Empowering Gateways with Functional Encryption

Yolan Romailler
@AnomalRoil

August 2019 — Cryptovillage

$ whoami

→ Cryptographer at Kudelski Security in Switzerland

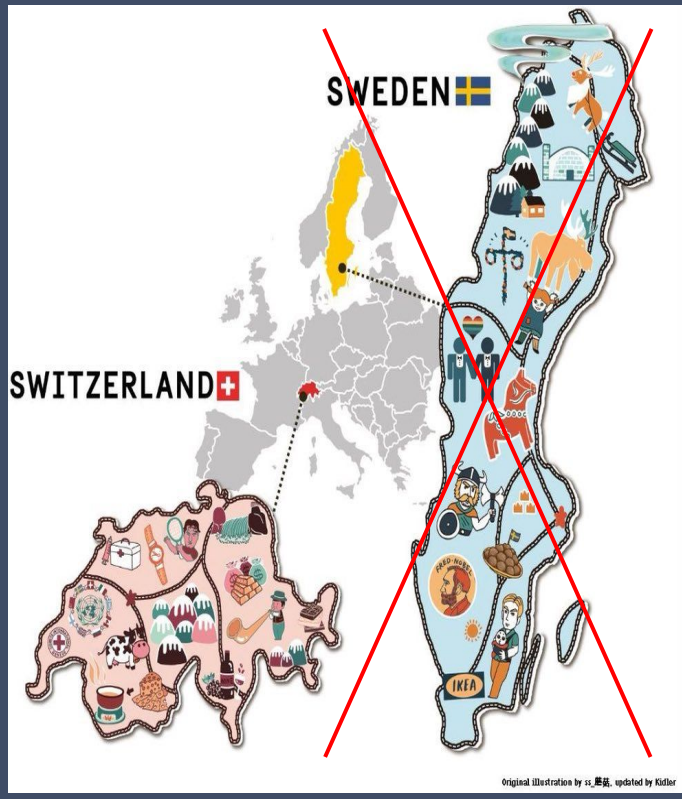
→ CTF player (mostly crypto, forensic & misc)

→ Board games player

→ Biker

→ Go coder (“one day I’ll have time, yknw”)

→ Speaker at conferences (BH, DF, NSec, ...)
Second time at Cryptovillage!



Agenda

- What is Functional Encryption (FE)
- Questions

Starting with encryption

Public-key encryption is ubiquitous on internet!

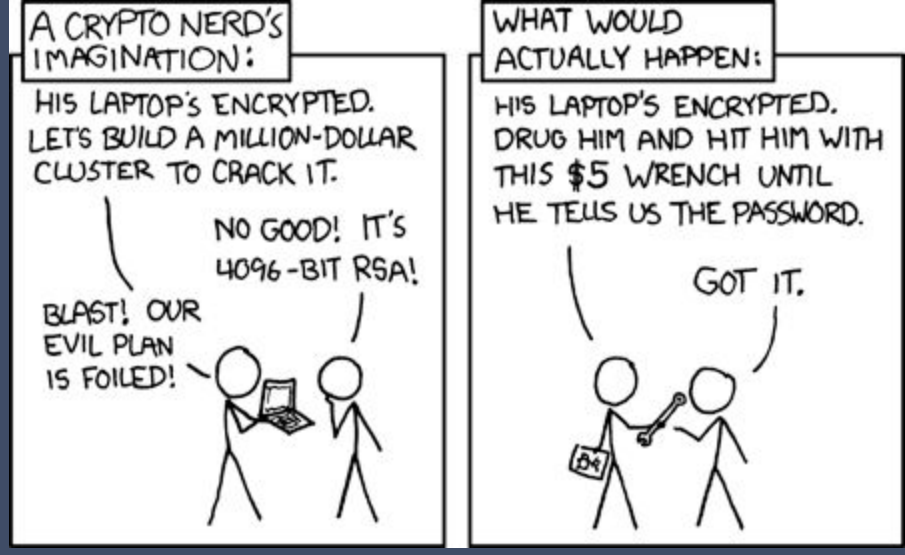
Be it HTTPS, SSH, messaging, voip, storage, … everything seems to rely on public-key schemes to ensure security nowadays.

But encryption is usually seen as being an **all or nothing** option:

- either you have it,
- or you don't!

In a nutshell, functional encryption is all about challenging that idea. :)

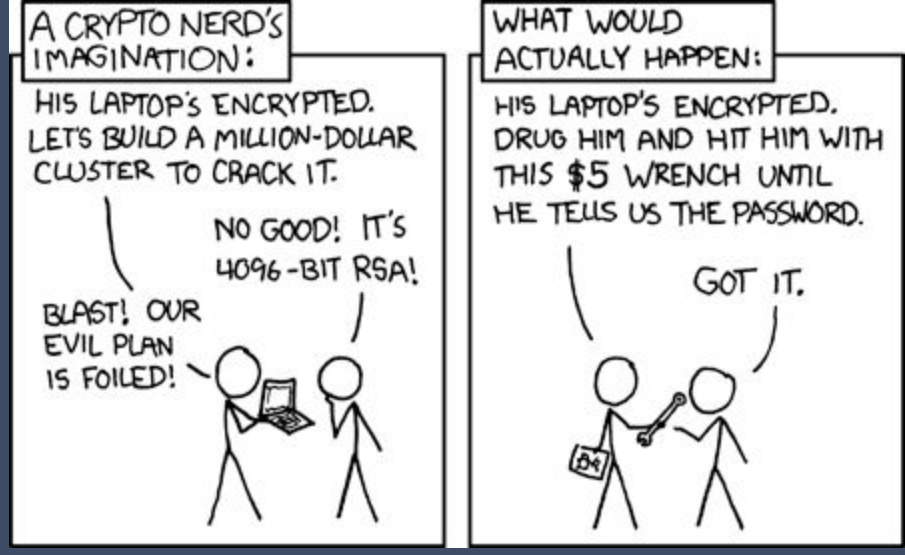
Crypto refresher: RSA



- RSA (Rivest–Shamir–Adleman)
  - Choose two **large prime numbers** p and q, typically 1024-2048 bits.
  - Public key **(n, e)**
    - with n = p * q
    - and some e such that e and λ(n) are coprime
  - Private key **(n, d)** where $d \equiv e^{-1} \pmod{\lambda(n)}$

RSA security relies on the hardness of the **integer factorization problem**

Crypto refresher: RSA



- RSA (Rivest–Shamir–Adleman)
  - Public key **(n, e)**
    - with n = p * q
    - **e** such that **e** and **λ(n)** are coprime
  - Private key **(n, d)** where $d \equiv e^{-1} \pmod{\lambda(n)}$
  - RSA encryption of message m < n:
    - $c = m^e \pmod{n}$
  - RSA decryption of ciphertext c:
    - $c^d \pmod{n} = m^{ed} \pmod{n} = m^1 \pmod{n} = m \pmod{n}$

Crypto refresher: ECC

ECC aka “Elliptic Curve Cryptography”

- Working with the subgroup of prime order **n** of an **elliptic curve C** generated by a given **generator G**.
- Security based on the hardness of the EC discrete logarithm problem (DLP).

In common EC schemes (ECDSA, ECDH, ECIES), we usually have that:

- A private key is simply an integer k < p (it is an element of $\mathbb{F}_p$)
- A public key is a point P = (x, y) on the curve generated by **scalar multiplication** of the private key with the base point: **P = kG**
  - where (x, y) are the coordinates of the point **on** the curve C

Back to business: Functional Encryption (FE)

- Functional encryption was first proposed by Amit Sahai and Brent Waters in 2005

- Then Dan Boneh, Amit Sahai and Brent Waters formalized the notion of functional encryption in a paper in 2010

- We can say FE is a **public-key encryption scheme** with different decryption key**s** allowing to learn the result of a specific function of the encrypted data.

What is FE all about?

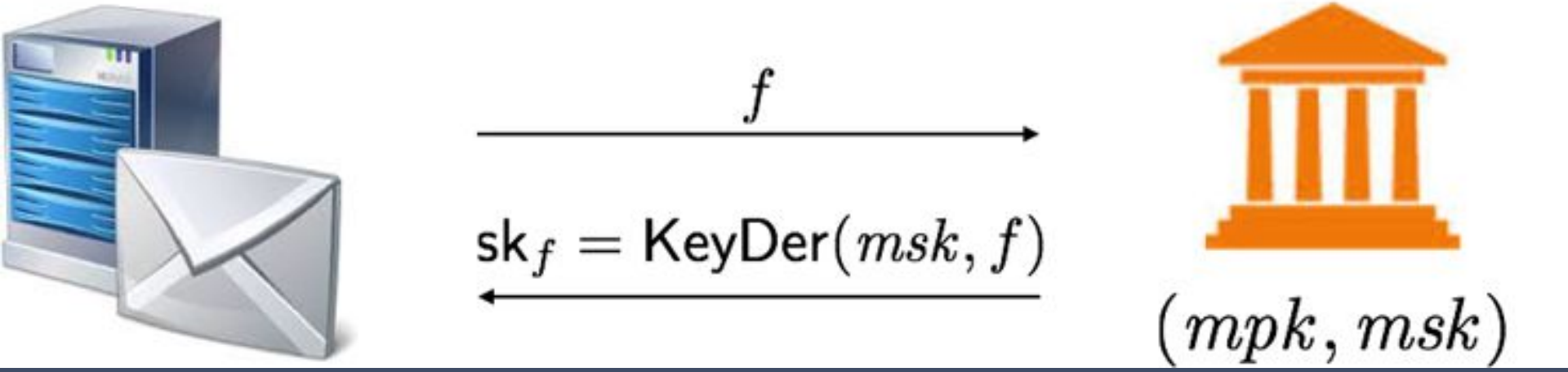
Before FE, it was accepted that:

- Public-key encryption is a method to send secret data **to a single entity** holding a given secret key corresponding to the public key.

- Access to the encrypted data is **all or nothing** – one can either decrypt and read the entire plaintext or one learns nothing about the plaintext.

Functional encryption tries to change these!

More details about the setup of FE schemes



FE vs FHE

A nice way to look at FE is to think about it as being similar to homomorphic encryption!

The idea is almost the same: you got encrypted data, and you want to evaluate functions on the data.

Excepted the result is in **clear** !

That is with FHE you would get **E(f(x))** out of **E(x)**
whereas with FE you get **f(x)** directly as a plaintext!

Why FE?

There are many different scenario in which we might want to evaluate a specific function on encrypted data without needing access to the plaintext data!

- Processing of data by a partially trusted 3rd party
- Delegating complex computation to remote, powerful servers
- Having a fine-grained access control on the encrypted data (who has access to what, and what they can compute)

Classical example

If $\mathsf{Eval}(\mathsf{Sk}_{\mathsf{SpamFilter}}, \mathsf{Enc}(m)) = \mathsf{True}$
Move to the Spam Folder.

Quick math refresher: the inner product

The inner product is a generalization of the dot product to generic vector spaces.

It is a way of multiplying vectors together in order to associate a scalar to every pair of vectors in that vector space.

Today I'll be working with the finite field $\mathbb{Z}_p$ for a prime p, and the inner product of two vectors x,y $\in \mathbb{Z}_p^{\ell}$ is :

$$\langle x, y \rangle = \sum_{i \in [\ell]} x_i y_i$$

Inner product FE scheme!

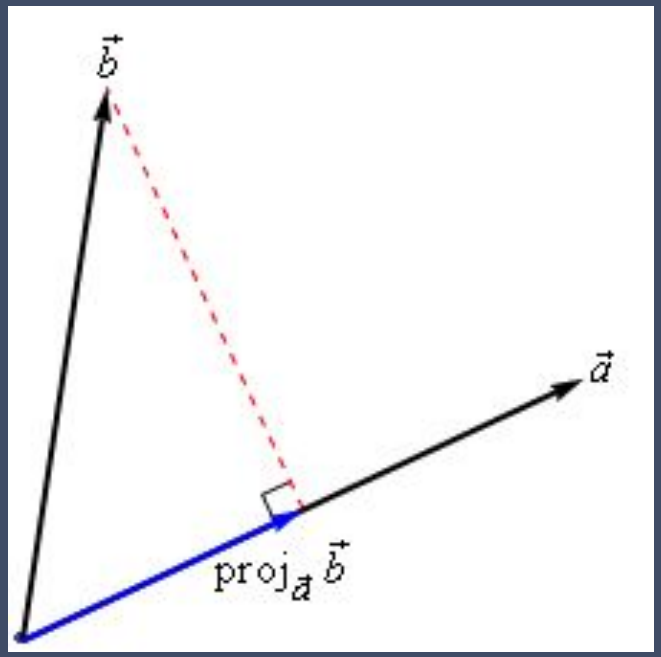
It is possible to build FE scheme that allow us to compute the inner product of an encrypted vector together with a vector specified by the decryption key!

Even cooler, the security of such scheme can be based on the DDH assumption.

See https://eprint.iacr.org/2015/017 for the gory details ;)

An information leaked!

When computing the inner product you are actually projecting the vector onto each other!



In some ways, FE can be seen as a **controlled information leak**

Other examples of FE schemes

Functional encryption can also be seen as a generalization of encryption schemes such as:

- Identity-based encryption (IBE)
- Attribute-based encryption (ABE)
- Searchable encryption

and other systems for predicate encryption!

A couple examples of use-cases

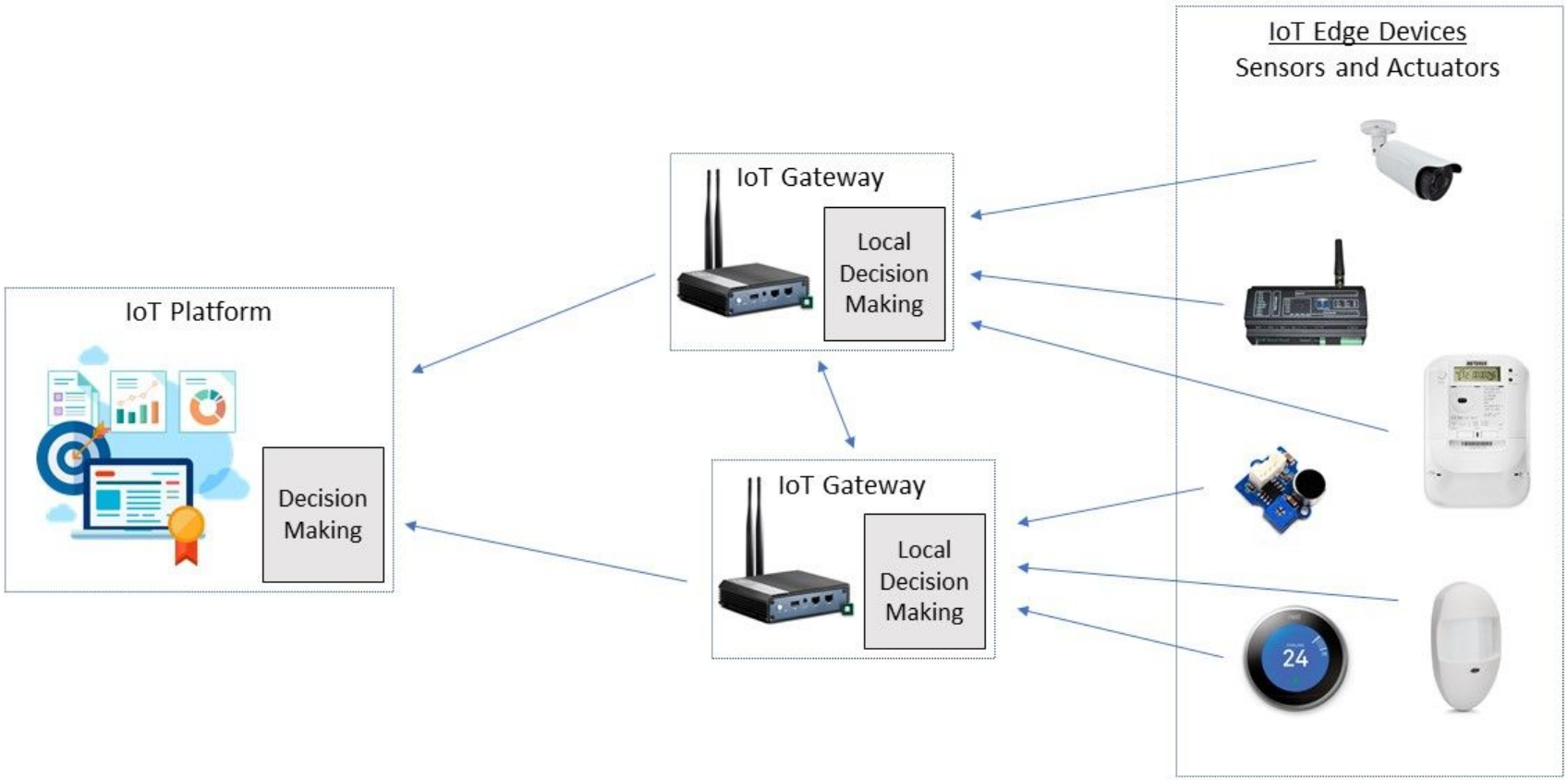
FE allows you to take decisions at the gateway level, even if the data in transit is end-to-end encrypted for the backend system.

One example: **local decision making in a smart grid network**.

Compliance requires end-to-end encryption between the smart meter in the house and the backend systems of the electricity provider.

You might want to take decision earlier at the gateways if something goes wrong on the network.

Local what?


IoT Edge Devices
Sensors and Actuators
IoT Gateway
Local Decision Making
IoT Platform
Decision Making
IoT Gateway
Local Decision Making
24

Another example

Machine learning on encrypted data!

Not even kidding:

Try it out:

https://github.com/fentec-project/neural-network-on-encrypted-data

which is implementing https://eprint.iacr.org/2018/206

Motion detection use-case

Now, how about using the inner product scheme to detect motion?

Guess what, MPEG video contains… motion vectors!

We could extract these motion vectors and rely on them to perform motion detection by summing their values using the inner product with the all 1 vector!

$$\left\langle \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_\ell \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \right\rangle = \sum_{i \in [\ell]} x_i$$

And then, if we reach a certain threshold, we've detected motion!

Quick demo?

Here we go!

A quick performance overview

There are currently three types of FE schemes:

- Inefficient, theoretical functional encryption for **all circuits**.
  This is interesting for science but is still far from practical.

- Predicate encryption, Attribute Based Encryption, Identity Based Encryption and schemes such as the Inner product schemes that have restricted functionalities that can be and have been implemented in practice. These are usually slow, but usable.

- Schemes relying on trusted third parties, secure enclaves, secure hardware or other “cheats”... ;) These can even be fast, but are maybe not exactly what we might want.

Now what?

Existing schemes are often very limited, like the inner product schemes.

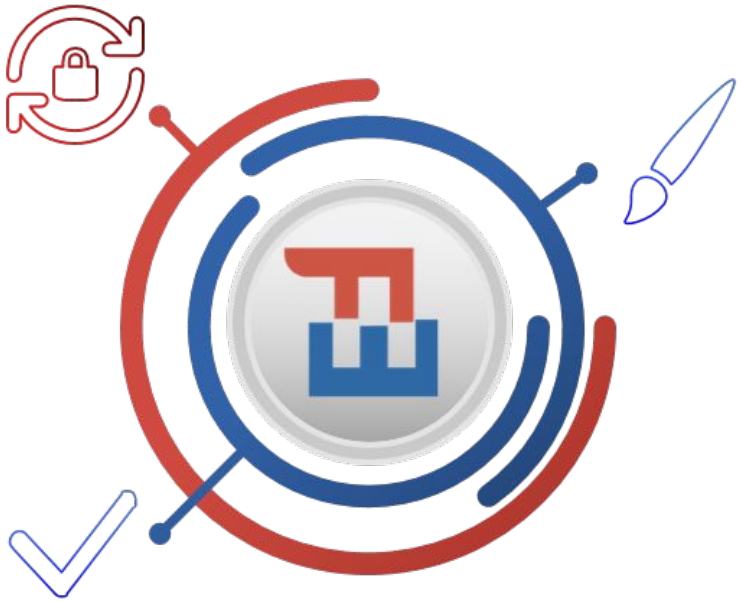
The question remains whether using lattices (which have really helped with FHE) can help achieve richer functionalities for FE.

Also, the FENTEC project tries to push the boundaries of Functional encryption further and develop new FE technologies.

FENTEC's consortium

The goals of the FENTEC project

Implement a unified **cryptographic API** of Functional Encryption systems



Design **functional encryption systems** with varying functional, security, **hardware** and **software** requirements

Validate and **demonstrate** FENTEC technologies and **solutions**

The FENTEC library!

You can try it today!

https://github.com/fentec-project/gofe

https://github.com/fentec-project/CiFEr

It implements:

- 5 inner product schemes (multi-input, decentralized, etc.) that can be instantiated under different assumptions, from DDH to (Ring)LWE.
- 1 quadratic polynomial scheme
- 1 CP-ABE scheme
- 1 KP-ABE scheme

Future works

- Hardware accelerators (work in progress at the university of Helsinki)
- Richer functionalities (work in progress by the other unis)
- Quantum-safe schemes
- Integrating more FE schemes into the FENTEC library (wip by XLAB)
- Implementing prototypes that use of the FENTEC library
- Getting you to use FE technologies!! ;)

Follow me on Twitter:
@AnomalRoil

The end & some links!

- http://fentec.eu/
- https://github.com/fentec-project/gofe
- https://github.com/fentec-project/CiFEr
- https://github.com/fentec-project/neural-network-on-encrypted-data

- https://research.kudelskisecurity.com