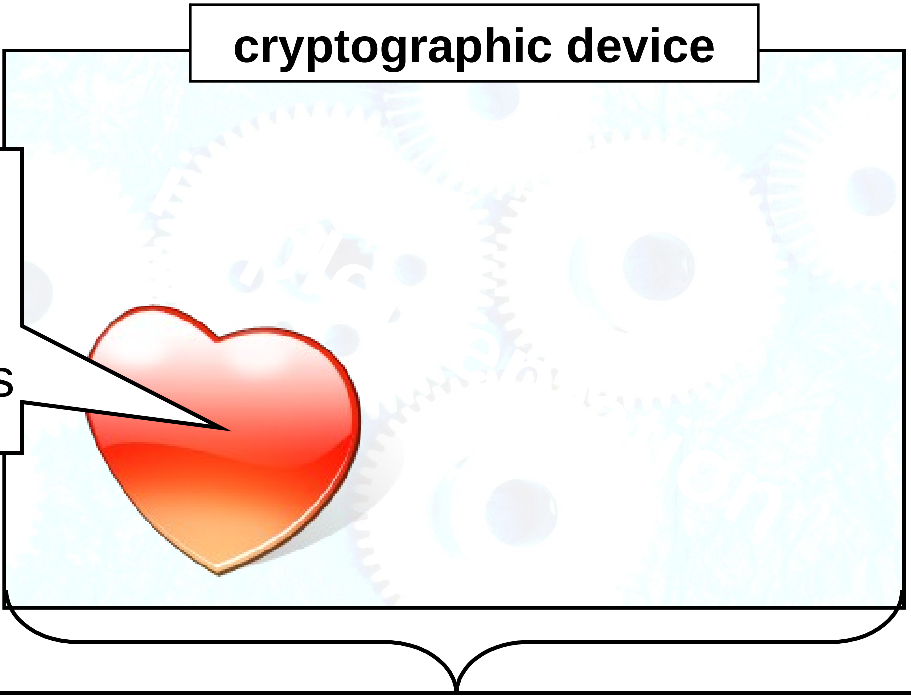
Leakage Resilient Circuits

(or: Leakage models for masking)

Sebastian Faust

EPFL, Switzerland

Cryptodevices



cryptographic device

very secure

– well-defined mathematical object

– often proof-driven security analysis

much less secure!

– many ways of implementing: details matter!

– new attacks possible on crypto implementations

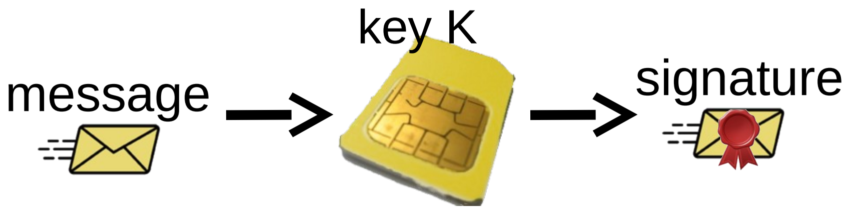
Goal of Leakage resilient crypto:

Proof-driven security analysis for implementations

Provable Security

1. Define model & security notion

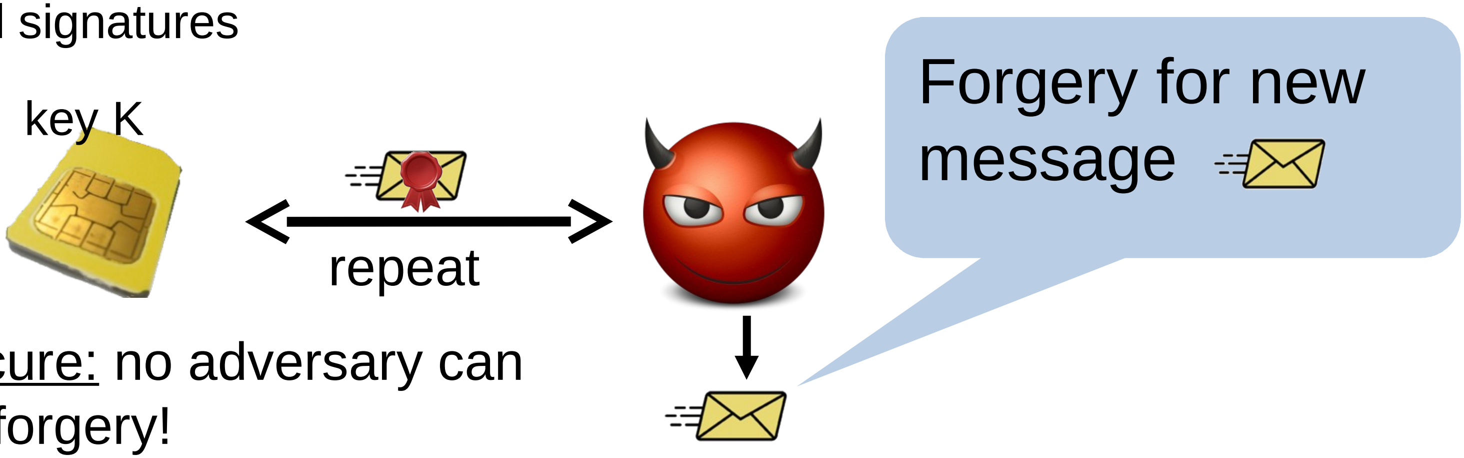
Example: Digital signatures



Provable Security

1. Define model & security notion

Example: Digital signatures



Forgery for new message

Scheme is secure: no adversary can output a valid forgery!

Provable Security

1. Define model & security notion

2. Design cryptoscheme

Usually described in mathematical language

3. Prove security

Reduce security of complex scheme to **simple** assumption, e.g., factoring

Information-theoretic proofs: information is “useless” to the adversary

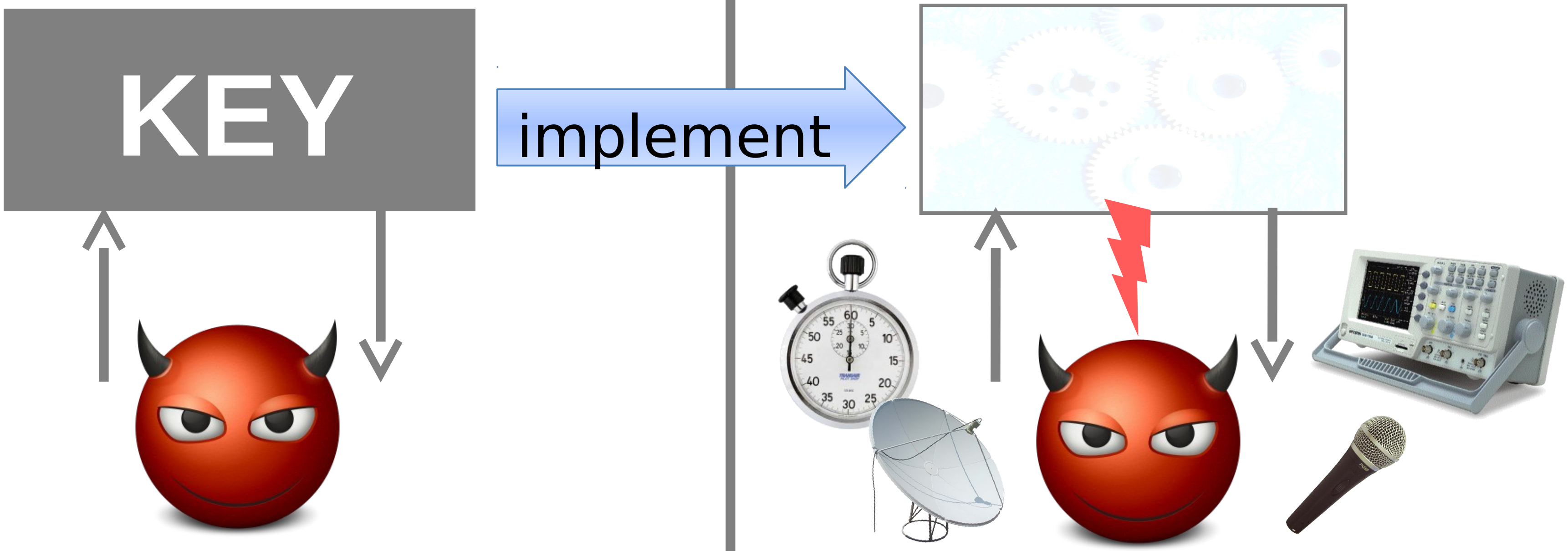
 Shows security not only against **one specific attack**, but **any** attack within the model (if assumption holds)

Security proven but in what model?

Theory vs. Reality

Attack algorithm:

Attack Implementation:



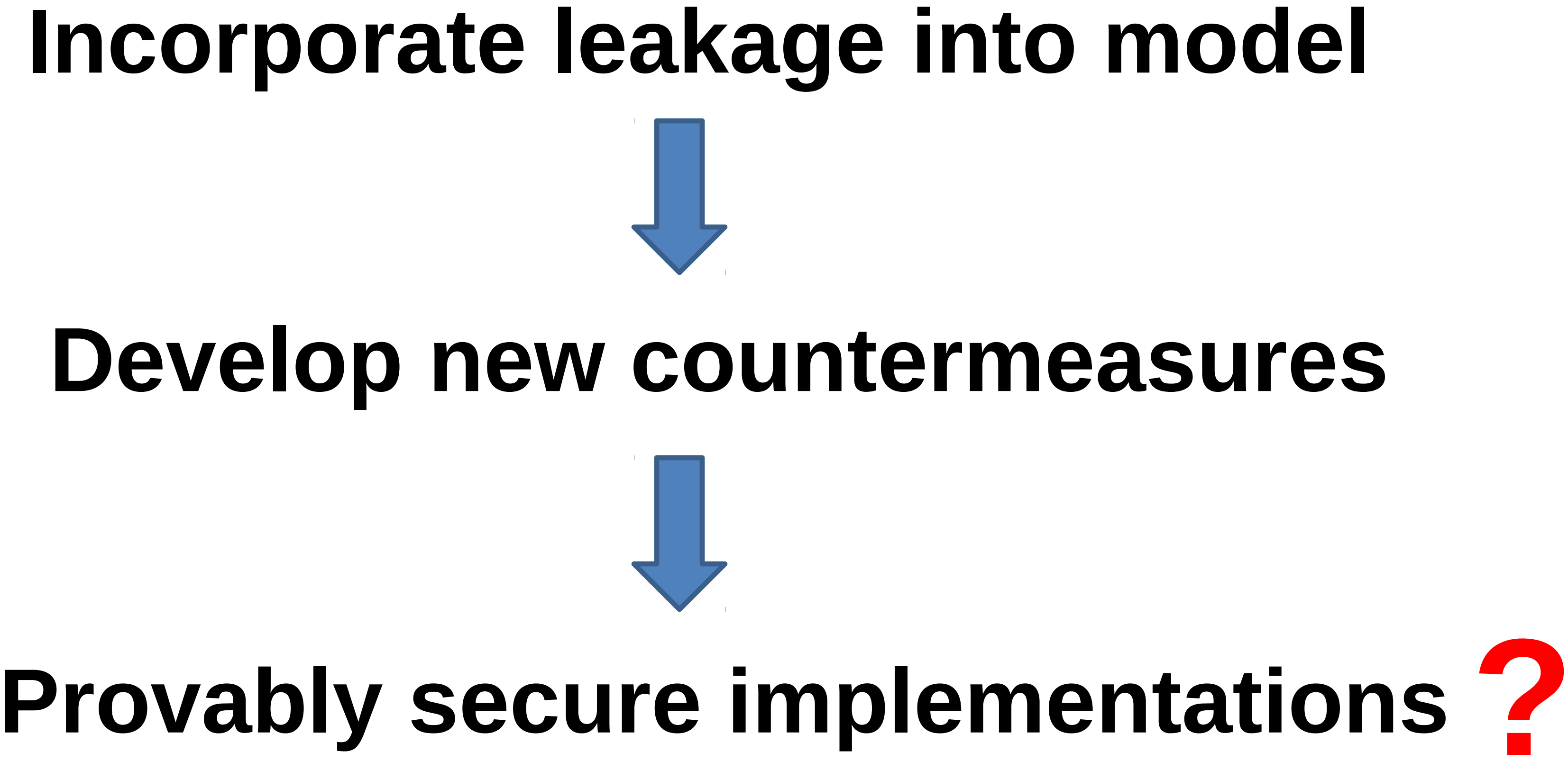
Controls inputs /outputs but internals stay hidden

Devices **leak** about internals

Best attack for AES: **2126.1**

Can break AES within hours with side-channel analysis

Goals of leakage resilience



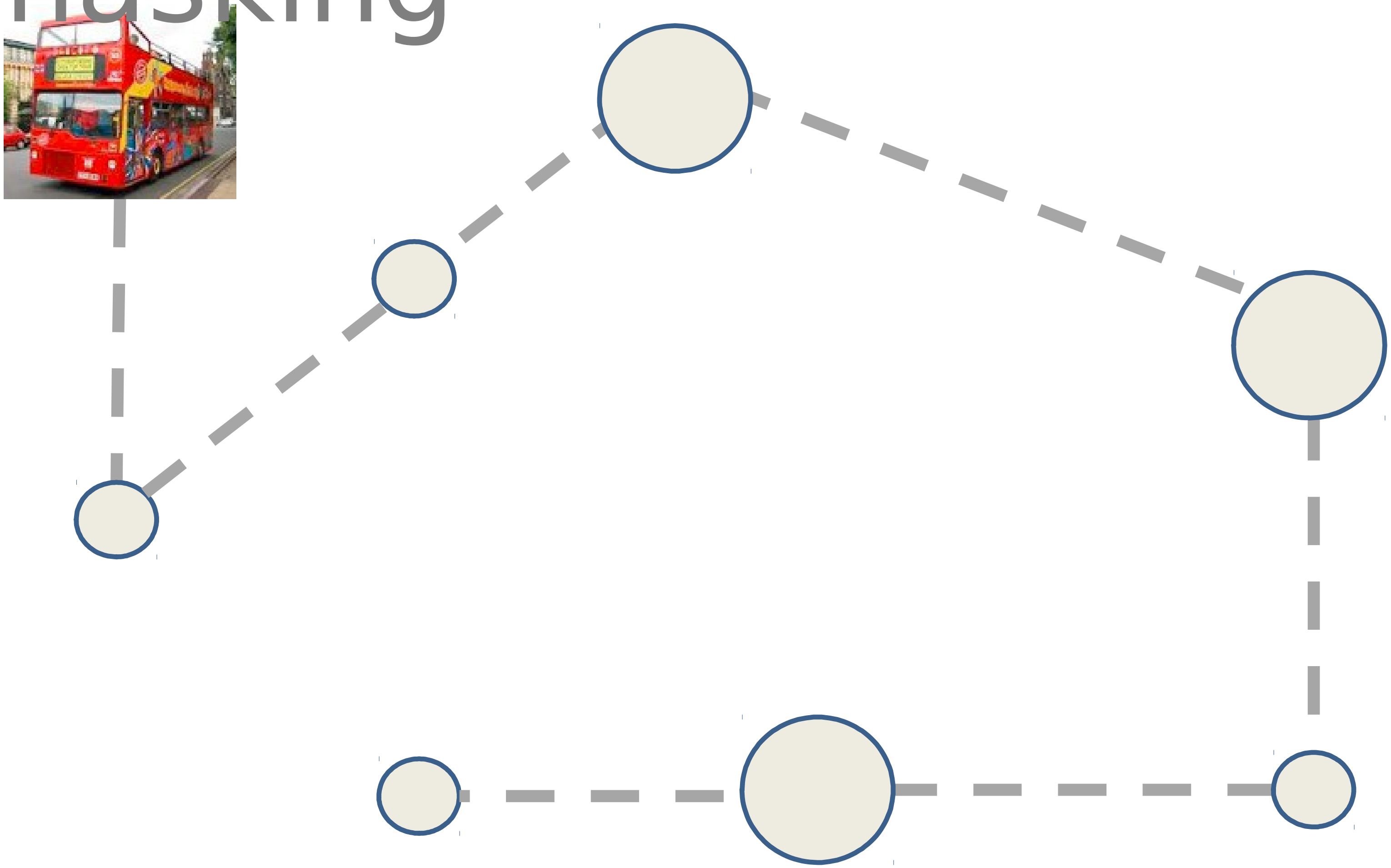
Incorporate leakage into model

Develop new countermeasures

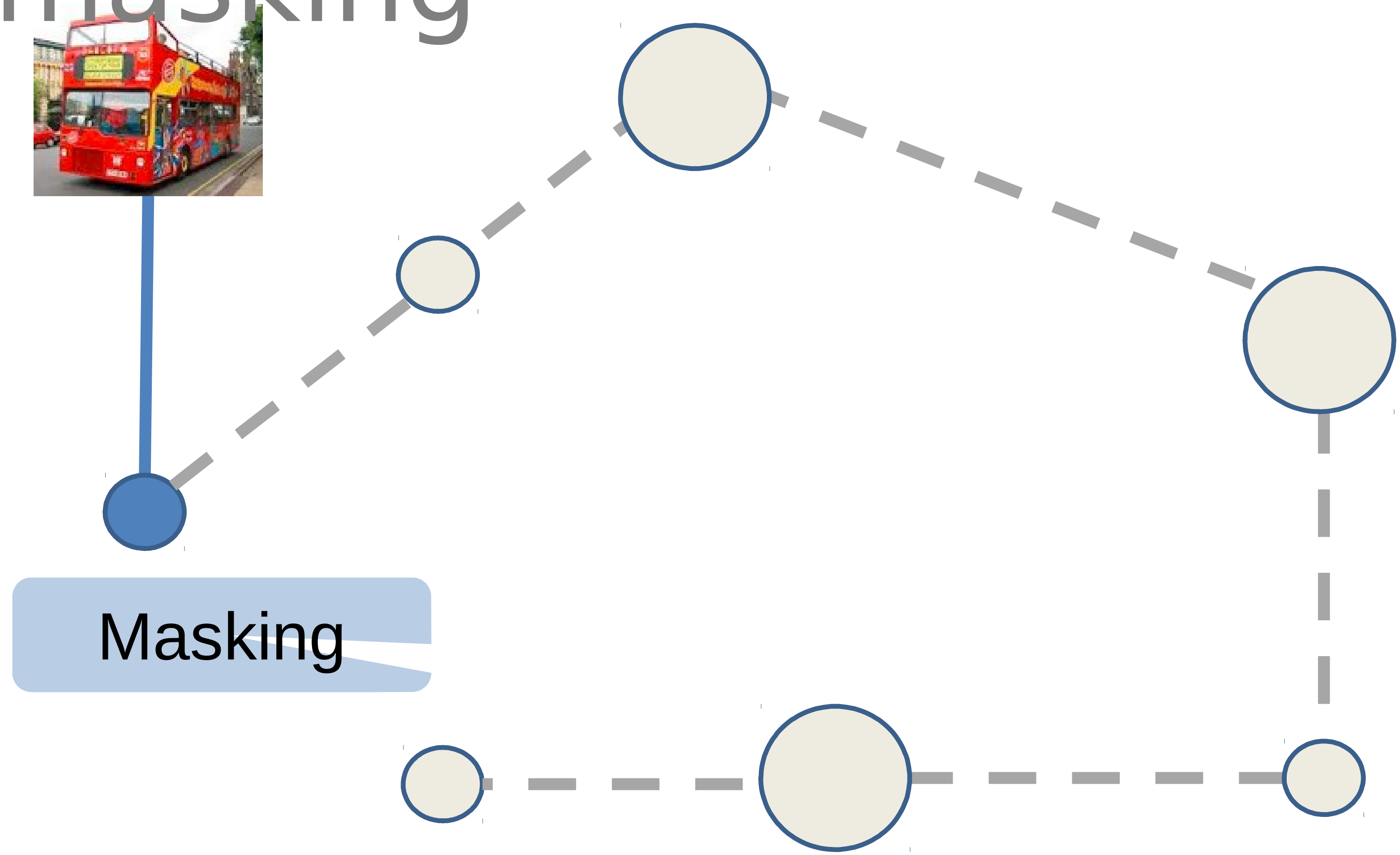
Provably secure implementations ?

Best answered by looking at examples

Leakage models for masking



Leakage models for masking



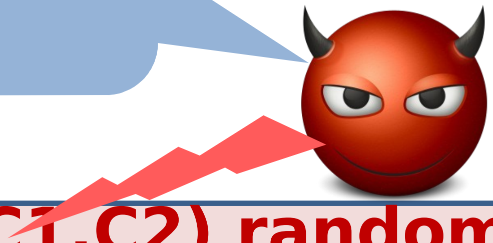
Masking

Basic idea of masking

Comm wer analysis

Idea: p randomized encodi

Learns nothing about S, if leakage depends only on one element



Additive secret sharing:

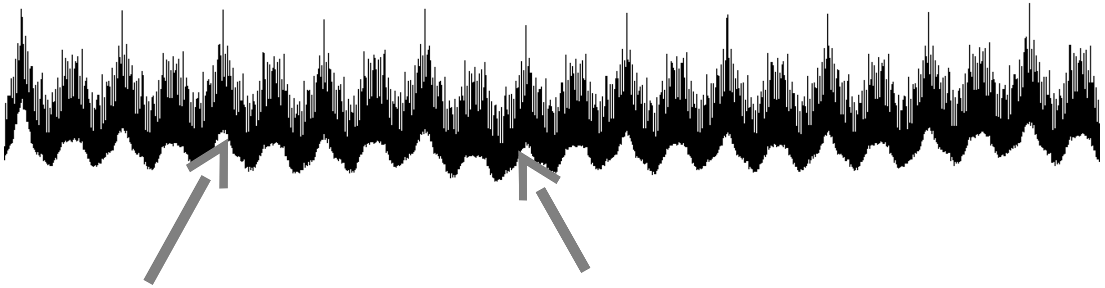
S

Encode

C := (C1,C2) random s.t. S = C1 + C2

Can protect against univariate attacks

Insecure when considering multivariate distributions

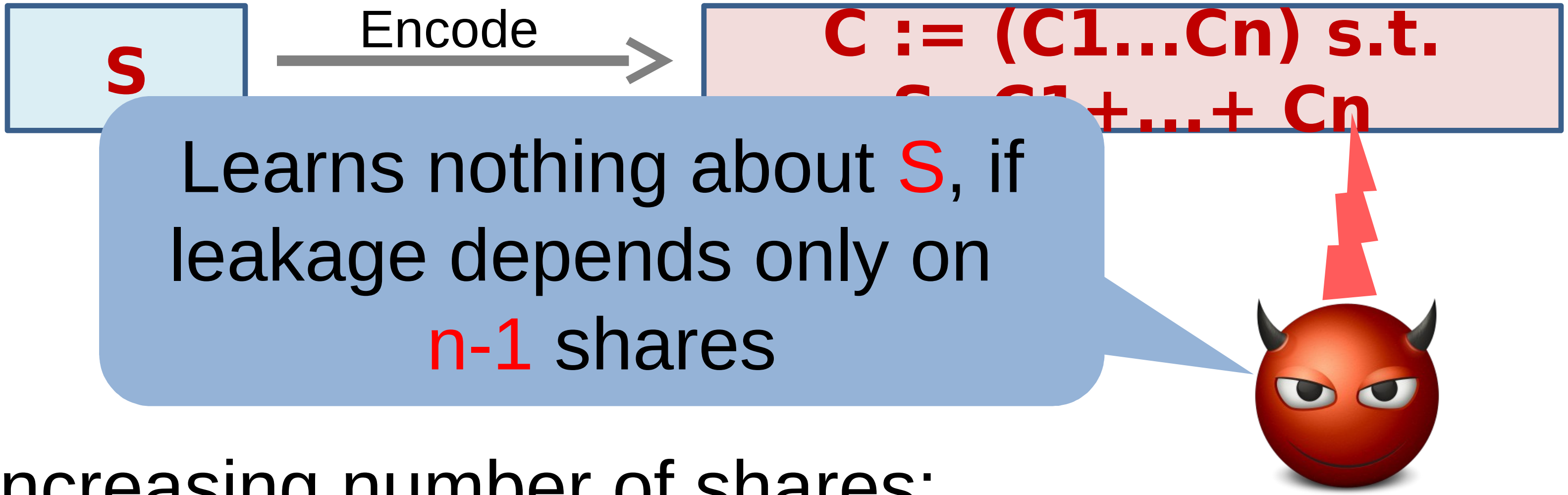


f(C1)

f(C2)

Basic idea of masking

Use **n** shares to protect against **(n-1)**-variate attacks



S

Encode

C := (C1…Cn) s.t.

S = C1+…+ Cn

Learns nothing about S, if leakage depends only on n-1 shares

Increasing number of shares:

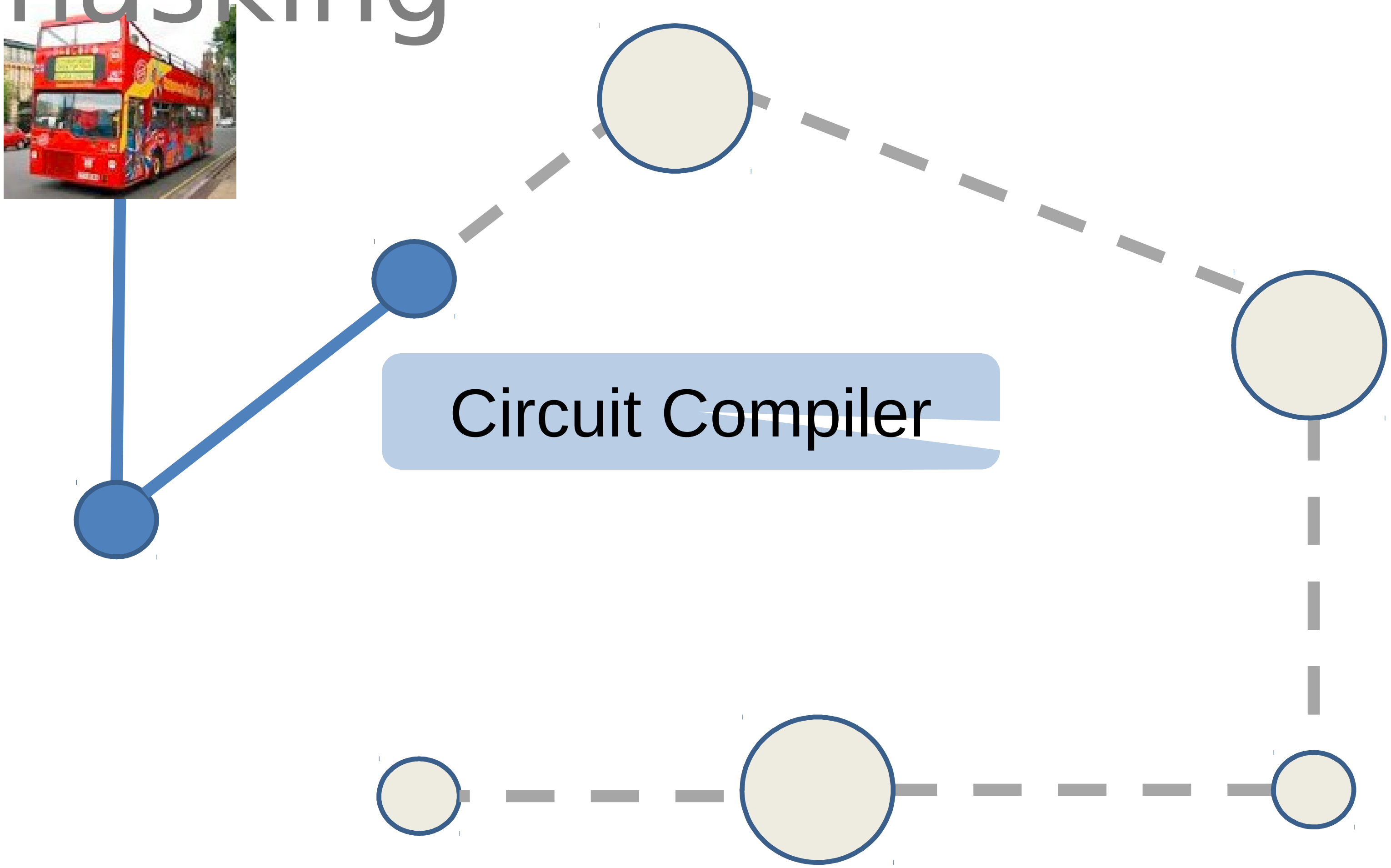
➔ Increases attack order

□ Increases attack difficulty

Two main questions:

- How to use shared secrets to protect cryptoscheme
- How to model security of complext algorithms

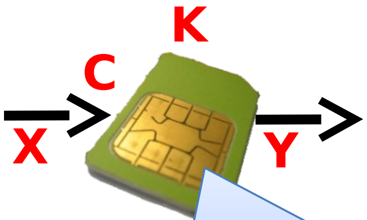
Leakage models for masking



Circuit Compiler

Leakage resilient circuits

Formalization of masking by Ishai-Sahai-Wagner-03

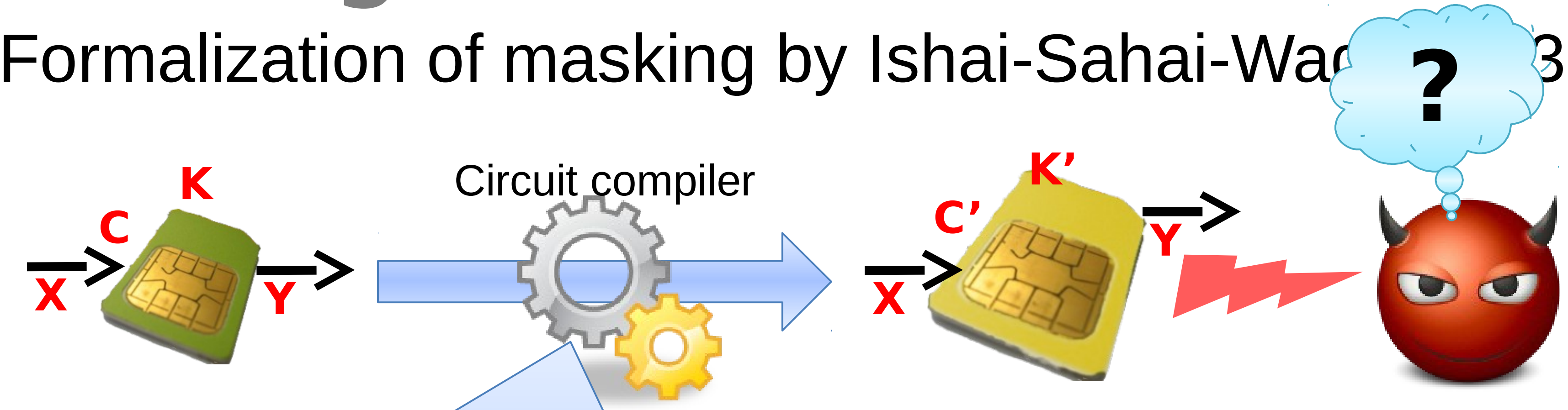


Arbitrary computation modeled as a circuit

Only abstraction to describe „arbitrary computation“ □ can also be software...

Leakage resilient circuits

Formalization of masking by Ishai-Sahai-Wa 3



<u>Circuit Compiler:</u> Run once at production time (no leakage)

<u>Input:</u> Description of circuit **C** with key **K** (e.g., circuit for AES)

<u>Output:</u> Description of circuit **C’** with key **K’** (**C’** is probabilistic)

<u>Correctness:</u> **C[K]** and **C’[K’]** have same functionality

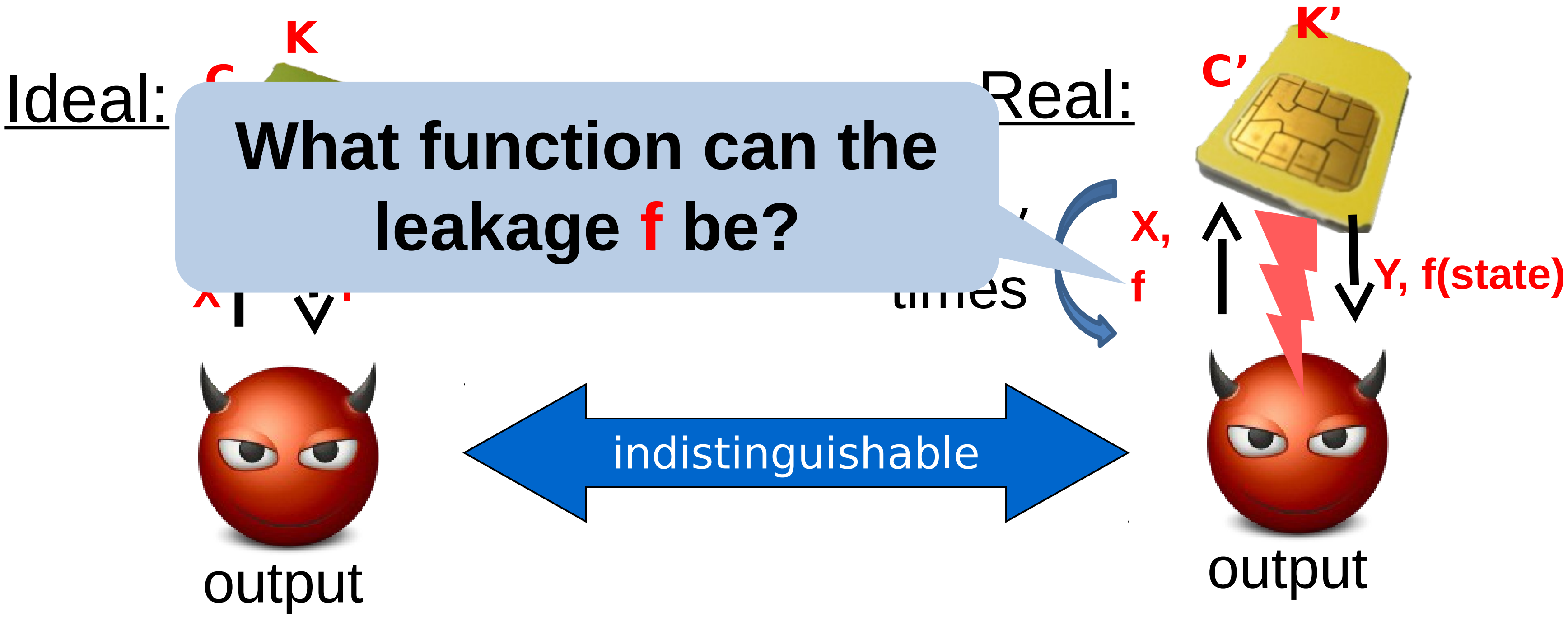
<u>Additionally:</u> **C’[K’]** **leakage resilient** for many executions

□ adversary learns nothing “useful” from leakage

How to formalize this?

Simulation-based security

Adversary learns no more than by black-box access



What function can the leakage f be?

<u>Continuous leakage:</u> many observations are possible

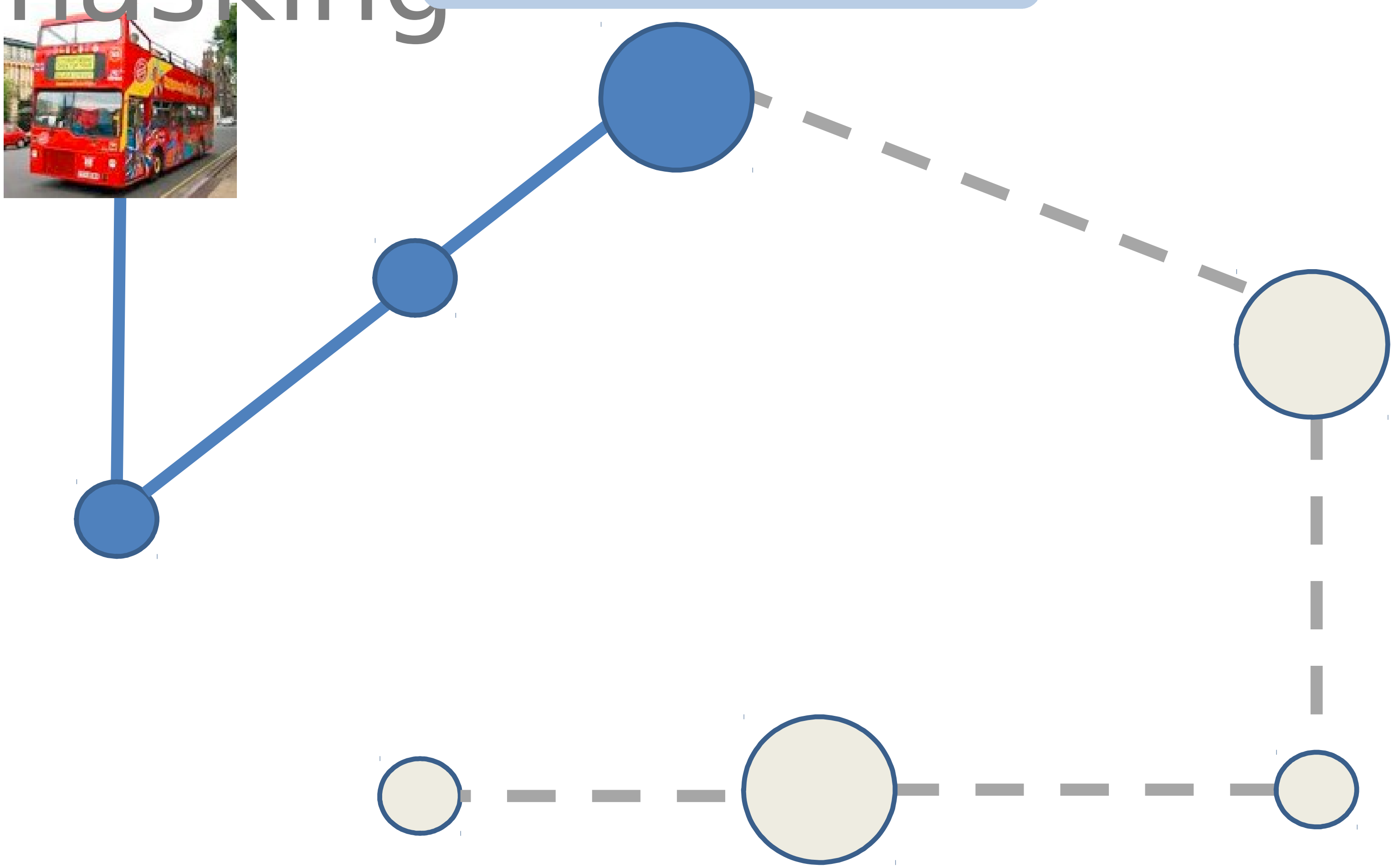
<u>What does it mean?</u>

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

Even more: Cannot break underlying security notion

Leakage models for masking

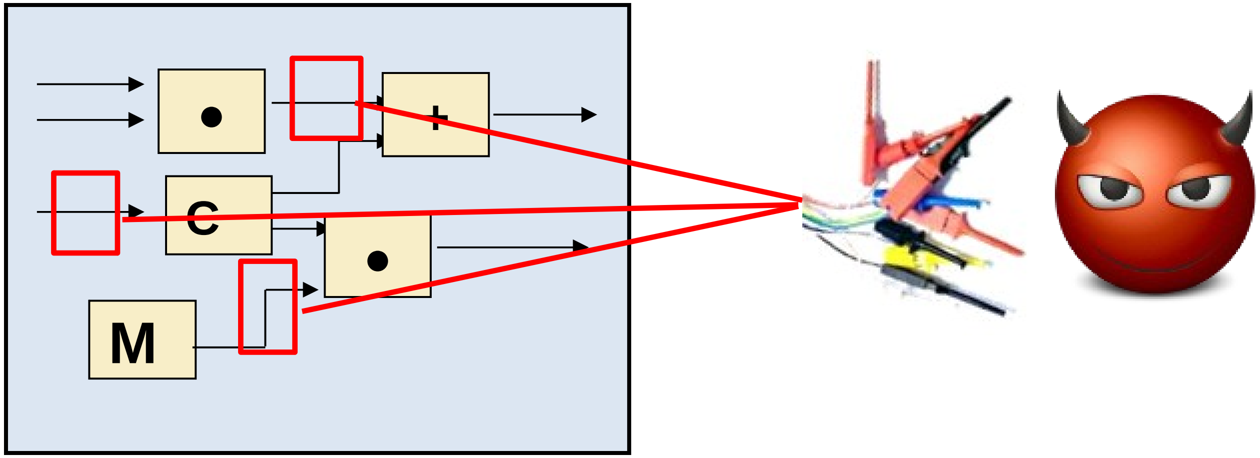
n-probing model



n-Probing adversary (ISW03)

Adversary gets **n** intermediate values of computation

□ **L** = { values on **n** adversarial chosen wires }



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

Basic ingredient: encoding scheme

S $\xrightarrow{\text{Encode}}$ **C := (C1…Cn) s.t. S=C1+…+ Cn**

Insecure in continuous setting!

Continuous leakage

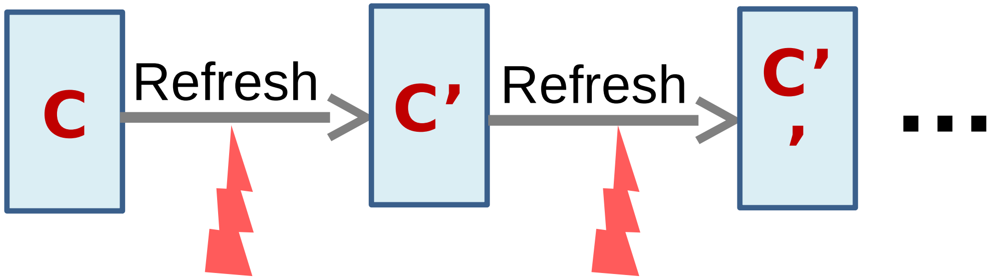
Idea: Prob. algorithm to refresh additive encoding:

Input: **C = Enc(s)** Output: fresh encoding **C’ = Enc(s)**

$$\overbrace{\begin{matrix}\mathbf{C1}\\ \mathbf{C2}\\ \mathbf{...}\\ \mathbf{...}\\ \mathbf{Cn+1}\end{matrix}}^{\mathbf{Enc(s)}} + \overbrace{\begin{matrix}\mathbf{R1}\\ \mathbf{R2}\\ \mathbf{...}\\ \mathbf{...}\\ \mathbf{Rn+1}\end{matrix}}^{\mathbf{Enc(0)}} = \overbrace{\begin{matrix}\mathbf{C'1}\\ \mathbf{C'2}\\ \mathbf{...}\\ \mathbf{...}\\ \mathbf{C'n+1}\end{matrix}}^{\mathbf{Enc(s)}}$$

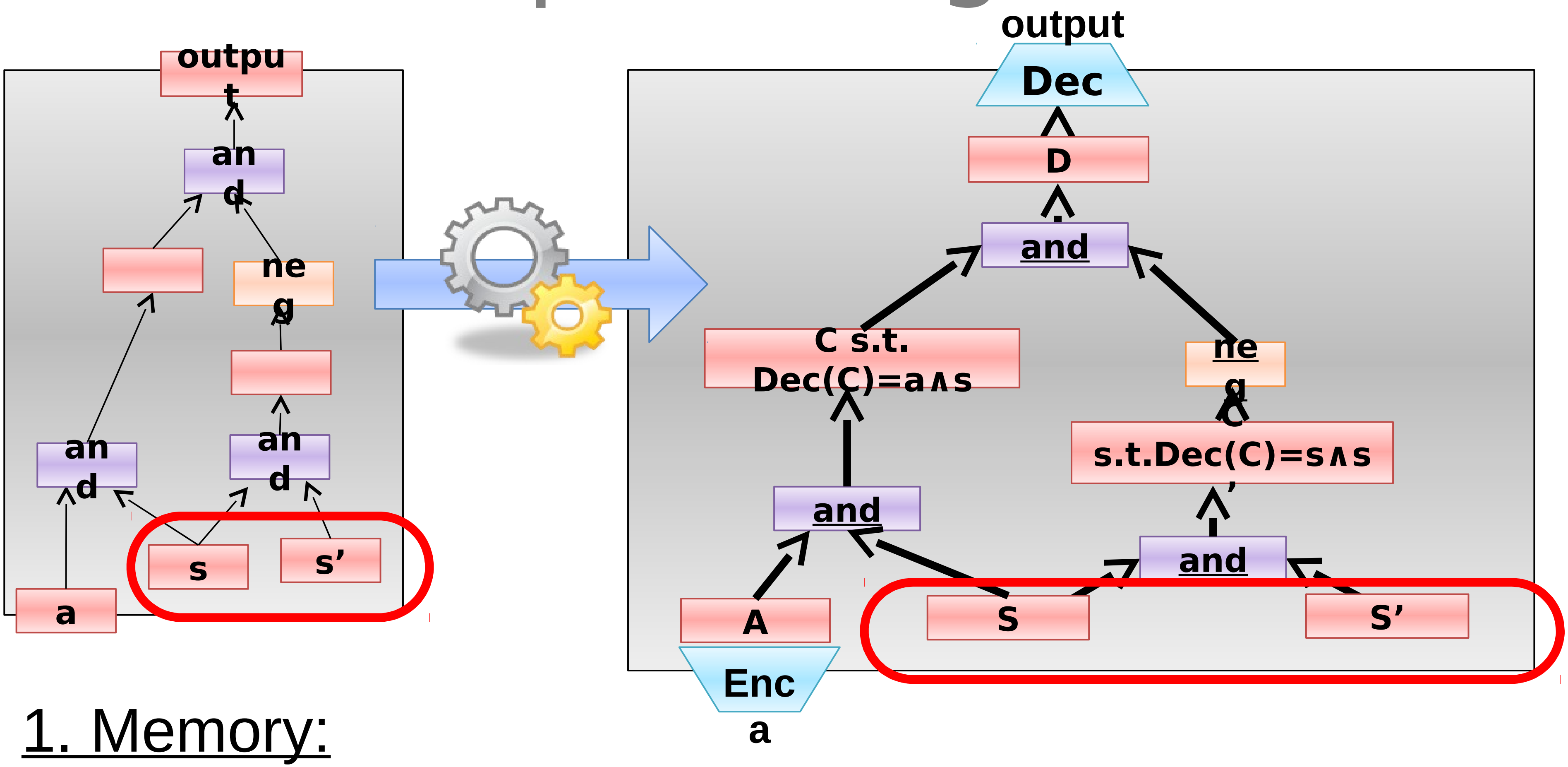
Correctness: By linearity **Enc(s) + Enc(0) = Enc(s)**

Security:



Secure for **n/2** probes per execution

ISW Compiler: High level

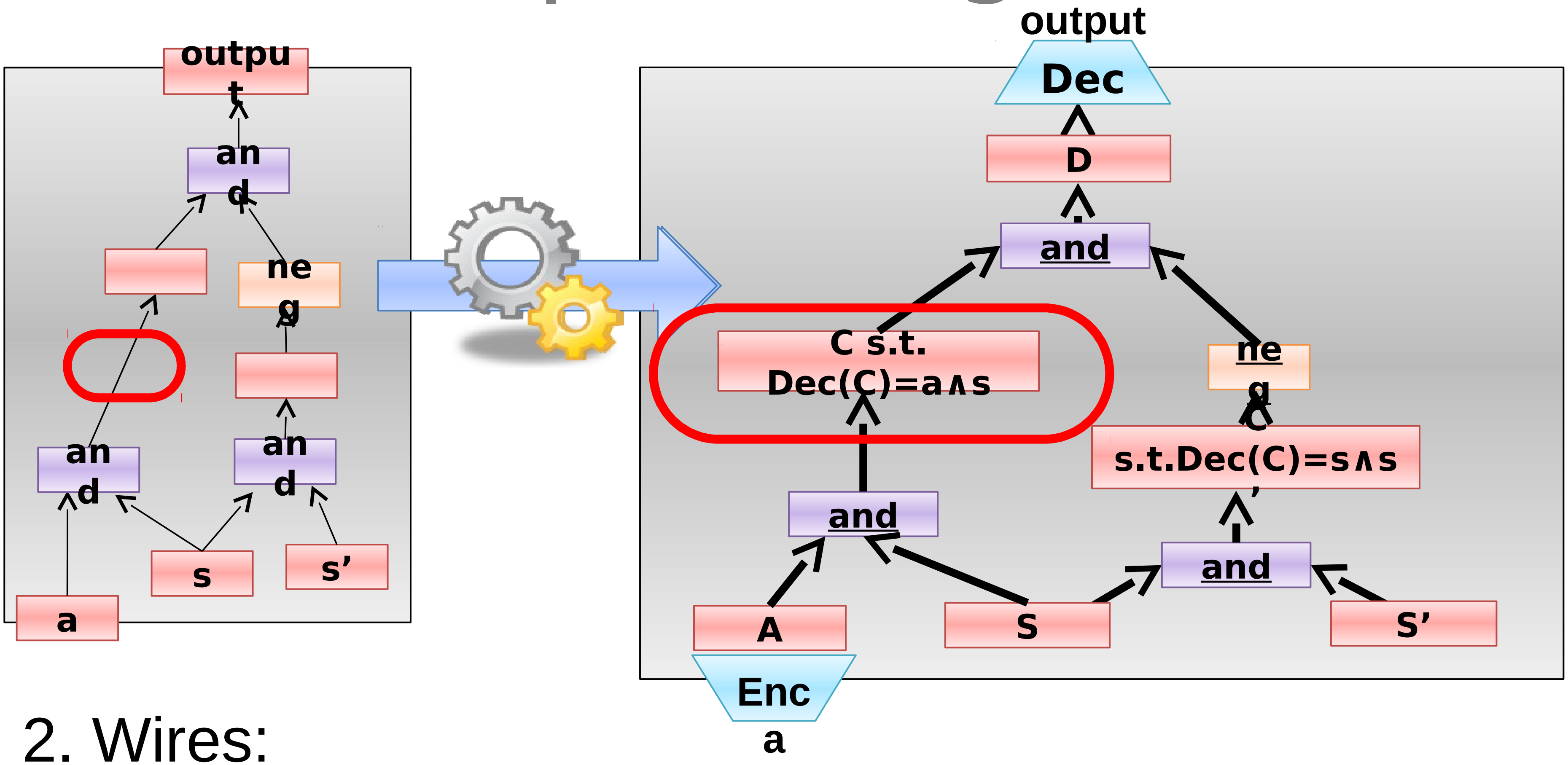


1. Memory:

A bit **s**

Encoded with Boolean masking, i.e., **S = (S1…Sn+1)** such that **s = S1 + … + Sn+1**

ISW Compiler: High level



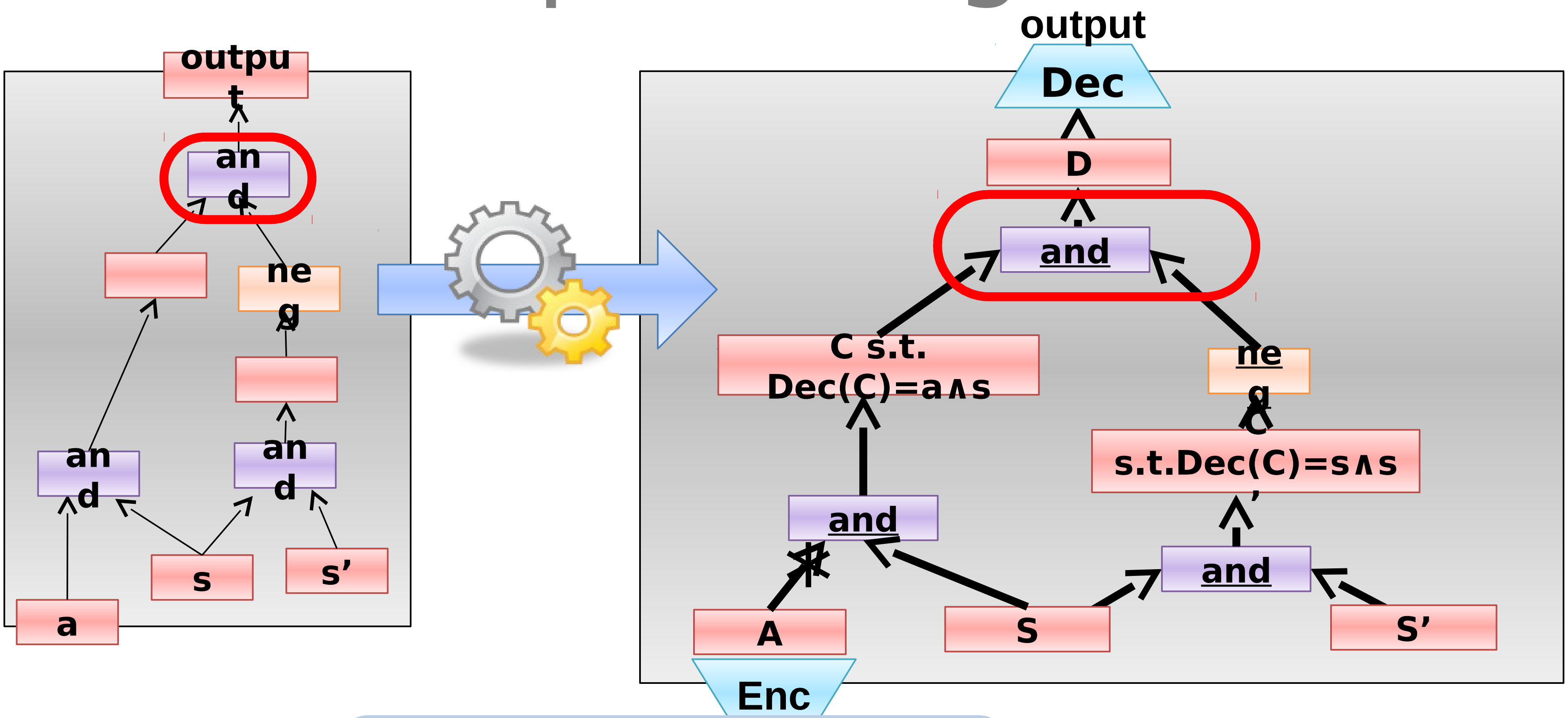
2. Wires:

Each wire
w = a ∧ b

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

Main challenge: computing on encoded inputs!

ISW Compiler: High level



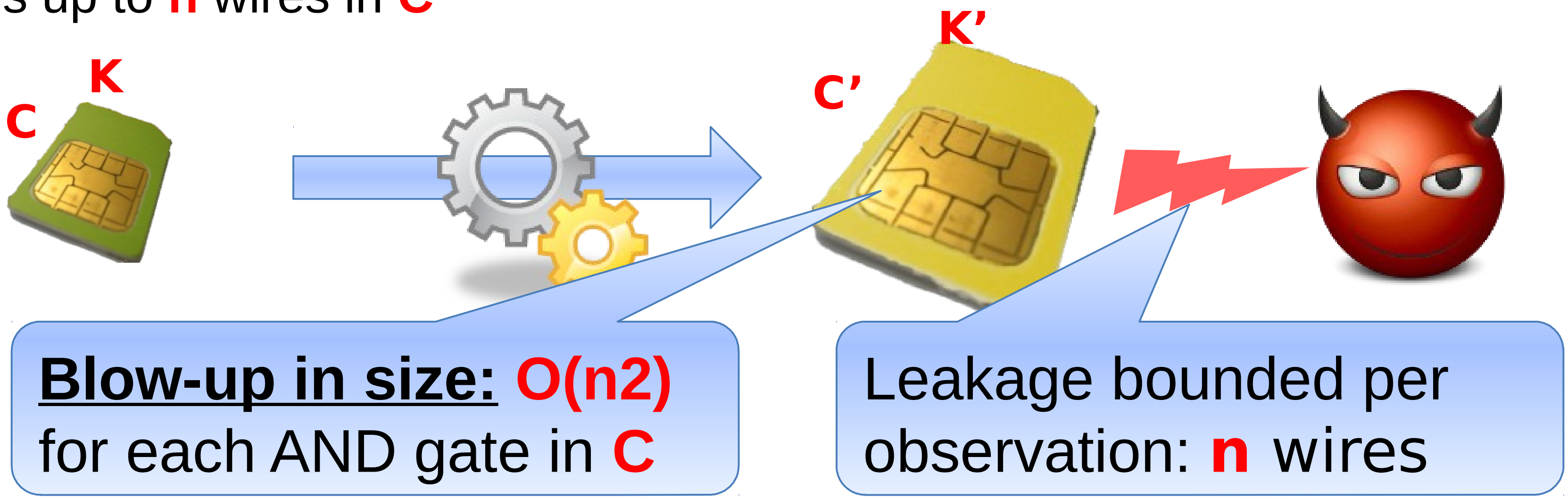
Uses refreshing protocol

3. Gates: uilt from standard ga erating on encodings

Main challenge: algorithm to securely compute AND!

ISW Compiler: Results

Theorem: A compiler that makes **any circuit** resilient to adversary that probes up to **n** wires in **C'**



Blow-up in size: O(n2) for each AND gate in **C**

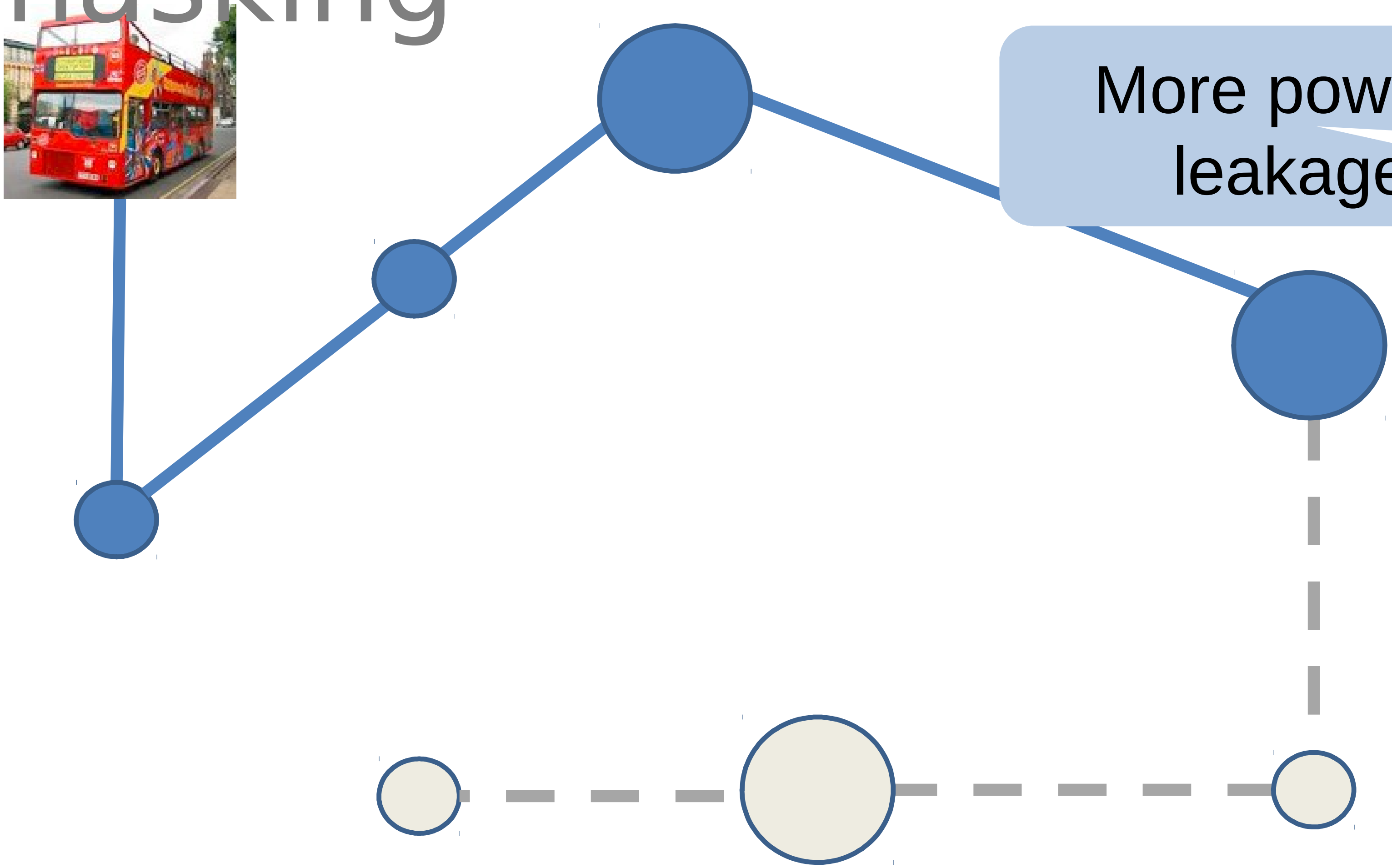
Leakage bounded per observation: **n** wires

Proofs in n-probing model: Systematic and simple tool to find **n**-th variate flaws in masking schemes

Prouff-Rivain-2010: Larger fields & more efficient

Drawback: **L** only probing □ oblivious of many wires

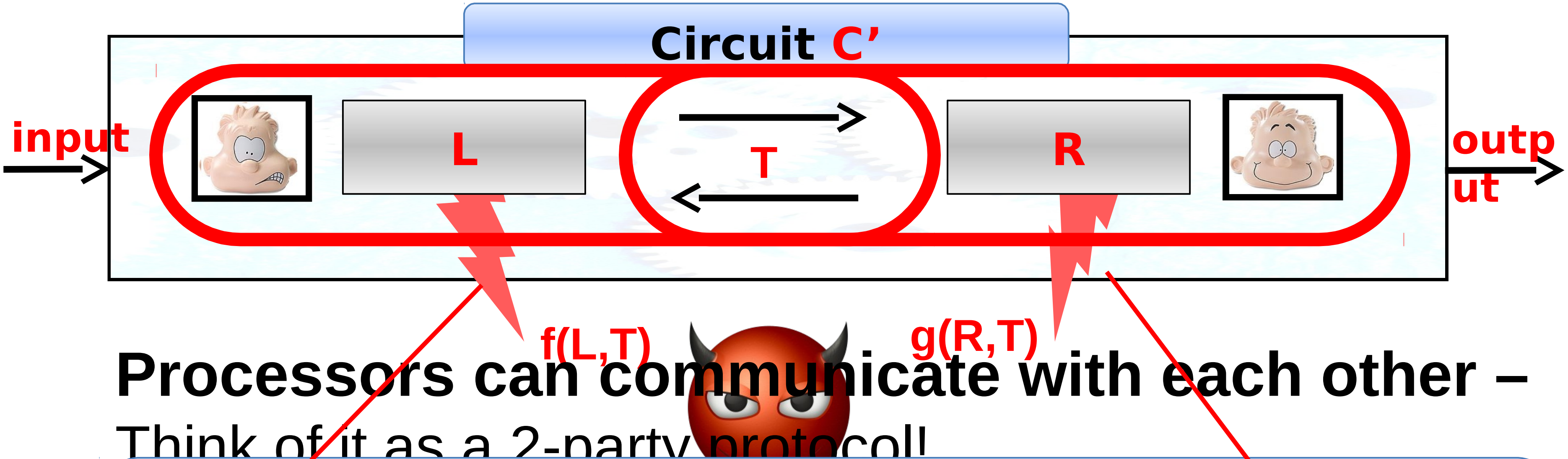
Leakage models for masking



More powerful leakages

New model for circuits

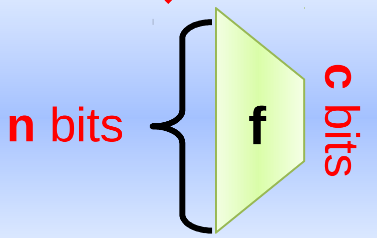
Bounded independent leakages



Processors can communicate with each other –

Think of it as a 2-party protocol!

dependently!



Bounded leakage function:

- Arbitrary efficient functions

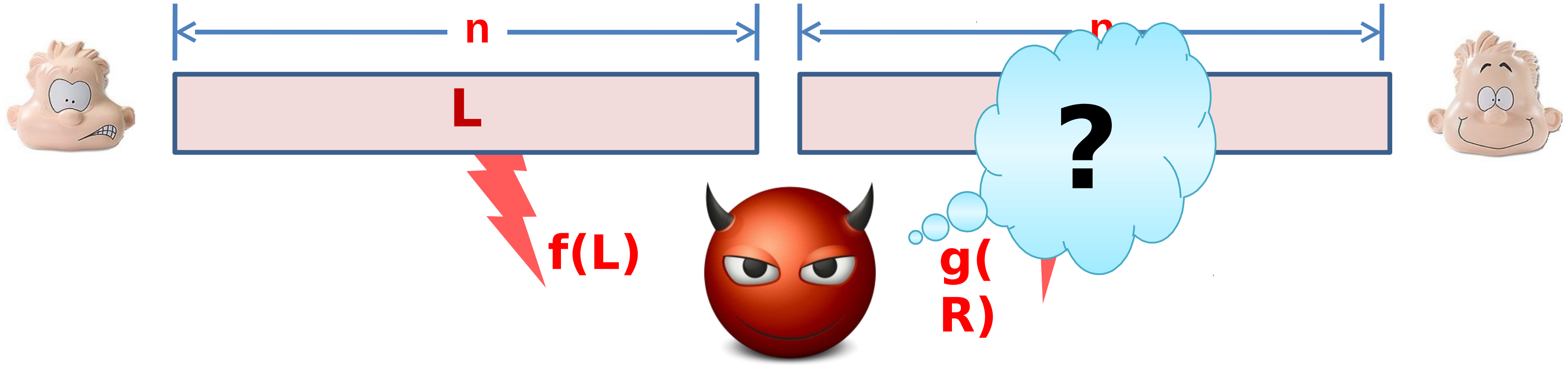
- Ony restriction: input shrinking, i.e., **c < n**

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

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

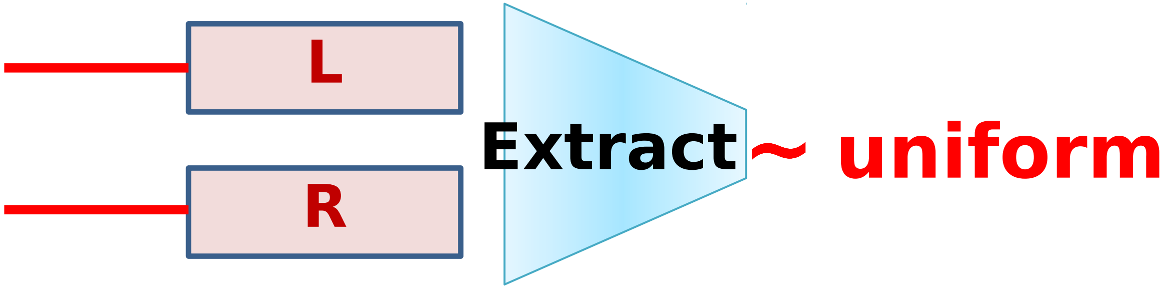
Inner Product Masking

Sample **L,R** uniformly in **{0,1}n** s.t. **S= <L,R> = Σ Li*Ri** and store parts separately on two processors



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

High min-entropy and independent sources

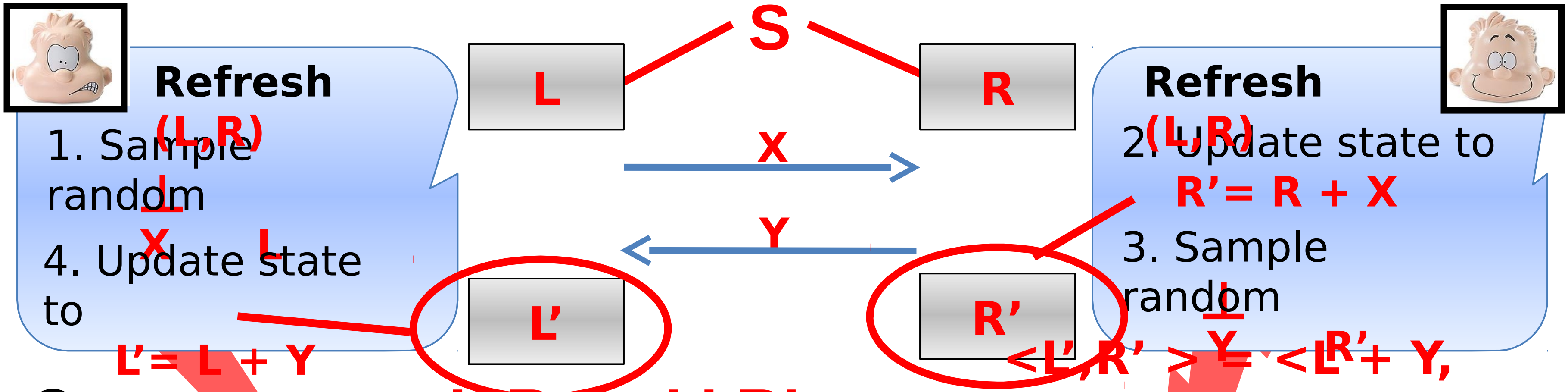


Continuous setting?

Idea: refreshing protocol for IP maskings –

Prob. algorithm: **(L,R) □ (L’,R’)** fresh encoding of **<L,R>**

Simple attempt:



Correctness: **<L,R>=<L’,R’>**

<L’,R’ > = <L + Y, R’>

= <L,R’> + <Y,R’>

= <L,R + X>

= <L,R> +<L, X>

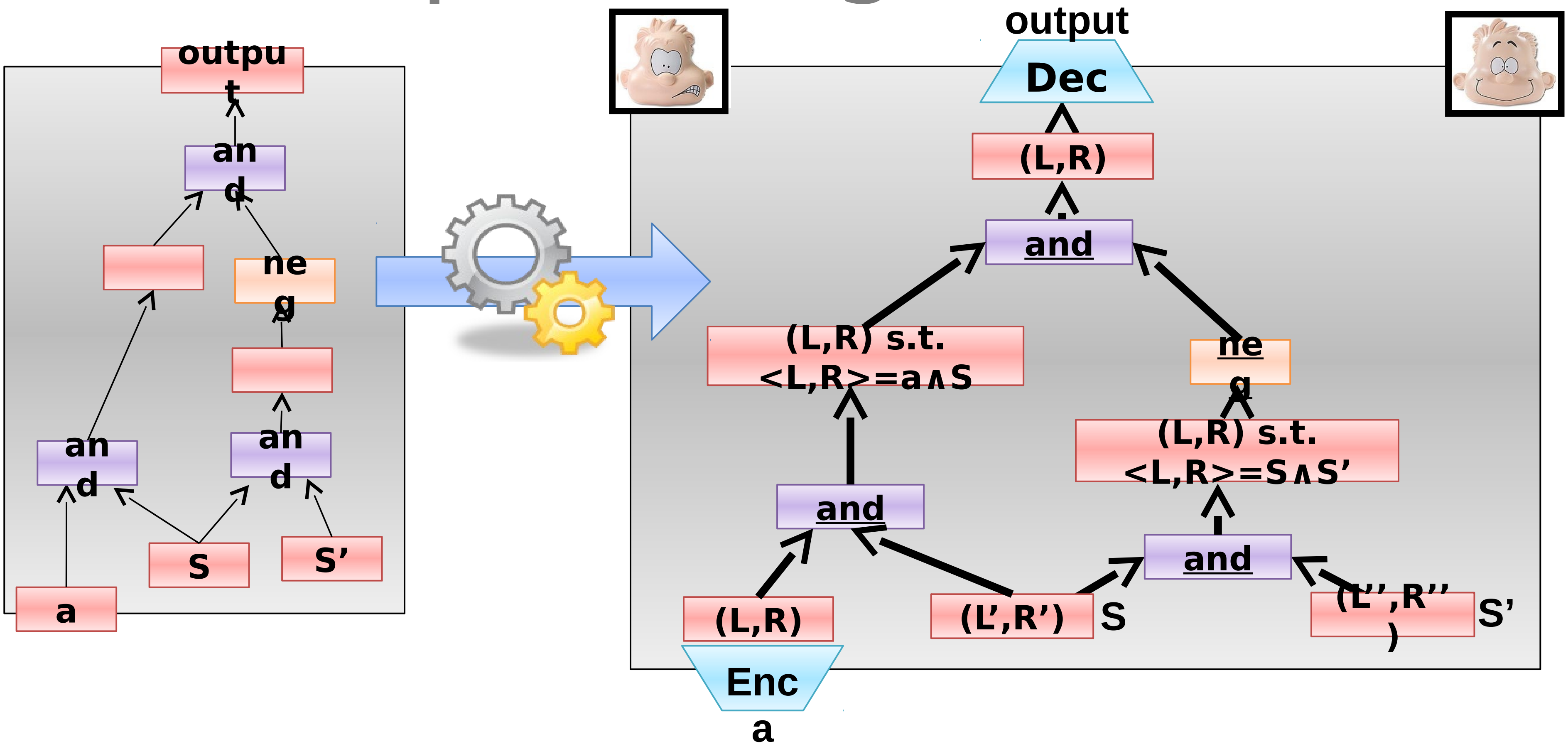
f(L,X

Contrived attack reveals **S**

□ cannot be prove

<u>Solution:</u> New protocol with leak-free source [DF12]

IP Compiler: High level

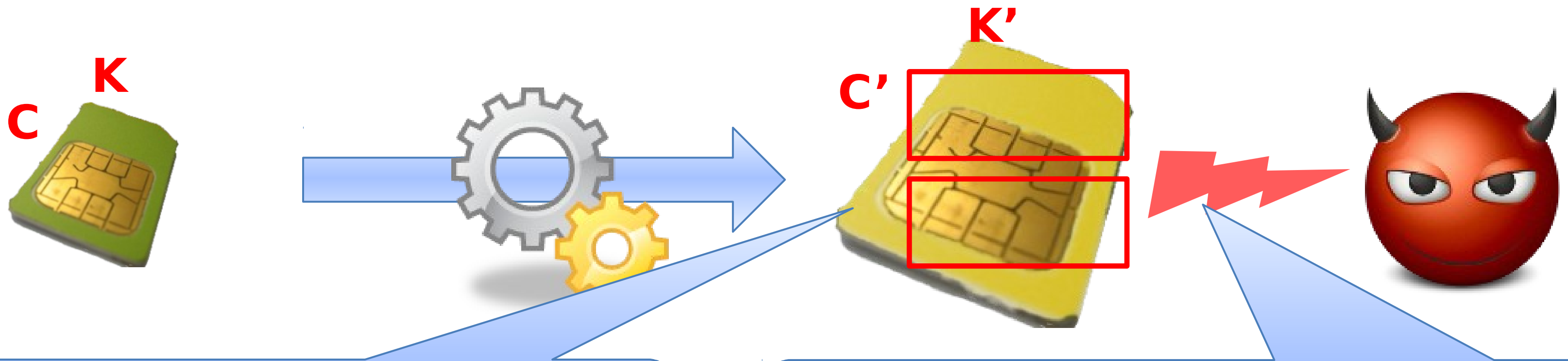


1. Wires and state is encoded using IP masking
2. Gates are replaced by protocols working on IP masking

- **Most difficult:** protocol to compute AND (see DF12)

The IP masking compiler

Theorem [DF12]:
A new **information theoretic** secure compiler with security against **continuous independent leakage**

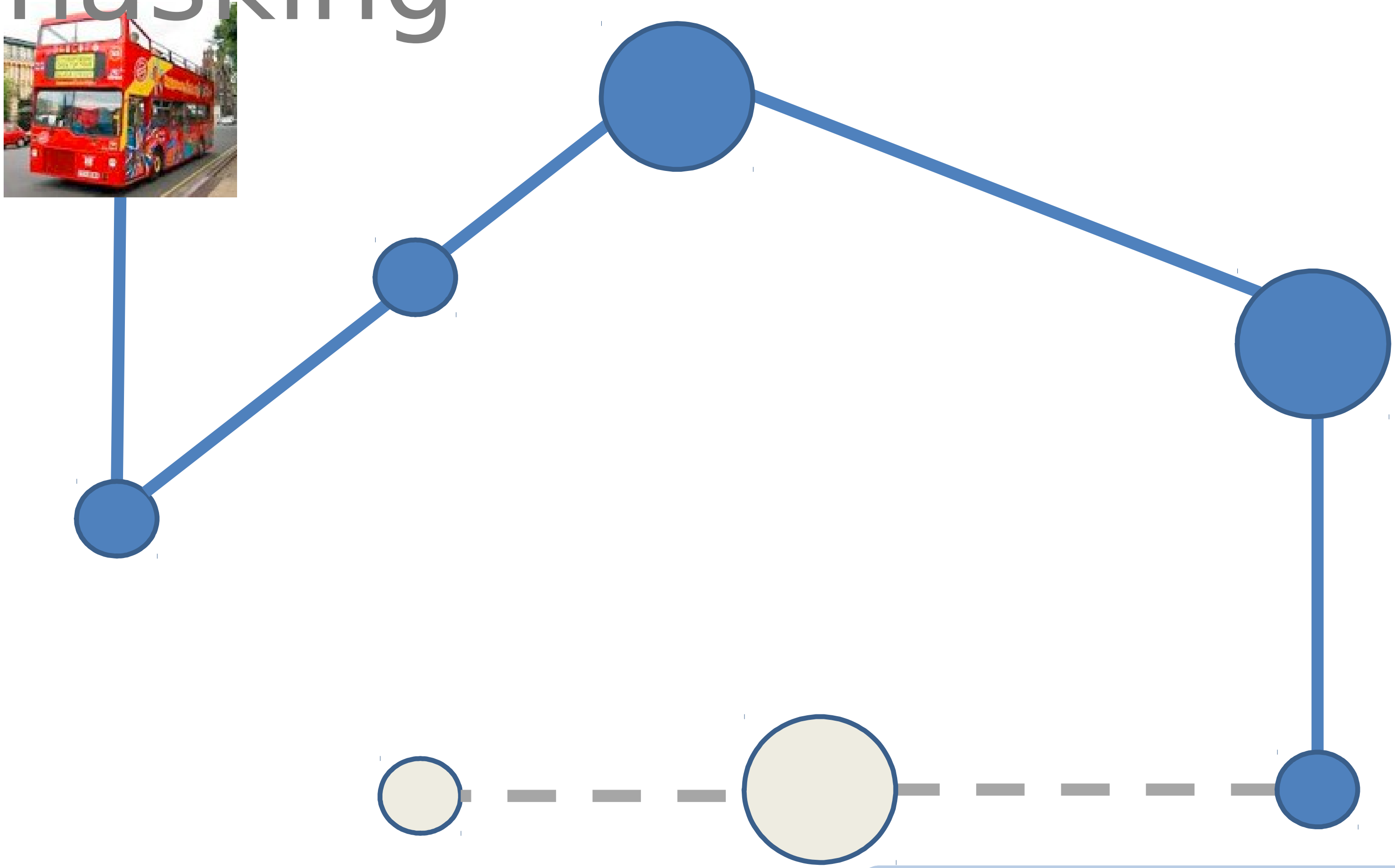


<u>Blow-up in size:</u> O(n2)
for each AND gate in **C**

Leakage bounded per exec.:
c bits from each processor

IP masking in practice?

Leakage models for masking



IP masking in practice

IP Masking in practice?

(BFGV12)

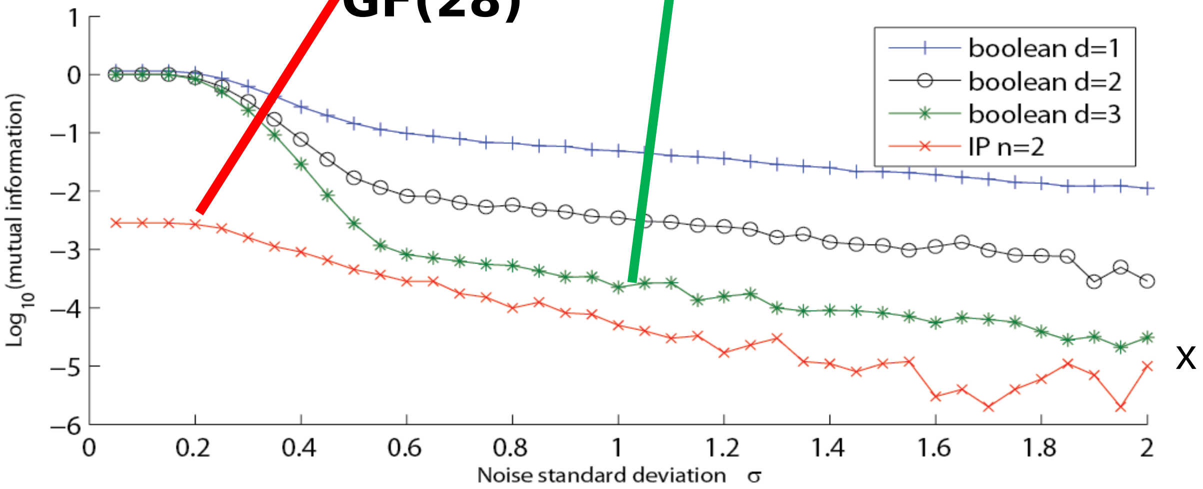
Analyzed for small s **-**

Security outperforms

<u>Green curve:</u> Boolean masking with 3 random shares in GF(28)

<u>Red curve:</u> IP masking with 3 random shares in GF(28)

Mutual information between HW leakage and secret



Weaker dependency between leakage & secret for IP masking

Main reason: Non-linear masking vs. linear masking

Implementation of AES

IP masked AES on 8-bit microcontroller

IP Masking “lifted” to GF(28)

Performance: Runs in 1.9 Mio clock cycles for **n=2**

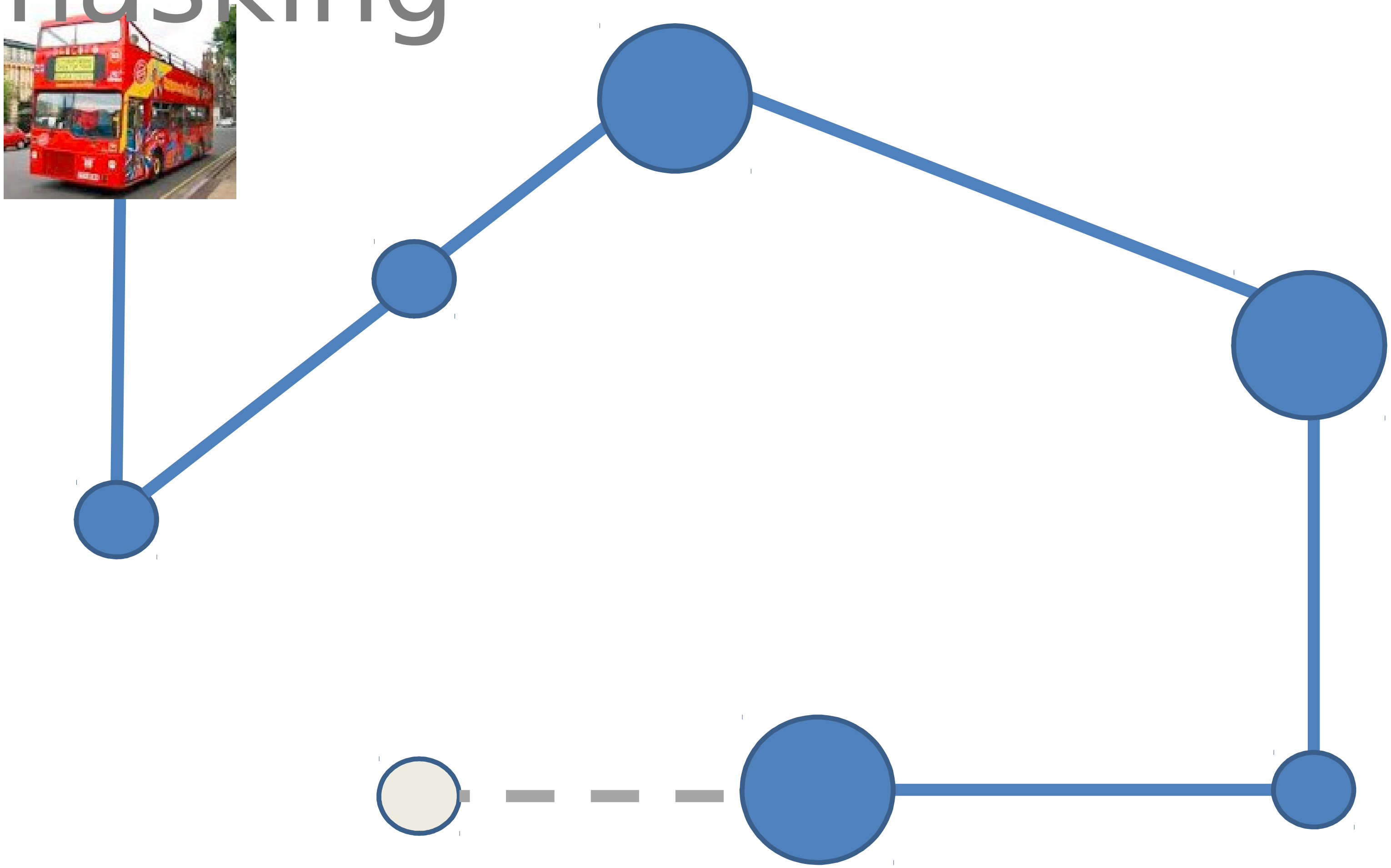
Minimize costs for masked multiplication:

- Use squaring whenever possible it’s cheap!
- Minimize multiplications in SubBytes
- Refreshing with complexity **O(n)** instead **O(n2)**

Unfortunately small univariate bias in IP-masking
[Prouff-Rivain-Roche-14]

But: Bias is small Future work: still exp. security?

Leakage models for masking



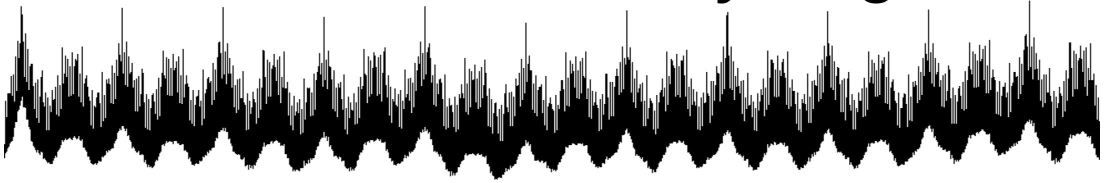
Noisy leakages

Bounded leakage in practice

Theoretician‘s perspecitve: beautiful concept ♥

Are leakages bounded? Probably not...

- Measurements described by large data ☹



- Not clear how to guarantee/verify bounded leakages in practice ☹

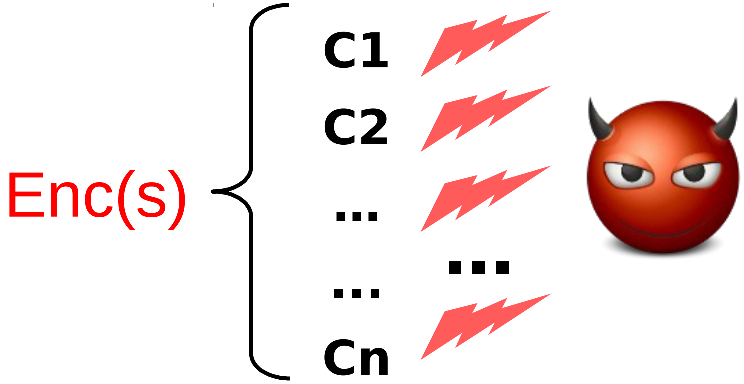
Physical leakages are inherently noisy

Difficulty in many attacks: how to eliminate the noise?

Noisy leakages

Noisy leakage model: Chari et al. Crypto'99

No quantitative bound on leakage, but leakage is noisy



Enc(s)

C1

C2

...

...

Cn

...

Leakage is **Ci** + Gaussian noise

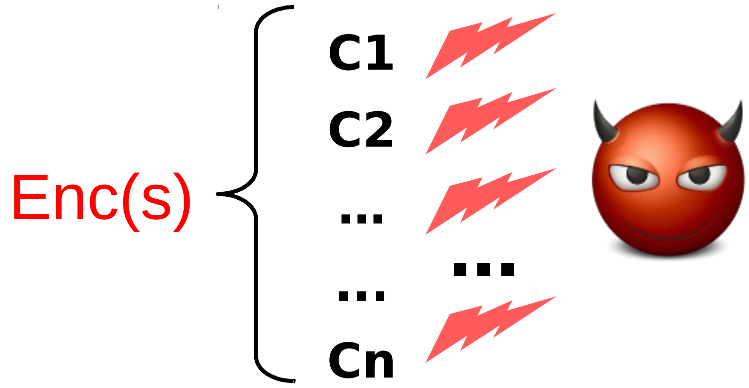
Chari et al. only consider security of a single masked secret

Long-standing open question: Generlize to computation

Prouff-Rivain, Eurocrypt 13:

- Prove security of a masked implementation of the AES
- Generlized noise model (not only Gaussian noise)

Noisy functions



Enc(s)

C1

C2

...

...

...

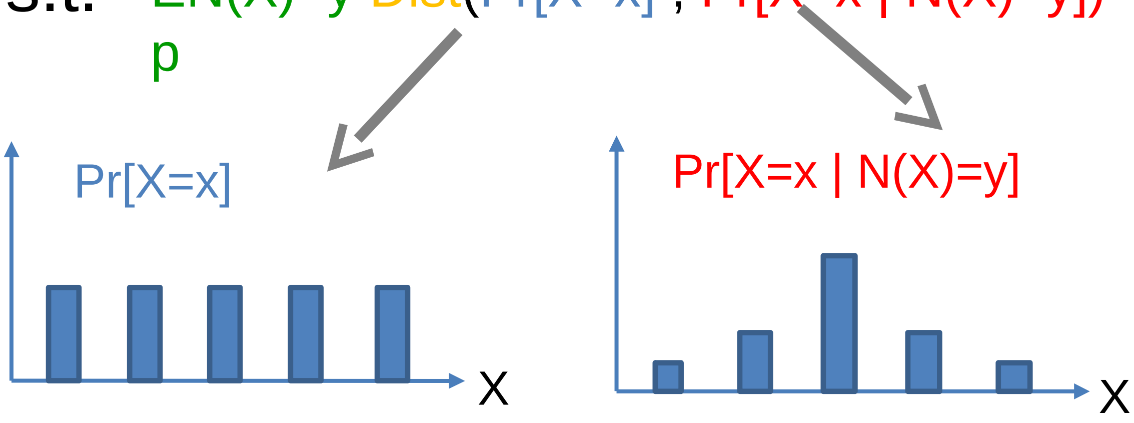
Cn

Noisy function N: adv. learns N(**Ci**)

e.g. N(**Ci**): compute Hamming weight and add Gaussian noise

All p-noisy functions N s.t. EN(X)=y Dist(Pr[X=x] ; Pr[X=x | N(X)=y]) < p

Weighted average over Noise distribution

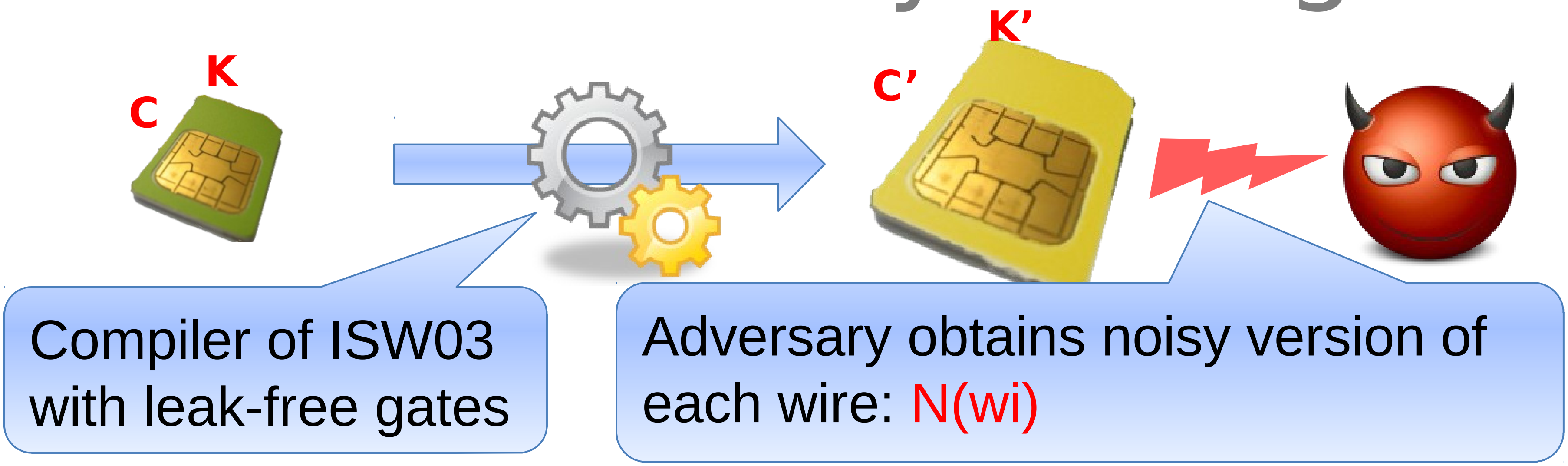


Alternative interpretation: MI(X, N(X)) < |X| p

Example p = 0: N is very noisy = non-informative leakage

Example p ≈ 1: N is identity = very informative leakage

Circuits for noisy leakage



Compiler of ISW03 with leak-free gates

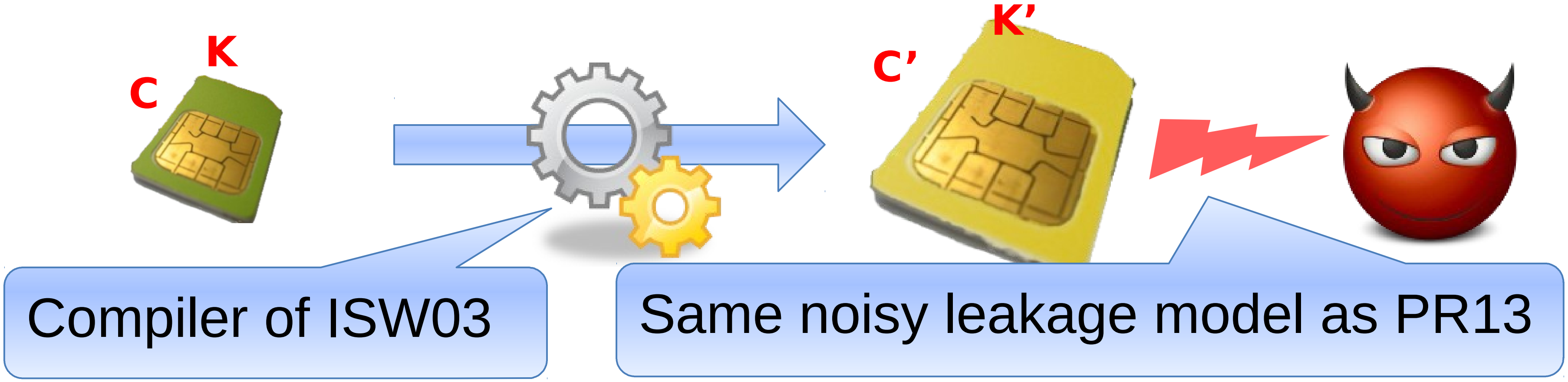
Adversary obtains noisy version of each wire: N(wi)

No quantitative bound on amount of leakage

Drawbacks of the analysis:

- Leak-free gates: no leakage from refreshing
- Security argument only for random-message attack
- Very technical proof

Duc-Dziembowski-F 14:



Compiler of ISW03

Same noisy leakage model as PR13

ISW03 is secure against noisy leakages

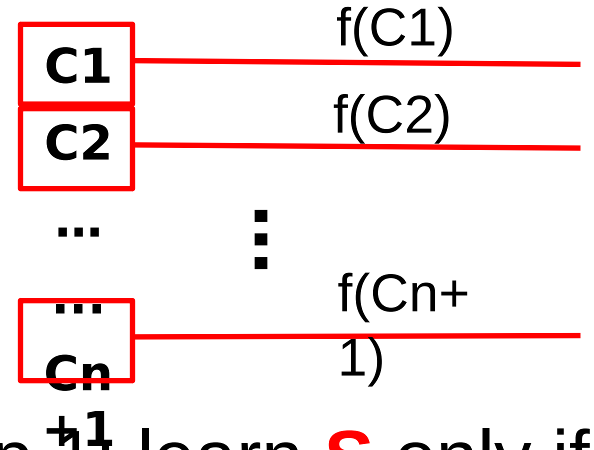
- No leak-free gates ☺
- Full simulation-based security analysis ☺
- Unifying leakage models: ☺

  n-probing security ▯ security against noisy leakage

↘ <u>Nice tool:</u> proofs in n-probing model much simpler than proofs in noisy model

Proof idea

New simpler noise model: Random probing



f(Ci) = Ci with prob. q;
otherwise f(Ci) = „?“

Step 1: learn **S** only if „lucky“ for each random probe

secure in **n**-probing □ secure in random probing

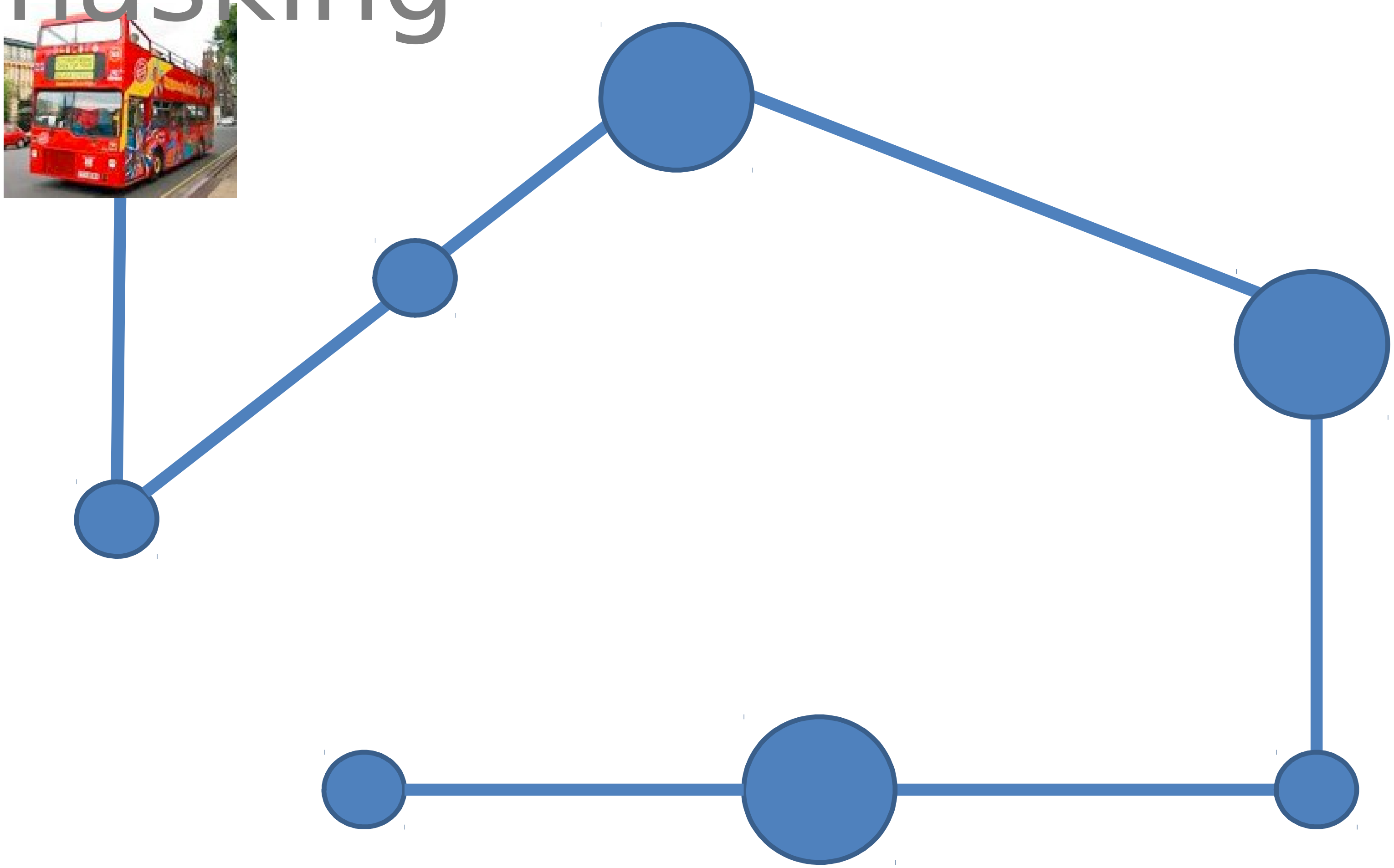
Step 2: noisy leakage = random probing (technical)

For any p-noisy function N there exists a simulated noise distribution N ‘ s.t. for any x: N‘(f(x)) = N (x)

(f is a q-random probing function with q < p|X|)

(1) + (2): n-probing □ secure against noisy leakages

Leakage models for masking



Provably secure implementations?

Provably secure? ***Probably not y***

Why leakage resilient crypto?

Theoretician's answer: Beautiful & natural questions

Is cryptography possible with weak (= non-uniform) keys?

Why to care in practice? Proofs are powerful tool!

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

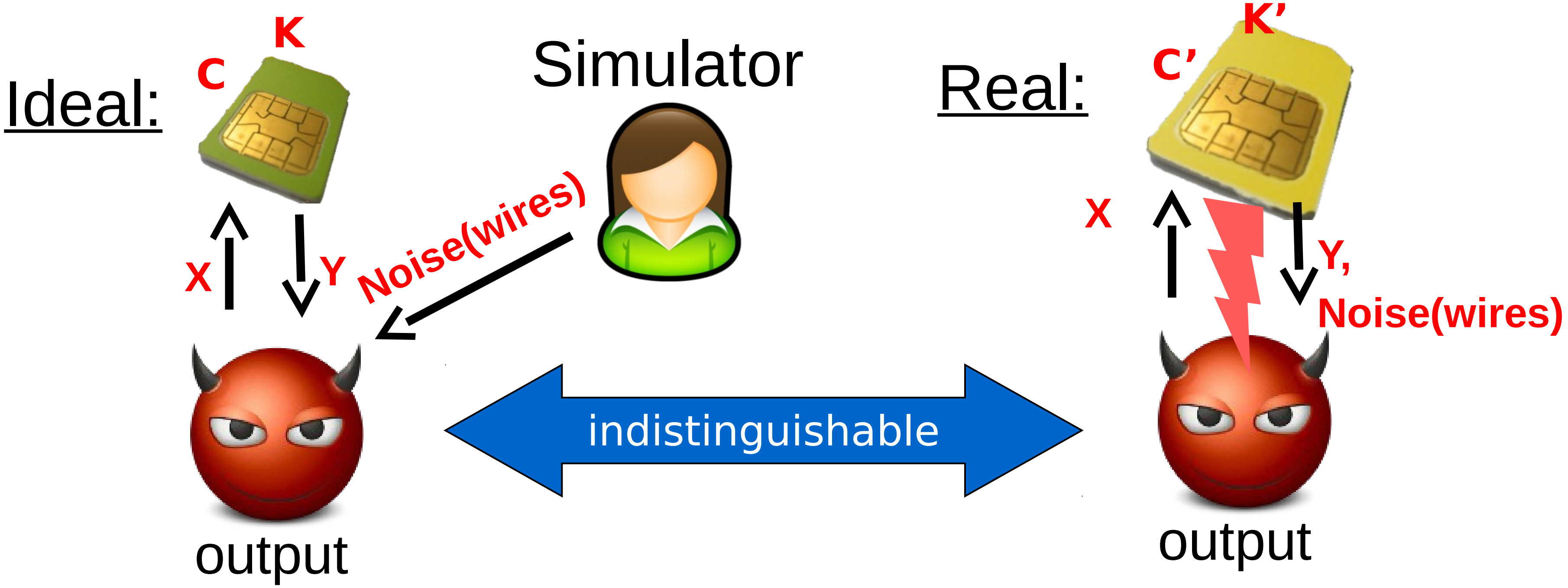
 How much noise do I need to use masking?

Thank you!

Thanks to the EU/FP7 for funding.
Thanks to organizers of COSADE for inviting me.
Thanks to co-authors for nice research questions and fun collaborations.
Thanks to Google for pictures

Security notion

Adversary learns no more than by black-box access



Ideal:

C

K

X

Y

Noise(wires)

Simulator

output

indistinguishable

Real:

C'

K'

X

Y,

Noise(wires)

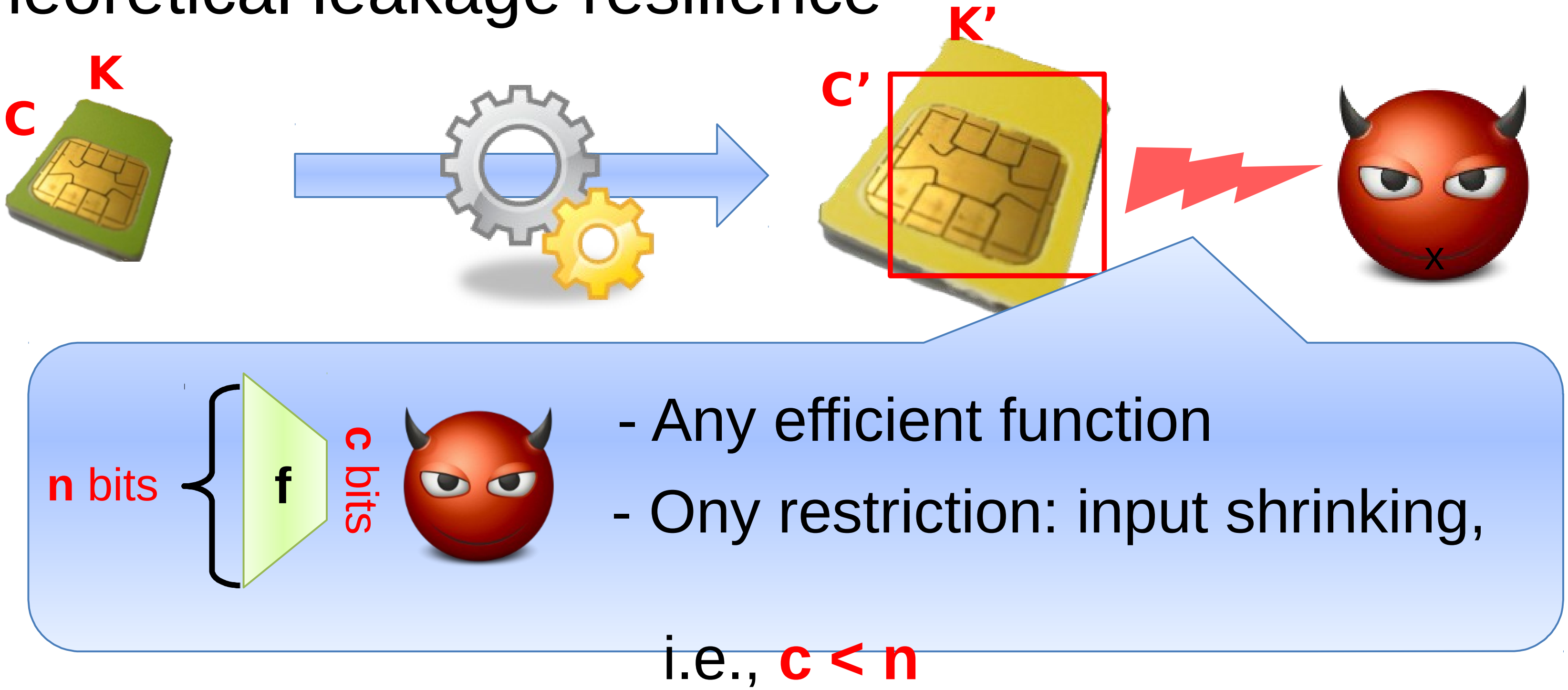
output

Standard proof method: build simulator that can simulate environment (=leakage) for adversary

- Adversary believes he is in real world
- Outputs are indistinguishable

Prominent model in theory

Bounded leakages: used in most papers on theoretical leakage resilience



- Any efficient function

- Ony restriction: input shrinking,

i.e., **c < n**

Bounded leakage: natural and clean abstraction

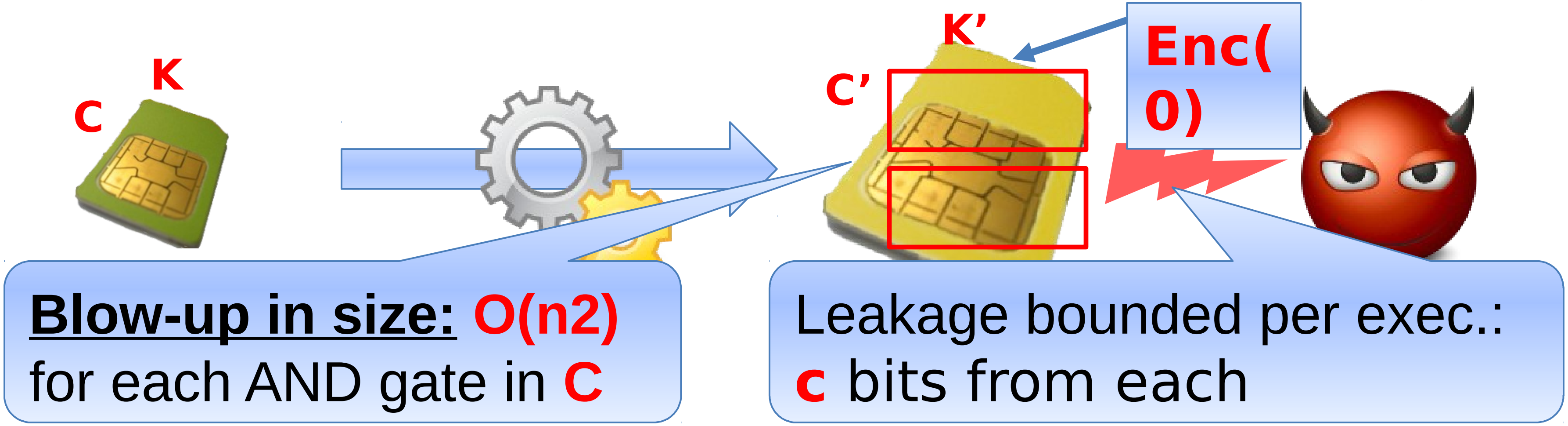
□ „everything leaks“

Impossible to build leakage resilient circuit compilers

The IP masking compiler

Theorem [DF12]:

A new **information theoretic** secure compiler with security against **continuous independent leakage**



<u>Blow-up in size:</u> O(n2) for each AND gate in **C**

Leakage bounded per exec.: **c** bits from each processor

Leak-free gate: leaks on inputs but not from internals

Enc(0) → (A,B) s.t. <A,B> = 0

Goldwasser-Rothblum-2012: Eliminate leak-free gates

□ Much less efficient!