

Is Theoretical Cryptography Any Good in Practice?

David Naccache

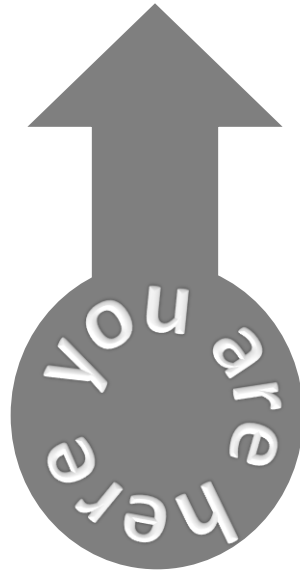
(invited talk to CRYPTO'10 and CHES'10)

Where can we go from here?



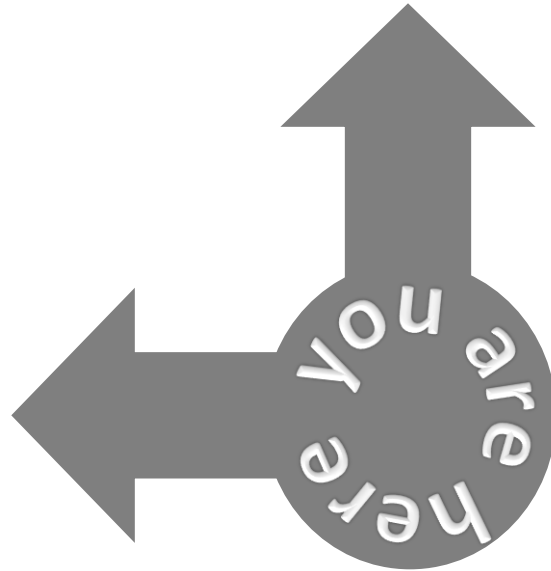


**Export theoretical approaches to
non-cryptographic security problems.**



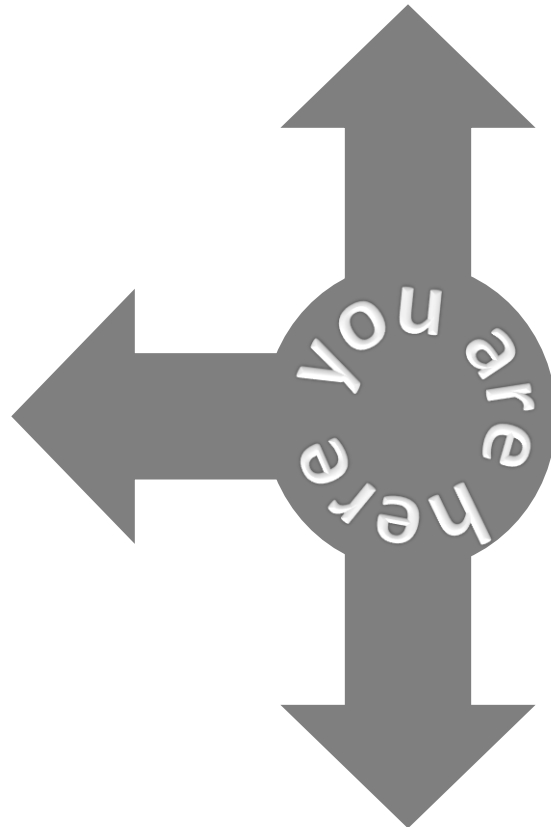
**Export theoretical approaches to
non-cryptographic security problems.**

**Create theory to
paracryptographic
problems**



**Export theoretical approaches to
non-cryptographic security problems.**

**Create theory to
paracryptographic
problems**



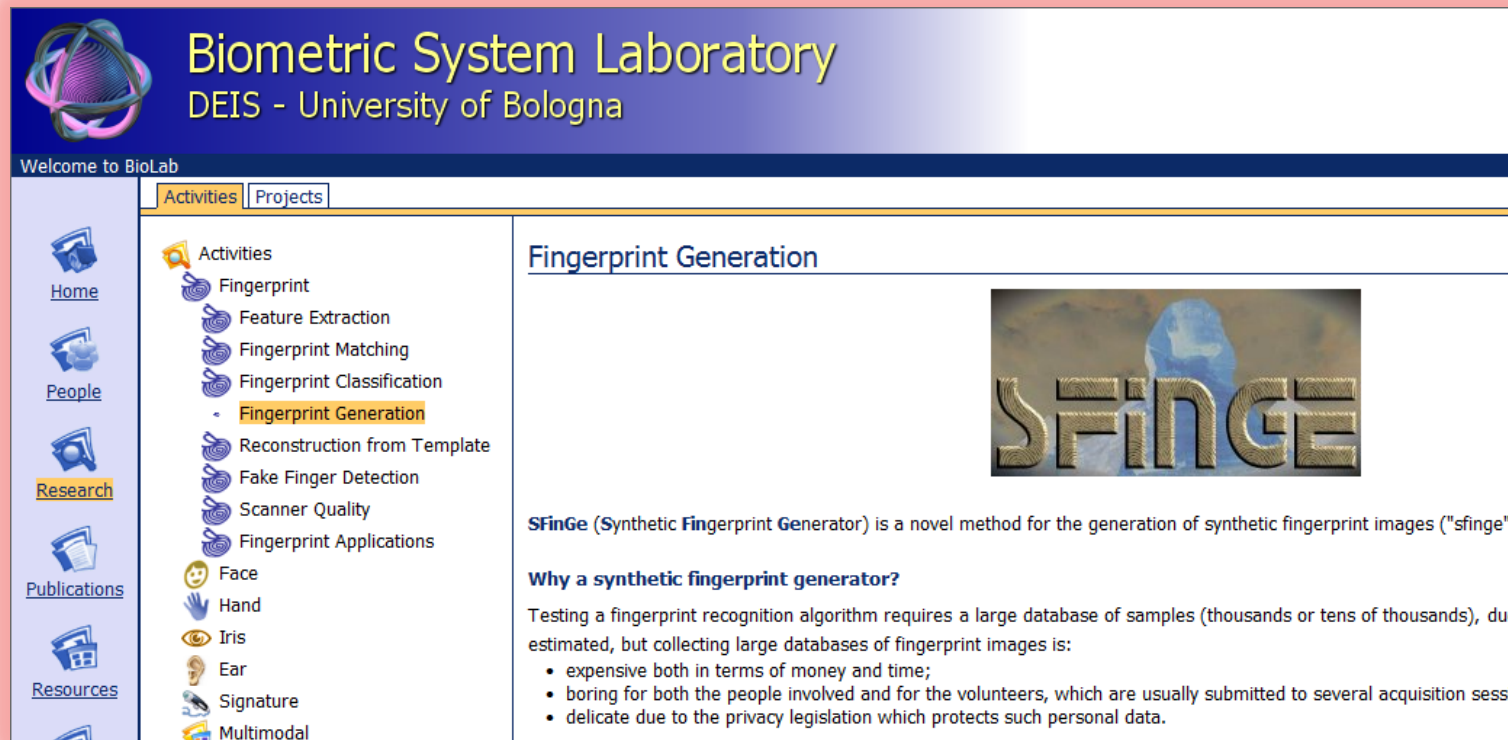
**Find theoretical solutions
to new real-life crypto
problems.**

Export

concepts

Export **Undistinguishability?**

Biometric practitioners currently test fingerprint recognition algorithms using *synthetic fingerprint generators*.



The screenshot shows the website for the Biometric System Laboratory at DEIS - University of Bologna. The page is titled "Fingerprint Generation" and features a navigation menu on the left with categories like Home, People, Research, Publications, and Resources. The main content area includes a sub-menu for "Fingerprint Generation" and a section titled "SFinGe (Synthetic Fingerprint Generator)" which describes the method and lists reasons for using a synthetic generator, such as cost, time, and privacy concerns.

Biometric System Laboratory
DEIS - University of Bologna

Welcome to BioLab


Activities | Projects

Home
People
Research
Publications
Resources

Activities

- Fingerprint
 - Feature Extraction
 - Fingerprint Matching
 - Fingerprint Classification
 - Fingerprint Generation**
 - Reconstruction from Template
 - Fake Finger Detection
 - Scanner Quality
 - Fingerprint Applications
- Face
- Hand
- Iris
- Ear
- Signature
- Multimodal

Fingerprint Generation



SFinGe (Synthetic **F**ingerprint **G**enerator) is a novel method for the generation of synthetic fingerprint images ("sfinge")

Why a synthetic fingerprint generator?

Testing a fingerprint recognition algorithm requires a large database of samples (thousands or tens of thousands), but estimated, but collecting large databases of fingerprint images is:

- expensive both in terms of money and time;
- boring for both the people involved and for the volunteers, which are usually submitted to several acquisition sessions;
- delicate due to the privacy legislation which protects such personal data.

What is a synthetic generator?

An algorithm G with two push buttons.



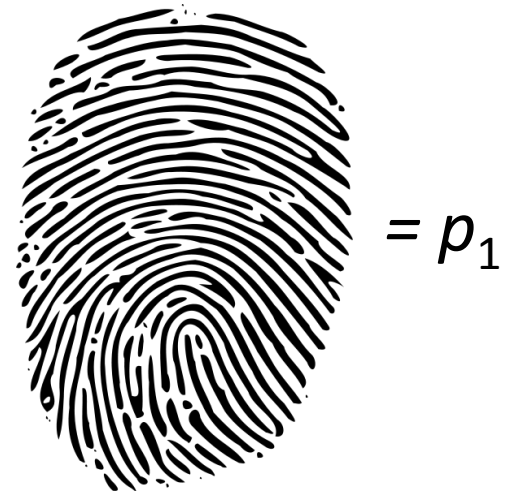
What is a synthetic generator?

When the right button is pressed G creates a new “virtual finger” model.



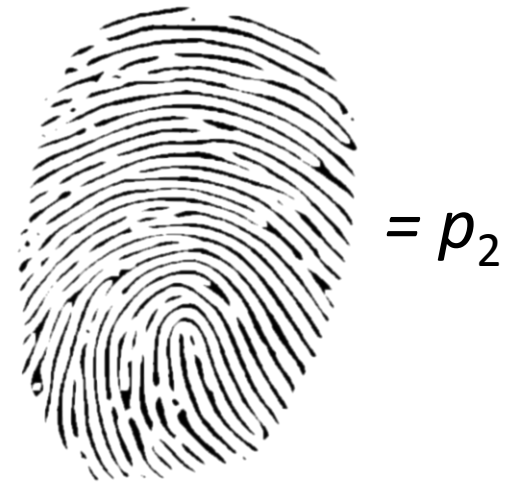
What is a synthetic generator?

The left button makes G output successive “fingerprints” p_i of the current virtual finger.



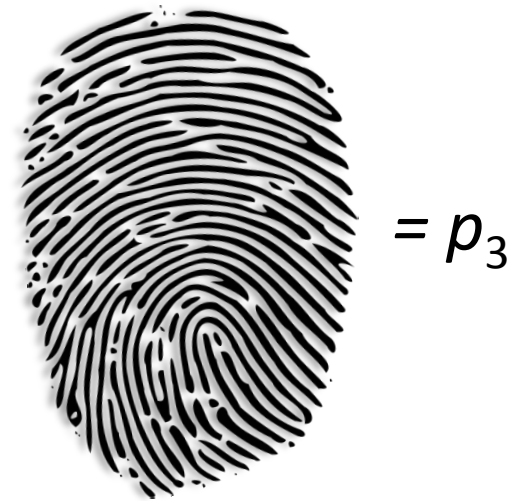
What is a synthetic generator?

The left button makes G output successive “fingerprints” p_i of the current virtual finger.



What is a synthetic generator?

The left button makes G output successive “fingerprints” p_i of the current virtual finger.



How realistic are these generators?

Currently there is no *theoretically sound* approach to fingerprint synthesis.

e.g. SFINGE uses Gabor-like space-variant filters, enhanced with several hand-tuned transforms.

But after all, *what sort of “sound theory” can one expect to apply to human biology?*

Can we export our methodologies?

Definition: A fingerprint database D is a family $F_{ID,i}$ of fingerprints (images) parameterized by an identity ID and an acquisition number i .

$$\text{FAR} = \Pr[1 \leftarrow R(F_{ID,i}, F_{ID',i'}) \mid F_{ID,i}, F_{ID',i'} \leftarrow D, ID \neq ID']$$

$$\text{FRR} = \Pr[0 \leftarrow R(F_{ID,i}, F_{ID',i'}) \mid F_{ID,i}, F_{ID',i'} \leftarrow D, ID = ID']$$

(Here R denotes a fingerprint recognition algorithm).

Definition: A synthetic generator G is $\{t, q, \varepsilon\}$ -indistinguishable from a database D if no algorithm A running in time $\leq t$ can distinguish G from D with an advantage better than ε in q queries, when IDs in D are randomly permuted.

Can we export our methodologies?

We can now have:

1. “Biographers” who design generators.

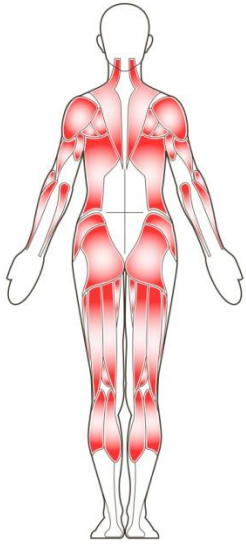


2. “Bioanalysts” who try to break generators by distinguishing them from human databases.



The game: design the mathematically *simplest possible* undistinguishable G .

In other words

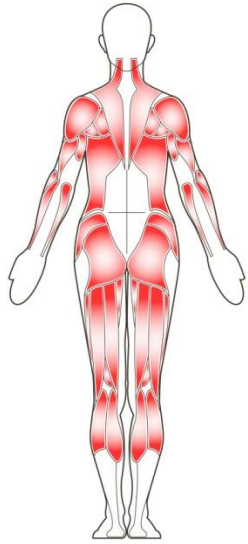


biology



maths

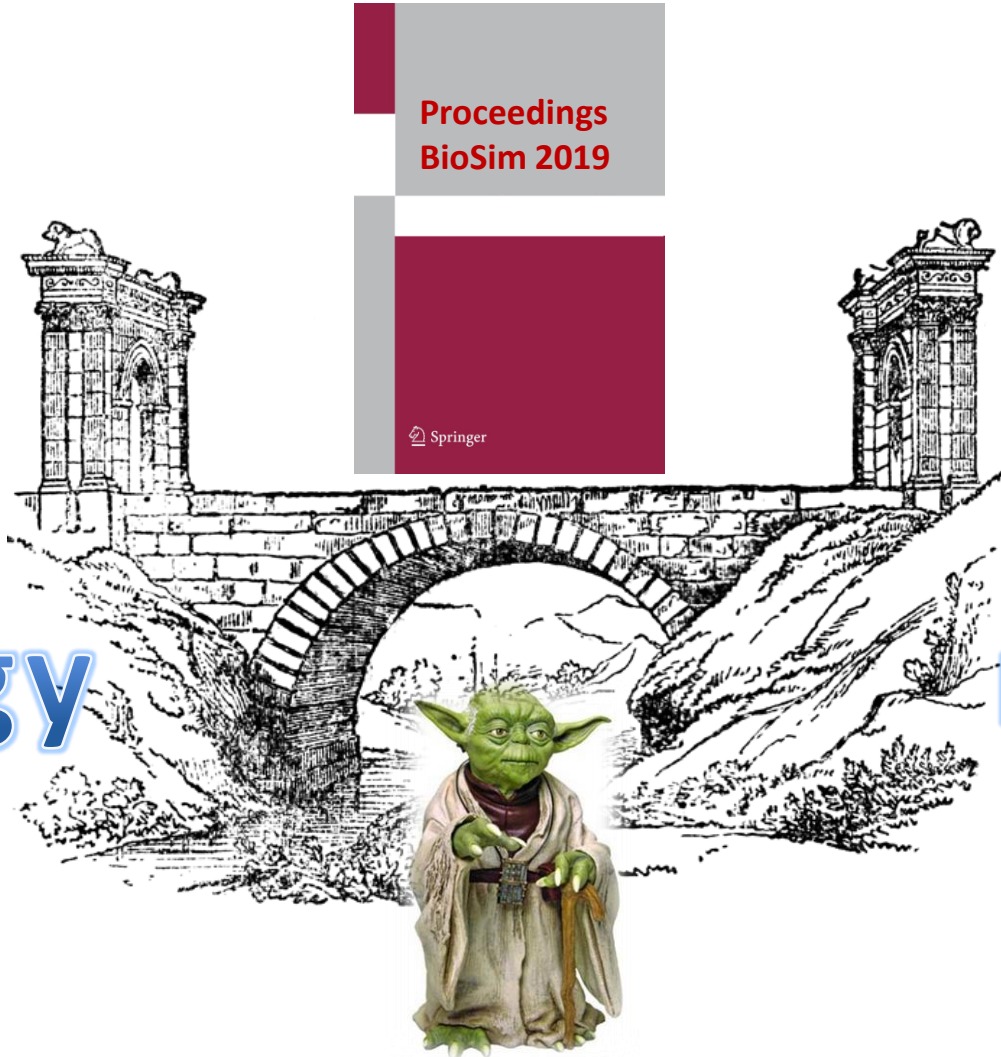
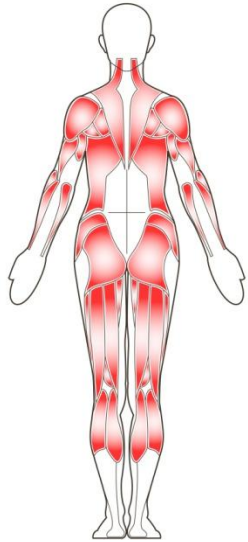
The biographer tries to bridge



biology

maths

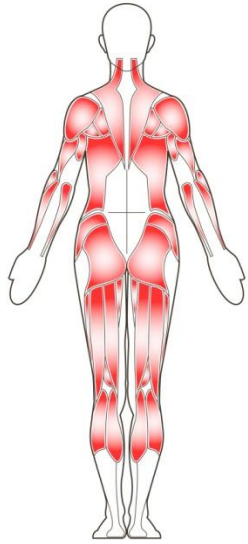
and publishes his G



biology

maths

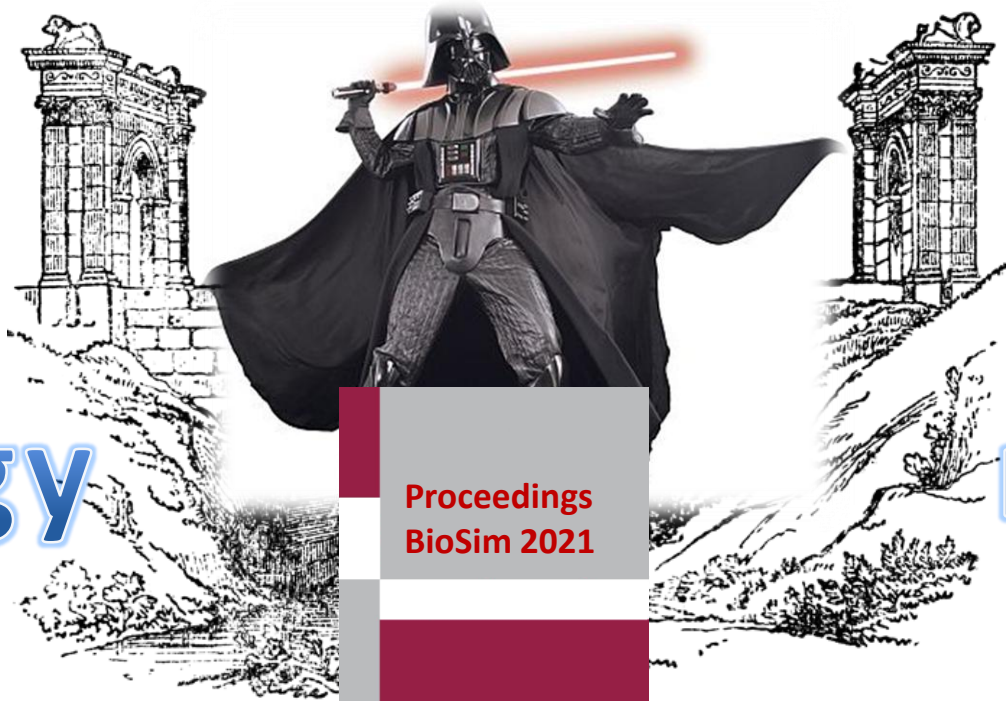
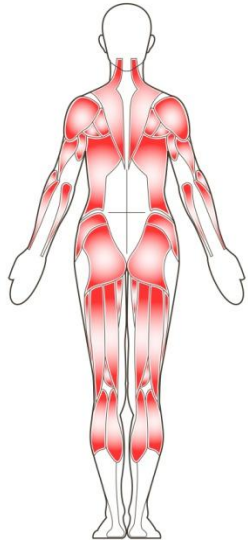
The bioanalyst tries to distinguish model from reality (break the bridge)



biology

maths

If successful, he publishes as well

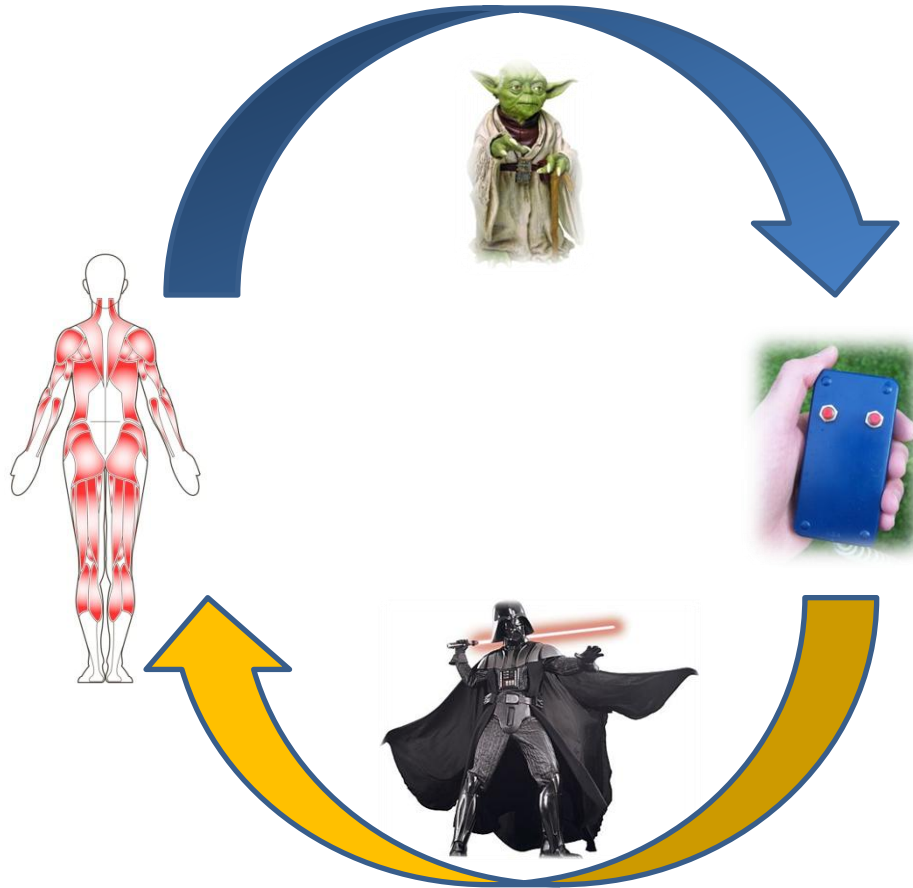


biology

maths

Proceedings
BioSim 2021

Note that so far we did not talk about security or identification at all.

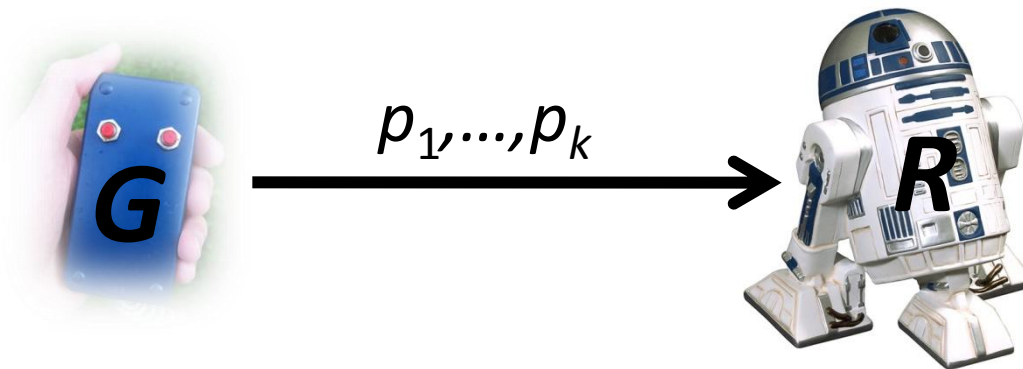


Throughout these iterations we only conceptualize (learn) what a human finger is.

And when the model is stable

Have biometricians do *pure maths*.

Given a synthetic generator G design a recognition algorithm R and *rigorously calculate its FAR and FRR performances*.

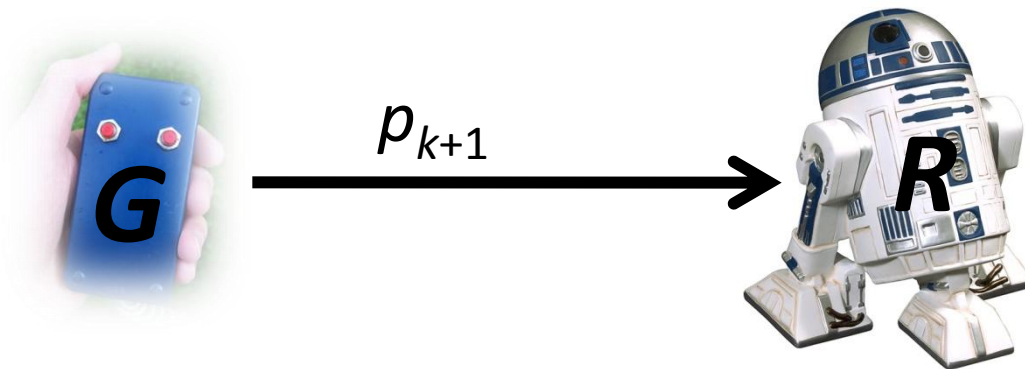


Enrolment phase

And when the model is stable

Have biometricians do *pure maths*.

Given a synthetic generator G design a recognition algorithm R and *rigorously calculate its FAR and FRR performances*.

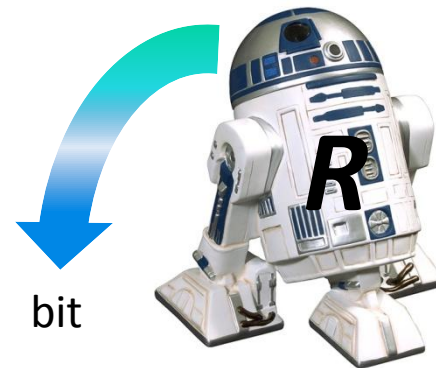


Challenge phase

And when the model is stable

Have biometricians do *pure maths*.

Given a synthetic generator G design a recognition algorithm R and *rigorously calculate its FAR and FRR performances*.



Decision phase

Biology was abstracted away

Decision bits follow a rigorously defined probability distribution $b_{G,R}$

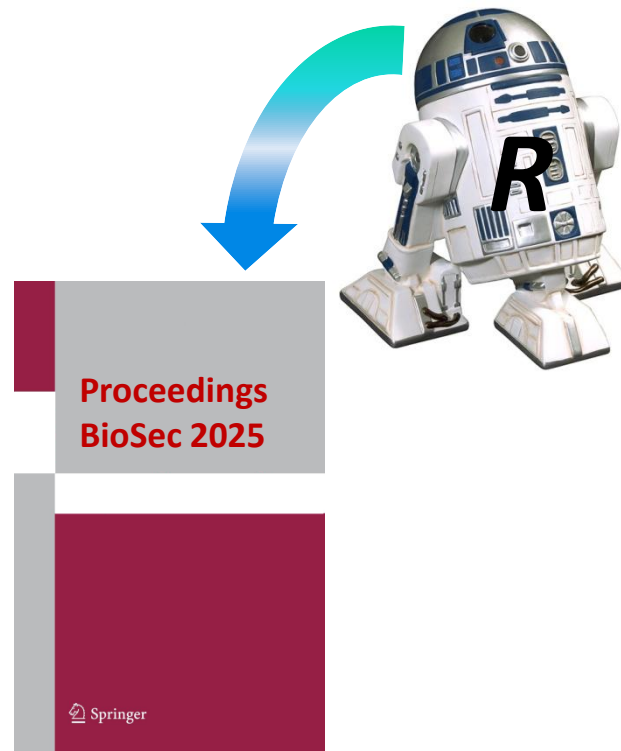
Two pure mathematical objects



We may then hope to *compute rather than measure* **crucial** security parameters such as FAR or FRR.

Evidently

A biometrician proposing a new R should also get a paper...



Export **security reductions**? (sort of)

Subleq is a Turing-complete machine having only one instruction.

subleq a b c

① $*(\text{b}) = *(\text{b}) - *(\text{a})$

② if the result is negative or zero, go to c else execute the next instruction.

The Subleq Machine

Since subleq has only three arguments and since there is no confusion of instructions possible (there is only one!), a subleq code can be regarded as a sequence of triples.

a_1 b_1 c_1

a_2 b_2 c_2

a_3 b_3 c_3

:

...interleaved with data

Since data can be embedded in the code, the sequence of triples can be interleaved with data. For instance:

a_1 b_1 c_1
data₁ data₂
 a_2 b_2 c_2
data₃
 a_3 b_3 c_3
:

How does it work

```
*address_2 = *address_1 - *address_2;  
if (*address_2 ≤ 0)  
    {  
        program_counter = address_3;  
    }  
else  
    {  
        program_counter = program_counter+3;  
    }
```

Allowing for comfort

Memory is loaded with instructions and data altogether (no distinction).

Hence the code can potentially self-modify and consider that any cell is a, b or c.

We can pre-store constants (like 0,1 etc)

e.g. we devote a cell called Z to contain zero, N to contain -1

Finally, the shorthand notation \$ will denote the address where the \$ symbol is.

What does this do?

```
subleq z z c
```

JMP c

subleq z z c

What does this do?

```
subleq a a $+1
```

CLR a

subleq a a \$+1

What does this do?

```
CLR      b
subleq   a  z  $+1
subleq   z  b  $+1
CLR      z
```

MOV b a

subleq b b \$+1

$*b=0$

subleq a Z \$+1

$Z=-*a$

subleq Z b \$+1

$*b=0 - (-*a) = *a$

subleq Z Z \$+1

$Z=0$

What does this do?

subleq a z \$+1

subleq b z \$+1

CLR c

subleq z c \$+1

CLR z

ADD a b c

subleq a z \$+1

$Z=0-*a$

subleq b z \$+1

$Z=-*a-*b$

subleq c c \$+1

$*c=*c-*c=0$

subleq z c \$+1

$*c=0+*a+*b$

sublez z z \$+1

$Z=0$

What does this do?

```
CLR      t
subleq   a  t  $+1
CLR      s
subleq   t  s  $+1
subleq   b  s  c
```

BLE a b c

```
subleq t t $+1      t=0
subleq a t $+1      *t=-*a
subleq s s $+1      *s=0
subleq t s $+1      *s=*a
subleq b s c        *s=*a-*b
                    if *a-*b≤0 goto c
```

What does this do?

```
CLR      t
subleq  a  t  $+1
CLR      s
subleq  b  s  $+1
subleq  s  t  $+1
subleq  N  t  c
```

BHI a b c

```
subleq t t $+1      *t=0
subleq a t $+1      *t=-*a
subleq s s $+1      *s=0
subleq b s $+1      *s=-*b
subleq s t $+1      *t=-*a+*b
subleq N t c        *t=-*a+*b-(-1)
                    if *b-*a+1≤0 goto c
```

What have we got so far?

JMP a goto a

MOV b a *b=*a

ADD a b c *c=*b+*a

BHI a b c if *b-*a+1≤0 goto c

if *b<*b+1≤*a goto c

if *b<*a goto c

if *a>*b goto c

BLE a b c if *a-*b≤0 goto c

if *a≤*b goto c

CLR a *a=0

Even more powerful

MOV L1 a


data Z

data Z

L1: data Z

BRX a

MOV	L1	a	*L1=*a
data	Z		
data	Z		
L1: data	Z		



What else do we need?

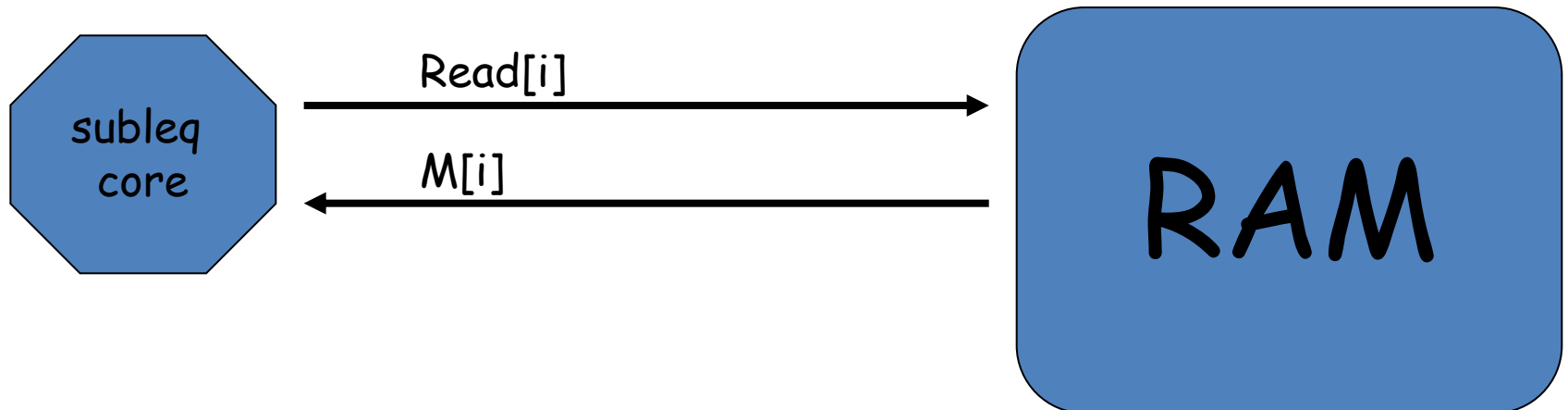
Boolean operations such as AND, XOR.

We only need one (e.g. using LUTs) because

$$A + B = \sum_{i=0}^7 2^i (B_i + A_i) = \sum_{i=0}^7 2^i (B_i \oplus A_i) + \sum_{i=0}^7 2^{i+1} B_i A_i = A \oplus B + 2(A \wedge B)$$

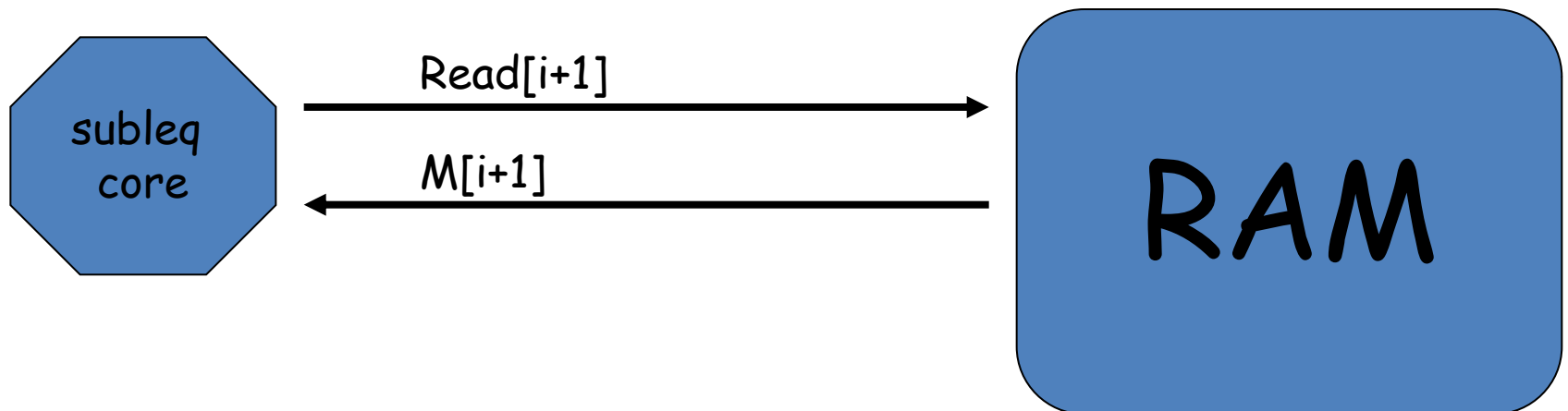
Hardware Architecture

We assume that we have a RAM initialized with the code.



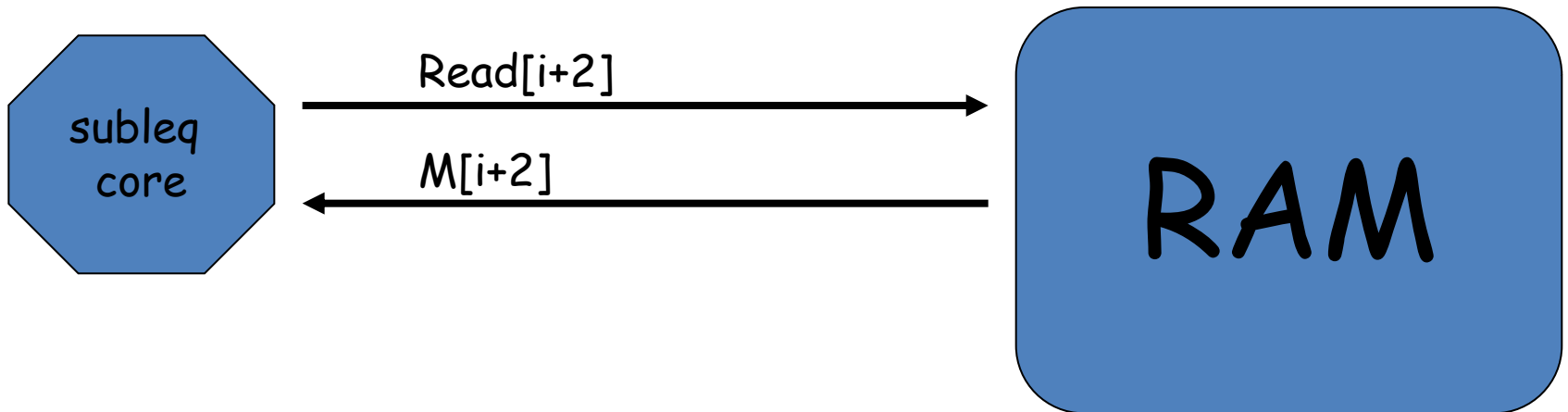
Hardware Architecture

We assume that we have a RAM initialized with the code.



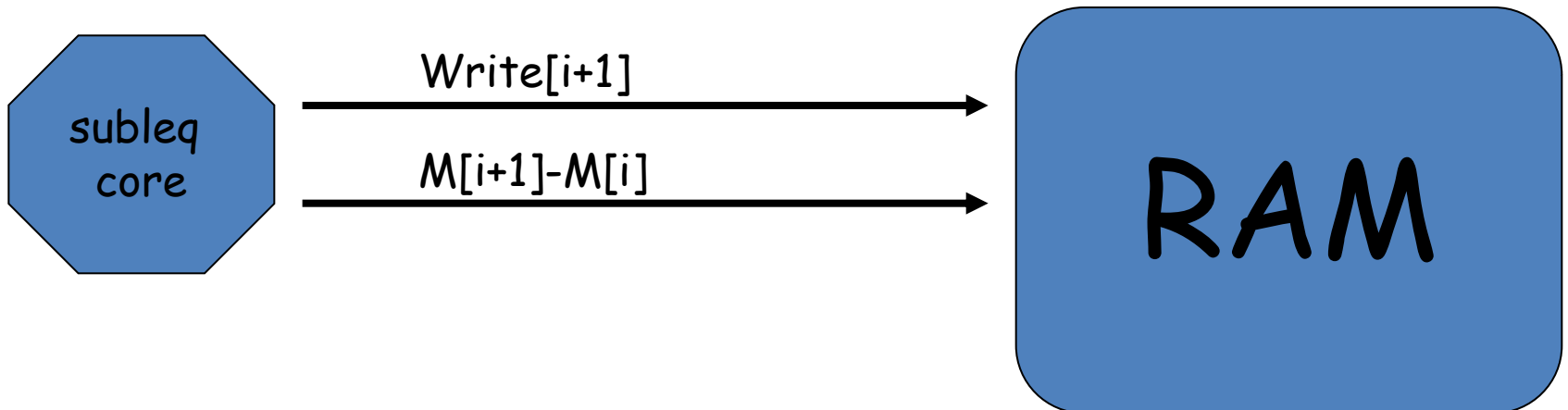
Hardware Architecture

We assume that we have a RAM initialized with the code.



Hardware Architecture

We assume that we have a RAM initialized with the code.



Where is all this heading?

Chips suffer from side channel leakage.

Protecting against side channels has grown into a science in itself (CHES).

Since a subleq machine can compute **any** algorithm (e.g AES, RSA, SHA), a universal secure core will **reduce** the problem of side channel resistance to the problem of protecting this core.

This isn't a new countermeasure but an attempt to reduce the global side channel resistance issue to the defending of a well defined hardware core.

The analogy

This isn't a new countermeasure but an attempt to reduce the global side channel resistance issue to the defending of a well defined hardware core.

Just as in crypto we endeavor to find reductions of resistance against various mathematical attacks to the solving of well defined hard problems.

In practice

A one instruction set is an extreme.

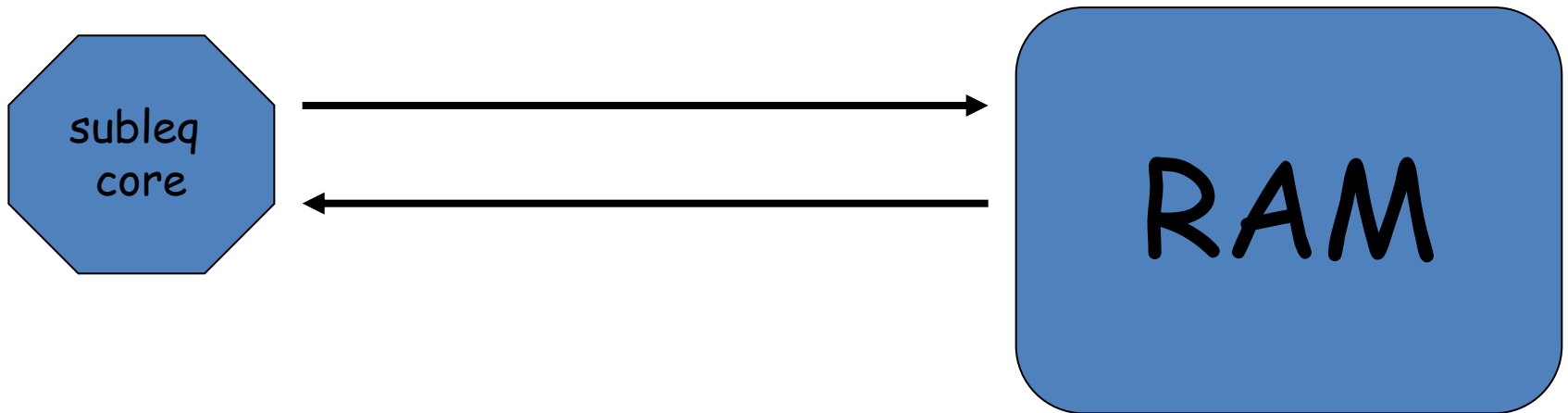
Practical solutions could relax it a bit.

Given the device's simplicity it is easy to model, understand and protect.

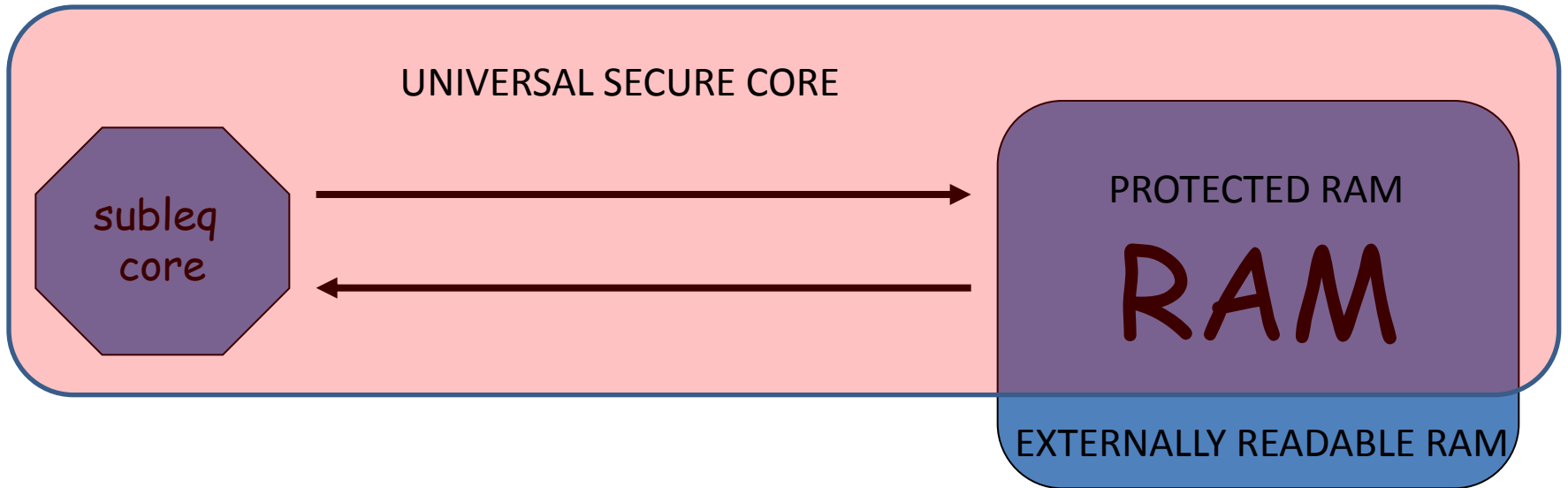
The device's regularity (always execute the same instruction) is a security advantage: only data varies.

Simplicity \Rightarrow fast clocking.

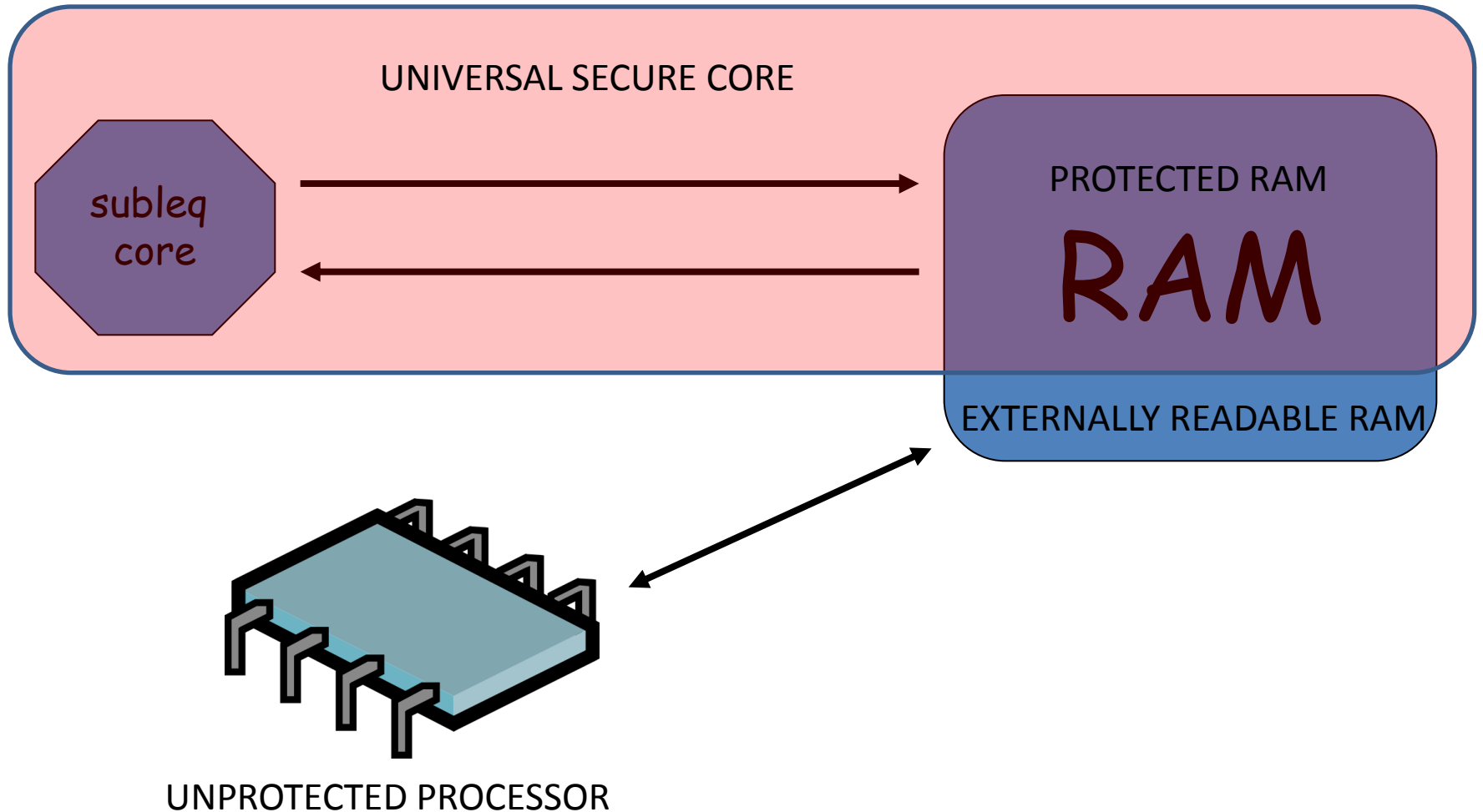
General Architecture



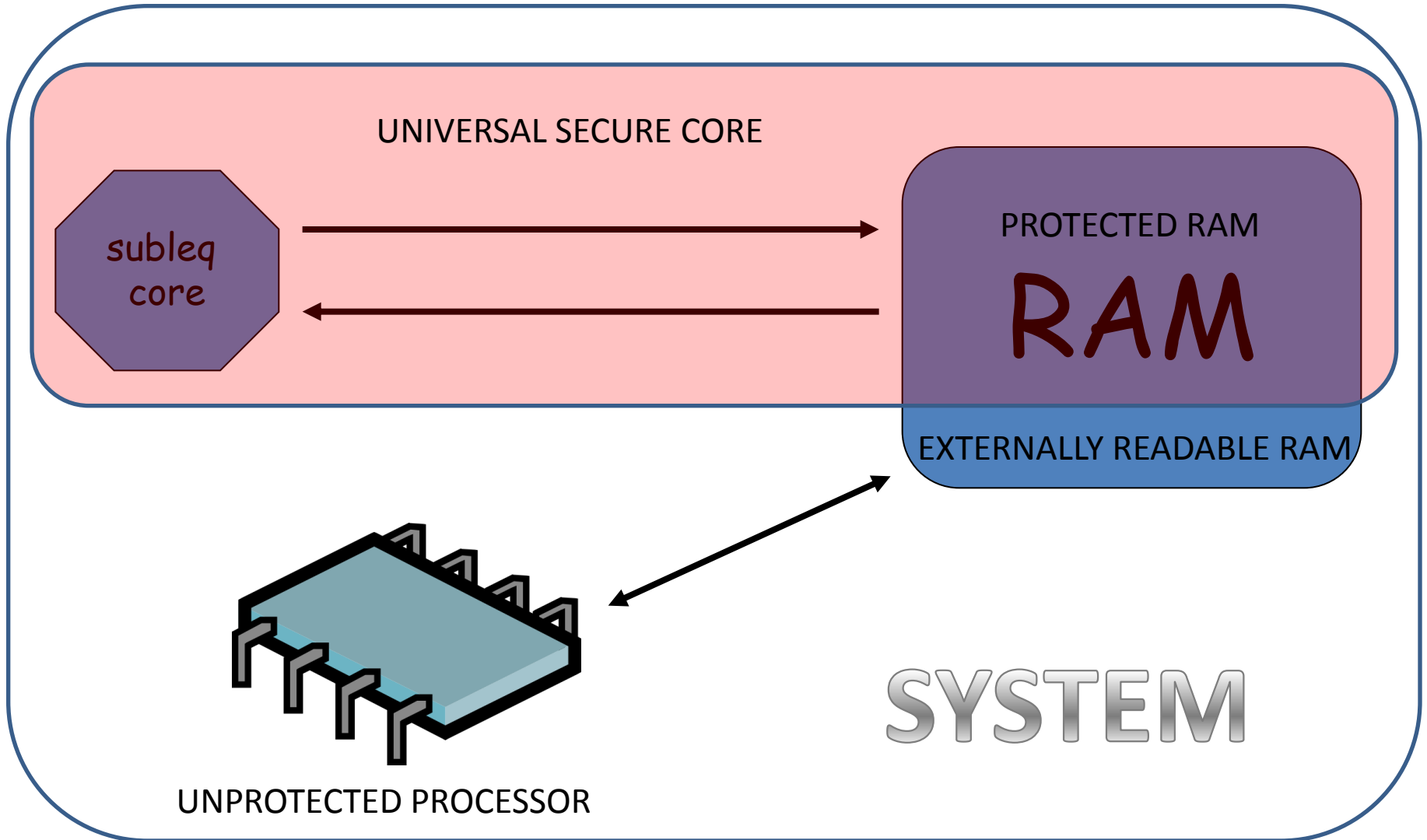
General Architecture



General Architecture



General Architecture

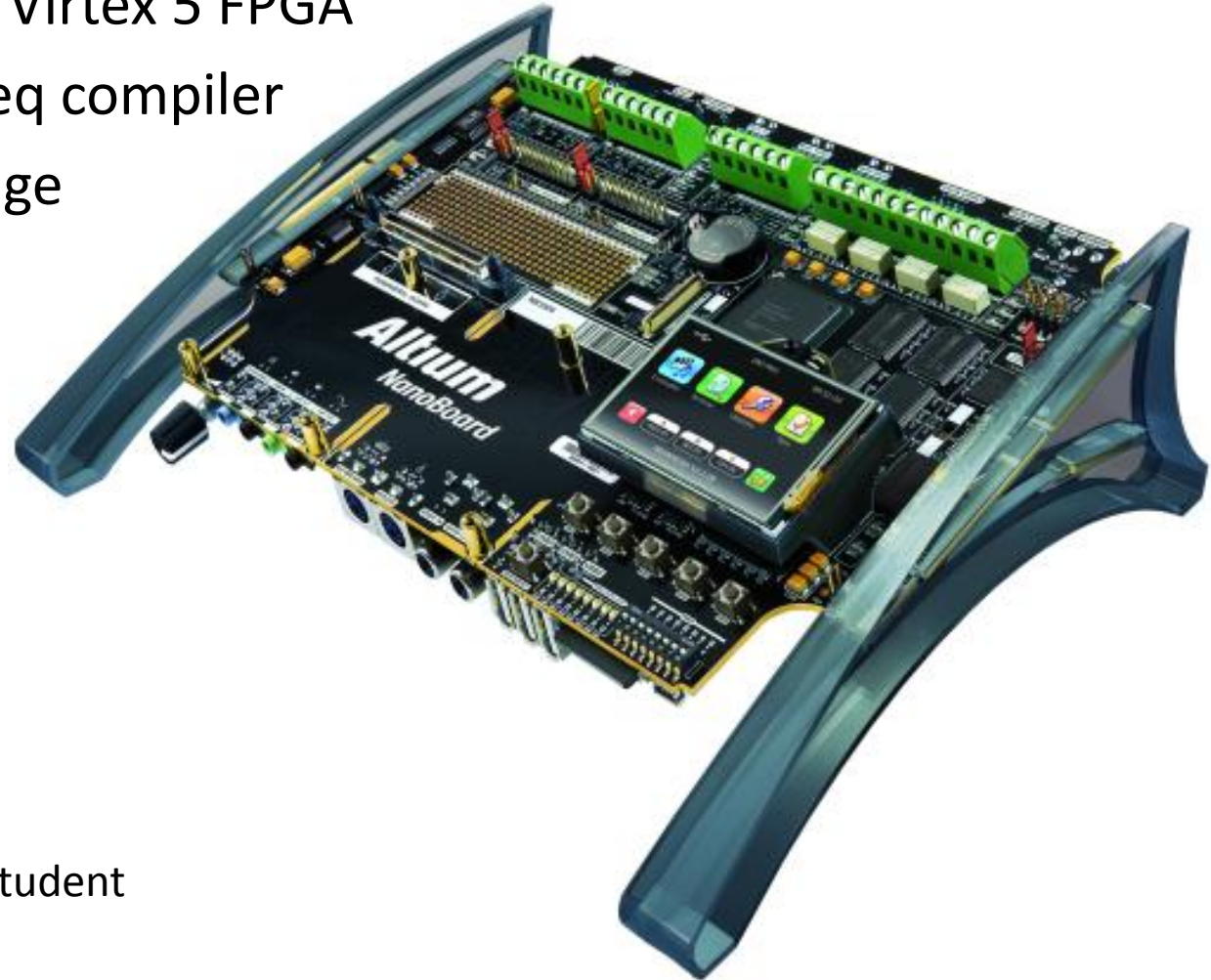


The machine was implemented

Prototype running on Virtex 5 FPGA

We coded a C to subeq compiler

Analyzed power leakage



For details: contact my Ph.D student
florian.praden@ens.fr

**Find new
theoretical
solutions**

Homomorphism is on fashion...

Imagine a signature scheme K, S, V with the following property: **S does not sign strings but propositions.**

The signature $S(p)$ of a proposition p symbolizes the statement that “ p is true”.

In addition, we want an *inference transform* R .

R transforms two signatures into a new signature:

$$R(S(p), S(q)) = S(r(p, q))$$

where r is an inference rule associated to R .

We call this **propositional signatures**

Example.

If R is the *modus ponens inference transform* then:

$$R(S(A \Rightarrow B), S(A)) = S(B)$$

Caveats:

We want R 's output to be indistinguishable from a directly calculated (*i.e.* non inferred) signature of B .

We require soundness *i.e.* if proposition c cannot be inferred from a and b then $S(c)$ should not be inferable from $S(a)$ and $S(b)$

More potent **logical signatures**?

First order logic signatures?

Lambda signatures?

Example:

$$R(S(\neg \exists x P(x))) = S(\forall x \neg P(x))$$

or

$$\eta(S(\lambda x.(P x))) = S(P)$$

or

$$\beta(S(\lambda n.n \times 2), S(7)) = S(7 \times 2)$$

Many practical applications

Such constructions will have many applications:

- Provide a user with a sequence of signed program instructions allowing him to prove that a given output was generated by the signed instructions and/or from signed input.

Caution: this is not a proof that signed instructions were used in the correct order! Proving ordered execution is an open problem that can be tackled before propositional signatures are exhibited.

- Boil down the signatures $S(\text{"paid \$10"})$, $S(\text{"paid \$40"})$ and $S(\text{"paid \$50} \Rightarrow \text{DVD unlock})$ into the signature $S(\text{"DVD unlock"})$ giving access to contents.

If you are not convinced

To the geeks

- Prove that you know a proof of Fermat's last theorem without revealing the proof.
- Either you know the proof or you broke S .

To the perverts

- Break S
- Instead of submitting the result to Crypto exhibit the signature S (" S is forgeable in polynomial time").
- To drive the community crazy.

non fundamental

My current list of 10 theory questions

1. Construct a propositional signature scheme.

Conjecture: possible.

2. Can a one bit public-key black box encryption resist active attackers?

- Bob is given a black box that PK-encrypts a bit
- Bob cannot see the box's random tape
- Bob sends a message m to Alice.
- Alice replies with $h(m)$

Can this be secured against active attackers?

Conjecture: no.

3. Construct a non-interactive authenticated PKE scheme:

- Alice encrypts a message for Bob and mixes with it a secret.
- Bob can ascertain that the message came from Alice.
- Bob cannot convey this conviction to anybody.

Conjecture: should be possible if not done yet. Maybe a candidate.

non fundamental

My current list of 10 theory questions

4. Construct a function $f(x)$ whose computation using $< g(x)$ space is provably reducible to hard problem.

Variant: find such a function with a trapdoor (allowing the trapdoor's owner to compute $f(x)$ in $< g(x)$ space).

Conjecture: probably possible using AONTs?

5. Clarify RSA fixed-pattern-based-forgery limits.

Given an a -bit pattern L find an $3a$ -bit perfect cube having L as middle bits.

Trivial when L is LSB, trivial when L is MSB...

Application: jailbreak a very popular mobile phone.

How far can one push the pattern size in poly-time Affine RSA forgery.

$\frac{1}{4}$ the size of n will be a stunning result. Current record: $\frac{1}{3}$ (Crypto'01).

Cryptanalysis of PKCS#1 v.1.5 signatures.

Frustrating. No security proof. No known attack

non fundamental

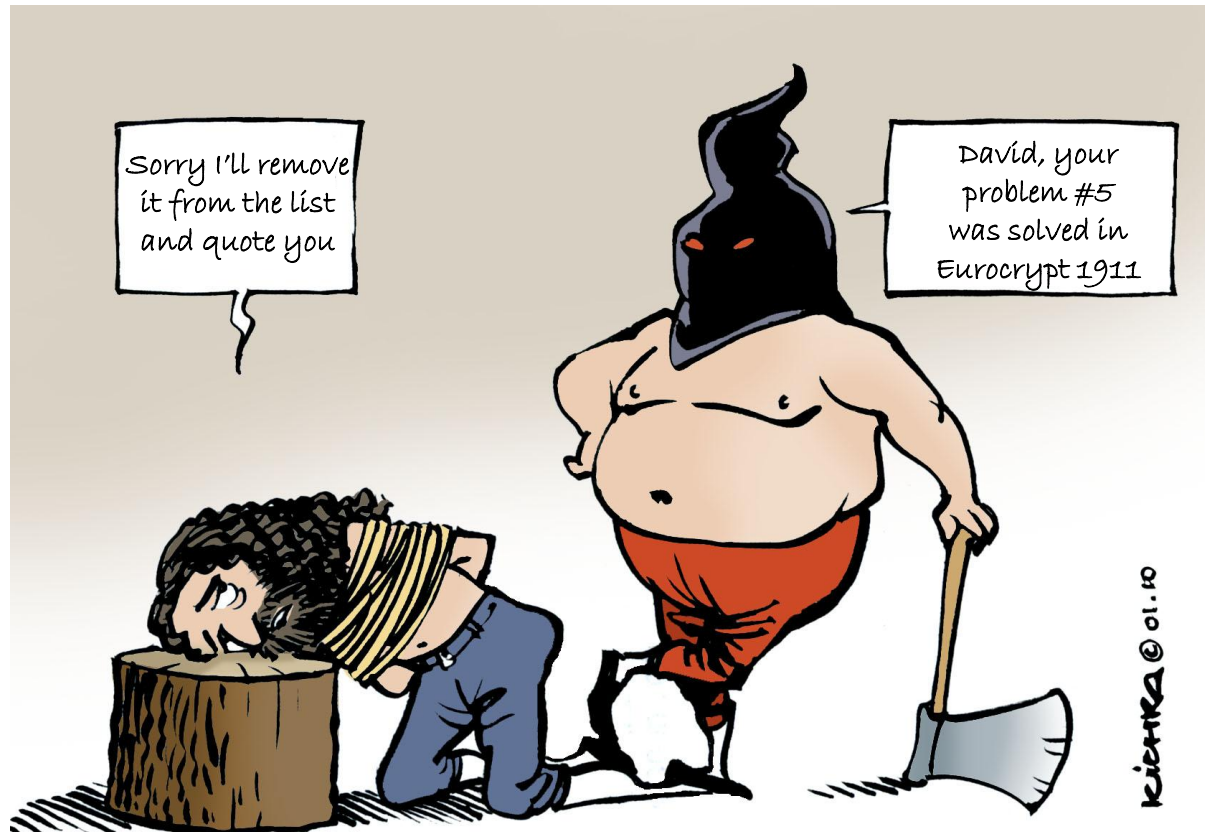
My current list of 10 theory questions

6. Side channel can be used to attack a platform and to fingerprint (attest) it.
Challenge: link physical side channel to the logical responses.
Tune leakage to attest while not revealing keys.
Promising approach: tripartite Diffie Hellman
7. Construct a one way permutation which is not a trapdoor OWP whose key was thrown away (nor a $g^m \bmod p$).
Important practical applications in the design of hash functions.
Strictly no idea.
8. Devise an identity-based fully homomorphic encryption scheme.
A promising starting point: a variant of Smart-Vercauteren PKC'10 scheme.
We also attempt to backwards generate the PK of other FHE schemes.
9. Generic fault models.
See next slides.
10. Privacy refractory functions.
See next slides.

Refer to the end of this slideshow for more information on some of the problems

Disclaimer

I am David, not David Hilbert. I don't read all LNCS proceedings. It is possible that some of the listed questions already has solutions or answers. Don't execute me.



**Create
theory**

Cryptography in an ideal model



Cryptography in reality

Paracryptography (fields bordering cryptography)
lack theoretical attention from the community.



Cryptography in reality

Paracryptography (fields bordering cryptography)
lack theoretical attention from the community.



Cryptography in reality

Paracryptography (fields bordering cryptography)
lack theoretical attention from the community.



Fault attack theoretical models

Imagine a subleq code P implementing CRT-RSA.

Allow the attacker to feed the device with input of his choosing and to *substitute* n subleqs.

This is a “clean” model abstracting away physics.

Goal: *Design code which is provably resilient against the modification of at most n instructions.*

Resilient: *P either halts or outputs results that can't be used for conducting an RSA fault attack.*

Fault attack theoretical models

Note that this is close to the Generic model where we perform computations through *pointers* on data rather than on data.

The three arguments of a subleq are *pointers*.

A subleq can never directly manipulate data.

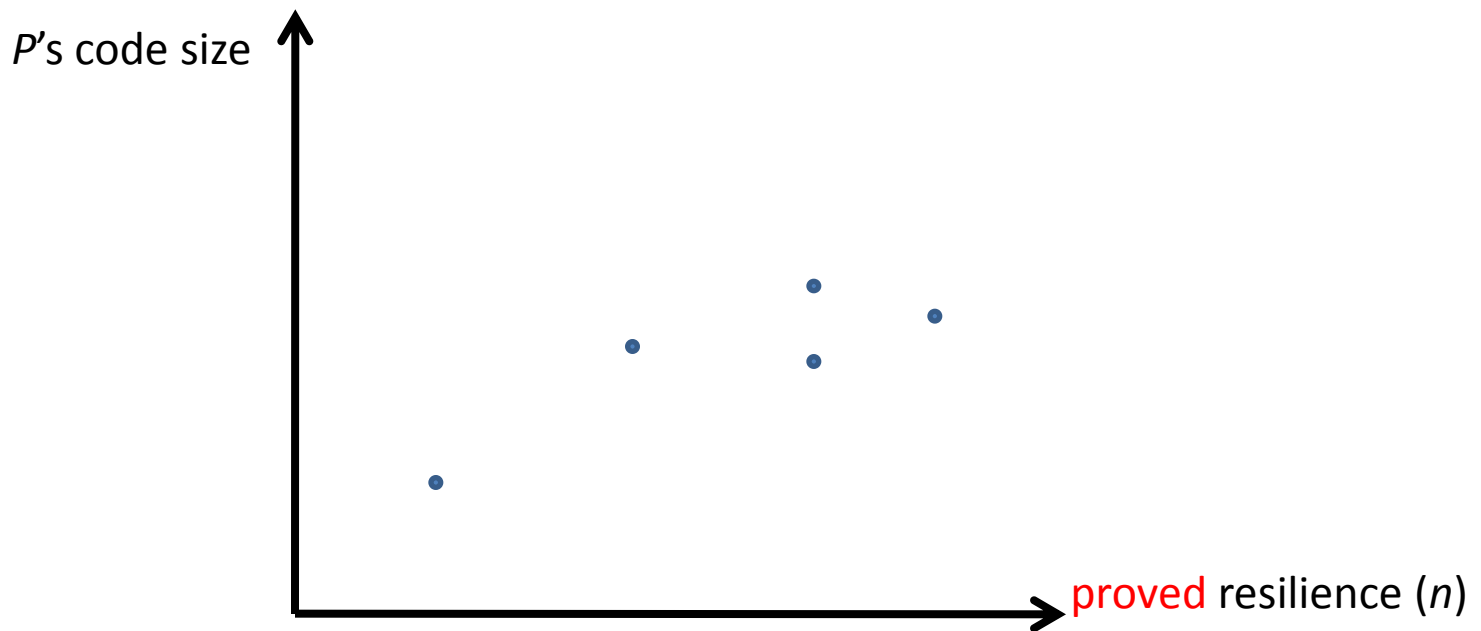
*In other words, we want a generic algorithm capable of resisting a certain number of **operation hijacks**.*

A new adversarial model.

Fault attack theoretical models

Can we plot n as a function of P 's code size?

Each dot is a code written and proved by someone.



Fault attack theoretical models

The need:

Analyze faults as effects on a mathematical virtual machine implementing cryptography, not only as faults in mathematical formulae.

Privacy refractory functions

Homomorphic encryption is great for privacy.

My banker can multiply my balance by 1.01 to credit interests, subtract a \$113 charge and add a \$221 cash deposit without decrypting.

But the tool is not perfect.

Consider the following game

Alice gives to Bob two integers a and b .

Bob combines a and b using the operations $+$, $-$, \times say to a multiplication depth of 10.

He gives the result r to Alice.

Can Alice tell how Bob computed r ?

Can Alice do so efficiently?

The tree grows quickly

step 0

{a,b}

The tree grows quickly

step 0 $\{a,b\}$

step 1 $\{a,b,a+b,ab,2a,2b,a^2,b^2\}$

The tree grows quickly

step 0 {a,b}

step 1 {a,b,a+b,ab,2a,2b,a²,b²}

step 2 68 terms

The tree grows quickly

- step 0 {a,b}
- step 1 {a,b,a+b,ab,2a,2b,a²,b²}
- step 2 68 terms
- step 3 2556 terms

The tree explodes quickly

step 0 {a,b}

step 1 {a,b,a+b,ab,2a,2b,a²,b²}

step 2 68 terms

step 3 2556 terms

step 4 `General::ovfl : Overflow occurred in computation. >>`

`Out[3]= Overflow[]`

**If we forget commutativity and distributivity
growth is doubly exponential.**

However...

r is a bivariate polynomial in a and b .

$r' = r \bmod a$ is a univariate polynomial mod a in b .

If the mult-depth is reasonable, we can bound the coefficients of r . Assume that they are all $< a$.

We can attempt to solve a modular knapsack in the powers of b . Let $r' = \sum c_i b^i \bmod a$

Subtract from r the $\sum c_i b^i$ we found in \mathbb{Z} , divide by a and start over!

r may reveal what was the computed function.

Data vs. Operations

Cryptanalysis: extract data (k) from data (c, m).



Computational reversion: Extract the *calculation history* from the data (k, c, m are all known, how was c computed from k and m ?).

Rarely possible but still sometimes possible.

What operations' history is inferable from data?

A practical concern that got little attention so far.

In conclusion

Theoretical cryptographers dig deep and efficiently.

In conclusion

*Theoretical cryptographers dig deep and efficiently.
Don't forget to also dig sideways!*



In conclusion

Theoretical cryptographers dig deep and efficiently.

Don't forget to also dig sideways!

And even to dig up!

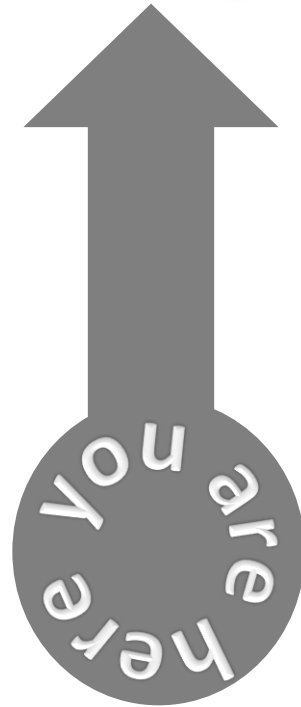


We can finally give a tentative answer

Where can we go from here?



**TO THE BEACH
BBQ!!!**



Appendix

Additional information on some of the problems.

More on problem 2

Bob uses a PK black-box $f(b)$ encrypting the bit b for Alice.

Alice uses a black-box $f^{-1}(f(b))=b$ to decrypt $c=f(b)$.

Assume that Bob authenticates Alice by sending an encrypted message $f(m_1), \dots, f(m_k)$ to which Alice replies with $h(m_1, \dots, m_k)$.

This is trivially attackable by an active adversary.

Charlie encrypts a zero bit using f , let c' be this ciphertext.

Charlie feeds Alice with $c', f(m_2), \dots, f(m_k)$. If Alice responds with the same hash than Charlie infers that $m_1=0$, else $m_1=1$.

The process is repeated for each bit in turn.

More on problem 2

Assume now that Bob authenticates Alice by sending an encrypted message $f(m_1), \dots, f(m_k)$ to which Alice replies with $r, h(r, m_1, \dots, m_k)$ where r is a fresh random picked by Alice.

This time Charlie picks a message u_1, \dots, u_k and challenges Alice with $f(m_1), f(u_2), \dots, f(u_k)$. This allows Charlie to infer m_1 .

The process is repeated for each bit in turn.

It *seems* impossible to protect the exchange against active attackers if no access to f 's randomness is granted to Bob, but this remains to prove.

Importance: what are the minimal secure operation conditions of a low bandwidth PKE scheme?

More on problem 4

Alice buys a 10TB smart disk.

A smart disk is equipped with both 10TB and a processor Bob.

Alice wants to check the disks capacity.

Trivial solution: fill it with random data and read back.

Disadvantage: communication. Alice needs to send in 10TB.

Idea: have Alice send to the disk a small seed x , Bob expands x into $g(x)$ where $g(x)$ is 10TB long. Bob responds to Alice with $h = \text{SHA}(g(x))$.

Problem 1: find a g provably not allowing Bob to cheat i.e. compute h in more time and less space.

Problem 2: find a g as in 1 that Alice can compute in less than 10TB while Bob cannot (Alice uses a trapdoor).

More on problem 5

Given the green chunk L , determine the red variables x, y, z so that the equation holds over the integers:

$$\boxed{x} \boxed{L} \boxed{y} = \boxed{z}^3$$

Easier variant? :

$$\boxed{x} \boxed{L} \boxed{y} = \boxed{z}^2$$

More on problem 6

Platform physical attestation consists in having a terminal inject into a target card T and a reference sample card R the same code. Once the code is installed in the two cards, the cards are fed with identical random input and run under identically varying voltage and clock. The power consumption of the two devices is correlated to ascertain that T is not a logical clone of R (that is, that T and R belong to the same family of physical devices).

While doing this, countermeasures and on-board RNGs must be turned off (so as not to disturb the correlation process).

In addition T cannot access any of its non volatile secrets (as these are unknown by R).

Note that T and R create a common secret (e.g. DH) unknown by the terminal before starting the attestation process.

More on problem 6

As the attestation is complete, countermeasures and RNGs are turned and the terminal executes a cryptographic protocol with T (for instance T signs a challenge message).

This does not ascertain that T is not a logical clone.

The attacker can use a genuine T during the attestation and when the attestation is over, switch to a logical clone.

The challenge consists in finding ways to “weld” the two phases.

For instance, have some data d that derives from the attestation phase (and which is only known by T and R) injected into the cryptographic protocol in a way that: d does not leak during the attestation (dangerous game: countermeasures are off!) and is still verifiable as correctly injected into the protocol by the terminal.

More on problem 7

We are looking for a OW permutation transforming, say 100 bits into 100 bits whose structure would not be based on “heavy” NT machinery.

The object should be something looking like a keyless block cipher whose operation is not reversible.

Passing message chunks through this OWP before hashing with a function h is interesting: if an attacker finds a collision in h , he still has to reverse the OWP to create a real message collision.

No candidate except “costly” public-key-like functions.