Leakage-Resilient Pseudorandom Functions and Generators using Re-keying

Seminar on Security of Embedded Electronic Systems

Sonia Belaïd[1,2]

École Normale Supérieure, 45 rue d'Ulm 75005 Paris

Thales Communications & Security

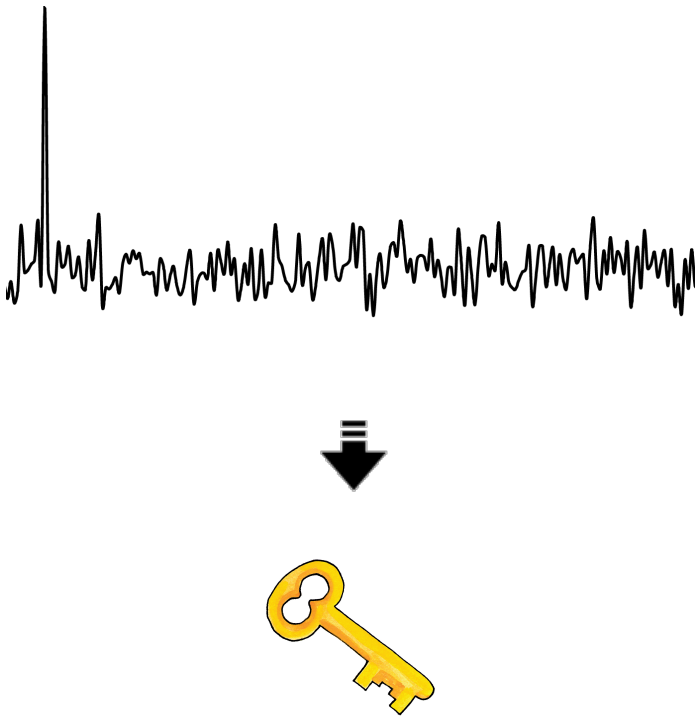
Sonia.Belaid@ens.fr

Outline

Outline

Side-Channel Attacks

- physical leakage
  - timing
  - power consumption
  - electromagnetic radiations
  - ...
- statistical treatment
- key recovery



Countermeasures against Side-Channel Attacks

Masking

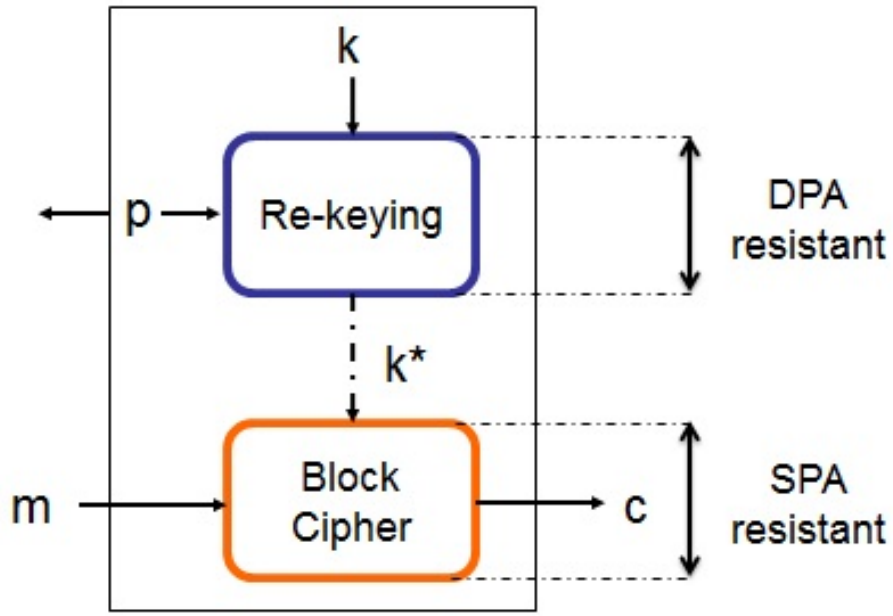
sensitive values randomized

x replaced by $x_m = x \star m$

Drawbacks of Masking

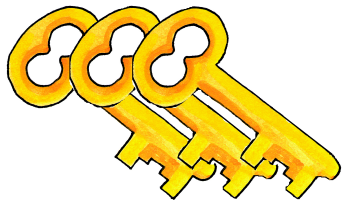
- ✘ higher-order attacks
- ✘ glitches

Re-keying



Contributions

Masking and Leakage-Resilient Primitives: One, the Other(s) or Both?
Sonia Belaïd, Vincent Grosso, François-Xavier Standaert
IACR Cryptology ePrint Archive 2014: 53 (2014)

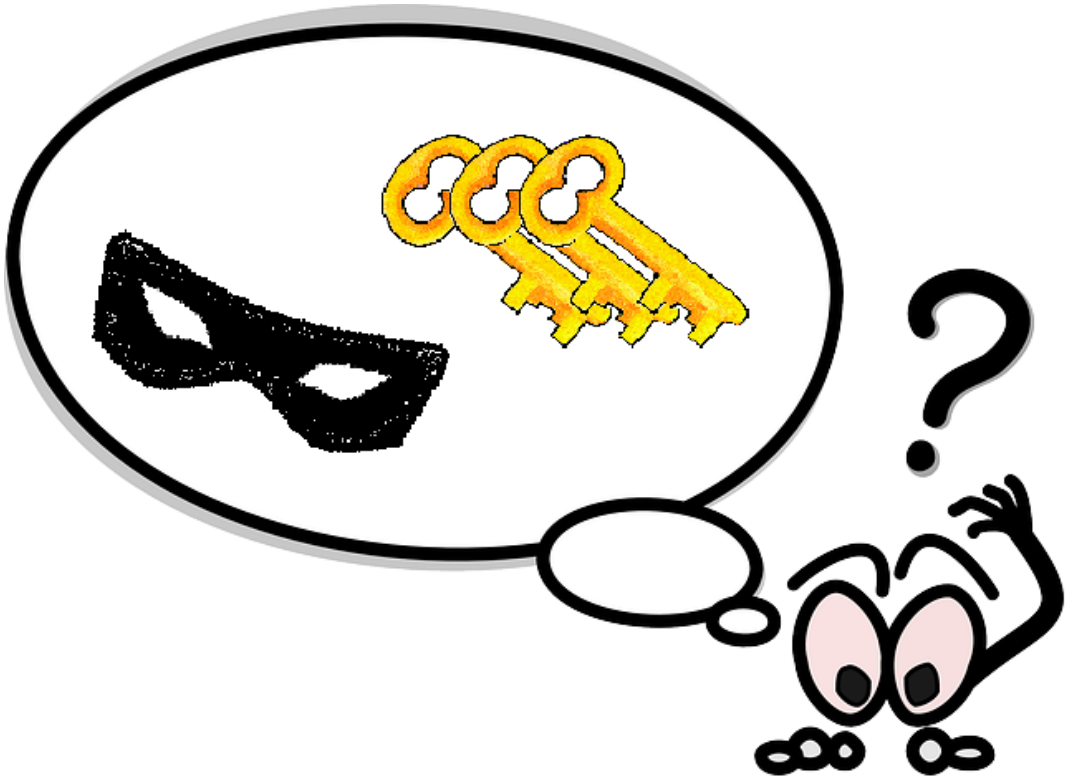


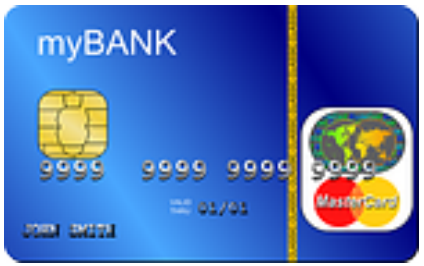
Leakage-Resilient Symmetric Encryption via Re-keying
Michel Abdalla, Sonia Belaïd, Pierre-Alain Fouque
CHES 2013: 471-488

Leakage-Resilient PRNG with Input (work in progress)
Michel Abdalla, Sonia Belaïd, David Pointcheval, Sylvain Ruhault, Damien Vergnaud
2014

Outline

Masking, Re-keying: One, the Other or Both?





Stateful PRG / Stateless PRF

Stateful PRG: limits the *number of measurements* with the same data by design

Stateless PRF: limits only the *data complexity* so an adversary can repeat the same measurement multiple times (e.g. to get rid of the physical noise)

Methodology

Target:	AES-128 for 80-bit, 100-bit and 120-bit security levels
Implementation:	software and hardware
Cost functions:	'code size $\times$ cycle counts' or 'area $/$ throughput'
Security Evaluation:	template attacks and security graphs
Global Cost Metric:	frequence of re-keying and performances

Results Stateful PRG

Stateful PRG: number of measurements $=$ data complexity
$\Rightarrow$ security-bounded implementations

Re-keying: after M measurements

Global Cost: $\frac{M}{M-1} \times$ (AES cost function)

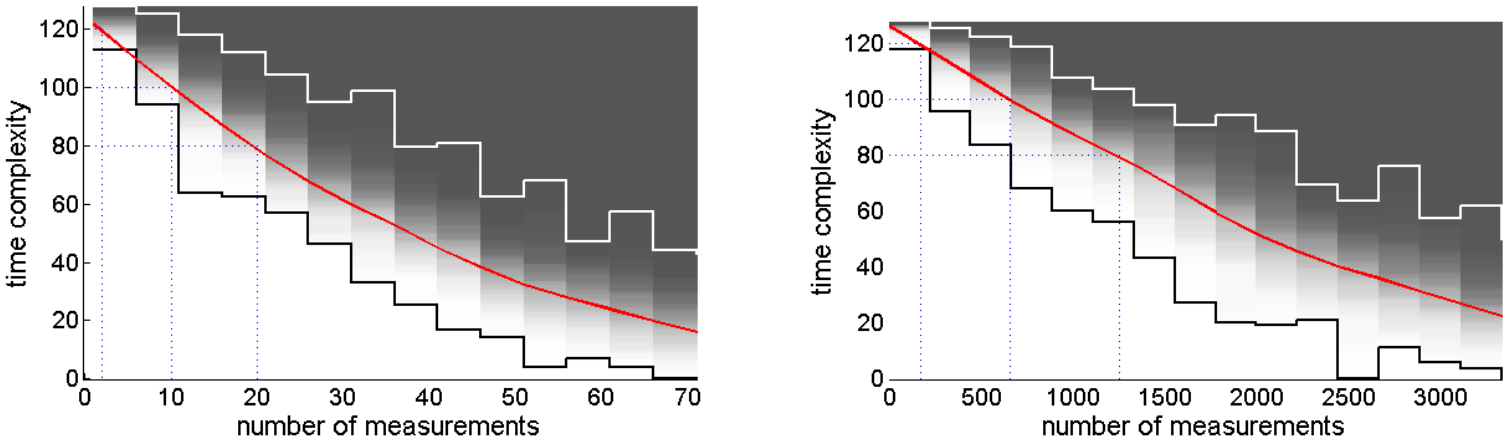


Figure: DPA-based security graphs for KSU (left) and KSB_1 (right).

Results Stateful PRG (2/2)

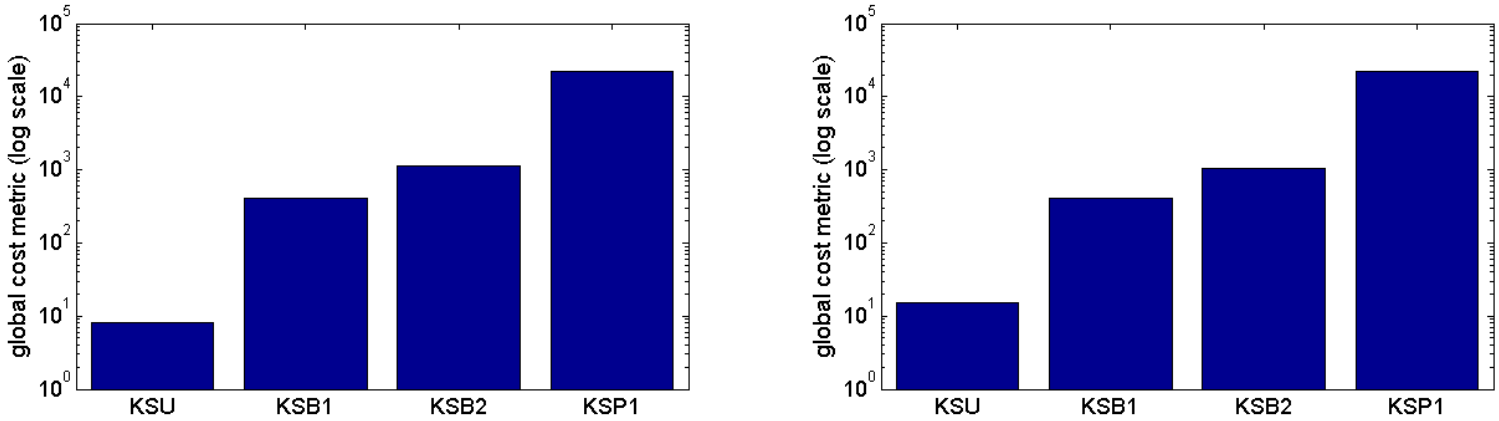


Figure: LR-PRGs in software. 80-bit (left) and 120-bit (right) security.

Results Stateful PRG (2/2)

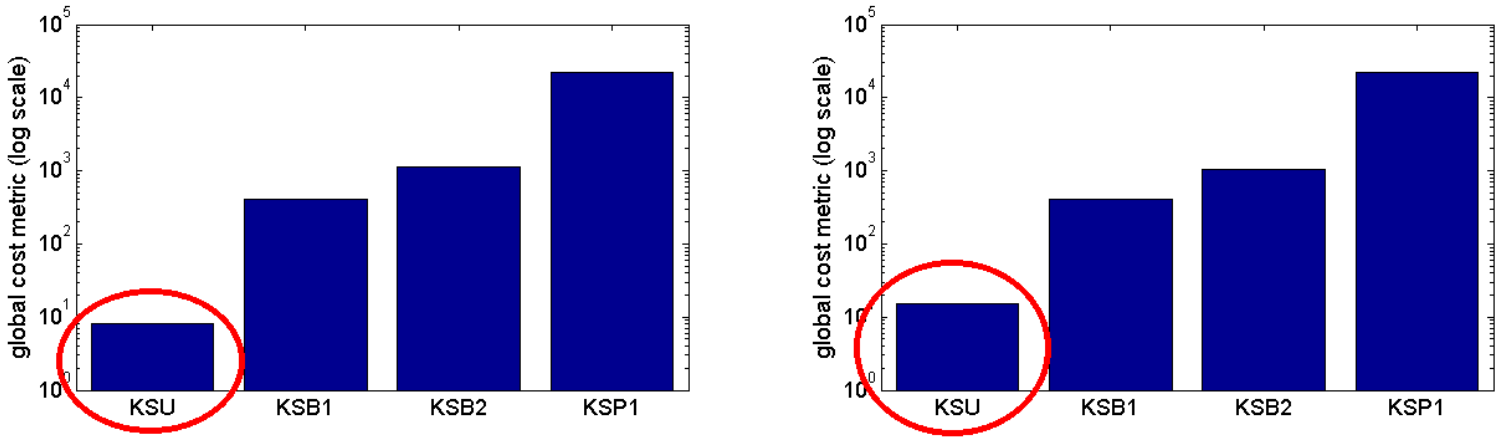


Figure: LR-PRGs in software. 80-bit (left) and 120-bit (right) security.

Conclusion: re-keying is the most efficient way to achieve every security level

Results Stateless PRF

Stateless PRF: number of measurements $\neq$ data complexity

$\Rightarrow$ security-unbounded implementations

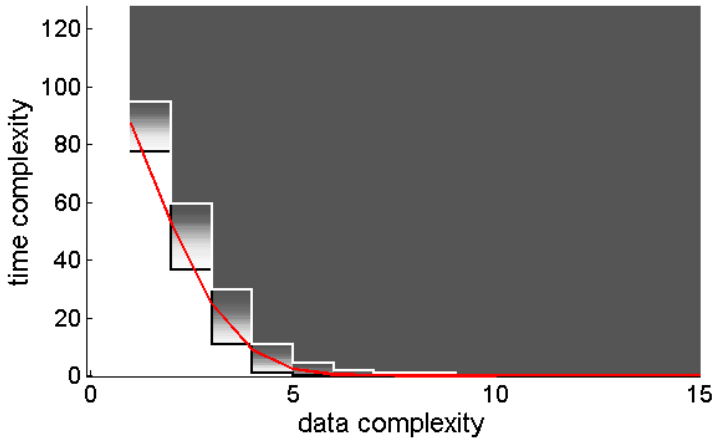


Figure: DPA-based security graph for repeating attack on AES.

Results Stateless PRF

Stateless PRF: number of measurements $\neq$ data complexity

$\Rightarrow$ security-unbounded implementations

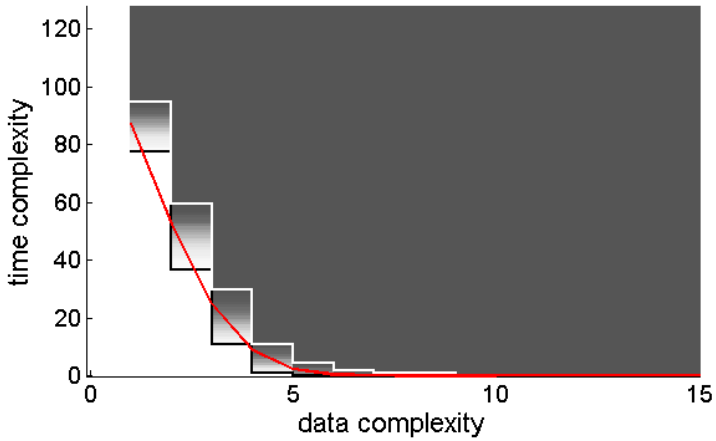


Figure: DPA-based security graph for repeating attack on AES.

Conclusion: the lifetime of the system must be limited according to the countermeasure

Results Stateless PRF (2/2)

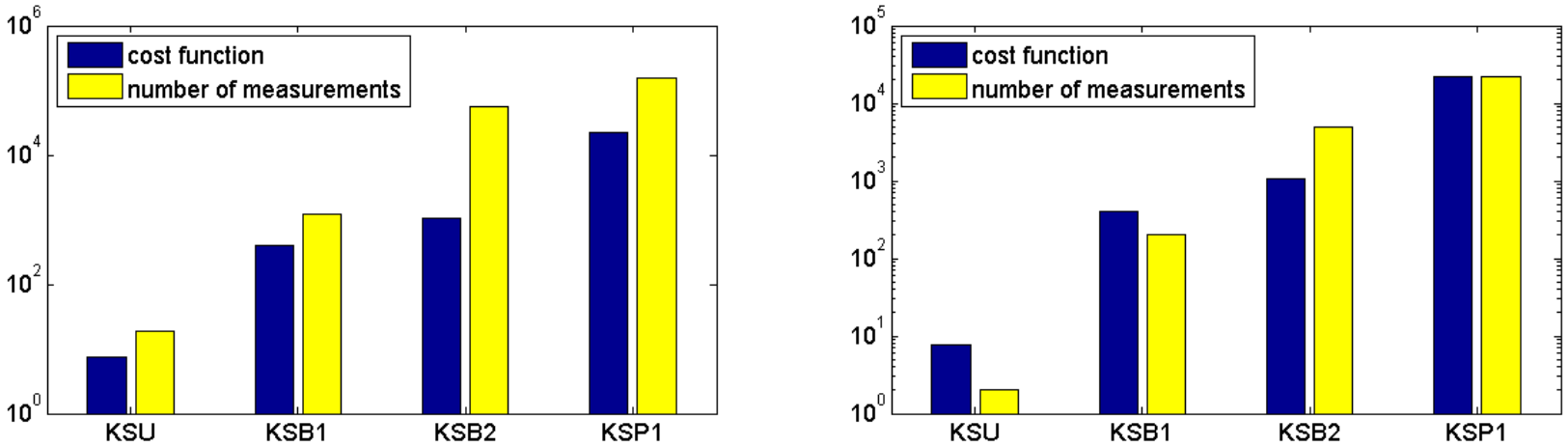


Figure: LR-PRFs in software with KP. 80-bit (left) and 120-bit (right) security.

Results Stateless PRF (2/2)

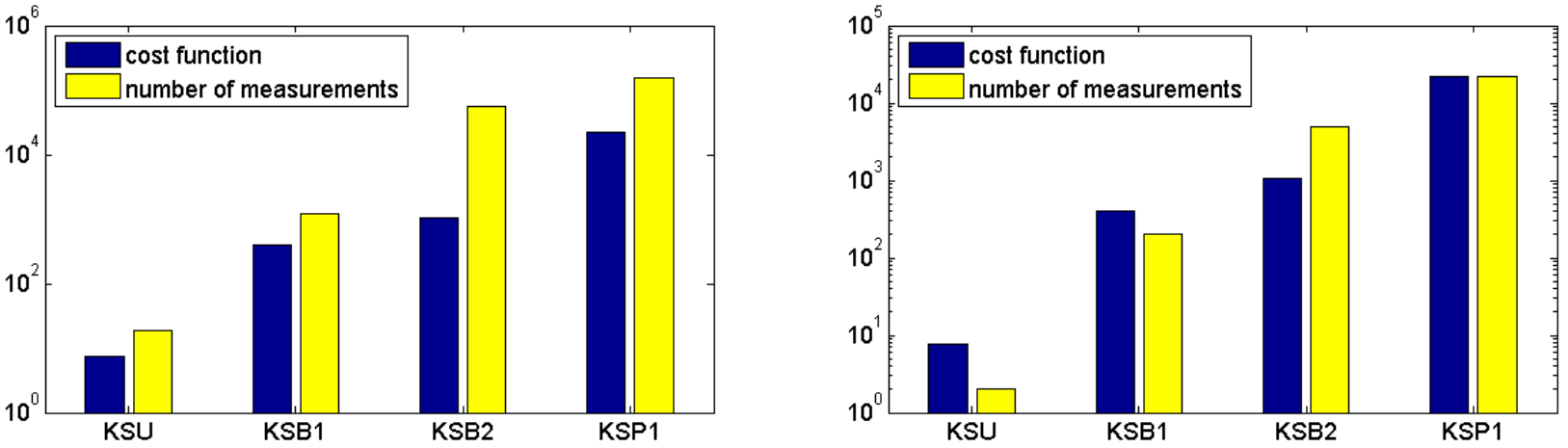
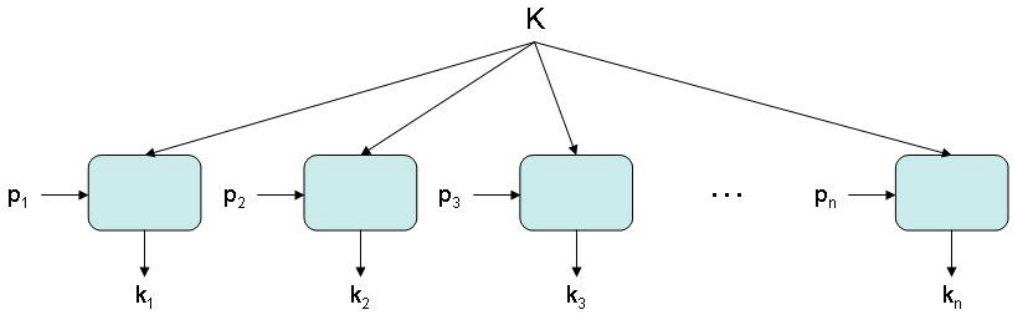


Figure: LR-PRFs in software with KP. 80-bit (left) and 120-bit (right) security.

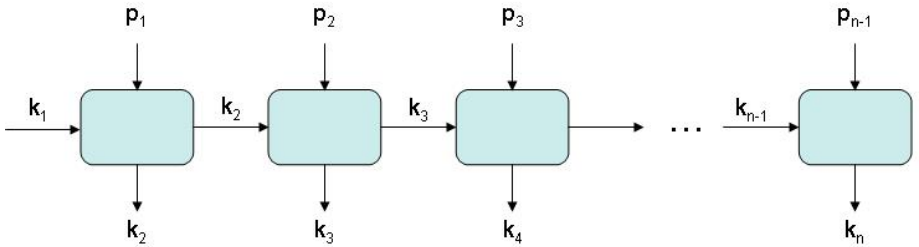
Conclusion: masking alone and limiting the lifetime is the best combination

Outline

Two Main Re-keying Schemes

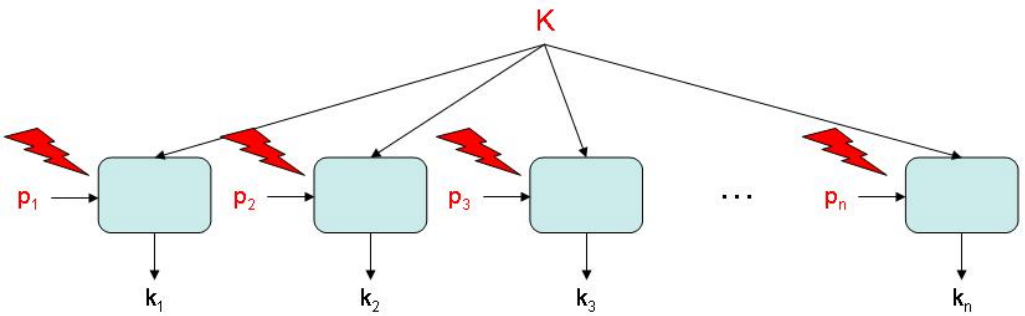


Parallel Scheme



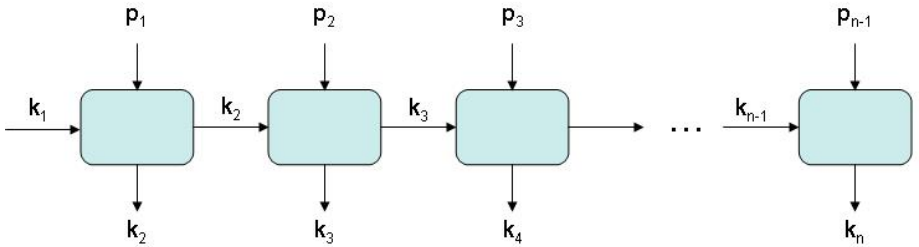
Sequential Scheme

Two Main Re-keying Schemes



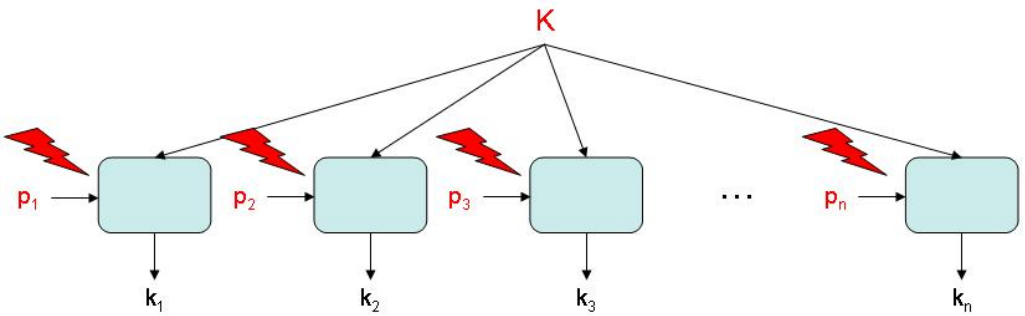
Parallel Scheme

vulnerable to *Differential Power Analysis*



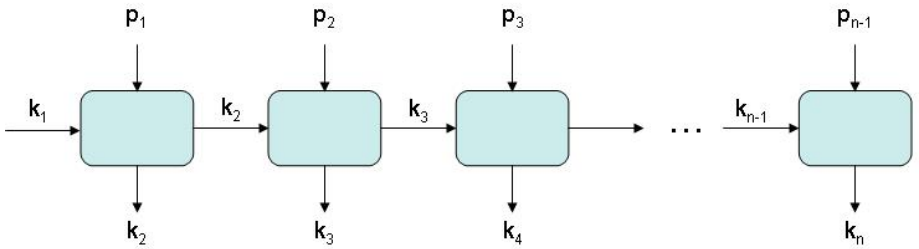
Sequential Scheme

Two Main Re-keying Schemes



Parallel Scheme

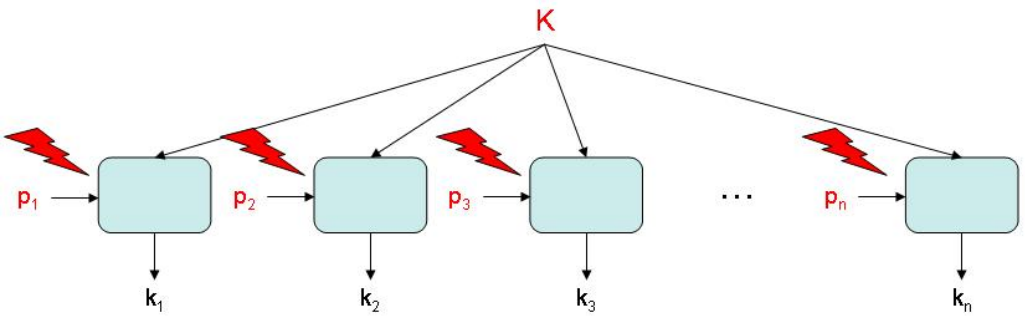
vulnerable to *Differential Power Analysis*



Sequential Scheme

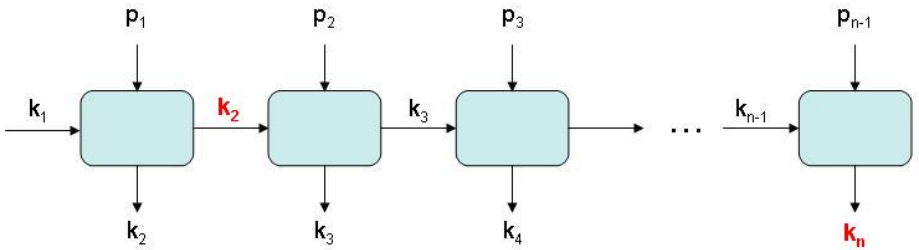
efficiency issue in case of *synchronization*

Two Main Re-keying Schemes



Parallel Scheme

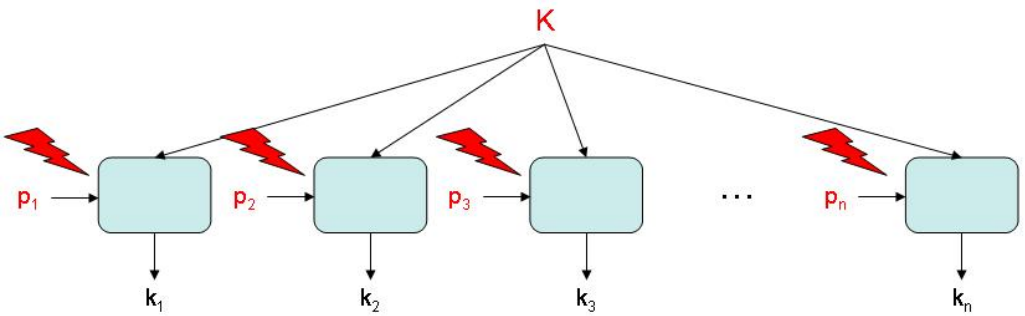
vulnerable to *Differential Power Analysis*



Sequential Scheme

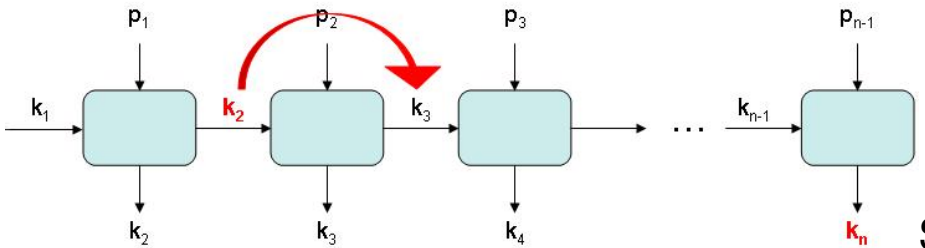
efficiency issue in case of *synchronization*

Two Main Re-keying Schemes



Parallel Scheme

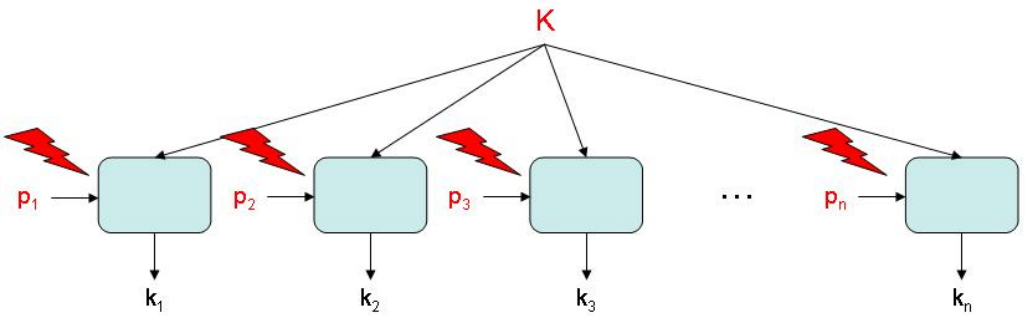
vulnerable to *Differential Power Analysis*



Sequential Scheme

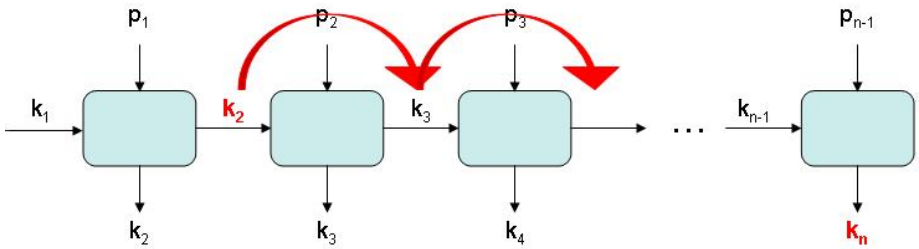
efficiency issue in case of *synchronization*

Two Main Re-keying Schemes



Parallel Scheme

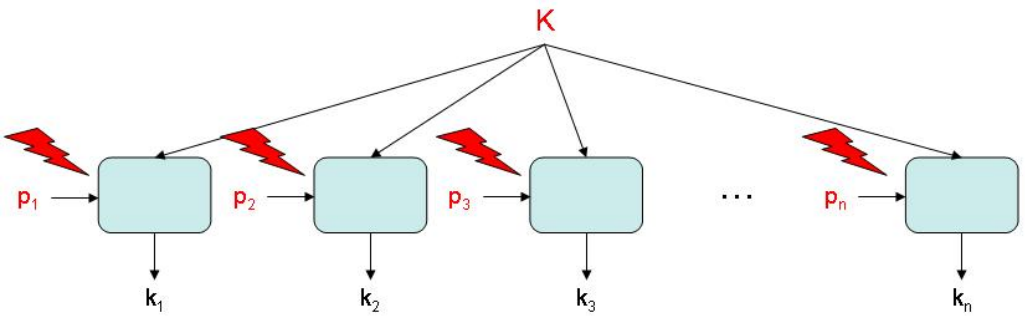
vulnerable to *Differential Power Analysis*



Sequential Scheme

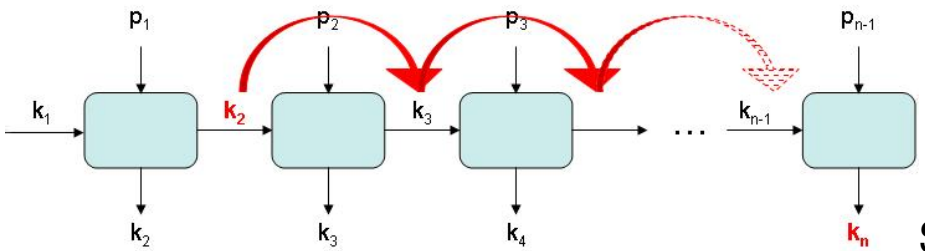
efficiency issue in case of *synchronization*

Two Main Re-keying Schemes



Parallel Scheme

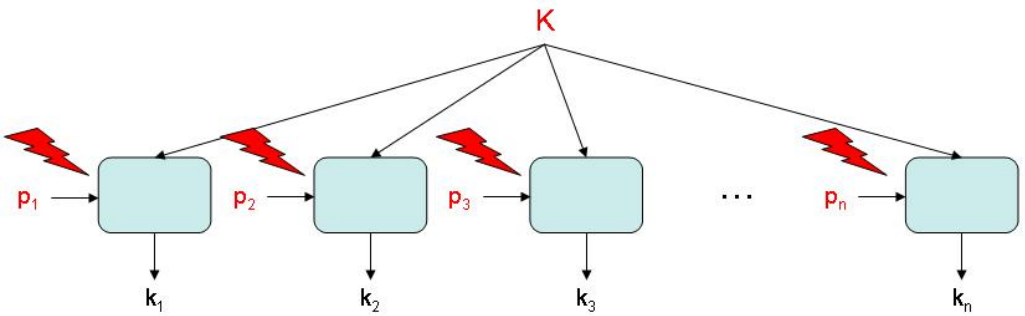
vulnerable to *Differential Power Analysis*



Sequential Scheme

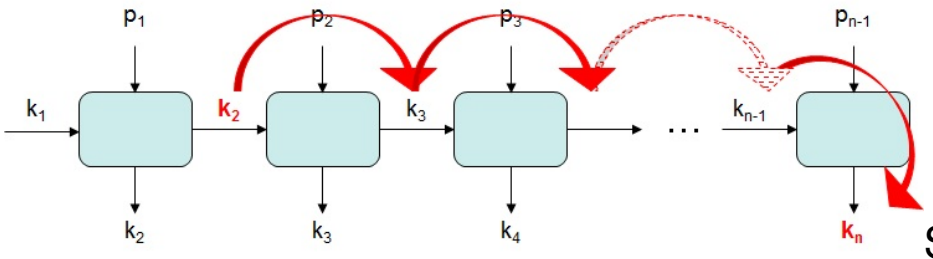
efficiency issue in case of *synchronization*

Two Main Re-keying Schemes



Parallel Scheme

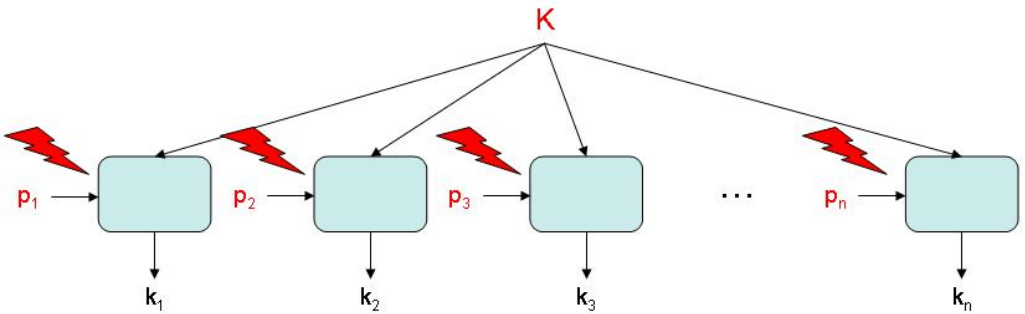
vulnerable to *Differential Power Analysis*



Sequential Scheme

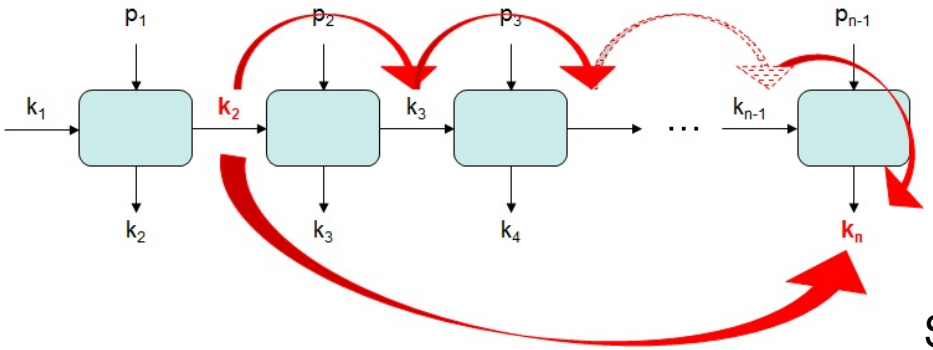
efficiency issue in case of *synchronization*

Two Main Re-keying Schemes



Parallel Scheme

vulnerable to *Differential Power Analysis*



Sequential Scheme

efficiency issue in case of *synchronization*

Our Contributions

- ✔ re-keying scheme (different from existing ones)
- ✔ solution to the synchronisation issue

but also

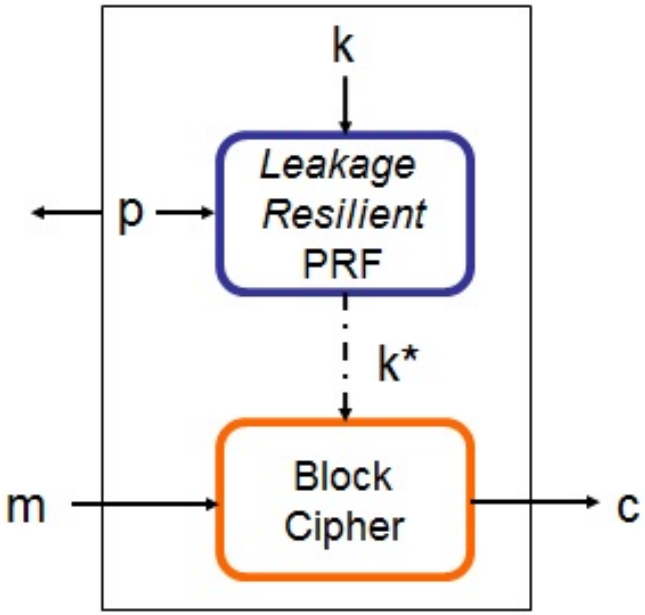
- ✔ limited use of each secret key
- ✔ proof of leakage-resilience for the whole encryption scheme

Leakage-Resilient Cryptography

- Leakage-Resilient Cryptography Model
  - only computation leaks
  - bounded amount of leakage per invocation
  - unlimited number of invocations
- Leakage-Resilient Encryption Scheme
  - challenge and leakage oracles
  - ciphertext indistinguishable from the encryption of a random string of the plaintext's size

Scheme 1: Symmetric Encryption from a LR PRF

- Re-keying Primitive
  - leakage-resilient PRF
  - non-adaptive leakage functions
  - non-adaptive inputs
- Block Cipher
  - as a PRF
  - not leakage-resilient



Theorem 1: This encryption scheme is a *non-adaptive leakage-resilient encryption scheme*.

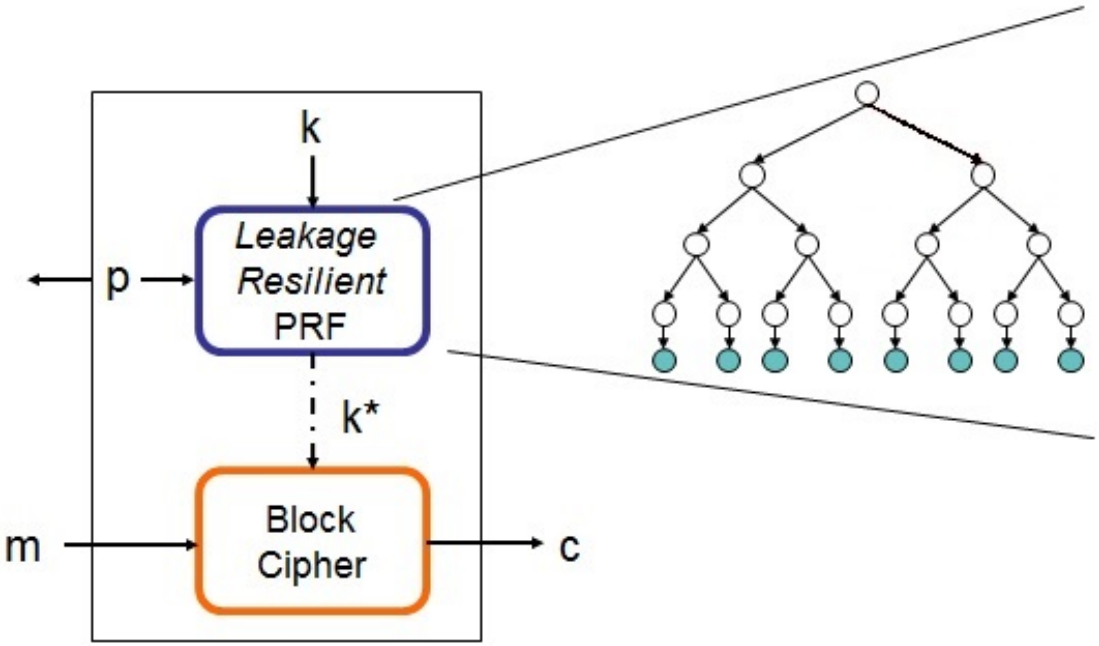
Scheme 1 instantiated with the CHES'12 PRF (1/2)

- instantiated with the Faust-Pietrzak-Schipper naLR naPRF

S. Faust, K. Pietrzak, J. Schipper: Practical Leakage-Resilient Symmetric Cryptography. CHES 2012

- inspired by the Goldreich-Goldwasser-Micali tree

O. Goldreich, S. Goldwasser, S. Micali: How to construct random functions. J. ACM 33(4) (1986)



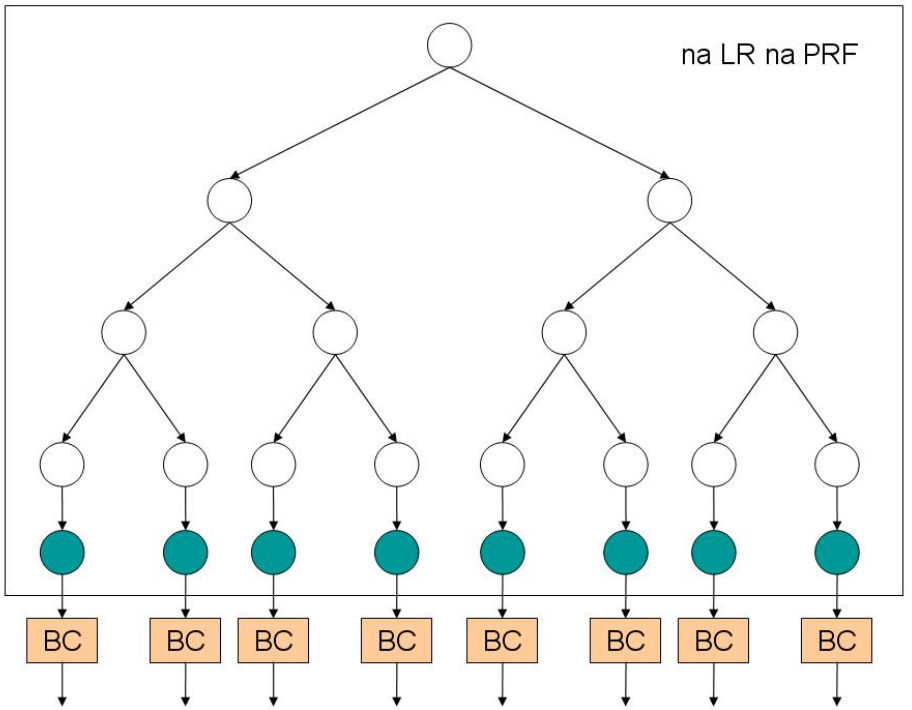
Scheme 1 instantiated with the CHES'12 PRF (2/2)

LR Encryption Scheme from

- ✔ naLR naPRF as re-keying scheme
- ✔ a SPA resistant block cipher

but

- ✘ not optimal
- ✘ no solution for the re-synchronization



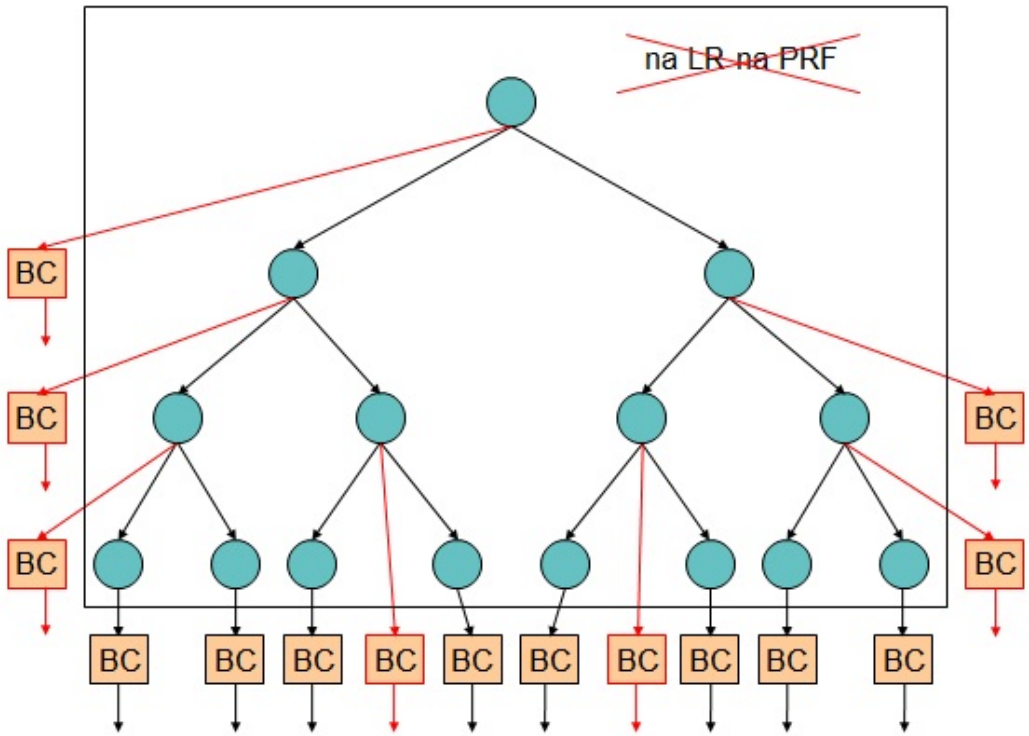
Scheme 2: Symmetric Encryption from a weak PRF

LR Encryption Scheme from

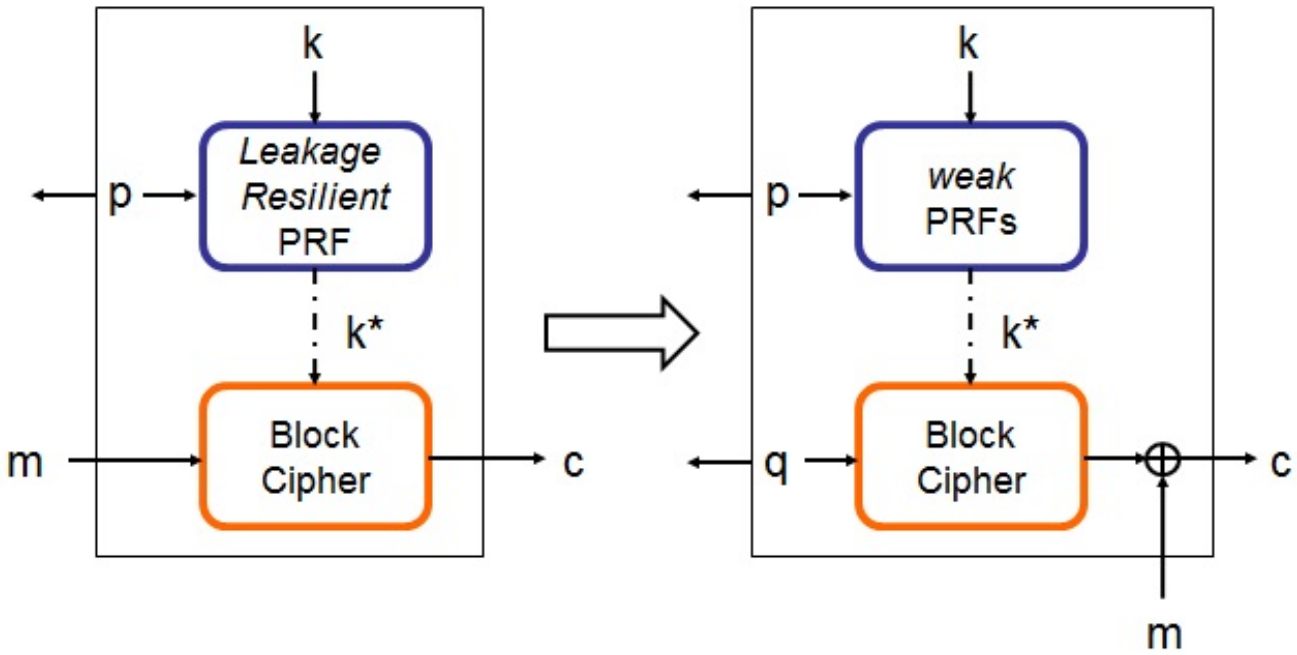
- ✔ only weak PRFs for the re-keying
- ✔ a SPA resistant block cipher
- ✔ more efficient
- ✔ with a solution for the re-synchronization

but

- ✘ additional constraint on the message

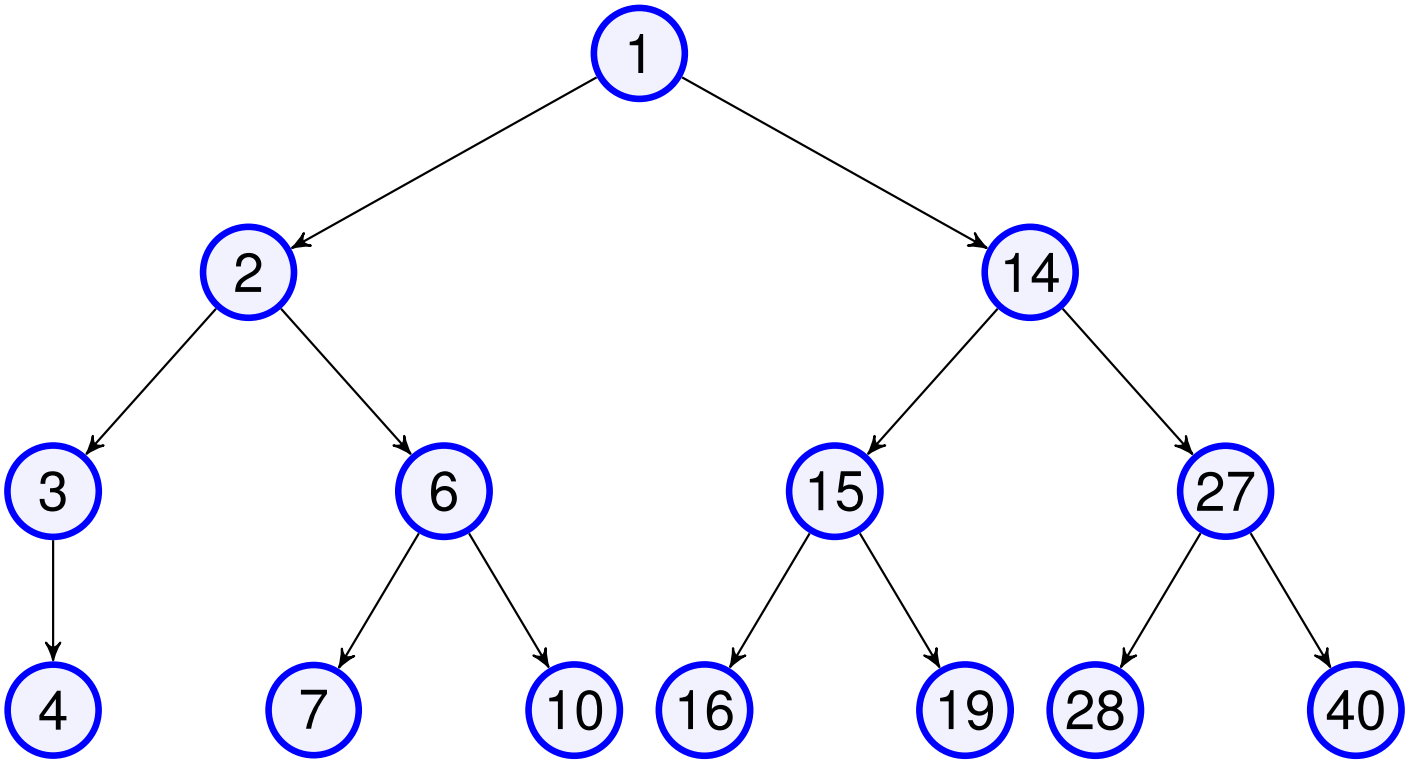


Security Aspects



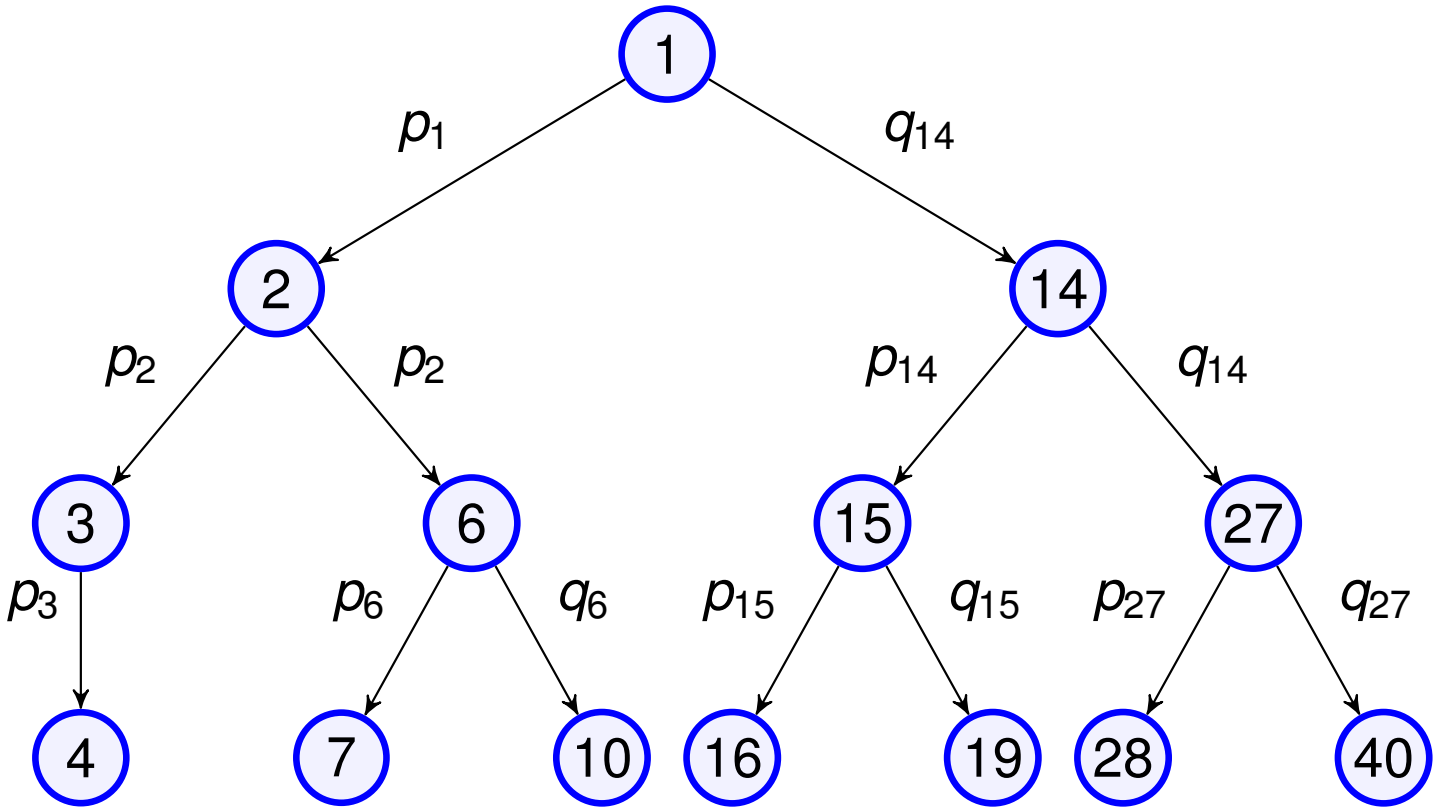
- block cipher with **random inputs**
- **same primitive** for the block cipher and the weak PRFs
- plaintext **before** or **after** the block cipher

Synchronization



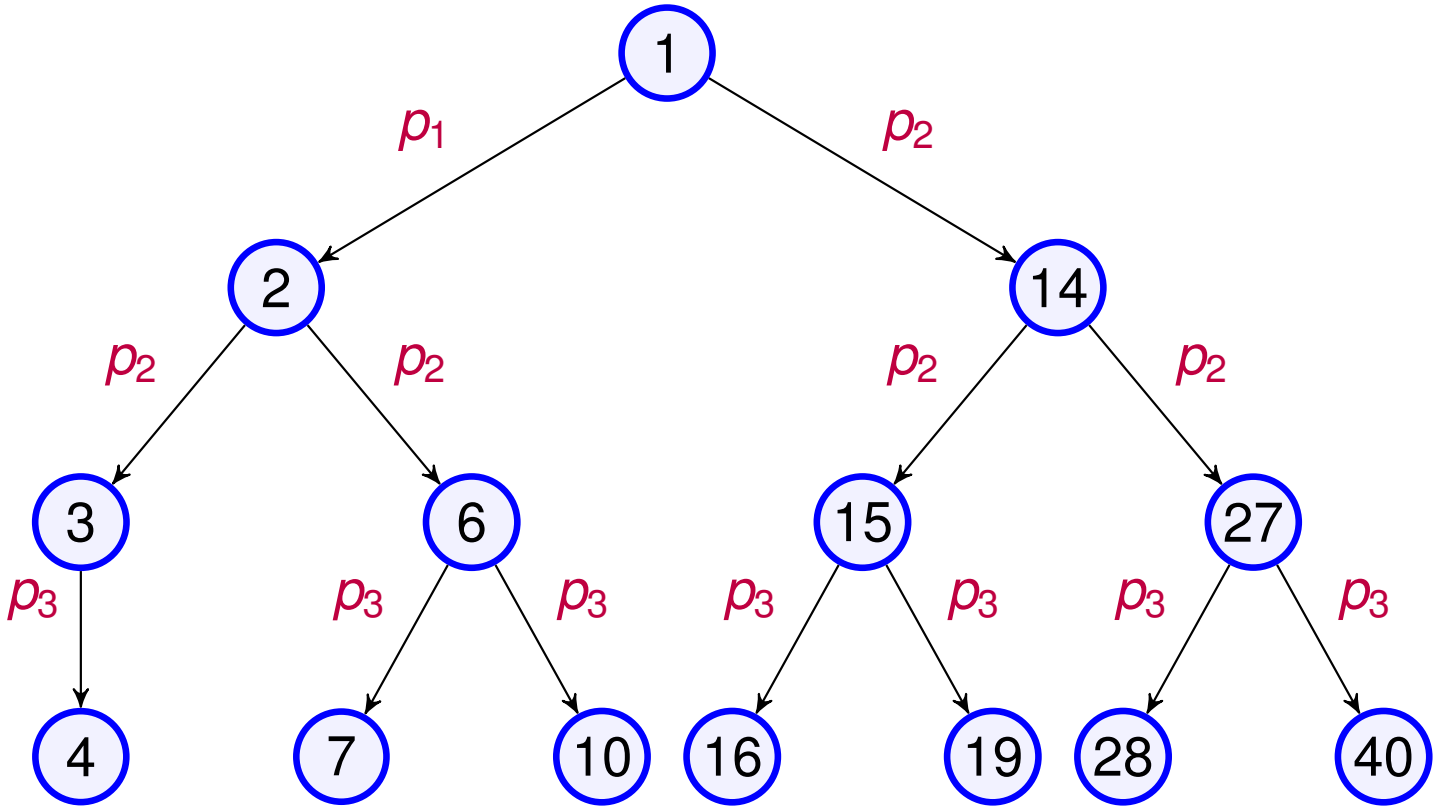
- short-cuts
- no additional relations between the secret keys

Random Value Generation: Naive Solution



Naive Solution: one fresh random value per derivation

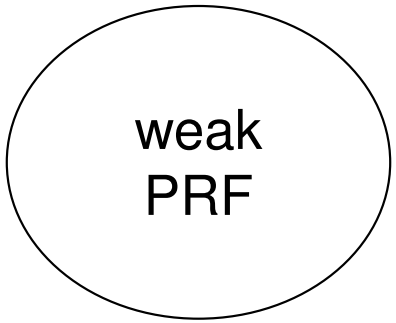
Random Value Generation: Improvement



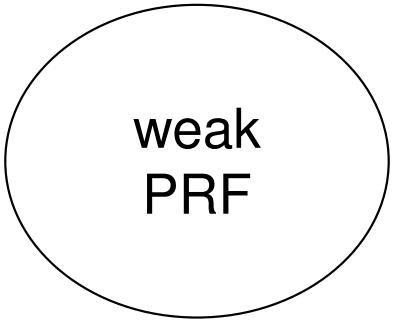
[FPS12]: one fresh random value per tree layer

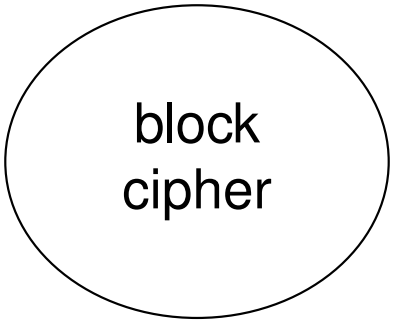
[YS13]: random values generated by a PRG **G**

Instantiation

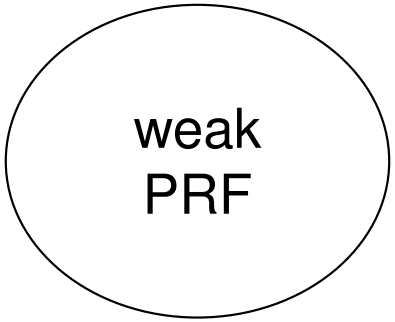


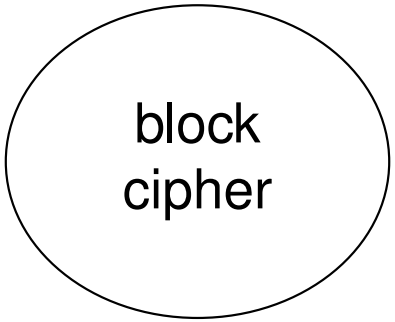
Instantiation

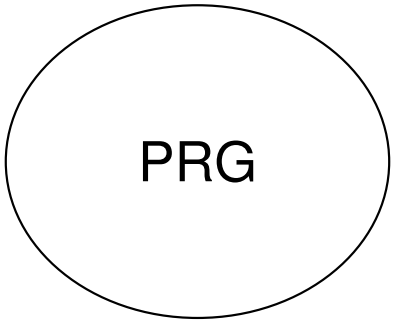




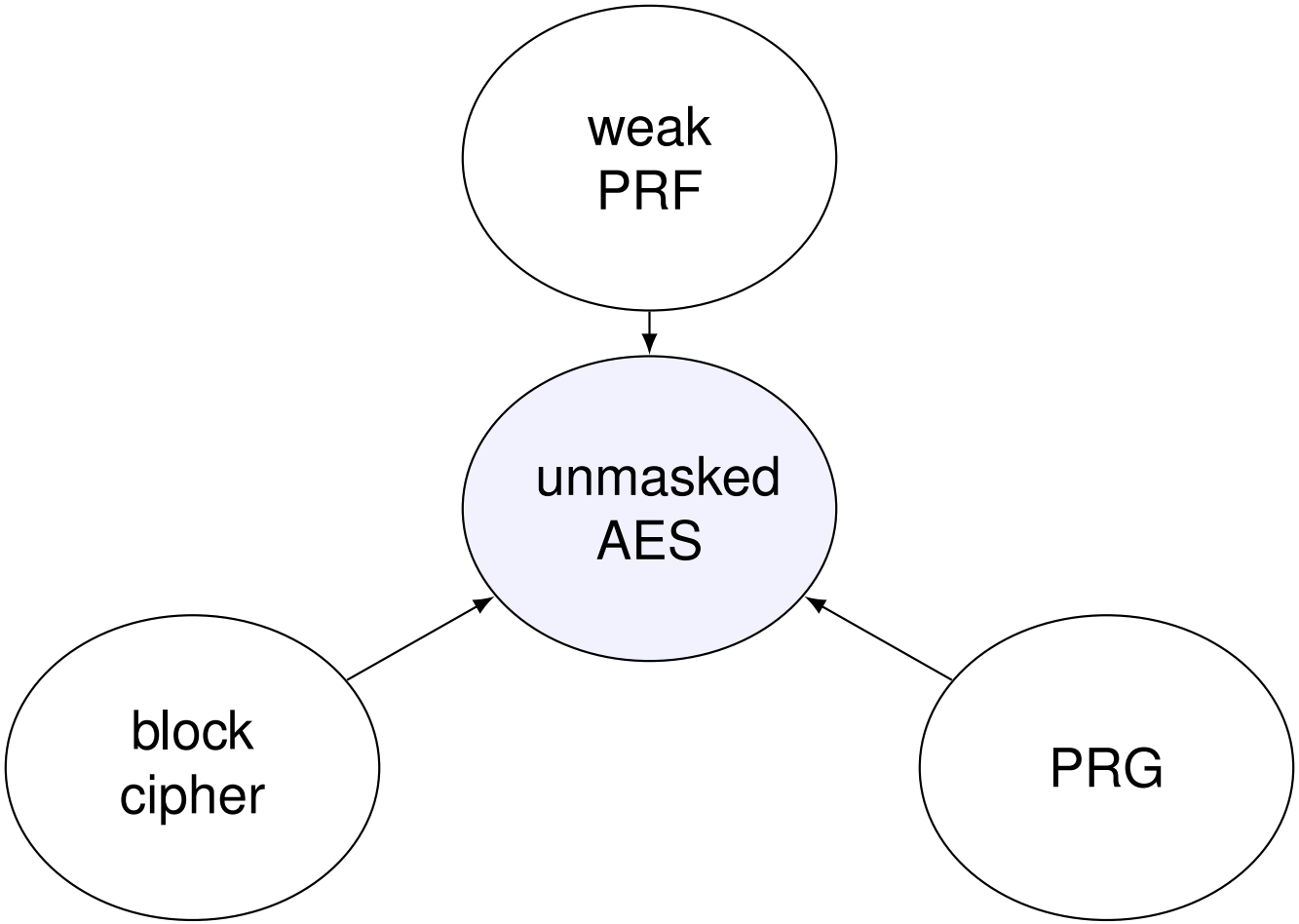
Instantiation





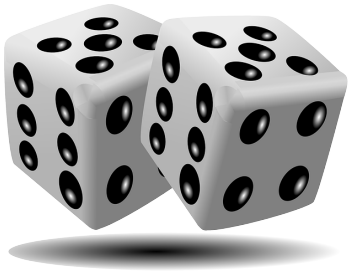


Instantiation



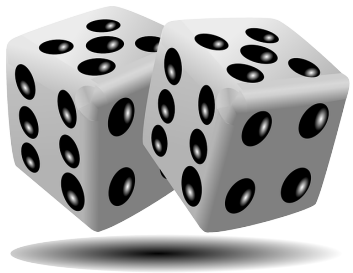
Outline

Pseudo-Random Generators



robust PRNG with input: *Dodis et al. CCS 2013* ✔

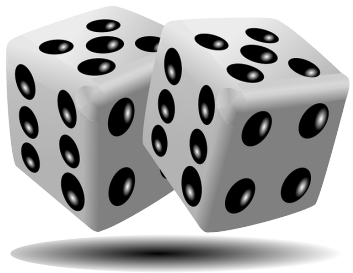
Pseudo-Random Generators



robust PRNG with input: *Dodis et al. CCS 2013* ✔

leakage-resilient secure PRNG: *Yu et al. CCS 2010* ✔

Pseudo-Random Generators



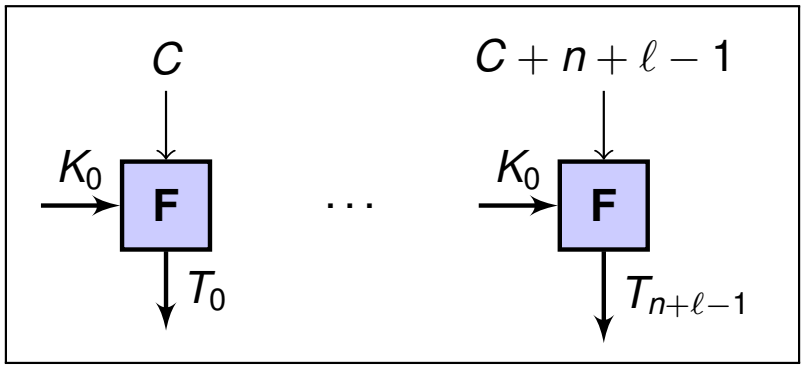
robust PRNG with input: *Dodis et al. CCS 2013* ✔

leakage-resilient secure PRNG: *Yu et al. CCS 2010* ✔

leakage-resilient robust PRNG with input: ✘

PRNG with Input from CCS 2013

- **setup**() outputs **seed** $= (X, X') \leftarrow \{0,1\}^{2n}$;
- $S = \mathbf{refresh}(S, I; X) = S \cdot X + I$,
  where all operations are over $\mathbb{F}_{2^n}$;
- $(S, R) = \mathbf{next}(S; X') = \mathbf{G}(U)$,
  where $U = [X' \cdot S]_1^m$ is the truncation of the product $(X'S)$.
  $\mathbf{G} : \{0,1\}^m \rightarrow \{0,1\}^{n+\ell}$ is a (t, ε)-secure PRG.



Security Properties

Attacker $\mathcal{A}$ Capabilities

- ask for outputs (S, R)

Security Properties

Attacker $\mathcal{A}$ Capabilities

- ask for outputs (S, R)
- compromise the inputs I

Security Properties

Attacker $\mathcal{A}$ Capabilities

- ask for outputs (S, R)
- compromise the inputs I
- compromise the internal state S by setting or getting it

Security Properties

Attacker $\mathcal{A}$ Capabilities

- ask for outputs (S, R)
- compromise the inputs I
- compromise the internal state S by setting or getting it

Robustness

$\mathcal{A}$ cannot distinguish (S, R) from a uniformly random string with a significant advantage.

New Security Properties

Attacker $\mathcal{A}^{\mathcal{L}}$ Capabilities

- ask for outputs (S, R)
- compromise the inputs I
- compromise the internal state S by setting or getting it

New Security Properties

Attacker $\mathcal{A}^{\mathcal{L}}$ Capabilities

- ask for outputs (S, R)
- compromise the inputs I
- compromise the internal state S by setting or getting it
- collect the leakage: λ bits of information on the manipulated data at each invocation

New Security Properties

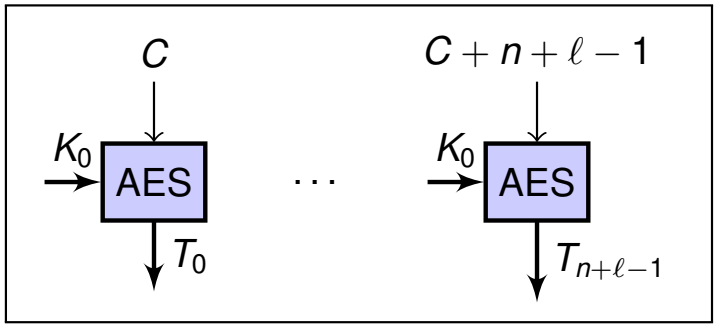
Attacker $\mathcal{A}^{\mathcal{L}}$ Capabilities

- ask for outputs (S, R)
- compromise the inputs I
- compromise the internal state S by setting or getting it
- collect the leakage: λ bits of information on the manipulated data at each invocation

Robustness with Leakage

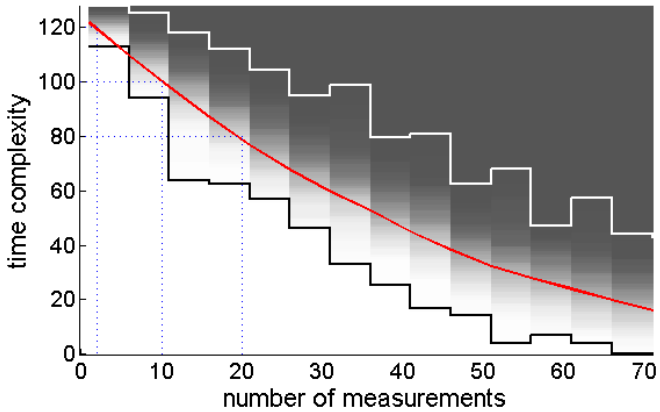
$\mathcal{A}^{\mathcal{L}}$ cannot distinguish (S, R) from a uniformly random string with a significant advantage.

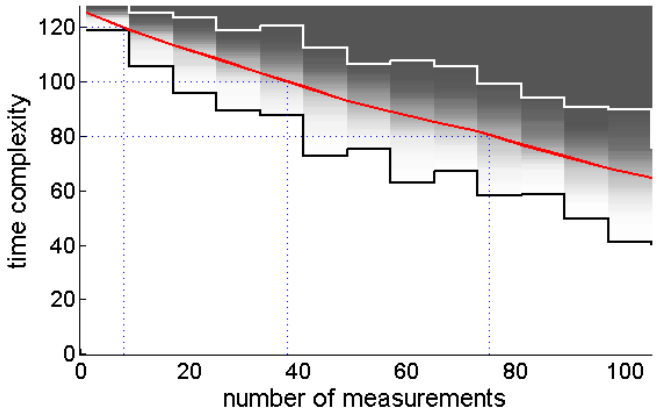
Limitations in Presence of Leakage: Generator **G**



Issue unbounded number of encryptions with the same key

Attack *Differential Power Analysis*





Attack Simulation on software (left) and hardware (right) with $f_{\text{AES}} = HW$

Limitations in Presence of Leakage: Function **next**

1. Choose of a leakage function for $U = [X'S]_m$

$$f_{\mathsf{next},\Pi}(S, X') = \left[\mathrm{AES}\left[X' \left(\mathrm{AES}_{[X'S]_1^m}(C_0) || \ldots || \mathrm{AES}_{[X'S]_1^m}(C_0 + \lceil \tfrac{n}{m} \rceil - 1) \right) \right]_1^m \left(C_0 + \left\lceil \frac{n+\ell}{m} \right\rceil \right) \right]_1^\lambda$$

2. Compromise of the state: $C \leftarrow C_0$
3. Refresh the random part S of the state
4. Collect the leakage during a **next**
5. Ask for a challenge: get $(S, R) = \mathbf{next}(S, X')$ if $b = 0$ or (S, R) uniformly random if $b = 1$

$$\begin{array}{lll} \text{return} & 0 & \text{if } [S]_1^\lambda = L \\ \text{return} & 1 & \text{otherwise} \end{array}$$

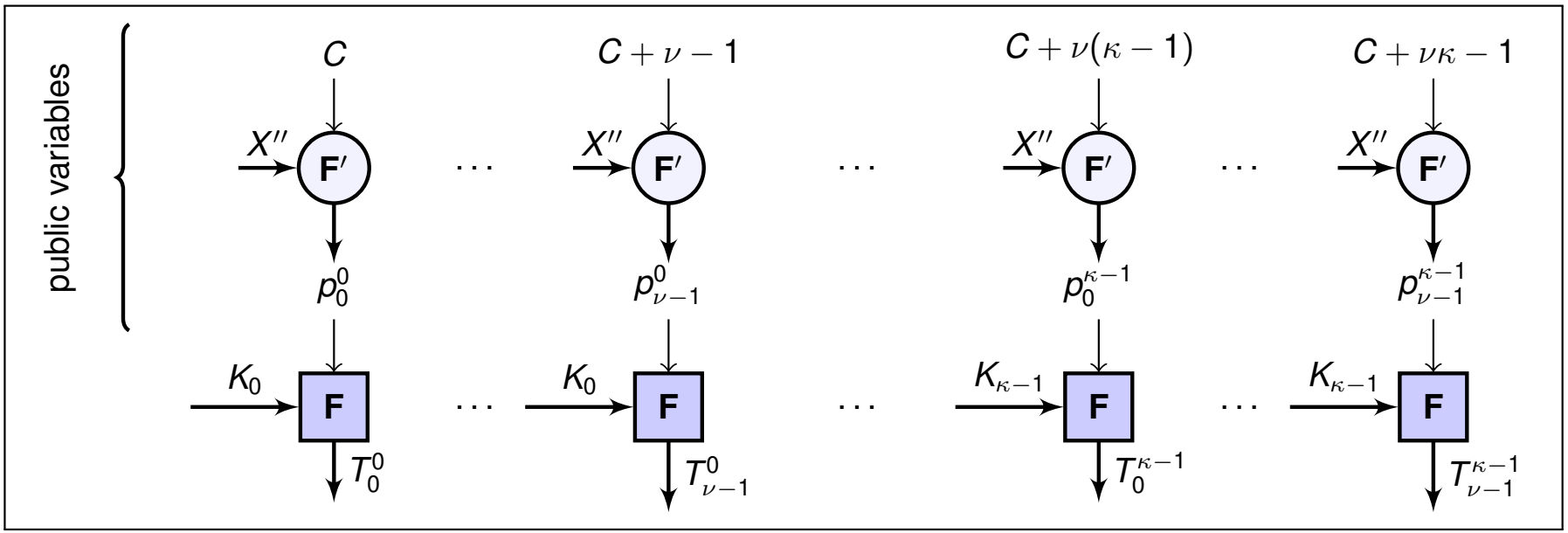
New Generic Construction

- **setup**() outputs **seed** $= (X, X', X'') \leftarrow \{0,1\}^{3n}$;
- $S = \mathbf{refresh}(S, I; X) = S \cdot X + I$,
  where all operations are over $\mathbb{F}_{2^n}$;
- $(S, R) = \mathbf{next}(S; X', X'') = \mathbf{G}(U)$,
  where $U = [X'S]_1^m$ is the truncation of the product $(X'S)$.

New security property for PRG **G**:

G : $\{0,1\}^m \rightarrow \{0,1\}^{n+\ell}$ is a (α, λ)-leakage-resilient and (t, ε)-secure PRG.

Leakage-Resilient Instantiation



$$(S, R) = (T_0^0, \ldots, T_{\nu-1}^{\kappa-1}) \leftarrow \mathbf{G}(K_0 || \ldots || K_{\kappa-1} || C)$$

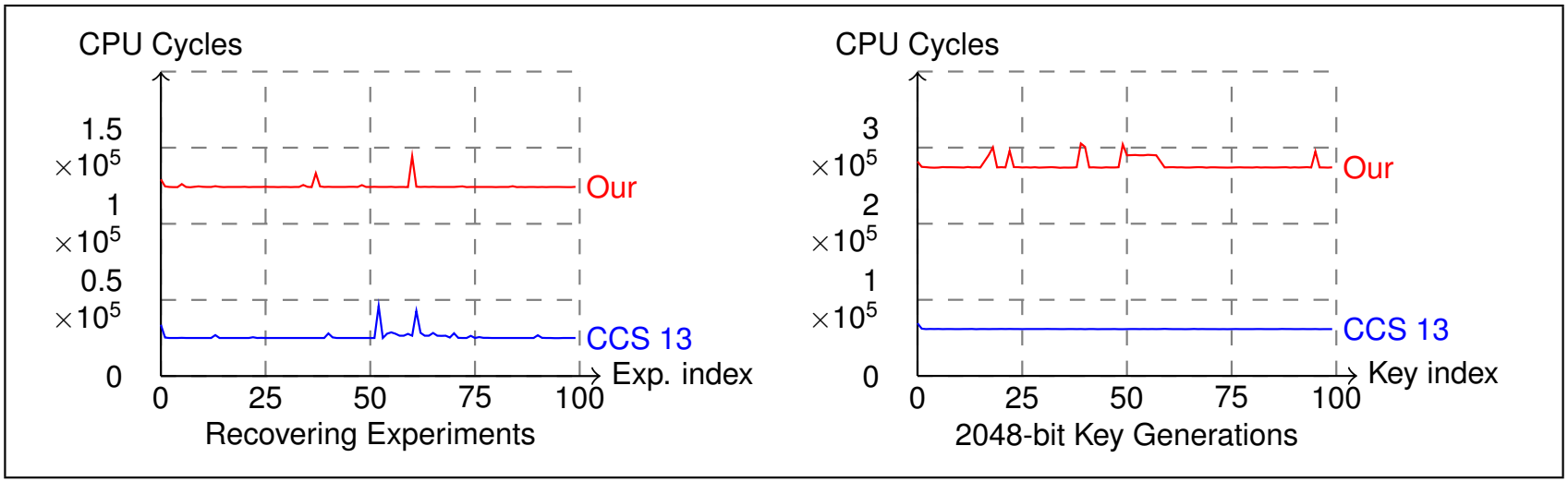
Benchmarks

CCS 13:

- internal state S: 489 bits
- threshold: $\gamma^* = 449$
- AES: 5 calls with 1 secret key

Our Construction:

- internal state S: 1408 bits
- threshold: $\gamma^* = 1370$
- AES: 12 calls with 6 secret keys (2 calls per secret key)



Outline

Conclusion

- Summary
  - ⋆ comparison between **masking** and **leakage-resilient primitives**
  - ⋆ **leakage-resilient** and **efficient** symmetric encryption
  - ⋆ **leakage-resilient** and **efficient** PRNG with input
- Further Work
  - ⋆ **more efficient** encryption schemes
  - ⋆ leakage-resilient encryption using **modes of operation**

Thank you