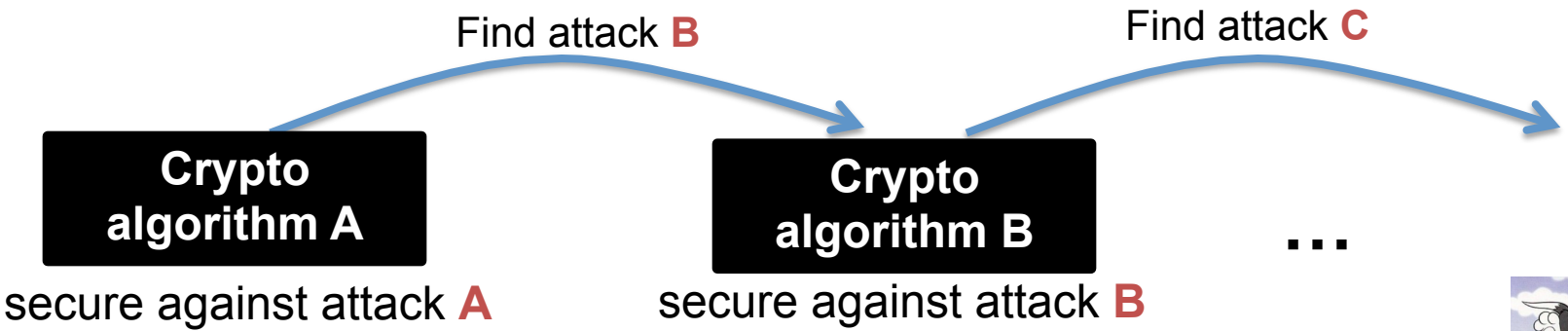
Leakage Resilient Masking Schemes

Sebastian Faust

Ruhr University Bochum

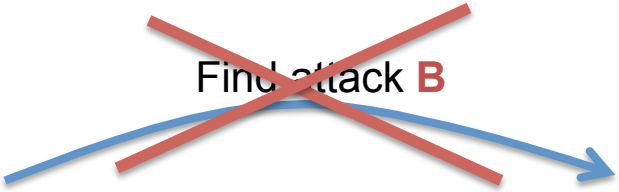
Modern cryptography

Until 1970s: Design crypto algorithm secure against **one** attack



Cat and mouse game between designer and attacker

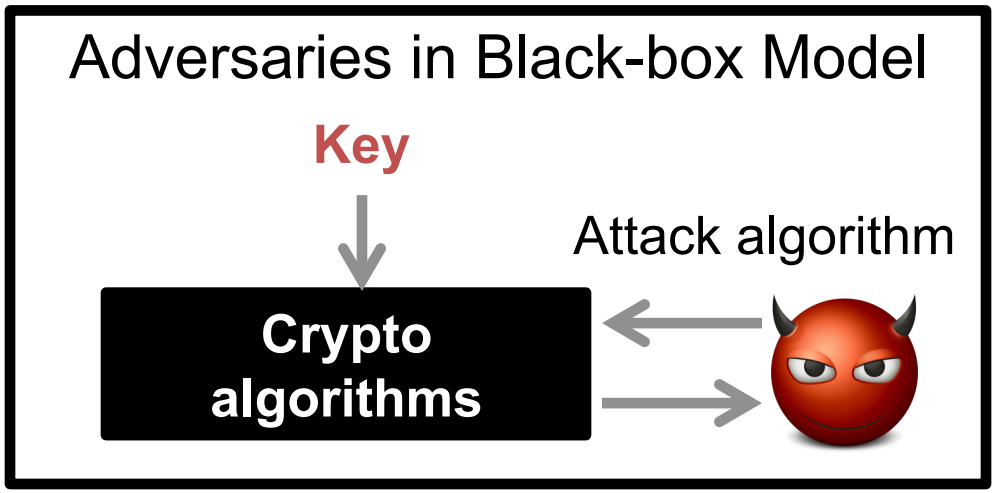
Modern crypto: Stop cat-and-mouse game



Main tool: security proofs

Secure against all attacks within a model

Black-box Model



Provable secure in black-box model ≈ unbreakable?

No! Crypto implementations get broken

Smart Cards broken by side-channel attacks
Much more efficient than traditional attacks on the algorithm

Adversaries in Black-box Model

Key

Attack algorithm

Crypto implementation

May reveal secret key **k**

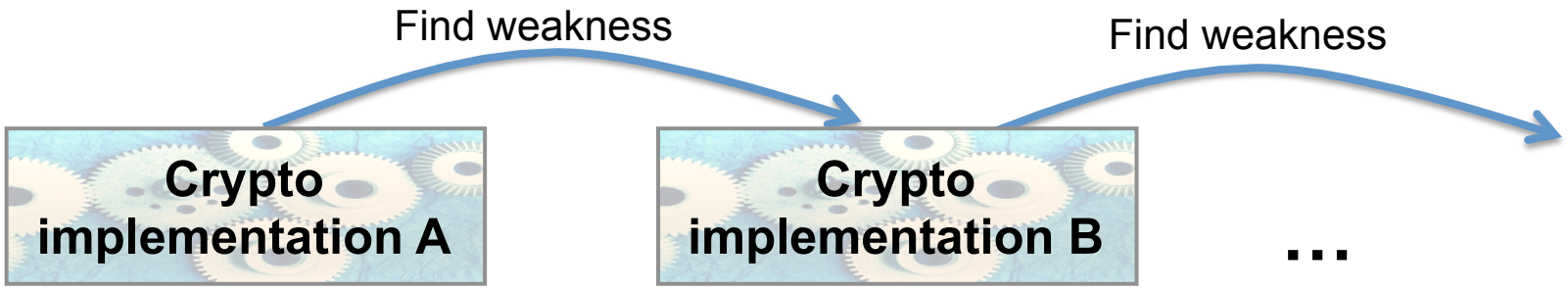
Power analysis exploits leakage

Side-channel attacks

Weaknesses outside of Black-box Model

The Problem

Cat and mouse game at implementation level



Goal of leakage resilient crypto

Extend the Black-box model

Design algorithms with better security at implementation-level

Rest of this talk:

Example for using proofs to design better countermeasures

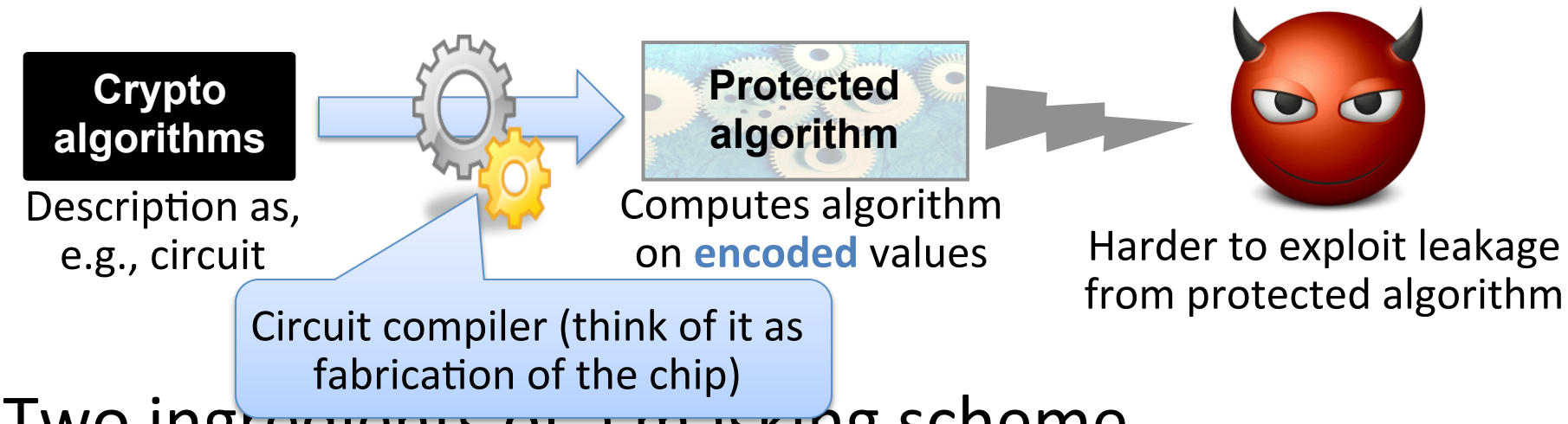
Masking countermeasure

Masking countermeasure

Masking

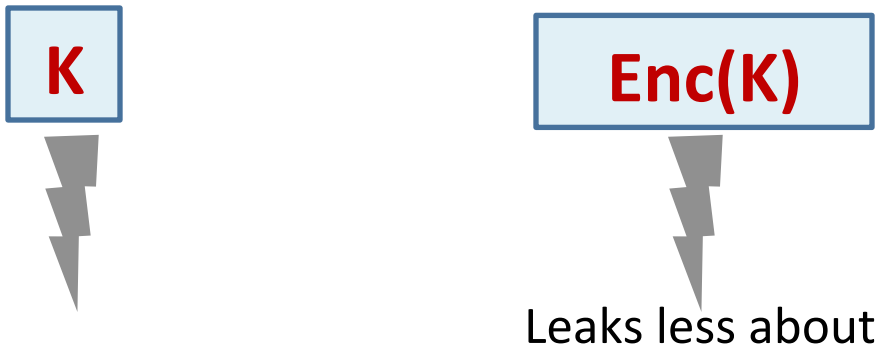
Important countermeasure

Masking: Protect all intermediate values of computation with randomized encoding **Enc**



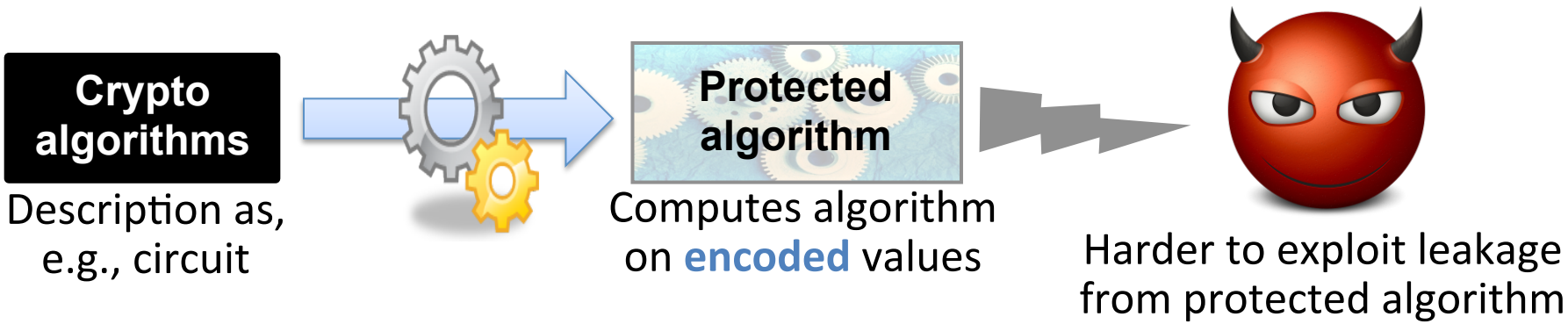
Two ingredients of a masking scheme

- Robust encoding function



Important countermeasure

Masking: Protect all intermediate values of computation with randomized encoding **Enc**



Two ingredients of a masking scheme

- Robust encoding function
- Operations computing with encoded values

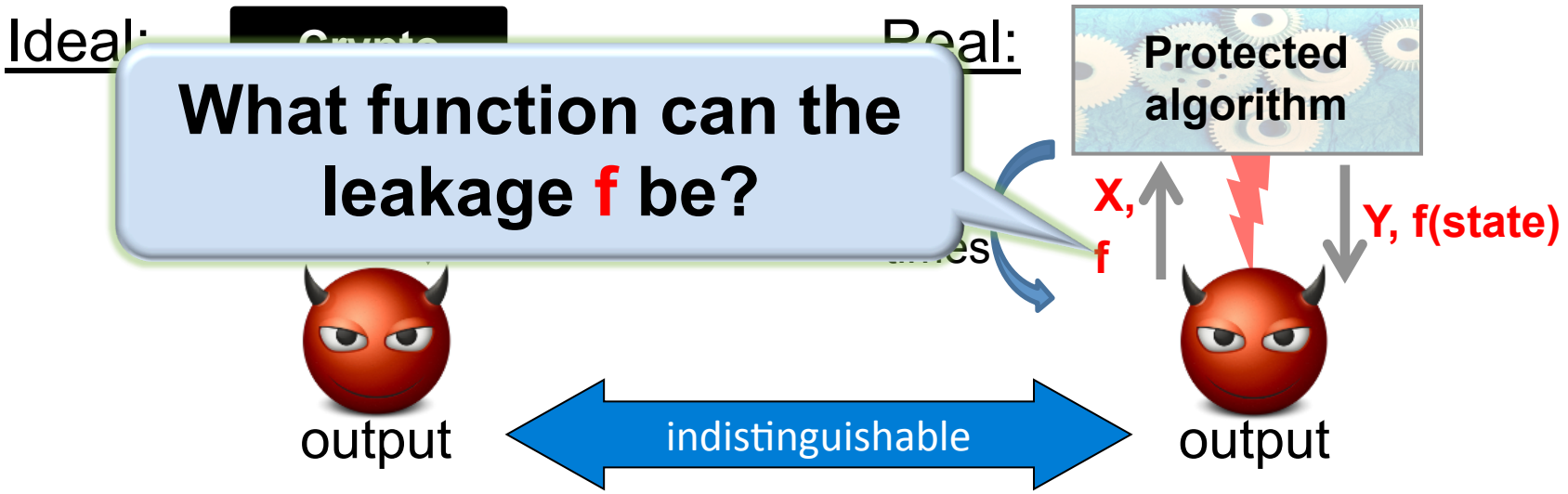
Masking countermeasure

Security definition

What does security mean?

Adversary learns no more than by black-box access

In other words: leakage is „useless“ to the adversary



Continuous leakage: many observations are possible

What does it mean?

For unbounded adversary: MI(Key ; f(.), ... f(.)) < negl

Even more: It is as secure as in the black-box world!

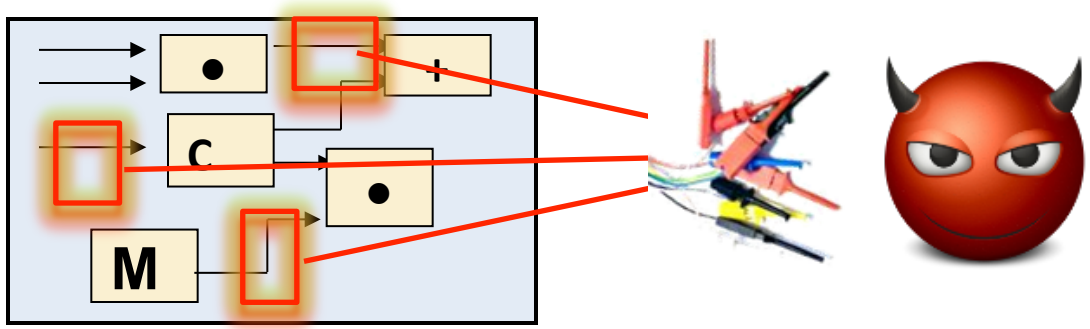
Masking countermeasure

Common: Bounded leakage models

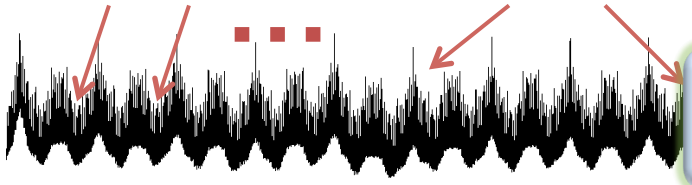
n-Probing adversary [ISW03]*

Adversary gets **t** intermediate values of computation

➔ **L** = { values on **t** adversarial chosen wires }



t-probing attack formalization of **t**-variate attacks



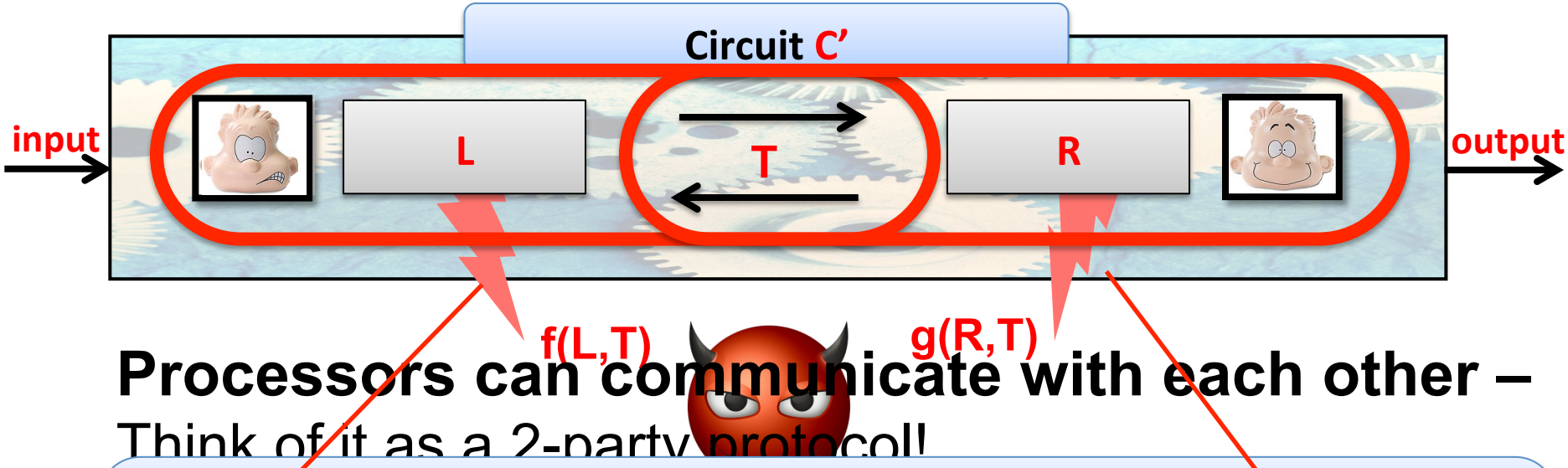
Learns nothing if **t < n**

Basic ingredient: Boolean masking

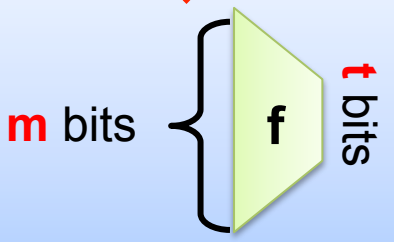
Drawback: leakage oblivious of many wires!

*Ishai, Sahai, Wagner (CRYPTO-03)

Bounded independent leakages [DF12]*



Processors can communicate with each other –
Think of it as a 2-party protocol!



Bounded leakage function:

- Arbitrary efficient functions
- Ony restriction: input shrinking, i.e., $t < m$

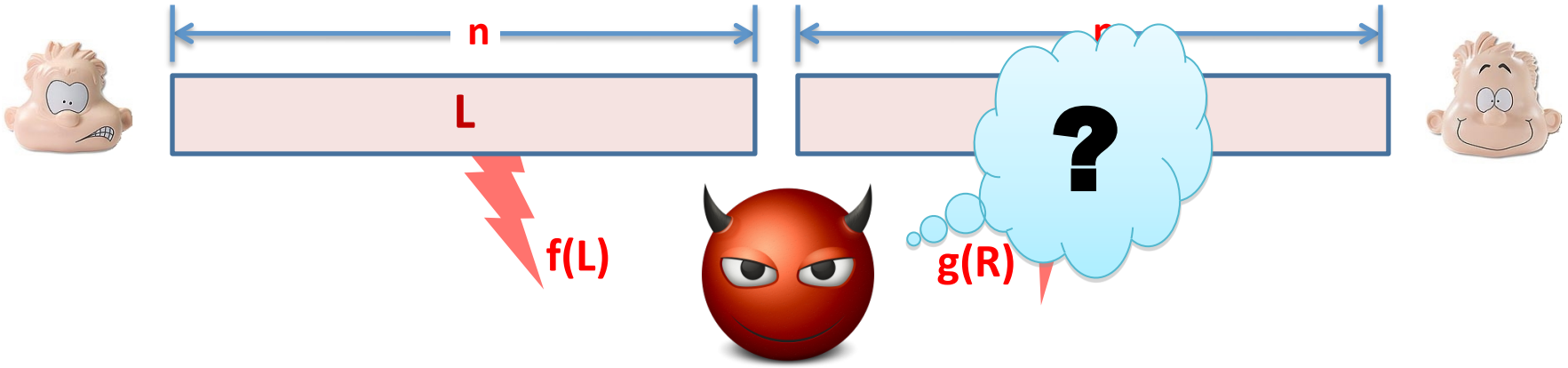
Realistic? Includes many functions, e.g. weighted sums

Additive masking? Insecure: learn parities of **L** & **R**

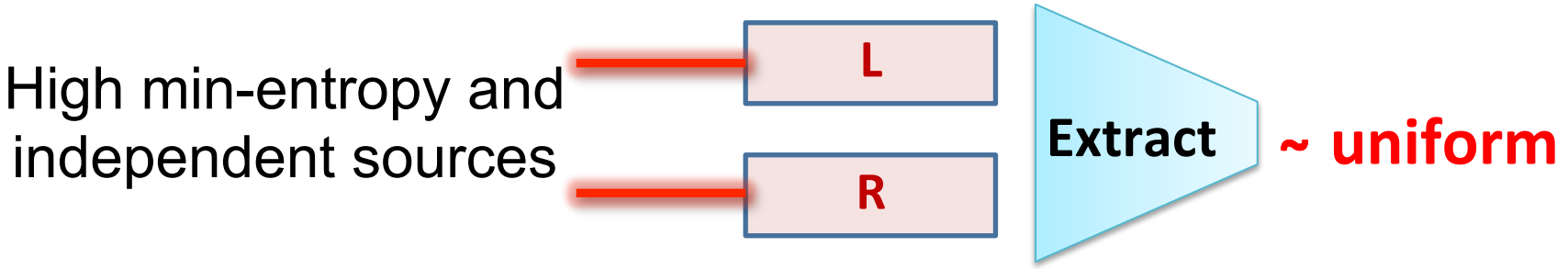
*Dziembowski-F(TCC-12) inspired from Dziembowski-Pietrzak (FOCS-08) for stream ciphers

Inner Product Masking

Sample **L,R** uniformly in $\{0,1\}^n$ s.t. $S = \langle L,R \rangle = \Sigma L_i * R_i$ and store parts separately on two processors



Thm [DDV10]*: if leakage is bounded in **total** to **t** bits then adversary learns nothing about **S**

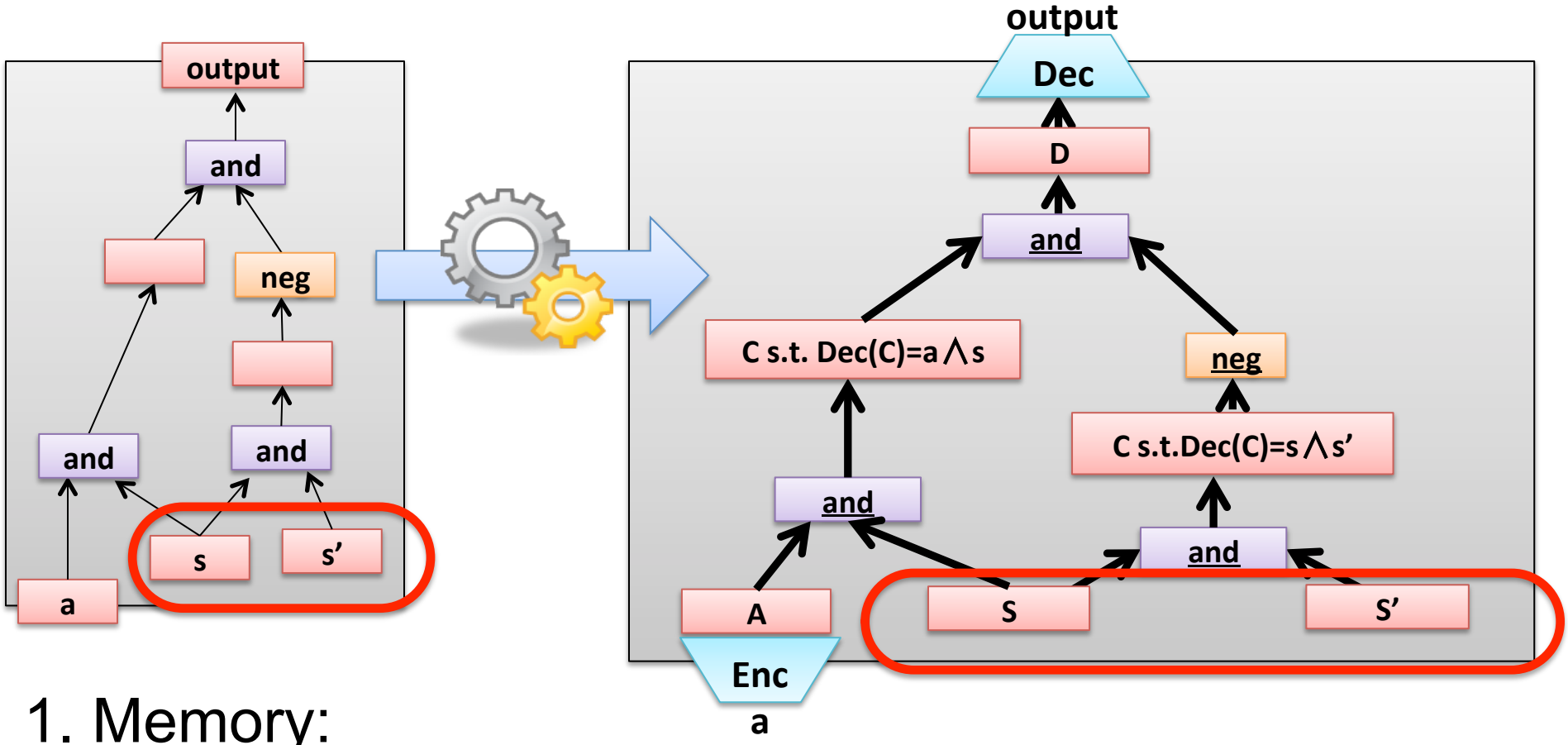


*Davi-Dziembowski-Venturi (SCN-12)

We know how to encode!

Next: How to compute!

High level idea of compiler

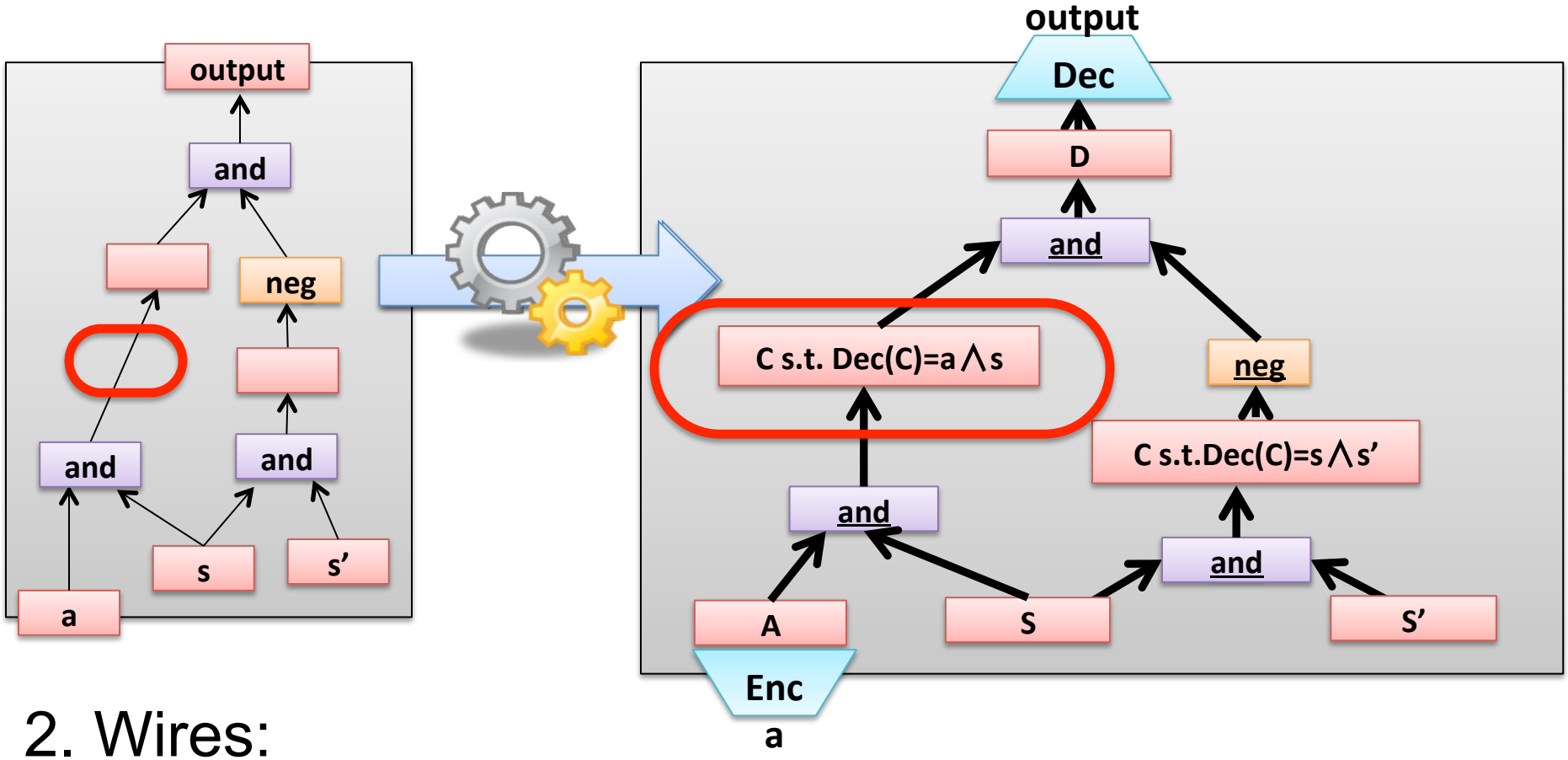


1. Memory:

A bit **s**

Encoded with coding scheme,
i.e., $S = (S_1 \ldots S_n)$ such that
$s = Dec(S_1, \ldots, S_n)$

High level idea of compiler



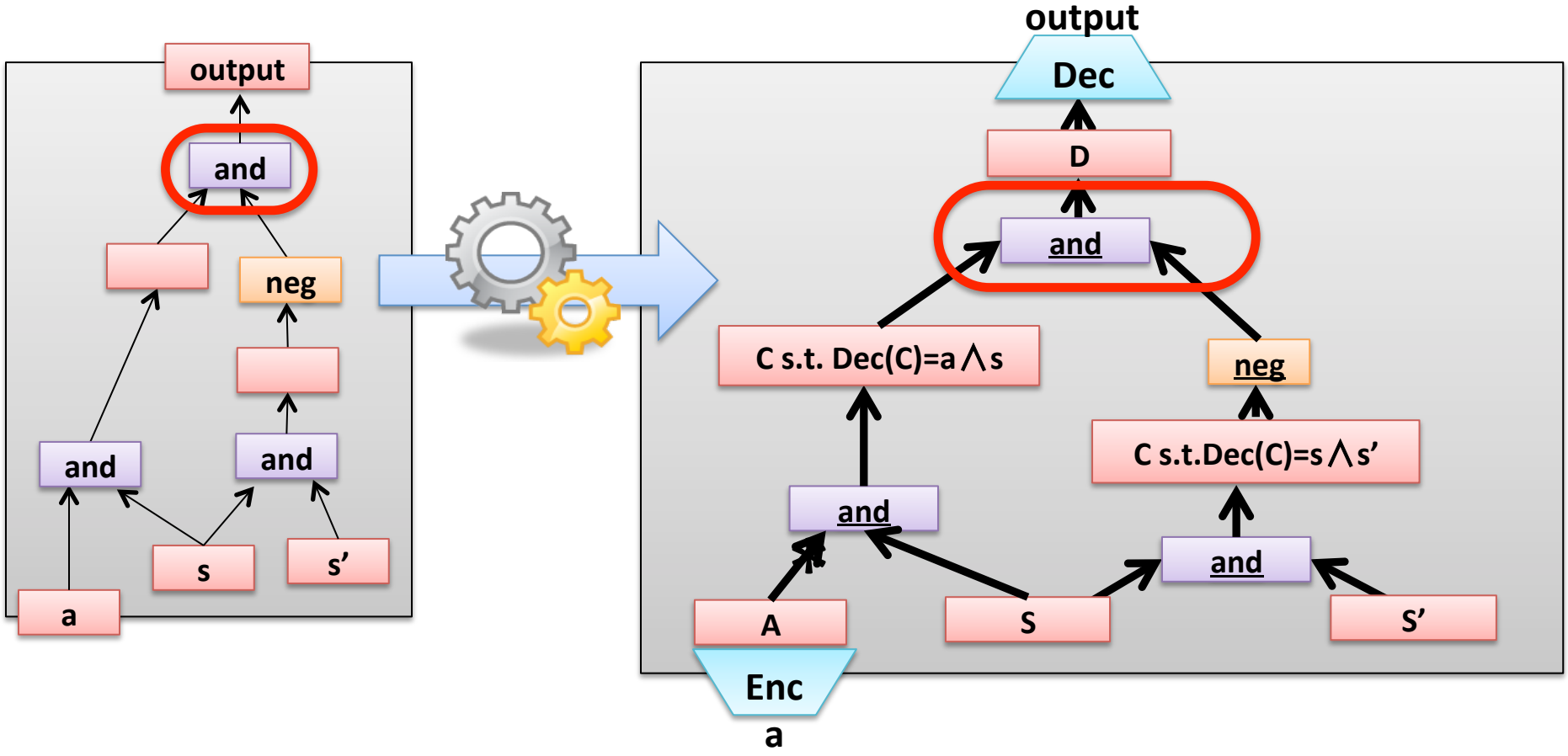
2. Wires:

Each wire
w = a ∧ b

Wire bundle carrying encoding
C such that **w = Dec(C)**

Main challenge: computing on encoded inputs!

High level idea of compiler



3. Gates:

Gadgets built from standard gates operating on encodings

Main challenge: algorithm to securely compute AND!

Compiler results

Theorem [ISW03]: A compiler that makes **any** C resilient to adversary that probes up to **t < n** wires in **C'**

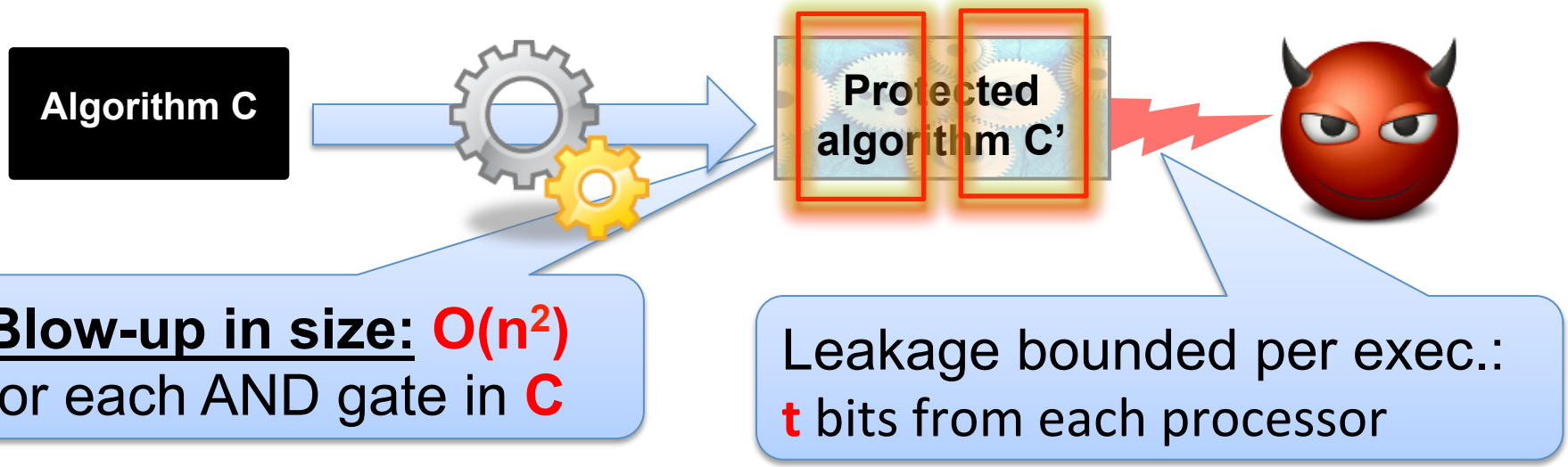
Blow-up in size: $O(n^2)$ for each AND gate in **C**

Leakage bounded per observation: **t < n** wires

Compiler results

Theorem [ISW03]: A compiler that makes **any** C resilient to adversary that probes up to **t < n** wires in **C'**

Theorem [DF12]: A compiler that makes **any** C resilient to adversary that learns bounded independent leakage from **C'**



Boolean vs. IP masking in practice?

Masking countermeasure

Implementations?

Boolean vs. IP2 (RFCV12, RFG15)*

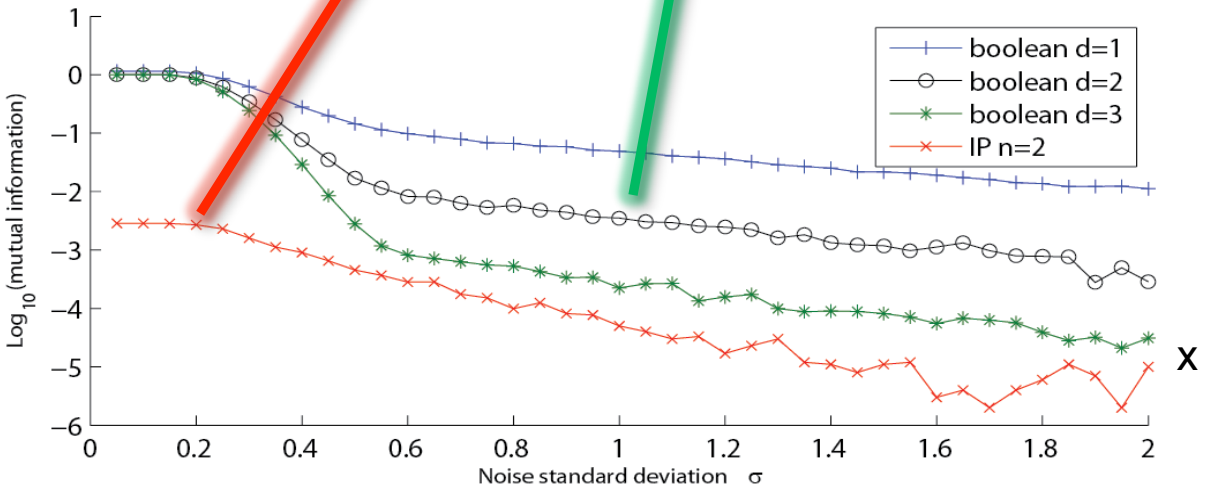
Analyzed for small s

Security outperforms

Black curve: Boolean masking with 2 random shares in GF(2^8)

Red curve: IP masking with 2 random shares in GF(2^8)

Mutual information between HW leakage and secret



Weaker dependency between leakage & secret for IP masking

Main reason: Higher algebraic complexity

Performance: Full AES implementation for **n=2**

- IP masking: 300k clock cycles
- Boolean masking: 100k clock cycles

*Balasch-F-Gierlichs-Verbauwhede (ASIACRYPT-12), Balasch-F-Gierlichs (EUROCRYPT-15)

Masking countermeasure

Noisy leakages

Bounded leakage?

Beautiful theory! Q: Does it match practice?

Models do **not** match with my engineering experience

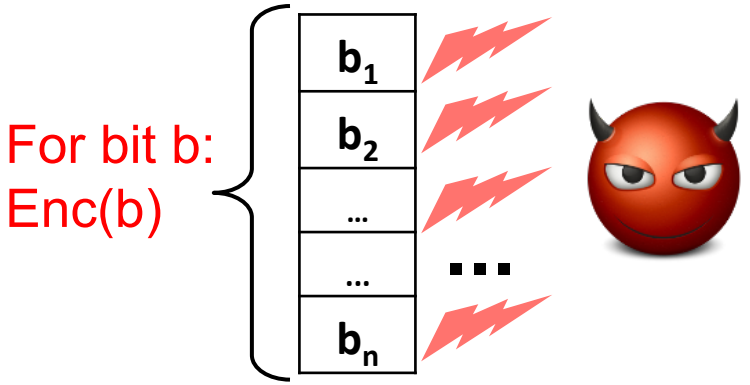
Leakages are not quantitatively bounded

Physical leakages are inherently noisy

Difficulty in many attacks: how to eliminate the noise?

Noisy leakages [CJRR99]

No quantitative bound on leakage, but leakage is noisy



Leakage is b_i + Gaussian noise

Chari et al. only consider security of a single masked bit

p-noisy functions N(.): More general noise distribution [PR13]

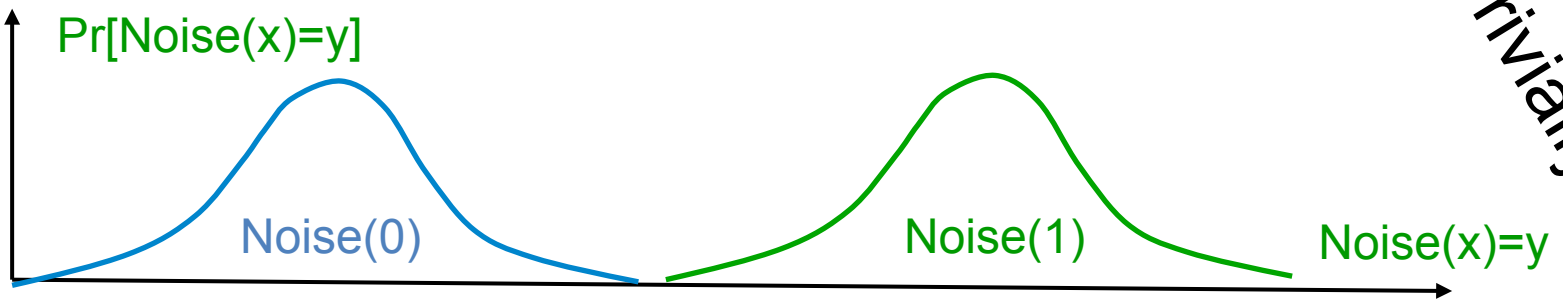
uniform X_0, X_1 → 1. $b \leftarrow \{0,1\}$ 2. $N(X_b)$

$N(X_0), X_0, X_1 \approx N(X_1), X_0, X_1$

N is **p**-noisy if statistical distance **< p**

*Chari-Jutla-Rao-Rohatgi (CRYPTO-99), Prouff-Rivain (EUROCRYPT-13)

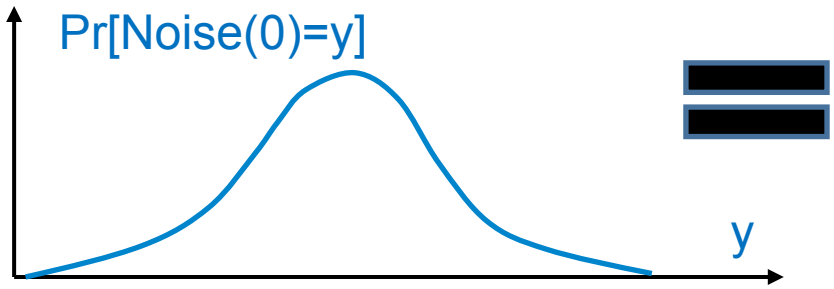
Some examples (F = GF(2))

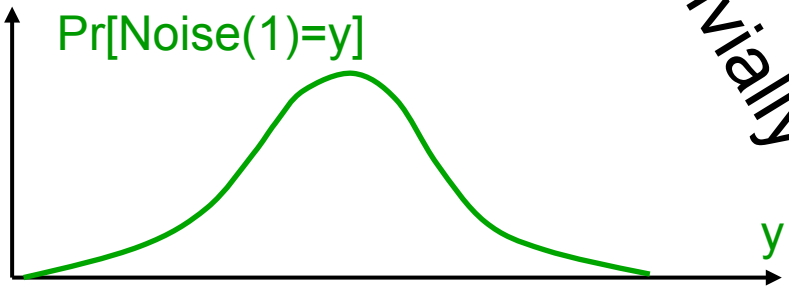


Trivially insecure

No noise **p ≈ 1**: very informative leakage

➔ Adv. learns **Noise(x)**: full knowledge about **x**





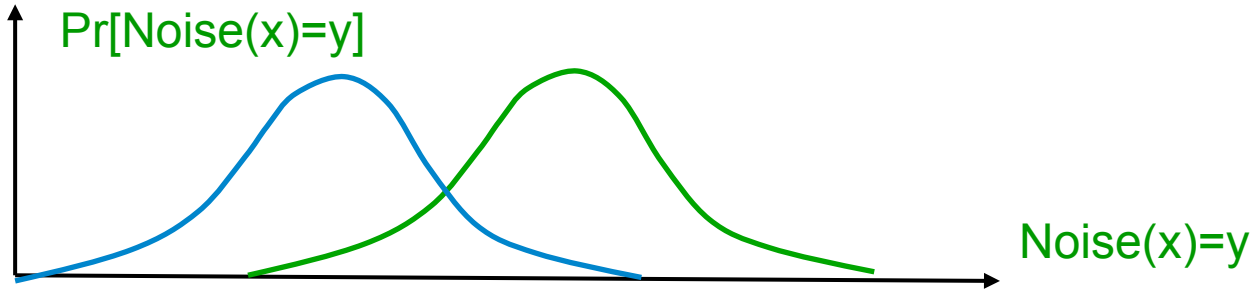
Trivially secure

High noise **p = 0**: non-informative leakage

➔ Adv. learns **Noise(x)**: no knowledge about **x**

Some examples (F = GF(2))

Interesting case: „some noise“



Some noise **p < 1**: information depends on **p**

➔ Adv. learns **Noise(x)**: some knowledge about **x**

Can we construct compilers for noisy leakages?

*(for interesting values of **p**)*

Compilers for noisy leakage

Compiler of ISW03

Adversary obtains **p**-noisy version of each wire: **$N(w_i)$** (for non-trivial **p**)

No quantitative bound on amount of leakage

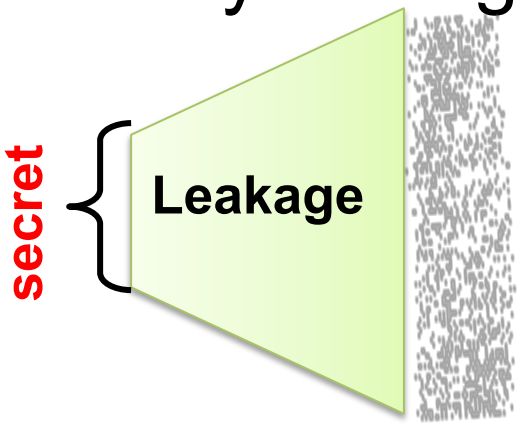
Two proof techniques:

1. **Direct proof** [PR13]**:** very technical proof, only random message security
2. **Leakage reductions** [DDF14, DFS15]*: simpler proofs, full simulation-based security

*Duc-Dziembowski-F (EUROCRYPT-14), Dziembowski-F-Skorski (EUROCRYPT-15)

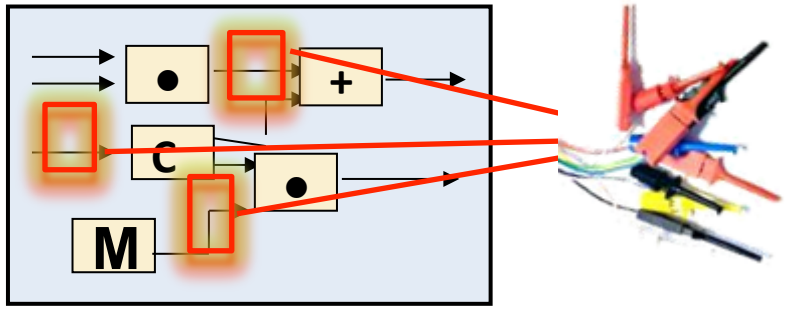
Main observation [DDF14]

Noisy leakage



Preferred model in practice

Probing leakage



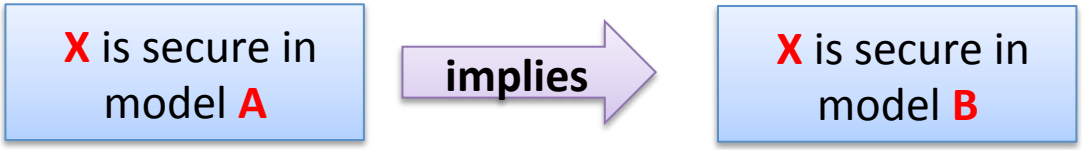
Simpler model than bounded leakage

Main technique: leakage reduction

What is a leakage reduction?

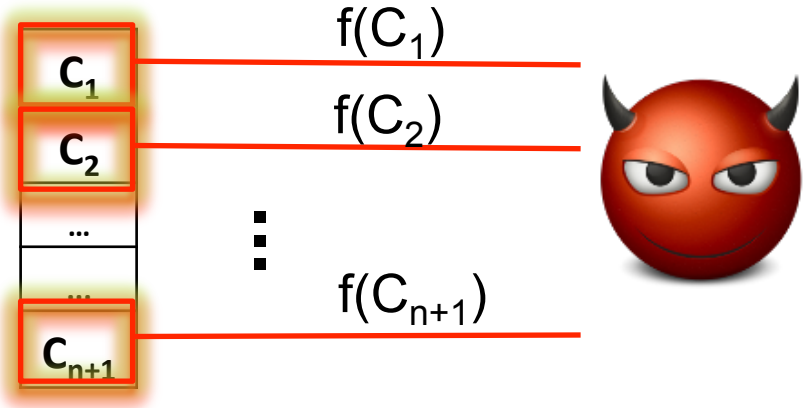
We want to show that model **A** is „stronger“ than a model **B**

More precisely: **for all** crypto schemes **X**



DDF14 shows:

(for certain parameters **p** and **q**)

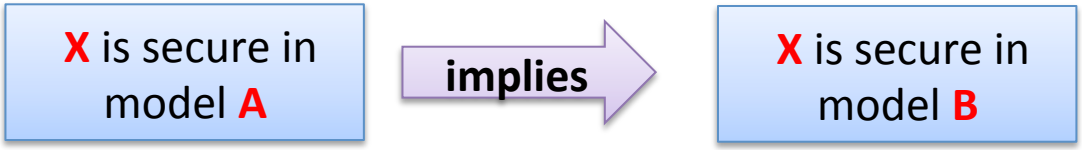


$f(C_i) = C_i$ with prob. **$q < 1$**;
otherwise **$f(C_i) =$ „?“**

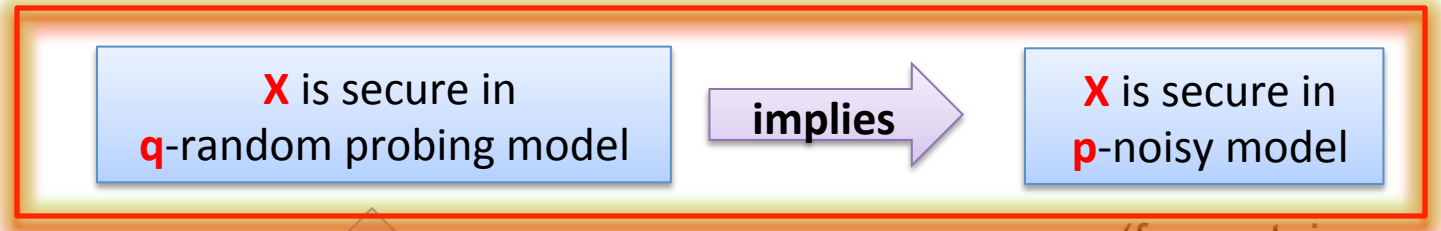
What is a leakage reduction?

We want to show that model **A** is „stronger“ than a model **B**

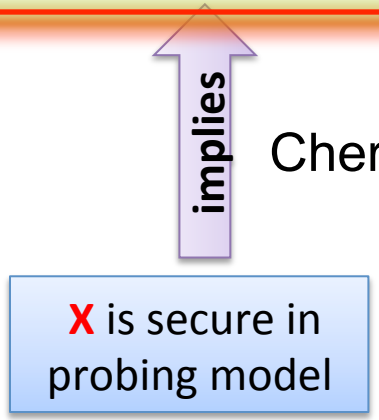
More precisely: **for all** crypto schemes **X**



DDF14 shows:



(for certain parameters **p** and **q**)

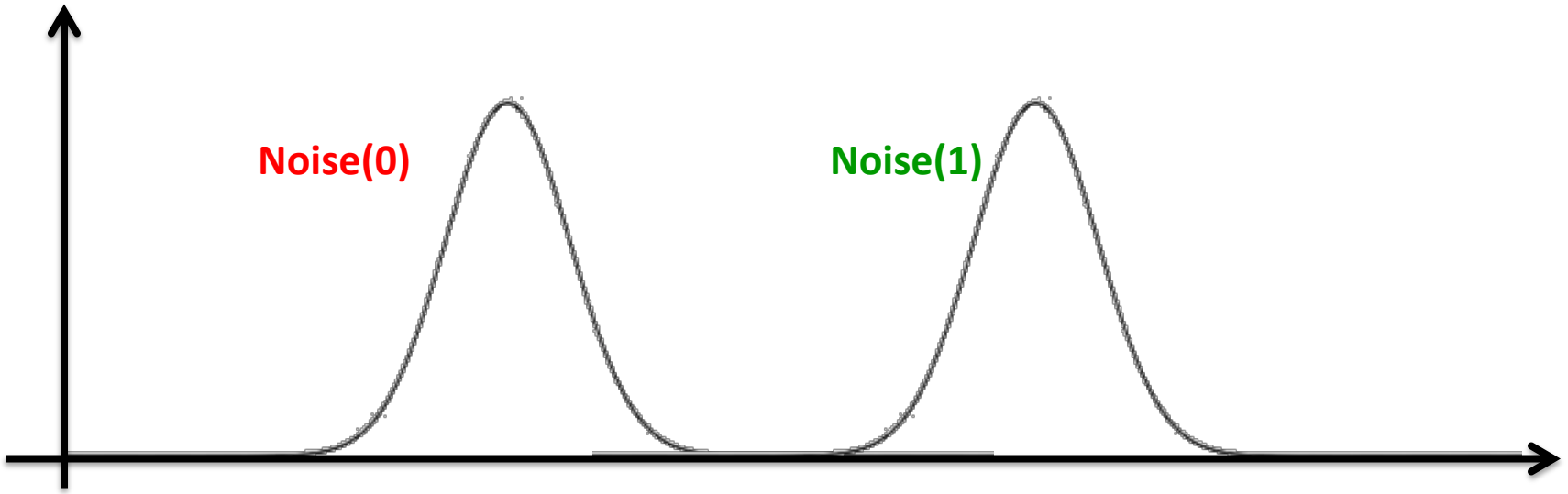


Proof outline

q-Random probing ➔ **p**-noisy leakage

For any **x** and **Noise(.)** there exists **Noise‘(.)** such that **Noise(x) = Noise‘(f(x))**

First extreme case: “no noise” (**p≈1**)



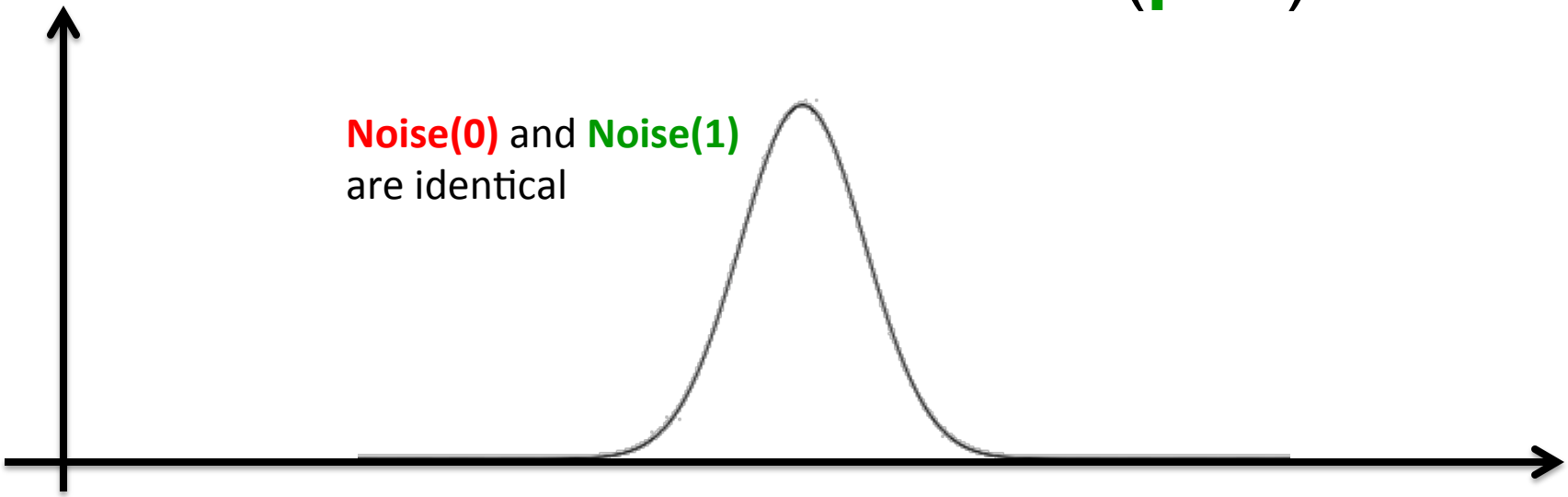
No way to “simulate” this noise except with random probing where **q=1** (i.e., reveals everything).

Proof outline

q-Random probing ➔ **p**-noisy leakage

For any **x** and **Noise(.)** there exists **Noise‘(.)** such that **Noise(x) = Noise‘(f(x))**

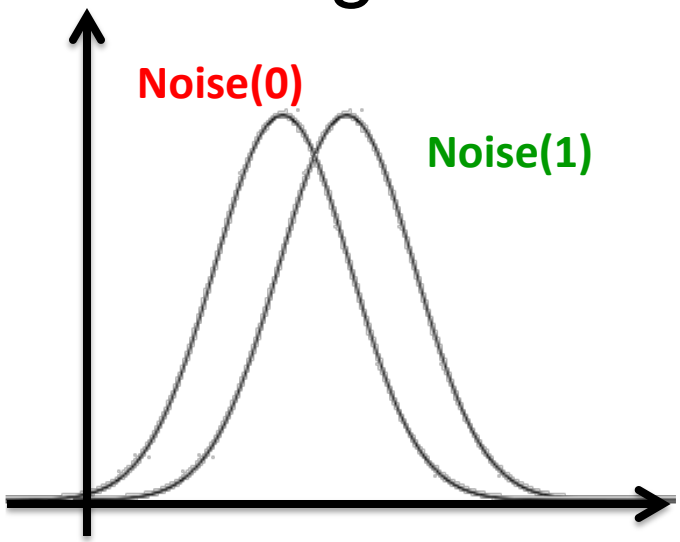
Second extreme case: “full noise” (**p=0**)

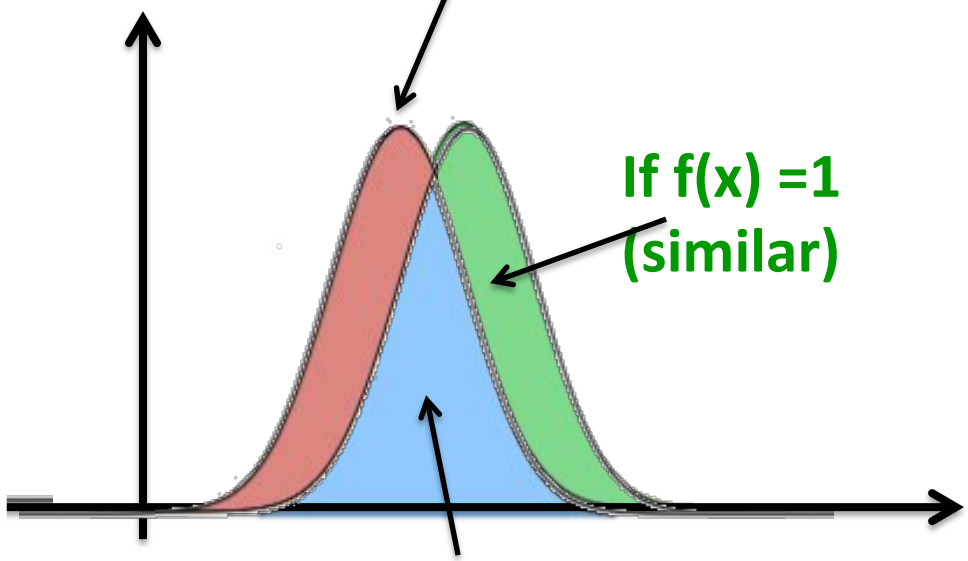


Set **Noise‘** = **Noise**: Simulation is possible without even probing: **q = 0** (i.e., reveals nothing)

Proof outline

Interesting case: “some noise”





We show that simulation of **Noise(.)** with random probing is good when probability **q** is exactly **Δ(Noise(0),Noise(1))**

(proof is essentially a simple „averaging argument“)

Noise parameters?

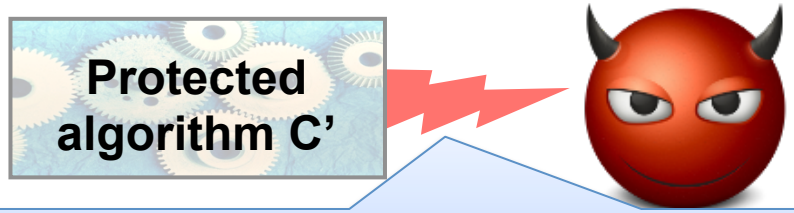
It depends on the tightness of the reduction!

DDF14: q-random probing secure → implies → q/|F|-noisy leakage secure

Better reduction?

DFS15: q-average random probing secure → implies → q-noisy leakage secure

Allows to derive asymptotically optimal bound for ISW



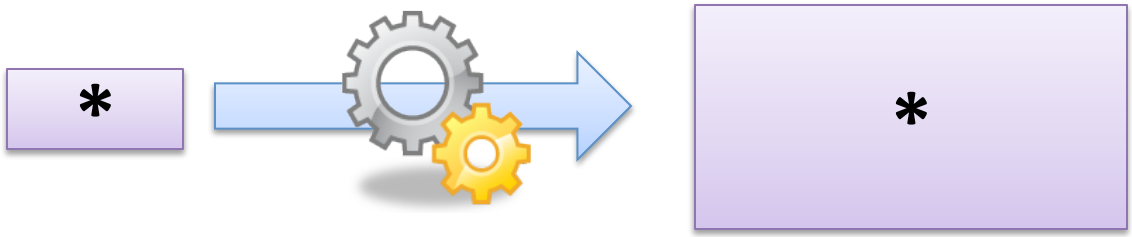
Adversary obtains $O(d^{-1})$-noisy leakage

Masking countermeasure

Better efficiency?

Better efficiency?

Main bottleneck: masked multiplication



Complexity & randomness blow-up $\mathbf{O(n^2)}$

1. Approach: Use techniques from the MPC

Packed-secret sharing gives us quasi-linear complexity

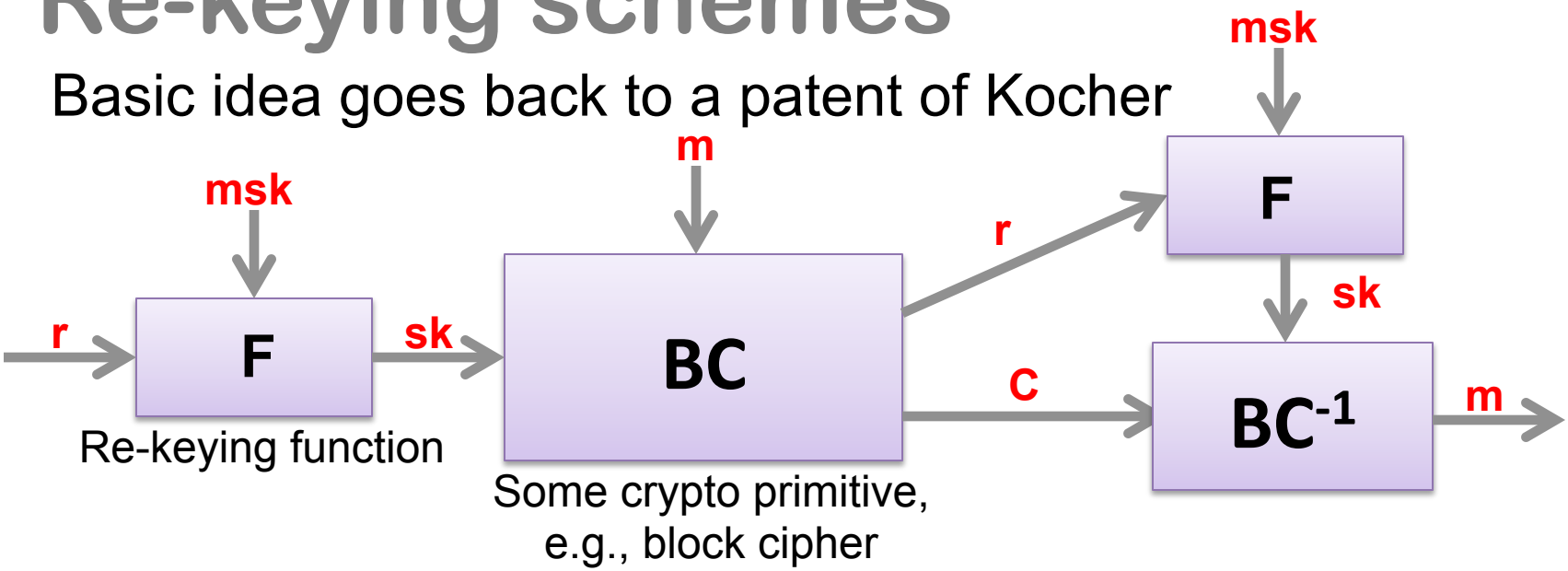
2. Approach: Secure specialized crypto schemes

LowMC cipher etc. (minimizes number of multiplications)

Re-keying schemes

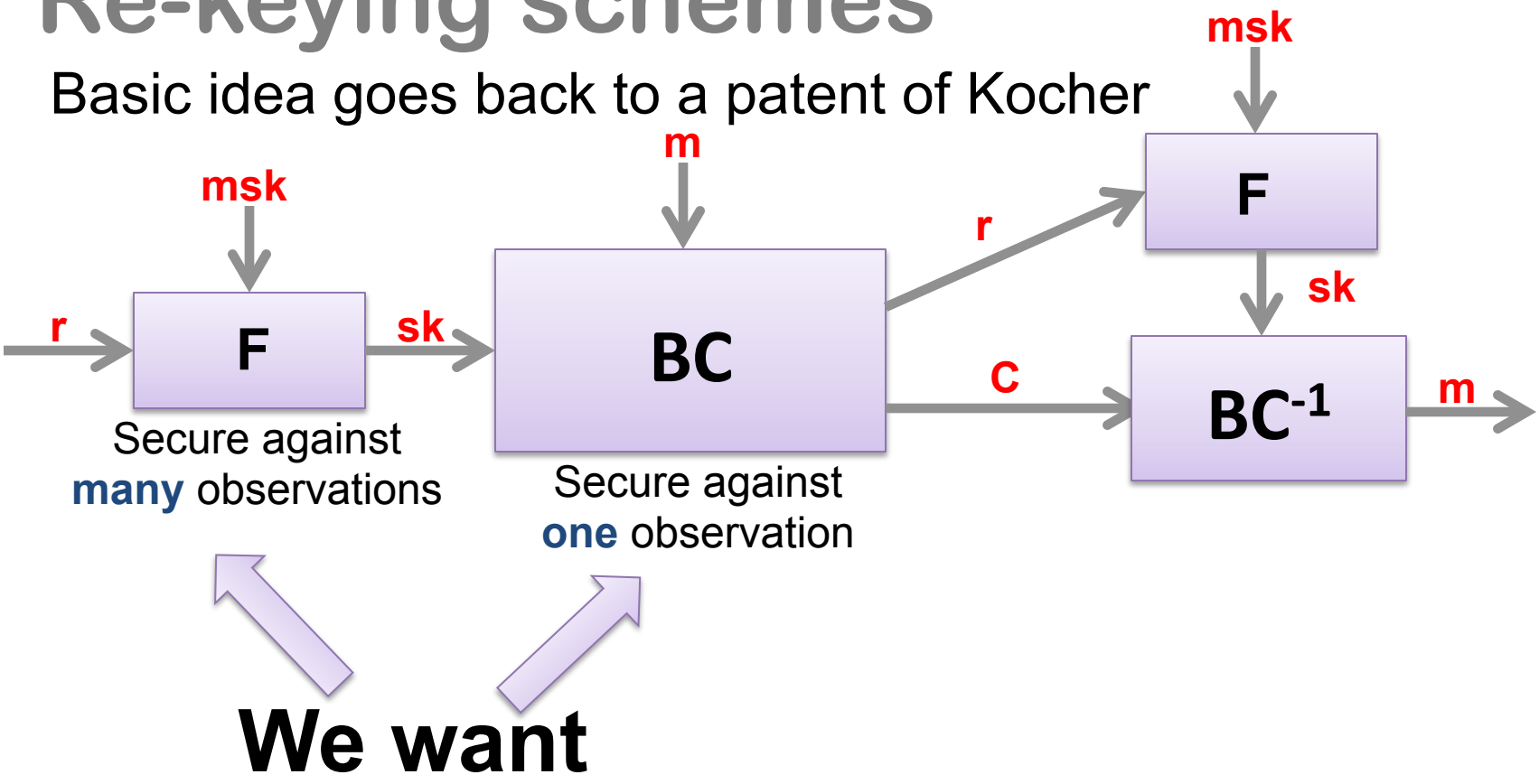
Re-keying schemes

Basic idea goes back to a patent of Kocher



Re-keying schemes

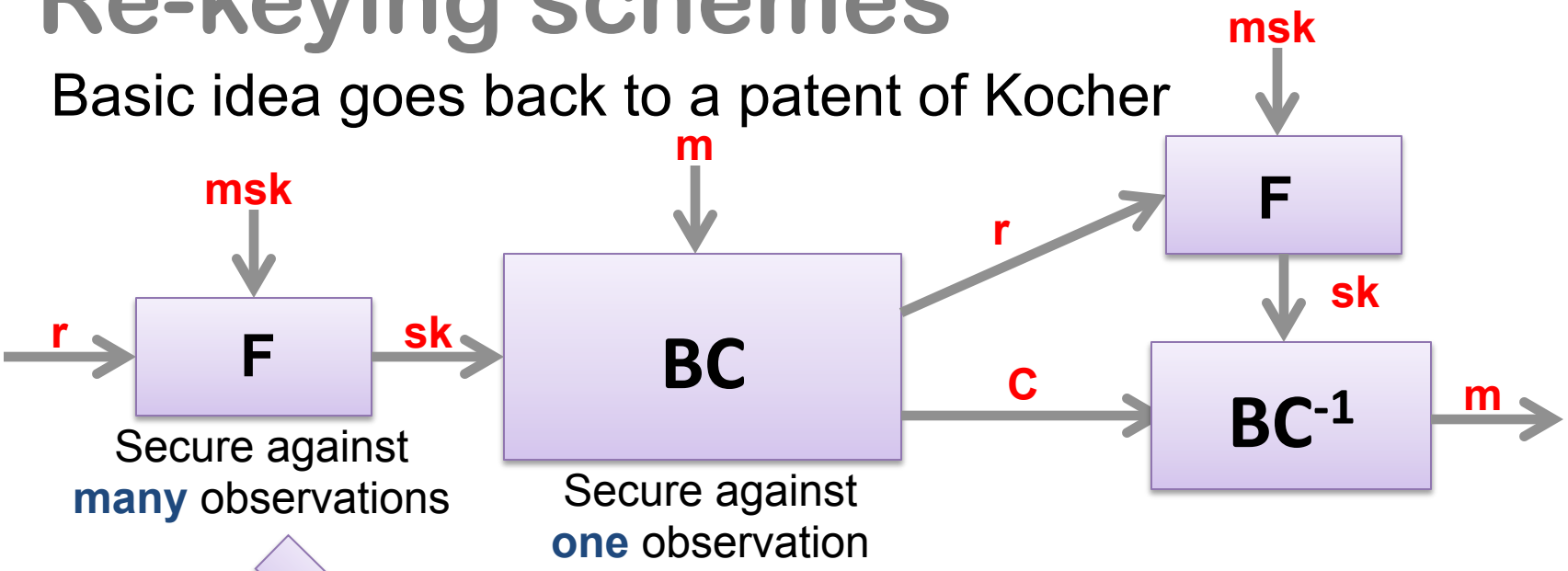
Basic idea goes back to a patent of Kocher



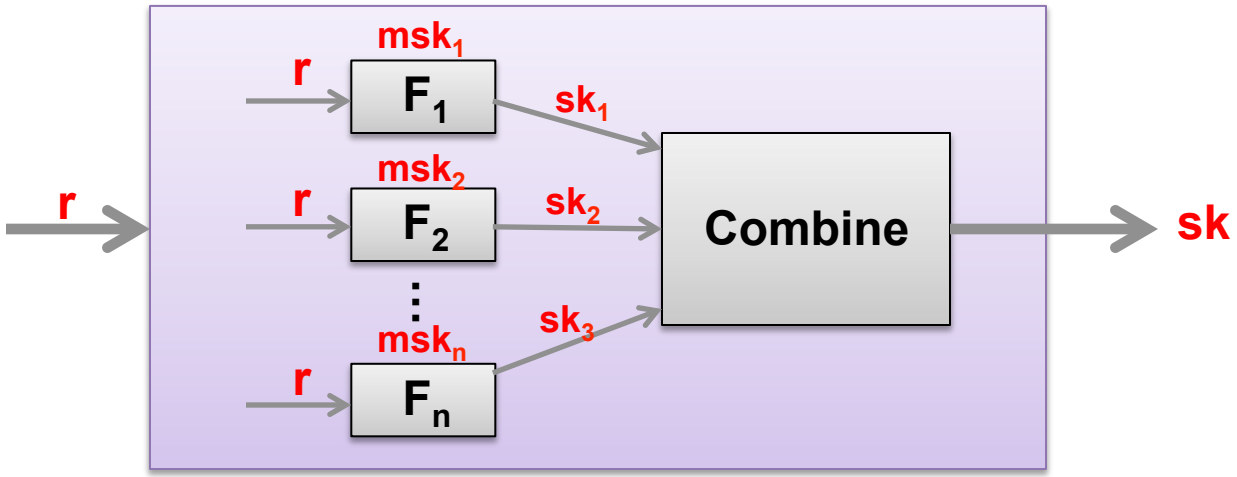
We want

Re-keying schemes

Basic idea goes back to a patent of Kocher



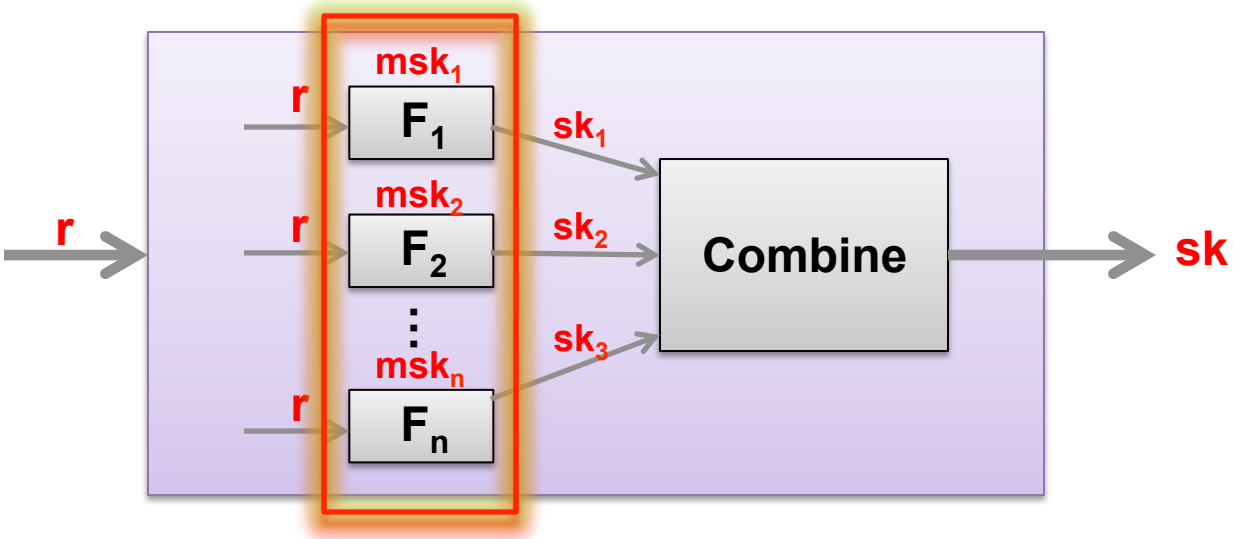
Idea: Use masking to protect **F** [MSGR10, FGM10]*



*Medwed, Standaert, Großschädl, Regazzoni (Africacrypt-10) & Fischer, Gammel, Mangard (US patent)

How to instantiate F

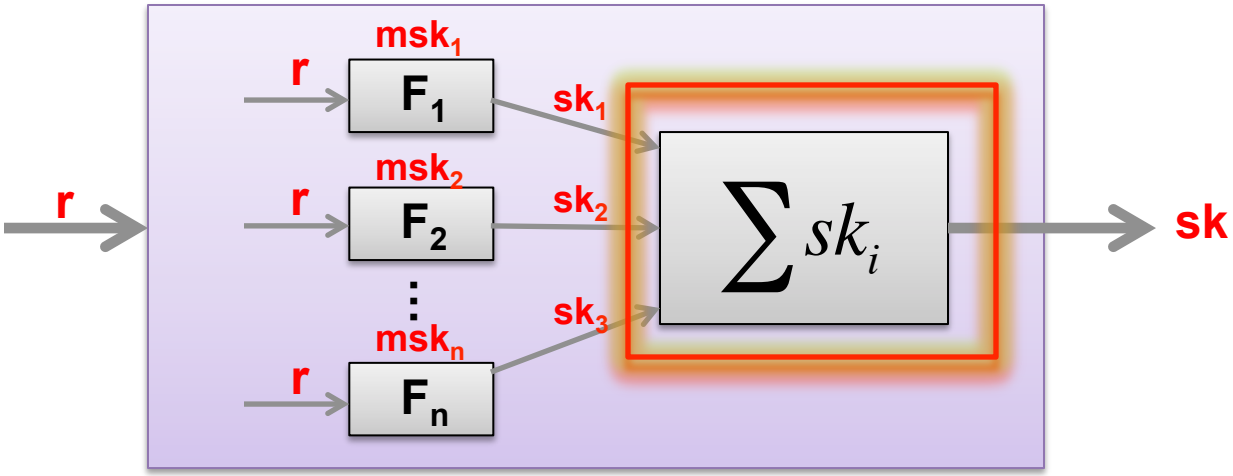
The approach of [MSGR10, FGM10]:



Ring multiplication: $sk_i = r * msk_i$

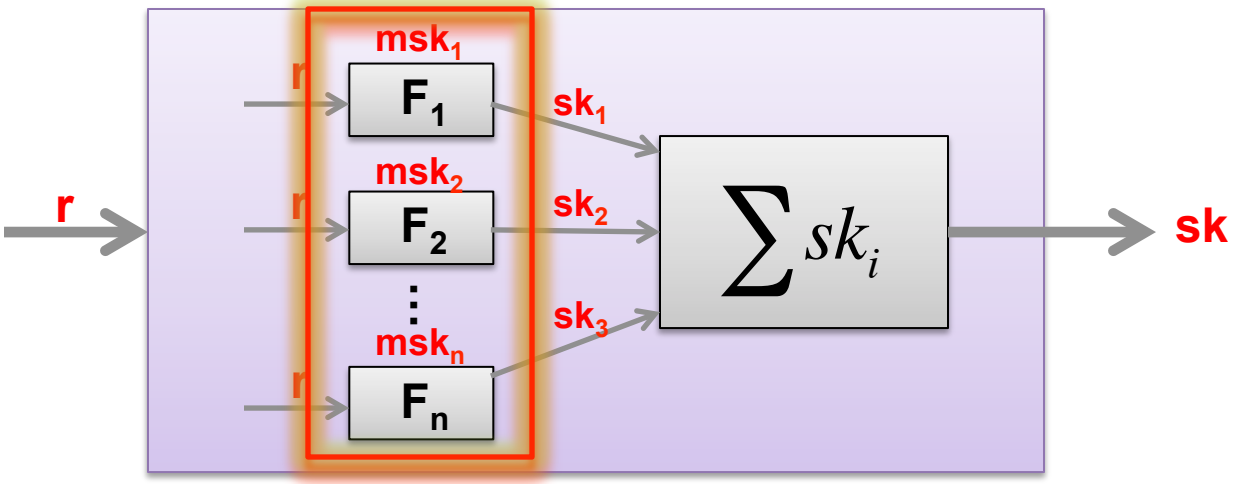
How to instantiate F

The approach of [MSGR10, FGM10]:



How to instantiate F

The approach of [MSGR10, FGM10]:

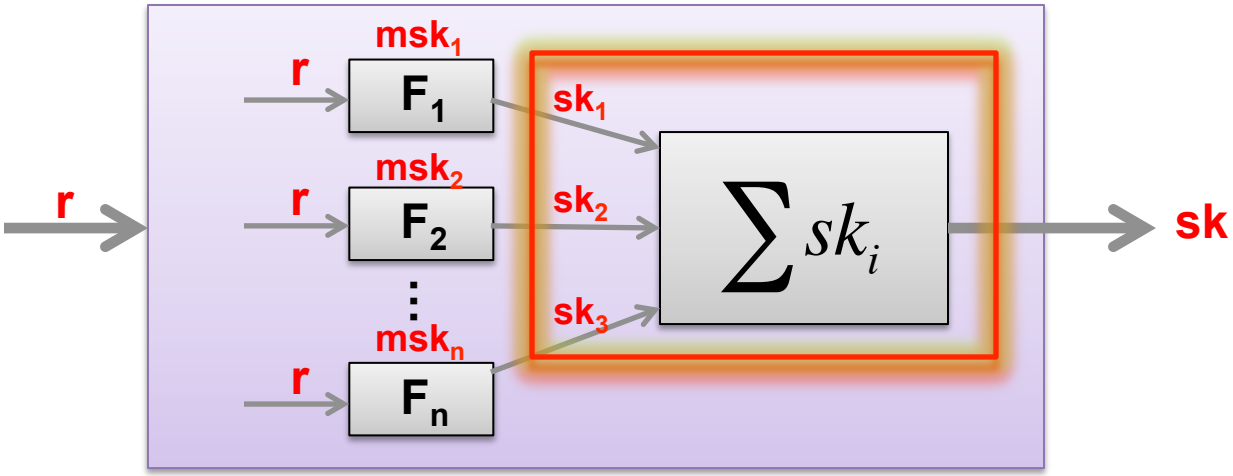


Security?

1. F_i is masked ➔ hard to attack (proof in probing model trivial)

How to instantiate F

The approach of [MSGR10, FGM10]:



Security?

1. F_i is masked ➔ hard to attack (proof in probing model trivial)

2. The values **sk_i**? Harder…

a) MSGR10: if individual bits of **sk_i** and **sk** learned can be broken

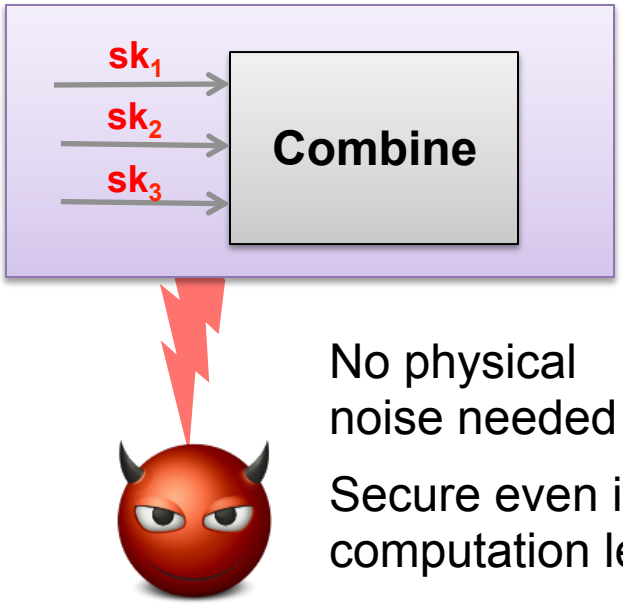
b) BCFGKP15*: if adversary obtains low-noisy version of **sk**

How to protect the **sk_i** and **sk** values?

*Belaid, Coron, Fouque, Gerard, Kammerer, Prouff (CHES-15)

The approach of [DFHMS15]*

No noise

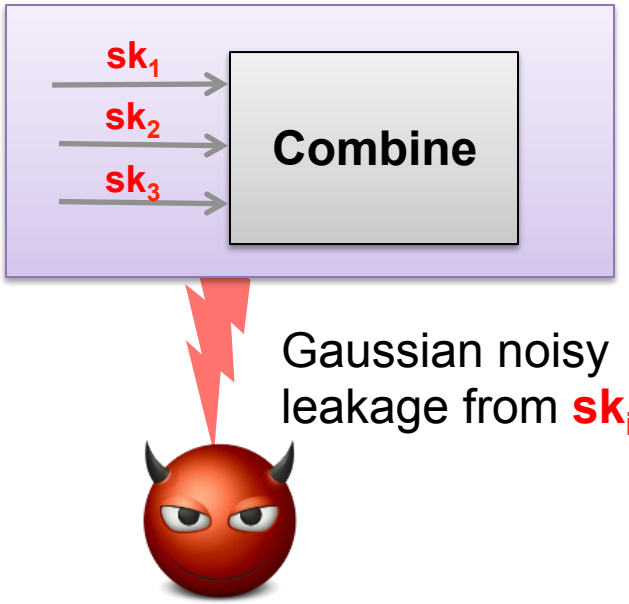


No physical noise needed

Secure even if entire computation leaks

Tool: Key homomorphic wPRF based on learning with rounding

Noisy case



Gaussian noisy leakage from sk_i

Tool: The LPL assumption
(reduction to Learning with parity)

These schemes are **practical** & and security **relies on proofs**!

*Dziembowski, Faust, Herold, Masny, Standaert 2015

Masking countermeasure

Conclusion

Conclusion

Why proofs for masking?

Systematic analysis to avoid flaws

➔ Proofs in n-probing model to check for n-th order flaws

New ideas and schemes

➔ IP masking an alternative for additive masking?

Formal requirements on hardware

➔ How much noise do I need to use masking?

Schemes that are easier to mask?

➔ Outperform general purpose masking schemes for larger orders

Conclusion

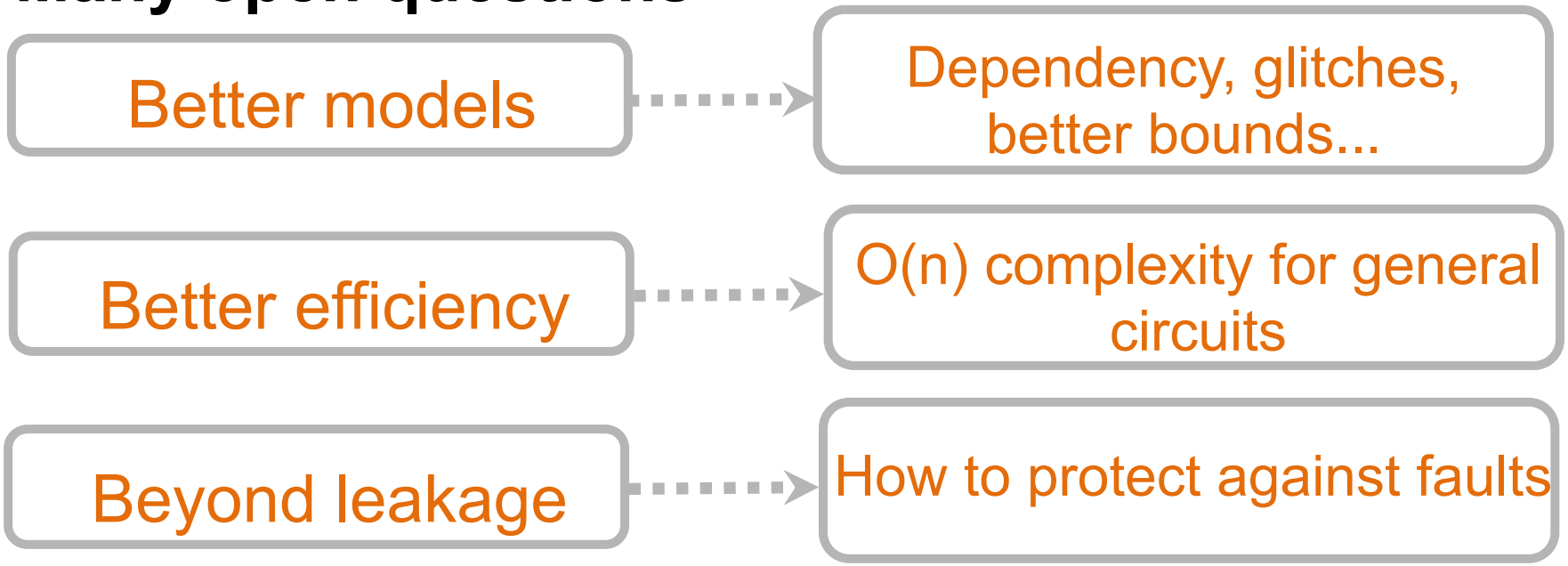
Many more works...

Threshold implementations [NRR06],...

Automated security proofs [BBDFGS15],...

New theory models with applications beyond SCA [MV13],..

Many open questions



*Nikova, Rechberger, Rijmen (ICIS06), Barthe, Belaid, Dupressoir, Fouque, Gregoire, Strub (EUROCRYPT15)

Thank you!