

Leakage Resilient Cryptography: (a Practical Overview)

F.-X. Standaert

UCL Crypto Group, Université catholique de Louvain

SKEW 2011 - Copenhagen, Denmark - February 2011



Acknowledgements

- ▶ Sebastian Faust
- ▶ Tal Malkin
- ▶ Elisabeth Oswald
- ▶ Olivier Pereira
- ▶ Moti Yung
- ▶ Yu Yu
- ▶ ...



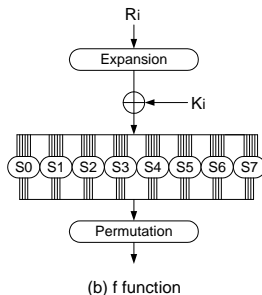
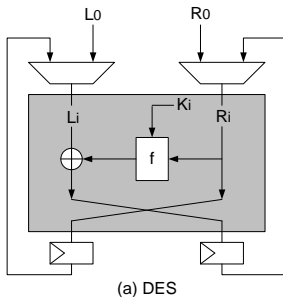
Side-Channel Attacks

- ▶ Take advantage of physical information leakage
- ▶ Leakage is device-dependent
- ▶ But any device shows leakage
- ▶ Less generic but more powerful than computational (e.g., linear, differential) cryptanalysis



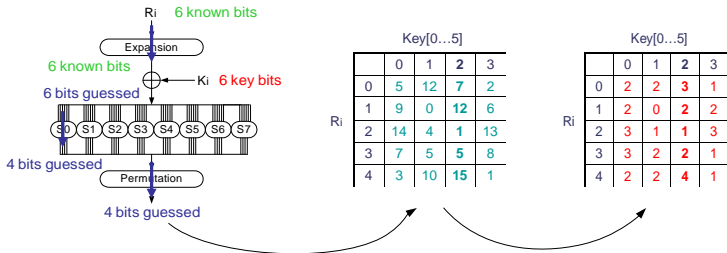
Exemplary attack against the DES

- ▶ The Data Encryption Standard
- ▶ FPGA implementation, loop architecture



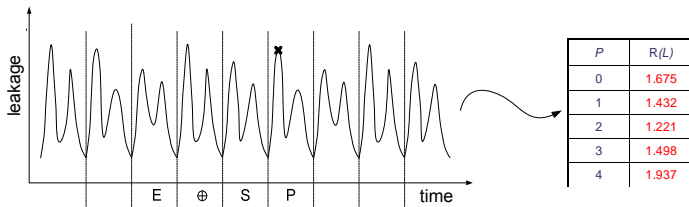
Exemplary attack against the DES

1. Input selection: random plaintexts
2. Internal values derivation
3. Leakage modeling (Hamming weights)



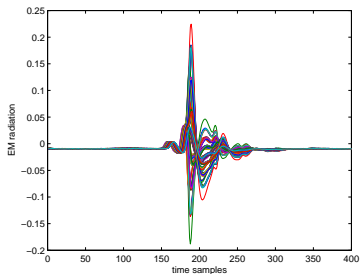
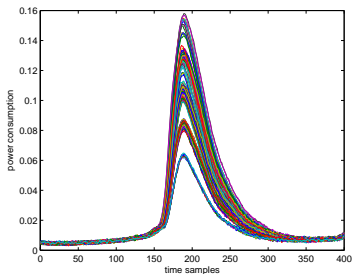
Exemplary attack against the DES

4. Leakage measurement
5. Leakage reduction (select representative samples)



Exemplary attack against the DES

- ▶ In practice, power consumption vs. EM radiation

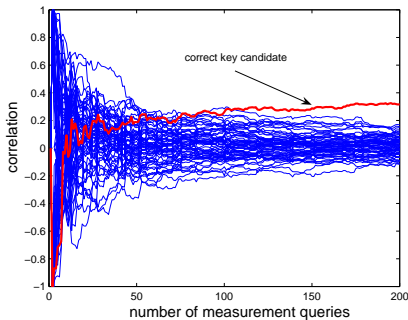


Exemplary attack against the DES

6. Statistical test

- e.g. correlation coefficient

| Key[0...5] | 0 | 1 | 2 | 3 |
|------------|-------|------|-------------|-------|
| corr | -0.09 | 0.05 | 0.32 | -0.11 |



Improved attacks

- ▶ Adaptive selection of the inputs
- ▶ Pre-processing of the traces (e.g. averaging, filtering)
- ▶ Improved leakage models by profiling, characterization
- ▶ Exploitation of multiple samples, multivariate statistics
 - ▶ Higher-order attacks
 - ▶ Template attacks
- ▶ Different statistical tests
 - ▶ Difference of mean
 - ▶ Correlation analysis
 - ▶ Bayesian classification
- ▶ ...



Countermeasures

- ▶ Implementation level (CHES-like), e.g.
 - ▶ Masking
 - ▶ Dual-rail logic styles
 - ▶ Time randomization
- ▶ Cryptographic level (TCC-like), e.g.
 - ▶ Physically observable crypto [MR04]
 - ▶ Leakage-resilient cryptography [DP08]
 - ▶ Bounded retrieval model [CLW06,D06]
 - ▶ Auxiliary input model [DKL09]



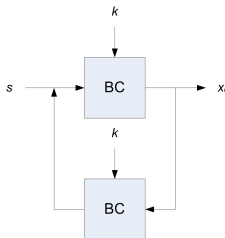
Countermeasures

- ▶ Goal: under certain conditions, the attacks' complexity should increase exponentially with a security parameter
- ▶ e.g. masking: security against DPA increases exponentially in the number of shares (given a sufficient amount of noise in the measurements)
- ▶ Cryptographic level's big claim: consider all PT adversaries (rather than some ad hoc ones)
- ▶ Note: evaluation ad hoc SCAs is not trivial [SMY09]

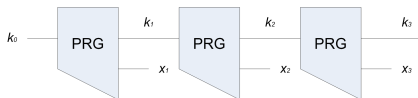


Crypto level's pros

- ▶ More formal security guarantee
- ▶ Design crypto with SCAs in mind can help, e.g.



ANSI X9.17 PRG



stateful PRG

⇒ Ask less to HW designers (protect 1 vs. q iterations)



Open issues in leakage resilience

“Does leakage resilience capture practical SCAs?”

- ▶ Issue 1: cost
- ▶ Issue 2: assumptions
 - A1. Polynomial time vs. AC0 leakage functions
 - A2. Adaptive vs. non adaptive leakage functions
 - A3. Random oracle based assumptions
 - A4. Limited information leakage
 - ▶ Bounded space
 - ▶ HILL pseudoentropy
 - ▶ Auxiliary input, seed preserving, ...



Open issues in leakage resilience

- ▶ Issue 2: assumptions (cont.)
 - A5. Independent leakage
 - A6. Only computation leaks
 - A7. Simulatable leakage
 - A8. Secure precomputations
- ▶ Issue 3: instantiation
- ▶ Issue 4: initialization
- ▶ Issue 5: untight bounds

This talk's goal: try to formalize engineering constraints



Issue 1: cost



Issue 1: cost

- ▶ SCAs are a threat for low cost devices
- ▶ We need low cost countermeasures
- ▶ Implementation cost usually left out of analysis
- ▶ Cryptographers' (fair) answer:
 "today's expensive is tomorrow's low cost"
- ▶ Well... let's leave it out for now...



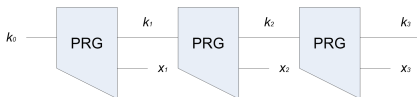
Issue 2: assumptions

- A1. Polynomial time vs. AC0 leakage functions
- A2. Adaptive vs. non adaptive leakage functions
- A3. Random oracle based assumptions
- A4. Limited information leakage
- A5. Independent leakage
- A6. Only computation leaks
- A7. Simulatable leakage
- A8. Secure precomputations



Poly. time vs. AC0 leakage functions

- ▶ Polynomial time leakage functions [MR04]
 - ▶ Overly strong adversaries: allows “future computation attacks”, i.e. leak one bit of k_3 while computing k_1



- ▶ Leakage function in the complexity class AC0 [F+10]
 - ▶ Do not capture the actual physical
 - ▶ e.g. no coupling (inner product) possible



Poly. time vs. AC0 leakage functions

- ▶ Summarizing: one is too strong, the other too weak
- ▶ Leakage functions cannot compute dozens of SHA3
- ▶ But they solve Maxwell's equations !
- ▶ e.g. on a standard desktop, simulating the power consumption of a single AES encryption with SPICE is much more complex than encrypting this plaintext
- ▶ Bounding the complexity of leakage functions hardly captures the realities of physical implementations
- ▶ Leakage functions are not simple, but they perform specific operations (like in the generic group model)



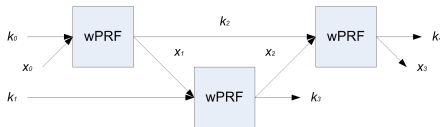
Issue 2: assumptions

- A1. Polynomial time vs. AC0 leakage functions
- A2. **Adaptive vs. non adaptive leakage functions**
- A3. Random oracle based assumptions
- A4. Limited information leakage
- A5. Independent leakage
- A6. Only computation leaks
- A7. Simulatable leakage
- A8. Secure precomputations



Adaptive leakage functions?

- ▶ Makes the adversary even stronger
 - ▶ e.g. allows one to accumulate several pieces of information leakage on the same future state
- ▶ Implies design tweaks to prevent the attack
 - ▶ e.g. alternating structure [DP08], [P09]



- ▶ Not efficient (doubled seed) and looks artificial



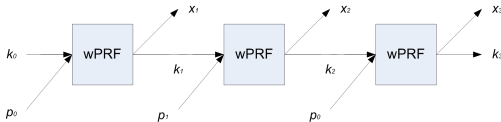
Adaptive leakage functions?

- ▶ In practice, the leakage function is usually a property of the target device (if the measurement setup is fixed)
- ▶ Only EM attacks allow moving the antenna on-the-fly
- ▶ More critical: the adaptivity of the leakage function anyway has to be prevented during initialization
 - ▶ Or full key leakage is possible with reset attacks
- ▶ Summarizing: non-adaptive leakages are more realistic
- ▶ The possible adaptivity of the meas. setup is better captured by increasing the information leakage (A4)



Adaptive leakage functions?

- ▶ + non adaptive leakage functions allow limiting the tweaks to face future computation attacks
- ▶ e.g. by using two public values p_0, p_1 chosen independently of the leakage function [YSPY10]



(some proof details still to be verified)



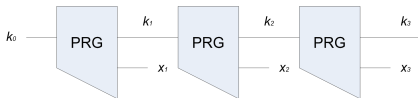
Issue 2: assumptions

- A1. Polynomial time vs. AC0 leakage functions
- A2. Adaptive vs. non adaptive leakage functions
- A3. **Random oracle based assumptions**
- A4. Limited information leakage
- A5. Independent leakage
- A6. Only computation leaks
- A7. Simulatable leakage
- A8. Secure precomputations



Random oracle based assumptions

- ▶ Assume PRG is a random oracle that can be queried by the adversary but not by the leakage function
- ▶ Allow proving “natural” constructions



- ▶ (with empirically verifiable assumptions, see later)
- ▶ (even with tight bounds [S+09], [YSPY10])



Random oracle based assumptions

- ▶ Summarizing: ROs are undesirable in theory
- ▶ But we use them differently than in black box proofs
- ▶ + ROs allow capturing many physical intuitions
- ▶ + they discriminate good and bad re-keying schemes
- ▶ Useful as a preliminary step (or more?)



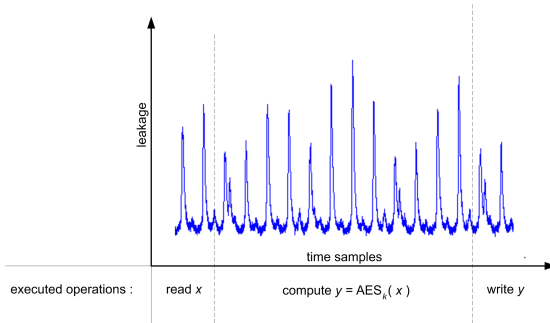
Issue 2: assumptions

- A1. Polynomial time vs. AC0 leakage functions
- A2. Adaptive vs. non adaptive leakage functions
- A3. Random oracle based assumptions
- A4. **Limited information leakage**
- A5. Independent leakage
- A6. Only computation leaks
- A7. Simulatable leakage
- A8. Secure precomputations



Bounded space

- ▶ $y = \text{AES}_k(x) \rightsquigarrow I$ with $|I|$ bounded
- ▶ But Adv. typically acquires data in the Gs/s rate
- ▶ \exists as many traces as there are x and k 's



Bounded space

- ▶ Summarizing: completely unrealistic
- ▶ Intuitively, leakages can be made of Gbits of data, but exploiting them may still be difficult. . .



HILL pseudoentropy

- ▶ Informally: $H_{\epsilon,s}^{HILL}[X|L] \geq n$ if \exists a distribution Y such that $H_{min}[Y|L] = n$ and Y is hard to distinguish from X with size s and advantage ϵ
- ▶ Assumption in [DP08]: $H_{\epsilon,s}^{HILL}[X|L] \geq n - \lambda$
- ▶ Can we guarantee this?
- ▶ Let $y_1 = \text{AES}_{k_1}(x) \rightsquigarrow l_1$ and $y_2 = \text{AES}_{k_2}(x) \rightsquigarrow l_2$. Having high HILL pseudoentropy requires that, given l_1, l_2 and k_i , it remains hard to predict i



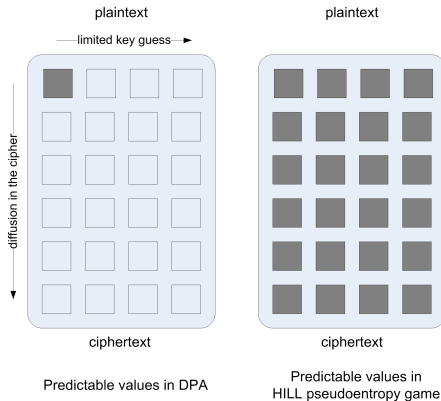
HILL pseudoentropy

- ▶ e.g. $L(k) = k[0 \dots 7] || H(k)$ implies $H_{\epsilon, S}^{HILL}[X|L] = 0$
- ▶ But it typically corresponds to a practical SCA, where the adversary predicts 8 bits (out of n) and the remaining bits constitute “algorithmic noise” (leakage that depends on a too large part of the key to be exploited in a divide-and-conquer attack)
- ▶ Summarizing: very hard to guarantee in practice



Intuitively

- Requires to secure the implementation against adversaries with infinite guessing power



Auxiliary input / unpredictability

- ▶ Given $L(k)$ it remains difficult to predict (one bit of) k
- ▶ Most natural type of assumptions
- ▶ Closely connects to actual SCAs
- ▶ But does not allow proving useful constructions (e.g. stream ciphers) in the standard model (up to now)
- ▶ Alternative: combine seed-preserving leakages with a RO based assumption [S+09], [YSPY10]



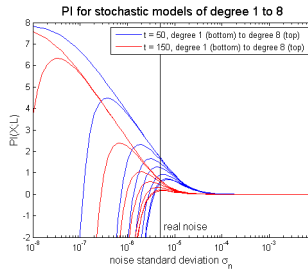
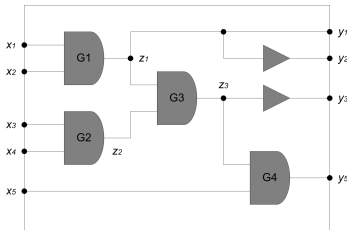
Issue 2: assumptions

- A1. Polynomial time vs. AC0 leakage functions
- A2. Adaptive vs. non adaptive leakage functions
- A3. Random oracle based assumptions
- A4. Limited information leakage
- A5. **Independent leakage**
- A6. Only computation leaks
- A7. Simulatable leakage
- A8. Secure precomputations



Independent leakages

- ▶ More precisely: \perp computations \Rightarrow \perp leakage
- ▶ Not correct at the gate level (to appear in EC2011)
- ▶ $L(x) = \sum \alpha_i x[i] + \sum \beta_{i,j} x[i]x[j] + \dots$



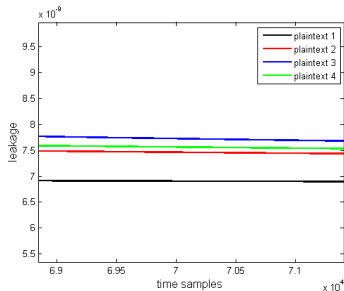
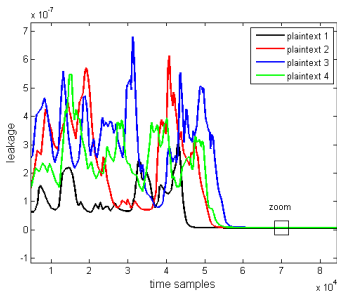
Issue 2: assumptions

- A1. Polynomial time vs. AC0 leakage functions
- A2. Adaptive vs. non adaptive leakage functions
- A3. Random oracle based assumptions
- A4. Limited information leakage
- A5. Independent leakage
- A6. **Only computation leaks**
- A7. Simulatable leakage
- A8. Secure precomputations



Only computation leaks

- ▶ Formally incorrect as devices scale below 65nm
- ▶ But static leakage still orders of magnitude smaller
- ▶ Summarizing: would be nice to include in the model



Issue 2: assumptions

- A1. Polynomial time vs. AC0 leakage functions
- A2. Adaptive vs. non adaptive leakage functions
- A3. Random oracle based assumptions
- A4. Limited information leakage
- A5. Independent leakage
- A6. Only computation leaks
- A7. **Simulatable leakage**
- A8. Secure precomputations



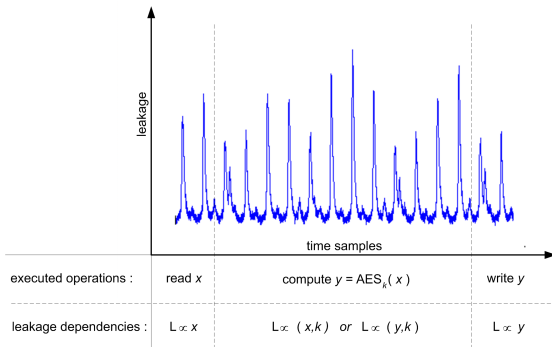
Simulatable leakage

- ▶ $\exists?$ *SIMU* such that $AES_k(x) \rightsquigarrow I$, $SIMU(x) \rightsquigarrow I'$ and (I, x) is hard to distinguish from (I', x) ?
- ▶ A proposal for block ciphers:
 1. Pick up $r' \xleftarrow{R} \{0, 1\}^k$;
 2. Perform $y_0'' = AES_{r'}(0) \rightsquigarrow I_a^{in} || I_a^{out}$;
 3. Compute $x_0' = AES_{r'}^{-1}(y_0'')$;
 4. Perform $y_0' = AES_{r'}(x_0') \rightsquigarrow I_b^{in} || I_b^{out}$;
 5. Return $I_0' = I_a^{in} || I_b^{out}$;
- ▶ (requires to concatenate traces)



Simulatable leakage

- ▶ Harder to achieve than seed-preserving L
- ▶ But easier to achieve than HILL pseudoentropy
- ▶ Is it useful?



Issue 2: assumptions

- A1. Polynomial time vs. AC0 leakage functions
- A2. Adaptive vs. non adaptive leakage functions
- A3. Random oracle based assumptions
- A4. Limited information leakage
- A5. Independent leakage
- A6. Only computation leaks
- A7. Simulatable leakage
- A8. Secure precomputations



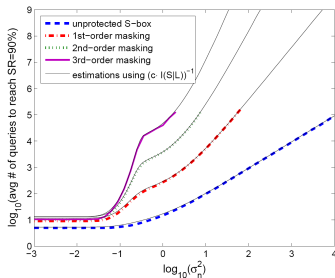
Secure precomputations

- ▶ Assume that the target device is sometimes operated in a secure environment, for refreshing
- ▶ e.g. one-time programs [GKR08]
- ▶ (or recent FOCS models [BKKV10,DHLW10])
- ▶ Can give rise to very simple intuitions



Secure precomputations

- ▶ e.g. Boolean masking: $x \rightarrow x \oplus m_1 \oplus m_2 \oplus \dots$
- ▶ Adversary can only recover x from the joint distribution: $(L(x \oplus M_1 \oplus M_2), L(M_1), L(M_2))$
- ▶ (so-called higher-order attacks)



Secure precomputations

- ▶ Now precompute $g_a(x, m) = x \oplus m \oplus a$
- ▶ (which requires storing a $2^{2n} \times n$ lookup table)
- ▶ The a share is only manipulated during precomputation
- ▶ Perfect security if “only computation leaks”
- ▶ Can be extended towards complete ciphers
- ▶ Not efficient but trivial proofs
- ▶ Strong assumption \Rightarrow strong security
- ▶ Are there better tradeoffs?

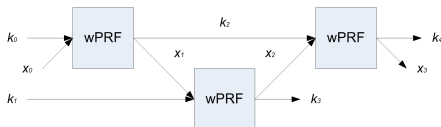


Issue 3: instantiation

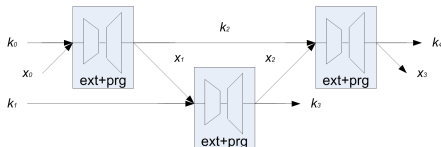


Issue 3: instantiation

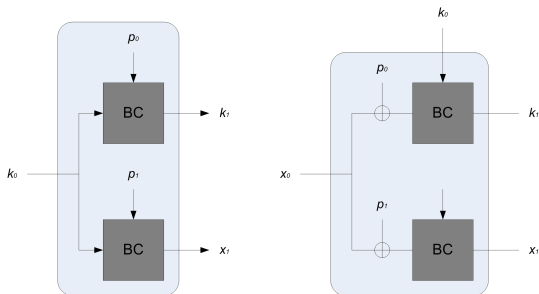
- ▶ wPRF-based stream cipher [P09]



- ▶ Extractor+PRG-based stream cipher [DP08]



AES-based w PRF and PRG



Summary

- ▶ 2 constructions
- ▶ [DP08] as significantly tighter reductions than [P09]
- ▶ Both are proven leakage resilient in the standard model, if the leakage per iteration is bounded to λ bits
- ▶ Open question: is an instance of [DP08] indeed more resistant against a standard DPA than an instance of [P09]? Or: how does the leakage of an extractor compare to the one of the wPRF and PRG?



Case study

1. [DP08] stream cipher components:

- ▶ Length tripling PRG instantiated with AES:

$$\text{PRG} : \{0, 1\}^n \mapsto \{0, 1\}^{3n} : x \mapsto \left(\text{AES}_x(c_1), \text{AES}_x(c_2), \text{AES}_x(c_3) \right)$$

- ▶ Extractor can be instantiated, e.g. with Vazirani 1987.
- ▶ (i.e., we extract 128 bits from two 196-bit sources)

2. 8-bit device, Hamming weight leakages, Gaussian noise

⇒ Which one is the weak point in the stream cipher?

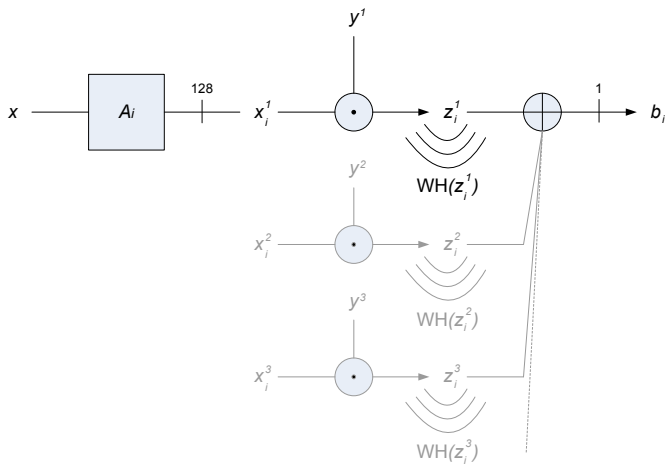


AES implementation

- ▶ Well known target for SCA
- ▶ PRG runs three AES computations
- ▶ Standard DPA: typically exploits one/two leaking points per AES computation (e.g. the key addition and/or S-box computation in the first round)



Leaking extractor implementation

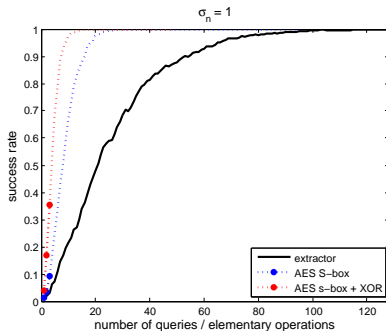
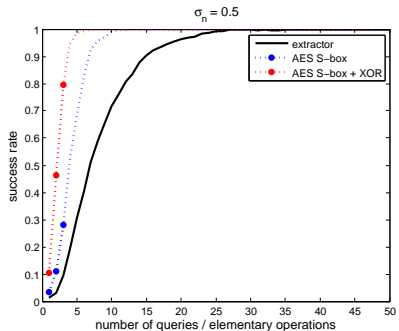


Main observations

- ▶ AES: 2 exploitable operations per key byte
- ▶ Extractor: 128 exploitable operations per secret byte
- ▶ AES: extensive use of bitwise XOR
- ▶ Extractor: extensive use of bitwise AND



Attack results



Summarizing

- ▶ λ -bit per AES iteration \ll λ -bit per Ext. iteration
- ▶ [DP08] has better security bounds than [P09]
- ▶ ... but it is easier to attacks with standard DPA
- ▶ The use of extractors can be paradoxical
- ▶ Similar to the general problem of trading security parameters (e.g. $(\epsilon^{1/3}, s)$ vs. $(\epsilon, s^{1/3})$ -secure PRGs)



Summarizing

- ▶ Results do not invalidate theoretical analyzes
- ▶ But show that their relevance to practice is limited
- ▶ Eventually, a useful construction needs to face the full complexity of side-channel attacks
 - ▶ i.e., not only assume λ -bit leakage but also find algorithms and implementations for which small leakages can be obtained: **instantiation matters**
- ▶ More research on extractors needed
- ▶ What about NIZK?

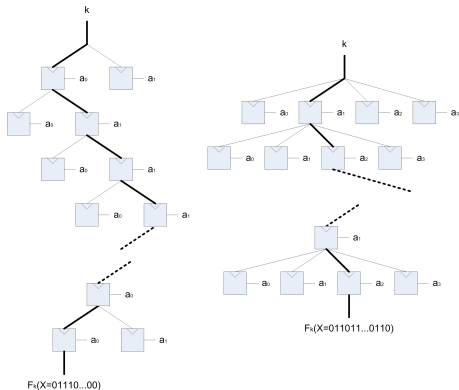


Issue 4: initialization



Issue 4: initialization

- ▶ Stream ciphers need to be securely initialized
- ▶ Only known solution is GGM tree [S+09,DP10]



Issue 5: tight bounds



Issue 5: tight bounds

- ▶ Bounds in leakage-resilience are not tight ($\epsilon^{1/4}$, $\epsilon^{1/12}$)
- ▶ Security guarantees vanish with the iterations
- ▶ Summarizing: present proofs validate constructions but do not allow determining security parameters
- ▶ (excepted with RO-based assumptions)



Conclusions

- ▶ Cryptographer's approach is too disconnected
- ▶ But implementation leakage and specificities are very difficult to capture with theoretical analysis
- ▶ Most problems remain open - no present solution is perfectly satisfying in theory and practice
- ▶ (we should not give up now)



Further research

- ▶ Always instantiate the proposed constructions
- ▶ If possible, implement (complexity matters!)
- ▶ Use empirically verifiable assumptions
- ▶ Find efficient initialization mechanisms
- ▶ Obtain tight bounds



THANKS

Questions?

