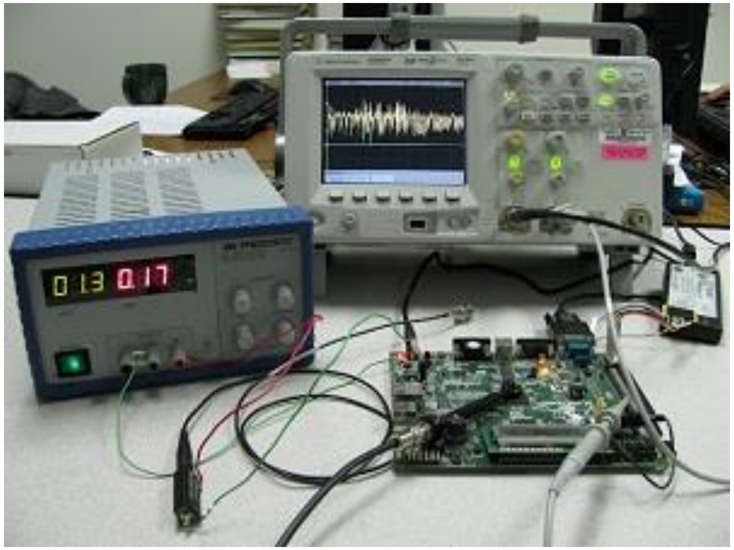
Leakage-Resilient (Symmetric) Cryptography



François-Xavier Standaert

UCL Crypto Group, Belgium

Summer school on real-world crypto, 2016

Outline

- Starting point (link with previous lecture)
- Seed results (TCC 2004, FOCS 2008)
- Primitives (PRGs/PRFs,PRPs)
  - If you don't care about proofs
    - The stateful/stateless separation
  - The proof/assumptions challenge
    - Ensuring independence
    - Bounding the leakage
    - The simulatable leakage attempt
- « Pragmatic » auth. & encryption (CCS 2015)
- Back to stateless primitives
- Conclusions & open problems

Outline

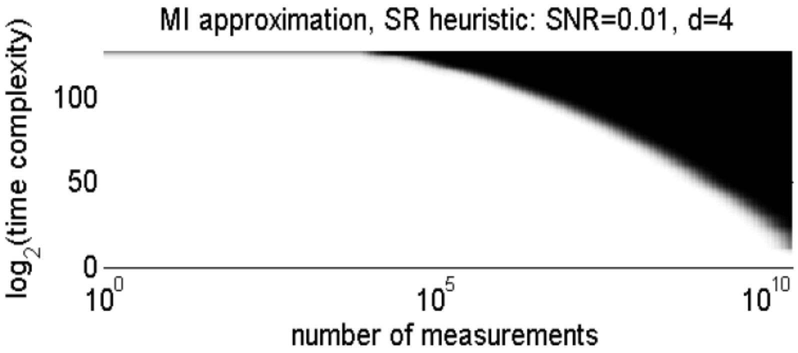
- **Starting point (link with previous lecture)**
- Seed results (TCC 2004, FOCS 2008)
- Primitives (PRGs/PRFs,PRPs)
  - If you don't care about proofs
    - The stateful/stateless separation
  - The proof/assumptions challenge
    - Ensuring independence
    - Bounding the leakage
    - The simulatable leakage attempt
- « Pragmatic » auth. & encryption (CCS 2015)
- Back to stateless primitives
- Conclusions & open problems

Masking

- Bound the information locally (i.e. on each share) and ensure independence (between the leakage of the shares) in order to obtain security globally (e.g. for AES implementations)

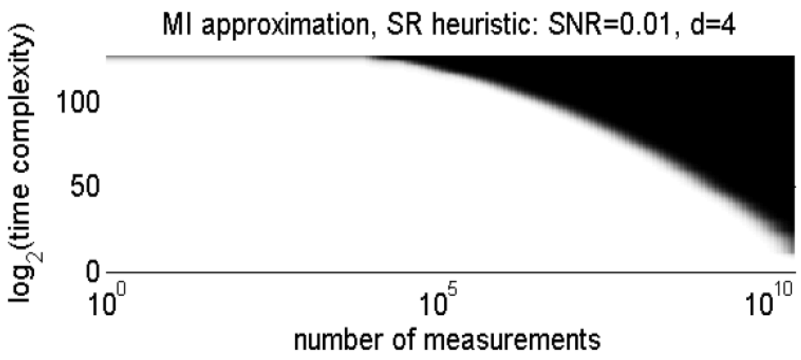
Masking

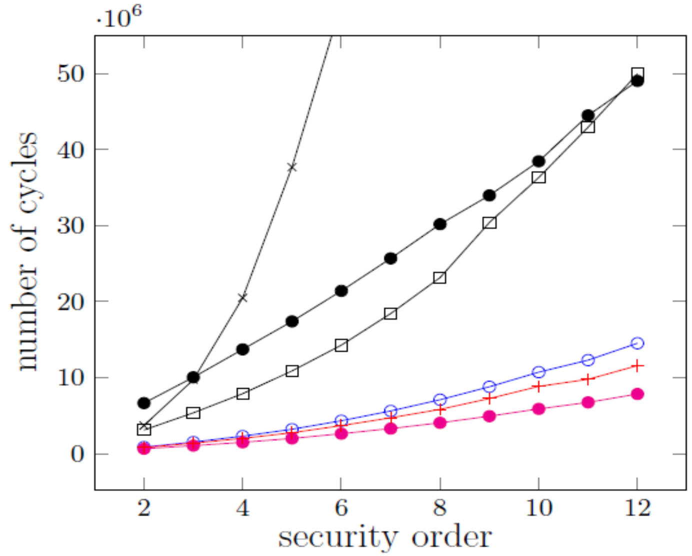
- Bound the information locally (i.e. on each share) and ensure independence (between the leakage of the shares) in order to obtain security globally (e.g. for AES implementations)
- Limitation: high security requires large # of shares



Masking

- Bound the information locally (i.e. on each share) and ensure independence (between the leakage of the shares) in order to obtain security globally (e.g. for AES implementations)
- Limitation: high security requires large # of shares ⇒ implies significant overheads





Leakage-resilience problem

- Concretely: can we *gain efficiency* by working at the block cipher level, i.e. bound the information (locally) for one execution, assume independence (for different executions) and gain security (globally) for many executions?

Leakage-resilience problem

- Concretely: can we *gain efficiency* by working at the block cipher level, i.e. bound the information (locally) for one execution, assume independence (for different executions) and gain security (globally) for many executions?

- Theoretically: can we prove the security of an implementation and what does it mean? (*How to reason generally about specific objects*?)

Outline

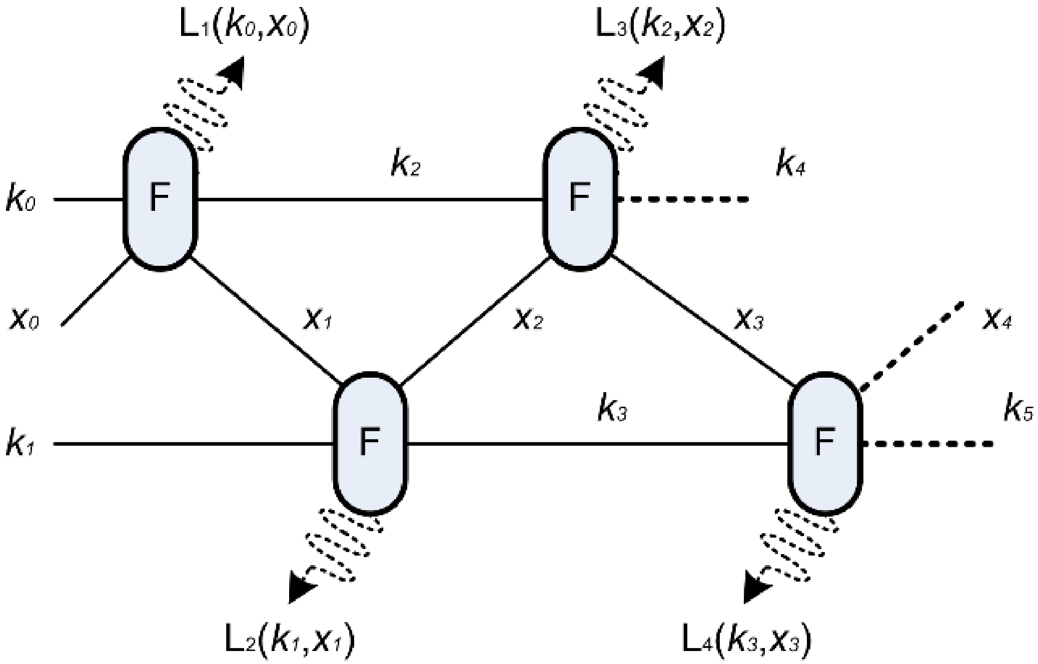
- Starting point (link with previous lecture)
- **Seed results (TCC 2004, FOCS 2008)**
- Primitives (PRGs/PRFs,PRPs)
  - If you don't care about proofs
    - The stateful/stateless separation
  - The proof/assumptions challenge
    - Ensuring independence
    - Bounding the leakage
    - The simulatable leakage attempt
- « Pragmatic » auth. & encryption (CCS 2015)
- Back to stateless primitives
- Conclusions & open problems

Micali & Reyzin 2004

- Physically observable cryptography
  - « Only computation leaks » assumption
    - Used in all following works
  - Indistinguishability ≠ unpredictability (with L)
    - Impact for encryption & authentication

Dziembowski & Pietrzak 2008

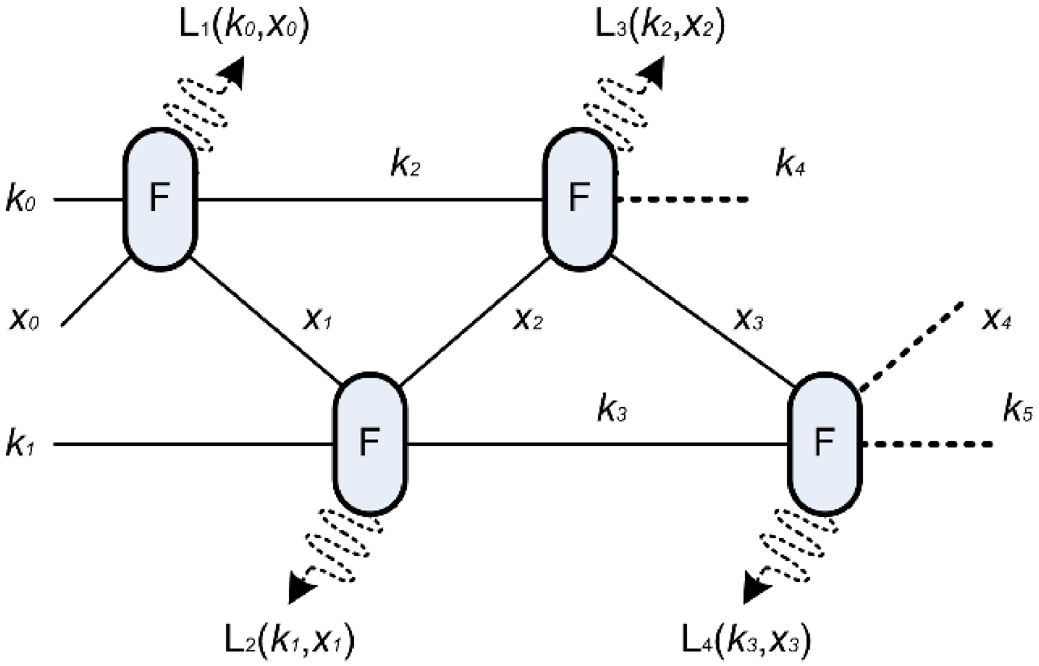
- Leakage-resilient cryptography

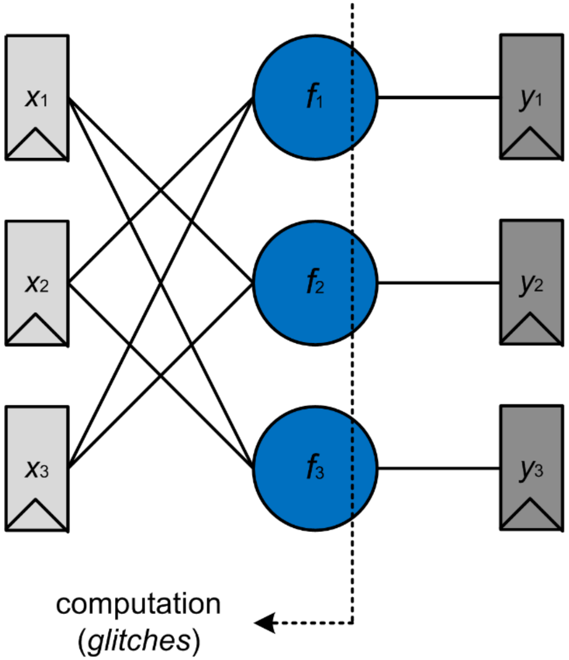


- Intriguing at first sight (alternating structure)

Dziembowski & Pietrzak 2008

- Leakage-resilient cryptography

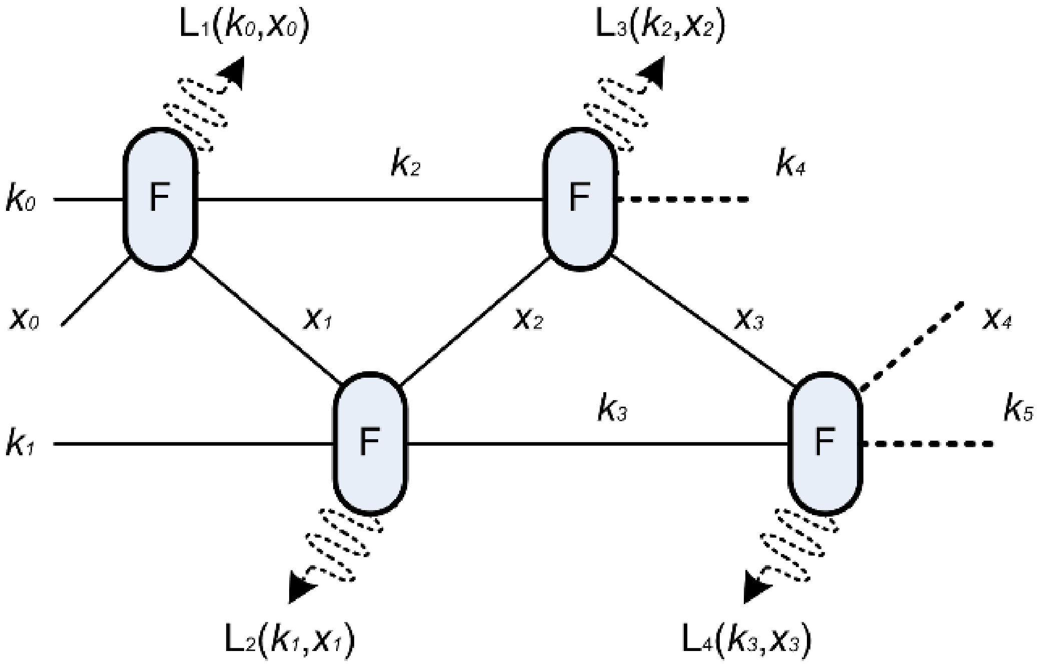


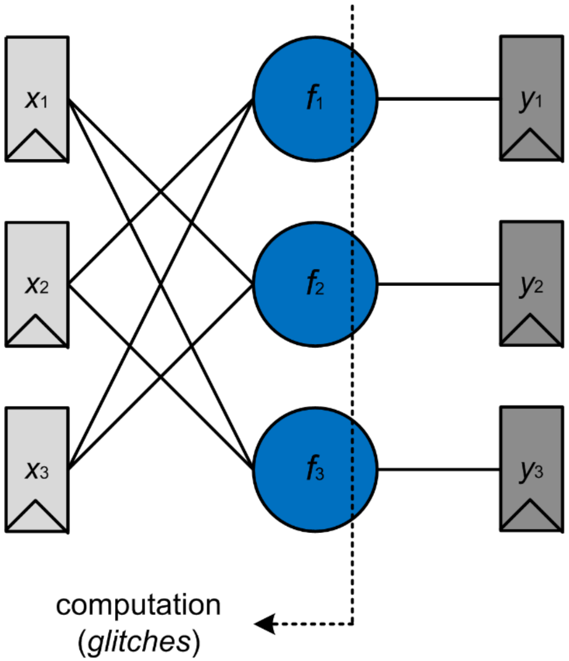


- Funnily similar to threshold implementations

Dziembowski & Pietrzak 2008

- Leakage-resilient cryptography





- Funnily similar to threshold implementations
  - Both exclude one input to gain independence

Outline

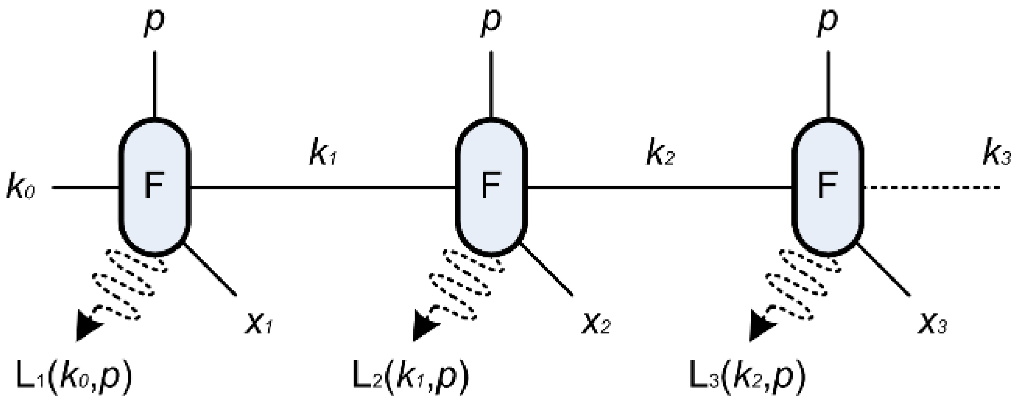
- Starting point (link with previous lecture)
- Seed results (TCC 2004, FOCS 2008)
- **Primitives (PRGs/PRFs,PRPs)**
  - If you don’t care about proofs
    - The stateful/stateless separation
  - The proof/assumptions challenge
    - Ensuring independence
    - Bounding the leakage
    - The simulatable leakage attempt
- « Pragmatic » auth. & encryption (CCS 2015)
- Back to stateless primitives
- Conclusions & open problems

Outline

- Starting point (link with previous lecture)
- Seed results (TCC 2004, FOCS 2008)
- **Primitives (PRGs/PRFs,PRPs)**
  - **If you don’t care about proofs**
    - **The stateful/stateless separation**
  - The proof/assumptions challenge
    - Ensuring independence
    - Bounding the leakage
    - The simulatable leakage attempt
- « Pragmatic » auth. & encryption (CCS 2015)
- Back to stateless primitives
- Conclusions & open problems

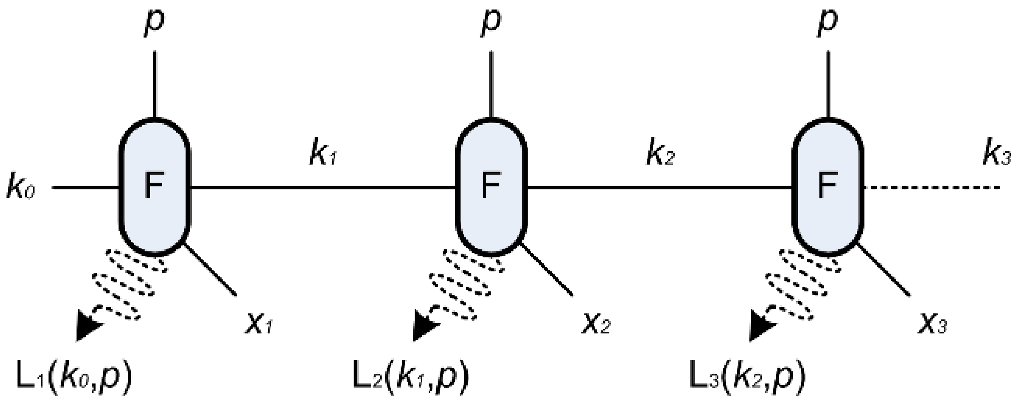
Stateful PRGs

- Most natural construction:
  - Forward-secure PRG [BY03]



Stateful PRGs

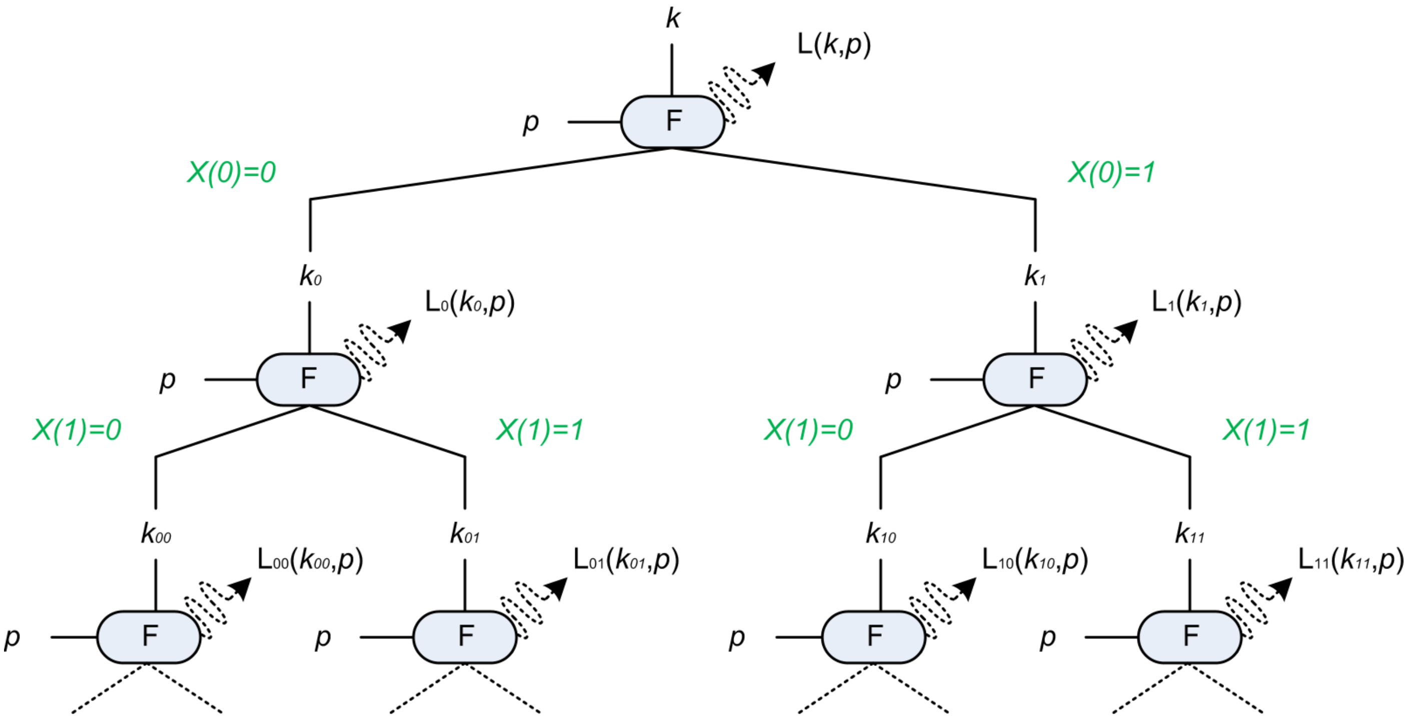
- Most natural construction:
  - Forward-secure PRG [BY03]



- Re-keying impact: bounds the number of (noisy) measurements per key (*prevents averaging*)

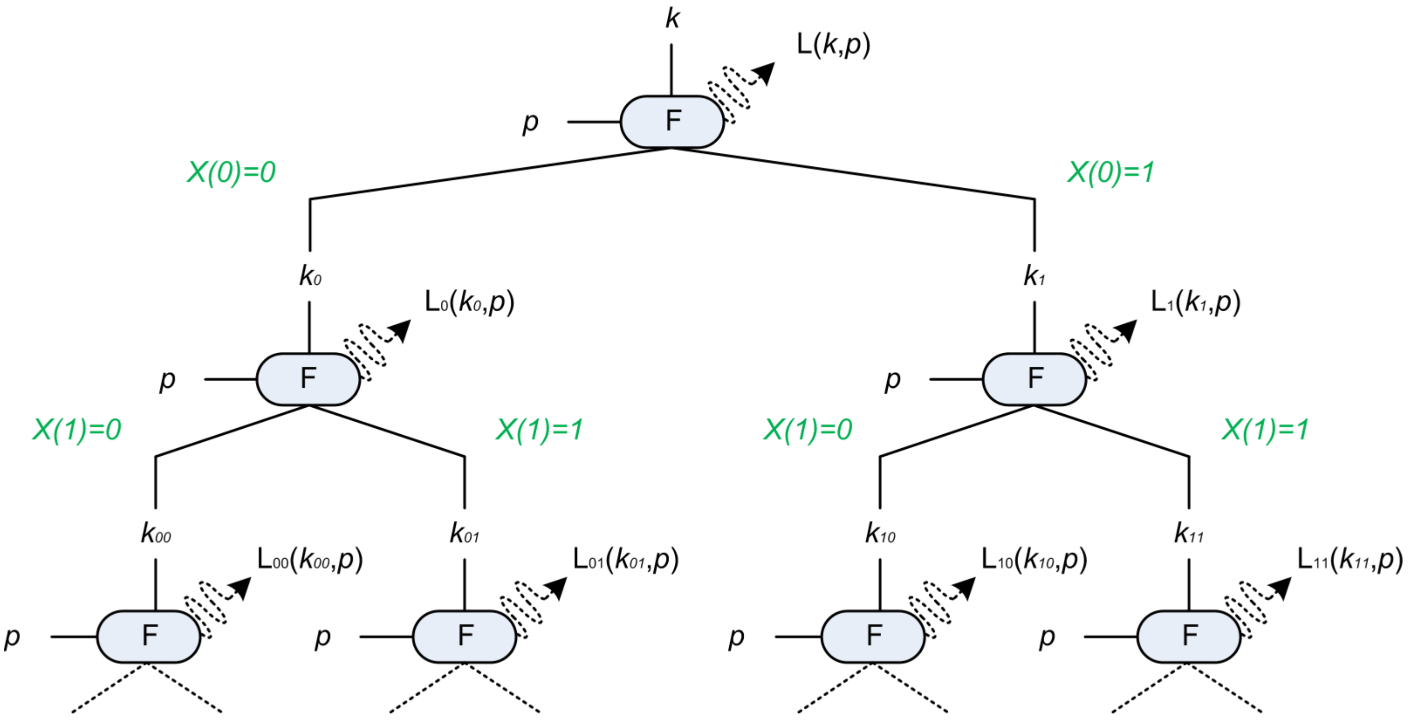
Stateless PRFs

- Most natural construction [GGM84]:



Stateless PRFs

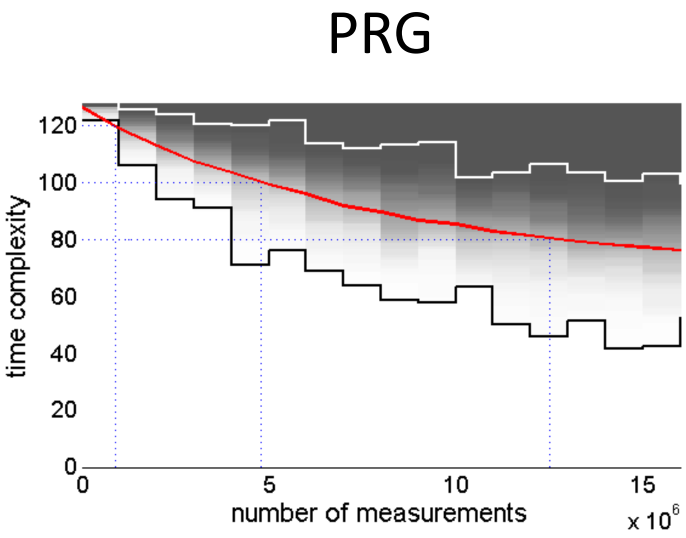
- Most natural construction [GGM84]:

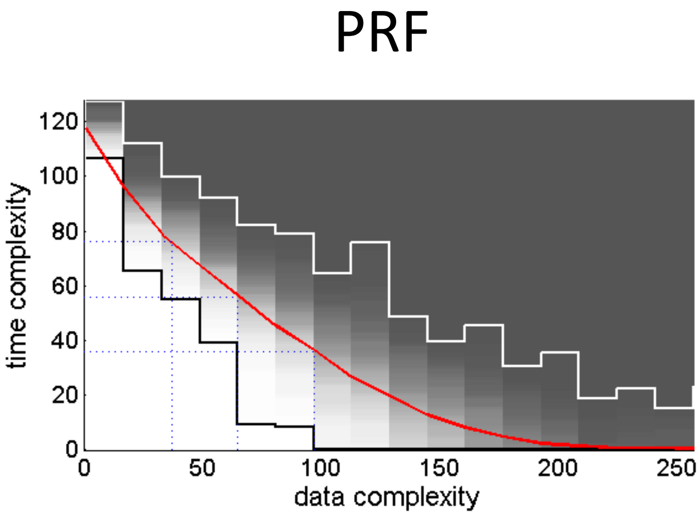


- Re-keying impact: bounds the number of noise-free observations per key (*allows averaging*)

The stateful / stateless separation

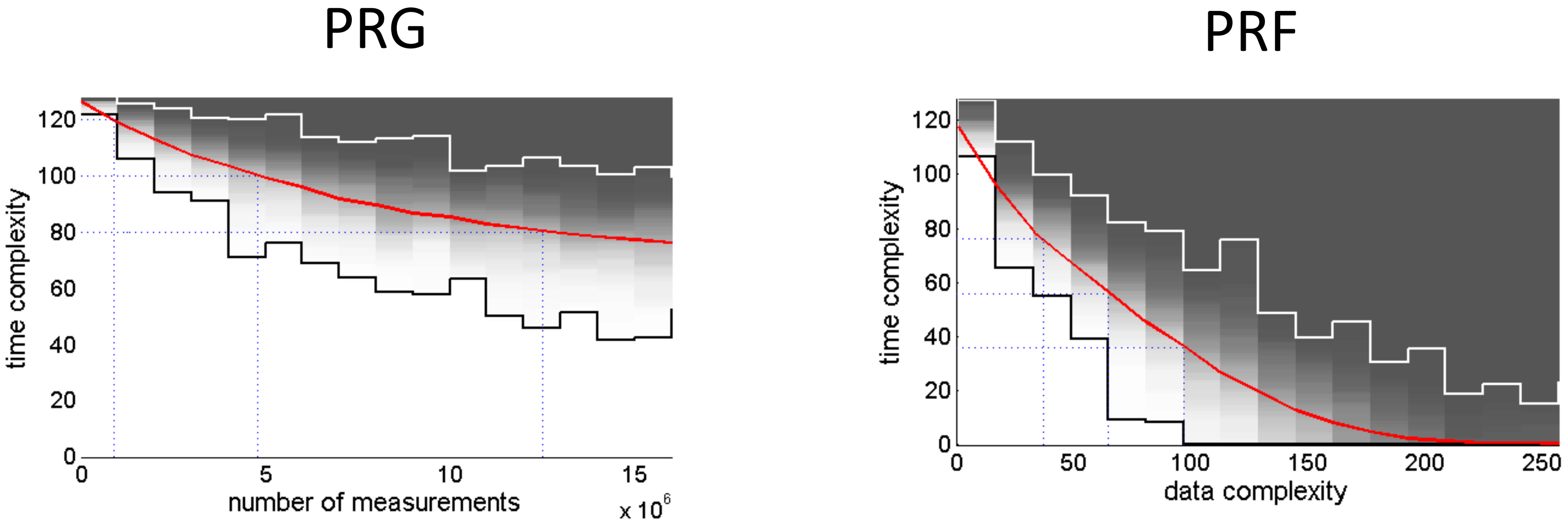
- Key recovery security (standard DPA) [BGS15]:





The stateful / stateless separation

- Key recovery security (standard DPA) [BGS15]:



- « Bounded security » for the PRG only
  - (Analytical/algebraic attacks not considered)

Take home message

- Leakage-resilience can at least provide good security guarantees (against key recovery attaks) for stateful primitives such as PRGs
  - With a constant overhead factor ≤ 2

Take home message

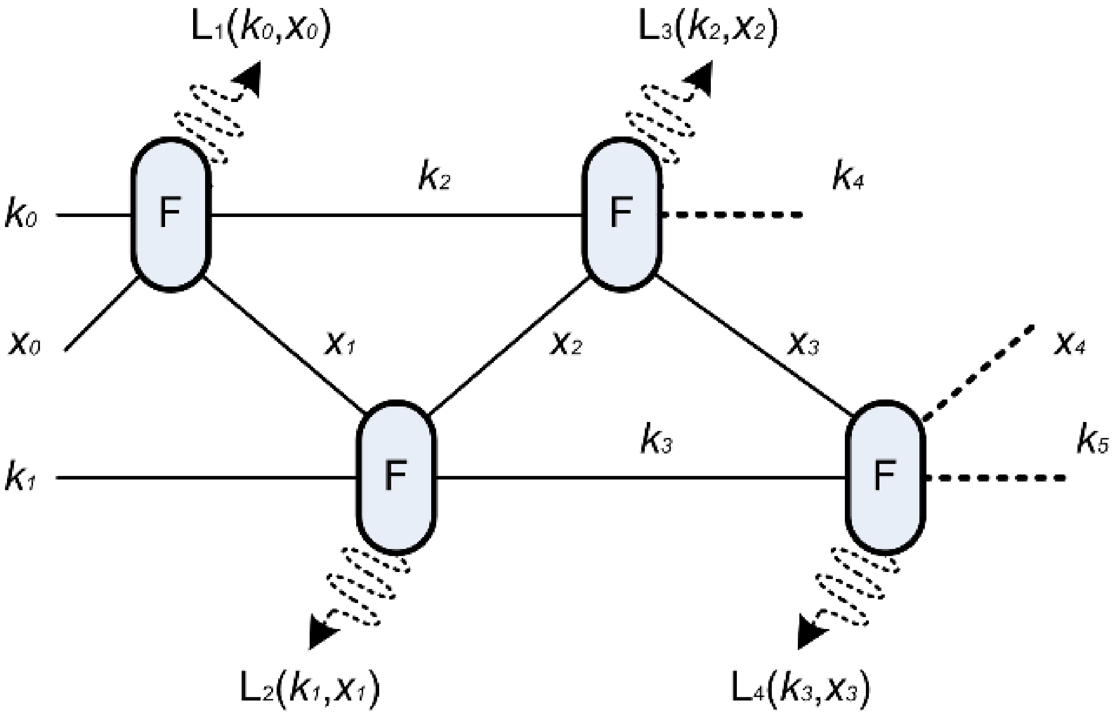
- Leakage-resilience can at least provide good security guarantees (against key recovery attaks) for stateful primitives such as PRGs
  - With a constant overhead factor ≤ 2

- Yet, we need at least one stateless primitive execution for initialization (that needs to be secured by other means such as masking)

Outline

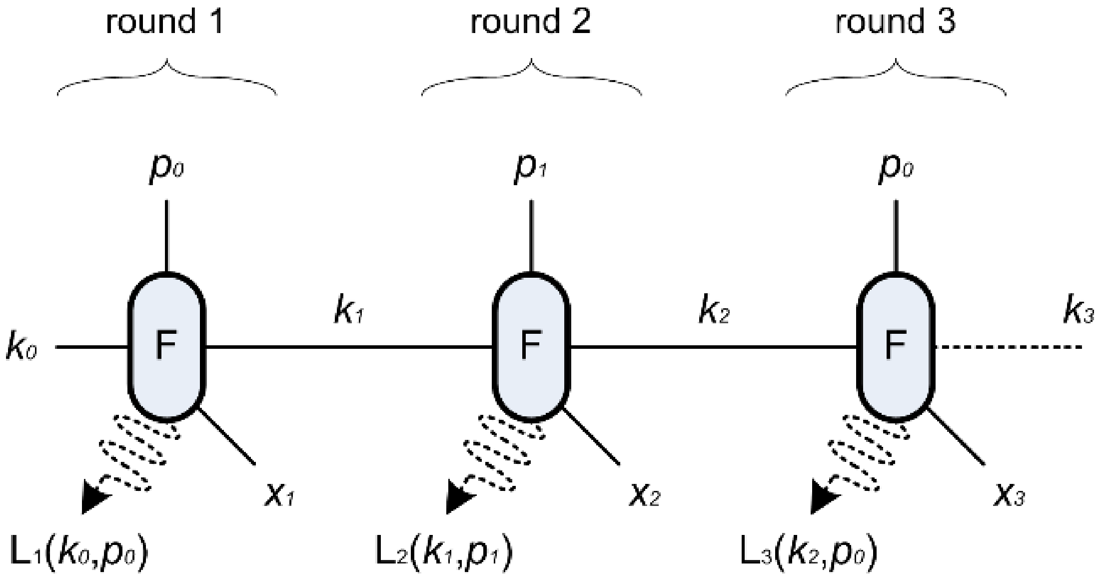
- Starting point (link with previous lecture)
- Seed results (TCC 2004, FOCS 2008)
- **Primitives (PRGs/PRFs,PRPs)**
  - If you don’t care about proofs
    - The stateful/stateless separation
  - **The proof/assumptions challenge**
    - **Ensuring independence**
    - Bounding the leakage
    - The simulatable leakage attempt
- « Pragmatic » auth. & encryption (CCS 2015)
- Back to stateless primitives
- Conclusions & open problems

FOCS 2008 / Eurocrypt 2009

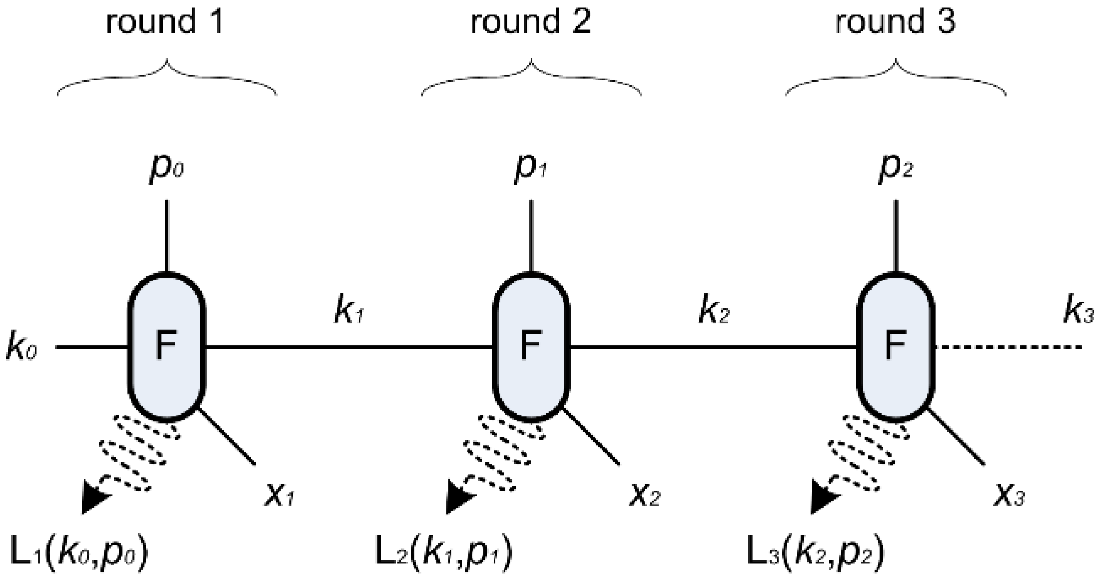


- L modeled as a polytime function => alternating structure prevents « precomputation attack »

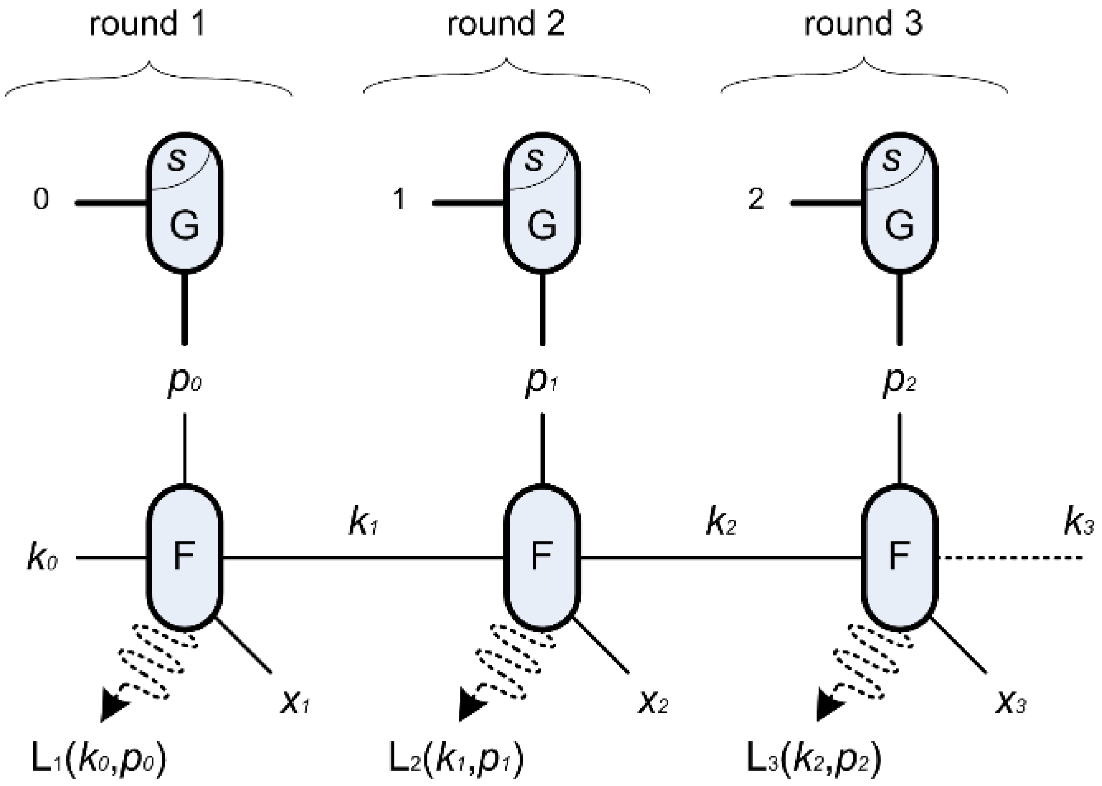
CCS 2010



- Alternating randomness (to save key material)
  - Unfortunately not sufficient (CHES 2012)…

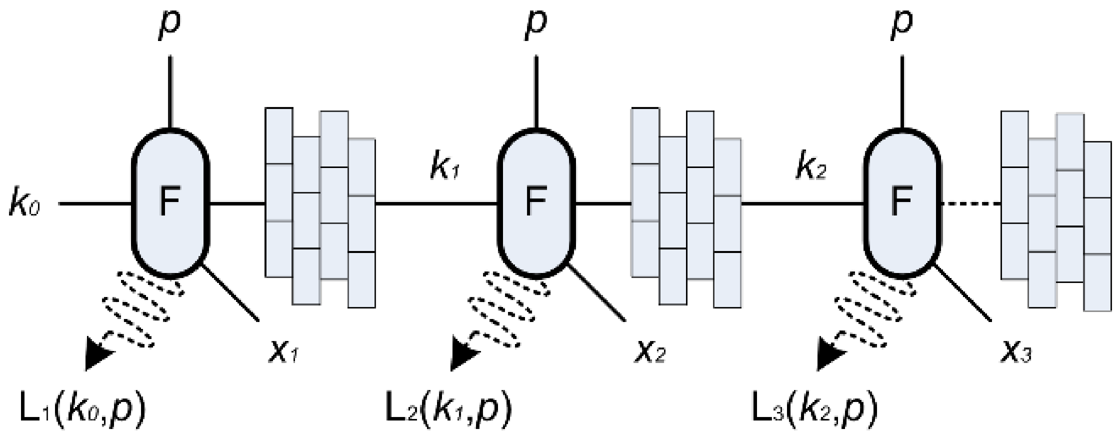


- Fresh randomness in each round
  - Sound but expensive (generated after L)



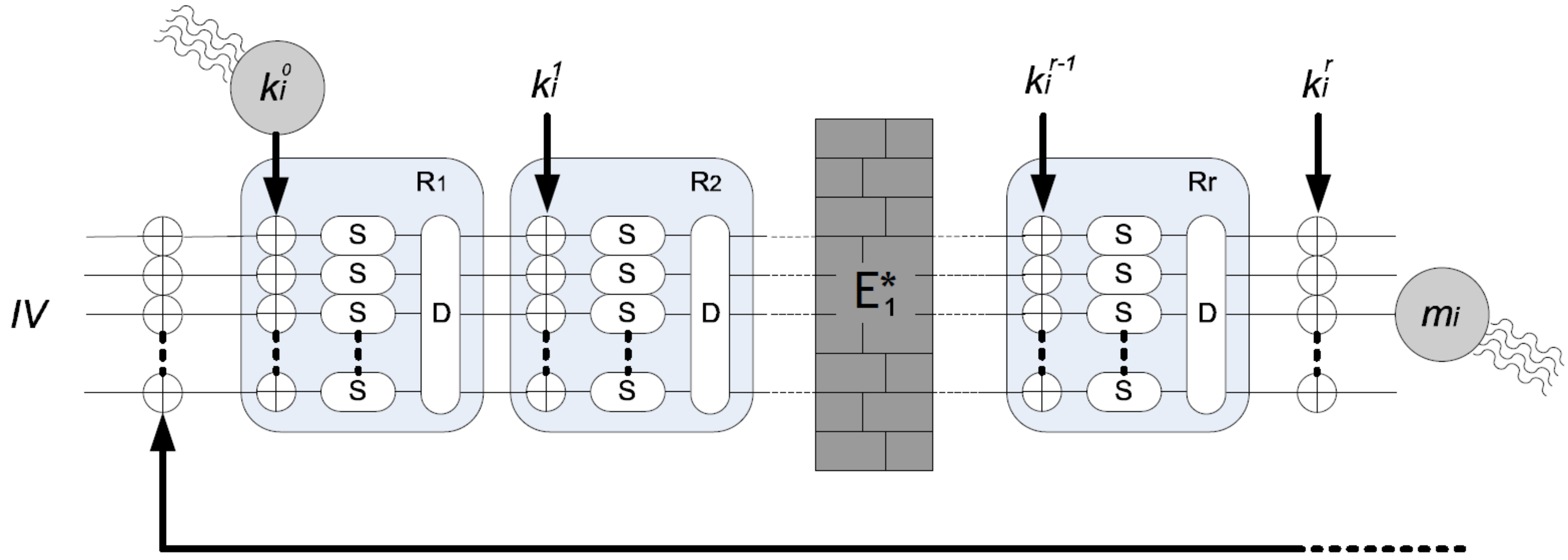
- Public randomness generated from a PRG
    - (Non quantitative) proof in MiniCrypt

CCS 2010 again (I)



- Most natural construction proven under a (non standard) random oracle assumption
  - L cannot query the random oracle

CCS 2010 again (II)



- ≈ formalization of early re-keying attempts
  - e.g. ASIACCS 2008: internal wall within AES
  - e.g. early patents in the field from CRI
  - (*Where it was already clear that init. is challenging*!)

Wrapping up

- Finding realistic & efficient ways to guarantte the independence between multiple PRG rounds is notorioulsy difficult (!)

Wrapping up

- Finding realistic & efficient ways to guarantte the independence between multiple PRG rounds is notorioulsy difficult (!)
    - No perfectly satisfying solution so far
    - Mostly because L is assumed polytime
    - & no other restrictions seem realistic

Wrapping up

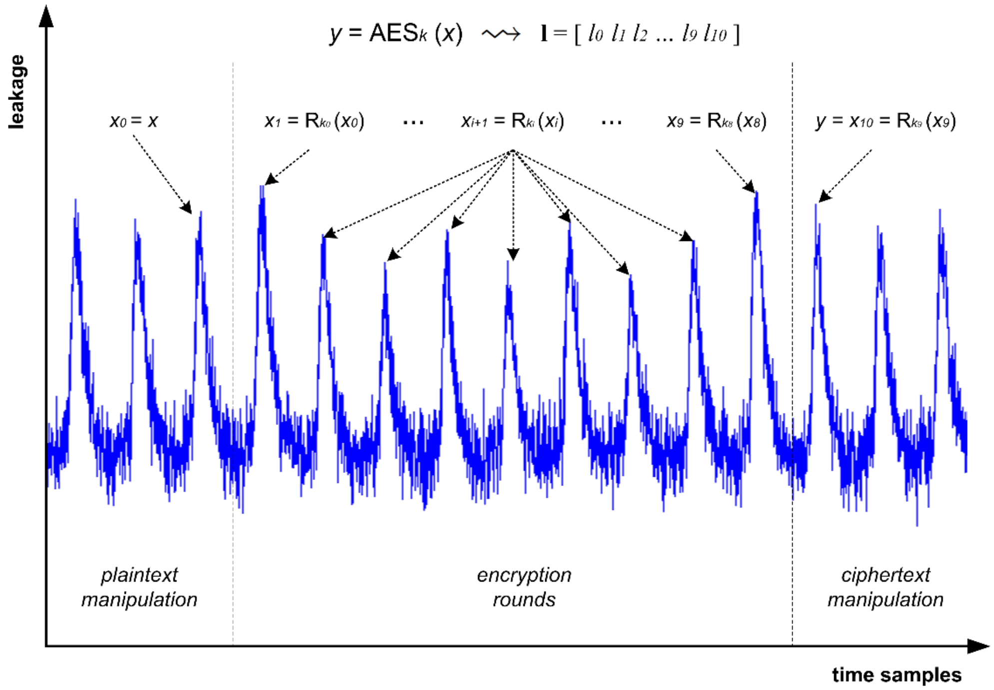
- Finding realistic & efficient ways to guarantte the independence between multiple PRG rounds is notorioulsy difficult (!)
    - No perfectly satisfying solution so far
    - Mostly because L is assumed polytime
    - & no other restrictions seem realistic

- Note: similar story for PRFs and PRPs (although less relevant in view of the separation in slide 7)

Outline

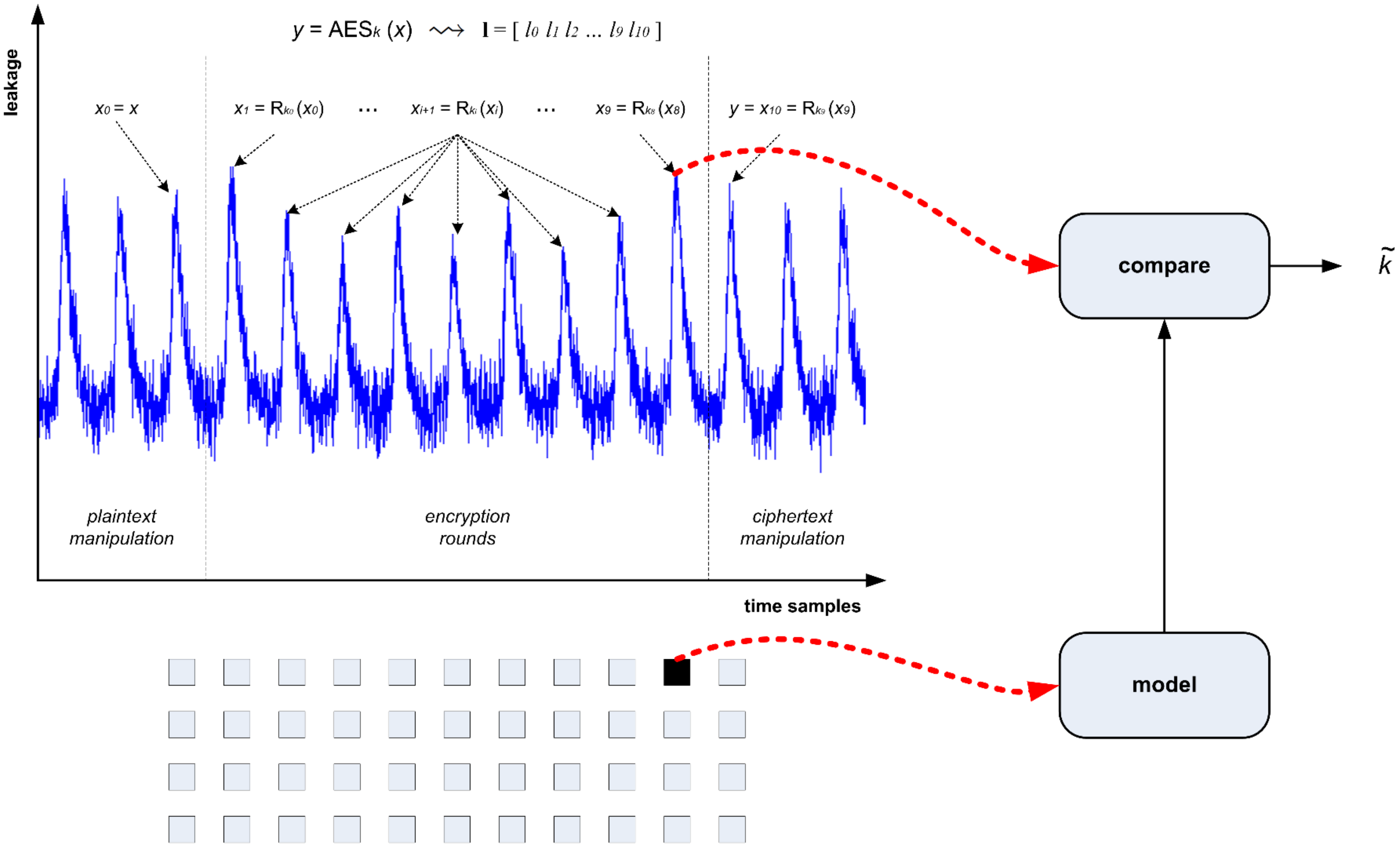
- Starting point (link with previous lecture)
- Seed results (TCC 2004, FOCS 2008)
- **Primitives (PRGs/PRFs,PRPs)**
  - If you don’t care about proofs
    - The stateful/stateless separation
  - The proof/assumptions challenge
    - Ensuring independence
    - **Bounding the leakage**
    - The simulatable leakage attempt
- « Pragmatic » auth. & encryption (CCS 2015)
- Back to stateless primitives
- Conclusions & open problems

Bounded range



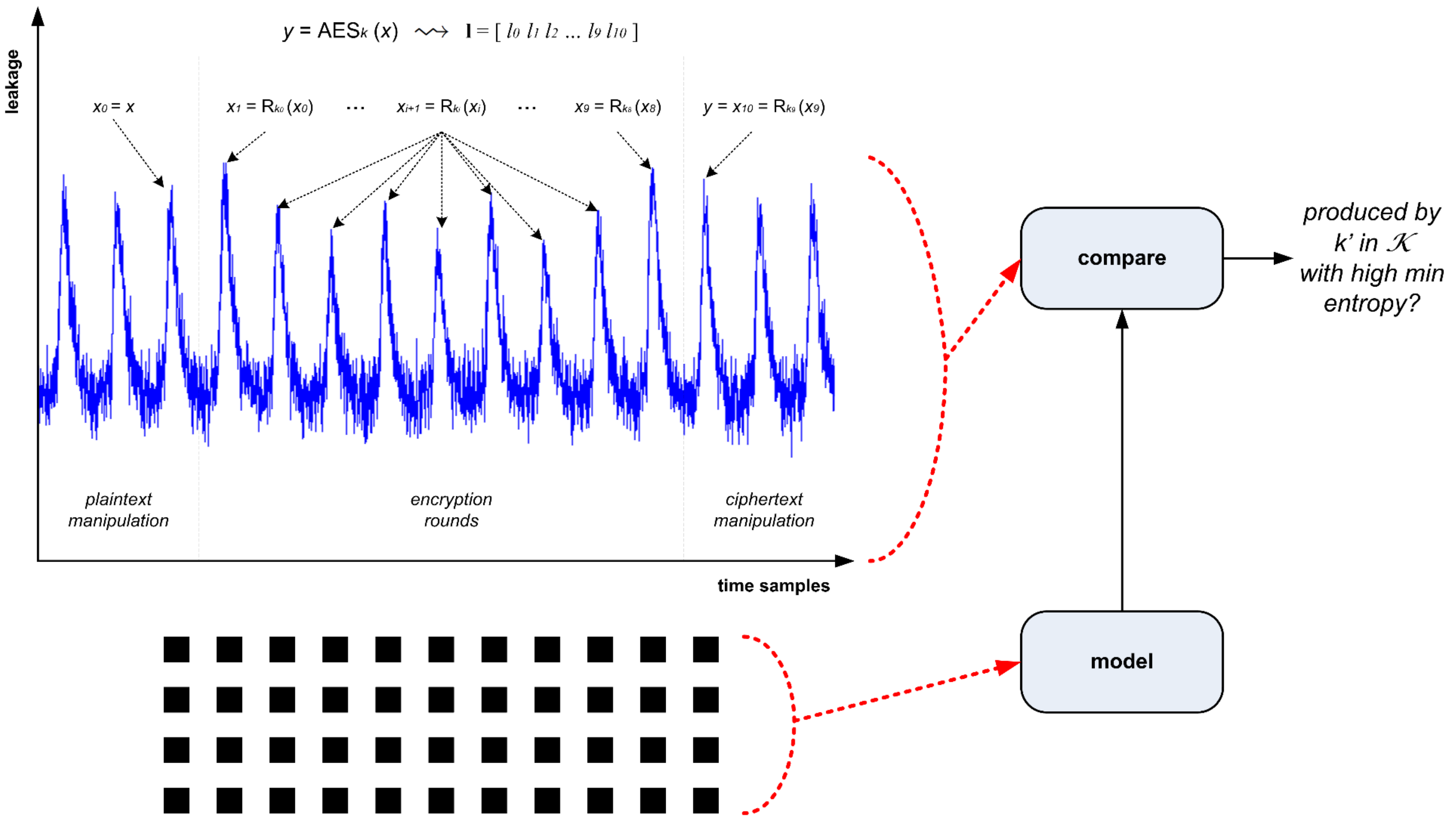
- Unrealistic: leakages ≈ Gbytes of data

Security against DPA



- Not sufficient to prove anything

Key has high HILL pseudoentropy



compare

produced by k' in $\mathcal{K}$ with high min entropy?

model

- Hard to guarantee (indistinguishability-based)

Wrapping up

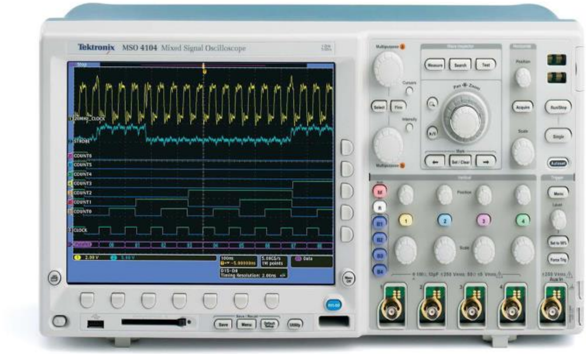
- Finding realistic ways to bound the leakage in leakage-resilient PRGs is notoriously difficult
  - No perfectly satisfying solution so far
  - ∃ a gap between what proofs require and what engineers can guarantee (evaluate)

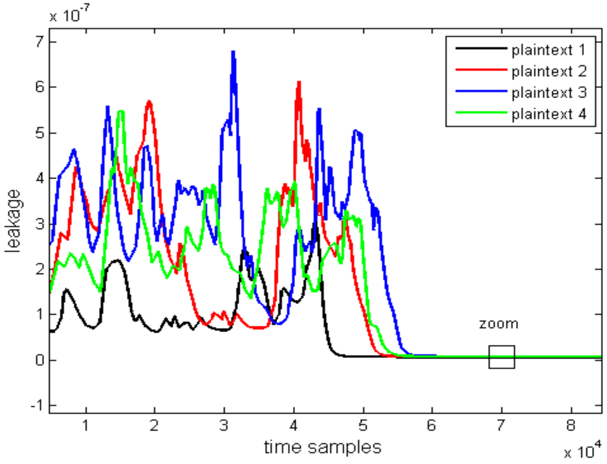
Outline

- Starting point (link with previous lecture)
- Seed results (TCC 2004, FOCS 2008)
- **Primitives (PRGs/PRFs,PRPs)**
  - If you don’t care about proofs
    - The stateful/stateless separation
  - The proof/assumptions challenge
    - Ensuring independence
    - Bounding the leakage
    - **The simulatable leakage attempt**
- « Pragmatic » auth. & encryption (CCS 2015)
- Back to stateless primitives
- Conclusions & open problems

Looking for physical assumptions

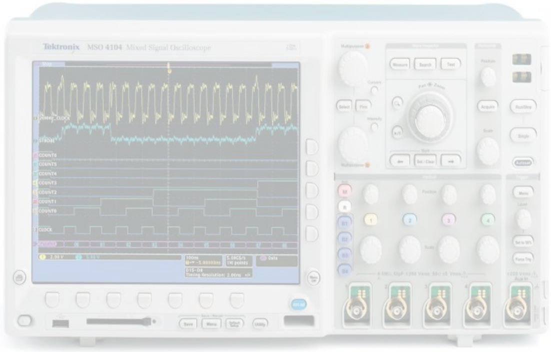
- Main issue: leakage function is hard to model
    - It solves Maxwell’s equations
    - But circuits give immediate solutions

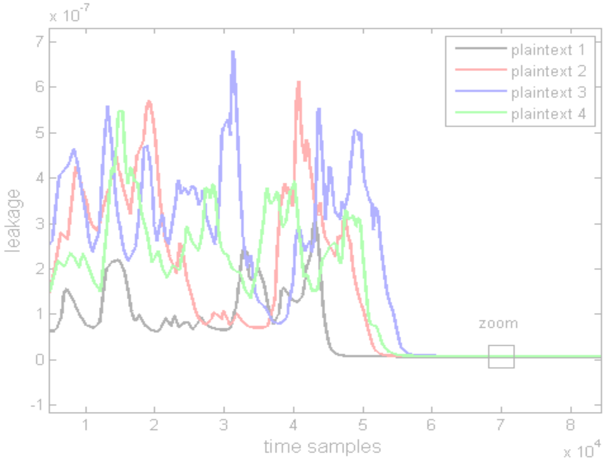




Looking for physical assumptions

- Main issue: leakage function is hard to model
  - It solves Maxwell's equations
  - But circuits give immediate solutions

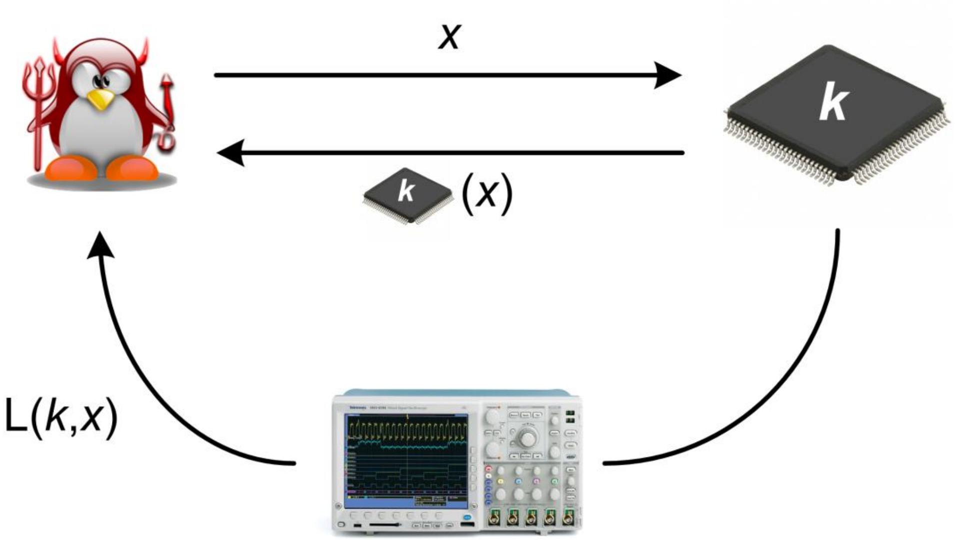




=> Just don't model it!

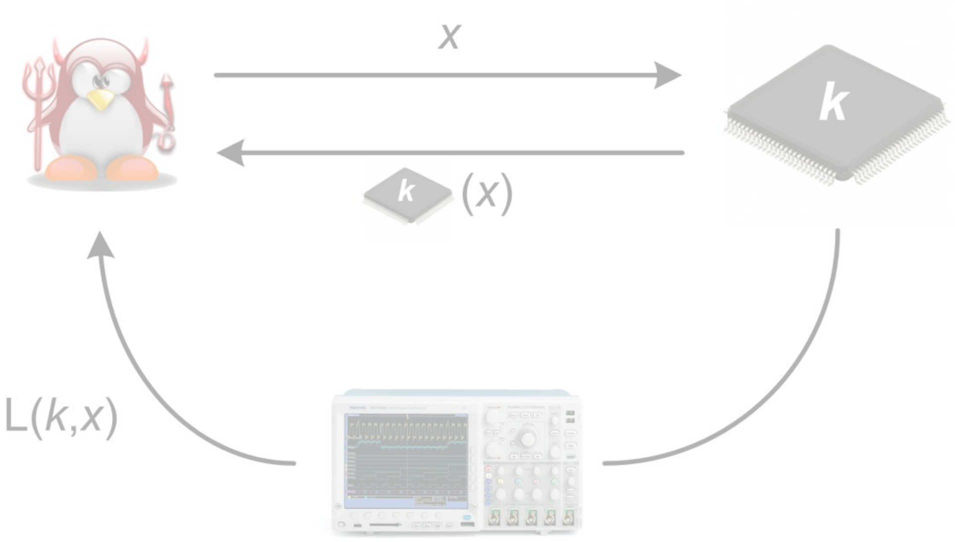
Our setting (Crypto 2013)

(a) Give public I/O access to device & setup



Our setting (Crypto 2013)

(a) Give public I/O access to device & setup



(b) Assume L(*k*,*x*) can be simulated

- Using the same HW as the target
- But without knowing the secret key *k*!

More formally

(*k* , ) has simulatable leakages if ∃ S^L such that the bit *b* in the following game is hard to guess

Game *q*-sim(Adv, *k*, S^L,*b*) with *k*,*k** uniformly random

q queries	response if *b*=0	response if *b*=1
Enc(*x*)	k(*x*), S^L(*k*,x,k(*x*))	k(*x*), S^L(*k**,x,k(*x*))
1 query	response if *b*=0	response if *b*=1
Gen(*x*)	S^L(*z*,*x*,k)	S^L(*z*,*x*,k*)

More formally

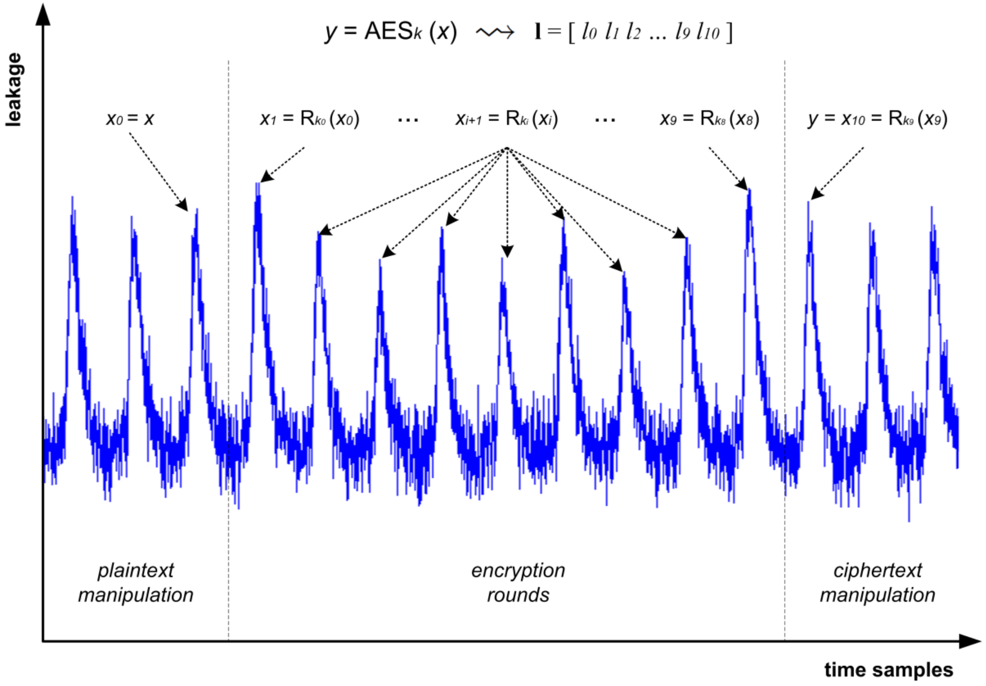
(k, ) has simulatable leakages if ∃ S^L such that the bit *b* in the following game is hard to guess

Game *q*-sim(Adv, k, S^L, *b*) with *k*, *k** uniformly random		
q queries	response if *b*=0	response if *b*=1
Enc(*x*)	k(*x*), $S^L(k,x,$k$(x))$	k(*x*), $S^L(k^*,x,$k$(x))$
1 query	response if *b*=0	response if *b*=1
Gen(*x*)	$S^L(z,x,k)$	$S^L(z,x,k^*)$

- With $S^L(k,x,$k$(x)) \overset{\text{def}}{=} L(k,x)$ (*makes our results dependent only on the number of calls to* S^L)

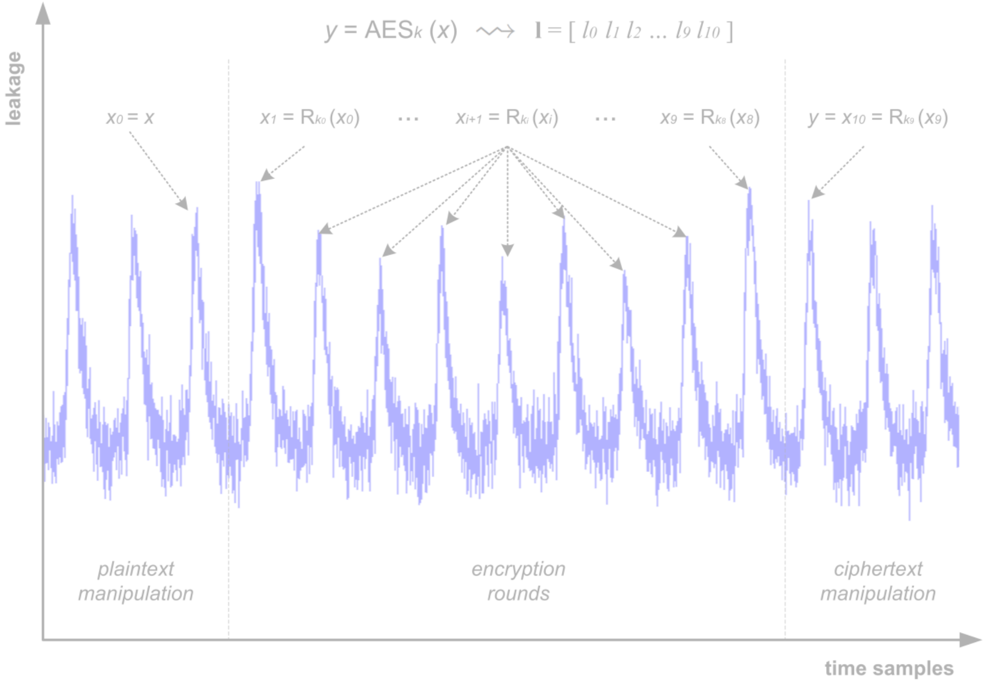
Block cipher leakage simulator

- Let $L(k,x) = l^p(k,x) || l^c(k, k(x))$
  - l^p corresponds to the first rounds of k
  - l^c corresponds to the last rounds of k
- e.g.



Block cipher leakage simulator

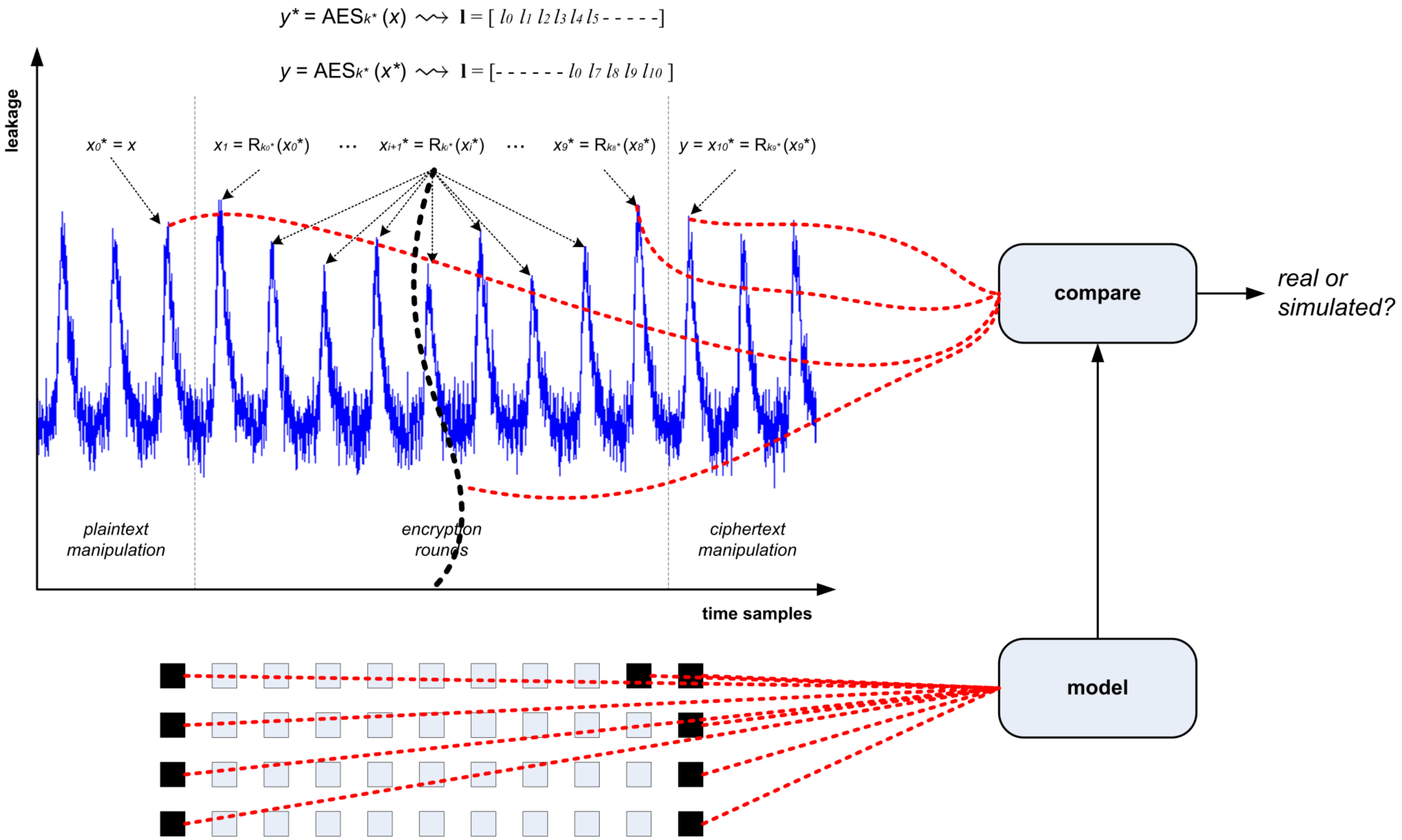
- Let $L(k,x) = l^p(k,x) \,||\, l^c(k, k(x))$
  - l^p corresponds to the first rounds of k
  - l^c corresponds to the last rounds of k
- e.g.



$\Rightarrow$ Instantiate $S^L(k,x,y) = l^p(k,x) \,||\, l^c(k,y)$

Why would this work?

Simulatable leakages ≈ DPA + I/O’s leakages



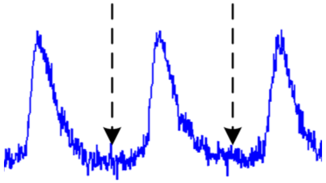
Summarizing

a. Attacks against *q*-sim. exploit the same leakages as DPA if the traces are consistent with the I/O's

   - *this is exactly what the simulator does*

b. Additionally needs concatenation

   - *OK if ∃ leakage samples without interest:*



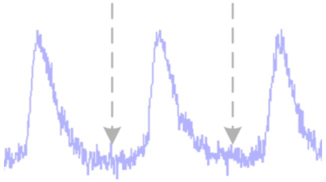
Summarizing

a. Attacks against *q*-sim. exploit the same leakages as DPA if the traces are consistent with the I/O's

  - *this is exactly what the simulator does*

b. Additionally needs concatenation

  - *OK if ∃ leakage samples without interest:*



c. *q*-sim. at least easier to guarantee than H^{HILL}

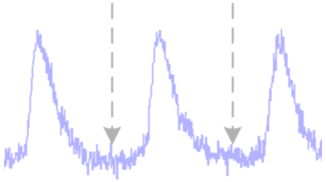
Summarizing

a. Attacks against *q*-sim. exploit the same leakages as DPA if the traces are consistent with the I/O's

- *this is exactly what the simulator does*

b. Additionally needs concatenation

- *OK if ∃ leakage samples without interest:*



c. *q*-sim. at least easier to guarantee than H^{HILL}

d. Engineering challenges

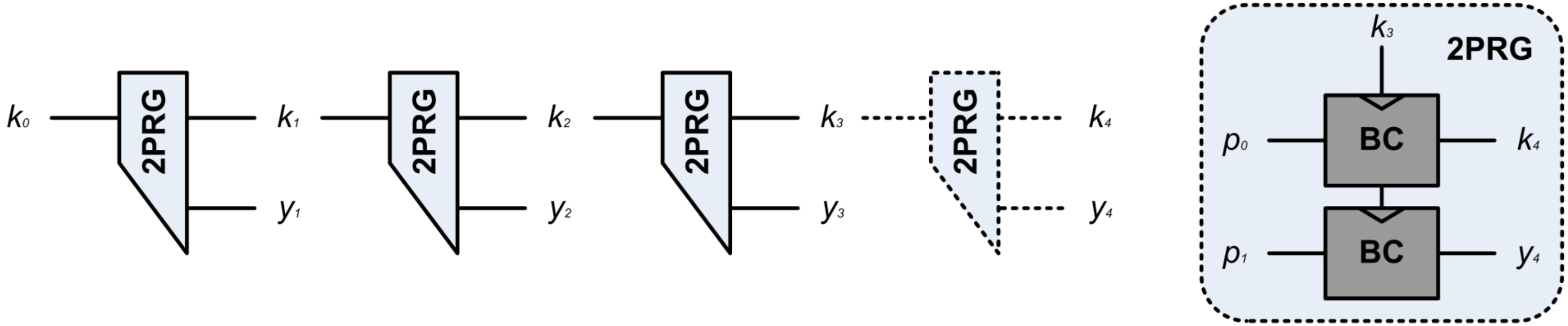
(constructive) Design alternative S^L instances

(constructive) Given S^L, design  with *q*-sim. leakages

(destructive) Given S^L and , break the *q*-sim. game

First instances falsified by Galea et al. (cfr. end of talk if time allows)

Most natural construction

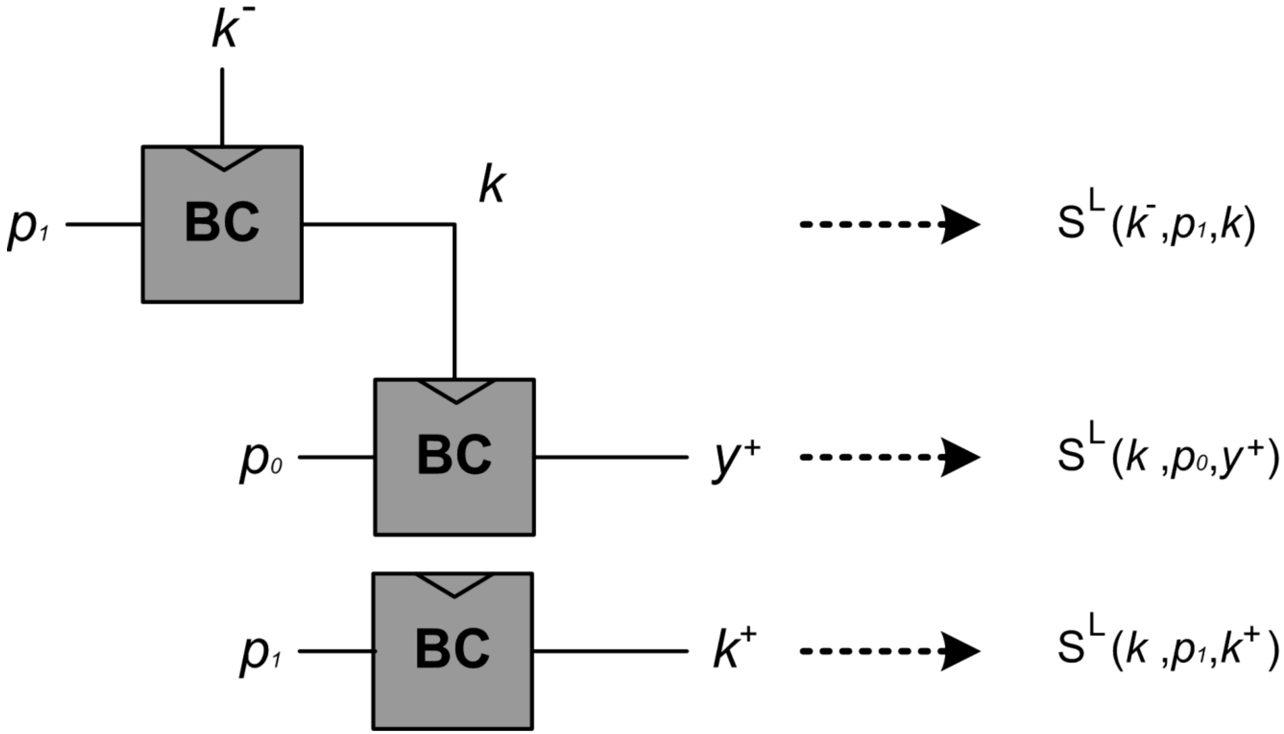


- Goal: remain secure after $\approx 10^6$ runs
- While relying on *q*-sim. for *q*=2

- Proving it was surprisingly difficult so far
  - (see slides 9 to 19 of this talk)

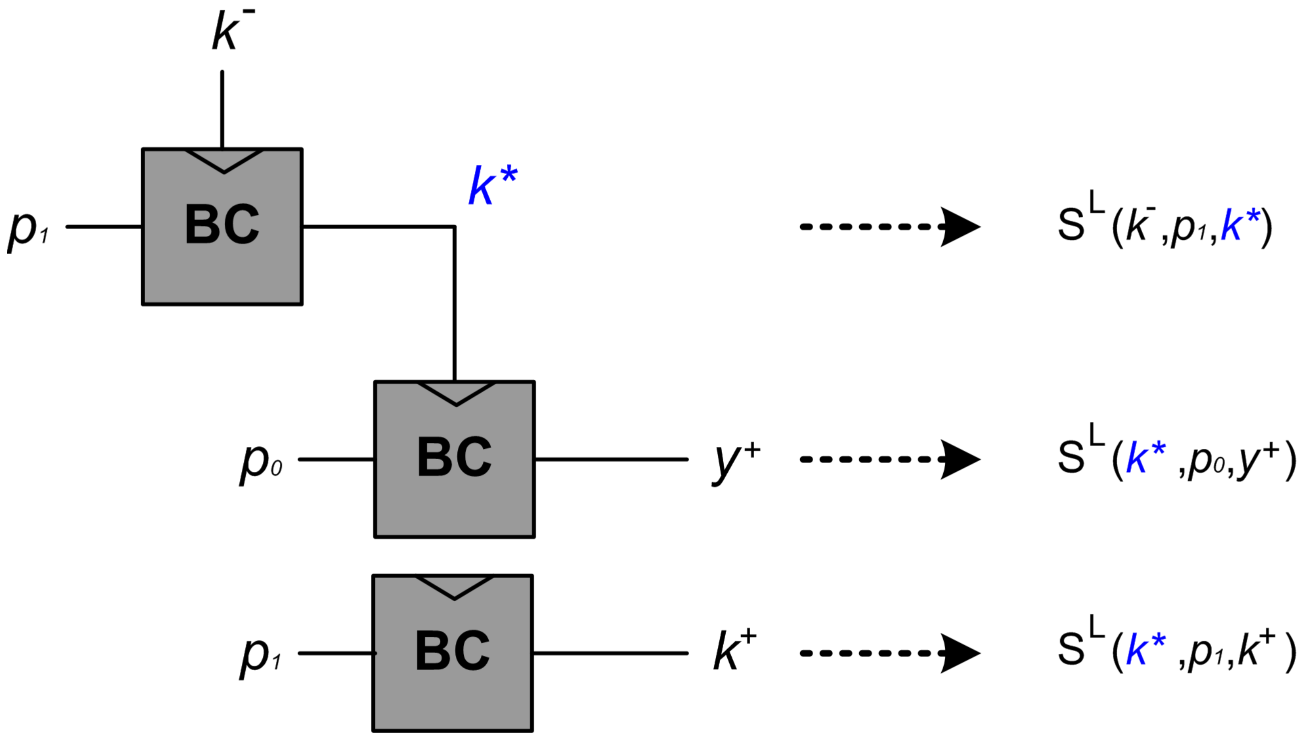
Proof idea #1: replacing lemma

Original view



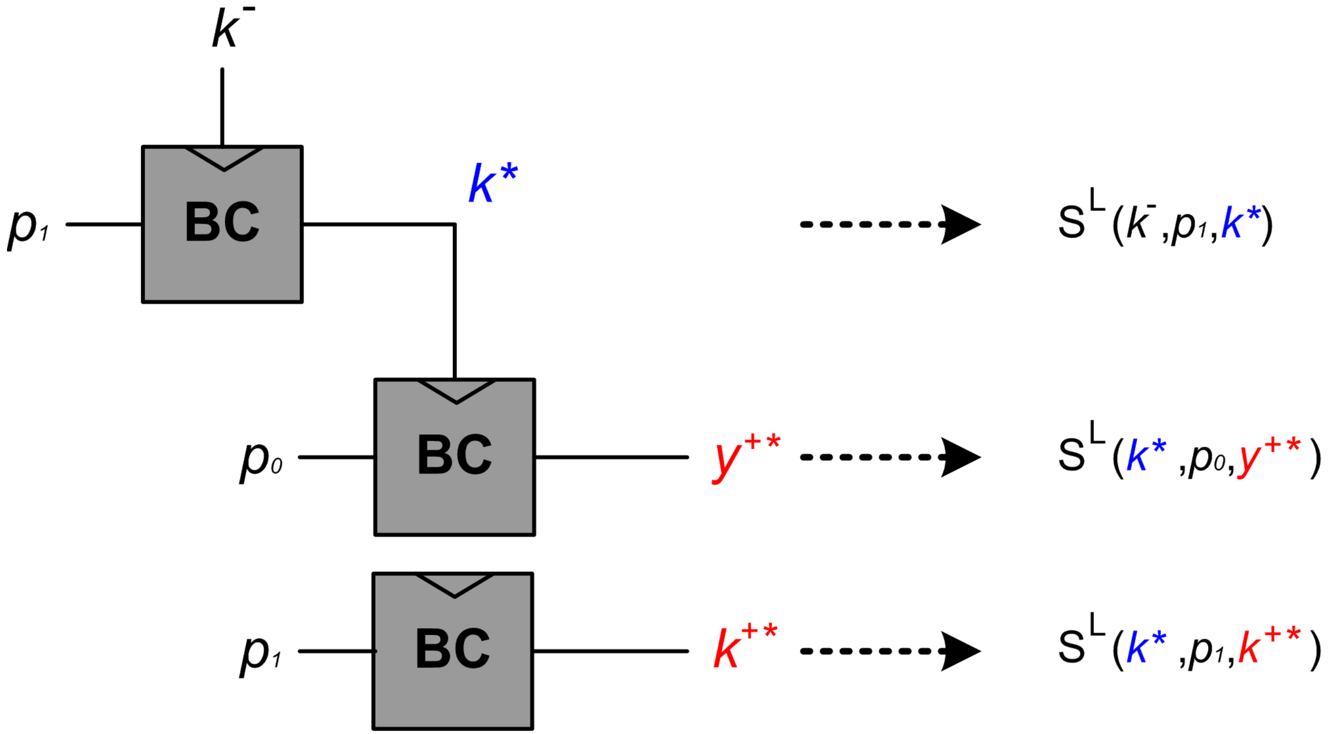
Proof idea #1: replacing lemma

a. Exploit the 2-sim. leakages assumption



Proof idea #1: replacing lemma

b. Exploit the BC ≈ PRF assumption



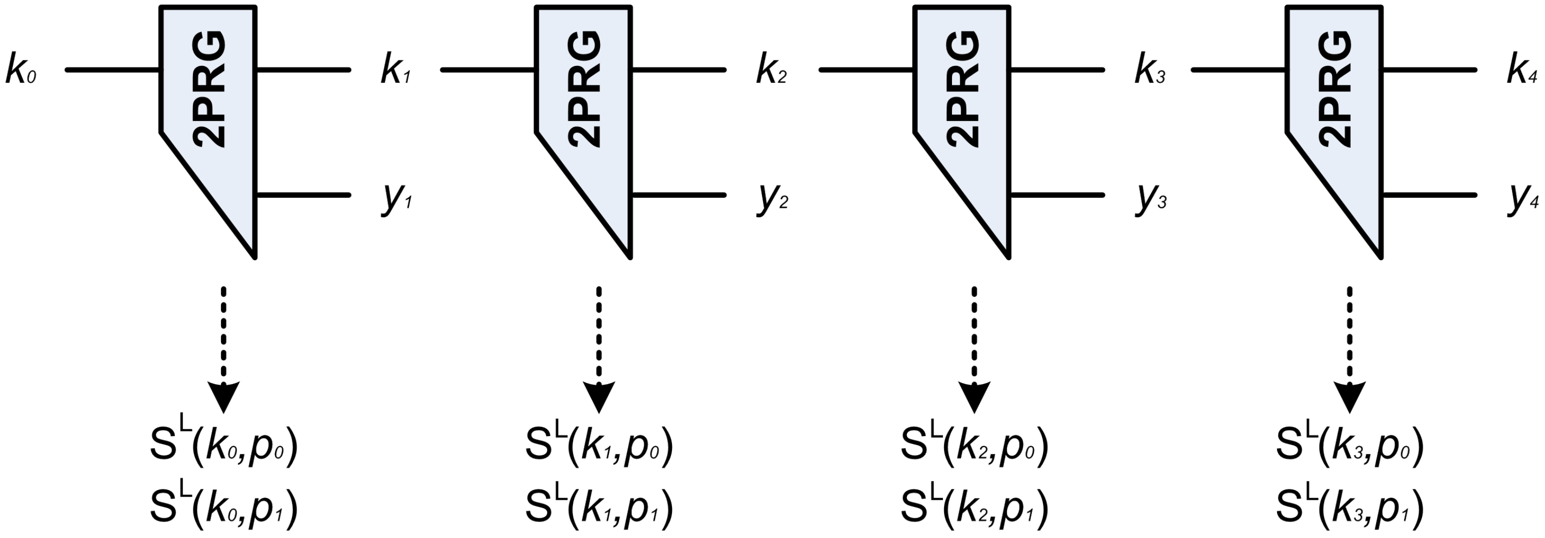
$S^L(k^-, p_1, k^*)$

$S^L(k^*, p_0, y^{+*})$

$S^L(k^*, p_1, k^{+*})$

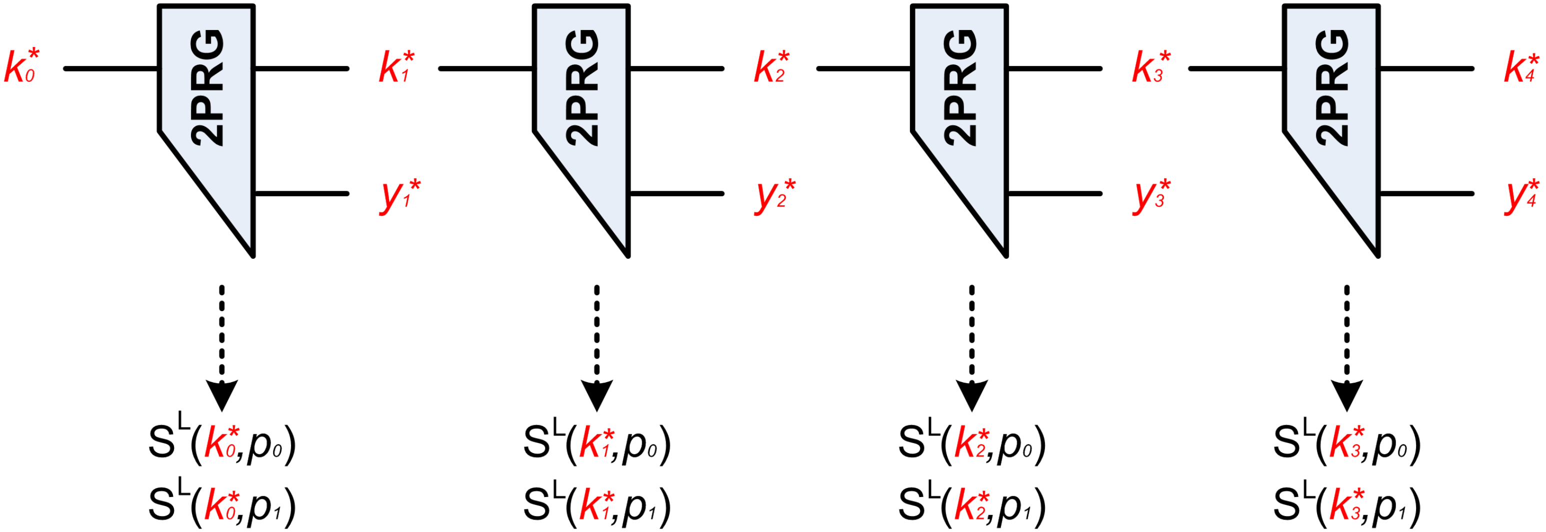
Proof idea #2: extend (hybrid argument)

Original view



Proof idea #2: extend (hybrid argument)

a. Completely random view (*l*=4 calls to S^L)



$S^L(k_0^*, p_0)$
$S^L(k_0^*, p_1)$

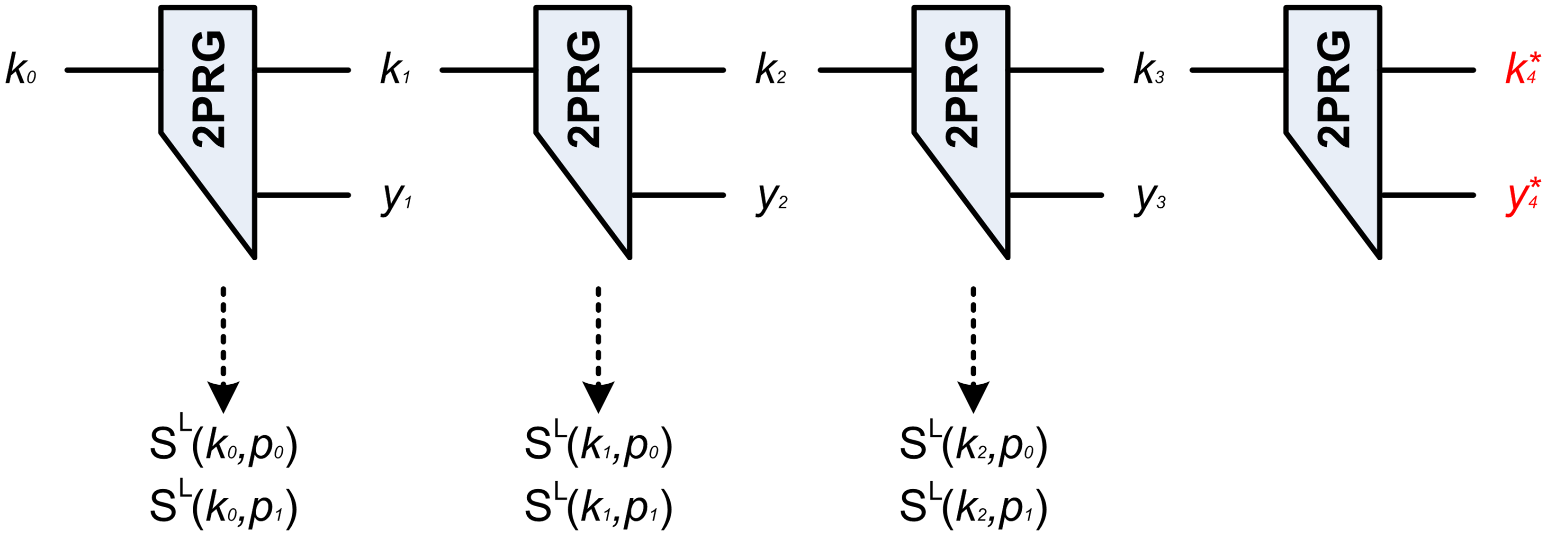
$S^L(k_1^*, p_0)$
$S^L(k_1^*, p_1)$

$S^L(k_2^*, p_0)$
$S^L(k_2^*, p_1)$

$S^L(k_3^*, p_0)$
$S^L(k_3^*, p_1)$

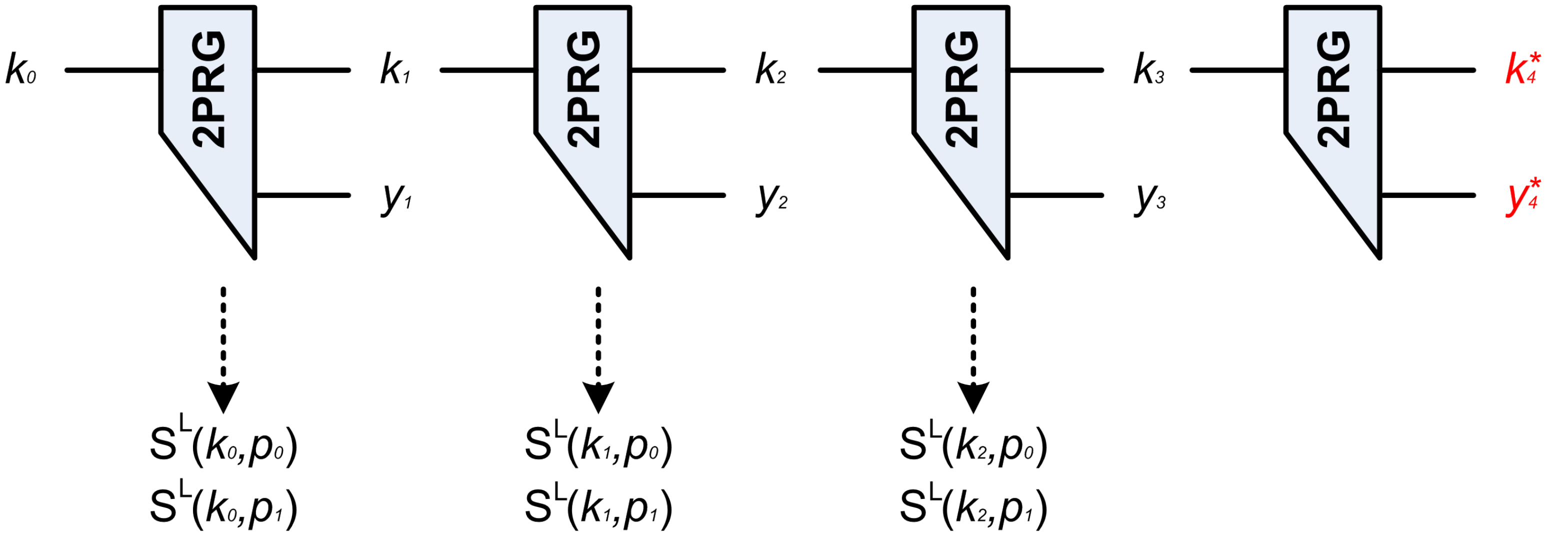
Proof idea #2: extend (hybrid argument)

b. Real view with random y_4 (l=4 calls to S^L)



Proof idea #2: extend (hybrid argument)

b. Real view with random y_4 (l=4 calls to S^L)



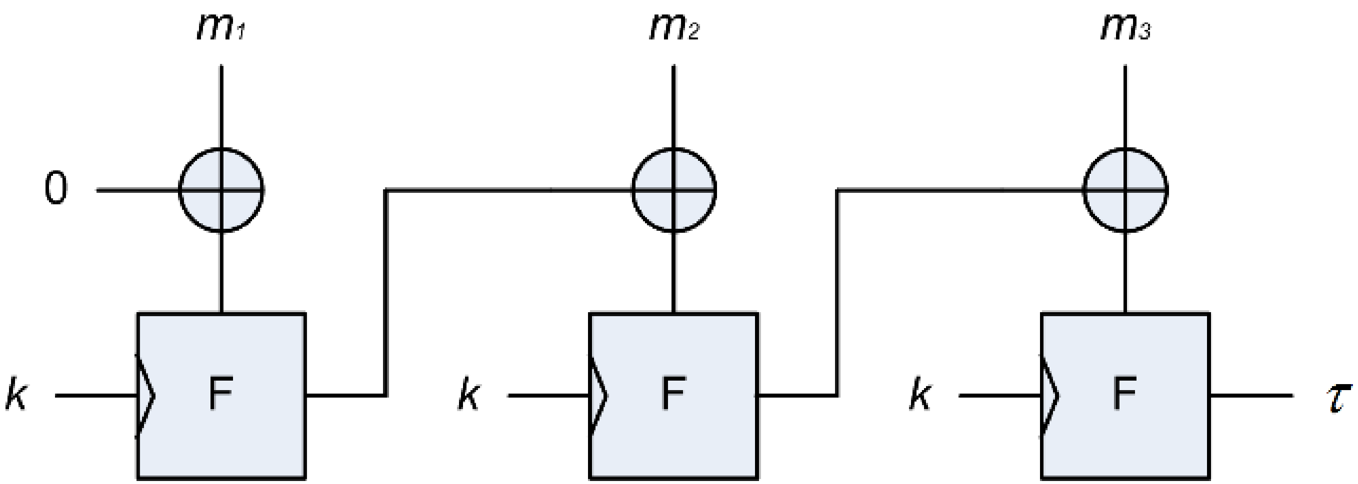
Theorem: $y_l \approx U_n$ given $y_1,\dots,y_{l-1}$,L(k_0),L(k_{l-2}) if BC is a PRF and has 2-simulatable leakages

(*with security degradation proportional to 2l*)

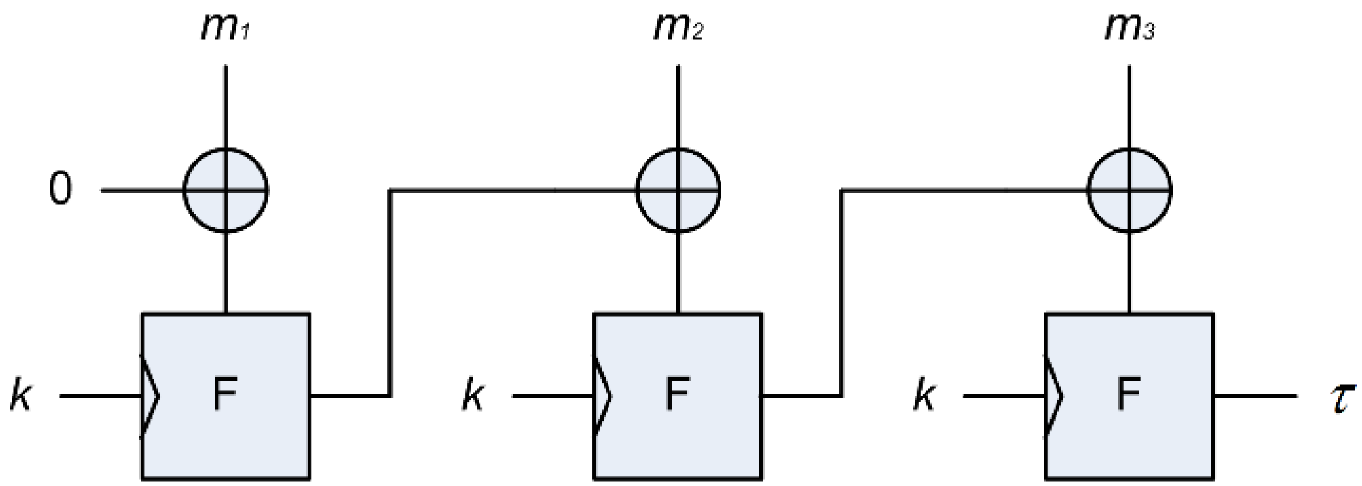
Outline

- Starting point (link with previous lecture)
- Seed results (TCC 2004, FOCS 2008)
- Primitives (PRGs/PRFs,PRPs)
  - If you don’t care about proofs
    - The stateful/stateless separation
  - The proof/assumptions challenge
    - Ensuring independence
    - Bounding the leakage
    - The simulatable leakage attempt
- **« Pragmatic » auth. & encryption (CCS 2015)**
- Back to stateless primitives
- Conclusions & open problems

CBC-MAC (is insecure)



CBC-MAC (is insecure)



- Master *k* key re-used multiple times

  ⇒ Eventually leaked in full (via DPA)

LR-MAC: security definition

- Natural extension of unforgeability without L

```
$\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$:
    $k \leftarrow \mathsf{KeyGen}(1^n)$
    $\mathcal{F} \leftarrow \emptyset$
    $(m,\tau) \leftarrow \mathcal{A}^{\mathsf{L},\mathcal{O}^{\mathsf{ML}}(\cdot),\mathcal{O}^{\mathsf{V}}(\cdot,\cdot)}(1^n)$
    If $m \in \mathcal{F}$, then Return $b := 0$
    $b \leftarrow \mathsf{Vrfy}(m,\tau,k)$
    Return $b$
```

```
Oracle $\mathcal{O}^{\mathsf{ML}}(m)$:
    $r \leftarrow \{0,1\}^n$
    $\mathcal{F} \leftarrow \mathcal{F} \cup m$
    Return $(\mathsf{Mac}(m,k;r),$
                $\mathsf{L}(m,k;r))$
Oracle $\mathcal{O}^{\mathsf{V}}(m,\tau)$:
    Return $\mathsf{Vrfy}(m,\tau,k)$
```

LR-MAC: security definition

- Natural extension of unforgeability without L

```
$\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$:
    $k \leftarrow \mathsf{KeyGen}(1^n)$
    $\mathcal{F} \leftarrow \emptyset$
    $(m, \tau) \leftarrow \mathcal{A}^{\mathsf{L}, \mathcal{O}^{\mathsf{ML}}(\cdot), \mathcal{O}^{\mathsf{V}}(\cdot,\cdot)}(1^n)$
    If $m \in \mathcal{F}$, then Return $b := 0$
    $b \leftarrow \mathsf{Vrfy}(m, \tau, k)$
    Return $b$
```

```
Oracle $\mathcal{O}^{\mathsf{ML}}(m)$:
    $r \leftarrow \{0,1\}^n$
    $\mathcal{F} \leftarrow \mathcal{F} \cup m$
    Return $(\mathsf{Mac}(m, k; r),$
            $\mathsf{L}(m, k; r))$
Oracle $\mathcal{O}^{\mathsf{V}}(m, \tau)$:
    Return $\mathsf{Vrfy}(m, \tau, k)$
```

- Adversary gets the leakage for tag generation

LR-MAC: security definition

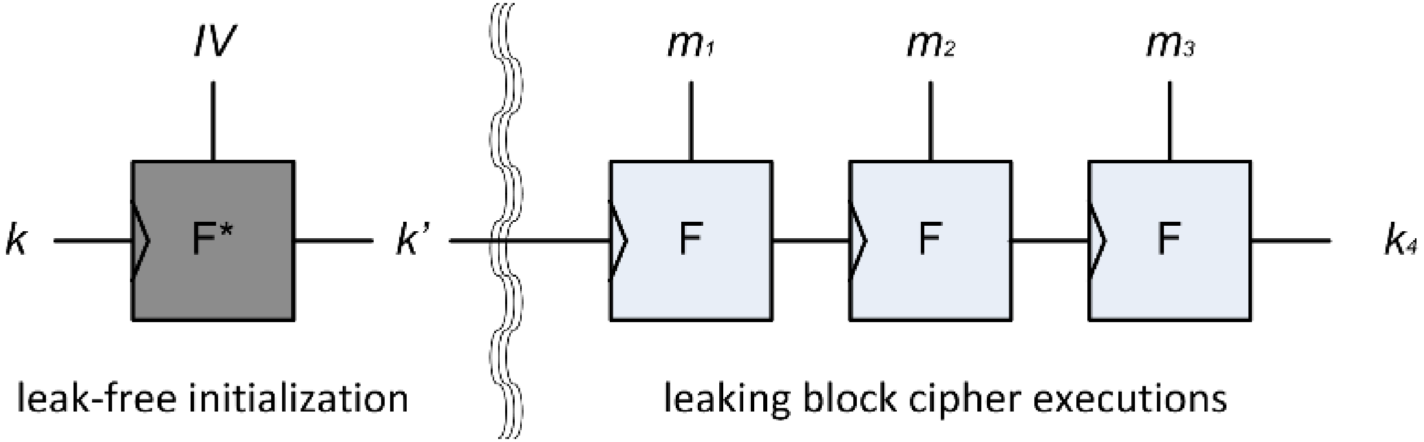
- Natural extension of unforgeability without L

```
$\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$:
    $k \leftarrow \mathsf{KeyGen}(1^n)$
    $\mathcal{F} \leftarrow \emptyset$
    $(m, \tau) \leftarrow \mathcal{A}^{\mathsf{L},\mathcal{O}^{\mathsf{ML}}(\cdot),\mathcal{O}^{\mathsf{V}}(\cdot,\cdot)}(1^n)$
    If $m \in \mathcal{F}$, then Return $b := 0$
    $b \leftarrow \mathsf{Vrfy}(m, \tau, k)$
    Return $b$
```

```
Oracle $\mathcal{O}^{\mathsf{ML}}(m)$:
    $r \leftarrow \{0,1\}^n$
    $\mathcal{F} \leftarrow \mathcal{F} \cup m$
    Return $(\mathsf{Mac}(m, k; r),$
               $\mathsf{L}(m, k; r))$
Oracle $\mathcal{O}^{\mathsf{V}}(m, \tau)$:
    Return $\mathsf{Vrfy}(m, \tau, k)$
```

- Adversary gets the leakage for tag generation
  - But not during the verification algorithm

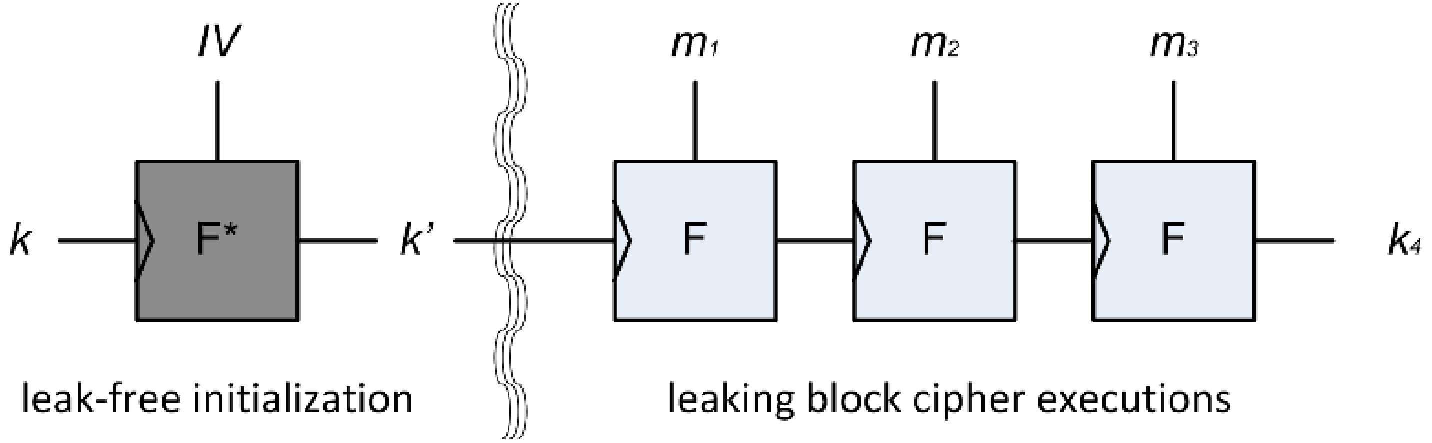
Construction I: re-keying MAC



leak-free initialization

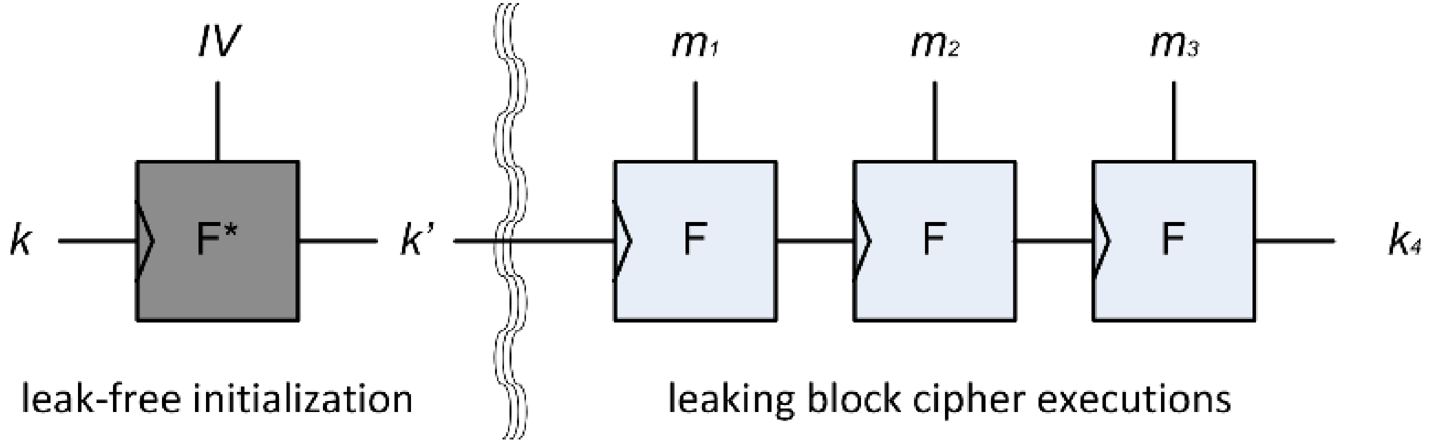
leaking block cipher executions

Construction I: re-keying MAC



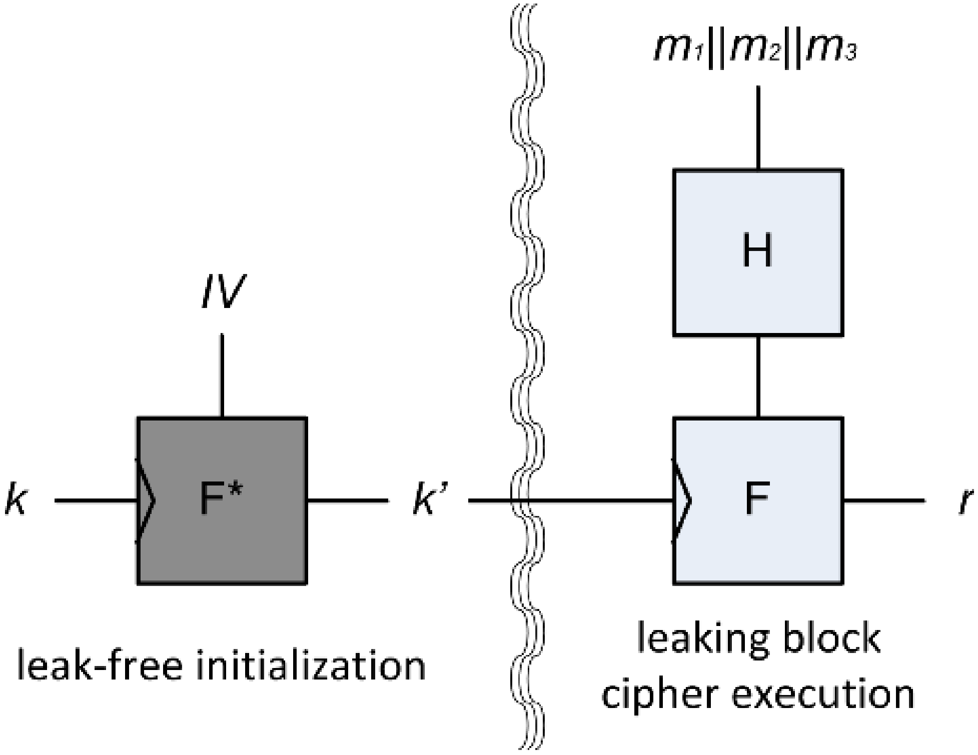
- Pragmatism: requires one leak-free block cipher execution for initialization (cfr. slide 8)
  - Then takes advantage of statefullness

Construction I: re-keying MAC



- Pragmatism: requires one leak-free block cipher execution for initialization (cfr. slide 8)
  - Then takes advantage of statefullness
- F expected to be (much) more efficient than F*

Construction II: hash-then-MAC

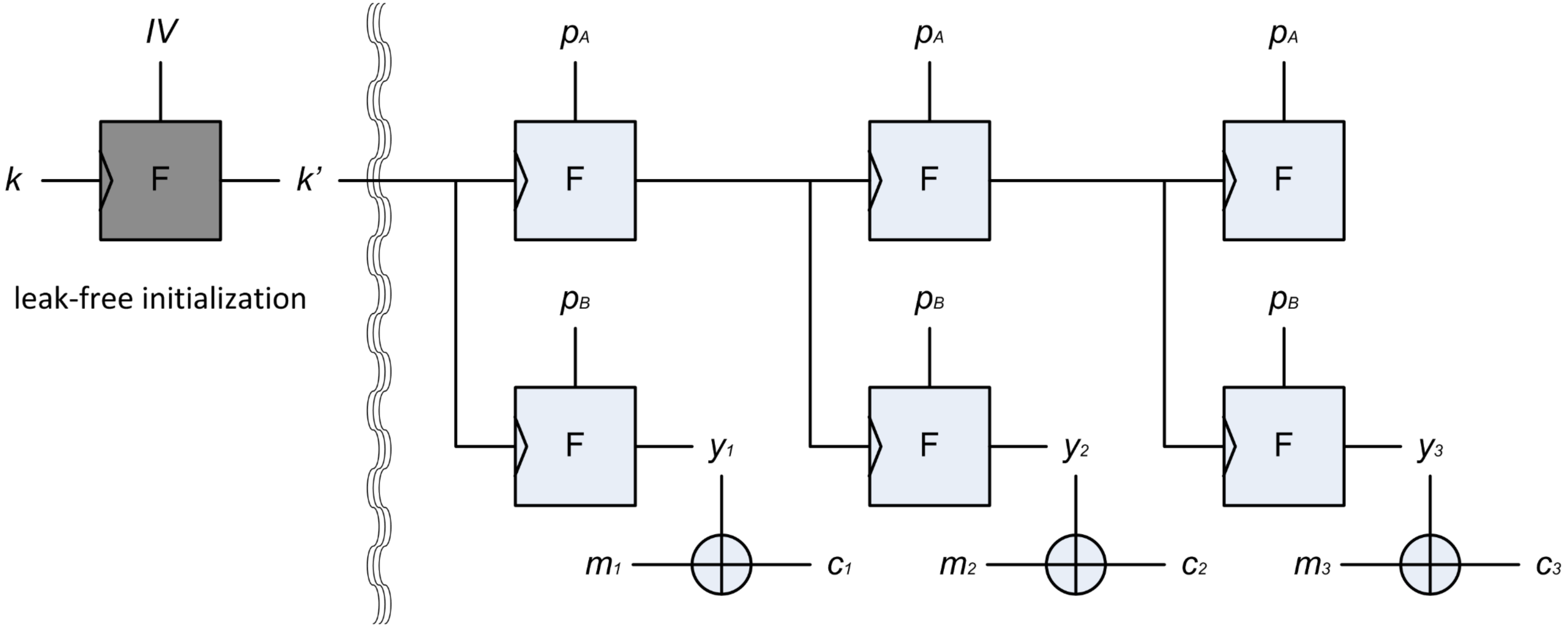


leak-free initialization

leaking block cipher execution

- Conceptually simpler (but requires a hash function)

Encryption: construction



- Essentially the LR-PRG as a stream cipher

Encryption: security definition

- Conceptual problem: distinguishing is always easy if L is given in the challenge phase

Encryption: security definition

- Conceptual problem: distinguishing is always easy if L is given in the challenge phase
- Theoretical approach: exclude L in the challenge phase (which is not justified in practice)

Encryption: security definition

- Conceptual problem: distinguishing is always easy if L is given in the challenge phase
- Theoretical approach: exclude L in the challenge phase (which is not justified in practice)
- Our (pragmatic) approach: admit semantic security is impossible. Leakage will always allow distinguishing plaintexts/ciphertexts!

Encryption: security definition

- Conceptual problem: distinguishing is always easy if L is given in the challenge phase
- Theoretical approach: exclude L in the challenge phase (which is not justified in practice)
- Our (pragmatic) approach: admit semantic security is impossible. Leakage will always allow distinguishing plaintexts/ciphertexts!

- CPA security reduction: security of R rounds reduces to security of 1 round (independent of what we can actualy achieve for 1 round)
  - *See our CCS 2015 paper for the details*

Outline

- Starting point (link with previous lecture)
- Seed results (TCC 2004, FOCS 2008)
- Primitives (PRGs/PRFs,PRPs)
  - If you don’t care about proofs
    - The stateful/stateless separation
  - The proof/assumptions challenge
    - Ensuring independence
    - Bounding the leakage
    - The simulatable leakage attempt
- « Pragmatic » auth. & encryption (CCS 2015)
- **Back to stateless primitives**
- Conclusions & open problems

How to instantiate the leak-free BC?

- Mask the AES (or masking-oriented ciphers)
  - But overheads always quadratic in d

How to instantiate the leak-free BC?

- Mask the AES (or masking-oriented ciphers)
  - But overheads always quadratic in *d*

- Use non-standard constructions
  - Heuristic (easy-to-mask) fresh re-keying
  - GGM PRF with chosen plaintexts

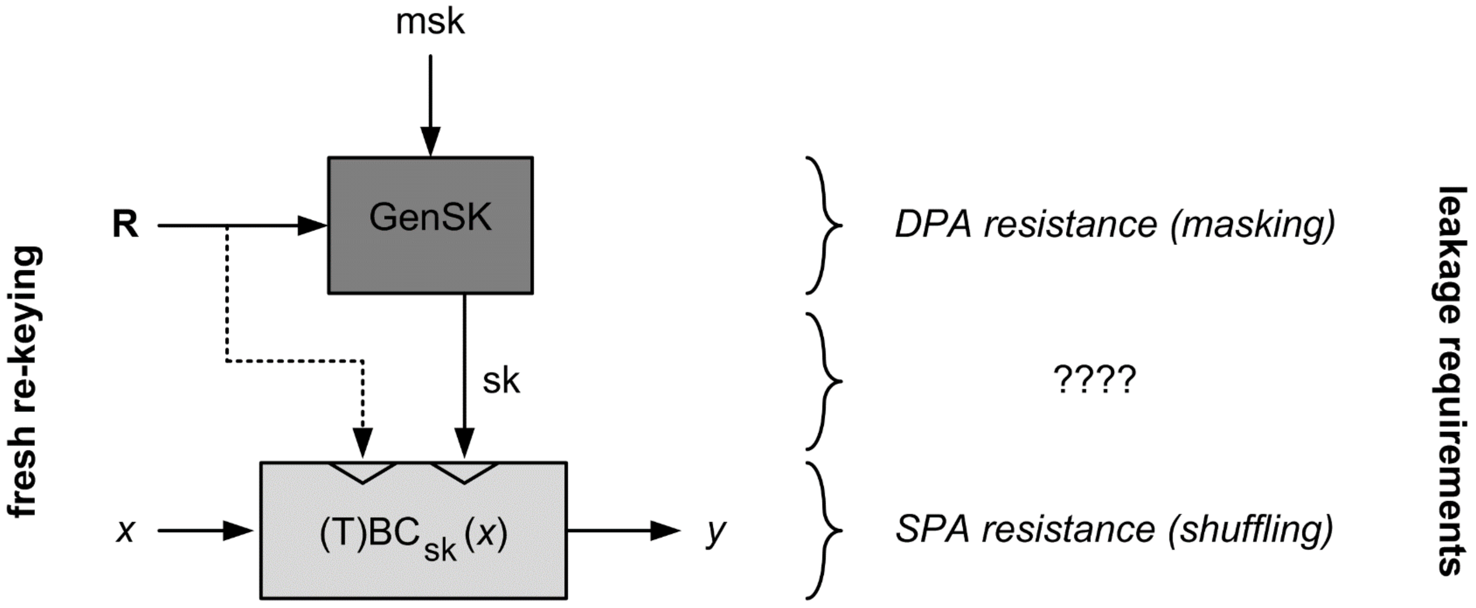
How to instantiate the leak-free BC?

- Mask the AES (or masking-oriented ciphers)
  - But overheads always quadratic in *d*

- Use non-standard constructions
  - Heuristic (easy-to-mask) fresh re-keying
  - GGM PRF with chosen plaintexts

- Exploit homomorphisms in asymmetric crypto
  - Overheads linear in *d* (but large for small *d*’s)

A recent proposal (Crypto 2016)

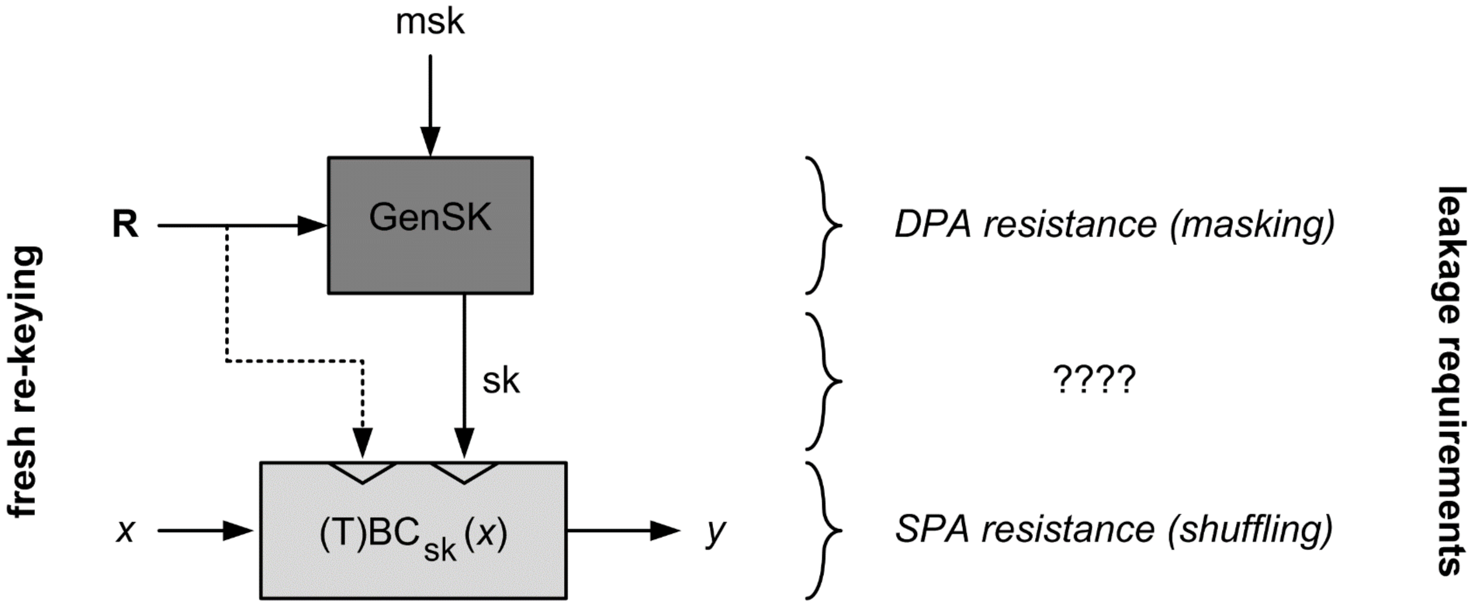


DPA resistance (masking)

????

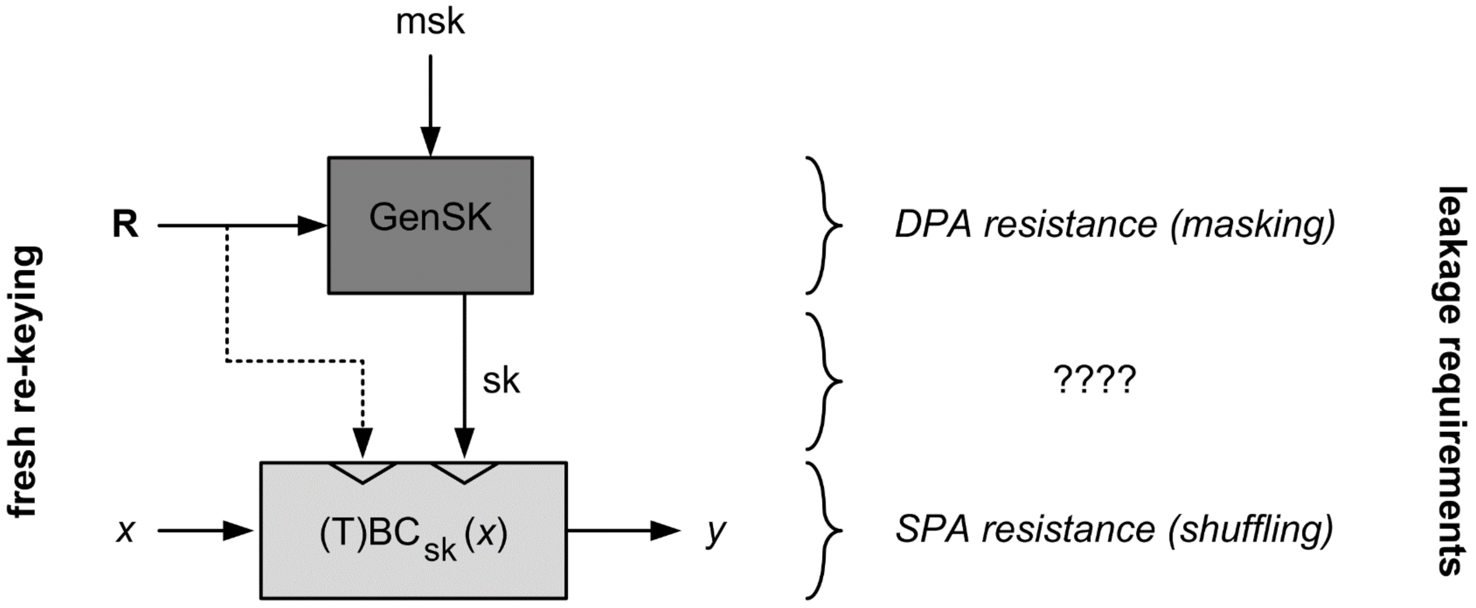
SPA resistance (shuffling)

A recent proposal (Crypto 2016)



- Cryptographically strong re-keying function
  - $\mathrm{sk} =< \mathbf{R}, \mathrm{msk} >= \sum(< \mathbf{R}, \mathrm{msk}_i >)$

A recent proposal (Crypto 2016)



- Cryptographically strong re-keying function
  - $\mathrm{sk} =< \mathbf{R}, \mathrm{msk} >= \sum(< \mathbf{R}, \mathrm{msk}_i >)$
- Security based on hard lattice problems
- Simple & efficient: all computations in GF(2^m)

Outline

- Starting point (link with previous lecture)
- Seed results (TCC 2004, FOCS 2008)
- Primitives (PRGs/PRFs,PRPs)
  - If you don’t care about proofs
    - The stateful/stateless separation
  - The proof/assumptions challenge
    - Ensuring independence
    - Bounding the leakage
    - The simulatable leakage attempt
- « Pragmatic » auth. & encryption (CCS 2015)
- Back to stateless primitives
- **Conclusions & open problems**

Conclusions

- Concretely, leakage-resilience is effective and efficient for stateful primitives such as PRGs

Conclusions

- Concretely, leakage-resilience is effective and efficient for stateful primitives such as PRGs

- Protection of stateless primitives such as PRFs and PRPs is much more challenging

Conclusions

- Concretely, leakage-resilience is effective and efficient for stateful primitives such as PRGs
- Protection of stateless primitives such as PRFs and PRPs is much more challenging
- Pragmatic solution: minimize the number of (leak-free) stateless primitives in leakage-resilient encryption and authentication

Open problems

- Sound (empirically falsifiable) assumptions
  - e.g. new instances of leakage simulators

- Can we better formalize CPA security with L?

- Leakage-resilient decryption & tag verification
  - Excluded from the analysis so far
  - Mostly because of IV control by the Adv.

- Leakage-resilient authenticated encryption

THANKS

http://perso.uclouvain.be/fstandae/

Related publications & further readings. Masking (slide 1). ***Security graph.*** Alexandre Duc, Sebastian Faust, François-Xavier Standaert: *Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device*. EUROCRYPT (1) 2015: 401-429. ***Performance figures.*** Vincent Grosso, François-Xavier Standaert, Sebastian Faust: *Masking vs. multiparty computation: how large is the gap for AES?* J. Cryptographic Engineering 4(1): 47-57 (2014). ***Physically observable cryptography (slide 3).*** Silvio Micali, Leonid Reyzin: *Physically Observable Cryptography (Extended Abstract).* TCC 2004: 278-296. **Leakage-resilient cryptography (slide 4).** Stefan Dziembowski, Krzysztof Pietrzak: *Leakage-Resilient Cryptography.* FOCS 2008: 293-302. **Threshold implementations (Slide 4).** Svetla Nikova, Vincent Rijmen, Martin Schläffer: Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. J. Cryptology 24(2): 292-321 (2011). **Stateful PRGs (slide 5).** Mihir Bellare, Bennet S. Yee: Forward-Security in Private-Key Cryptography. CT-RSA 2003: 1-18. **Stateless PRFs (slide 6).** Oded Goldreich, Shafi Goldwasser, Silvio Micali: *How to Construct Random Functions (Extended Abstract).* FOCS 1984: 464-479. **Stateless/stateful separation (slide 7).** Sonia Belaïd, Vincent Grosso, François-Xavier Standaert: Masking and leakage-resilient primitives: One, the other(s) or both? Cryptography and Communications 7(1): 163-184 (2015). **FOCS 2008/Eurocrypt 2009 stream ciphers (slide 9).** Stefan Dziembowski, Krzysztof Pietrzak: *Leakage-Resilient Cryptography.* FOCS 2008: 293-302. Krzysztof Pietrzak: *A Leakage-Resilient Mode of Operation*. EUROCRYPT 2009: 462-482. **CCS 2010 PRG (slide 10).** Yu Yu, François-Xavier Standaert, Olivier Pereira, Moti Yung: *Practical leakage-resilient pseudorandom generators.* ACM Conference on Computer and Communications Security 2010: 141-151. **CHES 2012 PRG (slide 11).** Sebastian Faust, Krzysztof Pietrzak, Joachim Schipper: Practical Leakage-Resilient Symmetric Cryptography. CHES 2012: 213-232. **CT-RSA 2013 PRG (slide 12).** Yu Yu, François-Xavier Standaert: *Practical Leakage-Resilient Pseudorandom Objects with Minimum Public Randomness.* CT-RSA 2013: 223-238. **Random oracle assumption (slides 13-14).** Yu Yu, François-Xavier Standaert, Olivier Pereira, Moti Yung: *Practical leakage-resilient pseudorandom generators.* ACM Conference on Computer and Communications Security 2010: 141-151. Christophe Petit, François-Xavier Standaert, Olivier Pereira, Tal Malkin, Moti Yung: *A block cipher based pseudo random number generator secure against side-channel key recovery.* ASIACCS 2008: 56-65. P. Kocher. *Leak resistant cryptographic indexed key update*. US Patent 6539092. **Leakage-resilient PRFs (slide 15).** François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, Elisabeth Oswald: *Leakage Resilient Cryptography in Practice.* Towards Hardware-Intrinsic Security 2010: 99-134. Yevgeniy Dodis, Krzysztof Pietrzak: *Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks.* CRYPTO 2010: 21-40. Sebastian Faust, Krzysztof Pietrzak, Joachim Schipper: Practical Leakage-Resilient Symmetric Cryptography. CHES 2012: 213-232. Yu Yu, François-Xavier Standaert: *Practical Leakage-Resilient Pseudorandom Objects with Minimum Public Randomness.* CT-RSA 2013: 223-238. Michel Abdalla, Sonia Belaïd, Pierre-Alain Fouque: *Leakage-Resilient Symmetric Encryption via Re-keying.* CHES 2013: 471-488. **Bounded range leakage / HILL pseudoentropy (slides 16 and 18).** *Leakage-Resilient Cryptography.* FOCS 2008: 293-302. François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, Elisabeth Oswald: *Leakage Resilient Cryptography in Practice.* Towards Hardware-Intrinsic Security 2010: 99-134. **Simulatable leakage assumption (slides 20-28).** François-Xavier Standaert, Olivier Pereira, Yu Yu: *Leakage-Resilient Symmetric Cryptography under Empirically Verifiable Assumptions.* CRYPTO (1) 2013: 335-352. **Bristol distringuisher (slide 25).** Jake Longo, Daniel P. Martin, Elisabeth Oswald, Daniel Page, Martijn Stam, Michael Tunstall: *Simulatable Leakage: Analysis, Pitfalls, and New Constructions.* ASIACRYPT (1) 2014: 223-242. **Leakage-resilient authentication & encryption (slides 29-34).** Olivier Pereira, François-Xavier Standaert, Srinivas Vivek: *Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives*. ACM Conference on Computer and Communications Security 2015: 96-108. **Leakage exclusion for challenge queries (slide 34).** Moni Naor, Gil Segev: *Public-Key Cryptosystems Resilient to Key Leakage*. CRYPTO 2009: 18-35. Carmit Hazay, Adriana López-Alt, Hoeteck Wee, Daniel Wichs: *Leakage-Resilient Cryptography from Minimal Assumptions.* EUROCRYPT 2013: 160-176. Michel Abdalla, Sonia Belaïd, Pierre-Alain Fouque: *Leakage-Resilient Symmetric Encryption via Re-keying.* CHES 2013: 471-488. **Instantiations of a leak-free block cipher (slide 35).** ***Masking.*** Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, Kerem Varici: *LS-Designs: Bitslice Encryption for Efficient Masked Software Implementations*. FSE 2014: 18-37. ***Fresh re-keying.*** B. Gammel, W. Fischer, and S. Mangard. Generating a Session Key for Authentication and Secure Data Transfer. US Patent App. 14/074,279. Nov. 2013. Marcel Medwed, François-Xavier Standaert, Johann Großschädl, Francesco Regazzoni: *Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices.* AFRICACRYPT 2010: 279-296. Christoph Dobraunig, François Koeune, Stefan Mangard, Florian Mendel, François-Xavier Standaert: *Towards Fresh and Hybrid Re-Keying Schemes with Beyond Birthday Security.* CARDIS 2015: 225-241. ***GGM PRF with chosen plaintexts.*** Marcel Medwed, François-Xavier Standaert, Antoine Joux: *Towards Super-Exponential Side-Channel Security with Efficient Leakage-Resilient PRFs*. CHES 2012: 193-212. ***Asymmetric cryptography.*** Eike Kiltz, Krzysztof Pietrzak: *Leakage Resilient ElGamal Encryption*. ASIACRYPT 2010: 595-612. Daniel P. Martin, Elisabeth Oswald, Martijn Stam, Marcin Wójcik: *A Leakage Resilient MAC*. IMA Int. Conf. 2015: 295-310. **Crypto 2016 re-keying schemes (slide 36).** Stefan Dziembowski, Sebastian Faust, Gottfried Herold, Anthony Journault, Daniel Masny, Francois-Xavier Standaert: *Towards Sound Fresh Re-Keying with Hard (Physical) Learning Problems.* IACR Cryptology ePrint Archive 2016: 573 (2016).

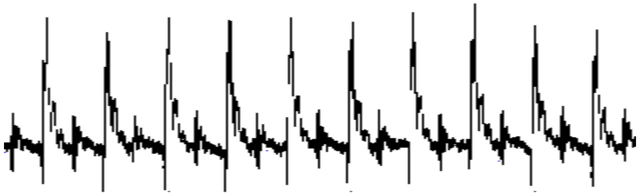
Additional slides

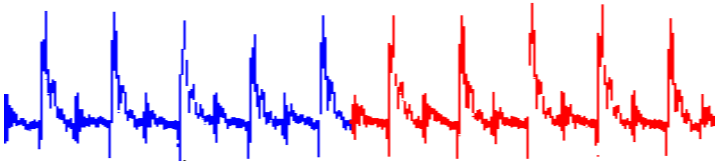
(leakage simulators & the Bristol distinguisher)

Background

- Split & Concatenate Simulator (CRYPTO 2013)

$$\mathrm{L}(x, k, y) \approx \mathrm{L}(x, \tilde{k}, y^*) || \mathrm{L}(x^*, \tilde{k}, y)$$





Background

- Split & Concatenate Simulator (CRYPTO 2013)

$$\mathrm{L}(x, k, y) \quad \approx \quad \mathrm{L}(x, \tilde{k}, y^*) || \mathrm{L}(x^*, \tilde{k}, y)$$

- Longo Galea et al (ASIACRYPT 2014): ∃ correlation between samples *within* real traces (e.g. $\rho > 0.5$) … that are significantly reduced in simulated ones

  ⇒ Allows distinguishing!

Background

- Split & Concatenate Simulator (CRYPTO 2013)

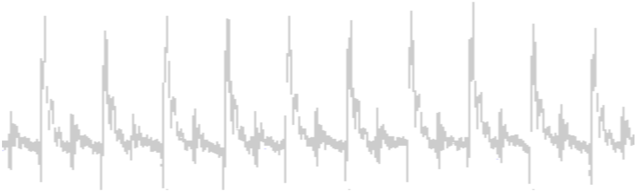
$$\mathrm{L}(x, k, y) \qquad \approx \qquad \mathrm{L}(x, \tilde{k}, y^*)||\mathrm{L}(x^*, \tilde{k}, y)$$

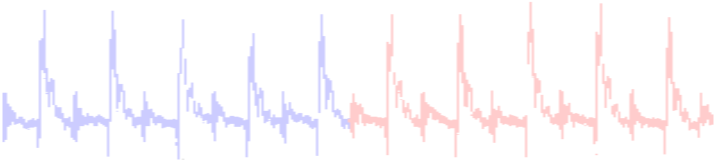
- Longo Galea et al (ASIACRYPT 2014): ∃ correlation between samples *within* real traces (e.g. $\rho > 0.5$)
… that are significantly reduced in simulated ones
⇒ Allows distinguishing!
- Proposed solution: very noisy implementations, *but it scales badly*: noise arbitrarily reduced with averaging

Background

- Split & Concatenate Simulator (CRYPTO 2013)

$$\mathrm{L}(x, k, y) \approx \mathrm{L}(x, \tilde{k}, y^*) || \mathrm{L}(x^*, \tilde{k}, y)$$





- Longo Galea et al (ASIACRYPT 2014): ∃ correlation between samples *within* real traces (e.g. $\rho > 0.5$) … that are significantly reduced in simulated ones

  ⇒ Allows distinguishing!

- Proposed solution: very noisy implementations, *but it scales badly*: noise arbitrarily reduced with averaging

Can we do better?

Origin of the intra-trace correlation

- Algorithmic? Unlikely: $\rho\left(x, \mathrm{Sbox}(x)\right) \ll 0.5$

Origin of the intra-trace correlation

- Algorithmic? Unlikely: $\rho\big(x, \mathrm{Sbox}(x)\big) \ll 0.5$
- Physical then ⇒ let’s use a simple physical model

Origin of the intra-trace correlation

- Algorithmic? Unlikely: $\rho\left(x, \mathrm{Sbox}(x)\right) \ll 0.5$
- Physical then ⇒ let's use a simple physical model

$$\mathrm{L}(x,k,y) = \underbrace{\delta(x,k,y)}_{\text{signal}} + \underbrace{N}_{\text{noise}}$$

Origin of the intra-trace correlation

- Algorithmic? Unlikely: $\rho\big(x, \mathrm{Sbox}(x)\big) \ll 0.5$
- Physical then ⇒ let's use a simple physical model

$$\mathrm{L}(x,k,y) = \underbrace{\delta(x,k,y)}_{\text{signal}} + \underbrace{N}_{\text{noise}}$$

⇒ Does the correlation come from signal or noise?

Origin of the intra-trace correlation

- Algorithmic? Unlikely: $\rho\big(x, \mathrm{Sbox}(x)\big) \ll 0.5$
- Physical then ⇒ let's use a simple physical model

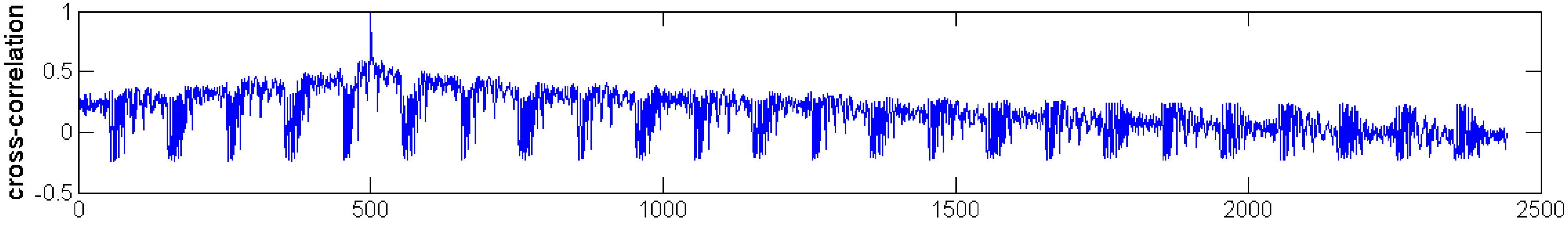
$$\mathrm{L}(x,k,y) = \underbrace{\delta(x,k,y)}_{\text{signal}} + \underbrace{N}_{\text{noise}}$$

⇒ Does the correlation come from signal or noise?

- In particular for *large parallel implementations* (since we know 8-bit AES implementations can be broken in one trace anyway – see SASCA paper)

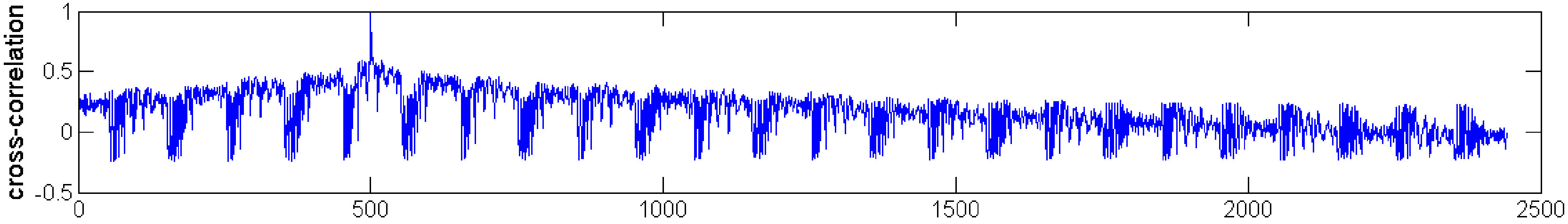
Repeating experiments with a 65nm ASIC

- Intra-trace correlation (real traces, sample 500)

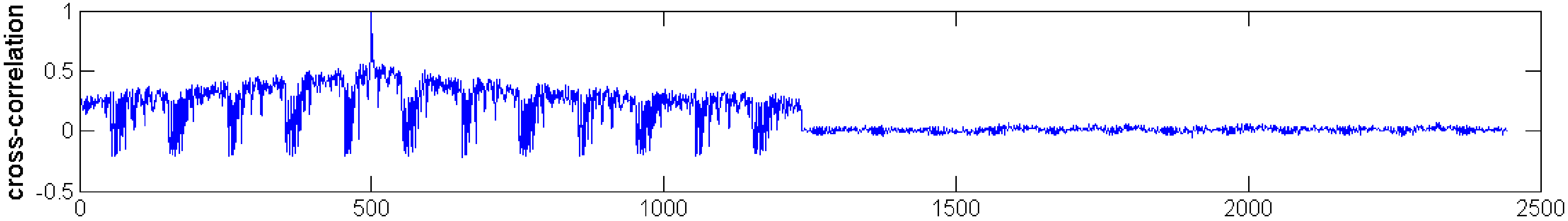


Repeating experiments with a 65nm ASIC

- Intra-trace correlation (real traces, sample 500)

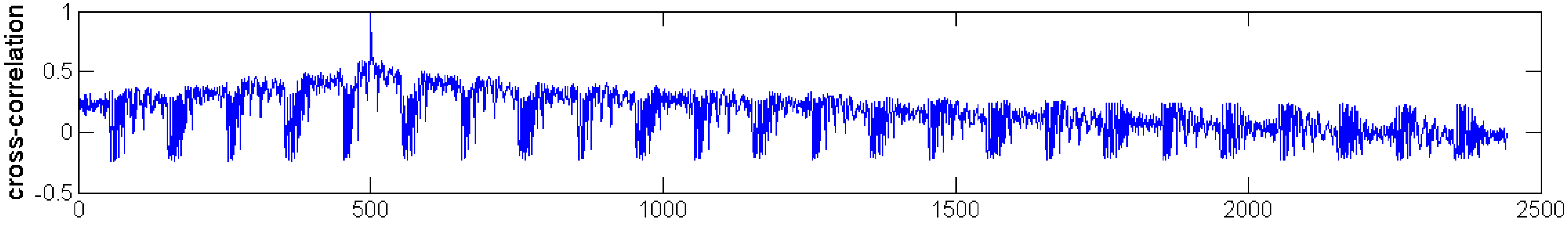


- Same, with simulated traces $\mathrm{L}(x, \tilde{k}, y^{*})||\mathrm{L}(x^{*}, \tilde{k}, y)$

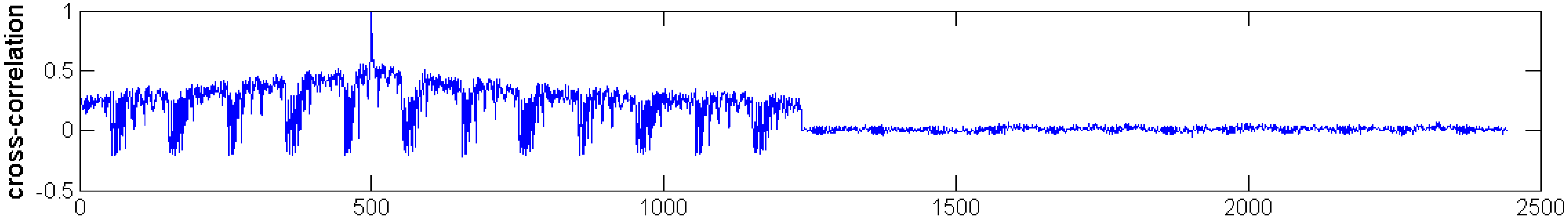


Repeating experiments with a 65nm ASIC

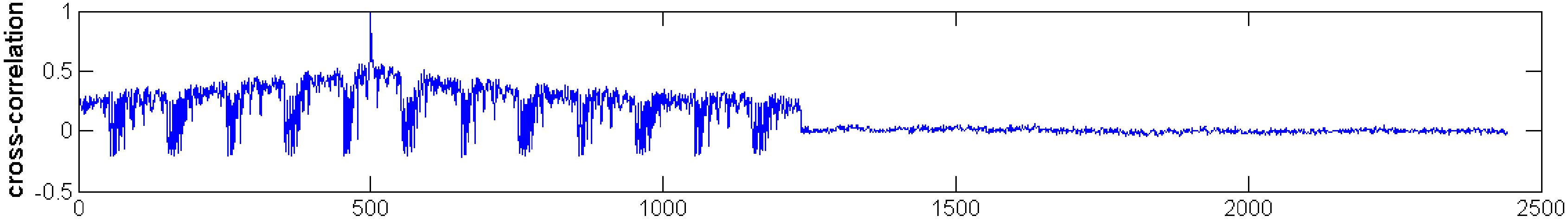
- Intra-trace correlation (real traces, sample 500)



- Same, with simulated traces $\mathrm{L}(x,\tilde{k},y^*)||\mathrm{L}(x^*,\tilde{k},y)$

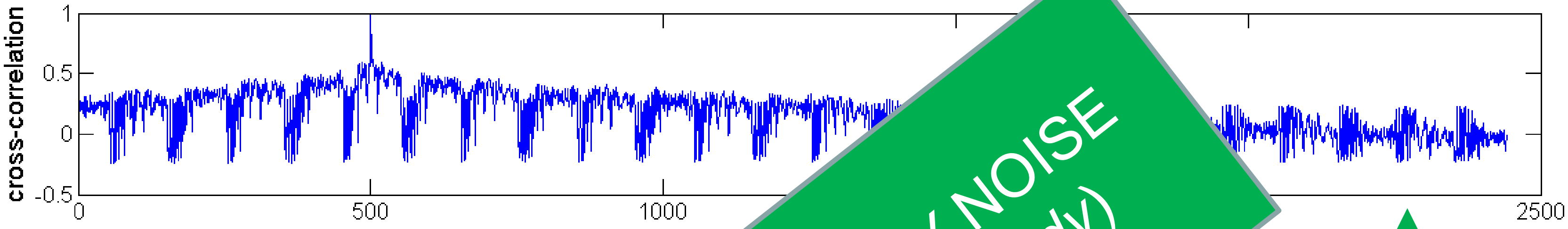


- & fake simulated traces $\delta(x,k,y)+N_1||\delta(x,k,y)+N_2$

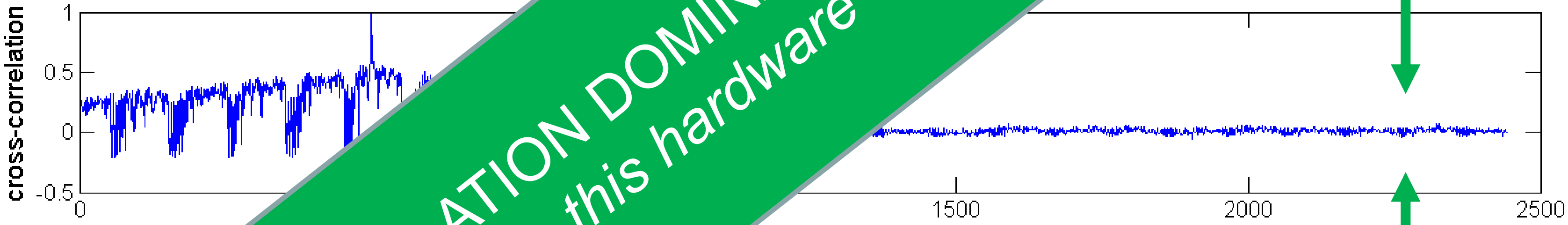


Repeating experiments with a 65nm ASIC

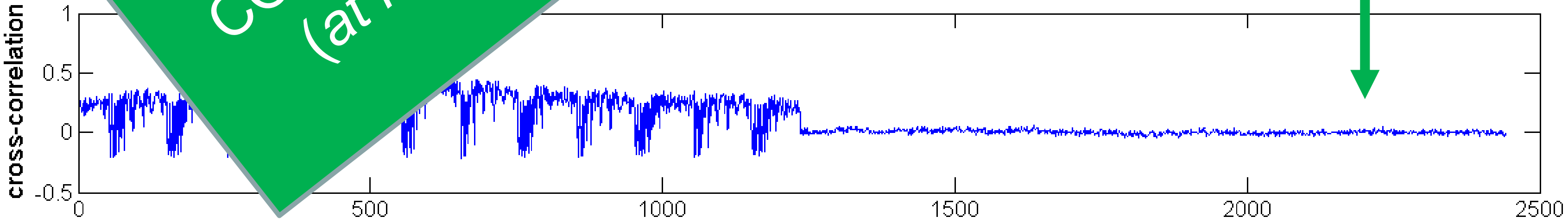
- Intra-trace correlation (real traces, sample 500)



- Same, with simulat $y^*)||\mathrm{L}(x^*,\tilde{k},y)$



- & fak es $\delta(x,k,y)+N_1||\delta(x,k,y)+N_2$



CORRELATION DOMINATED BY NOISE
(at least in this hardware case study)

A first improvement

- Sliding simulator

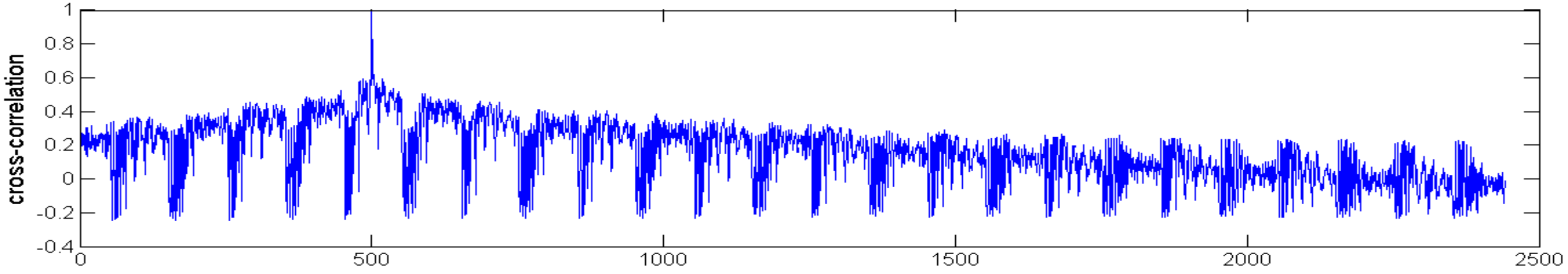
$L(x, \tilde{k}, y^*) \cdot$ ◣ $+$ $L(x^*, \tilde{k}, y) \cdot$ ◢

A first improvement

- Sliding simulator

$L(x,\tilde{k},y^*) \cdot$ ◣ $+$ $L(x^*,\tilde{k},y) \cdot$ ◢

- Real traces

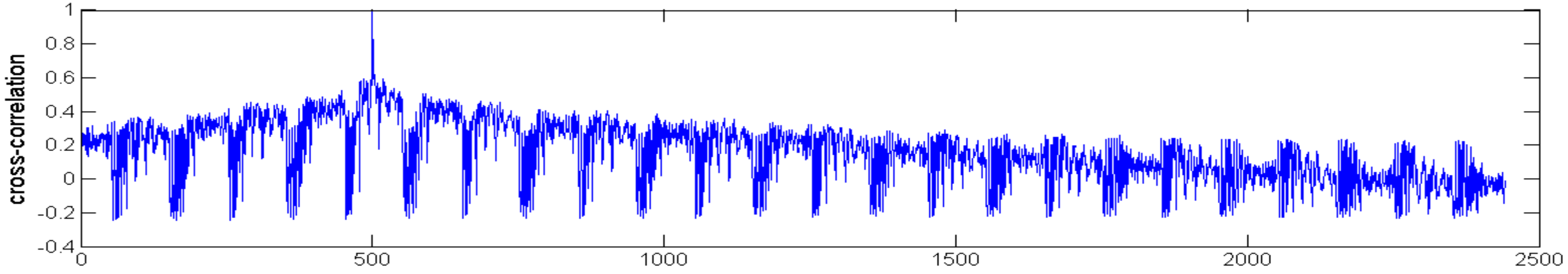


A first improvement

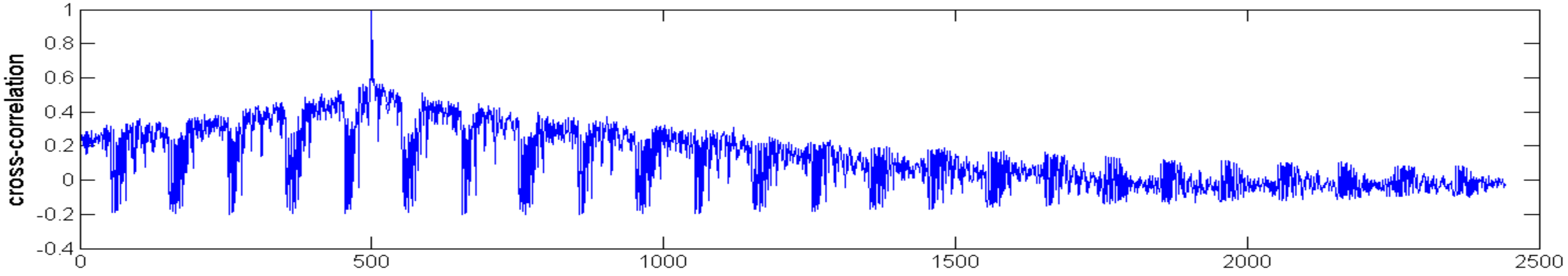
- Sliding simulator

$$L(x, \tilde{k}, y^*) \cdot \text{◣} + L(x^*, \tilde{k}, y) \cdot \text{◢}$$

- Real traces



- Simulated traces

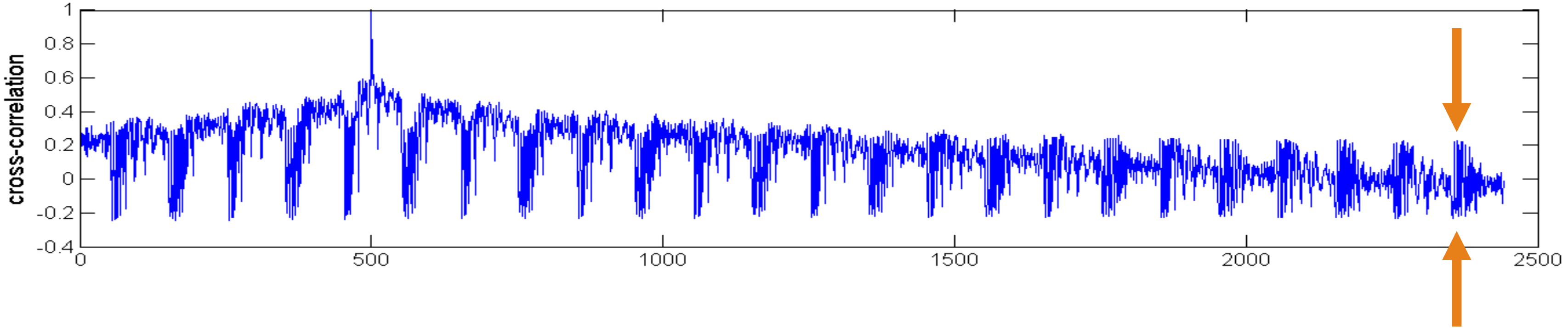


A first improvement

- Sliding simulator

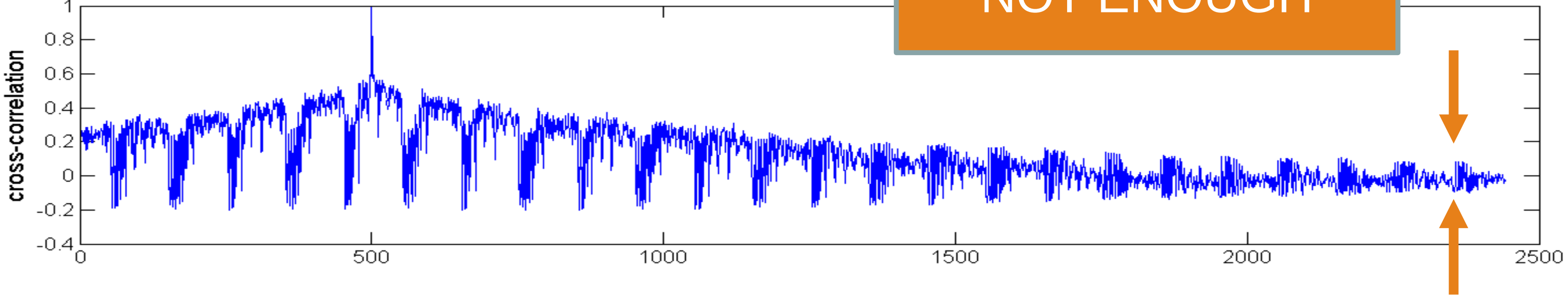
$L(x,\tilde{k},y^{*}) \cdot$ ◣ + $L(x^{*},\tilde{k},y) \cdot$ ◢

- Real traces



- Simulated traces

NOT ENOUGH



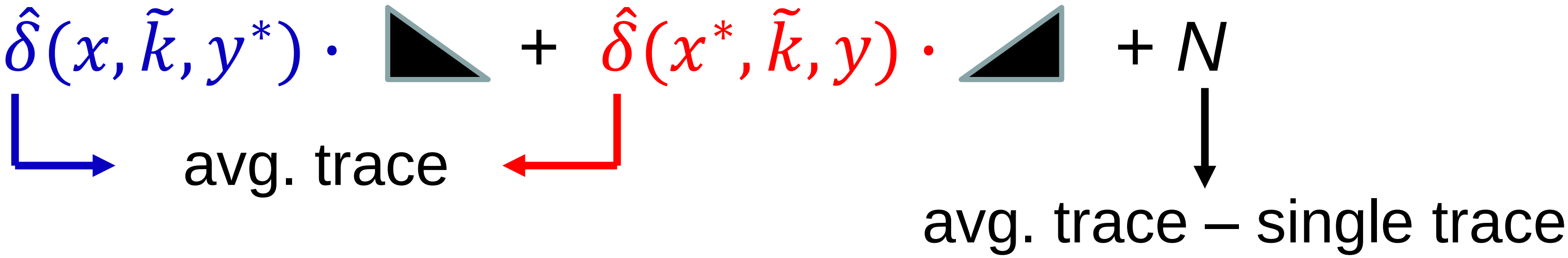
Another idea: separate signal and noise

- Sliding signal + noise simulator

$\hat{\delta}(x, \tilde{k}, y^*) \cdot$ ◣ $+$ $\hat{\delta}(x^*, \tilde{k}, y) \cdot$ ◢ $+ N$

Another idea: separate signal and noise

- Sliding signal + noise simulator



$\hat{\delta}(x, \tilde{k}, y^*) \cdot$ ◣ $+$ $\hat{\delta}(x^*, \tilde{k}, y) \cdot$ ◢ $+ N$

avg. trace

avg. trace – single trace

Another idea: separate signal and noise

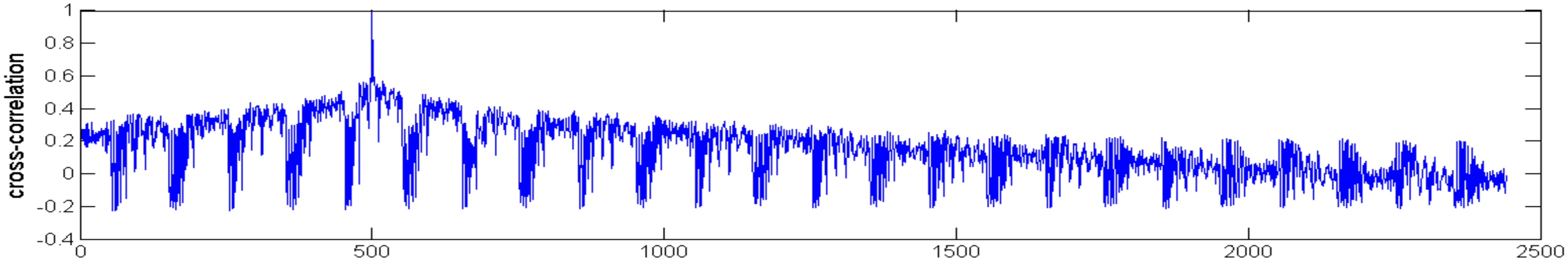
- Sliding signal + noise simulator

$\hat{\delta}(x,\tilde{k},y^*) \cdot$ ◣ + $\hat{\delta}(x^*,\tilde{k},y) \cdot$ ◢ + N

avg. trace

avg. trace – single trace

- Real traces



Another idea: separate signal and noise

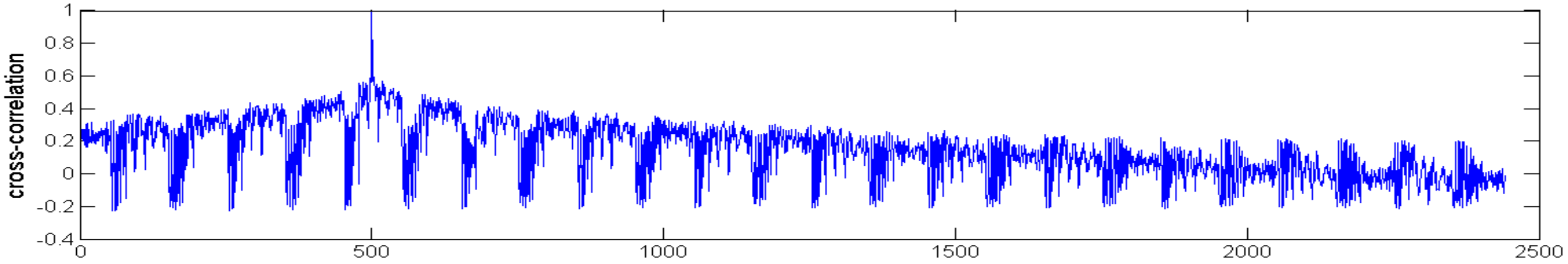
- Sliding signal + noise simulator

$$\hat{\delta}(x,\tilde{k},y^{*})\cdot ◣ + \hat{\delta}(x^{*},\tilde{k},y)\cdot ◢ + N$$

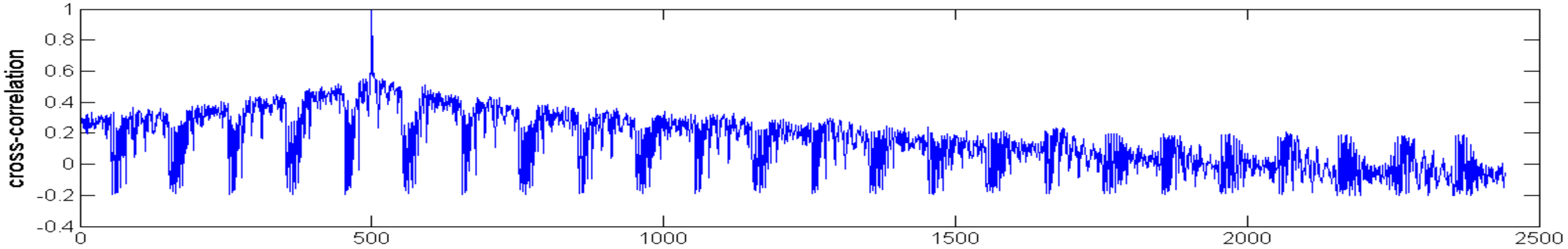
avg. trace

avg. trace – single trace

- Real traces

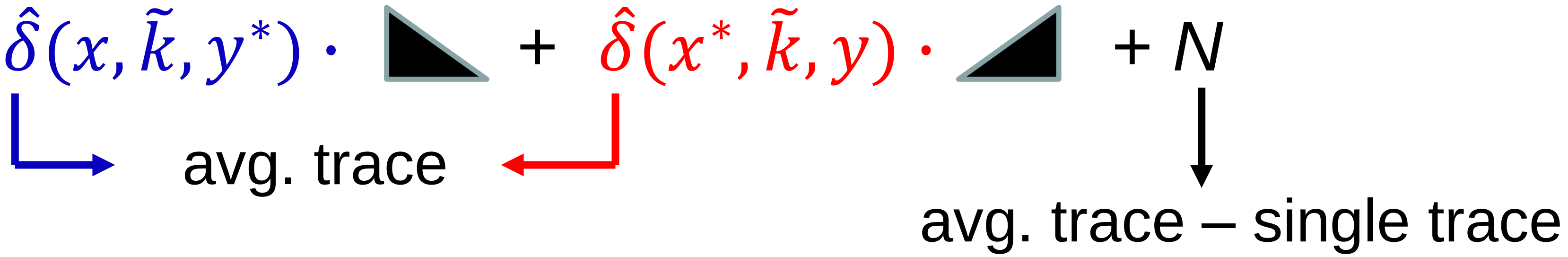


- Simulated traces



Another idea: separate signal and noise

- Sliding signal + noise simulator

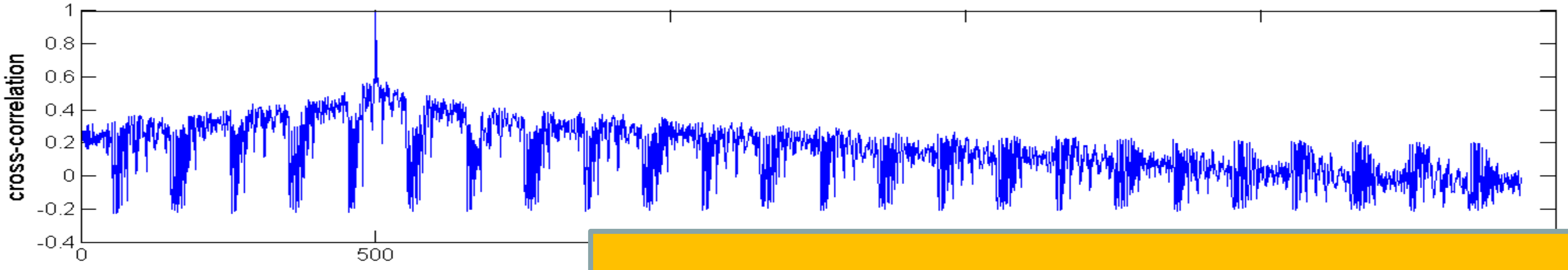


$\hat{\delta}(x,\tilde{k},y^{*})\cdot$ ◣ $+$ $\hat{\delta}(x^{*},\tilde{k},y)\cdot$ ◢ $+ N$

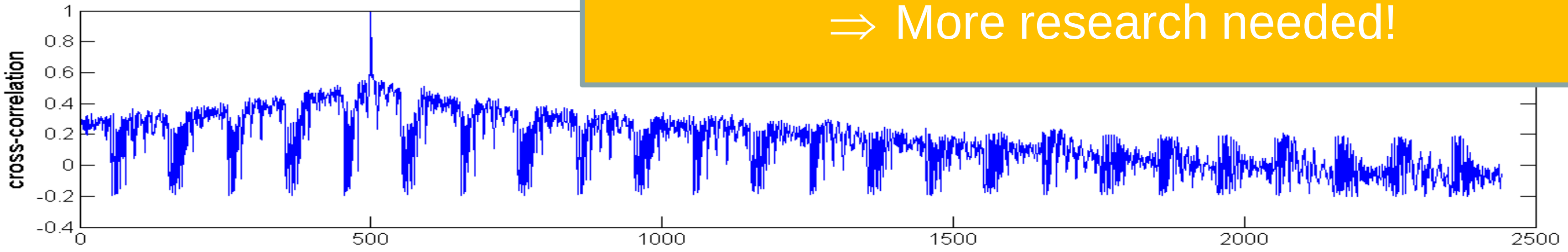
avg. trace

avg. trace – single trace

- Real traces



- Simulated traces



LOOKS BETTER

(but probably not enough for low-freq. events)

$\Rightarrow$ More research needed!