Leakage-Resilient Primitives using Re-keying

Journées Codage et Cryptographie 2014

Sonia Belaïd[1,2]

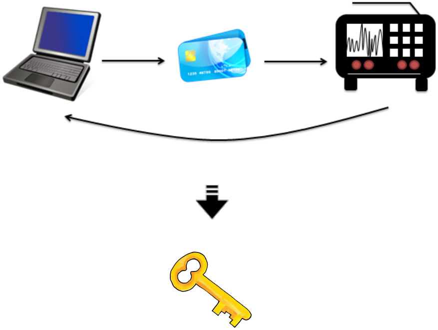
École Normale Supérieure, 45 rue d'Ulm 75005 Paris

Thales Communications & Security

Sonia.Belaid@ens.fr

Side-Channel Attacks

- physical leakage
  - timing
  - power consumption
  - temperature
  - ...
- statistical treatment
- key recovery



Countermeasures against Side-Channel Attacks

Masking

sensitive values randomized:

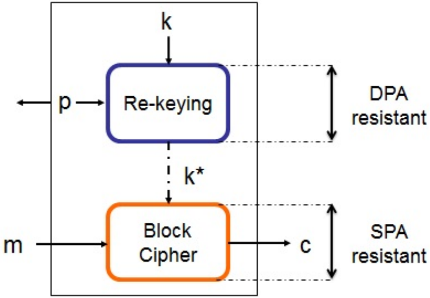
x replaced by $(x_m, m_0, \ldots, m_{d-1})$

$$x = x_m \star m_0 \star \cdots \star m_{d-1}$$

Drawbacks of Masking

- ✘ higher-order attacks
- ✘ performances

Re-keying



Countermeasures against Side-Channel Attacks

Masking

sensitive values randomized:

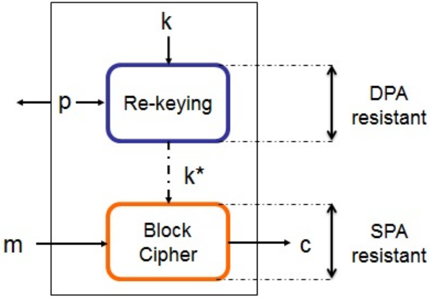
x replaced by $(x_m, m_0, \ldots, m_{d-1})$

$$x = x_m \star m_0 \star \cdots \star m_{d-1}$$

Drawbacks of Masking

- ✗ higher-order attacks
- ✗ performances

Re-keying



Goal

Goal:

- build leakage-resilient primitives
- practical for use in constrained devices

Leakage-Resilient Cryptography Model

- only computation leaks
- bounded amount of leakage per invocation
- unlimited number of invocations

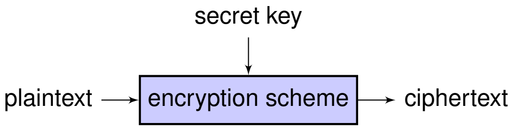
Practical constraints:

- limited code size
- limited storage
- reasonable execution time

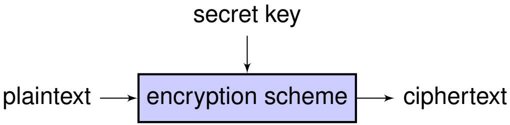
Outline

Outline

Encryption Scheme



Encryption Scheme

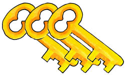


Goal: efficient and leakage-resilient encryption scheme

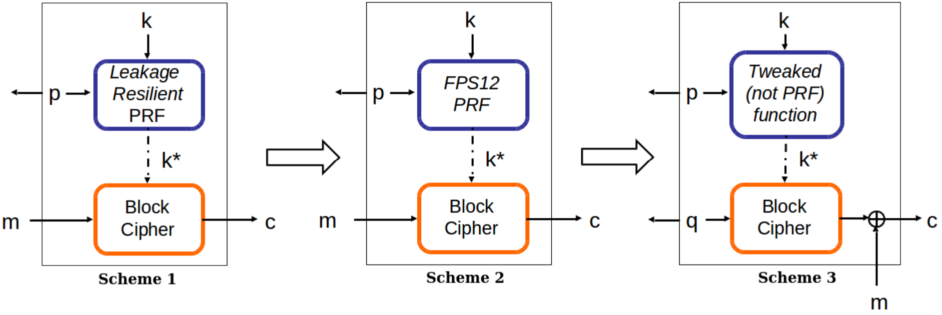
Related Work: Kocher's patent 1999 but

- multiple use of the same key
- no security proof

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



Contributions



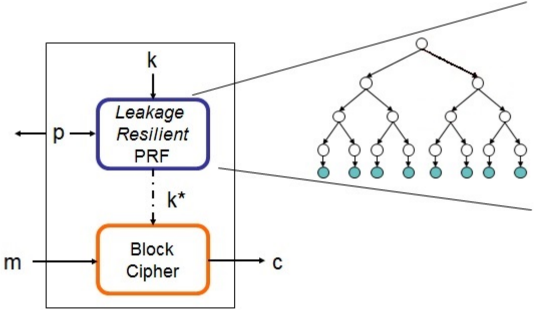
Scheme 1 LR encryption scheme using a LR PRF

Scheme 2 Instantiation of Scheme 1 with the [FPS12] LR PRF

Scheme 3 more efficient and still LR encryption scheme with a tweaked (not LR and not PRF) function

Schemes 1 and 2

- Re-keying Primitive
  - leakage-resilient PRF
- Block Cipher
  - as a PRF
  - not leakage-resilient

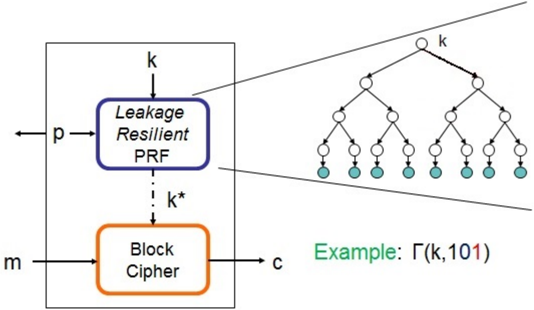


- instantiated with the [FPS12] LR PRF

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

Schemes 1 and 2

- Re-keying Primitive
  - leakage-resilient PRF
- Block Cipher
  - as a PRF
  - not leakage-resilient

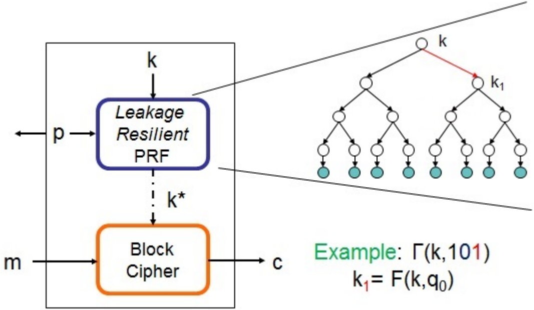


- instantiated with the [FPS12] LR PRF

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

Schemes 1 and 2

- Re-keying Primitive
  - leakage-resilient PRF
- Block Cipher
  - as a PRF
  - not leakage-resilient

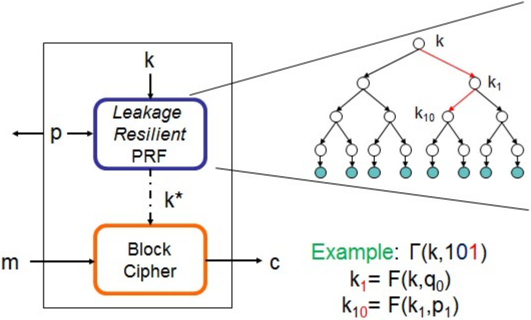


- instantiated with the [FPS12] LR PRF

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

Schemes 1 and 2

- Re-keying Primitive
  - leakage-resilient PRF
- Block Cipher
  - as a PRF
  - not leakage-resilient

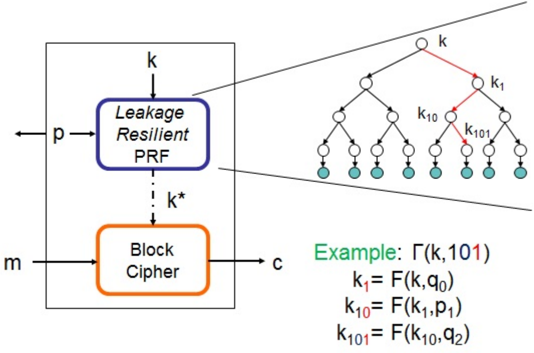


- instantiated with the [FPS12] LR PRF

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

Schemes 1 and 2

- Re-keying Primitive
  - leakage-resilient PRF
- Block Cipher
  - as a PRF
  - not leakage-resilient

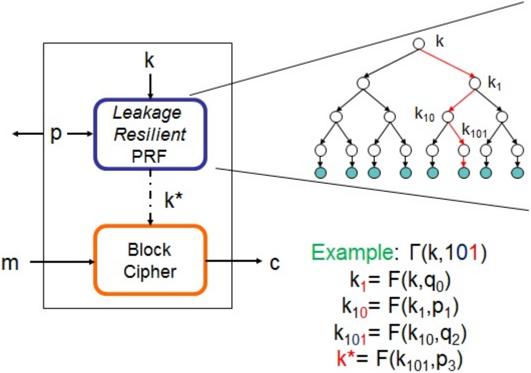


- instantiated with the [FPS12] LR PRF

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

Schemes 1 and 2

- Re-keying Primitive
  - leakage-resilient PRF
- Block Cipher
  - as a PRF
  - not leakage-resilient



- instantiated with the [FPS12] LR PRF

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

Scheme 3: More Efficient LR Encryption Scheme

LR Encryption Scheme from

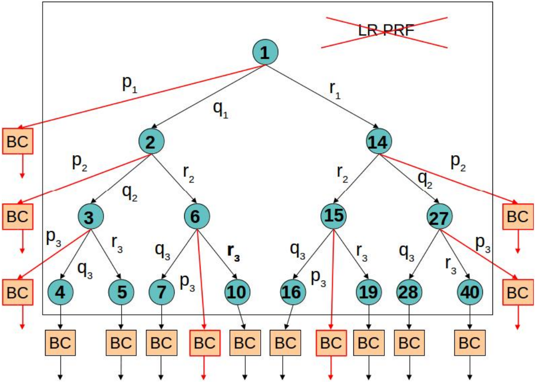
- ✔ re-keying function (not LR, not PRF)
- ✔ block cipher

with

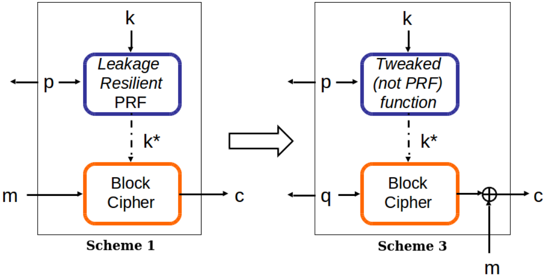
- ✔ short-cuts between keys
- ✔ uniformly random values:
  - one triplet per level [FPS12],
  - PRG with public seed [YS13]

but

- ✘ additional constraint on the message

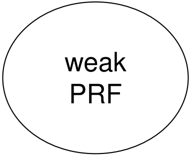


Security Aspects

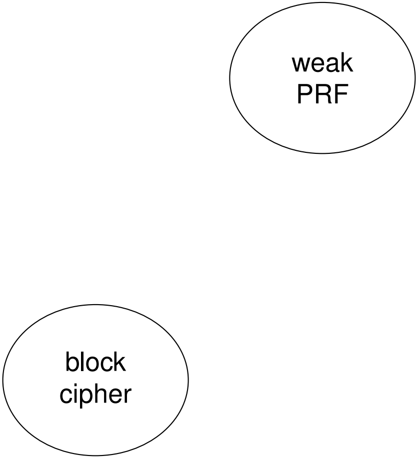


- block cipher with **random inputs** → plaintext added to the output
- **same primitive** for the block cipher and the weak PRFs
- secret keys used at most **three times** :
  → avoid the recomputation of previous keys

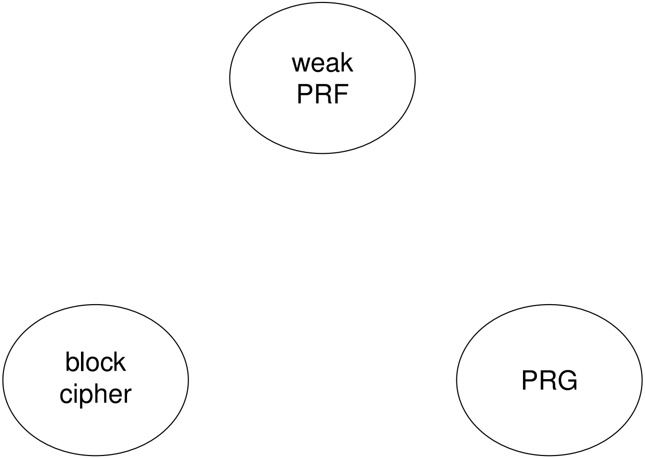
Instantiation



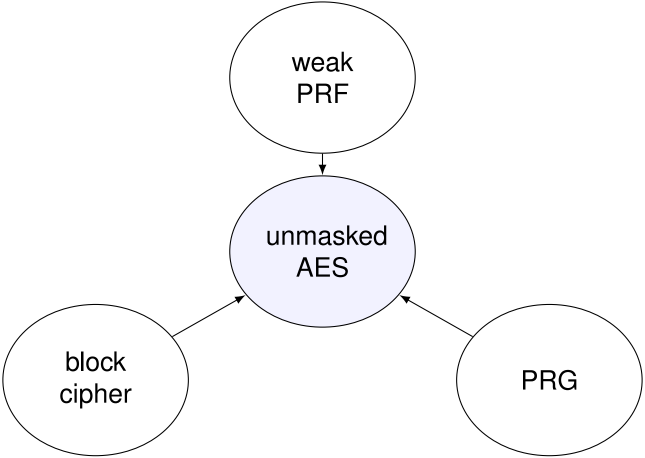
Instantiation



Instantiation

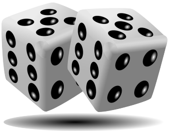


Instantiation

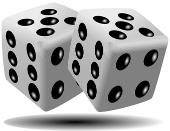


Outline

Pseudo-Random Generators



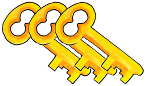
Pseudo-Random Generators



Goal: efficient and leakage-resilient PRNG

Related Works: PRNG

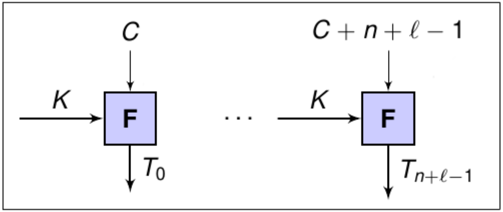
- robust with input: *Dodis et al. CCS'13*
- leakage-resilient: *Yu et al. CCS'10*



Contribution: Leakage-Resilient PRNG with Input
Michel Abdalla, Sonia Belaïd, David Pointcheval, Sylvain Ruhault, Damien Vergnaud

PRNG with Input from CCS 2013

- **setup**() outputs **seed** $= (X, X')$;
- $S =$ **refresh**$(S, I; X) = S \cdot X + I$;
- $(S, R) =$ **next**$(S; X') = \mathbf{G}([X'S]_1^m)$.



$\mathbf{G} : \{0,1\}^m \to \{0,1\}^{n+\ell}$ is a secure PRG ($K \leftarrow [X'S]_1^m$).

Security Properties

Attacker $\mathcal{A}$ Capabilities

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

Security Properties

Attacker $\mathcal{A}$ Capabilities

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

Robustness

If enough entropy is provided, $\mathcal{A}$ 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

New Security Properties

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

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

New Security Properties

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

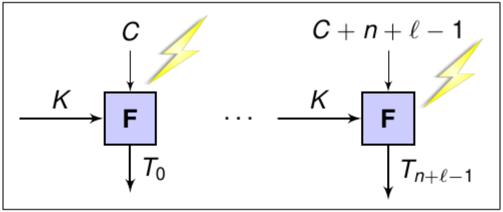
- ask for outputs (S, R)
- compromise the inputs I
- compromise the internal state S
- collect the leakage: λ bits of information on the manipulated data at each invocation

Robustness with Leakage

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

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

- **setup**() outputs **seed** $= (X, X')$;
- $S =$ **refresh**$(S, I; X) = S \cdot X + I$;
- $(S, R) =$ **next**$(S; X') = \mathbf{G}([X'S]_1^m)$.



$\rightarrow$ Diffential Power Analysis of **G**

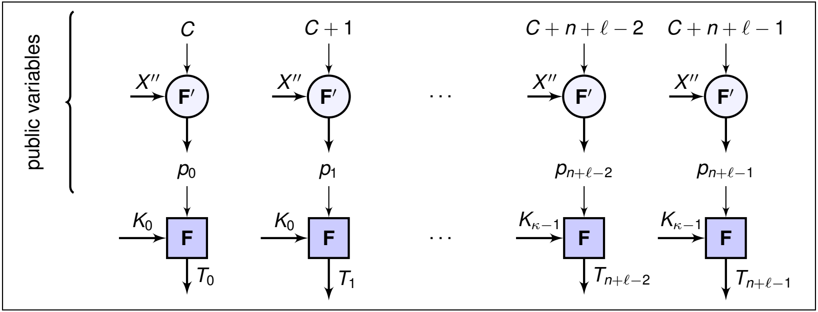
New Generic Construction

- **setup**() outputs **seed** $= (X, X', X'')$;
- $S =$ **refresh**$(S, I; X) = S \cdot X + I$;
- $(S, R) =$ **next**$(S; X', X'') =$ **G**$([X'S]_1^m, X'')$.

New security property for PRG G

G is a leakage-resilient and secure PRG.

Leakage-Resilient Instantiation



$$(K_0||\ldots||K_{\kappa-1}||C) \leftarrow [X'S]_1^m$$

m is $(\kappa+1)$ times larger than in CCS'13

Practical Analysis

CCS'13 (2^{-40} security):

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

Our Construction (2^{-40} security):

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

- Efficiency: about **five times** slower than CCS 13
- Security: higher security levels can be achieved with a tweaked instantiation

Outline

1. Leakage-Resilient Encryption Scheme
2. Leakage-Resilient PRNG with Input
3. Conclusion

Conclusion

- Summary
  - ⋆ **leakage-resilient** and **efficient** symmetric encryption
  - ⋆ **leakage-resilient** and **efficient** PRNG with input
- Further Work
  - ⋆ **more efficient** leakage-resilient primitives
  - ⋆ leakage-resilience evaluation of different **modes of operation**

Thank you