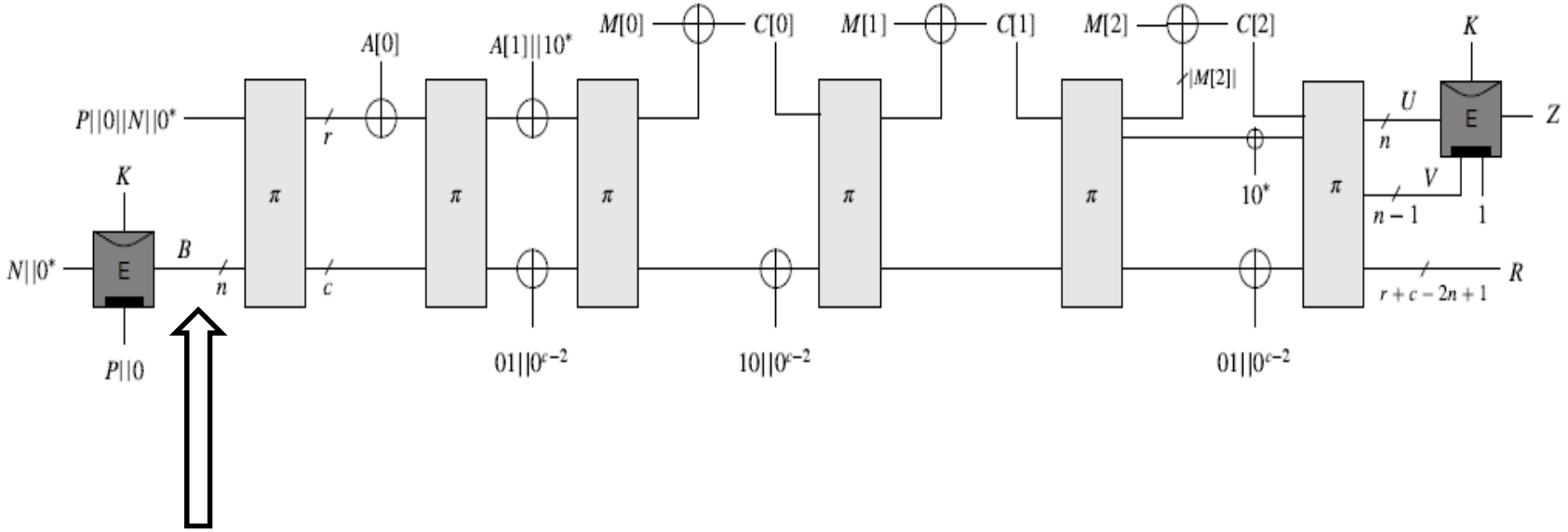
- Problem:
  - Interactions between the leakage of the (T)BC and the range-oriented pre-image resistance of the hash function
  - The attack is clearly artificial
  - Needed:
    - Good theoretical model
    - Good definitions

Hash-then-TBC (TEDT/SPOOK)



$$\epsilon_{\mathrm{eufL2}} = \epsilon_{\mathrm{cr}} + (q_{\mathrm{V}} + 1)\epsilon_{\mathrm{sUnpL}} + q_{\mathrm{V}}\epsilon_{\mathrm{pr}'}\epsilon_{\mathrm{sUnpL}}2^{n}$$

sUnpL for confidentiality



If B unpredictable then $\pi(B)$ unpredictable, thus Spook is CPAL2

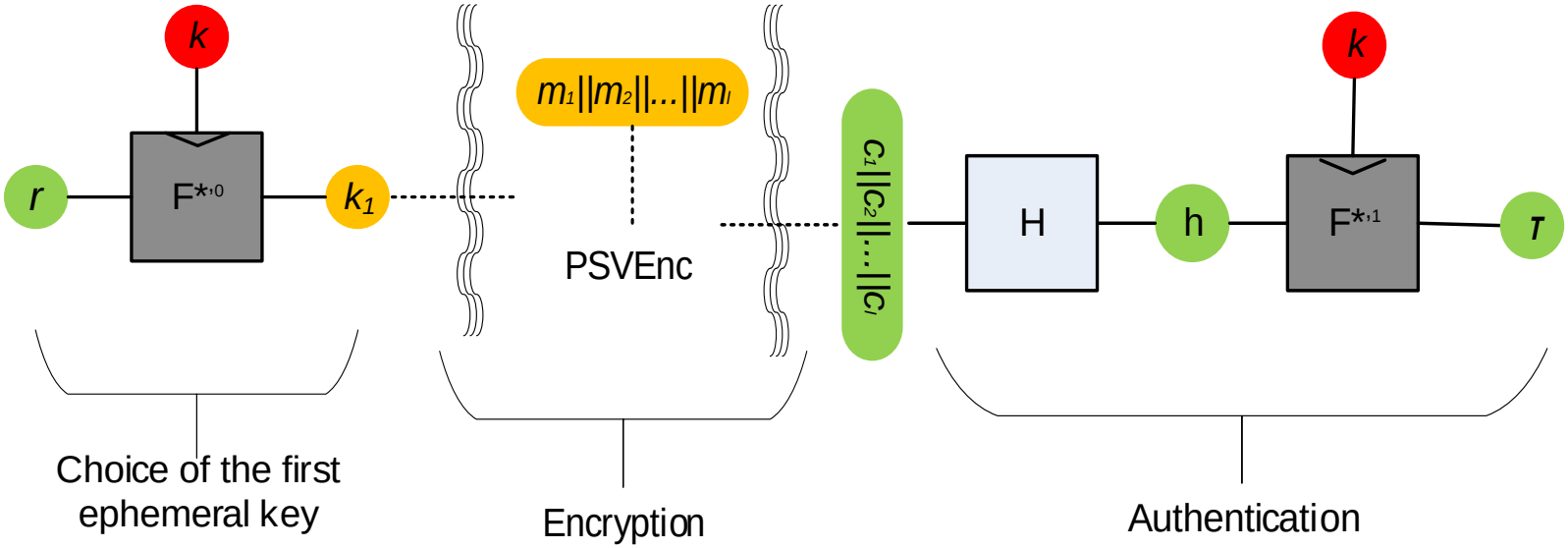
Chun and I observed that we can have confidentiality of Spook if the TBC is sUnpL.

Reducing the leak free calls

Outline:

- Analyzing the structure of EDT
- Reminder of PSV
- Construction of CONCRETE
- Security claims
- Security proof

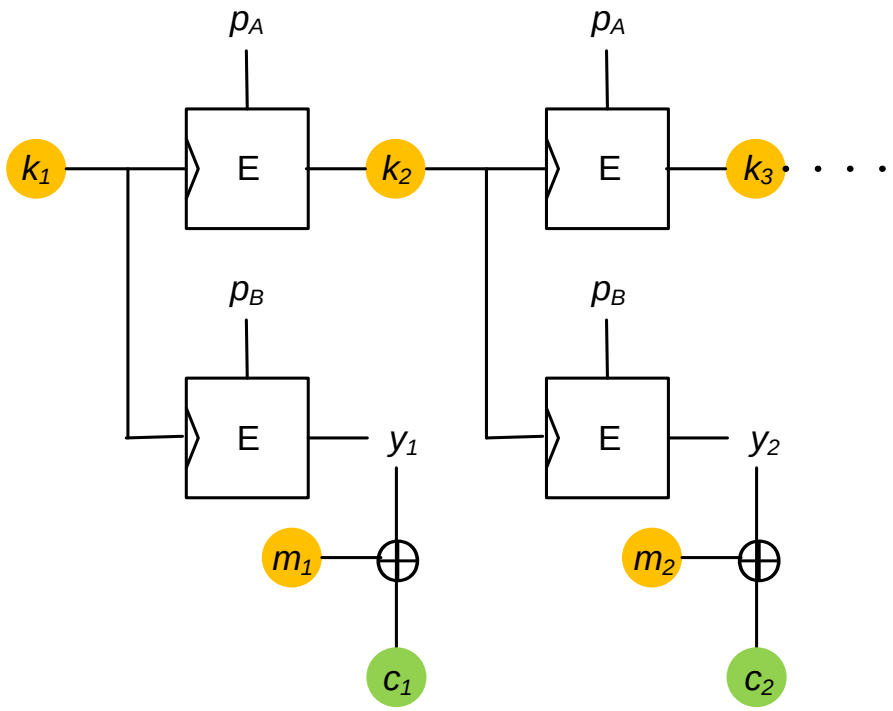
The structure of EDT



For decryption, recompute k_1, then PSVDec, and h. Then compare it with $\mathrm{h}' = \mathrm{F}_k^{*,1,-1}(\tau)$.

PSV [CCS15]

Based on rekeying, CPAL2-secure



For decryption invert the place of m_i and c_i.

Ideas of CONCRETE

COmmit-e**NCR**ypt-s**E**nd-**T**he-k**E**y

- Probabilistic scheme
- k_{1ep} selected uniformly at random
- k_{1ep} sent in the ciphertext
- In the ciphertext there is a commitment c_0 of k_{1ep}
- The encryption of k_{1ep}, c_{l+1} depends on all the ciphertext and the commitment

CONCRETE

COmmit-eNCRypt-sEnd-The-kEy

CONCRETE

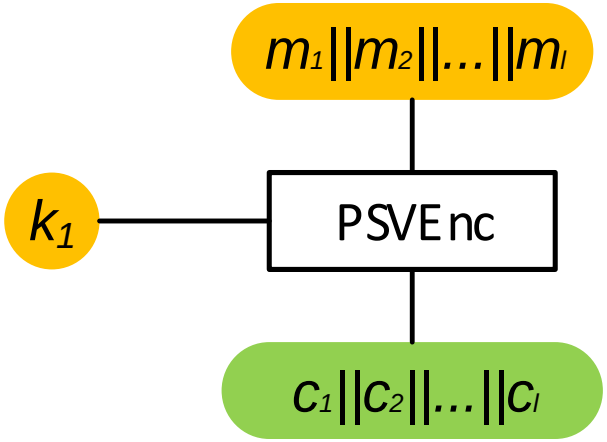
CONCRETE is a probabilistic AE scheme.

How it is built:

CONCRETE

CONCRETE is a probabilistic AE scheme.

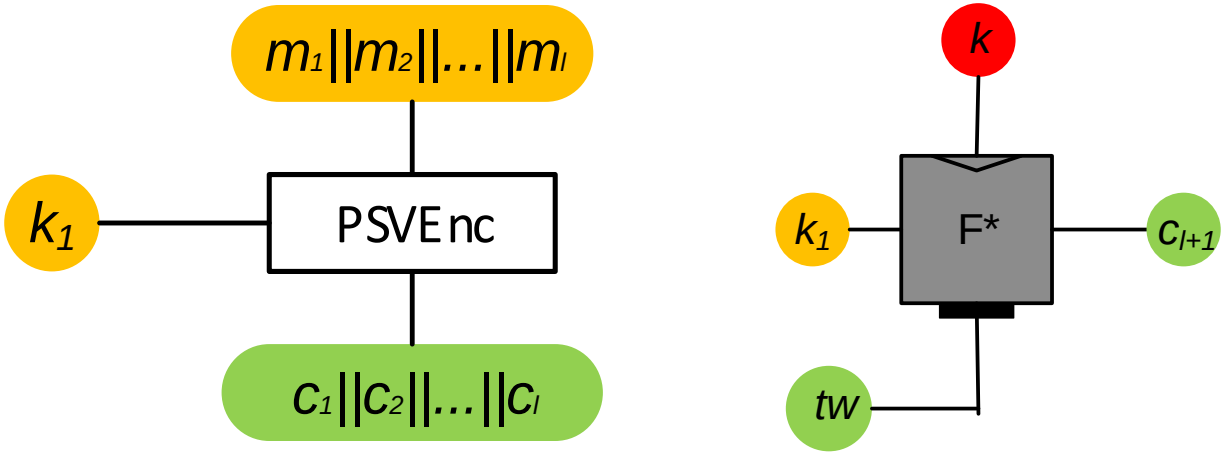
How it is built:



CONCRETE

CONCRETE is a probabilistic AE scheme.

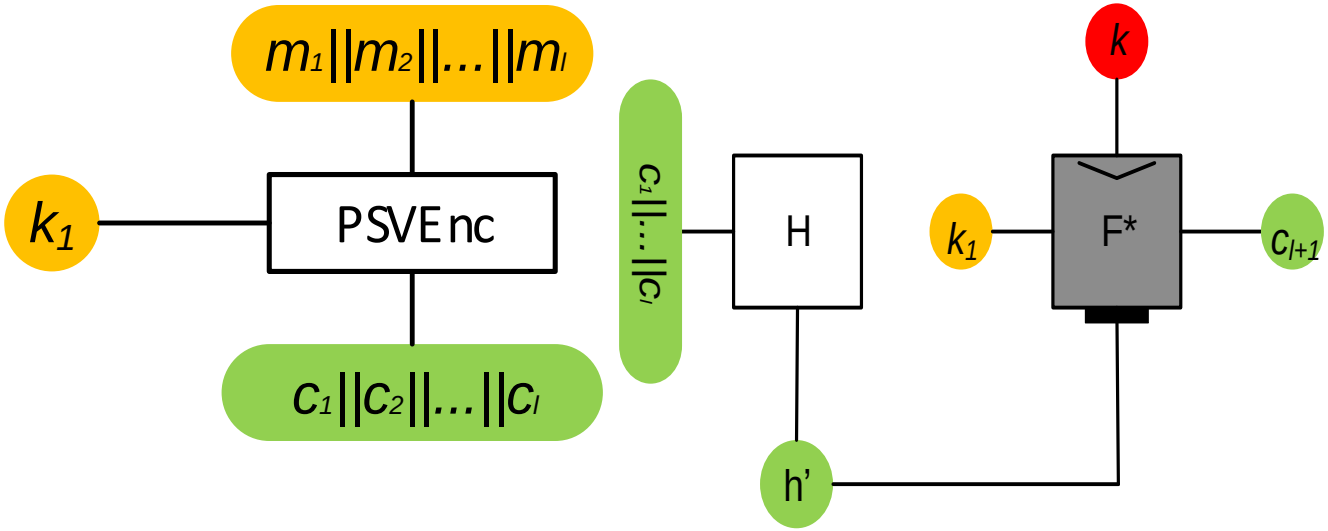
How it is built:



CONCRETE

CONCRETE is a probabilistic AE scheme.

How it is built:

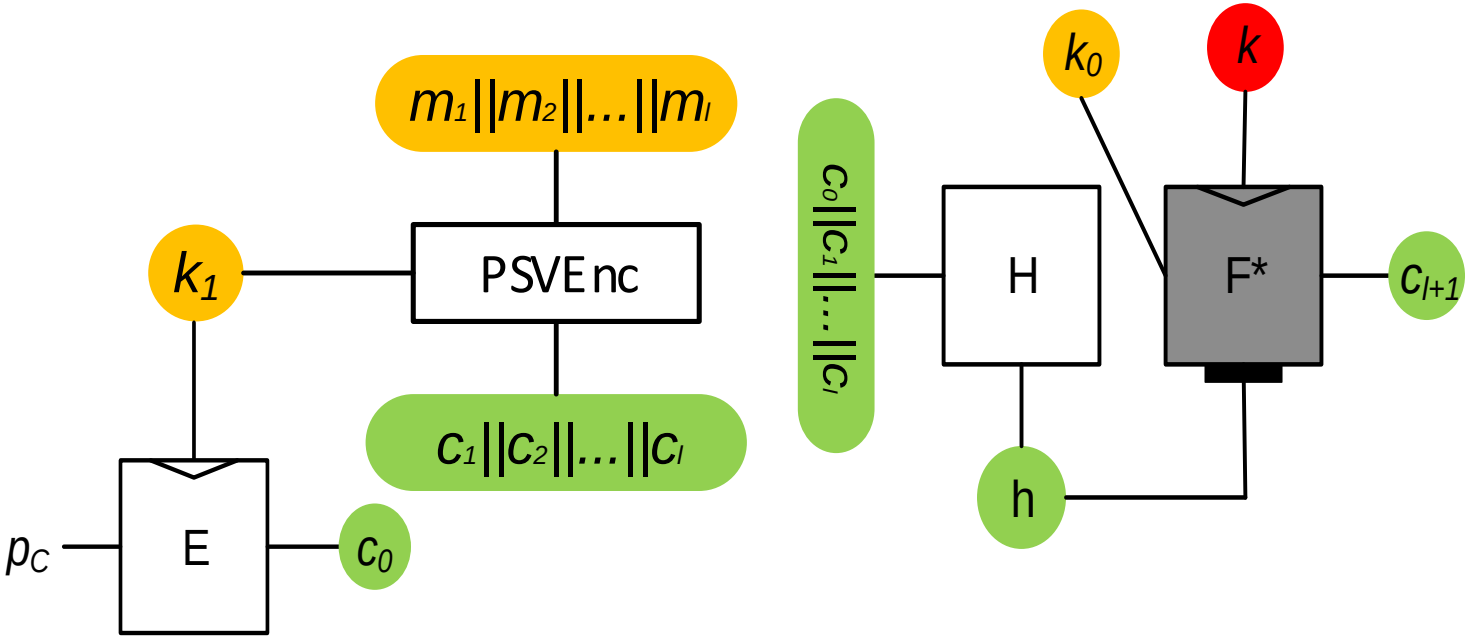


Decryption: retrieve k_1 and use PSVDec

CONCRETE

CONCRETE is a probabilistic AE scheme.

How it is built:

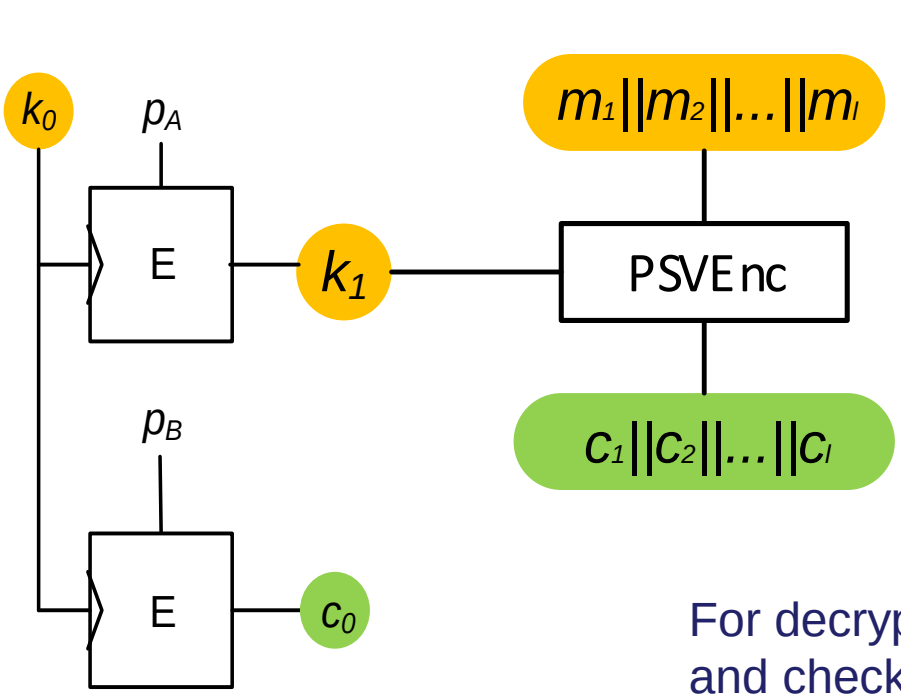


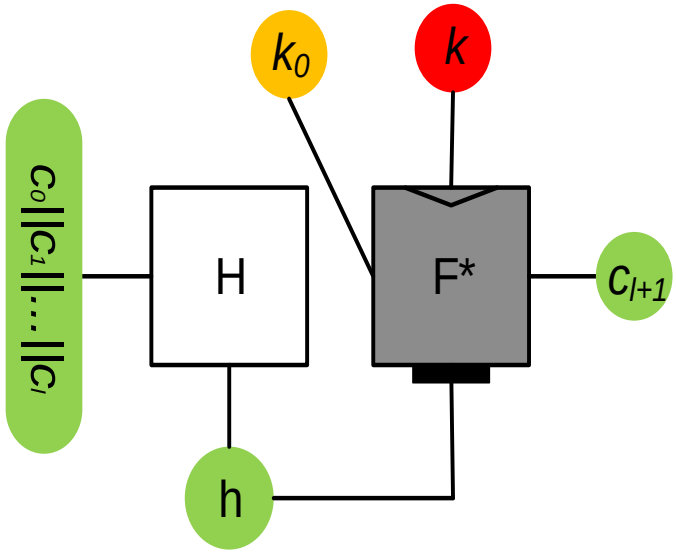
For decryption: retrieve k_1, recompute c_0 and check if correct, then, apply PSVDec.

CONCRETE

CONCRETE is a probabilistic AE scheme.

How it is built:





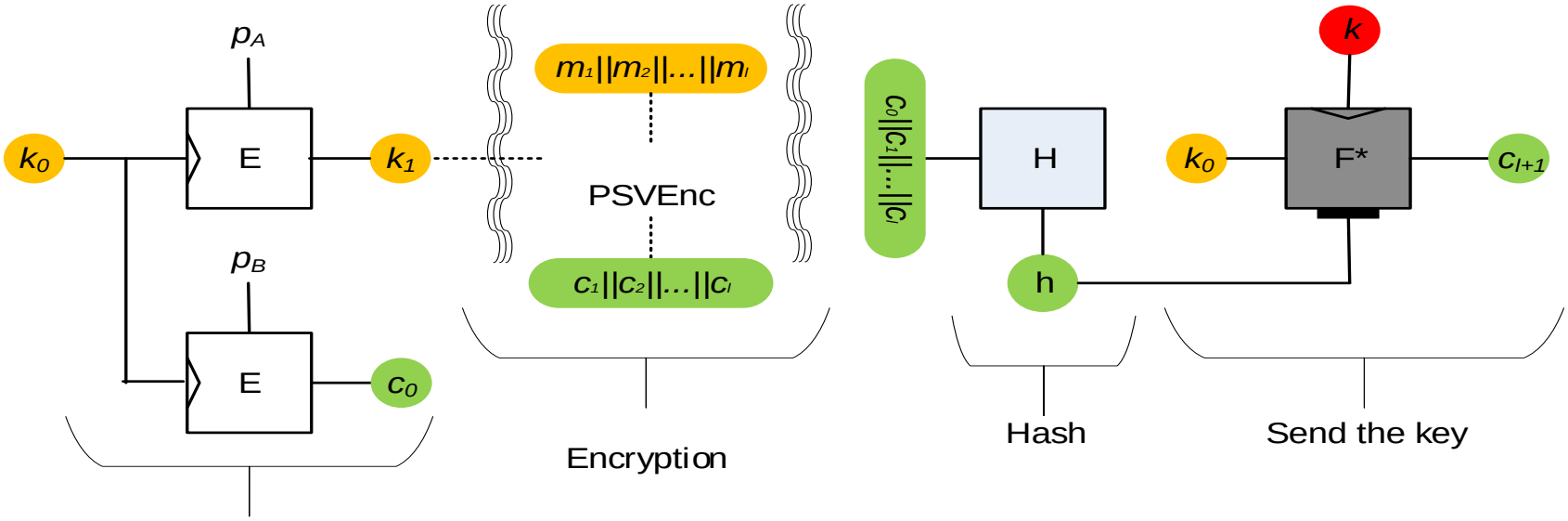
For decryption: retrieve k_1, recompute c_0 and check if correct, then, apply PSVDec.

CONCRETE security

CONCRETE is

- AE secure
- CIML2 secure
- RUPAE secure
- CPAL2 secure

CONCRETE proofs

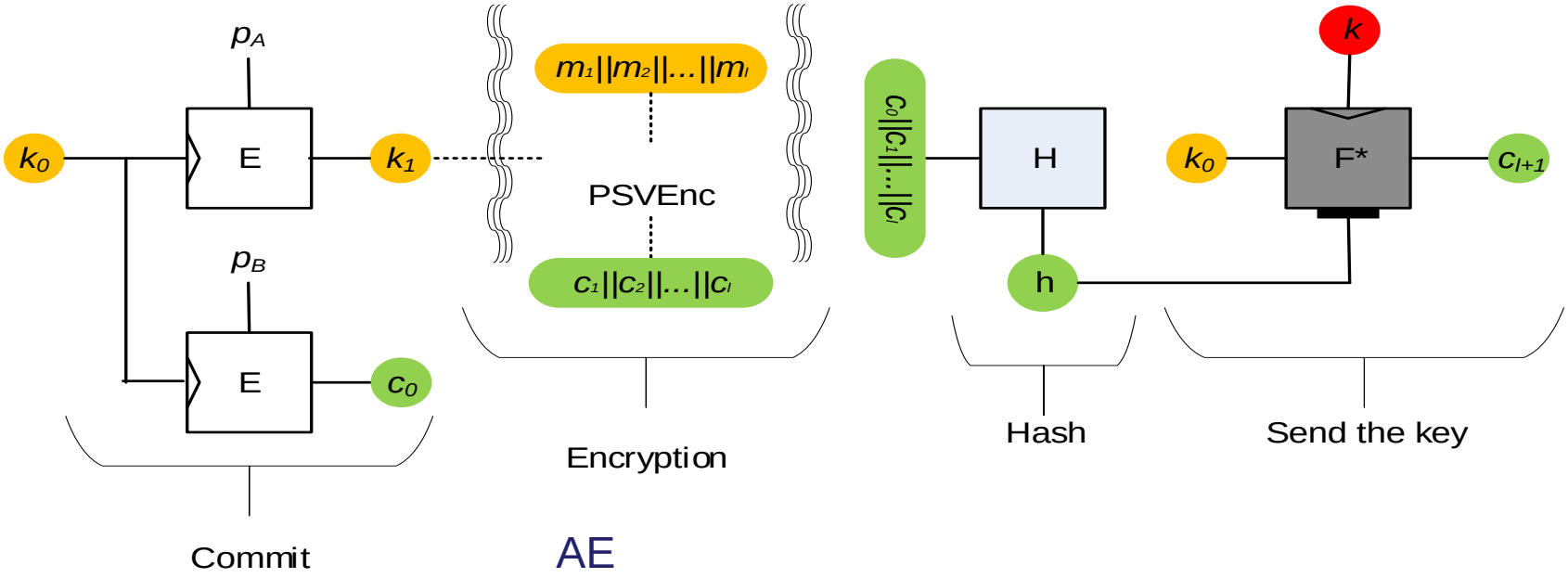


CIML2

Consider a forgery $(c_0, \dots, c_l, c_{l+1})$:

- if $(c_0, \dots c_l)$ is fresh then *h* is fresh (collision resistance).
  - Thus k_0 is fresh and $\Pr[c_0 = \mathrm{E}_{k_0}(p_B)]$ is negligible
- If $(c_0, \dots c_l)$ is fresh then $((c_0, \dots, c_l), c_{l+1})$
  - Thus $k_0 = F_k^{*,-1}(h, c_{l+1})$ is fresh and $\Pr[c_0 = \mathrm{E}_{k_0}(p_B)]$ is negligible

CONCRETE proofs

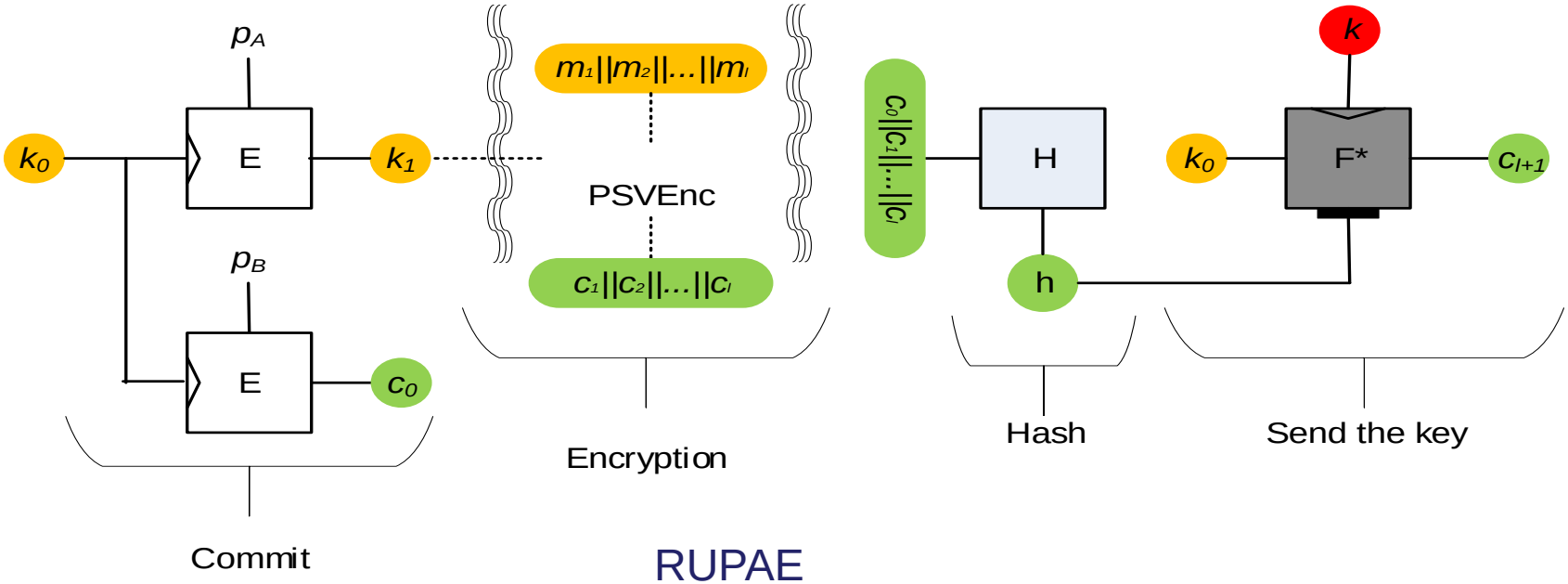


AE

CONCRETE is CIML2 secure, thus INT-CTXT.

- If k_0 is fresh then PSVEnc gives a random output
  - c_{l+1} s also fresh since *h* is fresh

CONCRETE proofs

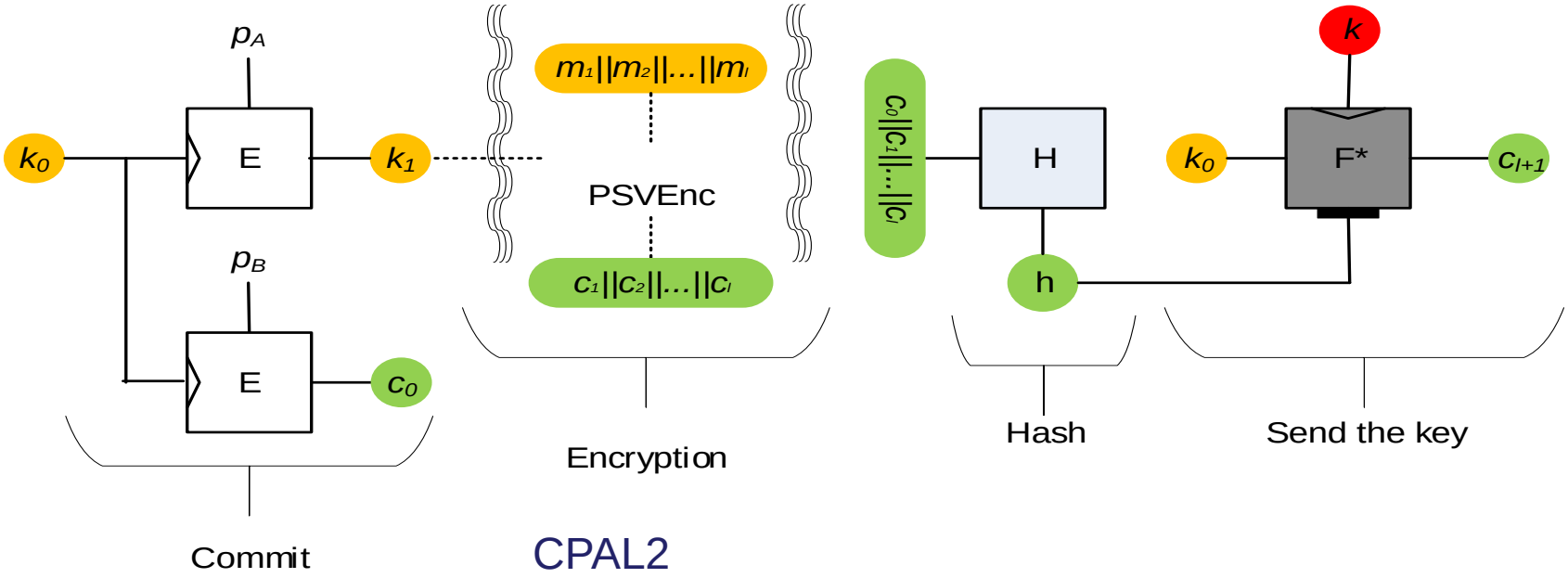


RUPAE

If $(c_0, \ldots, c_l, c_{l+1})$ is fresh then (c_{l+1}, h) is fresh,

- Thus k_0 is fresh
  - Thus PSVDec gives a random output

CONCRETE proofs



CPAL2

Given by PSVEnc, since k is always fresh in encryption

Conclusion

- CIML2 does not need a leak free TBC
  - It is enough strong unpredictability
- Spook may be CPAL even if the TBC is unpredictable
- CIML2-AE does not need 2 leak free calls to the TBC

Questions

THANK YOU