PUFKY

A FULLY FUNCTIONAL PUF-BASED CRYPTOGRAPHIC KEY GENERATOR

Anthony Van Herrewege, Roel Maes, Ingrid Verbauwhede

KU Leuven, COSIC

CHES 2012 - Leuven, Belgium
2012-09-14

KATHOLIEKE UNIVERSITEIT LEUVEN

Introduction

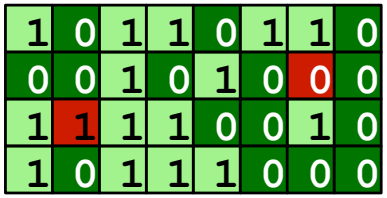
- Key generation
- Key storage

- Holy grail: PUF

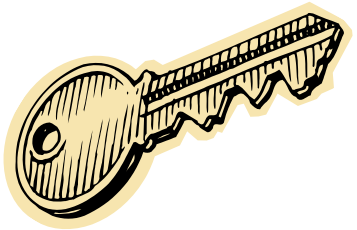
Concept

- PUF-based key generator

1. PUF



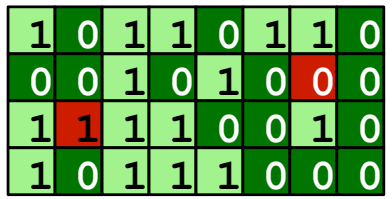
2. Reliability

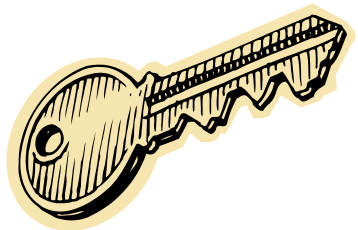


3. Unpredictability

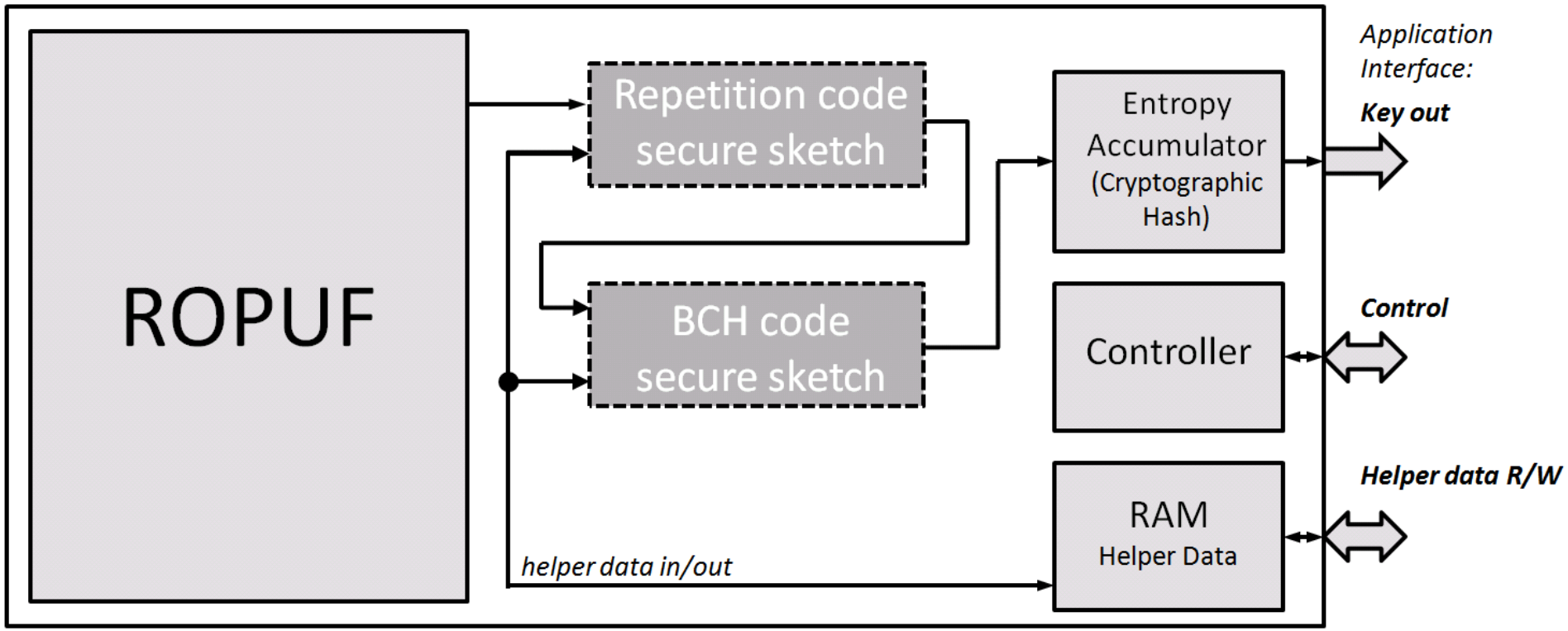
Concept

1. PUF
   - Error rate p_e
   - Entropy H
2. Error correction
   - $H' = H - (n - k)$
     > 0
   - $k \uparrow \Leftrightarrow t \downarrow \Rightarrow H' \uparrow$
   - Failure rate p_g
3. Entropy accumulation
   - $L \leq \text{Accum}(H')$

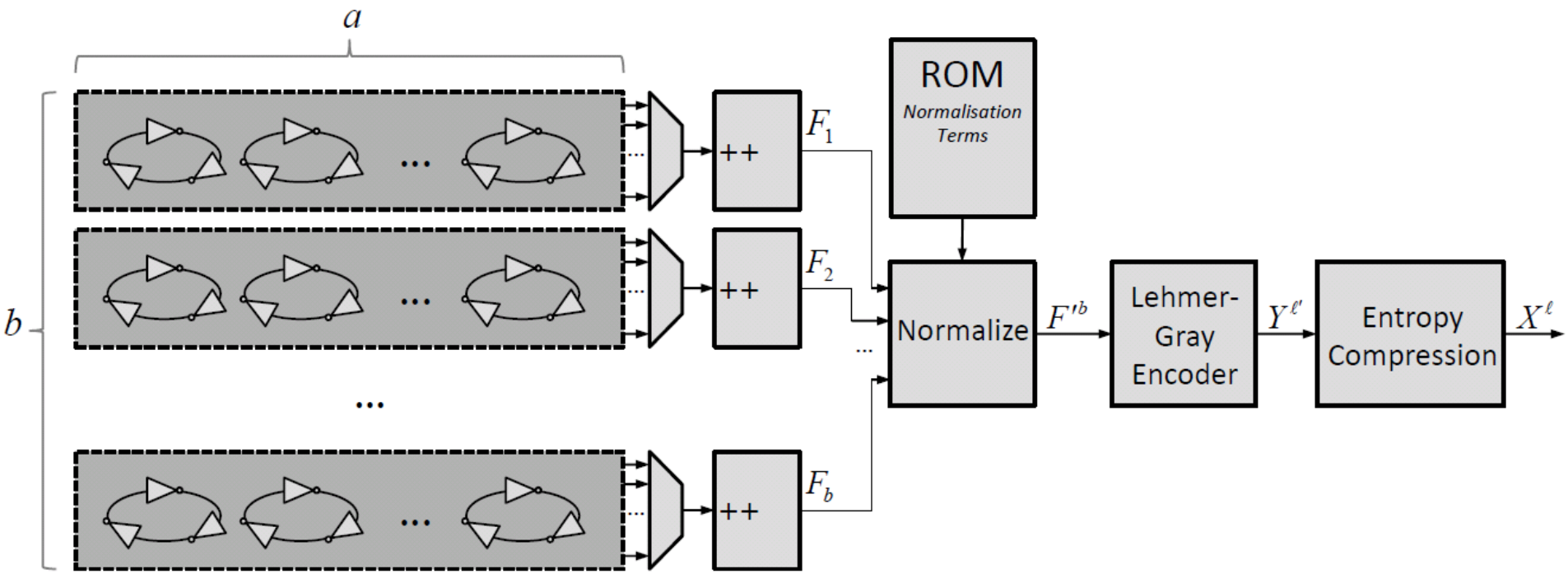




Design


ROPUF
Repetition code secure sketch
BCH code secure sketch
Entropy Accumulator (Cryptographic Hash)
Controller
RAM Helper Data
helper data in/out
Application Interface:
Key out
Control
Helper data R/W

Step 1: PUF


a
b
++
++
++
F_1
F_2
F_b
ROM
Normalisation Terms
Normalize
F'^b
Lehmer-Gray Encoder
$Y^{\ell'}$
Entropy Compression
X^{ℓ}

Step 1: PUF

- # possible oscillator orderings: $n!$
- Lehmer encoding: $\sum_{i=2}^{n} \lceil \log_2 i \rceil$ vs $\sum_{i=2}^{n} \log_2 i$

D B A C	→ 1	[2 bits]
B A C	→ 1 3	[2 bits]
B A	→ 1 3 1	[1 bit]
A	→ 1 3 1	

Step 1: PUF

- Gray encoding
- Entropy compression: increase H/bit

  100 → 10

  011 → 11

  001 → 01

  000 → 00

Step 2: Error correction

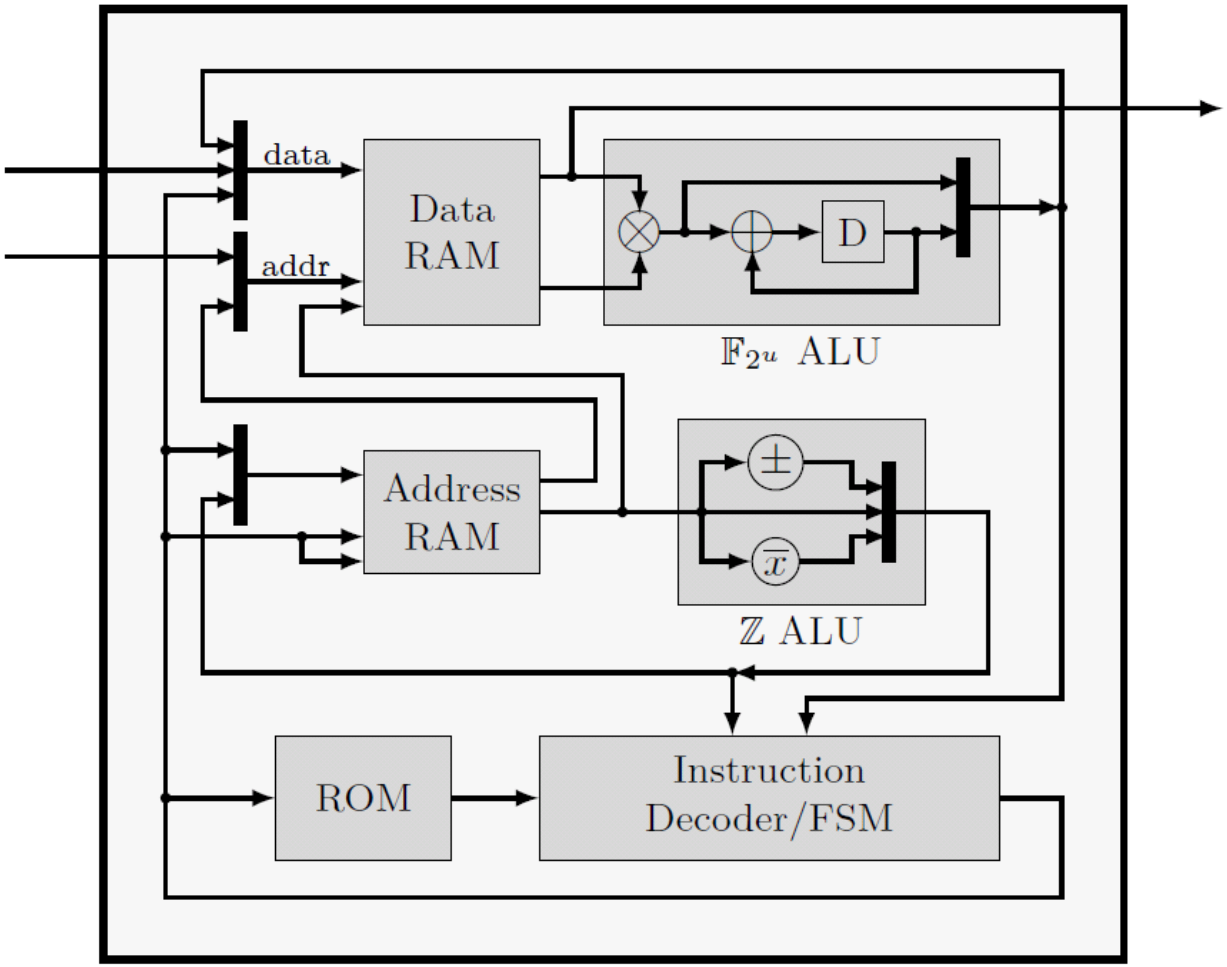
- Secure sketch construction

$$h = y \cdot H^T$$

$$y' = y + e$$

$$\begin{aligned} s &= h + y' \cdot H^T \\ &= (y + y') \cdot H^T \\ &= (y + y + e) \cdot H^T \\ &= e \cdot H^T \end{aligned}$$

Step 2: Error correction



See: A. Van Herrewege, and I. Verbauwhede, "Tiny Application-Specific Programmable Processor for BCH Decoding," In *International Symposium on System-on-Chip - SoC '12*, 2012 (to appear).

Step 2: Error correction

- Instruction set: 16 opcodes
- Conditional execution
- Array based programming

- Does **not** have to be large: 112 slices

Step 3: Entropy accumulation

- Universal hash
- Cryptographic hash: NIST recommendation

  (NIST Special Publication 800-90A)

  → SPONGENT-128

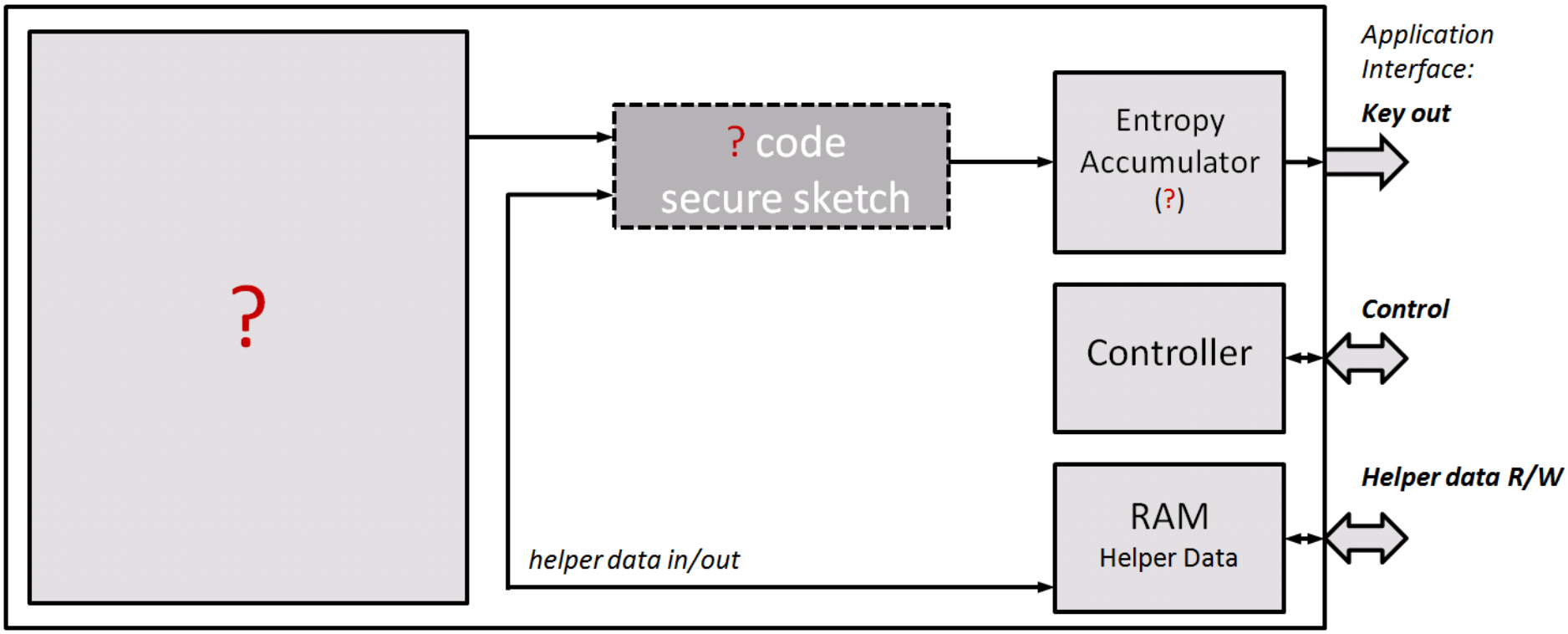
Results

- For 128-bit key, failure rate 10^{-9}
- Small: 1162 slices total
- Runtime: 5.62ms (82% PUF)

(a) Area consumption

Module	Size [slices]
ROPUF	952
REP decoder	37
BCH syndrome calc.	72
BCH decoder	112
SPONGENT-128	22
helper data RAM	38
Total	1162

Modularity


?
? code
secure sketch
Entropy
Accumulator
(?)
Controller
RAM
Helper Data
helper data in/out
Application
Interface:
Key out
Control
Helper data R/W

Conclusion

- **Small** PUF key generation **module**
- **Tiny BCH decoder** is possible
- Use of **standard algorithms**
- Use as **black box**

Questions?