

Bistream management in FPGA based secure applications

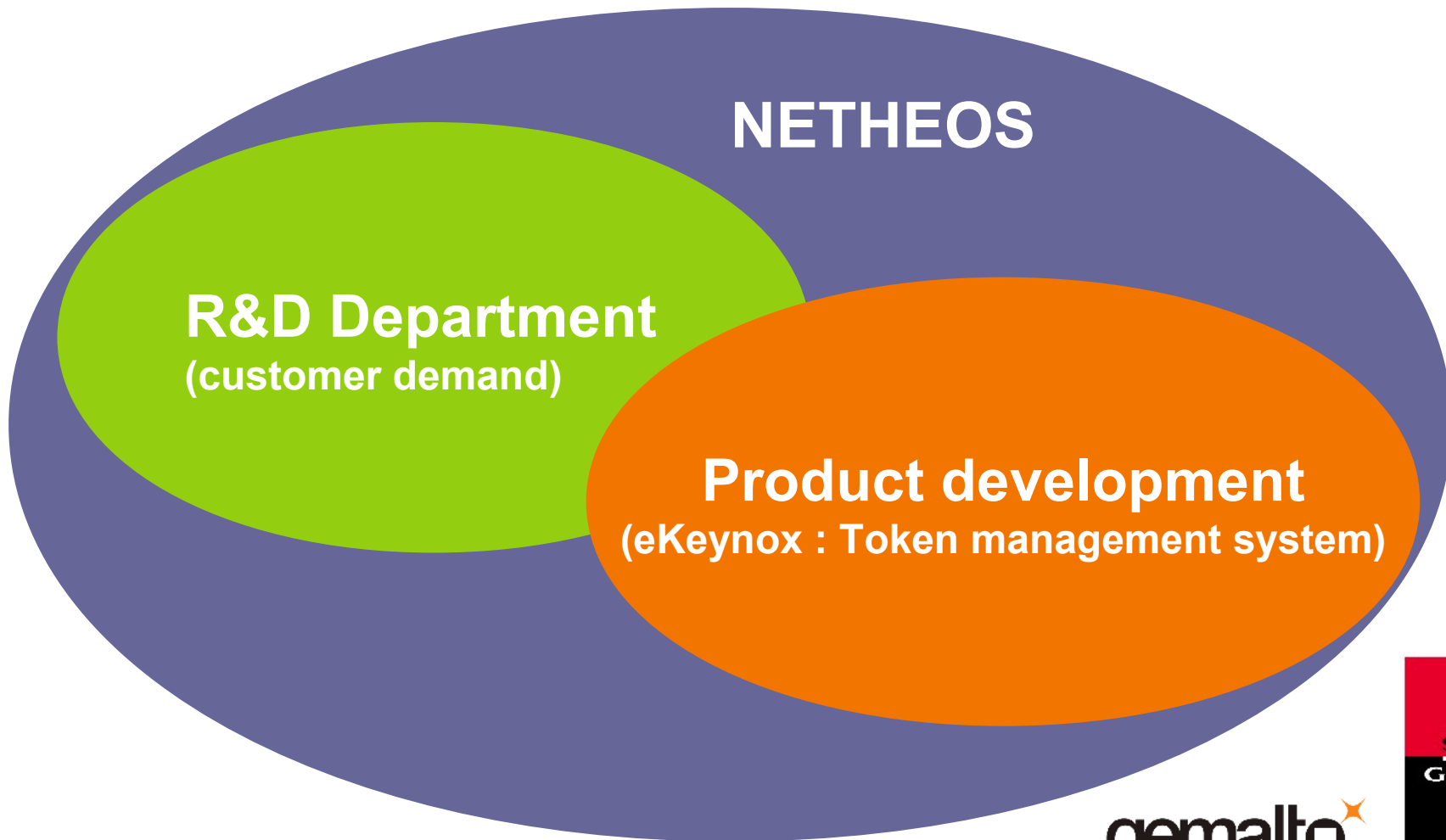
Benoît Badrignans, LIRMM / NETHEOS

Lionel Torres, LIRMM



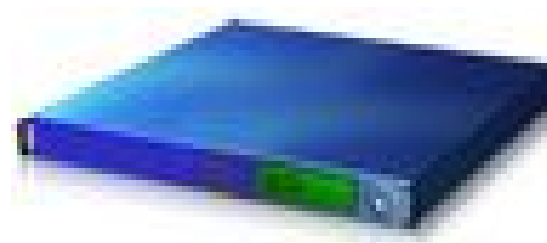
9 employees (Electronics, software and mathematics)

5 years old





+

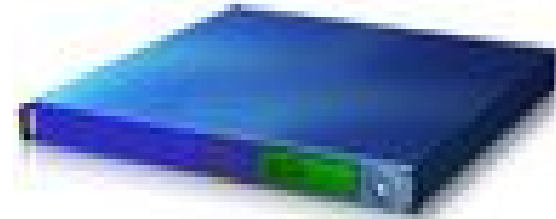


**Software running
on token**

Server



+



**Software running
on token**

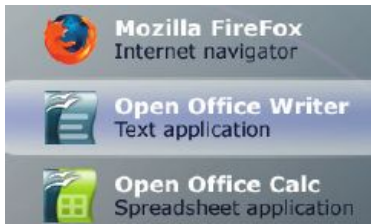
Server



Strong authentication



Private memory



Mobile desktop



Remote management

Skills :

- FPGA based secure products
- Secure micro-controllers
- Cryptography
- PCI Express
- USB

Research program : ANR ICTER

(Confidentiality and integrity of reconfigurable technologies)



Context

State of the art

Proposed solutions

Conclusion / Perspectives

FPGA for secure applications :

Why ?

- Suitable for small and medium markets
- Bitstream remote update (Bugs, vulnerabilities)
- Good performances

Which applications ?

- Personal security devices
- Set-top boxes
- Automotive

FPGA for secure application :

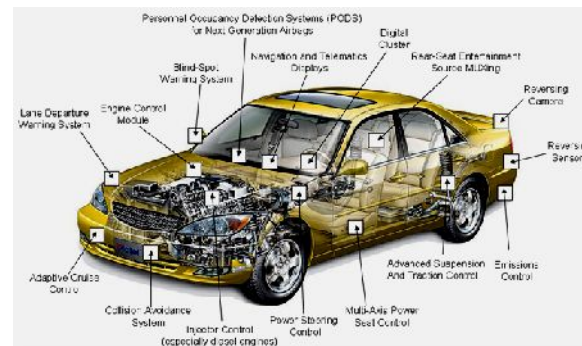
Why ?

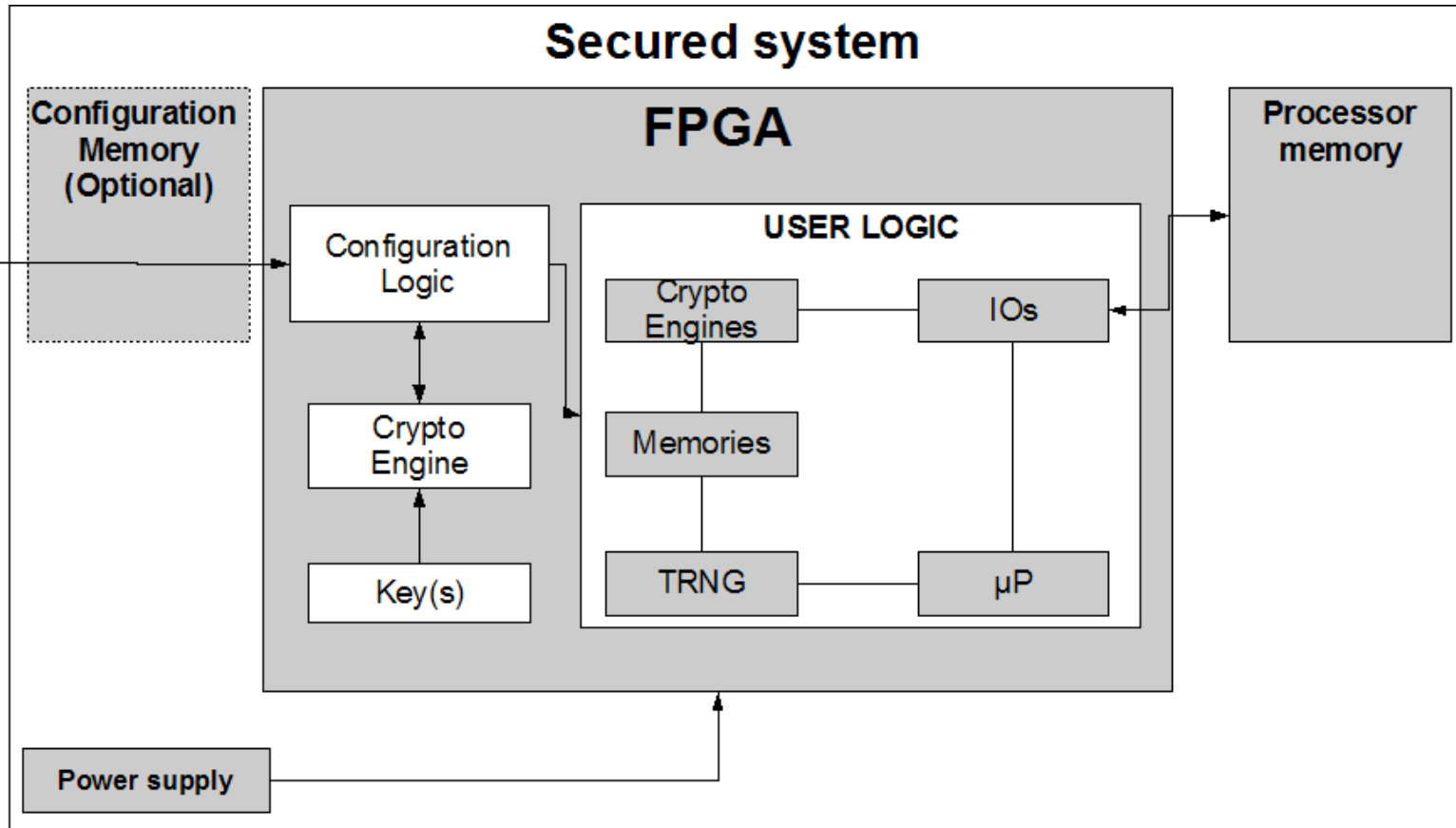
- Suitable for small and medium markets
- Bitstream remote update (Bugs, vulnerabilities)
- Good performances

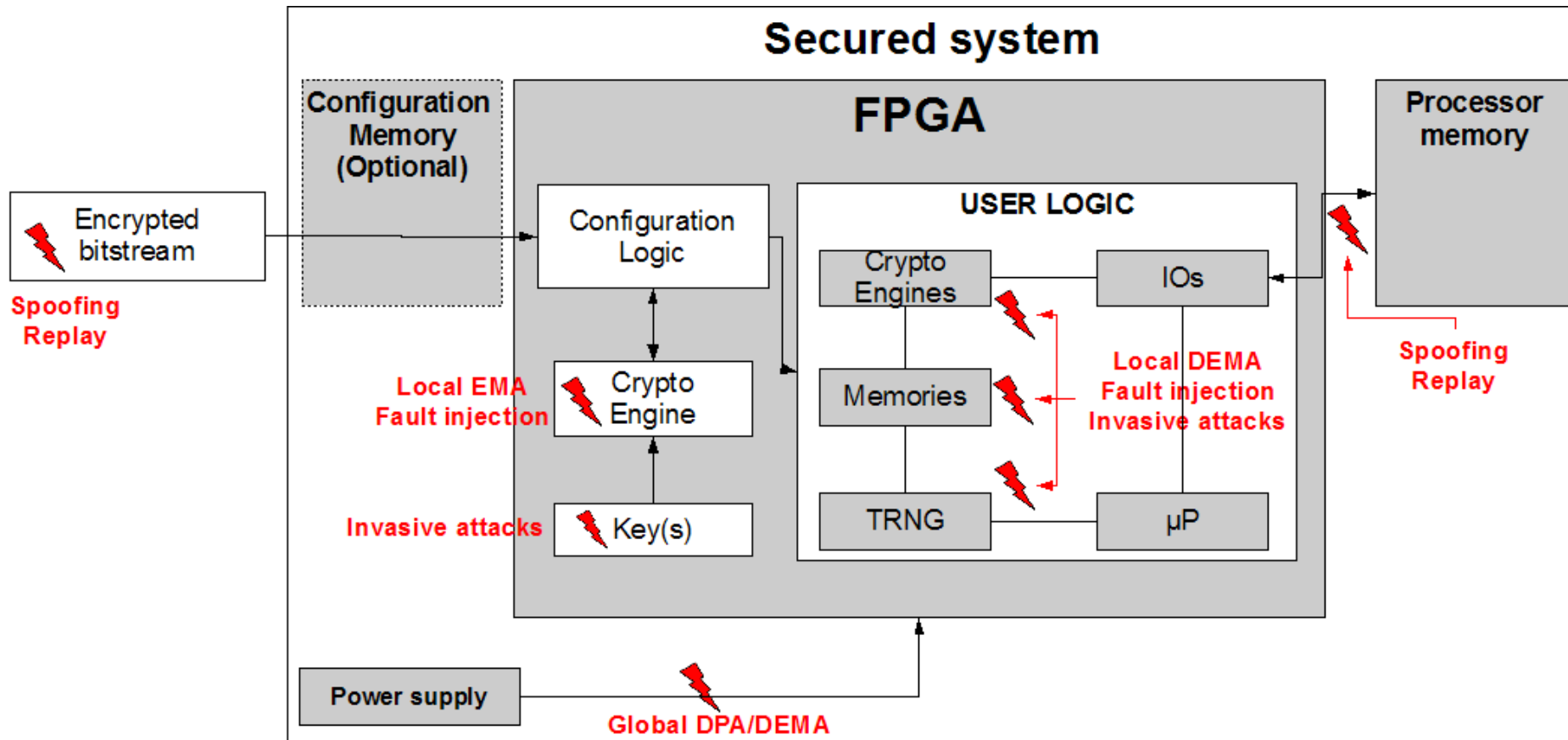


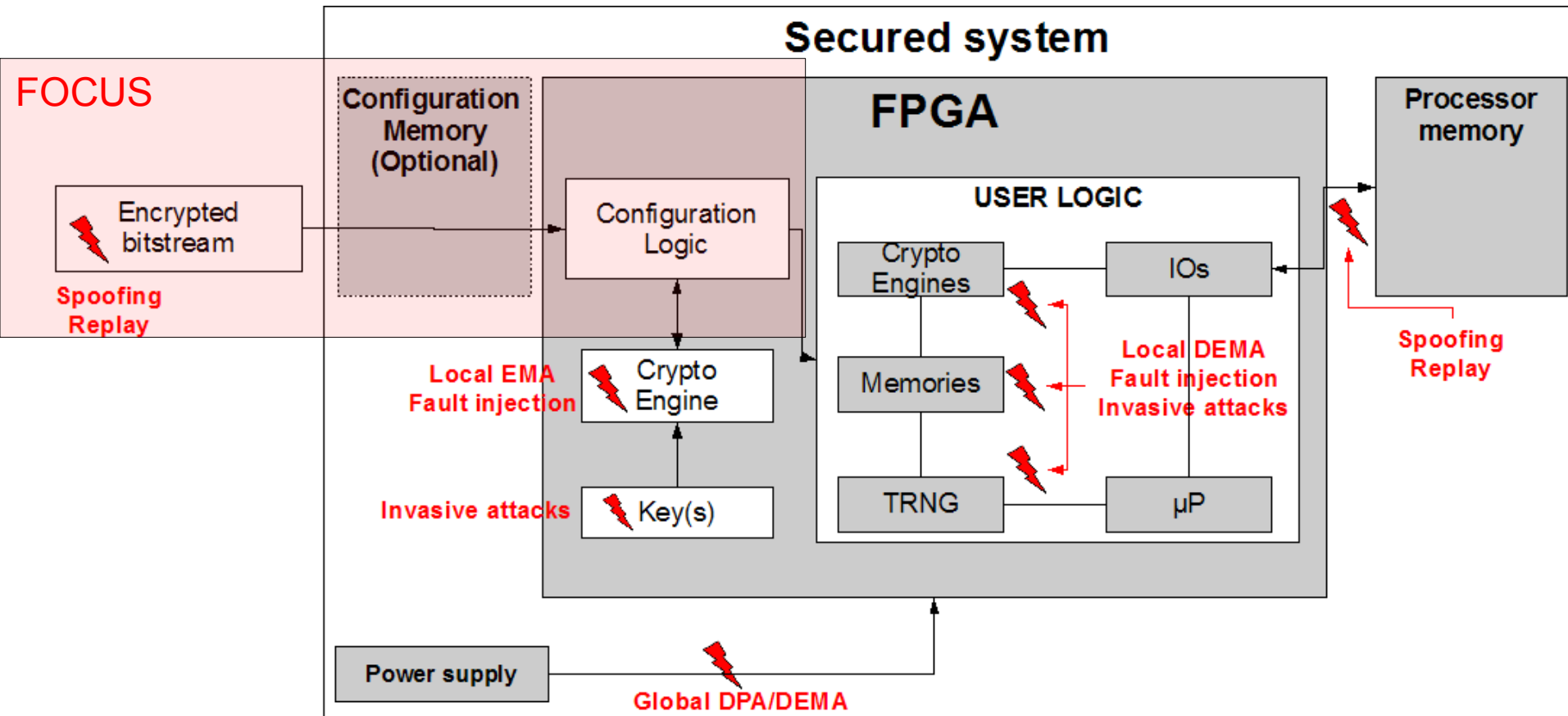
Which applications ?

- Personal security devices
- Set-top boxes
- Automotive









Bitstream importance

- Define hardware behaviour
- Boundary between FPGA vendors and system designer responsibilities
- FPGA vendors have to provide basic security mechanisms (encryption, integrity)

Bitstream security issues

- Copying or cloning
- Reverse engineering
- Tampering (modify hardware behaviour)
- Secure key management

Context

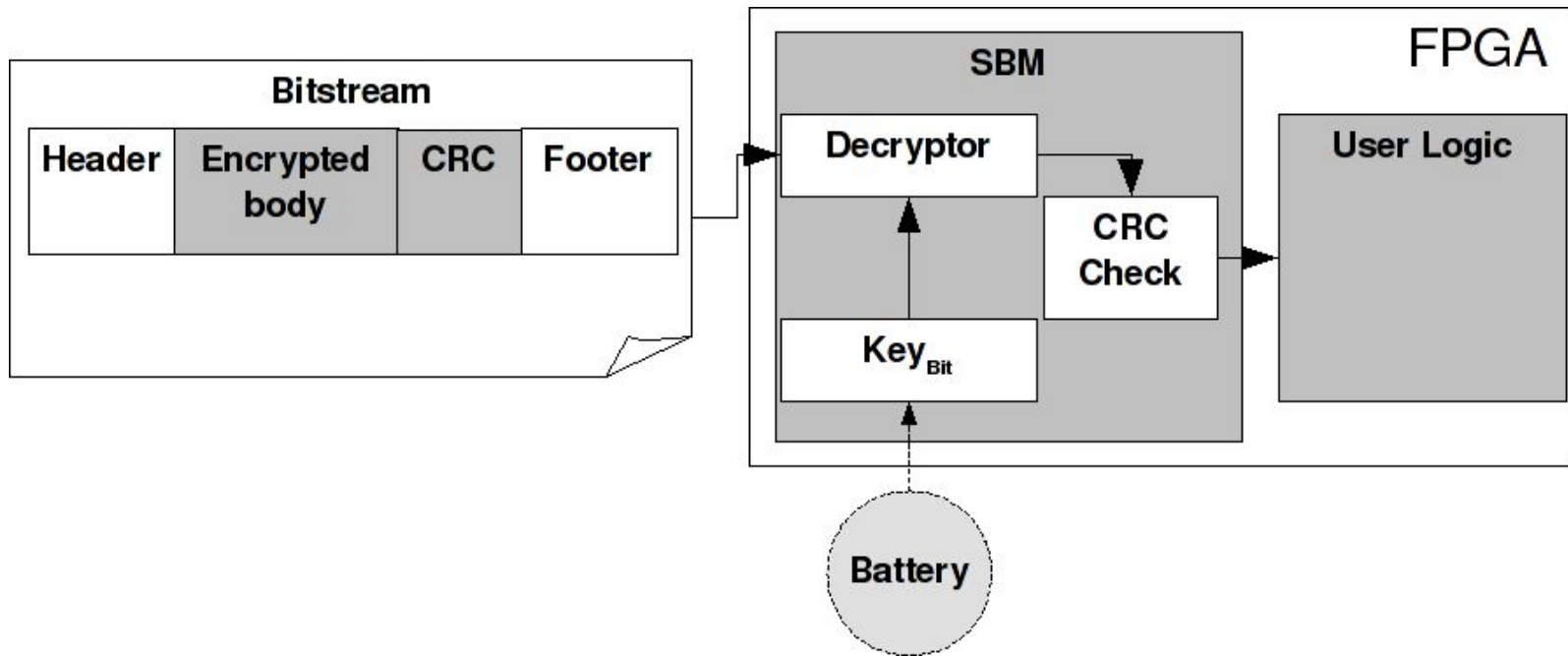
State of the art

Proposed solutions

Application and evaluation

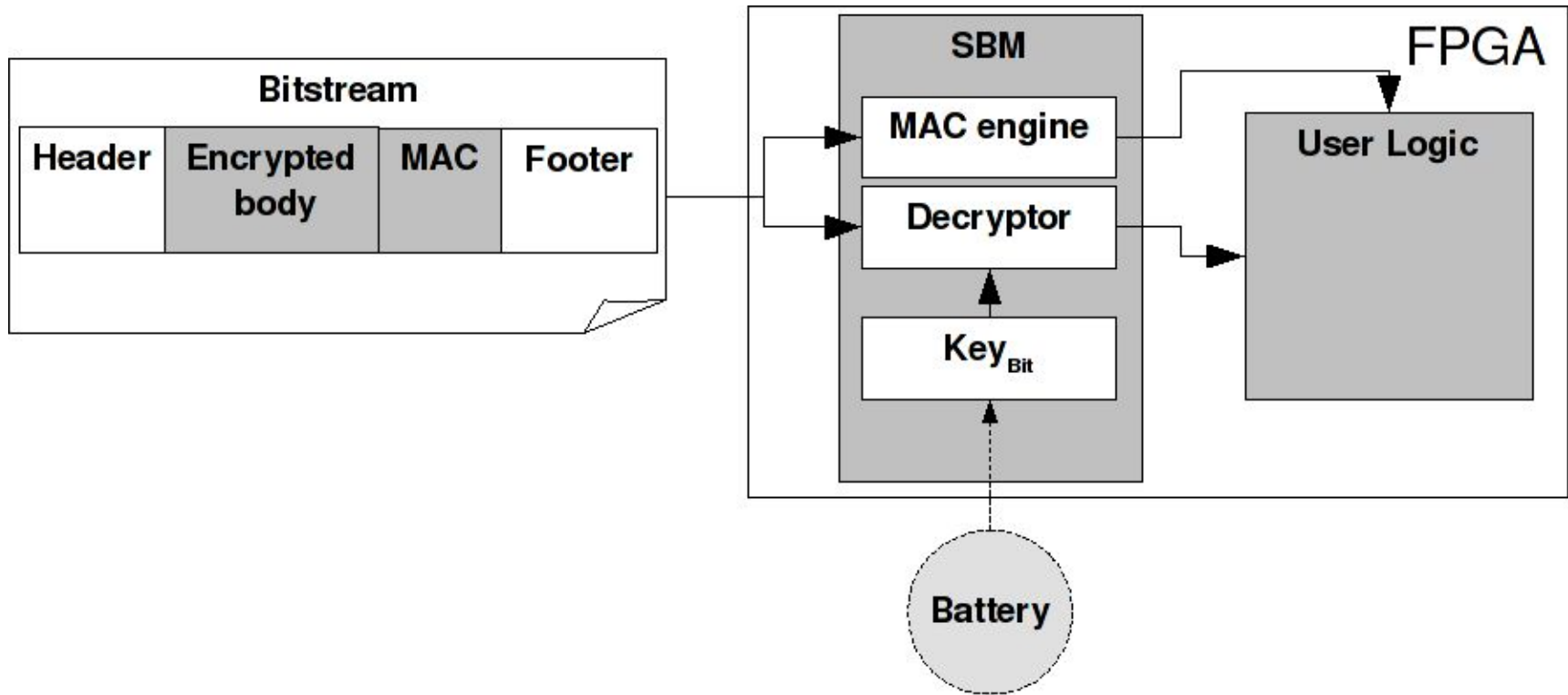
Conclusion / Perspectives

Bitstream encryption



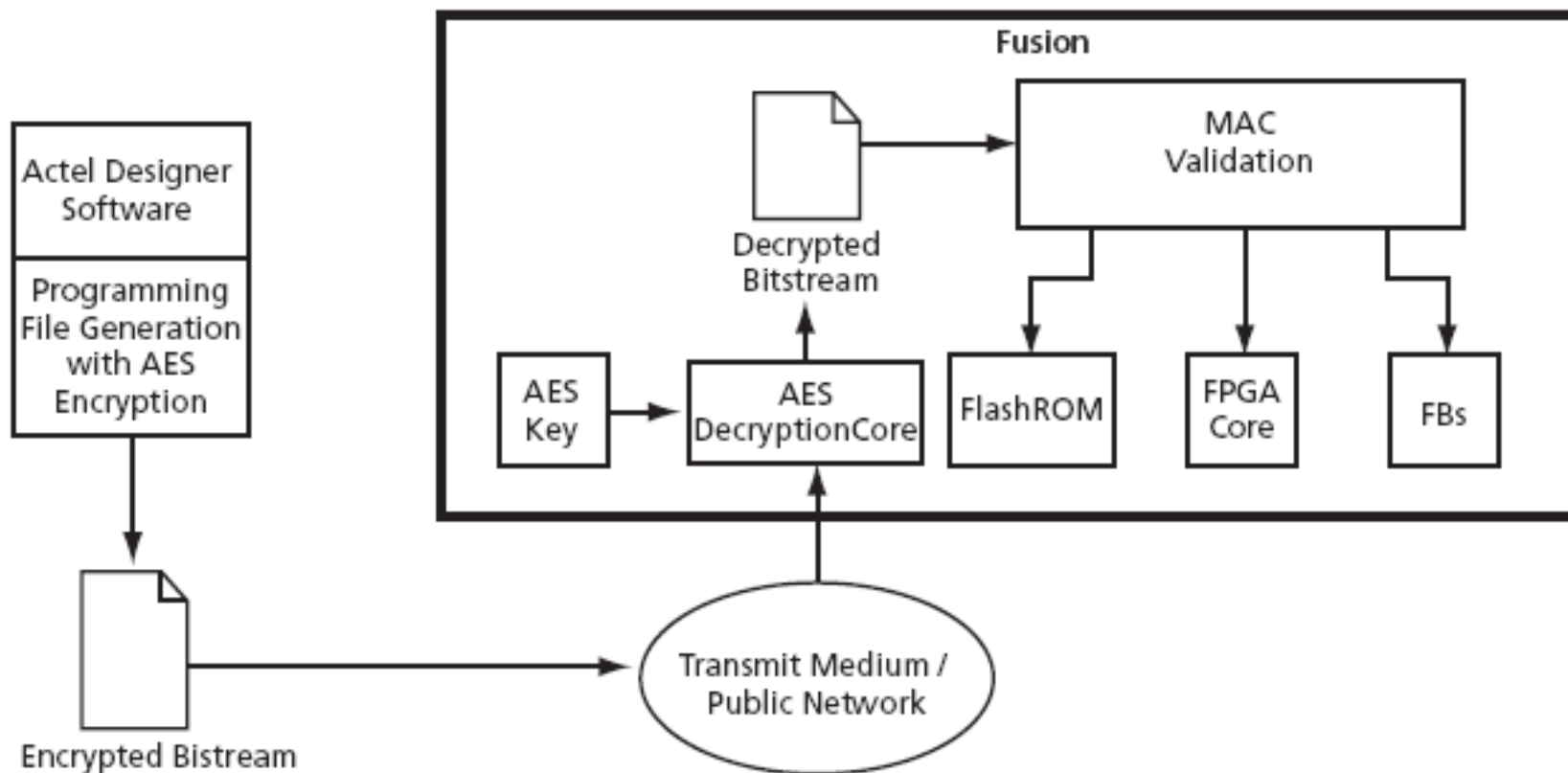
- Mainly used against copying and reverse engineering
- Makes tampering difficult
- Allows designers to hide secrets

Bitstream integrity



- Make tampering “impossible”
- Only ACTEL claims to provide strong integrity mechanism

Bitstream integrity



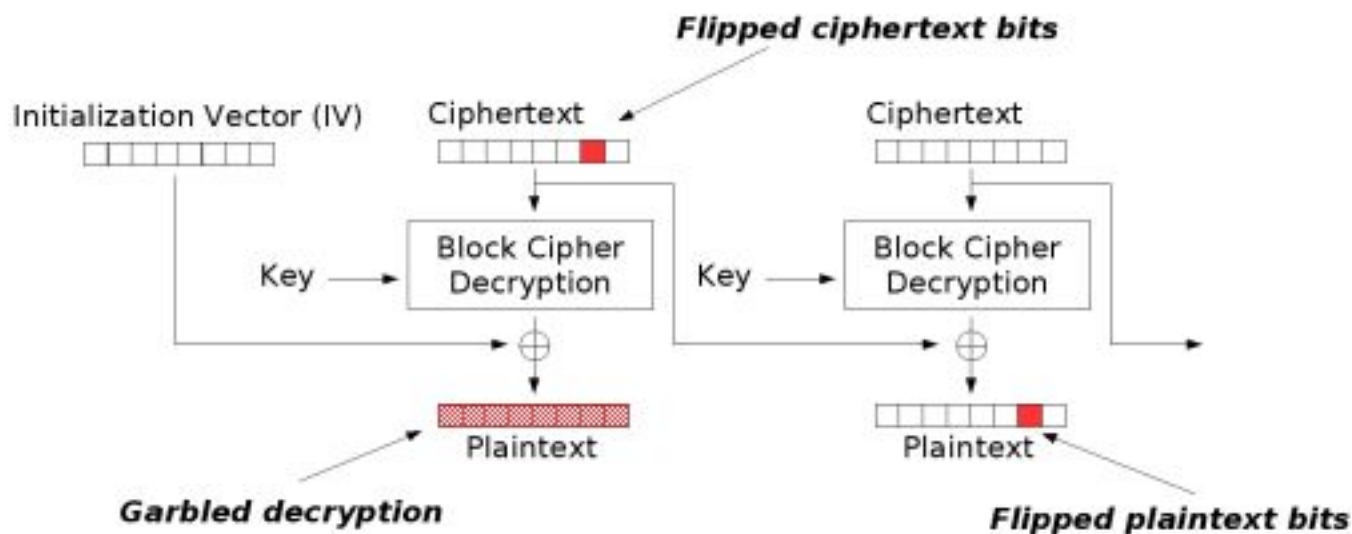
ACTEL ProASIC 3 documentation

Bitstream integrity : attacking CRC

- SRAM FPGAs use 32 bits CRC
- Attacker can perform attacks with only 2^{32} tries

Bitstream integrity : attacking CRC

- SRAM FPGAs use 32 bits CRC
- Attacker can perform attacks with only 2^{32} tries



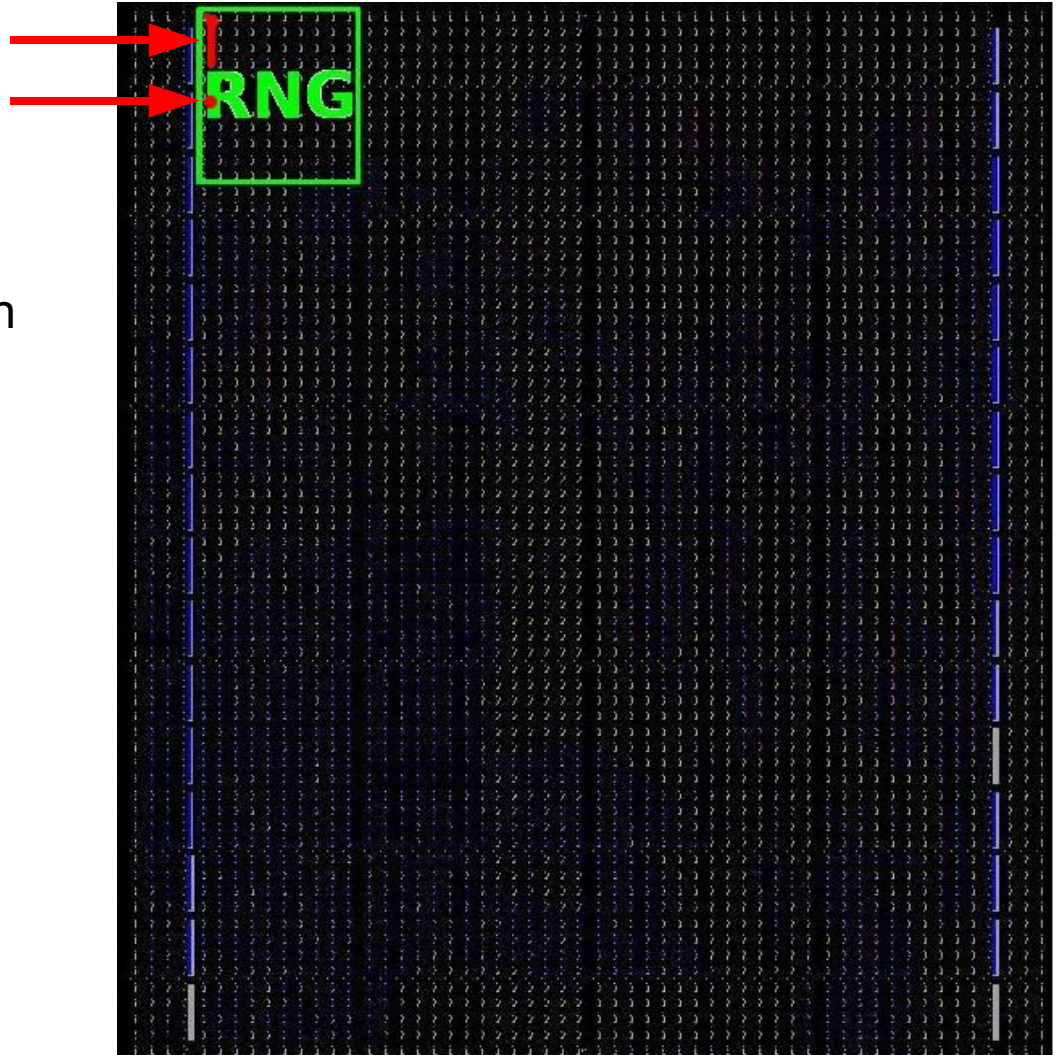
Modification attack or transmission error for CBC

- Attacker cannot finely control its attack for the first block

Bitstream integrity : attacking CRC

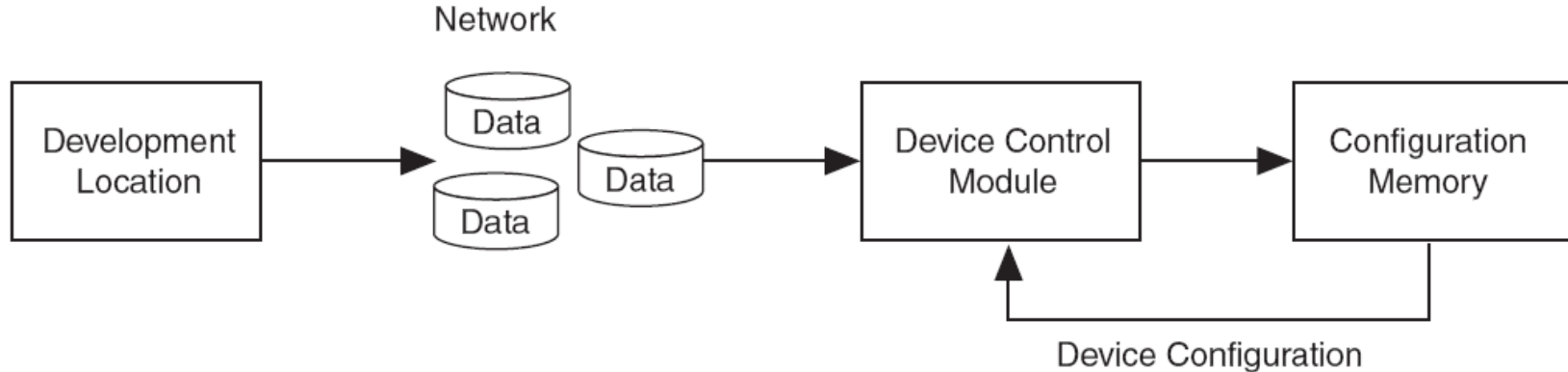
Example :

- Attacker targets a RNG block
- He changes one 128 bits block in the encrypted bitstream
- The system will probably be functional but the RNG don't



Secure remote update

- Remote update is encouraged by FPGA vendors :
 - * Altera IP core : Remote Update Circuitry (ALTREMOTE_UPDATE)
 - * Xilinx white paper : Internet Reconfigurable Logic

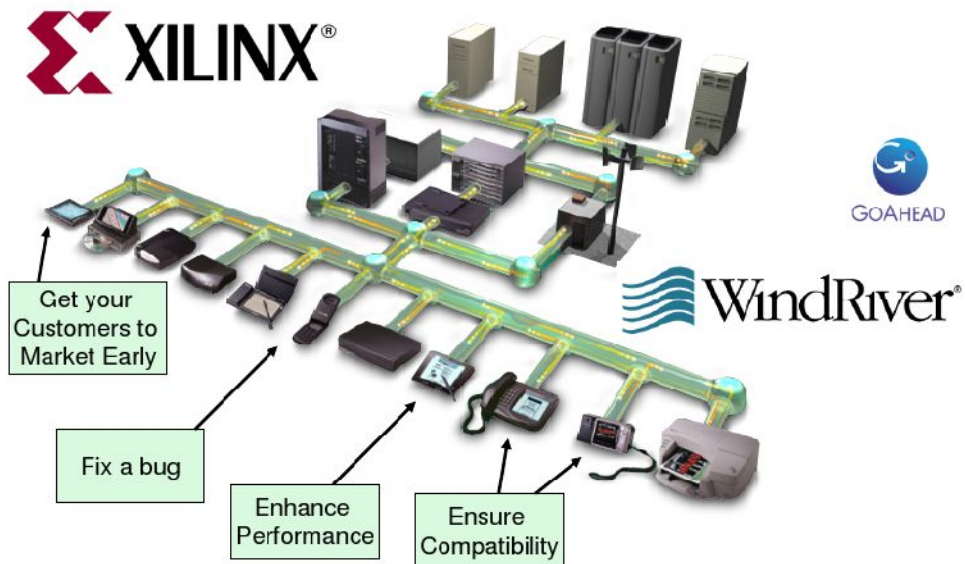


Altera remote update schematic

Secure remote update

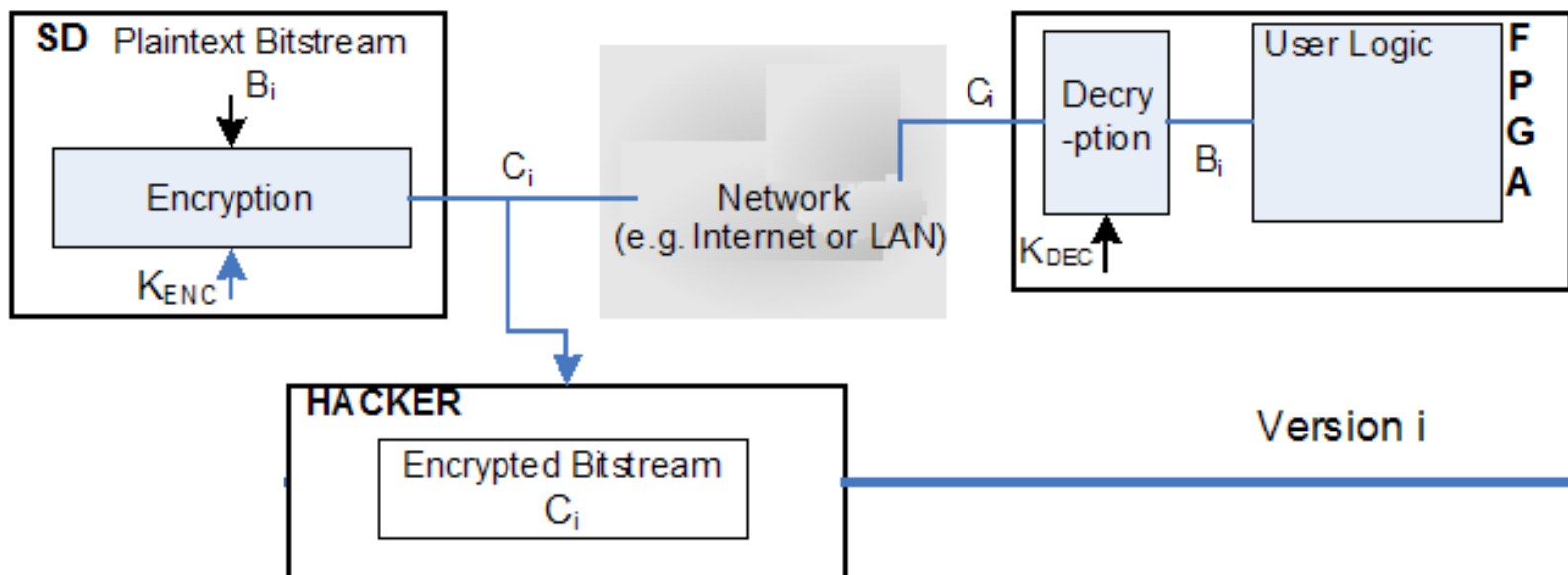
Internet Reconfigurable Logic (IRL)

Remote update of software and hardware

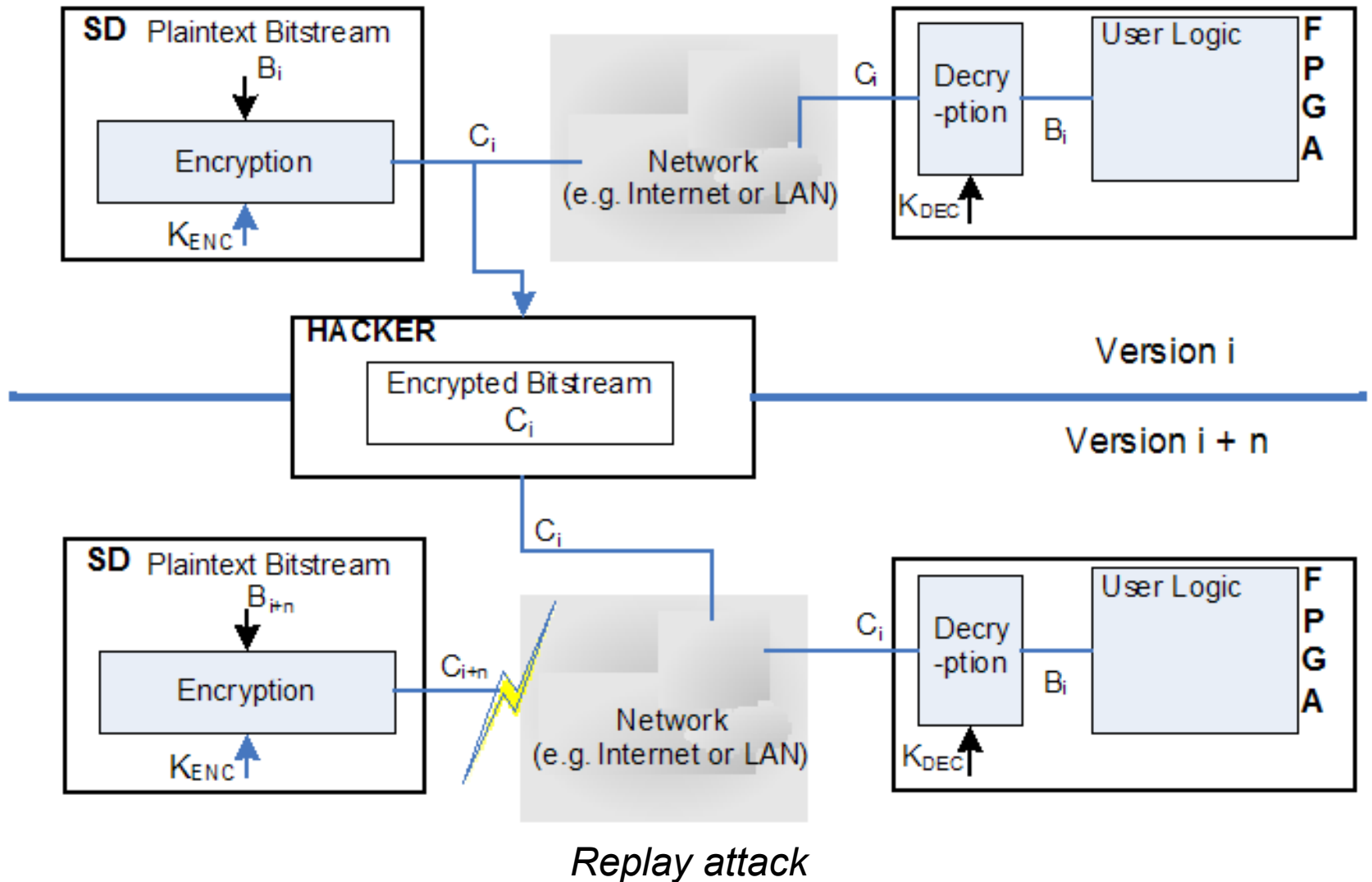


However they do not provide simple solution against replay

Secure remote update

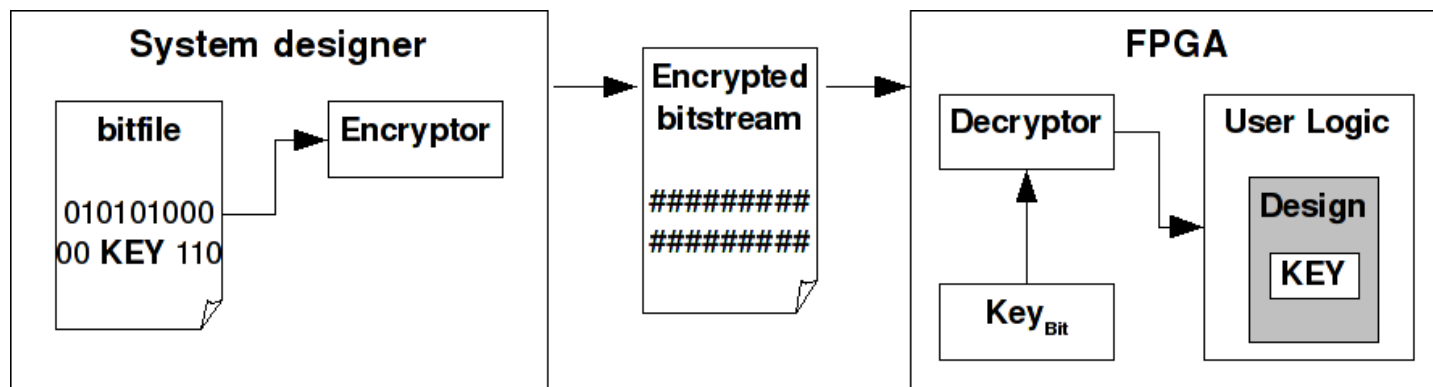


Secure remote update



Key management (SRAM FPGAs)

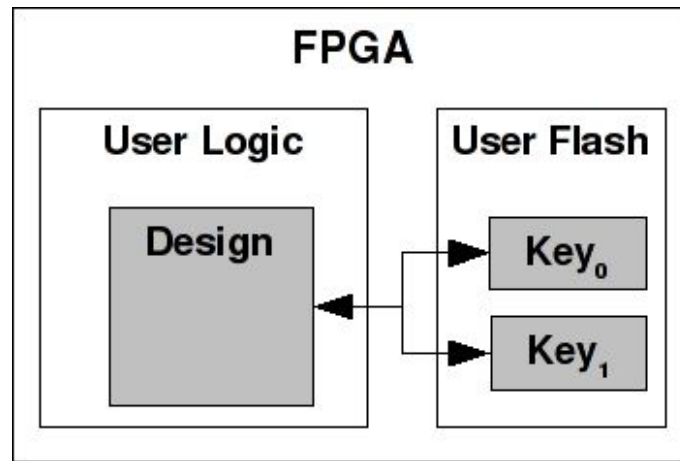
- No special feature for user key management
- System designer can hide keys inside bitstream



Hidden key inside bitstream





- **But** : No simple solution for system owner to personalize its own keys
- **But** : A bitstream per key is needed

Key management (User Flash)



- Key generation can be performed by the system owner
- The same bitstream can be used for each systems
- Non volatile storage allows PIN code implementation
- **But** no solution to erase keys when FPGA is not powered

Secure bitstream management summary

- Bitstream **encryption** is well supported 
- Bitstream **integrity** is generally weak 
- Bitstream **update** management is subject to replay 
- **Key management** is generally not easy 

Context

State of the art

Proposed solutions

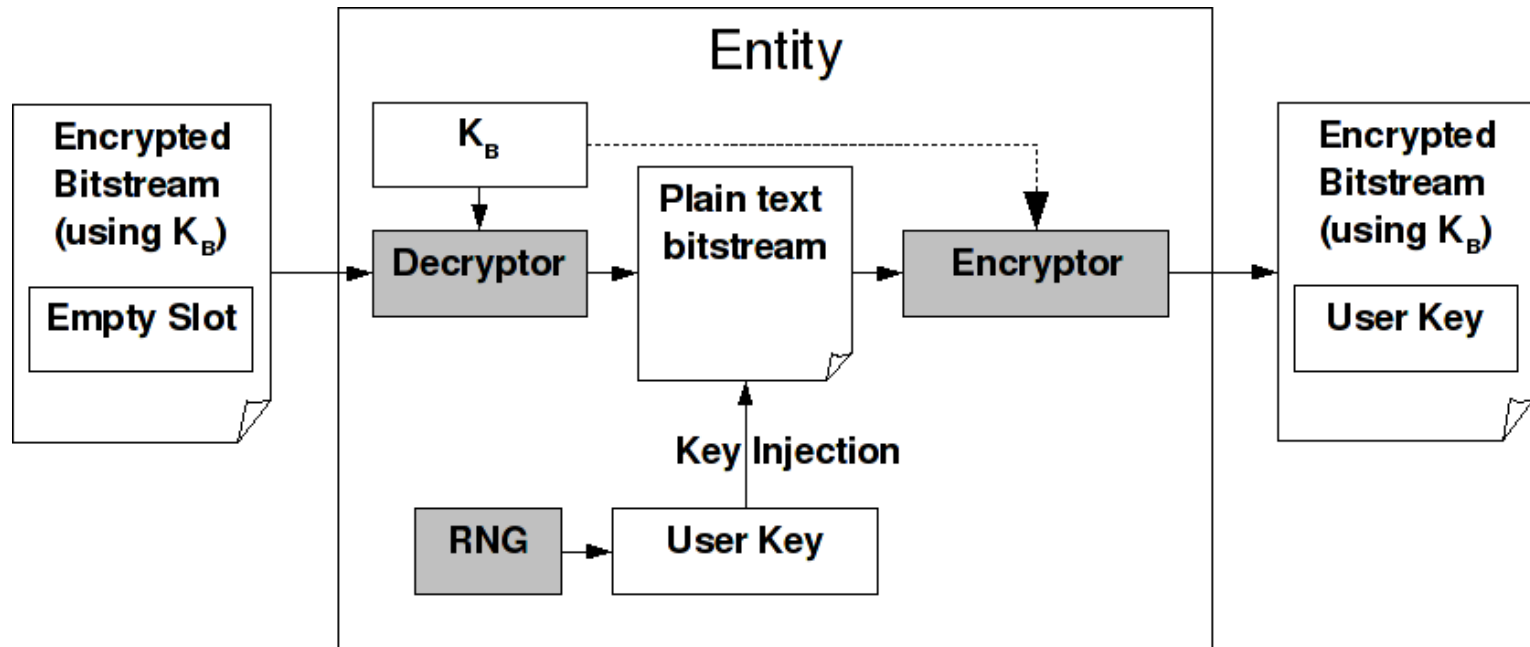
Application and evaluation

Conclusion / Perspectives

Key management (SRAM)



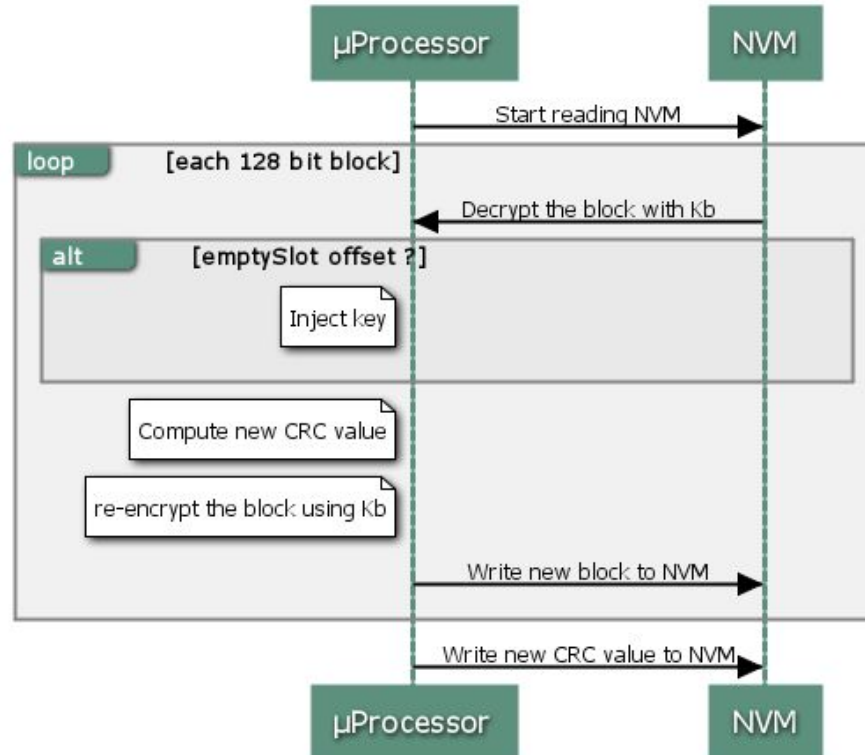
Key management (SRAM)



- An empty slot receives a user generated key
- Provide a solution for on the field key generation
- System designer can provide the same bitstream for each system

Key injection (application on NETHEOS platform)

- The secured μ Processor performs the key injection
- The μ Processor stores the bitstream key K_B in its secured memory

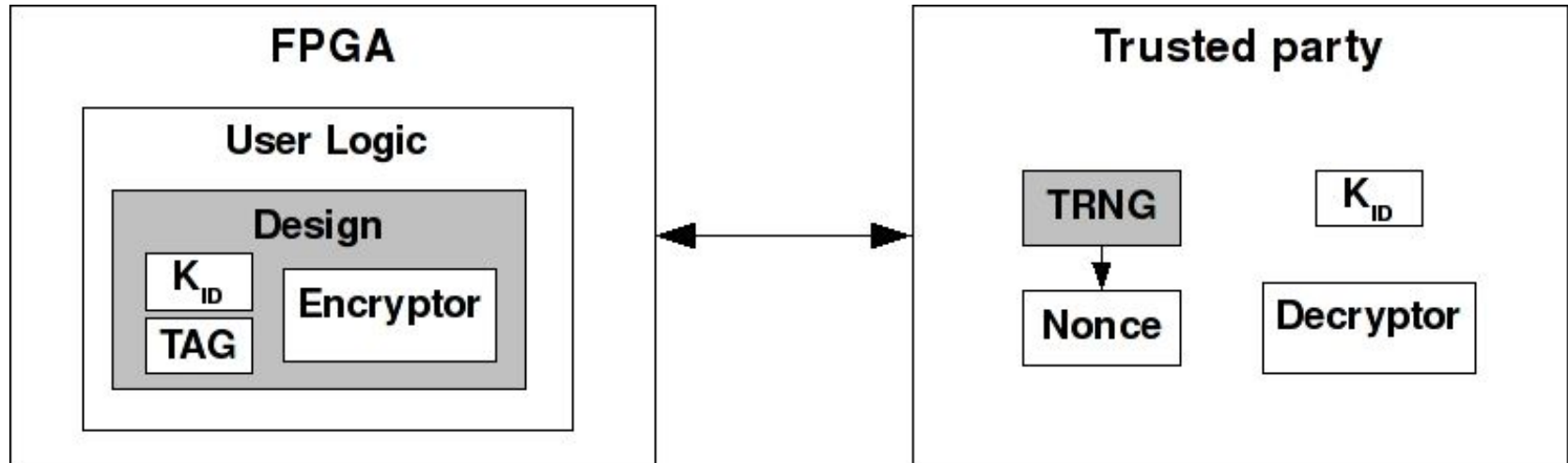


- This process can also be performed by the FPGA itself (not yet implemented)

Secure remote update (solution 1)

Main idea :

- Each bitstream version embeds a unique tag and a unique key (K_{ID})
- An external trusted party attest the current bitstream version

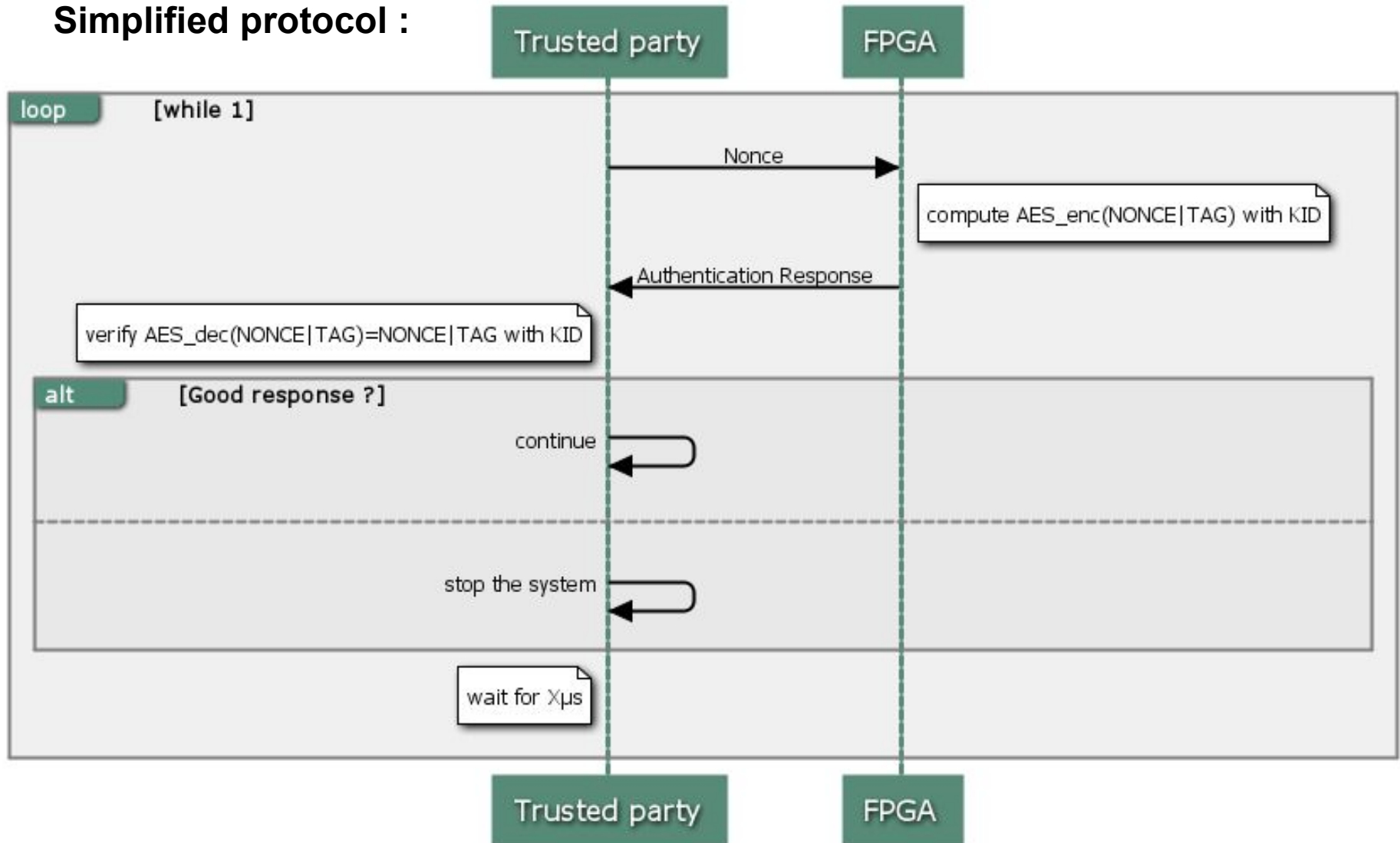


But :

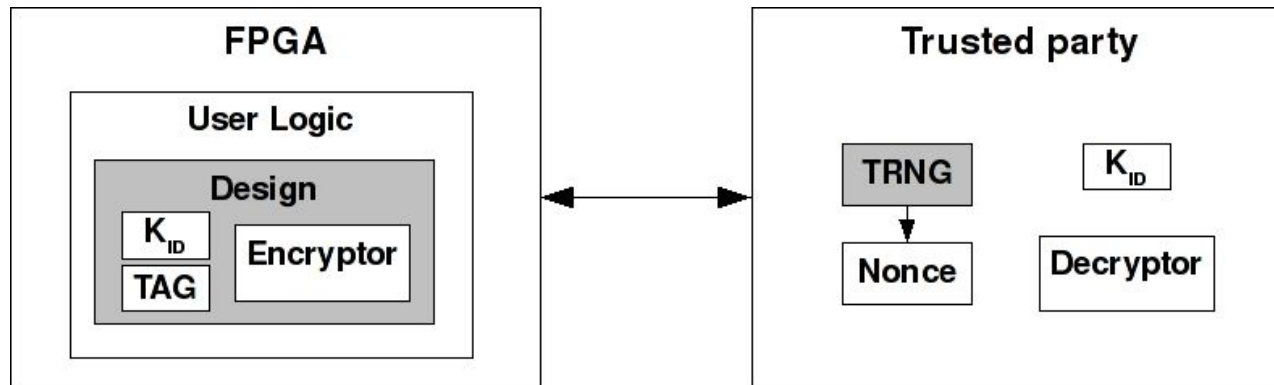
- System designer need to implement an encryptor
- Regular polling is needed
- The problem is reported to the trusted party

Secure remote update (solution 1)

Simplified protocol :



Implementation of secure remote update (solution 1)



Trusted party : secured μ Processor

- The μ Processor embeds a TRNG that generates nonces regularly
- An hardware AES decryptor is used to verify FPGA responses

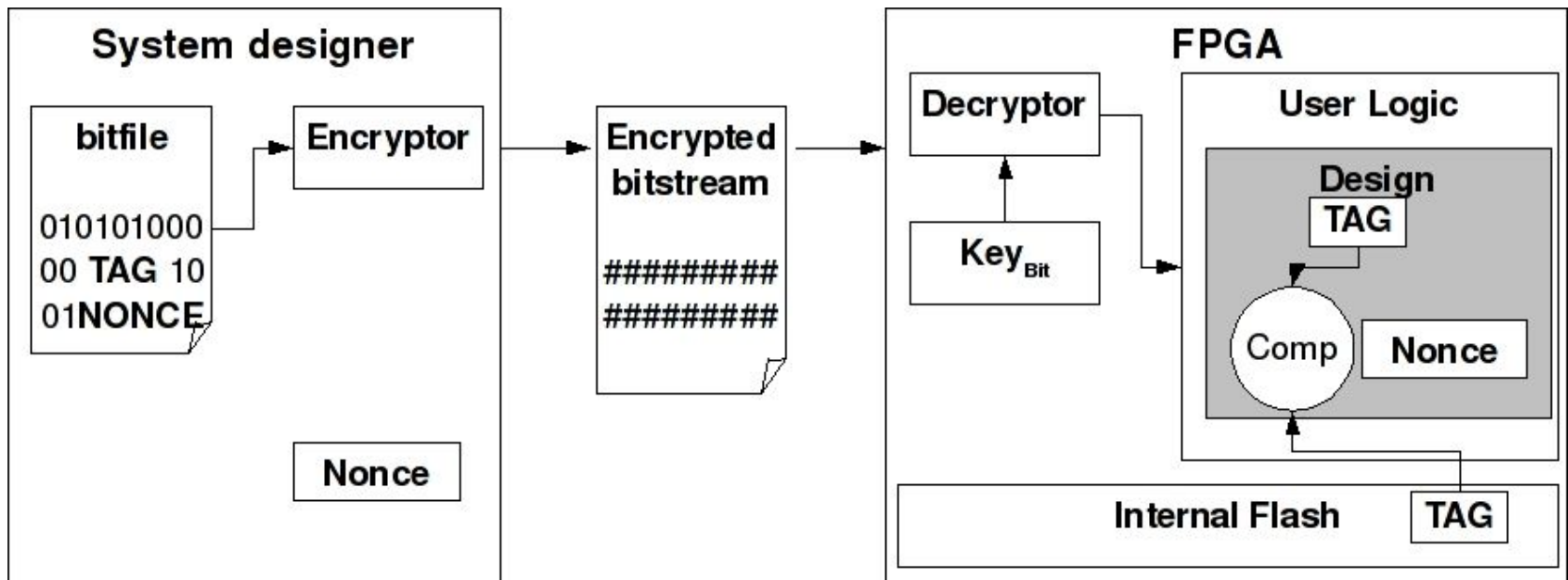
FPGA :

- An hardware AES encryptor is (re)used to generate responses
- The cost is about 800 Slices and 10 RAM blocks
- polling is performed every 10 seconds

Secure remote update (solution 2)

Main idea :

- Each design version embeds a unique tag and a nonce
- Internal Flash stores a reference TAG

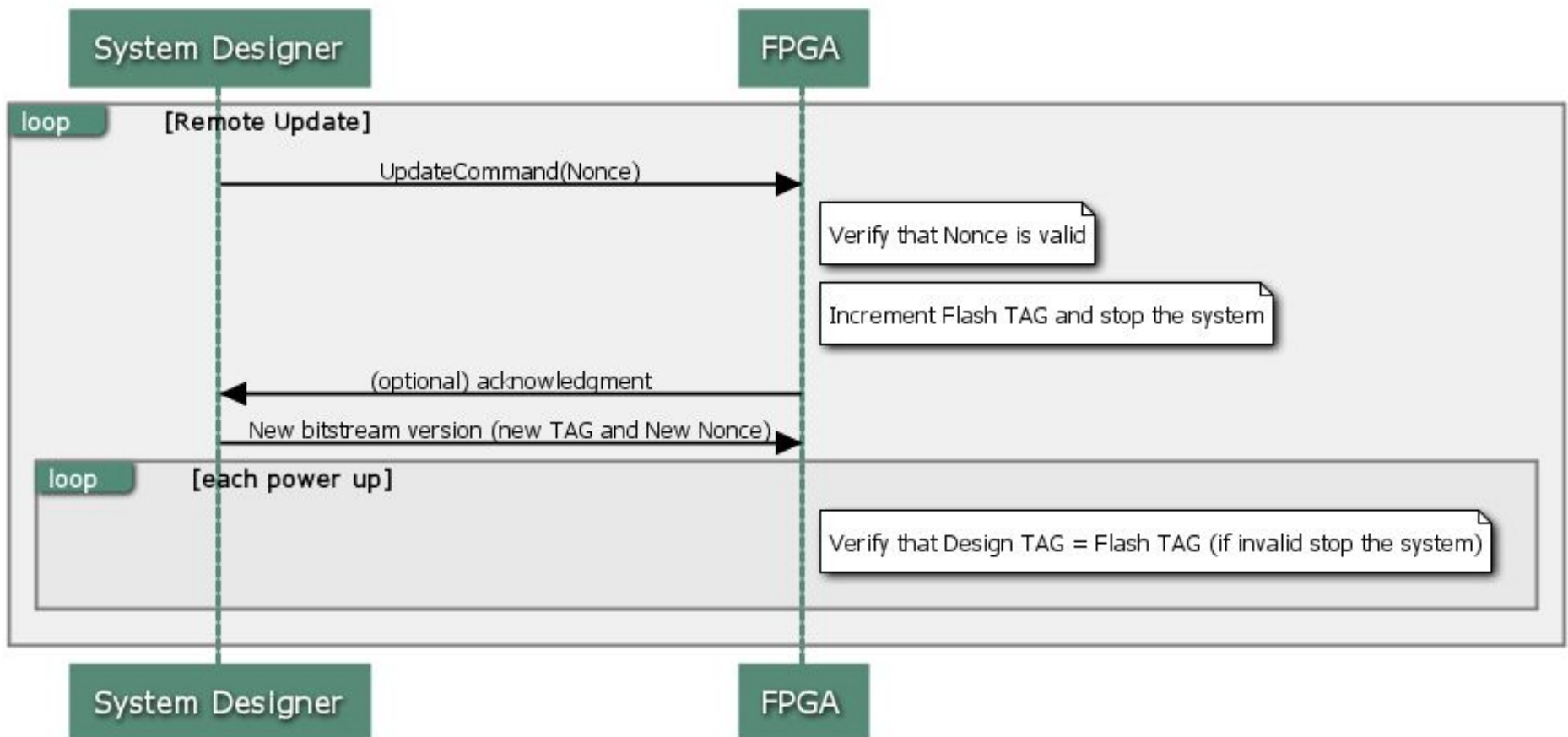


Advantage :

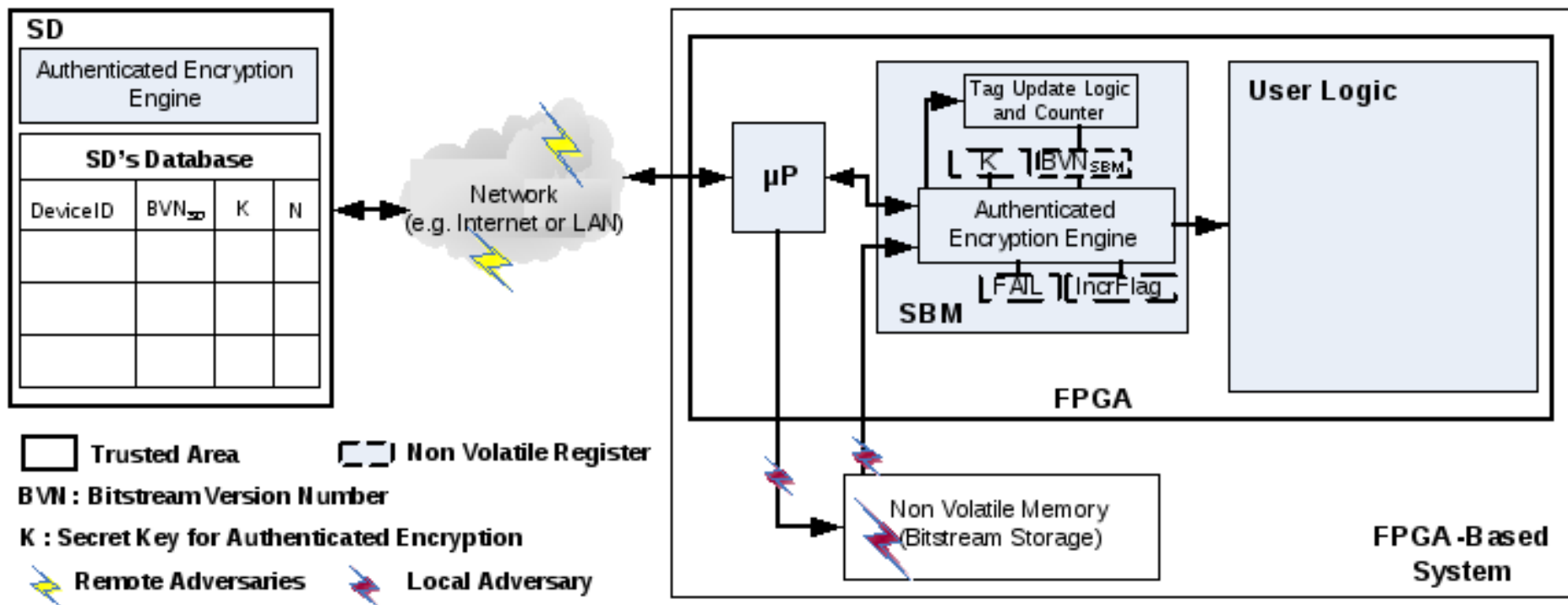
- System designer do not need to implement any cryptographic function
- An other Nonce can be used for acknowledgment

Secure remote update (solution 2)

Simplified protocol :

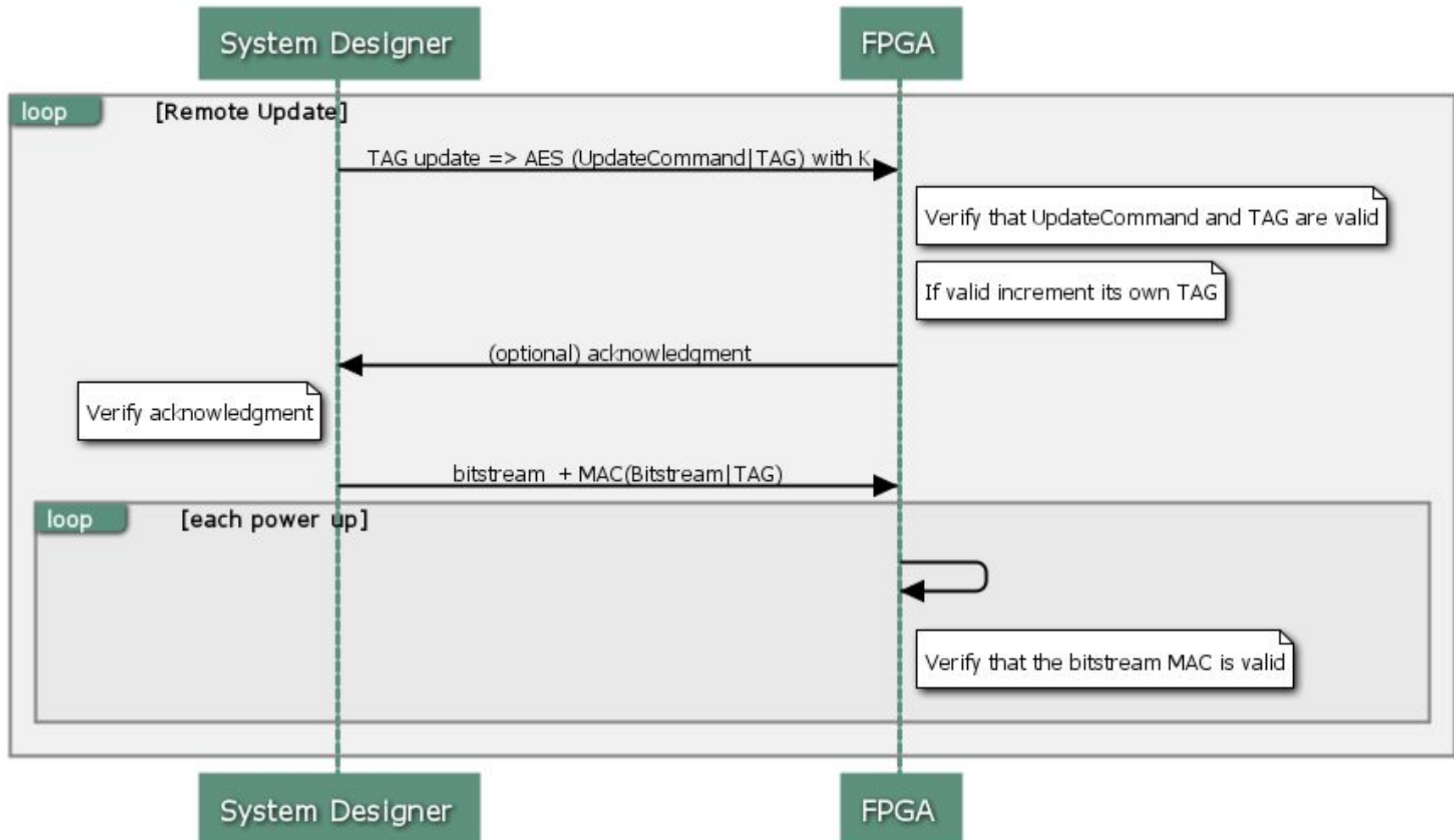


Secure remote update (solution 3)



Secure remote update (solution 3)

Simplified protocol :



Secure remote update (summary)

			Cost for FPGA vendors	Development time for System Designer	Logic gates cost for system designer	Additional cost
Solution	Suitable devices					
1	All encrypted FPGAs		None	High	High	Regular polling
2	ACTEL Fusion		Low for Flash based FPGAs	Medium	Low	None
3	Currently none		Medium	Low	None	None

Context

State of the art

Proposed solutions

Application and evaluation

Conclusion / Perspectives

The perfect FPGA for secured application

- Strong bitstream encryption



- Strong bitstream integrity mechanisms



- Fast bitstream keys deletion (using a battery)

SRAM

- User flash memory for less sensitive keys and certificates

Flash

- Non-volatile reprogram protections (like ACTEL Flash Lock)

Flash

- Downgrade protected (using solution 2 or 3)

Flash

- Battery powered user memories for PIN code implementation and sensitive key storage (unreachable from external I/Os)



Thank you

Benoît Badrignans

Benoit.Badrignans@lirmm.fr

b.badrignans@netheos.net

Lionel Torres

lionel.torres@lirmm.fr

LIRMM / NETHEOS

LIRMM