Kolokwium Wydziału MIM UW,
Jan 12, 2017

Non-Malleable Codes

Stefan Dziembowski
University of Warsaw

European Funds
Smart Growth

FNP
Foundation for Polish Science

European Union
European Regional Development Fund

Non-Malleable Codes

a tool for securing
computer memory against
physical tampering
attacks

Introduced in **[D., Pietrzak, Wichs, ICS 2010]**.

Plan

1. Short introduction to physical attacks
2. Non-malleable codes – the definition
3. Non-malleable codes – constructions secure w.r.t different function families:
    1. bit-wise tampering
    2. tempering functions from sets of bounded size
    3. split-state model
4. Subsequent work

Cryptography: art vs science

In the past:

the **art** of encrypting messages (mostly for the military applications).

design method: “trial and error’

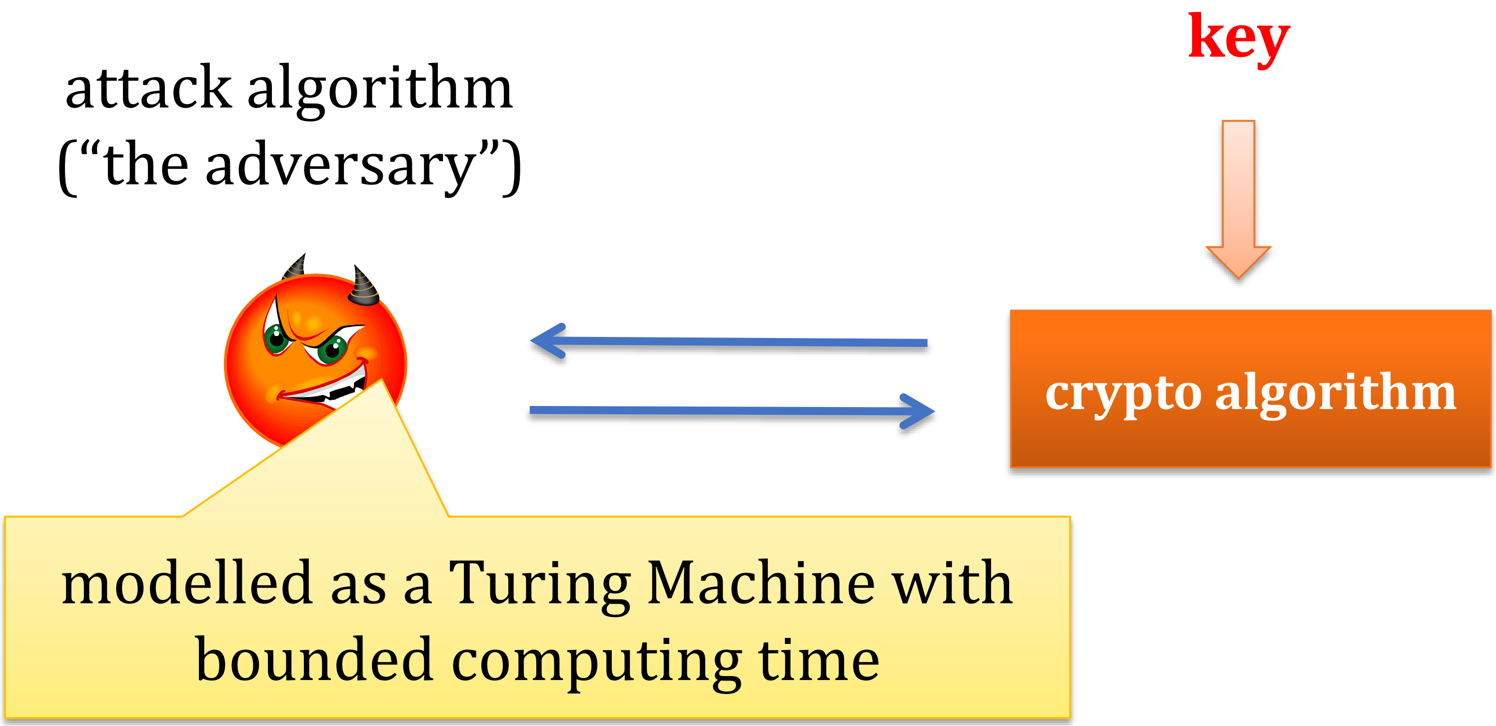
Nowadays:

the **science** of securing digital communication and transactions (encryption, authentication, digital signatures, e-cash, auctions, etc..)

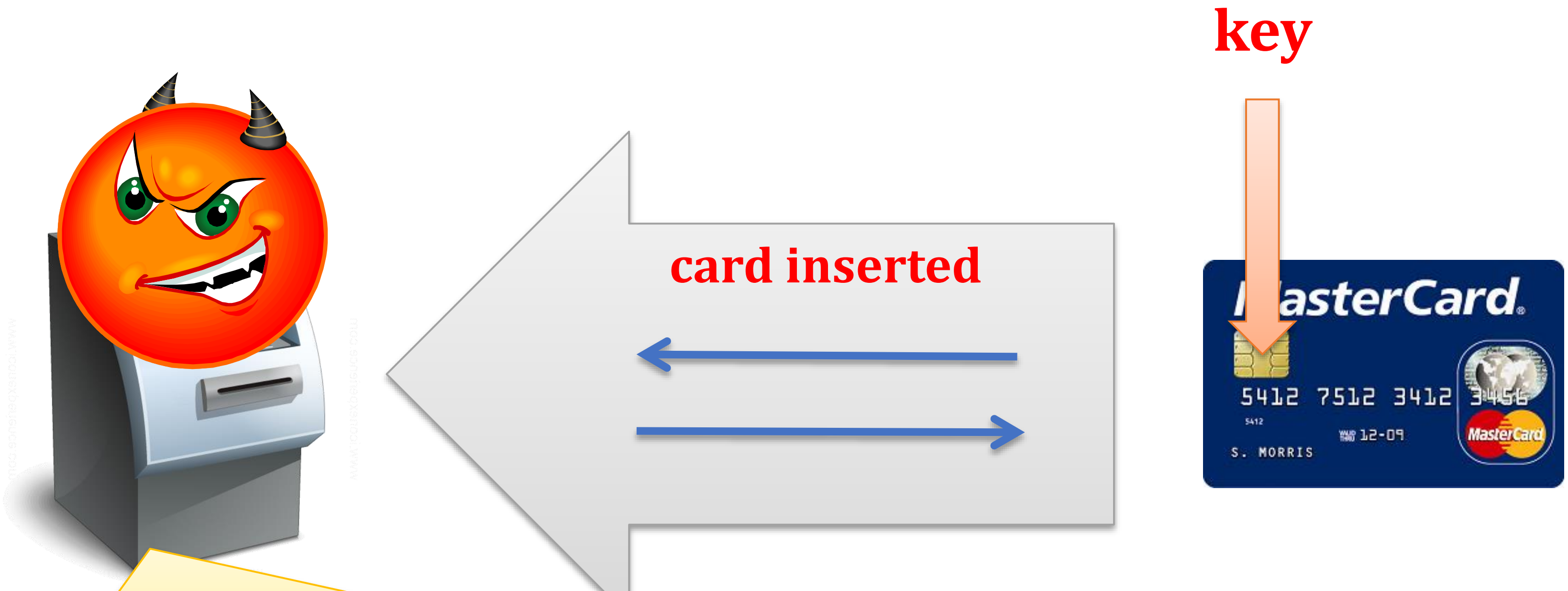
design method: “provable security”

proofs in a well-defined mathematical **model**

Standard model: black-box



Example: smart cards



even a **malicious** **ATM** should not be able to clone the card

Note: such cards are **much more secure** than cards with magnetic stripe.

“Black-Box Cryptography” – the situation

In general the problem of **constructing basic cryptographic protocols** secure in this model appears to be **solved**.

For example in symmetric encryption:

> even the “ancient” cipher **DES** (from 1970s) is broken only because its key is too short.
>
> (all other attacks are “theoretical”)

Can we relax?

Is the black-box model realistic?

No: Smart Cards can be broken by physical attacks.

The adversary obtains a temporary access to the device and can "play with it".
In particular: he can exploit its **physical properties**

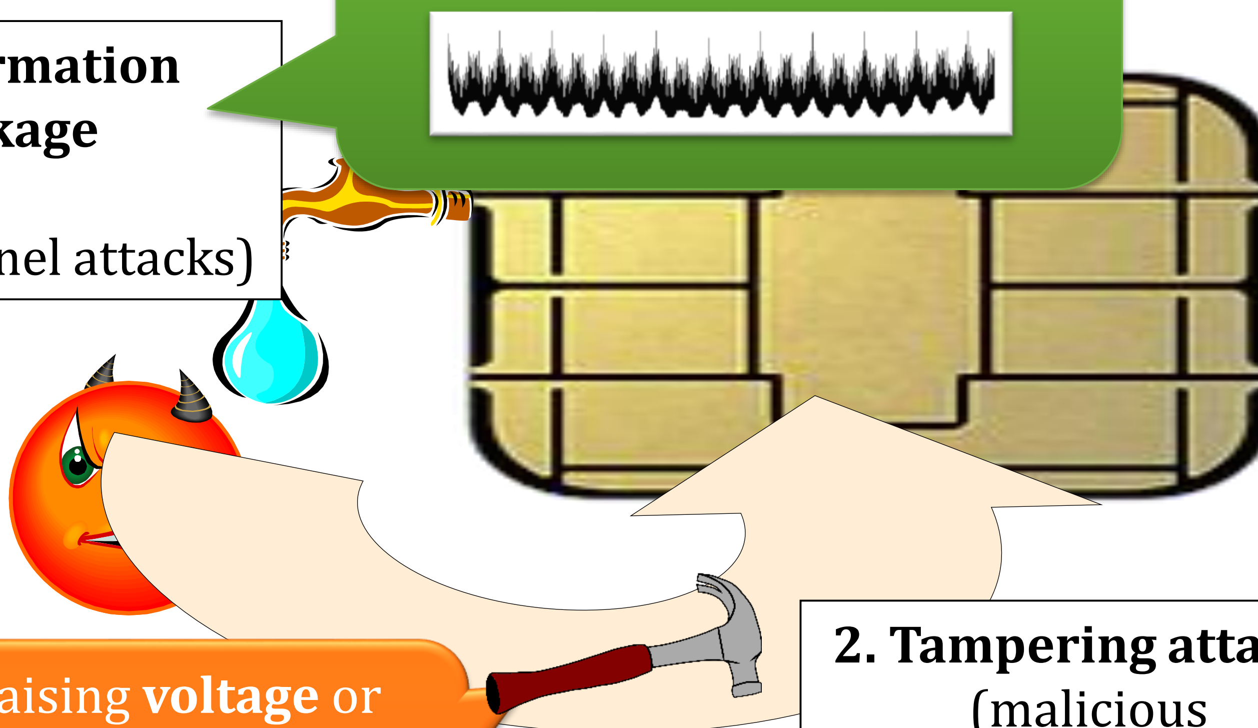
Much more powerful than the traditional "black-box" attacks!

Physical attacks on the implementation

Example: power consumption measurements

1. Information leakage

(side-channel attacks)



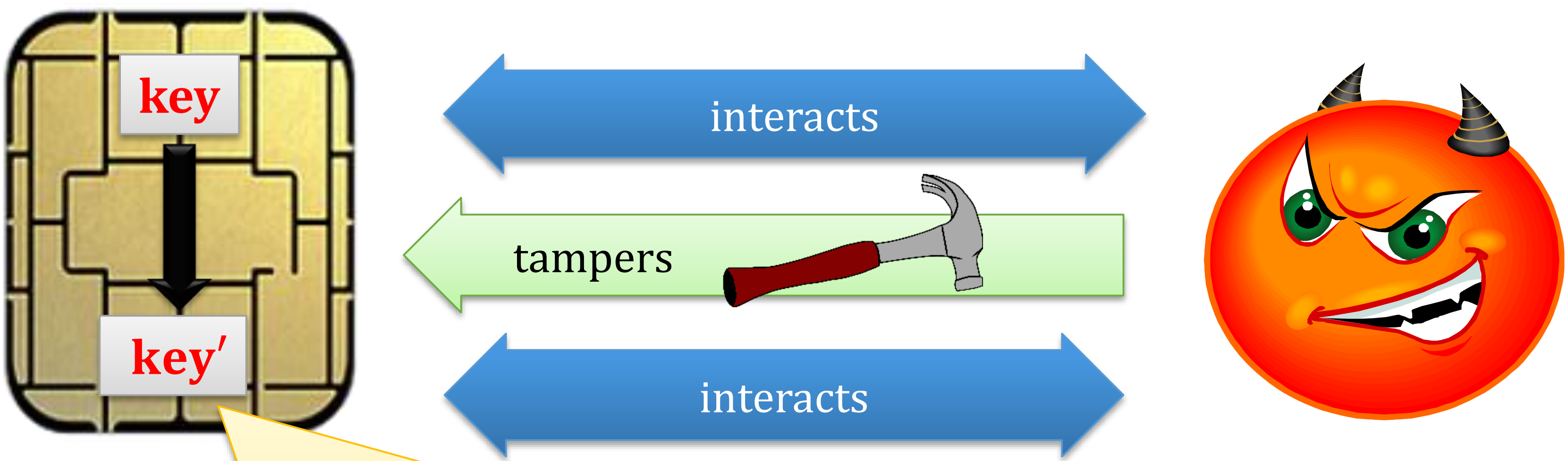
Example: raising **voltage** or **temperature**, tampering with **clock**, focusing **UV light** on the device…

2. Tampering attacks
(malicious modifications)

today we focus on this

How can the adversary exploit the tampering attacks?

Example: related key attacks

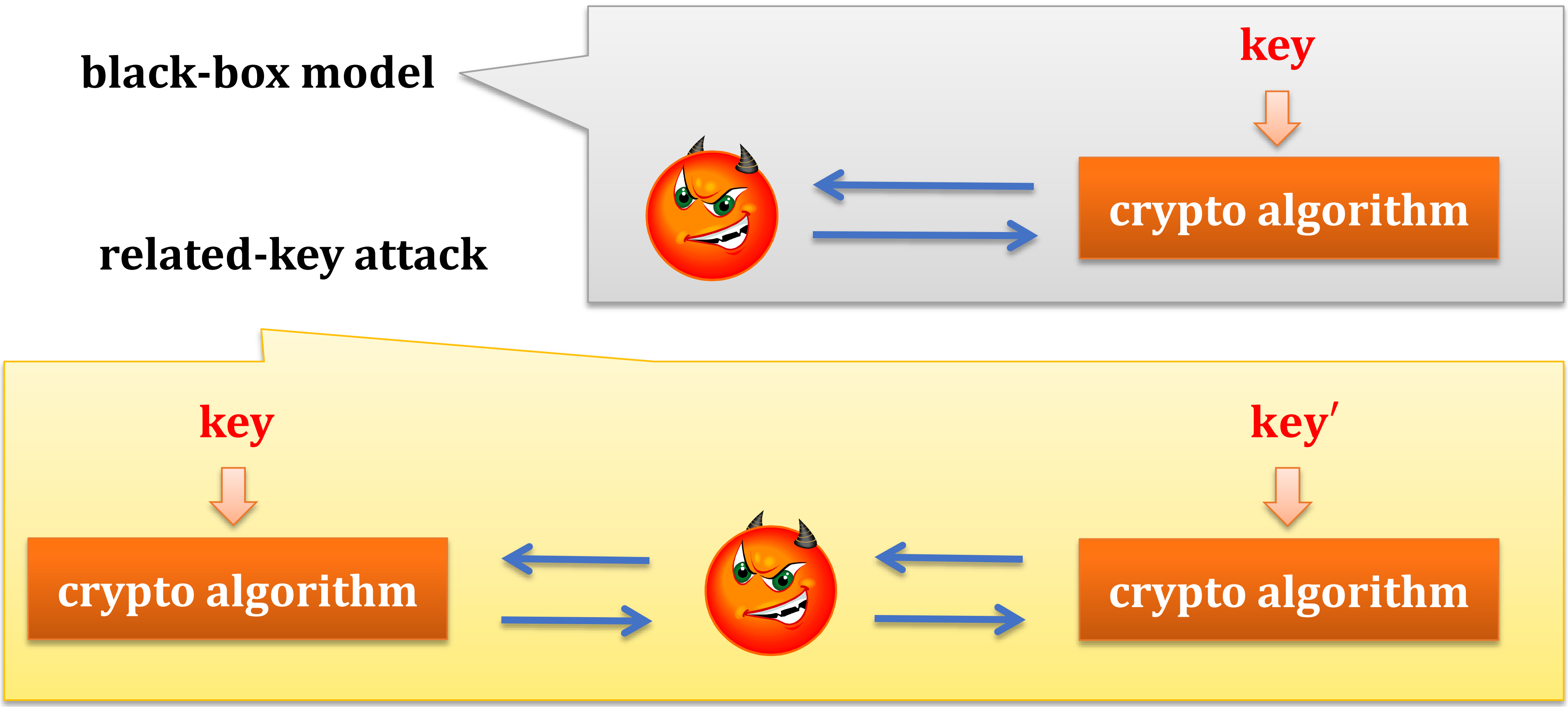


key′ is “related” to **key**

For example:

it’s equal to **key**, except that the first bit is negated.

This way the adversary obtains more power than in the black-box model!



Many successful attacks are based on this!

Countermeasures?

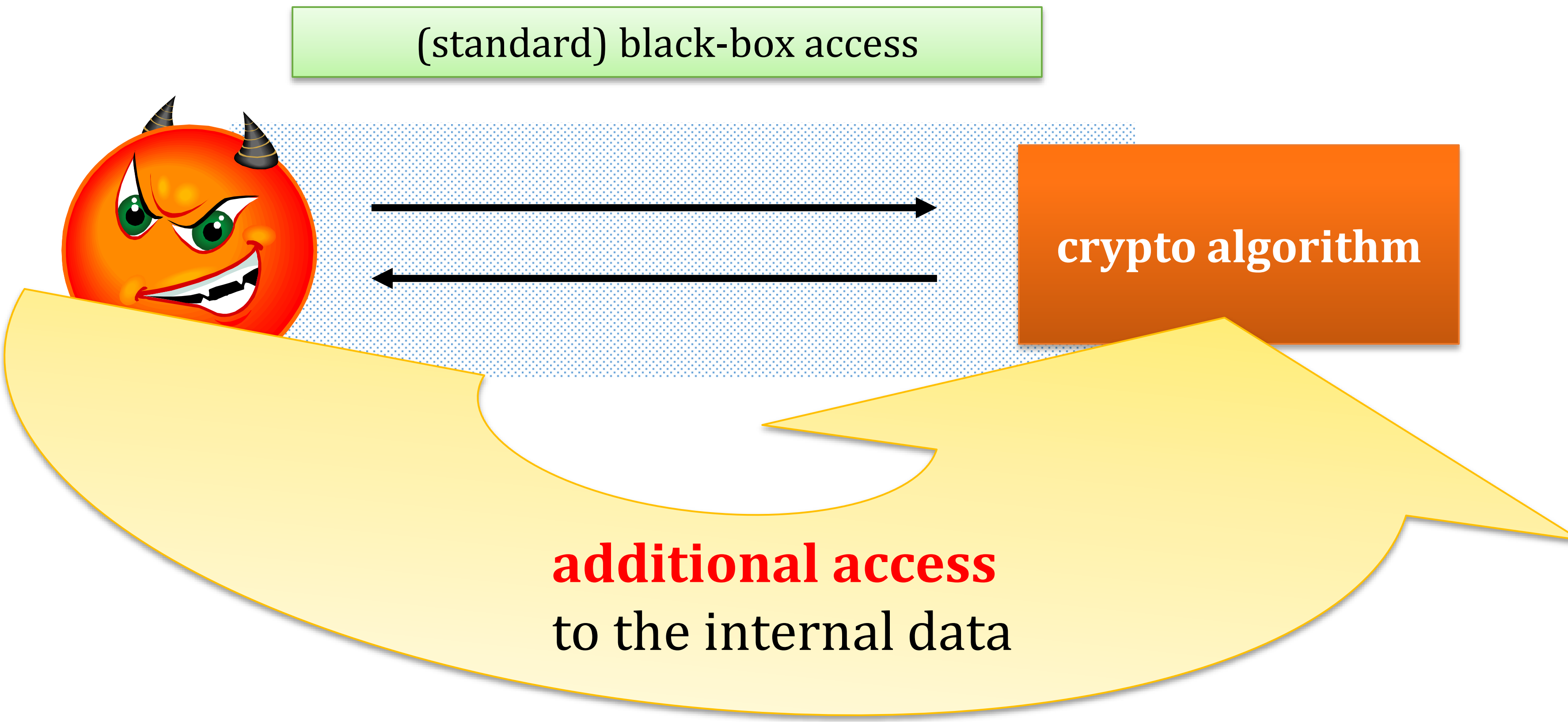
Lots of ad hoc practical solutions (a market worth billions of dollars)

cryptographic devices are everywhere: payment cards, tickets, SIM cards, pay TV cards, etc..

– usually based on a **trial-and-error methodology**…

A more formal approach?

An idea: incorporate physical attacks to the formal model

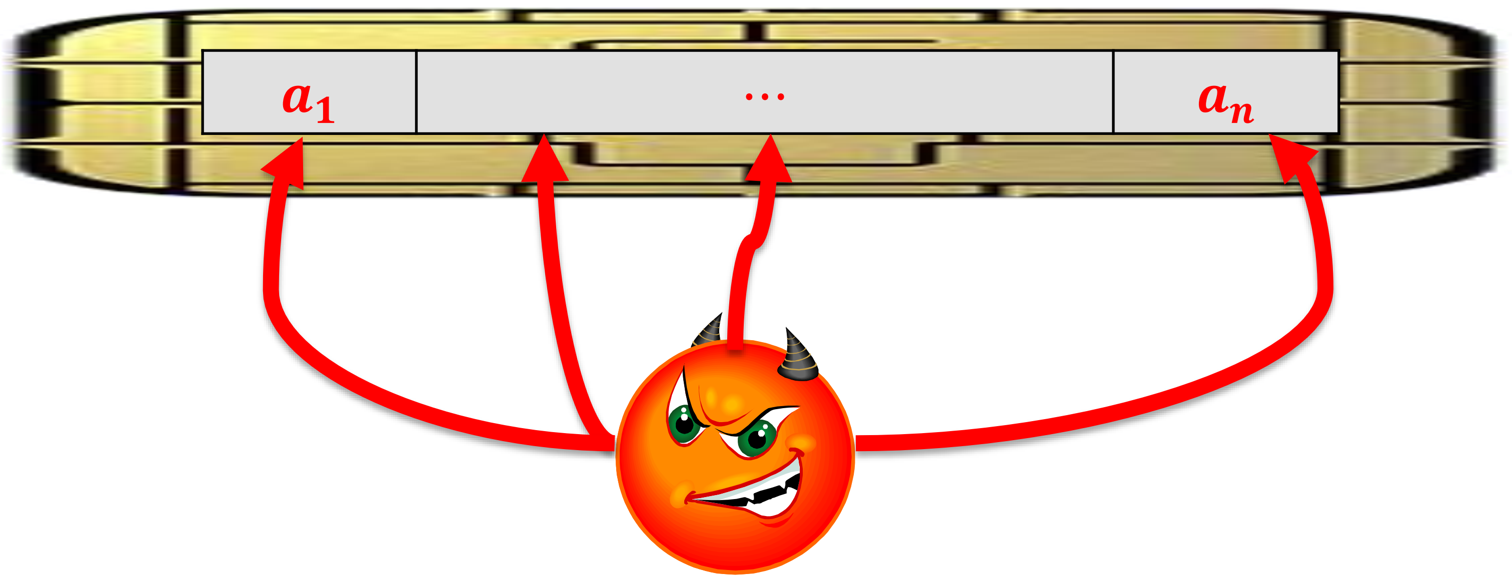


Hundreds of papers during the last decade!

Example: modelling leakage

"t-probing memory attacks"

The adversary can read-off up to t wires from the memory .



The fundamental building block

encoding schemes secure against the physical attacks.

Encoding scheme is a pair of algorithms

(Enc: $\mathcal{M} \rightarrow \mathcal{C}$, Dec : $\mathcal{C} \rightarrow \mathcal{M}$)

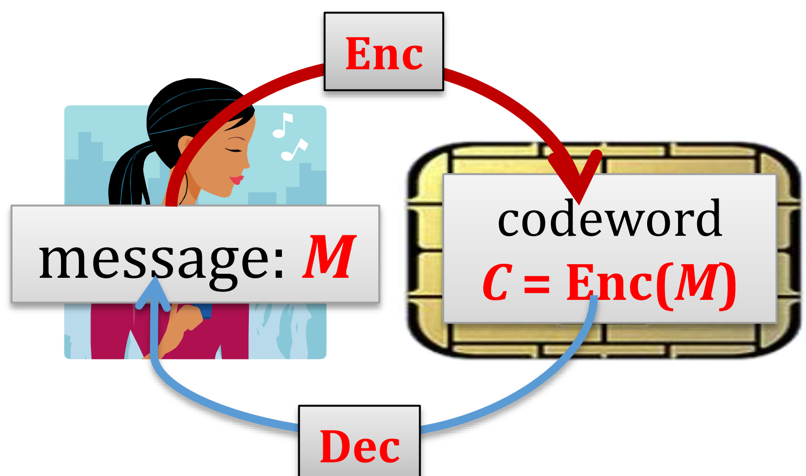
note: no secret key

$\mathcal{M}$ – set of **messages**

$\mathcal{C}$ – set of **codewords**

such that:

- **Enc** can be randomized,
- and $\forall_M$ **Dec(Enc(M)) = M**



Example

A bit encoding scheme $(\mathbf{Enc}: Z_2 \to Z_2^n, \mathbf{Dec}: Z_2^n \to Z_2)$ secure against $(n-1)$-**probing memory attacks**.

Let n be some natural parameter. To encode a bit $M \in Z_2$ take $a_1, \ldots, a_n \in Z_2$ uniformly at random such that

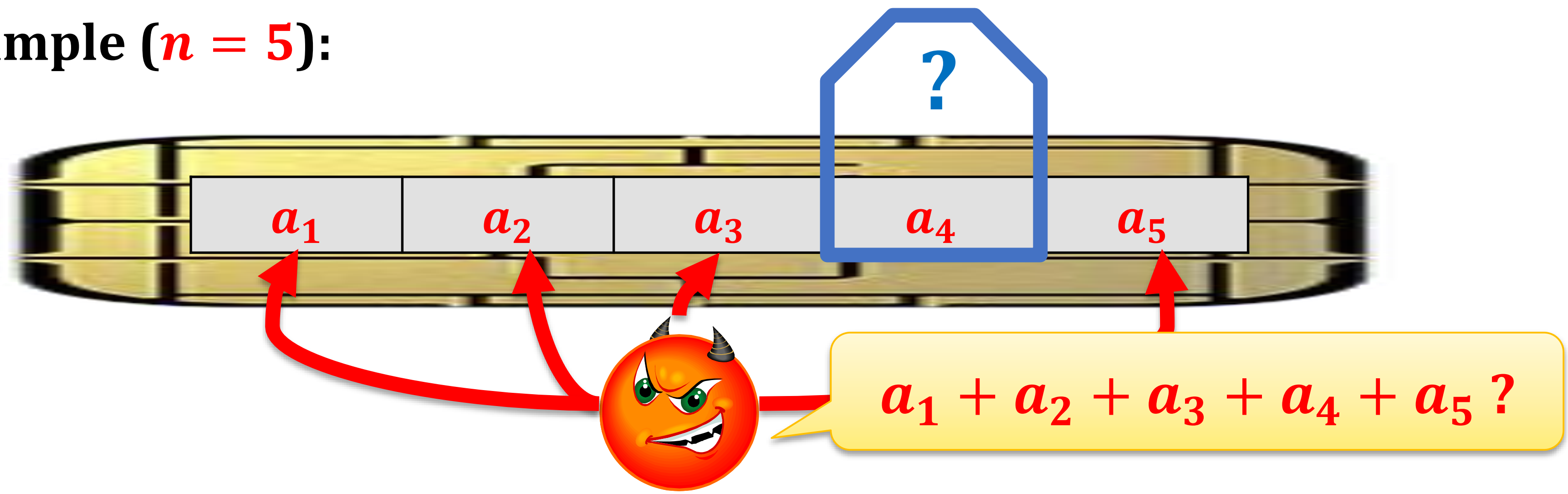
$$M = a_1 + \cdots + a_n \bmod 2$$

and let $\mathbf{Enc}(M) := (a_1, \ldots, a_n)$ and $\mathbf{Dec}(a_1, \ldots, a_n) = a_1 + \cdots + a_n$.

Now suppose that $C := \mathbf{Enc}(M)$ is stored on the device.

Then M remains secret even if the adversary learns up to $n-1$ bits of C.

Example ($n = 5$):



How to use such encoding schemes?

Note:

the encoding schemes are just a **building-block**, since they only provide security of "**memory**".

In practice we are usually interested in securing the **computation**.

This can be done by exploiting some **"homomorphic" properties of the encodings.**

For example: the encoding from the previous slide is linear.

Leakage-resilient encodings

The "leakage model" on the previous slide is **very simple** (the adversary learns "up to $\boldsymbol{n-1}$ bits").

This is not always realistic. In practice we need something stronger.

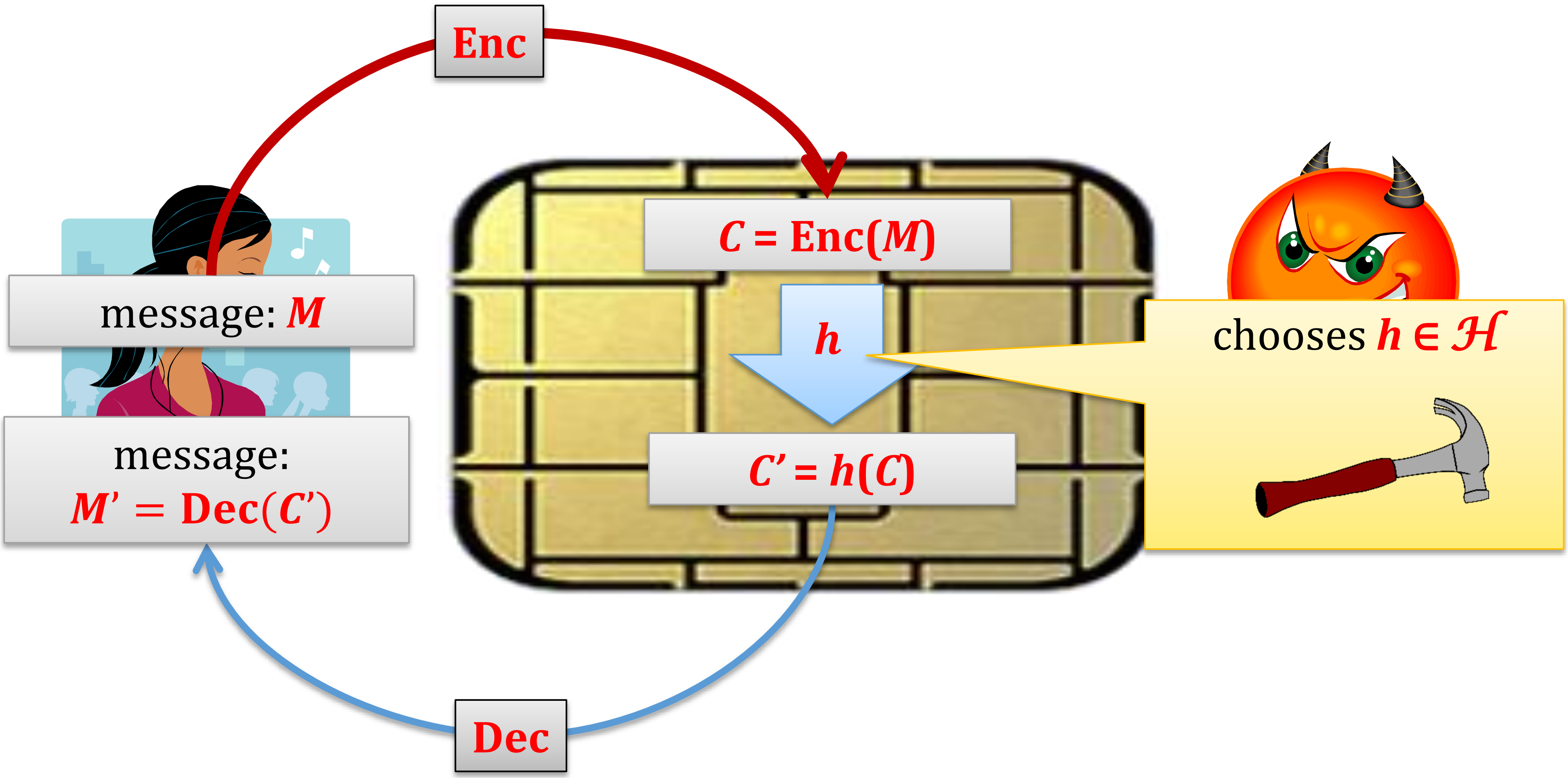
Tons of different models and constructions.

What about encoding secure against tampering?

Plan

1. Short introduction to physical attacks
2. Non-malleable codes – the definition
3. Non-malleable codes – constructions secure w.r.t different function families:
   1. bit-wise tampering
   2. tempering functions from sets of bounded size
   3. split-state model
4. Subsequent work

Tampering attacks



"Induced functions"

fix some $h \in \mathcal{H}$

Consider the tampering experiment (for some fixed M):

$$M \xrightarrow{\mathbf{Enc}} C = \mathbf{Enc}(M) \xrightarrow{h} C' = h(C) \xrightarrow{\mathbf{Dec}} M' = \mathbf{Dec}(C')$$

We say that $h: C \to C$ **induces** $h': \mathcal{M} \to \mathcal{M}$ defined for every M as:

$$h'(M) = \mathbf{Dec}\big(h\big(\mathbf{Enc}(M)\big)\big)$$

$$M \xrightarrow{\qquad h' \qquad} M' = \mathbf{Dec}(C')$$

What functions can the adversary induce?

Even for very restricted families $\mathcal{H}$ he can

- make $h'(M) = M$

by choosing $h(C) := C$

or

- make $h'(M)$ = constant X "independent from M"

by choosing $h(C) := \mathbf{Enc}(X)$

Non-Malleable Codes (NMC)

Main idea

The "identity" and the "constant" attacks should be the only thing that the adversary can do.

In other words: $\boldsymbol{M}$ should be either

- **equal** to $\boldsymbol{M}$
- or **unrelated** to it.

Informally

(Enc, Dec) is **non-malleable with respect to family** $\mathcal{H}$ if $\boldsymbol{h'}$ can be represented as a **probabilistic combination** of:

- the **identity** function
- and **constant** functions

(we formalize it a bit later)

Non-malleability in cryptography

Introduced in **[Dolev, Dwork, and Naor, STOC'91]**

Informally:

a cryptographic primitive $\boldsymbol{X}$ (with a secret key $\boldsymbol{S}$) is **malleable** if there exists an adversary who is able to produce output "related to" $\boldsymbol{X}(\boldsymbol{S})$, but not equal to it (even if he does not know $\boldsymbol{S}$).

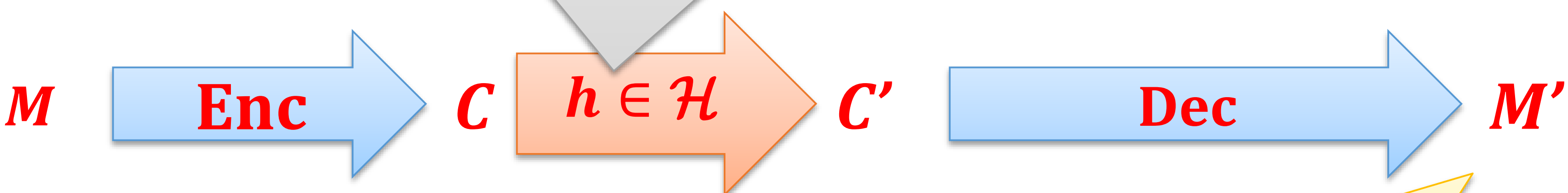
(it is **non-malleable** otherwise)

Can we have an NMC secure against the family of all functions?

no!

Attack example:

1. Decodes $\boldsymbol{M} = \mathbf{Dec}(\boldsymbol{C})$
2. Let $\boldsymbol{M}' := \boldsymbol{M}$ with all bits negated
3. Let $\boldsymbol{C}' := \mathbf{Enc}(\boldsymbol{M}')$



Clearly: $\boldsymbol{M}'$ is related to $\boldsymbol{M}$ (but $\boldsymbol{M}' \neq \boldsymbol{M}$)

Moral

$\mathcal{H}$ **has to be restricted in some way.**

Popular variants:

- **independent bit tampering** – C is a bit-string and h tampers with each bit independently,
- **split state model** – C is divided into **2** (or more) independent parts, and the adversary can tamper with each part **independently**,
- **low complexity tampering** – h has to be represented by a small circuit

How to formalize that *h'* is a probabilistic combination of **constant** functions?

$(\mathbf{Enc}\colon \mathcal{M} \to \mathcal{C}, \mathbf{Dec}\colon \mathcal{C} \to \mathcal{M})$ is **non-malleable w.r.t.** $\mathcal{H}$ if

$\forall$ $h \in \mathcal{H}$ $\exists$ D – random variable taking values from $\mathcal{M}$

such that $\forall$ $M \in \mathcal{M}$ $h'(M) \equiv D$

equality of distributions

Question: what with the "**identity**" function?

Solution

$\boldsymbol{D}$ – random variable taking values from $\mathcal{M} \cup \{\underline{\textbf{same}}\}$

For $\boldsymbol{M} \in \mathcal{M}$ and $\boldsymbol{d} \in \mathcal{M} \cup \{\underline{\textbf{same}}\}$ define:

$$\mathbf{Tamper}_{\boldsymbol{M}}(\boldsymbol{d}) := \begin{cases} \text{if } \boldsymbol{d} = \underline{\textbf{same}} \text{ then } \textbf{output } \boldsymbol{M} \\ \text{otherwise } \textbf{output } \boldsymbol{d} \end{cases}$$

$(\mathbf{Enc}, \mathbf{Dec})$ is **non-malleable w.r.t.** $\mathcal{H}$ if

$$\underset{\boldsymbol{h} \in \mathcal{H}}{\forall} \quad \underset{\boldsymbol{D}}{\exists} \quad \textbf{such that} \quad \underset{\boldsymbol{M} \in \mathcal{M}}{\forall} \quad \boldsymbol{h}'(\boldsymbol{M}) \equiv \mathbf{Tamper}_{\boldsymbol{M}}(\boldsymbol{D})$$

In practice it's useful to relax this definition a bit

$(\mathbf{Enc}, \mathbf{Dec})$ is ϵ-**non-malleable w.r.t.** $\mathcal{H}$ if

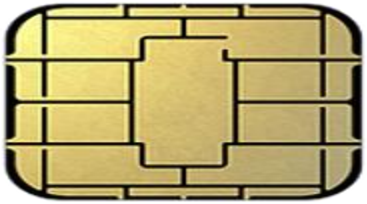
$$\underset{h \in \mathcal{H}}{\forall} \quad \underset{D}{\exists} \text{ such that } \underset{M \in \mathcal{M}}{\forall} \quad h'(M) \approx^{\epsilon} \mathbf{Tamper}_M(D)$$

ϵ-closeness of distributions
(we skip the formal definition)

One way to look at it

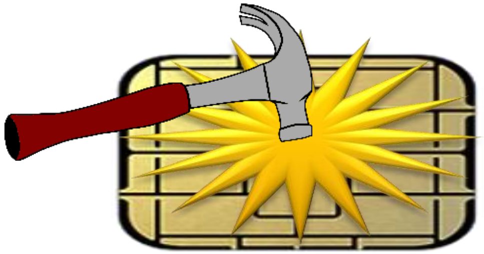
The adversary can either

- leave the device **unchanged**,

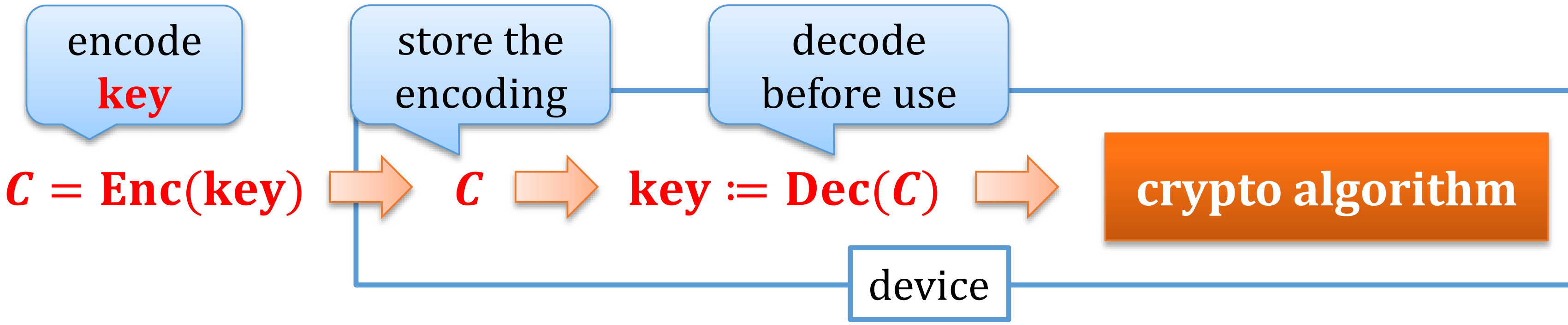


or

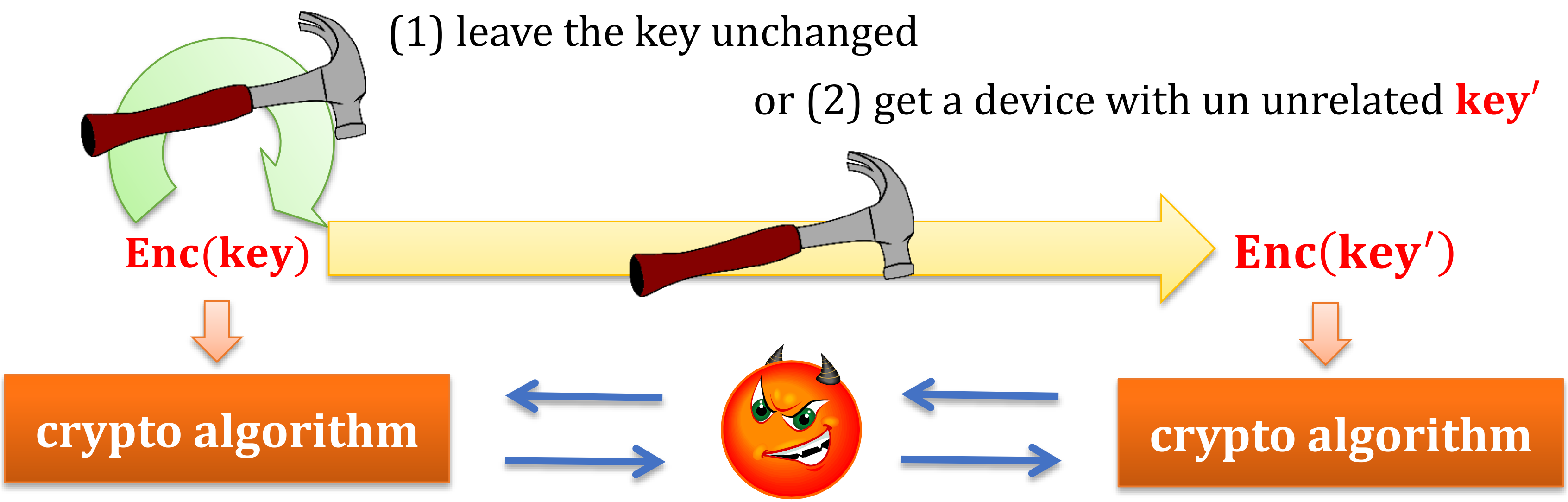
- **destroy it completely**



How to use NMCs to protect against the related key attacks?



What can the adversary do?



This gives him no more power than in the black-box model!

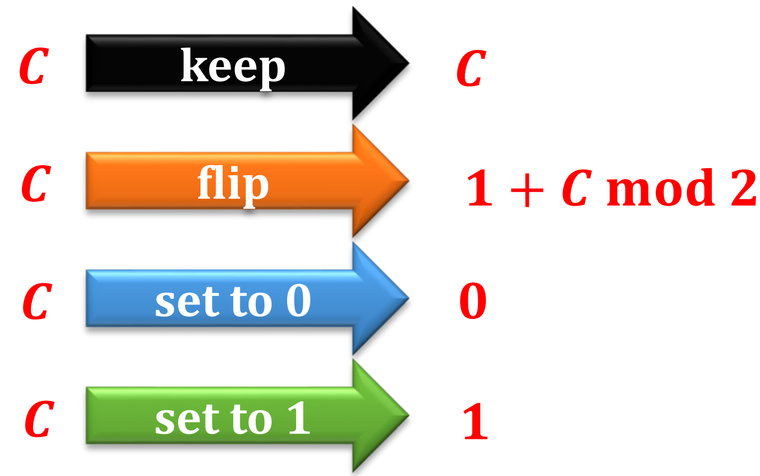
Plan

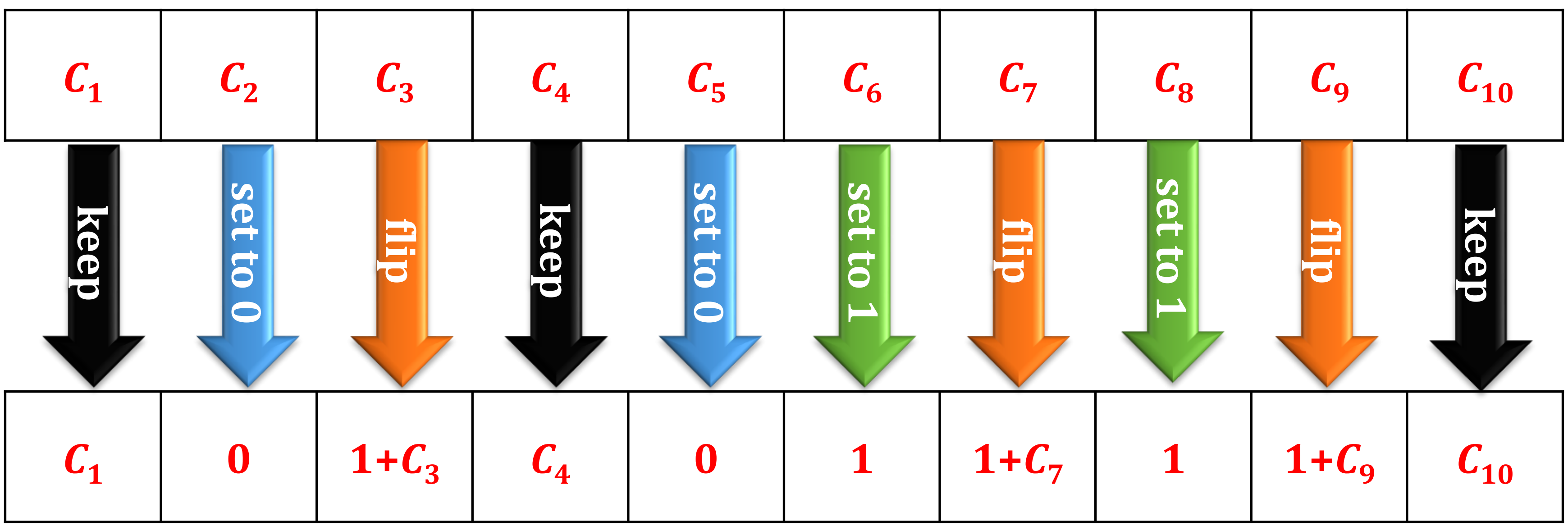
1. Short introduction to physical attacks
2. Non-malleable codes – the definition
3. Non-malleable codes – constructions secure w.r.t different function families:
   1. bit-wise tampering
   2. tempering functions from sets of bounded size
   3. split-state model
4. Subsequent work

Independent Bit Tampering

set of codewords: Z_2^n
(where n is some parameter)

4 types of functions acting on bits:





How to design an NMC secure w.r.t. such tampering?

Simple ideas don't work.

For example take the encoding that we constructed before:

To encode a bit $\boldsymbol{M} \in \boldsymbol{Z_2}$ take $\boldsymbol{a_1, \dots, a_n} \in \boldsymbol{Z_2}$ uniformly at random such that $\boldsymbol{M = a_1 + \cdots + a_n \bmod 2}$ and let

$$\mathbf{Enc}(\boldsymbol{M}) \coloneqq (\boldsymbol{a_1, \dots, a_n}) \text{ and } \mathbf{Dec}(\boldsymbol{a_1, \dots, a_n}) = \boldsymbol{a_1 + \cdots + a_n}.$$

This is clearly **malleable** w.r.t. independent bit tampering because negating one bit negates the message:

$$\mathbf{Dec}(\mathbf{1} + \boldsymbol{a_1, \dots, a_n}) = \mathbf{1} + \boldsymbol{a_1 + \cdots + a_n} = \mathbf{1} + \boldsymbol{M}.$$

Non-malleable code secure against independent bit tampering

[DPW10]:
A construction of an efficient non-malleable code secure against independent bit tampering.

It achieves the rate of **≈ 0.1887**.

(later improved in some subsequent work)

Uses the **algebraic manipulation detection** codes **[CDFPW08]**.

Plan

1. Short introduction to physical attacks
2. Non-malleable codes – the definition
3. Non-malleable codes – constructions secure w.r.t different function families:
    1. bit-wise tampering
    2. tempering functions from sets of bounded size
    3. split-state model
4. Subsequent work

The existential result

Consider codes with the set of codewords $\boldsymbol{Z_2^n}$ (for some parameter $\boldsymbol{n}$).

Theorem [DPW10]

Suppose $\mathcal{H}$ is a subset of tampering functions $\boldsymbol{Z_2^n} \rightarrow \boldsymbol{Z_2^n}$ such that

$$\mathbf{log_2(log_2(|\mathcal{H}|)) < \boldsymbol{n}.}$$

Then there exists a code that is non-malleable with respect to $\mathcal{H}$.

In particular: a random code is non-malleable with a very high probability.

Note:

The set of ALL functions $\boldsymbol{h}\colon \boldsymbol{Z_2^n} \rightarrow \boldsymbol{Z_2^n}$ is such that

$$\mathbf{log_2(log_2(|ALL|)) = \boldsymbol{n} + log_2 \boldsymbol{n}}$$

Because: $\mathbf{2}^{\boldsymbol{n}\cdot \mathbf{2}^{\boldsymbol{n}}} \xrightarrow{\mathbf{log_2}} \xrightarrow{\mathbf{log_2}} \mathbf{log_2}\boldsymbol{n} + \boldsymbol{n}$

Plan

1. Short introduction to physical attacks
2. Non-malleable codes – the definition
3. Non-malleable codes – constructions secure w.r.t different function families:
    1. bit-wise tampering
    2. tempering functions from sets of bounded size
    3. split-state model
4. Subsequent work

The “split-state model”

Suppose that

$$\mathbf{Enc}\colon \mathcal{M} \to \mathcal{L} \times \mathcal{R}$$
$$\mathbf{Dec}\colon \mathcal{L} \times \mathcal{R} \to \mathcal{M}$$

and $\mathbf{Enc}(M) = (L, R)$

(f, g) – arbitrary tampering functions.

f and g are applied separately to L and R:

Formally: $\mathcal{H} = \{(f, g)\colon f\colon \mathcal{L} \to \mathcal{L},\, g\colon \mathcal{R} \to \mathcal{R}\}$.

Split-state model – motivation

- easily **implementable** in practice

- well-studied model in the **leakage**-resilient crypto
- **generalizes** some other models (e.g. the independent bit tampering)

Consequence of the existential result

Observation

If $\mathcal{L} = \mathcal{R} = Z_2^{n/2}$ then a random code is secure against the split-state encoding.

Proof.

The set of codewords is: $\mathcal{L} \times \mathcal{R} = Z_2^{n}$ and hence the number of tampering functions is:

$$\left(2^{\frac{n}{2}\cdot 2^{n/2}}\right)^2 = 2^{n\cdot 2^{n/2}} \xrightarrow{\log_2} \xrightarrow{\log_2} \log_2 n + \frac{n}{2} < n$$

Therefore a random code is non-malleable w.r.t. such functions.

An open problem from [DPW10]

Construct an explicit and efficient non-malleable code secure in **the split-state model**.

Closed in the recent **(2012-2015)** line of work.

We will now talk more about it.

In particular, we will describe NMCs in this model that works for **one-bit messages**.

Progress towards solving this problem

1. **[DPW10]**: existential result
2. **[Liu and Lysyanskaya, Crypto 2012]**: **computational**-security, assuming **common reference string**

we will show this now

3. **[D., Kazana, Obremski, Crypto 2013]**: secure encoding for **1-bit messages**
4. **[Aggarwal, Dodis, and Lovett, STOC 2014]:** first result for messages of arbitrary length.
5. **[Chattopadhyay and Zuckerman, FOCS’14]**:**[Aggarval, Dodis, Kazana, Obremski, STOC 2015]**: improving capacity..

NMCs for **1**-bit messages?

Not directly useful.

But interesting as a building block.

Easier to analyze since in this case NMCs have a simpler (but equivalent) definition.

Fact

For any $\mathcal{H}$ any scheme $(\mathbf{Enc}: \boldsymbol{Z_2} \rightarrow \mathcal{C}, \mathbf{Dec}: \mathcal{C} \rightarrow \boldsymbol{Z_2})$ is **non-malleable w.r.t. a family** $\mathcal{H}$ if:

Recall:
$\boldsymbol{h'(M) = \mathbf{Dec}(h(\mathbf{Enc}(M)))}$

$$\forall_{h \in \mathcal{H}} \quad \boldsymbol{P(M \neq h'(M)) \leq \frac{1}{2}}$$

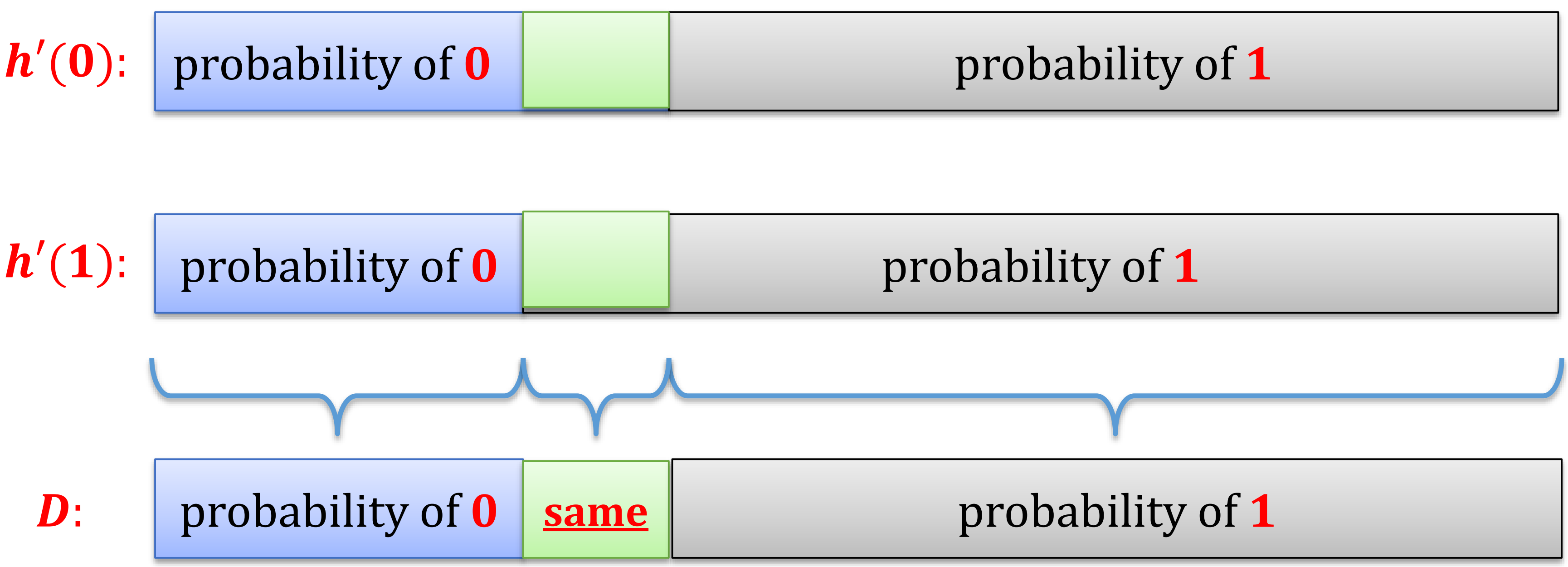
where $\boldsymbol{M}$ is uniformly distributed over $\boldsymbol{\{0, 1\}}$.

hard to negate ⇒non-malleable

$$\frac{1}{2}\cdot P(h'(1)=0)+\frac{1}{2}\cdot P(h'(0)=1)\leq\frac{1}{2}$$

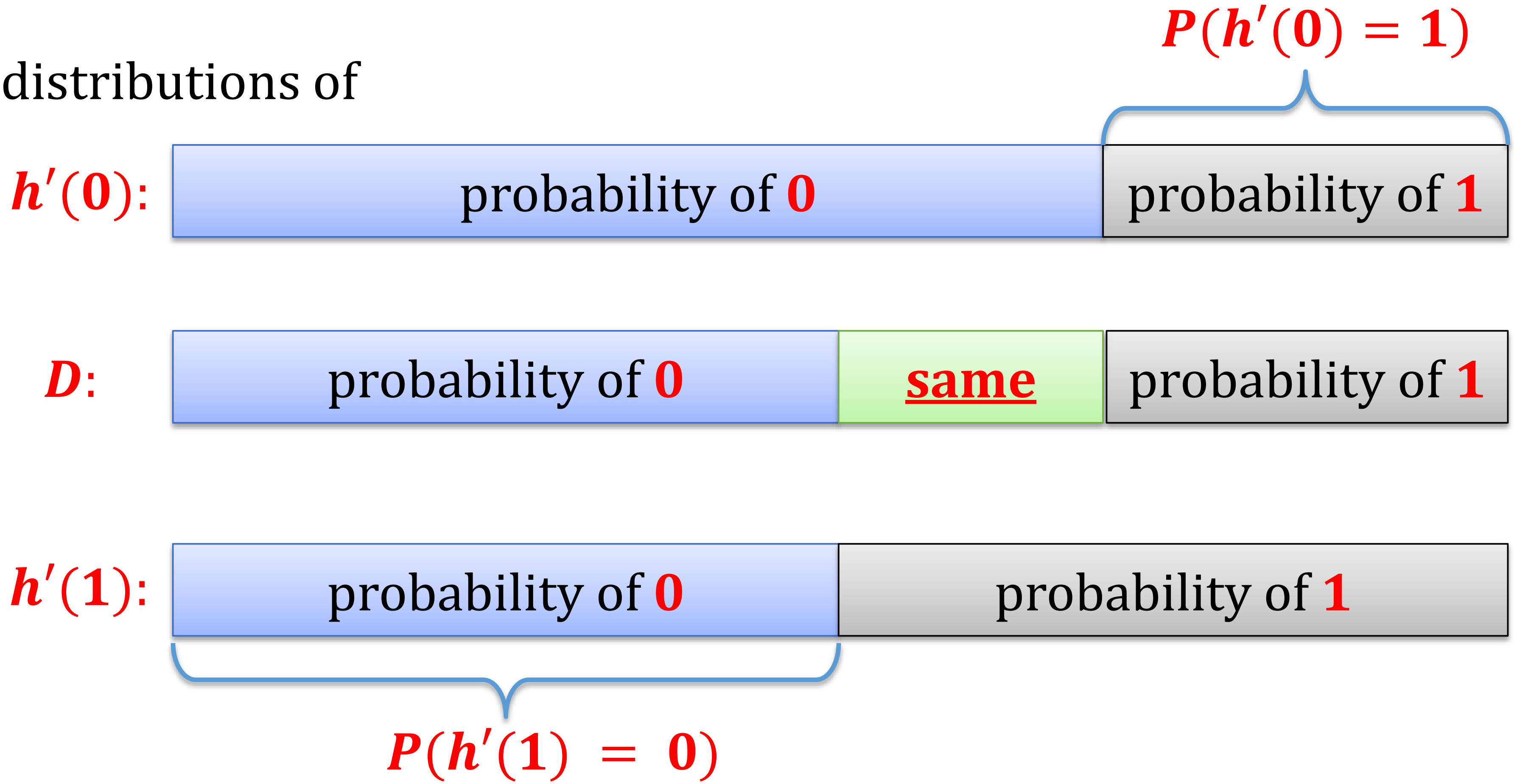
$$P(h'(1)=0)+P(h'(0)=1)\leq 1$$

look at the distributions of:



non-malleable ⇒ hard to negate

distributions of



Hence: $P(h'(1) = 0) + P(h'(0) = 1) \leq 1$

Look again at our problem:

M – uniformly random over Z_2

$\mathrm{Enc}(M)$

$L' = f(L)$

$M' := \mathrm{Dec}(L', R')$

$R' = g(R)$

Goal:

construct encoding such that for every f, g we have:

$$P(M' \neq M) \leq \frac{1}{2}$$

Our construction

Based on the “inner product function”:

F – finite field

$$\langle \underbrace{(L_1, \ldots, L_k)}_{L \in \mathbf{F}^k}, \underbrace{(R_1, \ldots, R_k)}_{R \in \mathbf{F}^k} \rangle = \sum_{i=1}^{k} L_i \times R_i$$

where $\forall_i \; L_i, R_i \in \mathbf{F}$

How to base encoding on this?

Define the following encoding for messages $\boldsymbol{M} \in \mathbf{F}$:

- $\mathbf{Enc}(\boldsymbol{M}) = (\boldsymbol{L}, \boldsymbol{R})$

- where $\boldsymbol{L}, \boldsymbol{R}$ are random vectors from $\mathbf{F}^m$ such that $\langle \boldsymbol{L}, \boldsymbol{R} \rangle = \boldsymbol{M}$

and

- $\mathbf{Dec}(\boldsymbol{L}, \boldsymbol{R}) = \langle \boldsymbol{L}, \boldsymbol{R} \rangle$

This encoding is very useful for protecting against physical attacks

Why?

Informally:

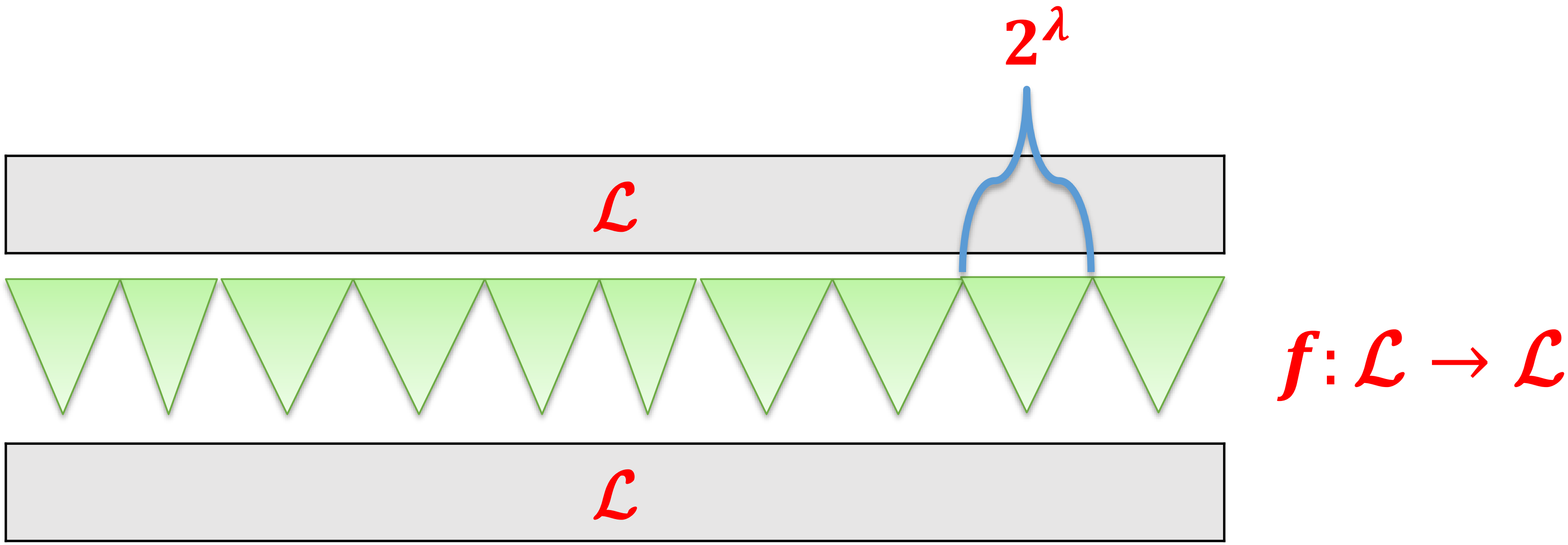
"incomplete information" about $\boldsymbol{L}$ (and complete information about $\boldsymbol{R}$) gives (almost) no information about $\langle \boldsymbol{L}, \boldsymbol{R} \rangle$.

For example: for every $\boldsymbol{L} \in \mathbf{F}^m$
$|\{L' : f(L') = f(L)\}| \geq 2^{\lambda}$ for some large λ.

In particular if one applies a function $\boldsymbol{f}$ to $\boldsymbol{L}$ that "glues" many elements together then $(\boldsymbol{f}(\boldsymbol{L}), \boldsymbol{R})$ gives almost no information about $\langle \boldsymbol{L}, \boldsymbol{R} \rangle$.

Of course: it's symmetric for $\boldsymbol{R}$.

"Gluing 2^λ elements together"



Example: a function that "forgets" first λ bits of input

$$f(a_1, \dots, a_n) = (0, \dots, 0, a_{\lambda+1}, \dots, a_n)$$

“incomplete information” about $\boldsymbol{L}$ (and complete information about $\boldsymbol{R}$) gives (almost) no information about $\langle \boldsymbol{L}, \boldsymbol{R} \rangle$.

Some intuition why this is true

Suppose $\mathbf{F} = \boldsymbol{Z_2}$. Then

$$\boldsymbol{M} = \langle (\boldsymbol{L_1}, \dots, \boldsymbol{L_k}), (\boldsymbol{R_1}, \dots, \boldsymbol{R_k}) \rangle =$$

parity of the set $\{\boldsymbol{i}: \boldsymbol{L_i} = \boldsymbol{R_i} = \mathbf{1}\}$

Intuitively:

If one learns only partial information about $(\boldsymbol{L_1}, \dots, \boldsymbol{L_k})$ then $\boldsymbol{M}$ is hidden (the same for $(\boldsymbol{R_1}, \dots, \boldsymbol{R_k})$).

Why it looks useful?

If the adversary uses a function $\boldsymbol{f}$ or $\boldsymbol{g}$ that is "gluing" many inputs then for sure $\mathbf{Dec}\big(\boldsymbol{f}(\boldsymbol{L}),\boldsymbol{g}(\boldsymbol{R})\big)$ is independent from $\mathbf{Dec}(\boldsymbol{L},\boldsymbol{R})$.

Moral: the adversary has to choose functions that do not glue too many inputs.

In other words: they have to be **close to being bijections**.

Hope: maybe this is easier to analyze?

Is this encoding non-malleable?

$\mathbf{F}$ – finite field

- $\mathbf{Enc}(M) :=$ **random** (L, R) **such that** $\langle L, R \rangle = M$
- $\mathbf{Dec}(L, R) = \langle L, R \rangle$

Problem: linearity of the inner product (let $c \in \mathbf{F}$)

$$\langle c \cdot L, R \rangle = c \cdot \langle L, R \rangle$$

So: if we choose

$$f(L) = c \cdot L \text{ and } g(R) = R$$

then $M' = c \cdot M$

Observation

If $\mathbf{F} = \boldsymbol{Z_2}$ then then $\boldsymbol{c}$ can only be $\mathbf{0}$ or $\mathbf{1}$

- if $\boldsymbol{c} = \mathbf{0}$ then it is a "**constant attack**":

$$\boldsymbol{M'} = \mathbf{0} \text{ for every } \boldsymbol{M}$$

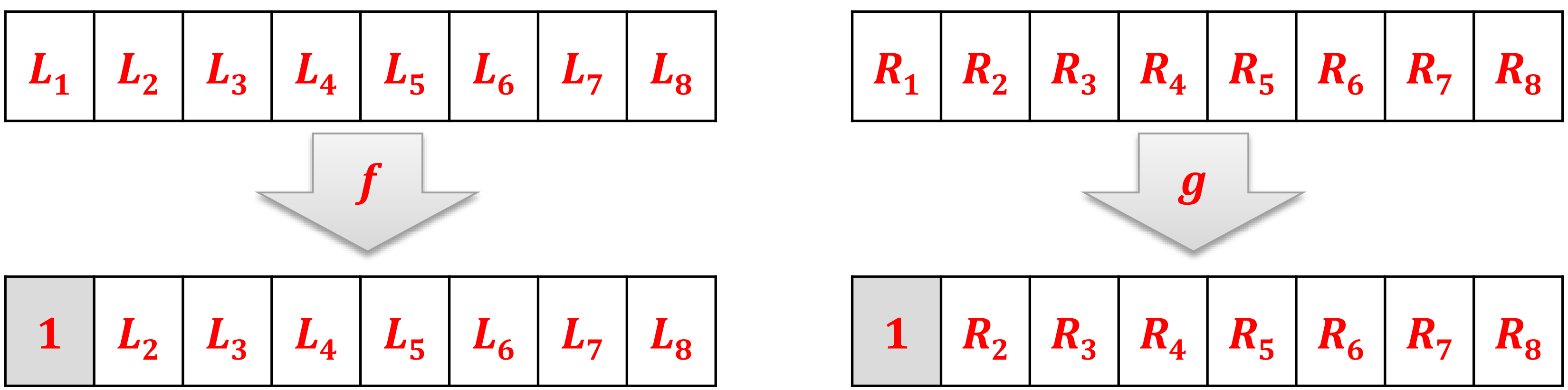
- if $\boldsymbol{c} = \mathbf{1}$ then it is an "**identity attack**"

$$\boldsymbol{M'} = \boldsymbol{M} \text{ for every } \boldsymbol{M}$$

Hope: maybe it works over $\mathbf{Z}_2$?

Unfortunately in this case another attack is possible:

"the tampering functions set $L_1 := 1$ and $R_1 := 1$"



Note: the inner product changes iff $L_1 R_1 = 0$.
This happens with probability $3/4$.

Observation

The attack from the previous slide does not work if **$|\mathbf{F}|$** is exponential.

This is because

$$\boldsymbol{P(L_1 \cdot R_1 = 0) \approx \frac{2}{|F|}}$$

So, this is the situation:

	large **F**	$\mathbf{F} = \boldsymbol{Z_2}$
the "linear attack"	**works**	**doesn't work**
the "$\boldsymbol{L_1 := 1}$ and $\boldsymbol{R_1 = 1}$" attack	**doesn't work**	**works**

Question: is it possible to combine these two solutions so that none of these attacks works?

Answer: **yes**! (for messages of length **1**)

Let $\mathbf{F}$ be a field of exponential size.

Define

$$(\mathbf{Enc}\colon \boldsymbol{Z}_2 \to \mathbf{F}^n \times \mathbf{F}^n,\ \mathbf{Dec}\colon \mathbf{F}^n \times \mathbf{F}^n \to \boldsymbol{Z}_2)$$

as

$$\mathbf{Enc}(\boldsymbol{M}) := \begin{cases} \text{random } (\boldsymbol{L}, \boldsymbol{R}) \text{ such that } \langle \boldsymbol{L}, \boldsymbol{R} \rangle = \mathbf{0} \text{ if } \boldsymbol{M} = \mathbf{0} \\ \text{random } (\boldsymbol{L}, \boldsymbol{R}) \text{ such that } \langle \boldsymbol{L}, \boldsymbol{R} \rangle \neq \mathbf{0} \text{ if } \boldsymbol{M} = \mathbf{1} \end{cases}$$

$\mathbf{Dec}(\boldsymbol{L}, \boldsymbol{R})$ just computes $\langle \boldsymbol{L}, \boldsymbol{R} \rangle$ and checks if it is $\mathbf{0}$.

For security proof – see the paper.

Encoding for messages of arbitrary length [Aggarwal, Dodis, and Lovett, STOC 2014]

General outline of their method:

1. show that mauling the inner product encoding can induce only **affine functions** $\boldsymbol{h'}$ (or their random combinations)
2. on top of it use encoding that is **resilient to affine mauling**.

A drawback of their construction:

$$|\boldsymbol{C}| = \boldsymbol{O}(|\boldsymbol{M}|^7)$$

Why affine? Look at this:

Let $\boldsymbol{M} = \langle \boldsymbol{L}, \boldsymbol{R} \rangle, \boldsymbol{M}' = \langle \boldsymbol{f}(\boldsymbol{L}), \boldsymbol{g}(\boldsymbol{R}) \rangle$

How can $\boldsymbol{M}'$ depend on $\boldsymbol{M}$?

- if $\boldsymbol{f}(\boldsymbol{L}) = \boldsymbol{a} \cdot \boldsymbol{L}$ (for $\boldsymbol{b} \in \mathbf{F}$) then
$$\boldsymbol{M}' = \boldsymbol{a} \cdot \boldsymbol{M}$$
- the adversary can also make $\boldsymbol{M}$ equal to any constant $\boldsymbol{b}$ chosen by him.

these are affine functions
$$\boldsymbol{f}(\boldsymbol{M}) = \boldsymbol{a} \cdot \boldsymbol{M} + \boldsymbol{b}$$

Main observation of **[ADL14]**

The affine functions are **the only ones** that the adversary can induce!

> They show it using the techniques from additive combinatorics (***Quasi-polynomial Freiman-Ruzsa Theorem***)

Plan

1. Short introduction to physical attacks
2. Non-malleable codes – the definition
3. Non-malleable codes – constructions secure w.r.t different function families:
    1. bit-wise tampering
    2. tempering functions from sets of bounded size
    3. split-state model
4. Subsequent work

Non-Malleable Codes – subsequent work

A very active area of research!

- constructions secure w.r.t. different function families,
- efficiency improvements
- extensions (continual tampering, updatable codes, local decodability…)
- new applications.

Conference papers from 2014–2016 with "non-malleable codes" in the title

- A. Kiayias, F. L., Y. Tselekounis: **Practical Non-Malleable Codes from l-more Extractable Hash Functions**. **ACM CCS 2016**
- M. Ball, D. Dachman-Soled, M. Kulkarni, T. Malkin: **Non-malleable Codes for Bounded Depth, Bounded Fan-In Circuits**. **EUROCRYPT 2016**
- N. Chandran, V. Goyal, P. Mukherjee, O. Pandey, J. Upadhyay: **Block-Wise Non-Malleable Codes**. **ICALP 2016**
- D. Aggarwal, J. Briët: **Revisiting the Sanders-Bogolyubov-Ruzsa theorem in Fpn and its application to non-malleable codes**. **ISIT 2016**
- E. Chattopadhyay, V. Goyal, X. Li: **Non-malleable extractors and codes, with their many tampered extensions**. **STOC 2016**
- N. Chandran, B. Kanukurthi, S. Raghuraman: **Information-Theoretic Local Non-malleable Codes and Their Applications**. **TCC 2016**
- D. Aggarwal, S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, M. Prabhakaran: **Optimal Computational Split-state Non-malleable Codes**. **TCC 2016**
- E. Chattopadhyay, V. Goyal, X. Li: **Non-malleable extractors and codes, with their many tampered extensions**. **STOC 2016**
- S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, M. Prabhakaran: **Explicit Non-malleable Codes Against Bit-Wise Tampering and Permutations**. **CRYPTO 2015**
- S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, M. Prabhakaran: **A Rate-Optimizing Compiler for Non-malleable Codes Against Bit-Wise Tampering and Permutations.** **TCC 2015**
- D. Aggarwal, S. Dziembowski, T. Kazana, M. Obremski: **Leakage-Resilient Non-malleable Codes**. **TCC 2015**
- D. Dachman-Soled, F. L., E. Shi, H.-S. Zhou: **Locally Decodable and Updatable Non-malleable Codes and Their Applications**. **TCC 2015**
- Z. Jafargholi, D. Wichs: **Tamper Detection and Continuous Non-malleable Codes**. **TCC 2015**
- S. Coretti, U. Maurer, B. Tackmann, D. Venturi: **From Single-Bit to Multi-bit Public-Key Encryption via Non-malleable Codes**. **TCC 2015**
- S. Faust, P. Mukherjee, D. Venturi, D. Wichs: **Efficient Non-malleable Codes and Key-Derivation for Poly-size Tampering Circuits**. **EUROCRYPT 2014**
- E. Chattopadhyay, D. Zuckerman: **Non-malleable Codes against Constant Split-State Tampering.** **FOCS 2014**
- M. Cheraghchi, V. Guruswami: **Non-malleable Coding against Bit-Wise and Split-State Tampering**. **TCC 2014**
- M. Cheraghchi, V. Guruswami: **Capacity of non-malleable codes**. **ITCS 2014**
- D. Aggarwal, Y. Dodis, S. Lovett: **Non-malleable codes from additive combinatorics**. **STOC 2014**
- S. Faust, P. Mukherjee, J. Buus Nielsen, D. Venturi: **Continuous Non-malleable Codes**. **TCC 2014**

Thank you!

©2017 by Stefan Dziembowski. Permission to make digital or hard copies of part or all of this material is currently granted without fee *provided that copies are made only for personal or classroom use, are not distributed for profit or commercial advantage, and that new copies bear this notice and the full citation*.