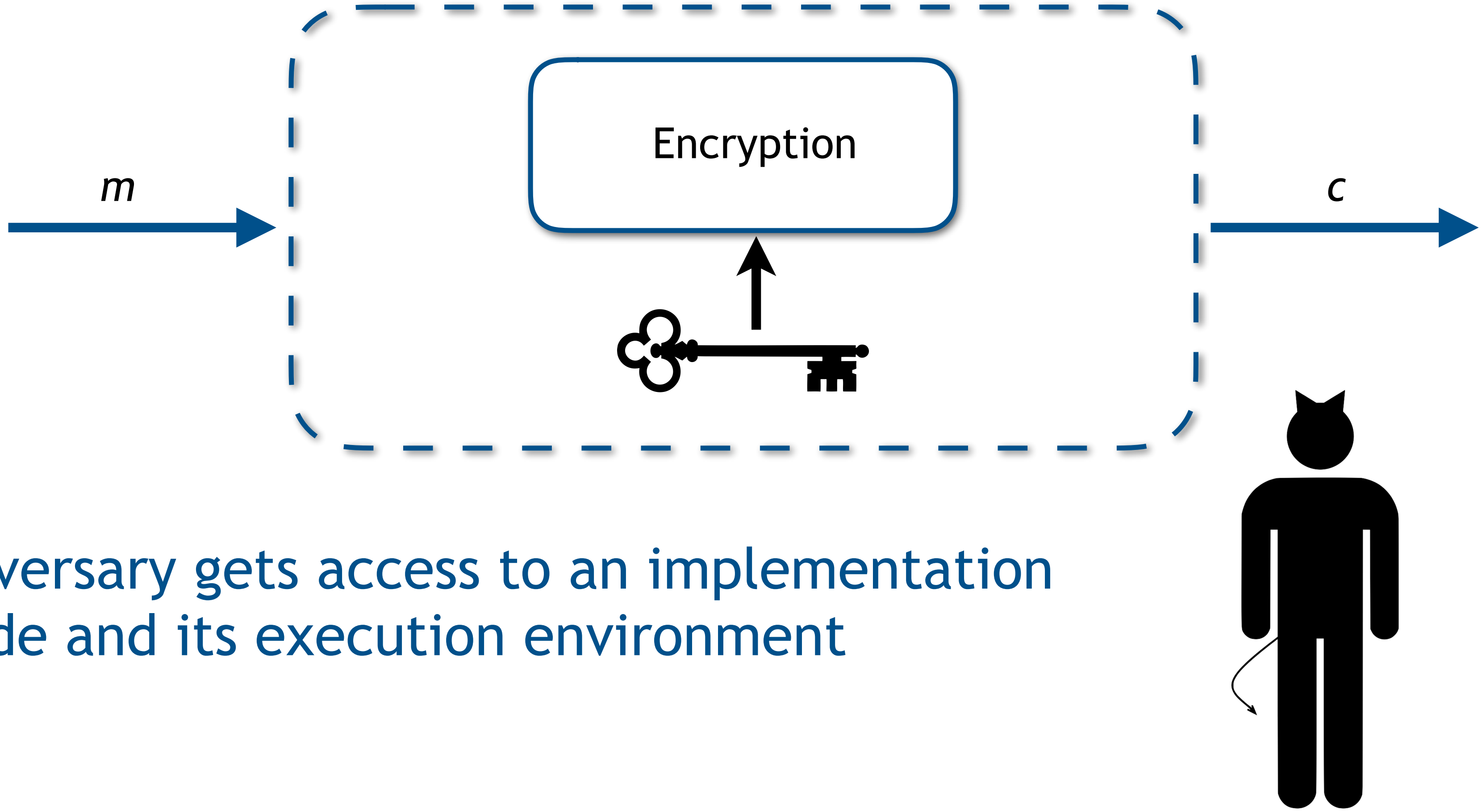
On the Security Goals of White-box Cryptography

Estuardo Alpírez Bock, Alessandro Amadori, Chris Brzuska, Wil Michiels

TU/e NXP Aalto University

White-box attack scenario



Adversary gets access to an implementation code and its execution environment

→ WB Cryptography aims to provide security even under such attack threats

On the security goals of white-box cryptography

Discuss the use cases of white-box cryptography (mobile payment and DRM applications, use of symmetric schemes to implement public key operations, etc.)

Discuss popular security notions for white-box crypto and their usefulness on different application scenarios

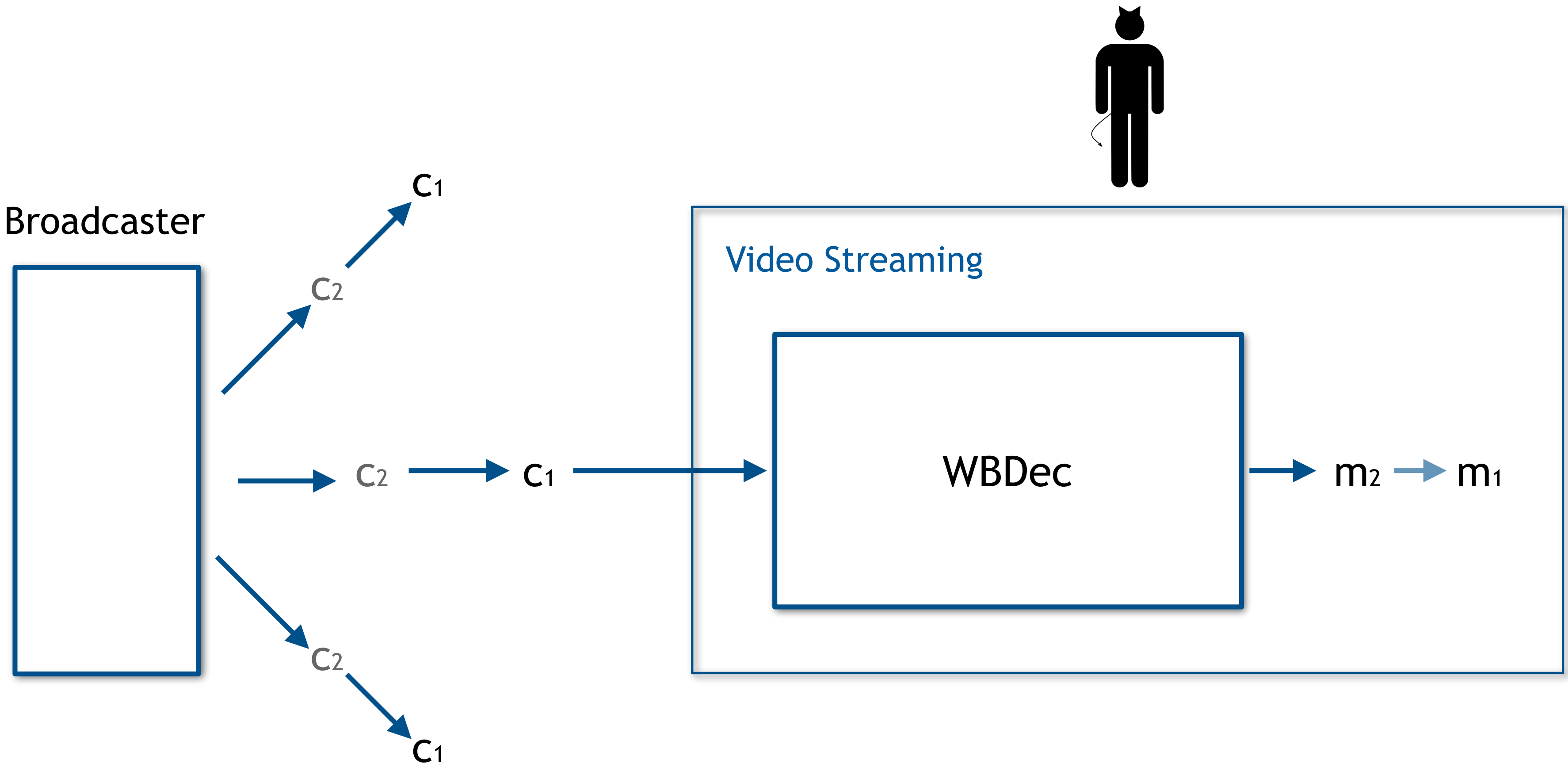
Propose to focus on the goals of hardware- and application-binding for achieving security for mobile payment applications. Provide a security definition for white-box encryption with hardware-binding

Present an impossibility result for general white-box compilers

Use cases in practice:

DRM and mobile payment applications

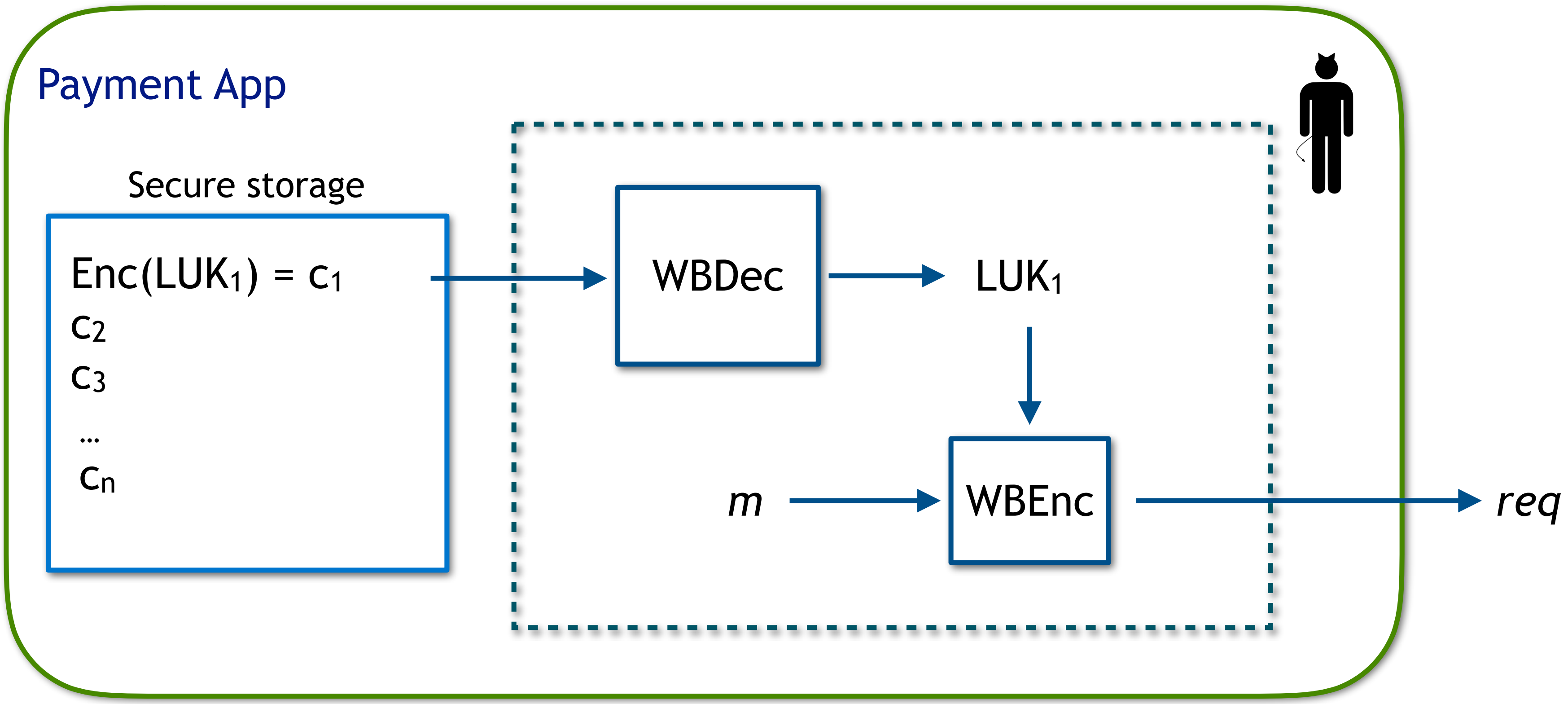
White-box crypto for DRM



- White-box crypto for mitigating piracy
- The owner of the application is considered to be the adversary

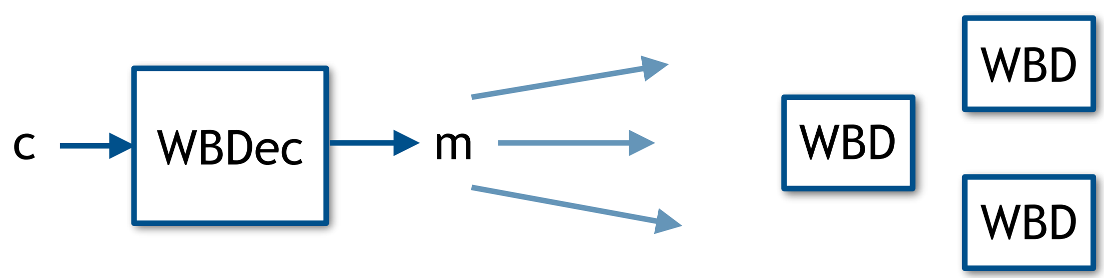
White-box crypto for payment applications

- Limited use keys (LUKs) used for encrypting a transaction request message



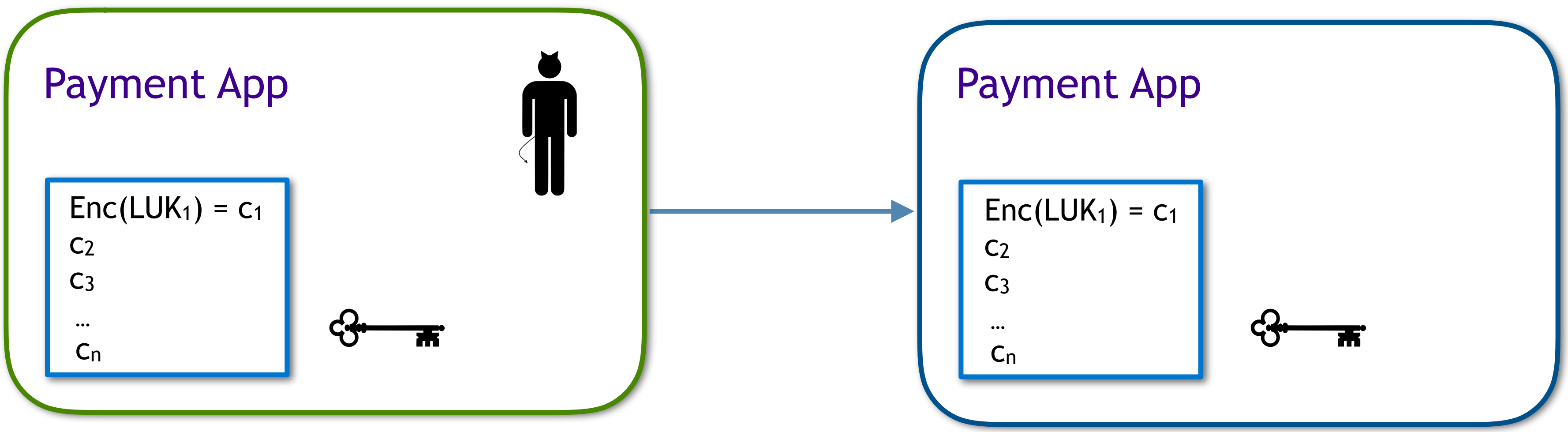
What are the goals of white-box crypto?

- Depending who we ask, the goal might be:
  - Hiding the key of a cipher (special purpose obfuscation)
    - Given access to implementation code, key extraction is a big threat
  - Hiding the key of an AES implementation (special purpose obfuscation)
    - Opinion motivated by the popular goal of white-boxing AES (Popularity of AES, first white-box paper by Chow et al., WhibOx competitions, etc.)
  - Mitigate redistribution attacks
    - Motivated by the use case of white-box crypto in DRM applications



White-box crypto for payment applications

- An adversary can copy the app and run it at a phone and terminal of its choice



We need protection against *code-lifting* attacks

Popular mitigation techniques against code-lifting attacks on white-box implementations:

Traceability and Incompressibility

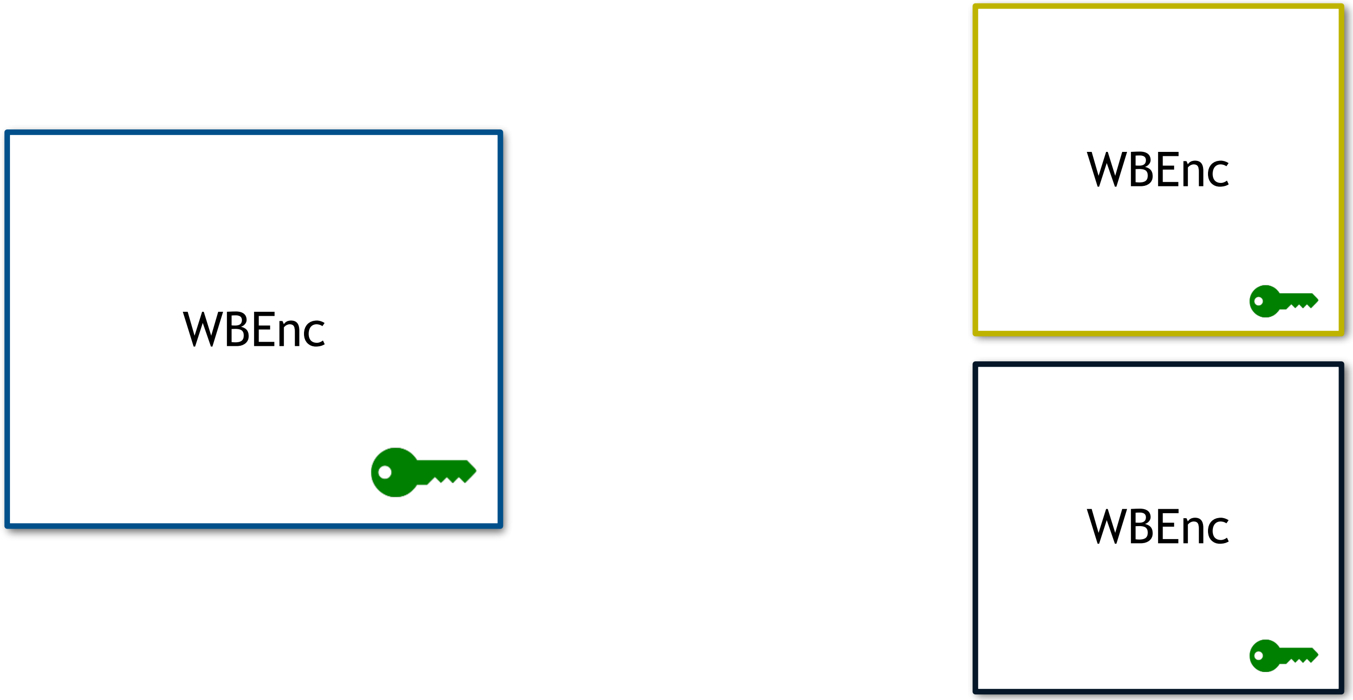
Popular notions and mitigation techniques

- The properties of traceability and incompressibility have gained popularity in the white-box community
- Security notions and constructions have been proposed e.g. in:
  - Delerablée, Lepoint, Paillier, Rivain - White-box security notions for symmetric encryption schemes, SAC 2013
  - Fouque, Karpman, Kirchner, Minaud - Efficient and provable white-box primitives, ASIACRYPT 2016
  - Bogdanov, Isobe, Tischhauser - Towards practical white box cryptography: optimizing efficiency and space hardness, ASIACRYPT 2016
  - Alpirez Bock, Amadori, Bos, Brzuska, Michiels - Doubly half-injective PRGs for incompressible white-box cryptography, CT-RSA 2019
  - Alex Biryukov - White-box and asymmetrically hard crypto design, WhibOx 2019 Workshop

These properties are considered due to the DRM use case. But how can they help us for protecting mobile payment applications?

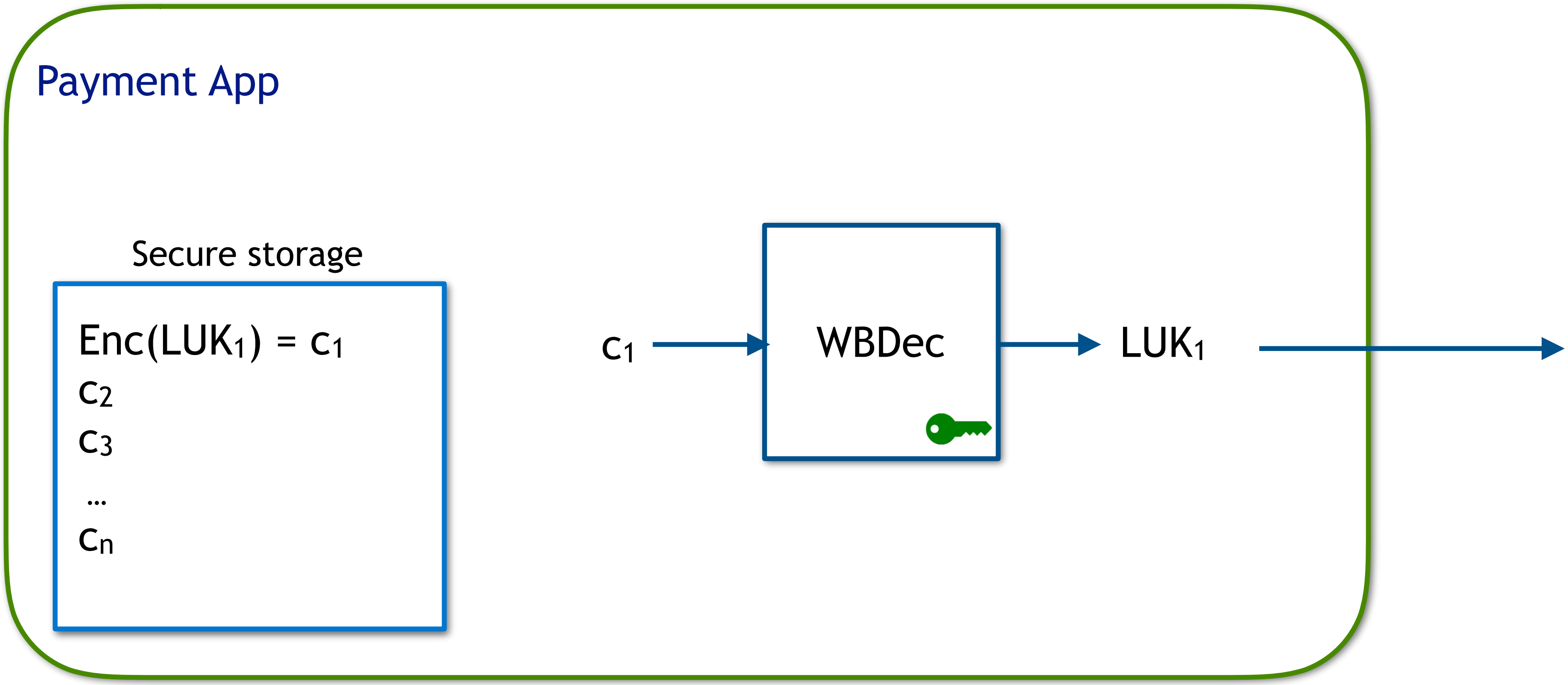
Traceability

- A white-box program is watermarked with a *tracing key*. Each program has its own tracing key.



The tracing key helps identify the origin of the copied program

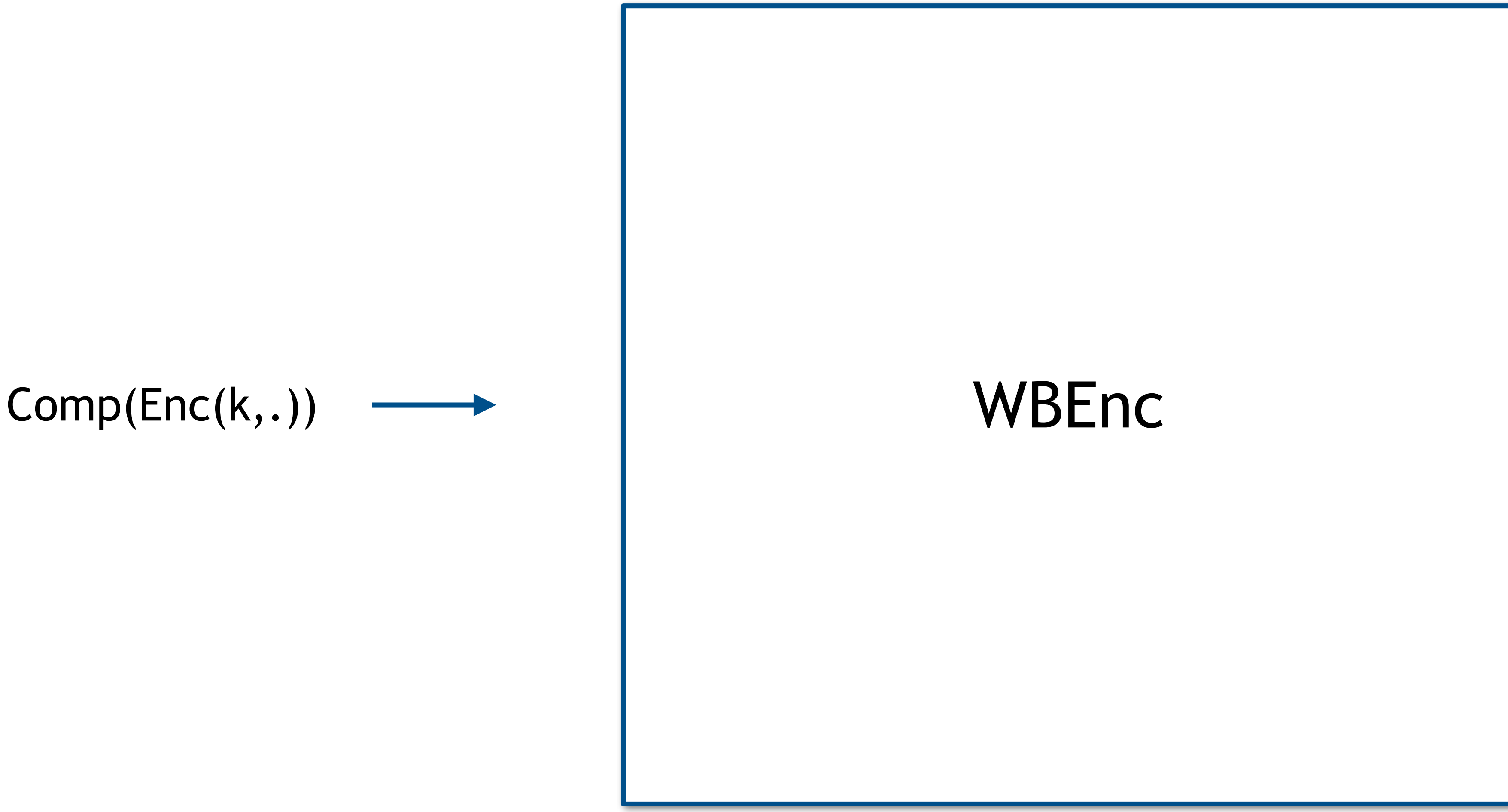
Traceability



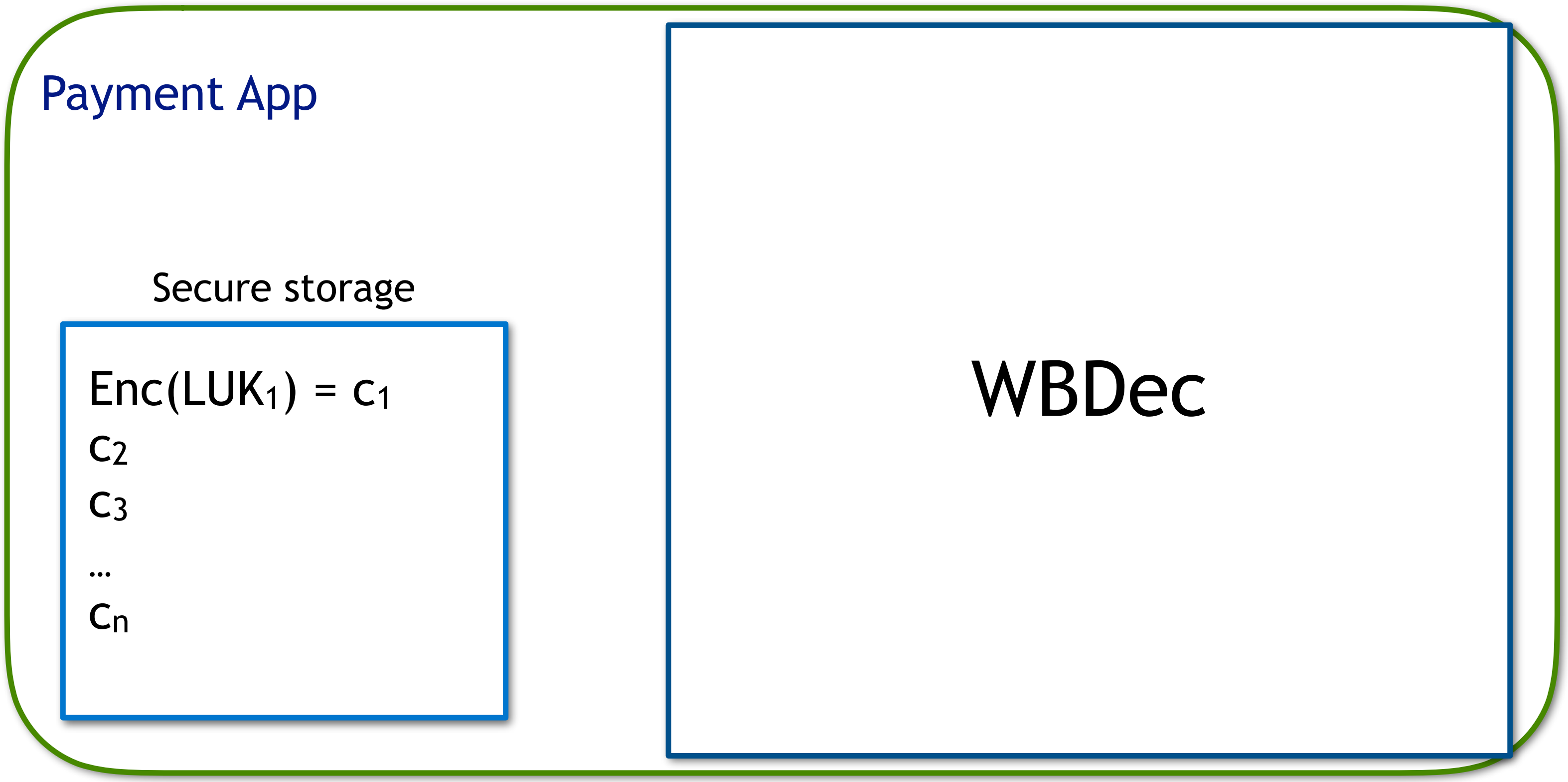
The owner of a payment application will not make copies of it and share it
This would enable people to access the user's keys, i.e. the user's money.

Incompressibility

- **Make a program very large in size. If the program is compressed or fragments are removed, the program loses its functionality.**



Incompressibility



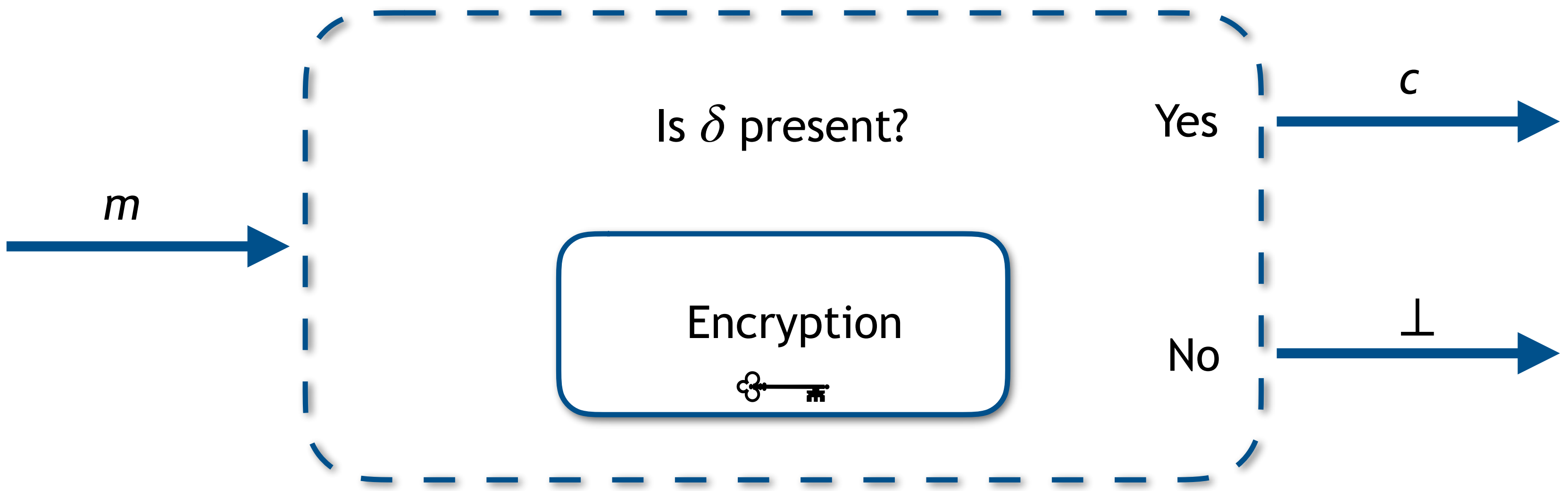
Large programs take too much space from a mobile application - contrast to IoT

Large programs are also difficult to distribute *legally*

Alternative methods for mitigating code-lifting attacks: hardware- and application-binding

Alternative: hardware-binding

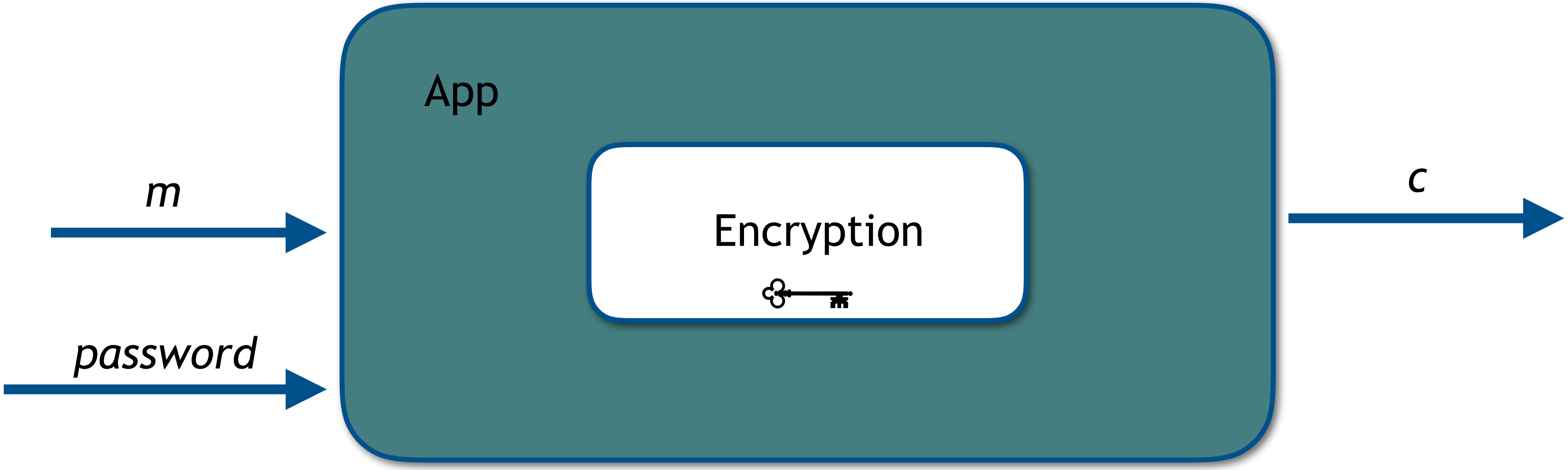
- An encryption program should only be executable on one specific device. The execution is dependable on a unique hardware identifier δ.



Alternative: application-binding

- **An encryption program should only be executable within one specific application**

Useful in the case that the application performs authentication operations



Defining hardware-binding

Defining Hardware-binding

For defining hardware-binding for white-box encryption, we follow the approach presented in [1]

[1] defines hardware binding for white-box KDFs and mobile payment applications in combination of a *hardware module*.

The work presents feasibility results based on indistinguishability obfuscation and puncturable PRFs

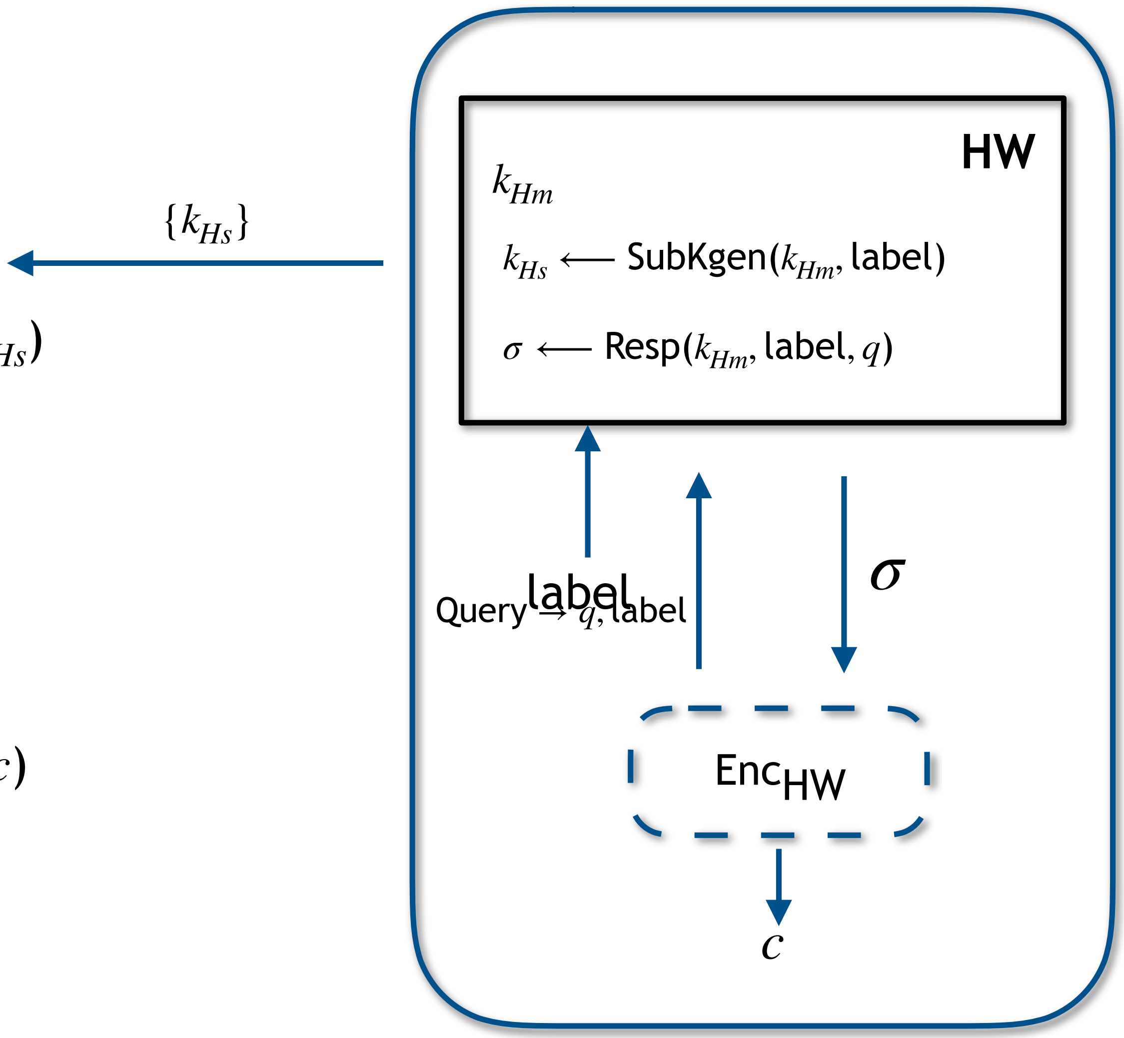
[1] E. Alpirez Bock, C. Brzuska, M. Fischlin, C. Janson, W. Michiels: Security reductions for white-box key storage in mobile payments, to appear in Asiacrypt 2020

Hardware module

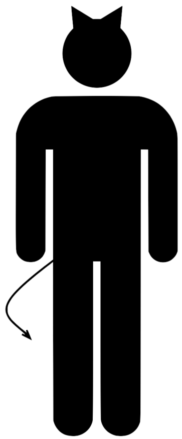
$$\text{Query}, \text{Enc}_{\text{HW}} \longleftarrow \$\text{Comp}(k, k_{Hs})$$

$$q \leftarrow \text{Query}(m, nc)$$

$$\text{Enc}_{\text{HW}}(m, nc, \sigma) = \text{Enc}(k, m, nc)$$



Security of White-box encryption



Enc_{HW}

Query

q →

← σ

HW(q)

assert $q \notin Q$

$Q := Q \cup \{q\}$

$\sigma \longleftarrow \text{Resp}(k_{Hm}, \text{label}, q)$

m_0, m_1 →

← (c, nc)

Enc(m_0,m_1)

assert $|m_0| = |m_1|$

$nc \longleftarrow \$\{0,1\}^n$

assert $q_i \notin Q$

with $q_i \leftarrow \text{Query}(m_i, nc)$

$Q := Q \cup \{q_i\}$

$c \longleftarrow \text{Enc}(k, m_b, nc)$

$C := C \cup \{(c, nc)\}$

(c, nc) →

← m

Dec(c)

assert $(c, nc) \notin C$

$m \leftarrow \text{Dec}(k, c, nc)$

assert $q \notin Q$

with $q \leftarrow \text{Query}(m, nc)$

if $b = 1$ $\quad m \leftarrow \perp$

else return m

Challenges defining application-binding

- What exactly is an application?

- Alternative: focus on specific applications, e.g. applications performing authentication operations:
  - A user authenticates himself via passwords or fingerprints. However, such values can be intercepted by a white-box adversary
    - Alternative: weaken the attack model. However, this leads to the following issues:
      - Presents an inconsistent attack scenario

      - In order to define security, we need to consider long enough secret authentication values. In that case, we could even consider a keyless white-box implementation

Conclusions

- White-box cryptography needs to achieve more than *only* security against key extraction
- Hardware binding seems to be a reasonable technique for achieving this
  - It seems necessary and effective for most use cases. We propose a security definition for white-box encryption.
    - Known feasibility results are based on iO and puncturable PRFs
- Application binding seems also a reasonable goal for real life applications of white-box crypto
  - It is however more difficult to define formally

Thank you for your attention!