nano-tera.ch
swiss scientific initiative in health / security / environment systems

QCrypt RTD 2010

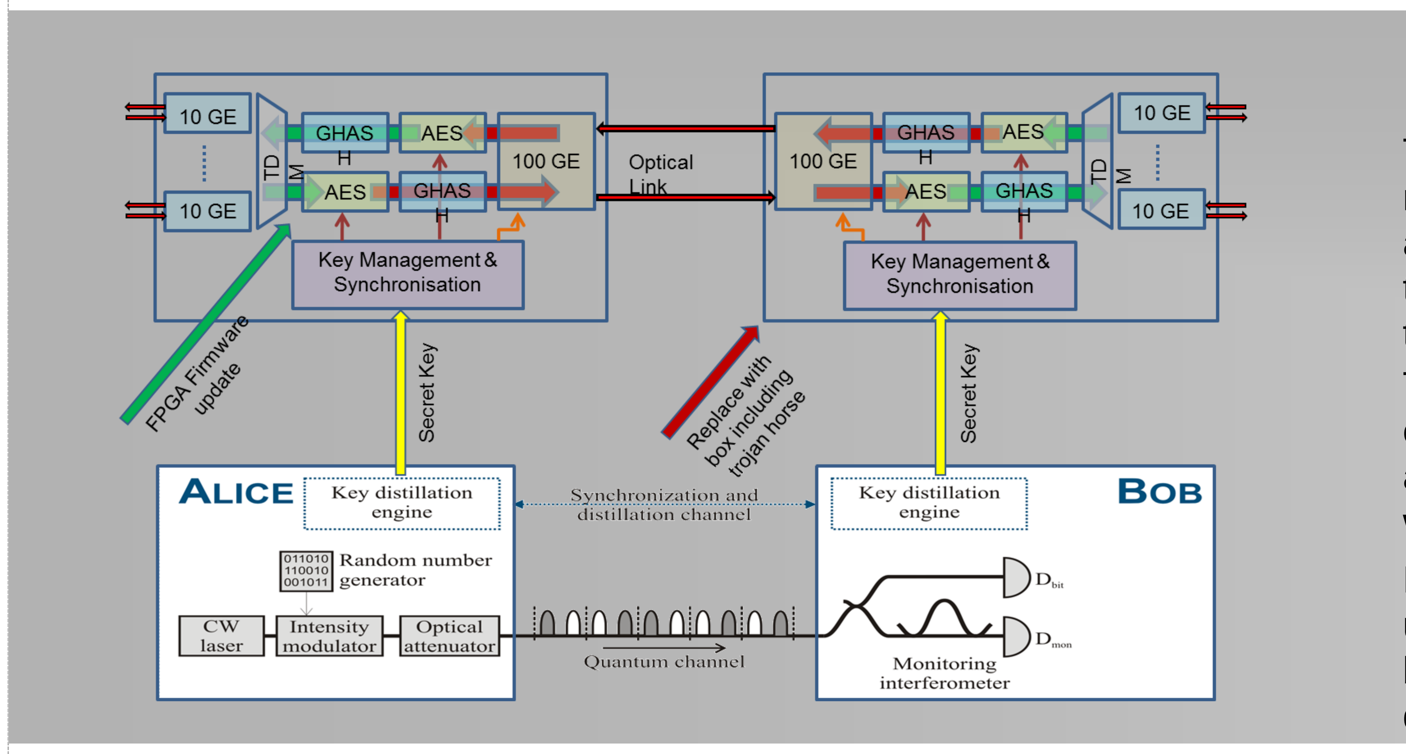
FNSNF

Physically Unclonable Functions for Secure Hardware

Ch. Keller[1], N. Felber[1], F. Gürkaynak[1], H. Kaeslin[1], P. Junod[2]

[1]IIS ETHZ, [2]IICT HEIG-VD

UNIVERSITÉ DE GENÈVE · ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE · ETH Zürich · Hes·so Haute Ecole Spécialisée de Suisse occidentale · IDQ



Securing the System

The current system has a few possible loopholes which could be used to alter the system, read out information and manipulate the information which is sent through it. A modified FPGA configuration bit stream, or firmware, could be applied to the board. I.e., such a modified firmware could constantly disable the encryption such that all the payload is transmitted in plaintext. A second scenario could be that one of the hardware platforms is being replaced by a fake one. A third scenario could be the eavesdropping of the secret key link.

To secure the system, we need to have a subsystem on the board that helps preventing manipulation of the FPGA configuration bit stream, authenticates all the involved hardware and stops operation if unknown hardware is connected, and authenticates and encrypts the secret key, which is transferred between the QKD and the enCryptor.

We decided to build such a subsystem by using so called physically unclonable functions (PUF).

Physically unclonable functions are devices which exploit physical variations of integrated circuits (IC) to generate a unique, device specific output pattern. The physical variations are introduced in the manufacturing process and tend to be highly random. Therefore, even with complete manufacturing instructions, the behavior of the PUF can never be duplicated – it is unclonable.

DRAM PUF

PUFs proposed so far:

- Race conditions: delay differences in "equal" pairs of signal paths
- Ring oscillators: frequency variations
- Static RAM (SRAM): power-up pattern
- Optical: light propagation in passivation layer to on-chip photo diodes

Our proposal:

→ Dynamic RAM PUF (DRAM PUF)

Patent filed:
"Generating Unique Numbers Using Charge Decay Phenomena"

(the patent covers several other charge-decay based effects suitable for PUFs)

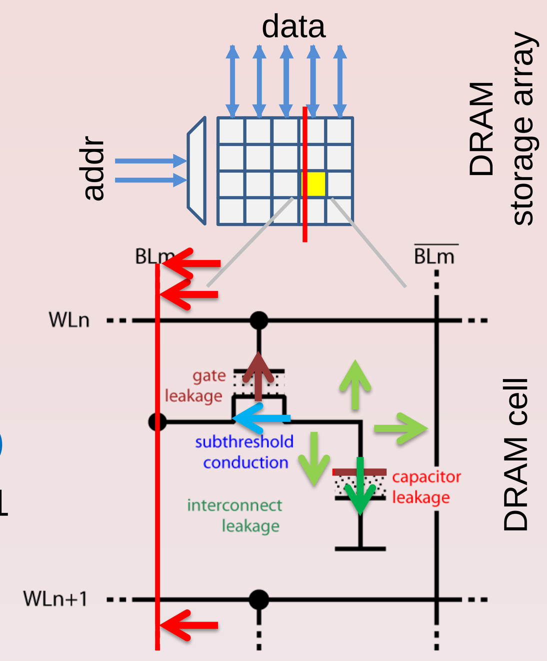
	unclonable	inputs	outputs	random
Race conditions	yes	some	few	(yes)
Ring oscillators	yes	some	few	(yes)
Optical	yes	some	few	(yes)
SRAM	yes	none	many	no
DRAM	yes	many	many	yes

Using DRAM as a PUF circuit has some advantages over the other implementations. The most significant one is the large input space. An arbitrary input pattern can be written to the memory array and a corresponding output pattern can be gathered which is, ideally, statistically independent of the input pattern.

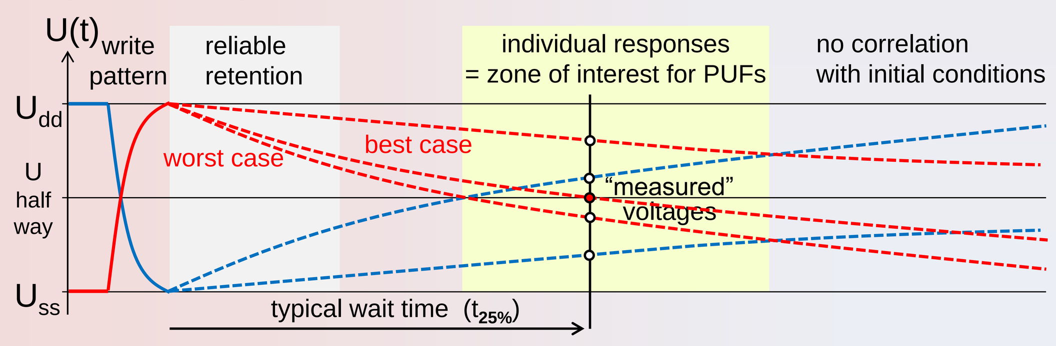
DRAM PUF Operation

The PUF operation conducts the following tasks:

- write pattern = PUF input (raw)
- state is stored on capacitors
- refresh is disabled
- leakage (de)charges capacitors
  physical variations
- read word(s) = PUF output (raw)
- sense amplifiers discriminate 0|1
  physical variations



When retrieving the node charge, timing is an important factor. The reliability of the output pattern of the PUF directly depends on that timing. If the storage nodes are read out too early, almost no changes have occurred. If we wait too long, the charges have vanished and no information can be extracted. Therefore, the optimal time window has to be found after every startup and even during normal operation.

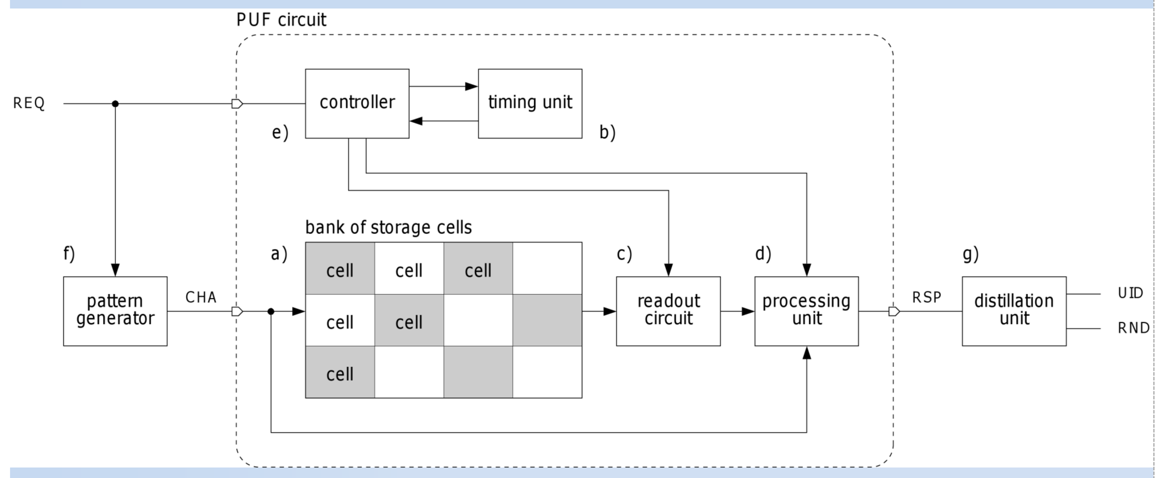


Initialization: find wait time
repeat
- write pattern
- wait t(k++)
- read pattern

until 25% cells toggled

PUF readout: evaluate function
- (input pre-processing)
- write pattern
- wait $t_{25\%}$
- read word(s)
- (output post-processing)

Data Processing



Timing is not the only property which needs to be controlled and adapted to the current operating conditions. Pre- and post-processing of the input data pattern and the output data pattern may be applied.

It is possible, that certain input pattern are not suitable for the PUF as their output is not statistically independent of the input. This could be countered with a pre-processing unit (f). One possibility is the usage of a specifically adapted hash function to prevent the input pattern from being too regular.

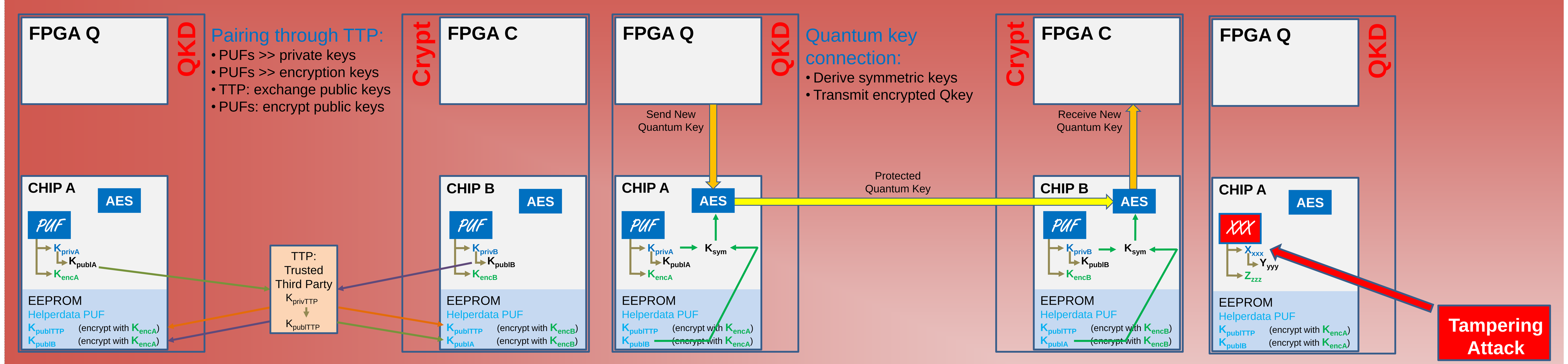
The output pattern will most certainly vary over time when applying the same input pattern. This can be caused by the leakage varying over time, temperature, or radiation. In this case, the output can be seen as a noisy signal. It has been proposed to use forward error correction to generate a static output.

When input ***C*** is applied to the DRAM PUF for the first time, the corresponding output ***R*** is read out. In the distillation unit (g), a code word ***E*** is calculated for this output ***R***.

If the same input ***C*** is applied at a later time, a noisy output ***R**** is extracted. With ***E***, the original ***R*** can be restored.

Further more, the random part of ***R**** can be extracted and be used as true-random numbers.

Key Exchange Secured with PUF Device



From the PUF, a private key and an encryption key is extracted.

This private key is used to calculate a public key.

Using a Trusted Third Party, the respective public keys are exchanged.

A new pairing can only be done through the same TTP.

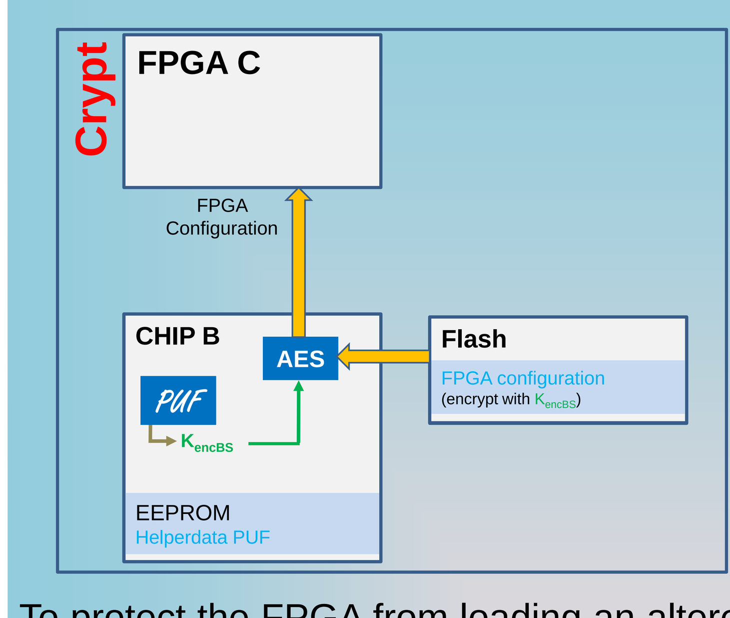
Using the own private key and the public key of the other device, a symmetric key is calculated. (Diffie-Hellman)

This key is then used to encrypt the data to be transmitted between QKD and Crypt.

The authentication of the other device is done implicitly. If i.e. the PUF device on the Crypt board is replaced, it cannot decrypt the transmitted Quantum Key since it does not have the public key of CHIP A.

If the system is physically attacked, the structures of the IC are altered. The PUF will therefore output an altered pattern and the original private key and encryption key is irretrievably destroyed.

Further Applications



To protect the FPGA from loading an altered configuration, the configuration is encrypted at the manufacturer with a device specific key. This key is retrieved during first setup at the manufacturer.

When the configuration is loaded, it is decrypted and applied to the FPGA. In case of a manipulated configuration file, the decryption will produce a random bit stream and the FPGA will not be working.

Principal Investigator: Nicolas Gisin (Nicolas.Gisin@unige.ch)

Contact: Norbert Felber (felber@iis.ee.ethz.ch)