Lightweight Cryptography

Thomas Peyrin

NTU - Singapore

Cluster of Excellence for Cyber Security (CASA)

Ruhr-Universität Bochum

February 24, 2022

Outline

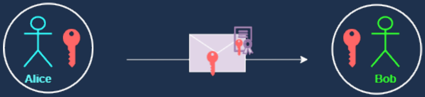
1. Lightweight cryptography : a multi-dimensional problem
2. How to design a (lightweight) cipher ?
3. Lightweight cryptography competitions

What is symmetric/asymmetric-key cryptography ?

Symmetric-key cryptography :

Alice and Bob share the same secret key : Alice sends an encrypted message to Bob using its secret key, Bob deciphers using the same key.

+ usually fast and small !
– about n^2 keys for n users
– need to pre-share the keys



Asymmetric-key cryptography :

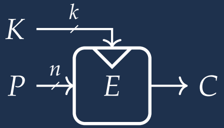
A pair of private/public keys are given to every user. Alice sends an encrypted message to Bob using Bob's public key. Only Bob can decipher using its own private key.

– usually slow and large !
+ $2n$ keys for n users
+ no need to pre-share the keys

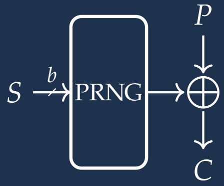
Symmetric-key cryptography for Boomers

Symmetric-key cryptography can be divided into two main groups : block ciphers and stream ciphers
(hash functions like `SHA-2` have no key)

A block cipher (BC) is a family of permutations parametrized by a secret key K (stateless). Used in operating modes to provide encryption, authentication, etc.
Example : `DES`, `AES`



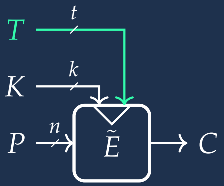
A stream cipher is a pseudo-random generator parametrised by a secret key K and an initial value that generates a keystream to cipher a message (stateful).
Example : `RC4`, `ChaCha20`



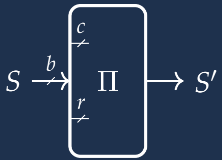
Symmetric-key cryptography for Gen Z

Symmetric-key cryptography is now divided into two main groups : tweakable block ciphers and permutation-based

A tweakable block cipher (TBC) is a family of permutations parametrized by a secret key K and a public **tweak value** T (stateless). Used in operating modes to provide encryption, authentication, etc.
Example : `Deoxys`, `SKINNY`



A permutation on $b = c + r$ bits, with capacity c and rate r, placed in a sponge mode (variable input/output size) to provide encryption, authentication, etc. (stateful).
Example : sponge framework [BDPV-07]



Outline

1. Lightweight cryptography : a multi-dimensional problem
2. How to design a (lightweight) cipher ?
3. Lightweight cryptography competitions

Lightweight cryptography ?

- Computing devices are becoming cheaper and smaller
- **Applications :** RFID tags, vehicle access control, smart cards, medical sensors, wireless sensors, home automation, ...
- Many will be connected to form the Internet of Things (IoT). It is estimated that there will be 50 Billion IoT devices by 2030.

Lightweight cryptography ?

Problem(s) :

- ▷ The Internet is an insecure place
- ▷ These devices are usually operating in physically unsecured environments
- ▷ They are often manipulating data that can be sensible (user data, or critical systems data)

Lightweight cryptography ?

Problem(s) :

- The Internet is an insecure place
- These devices are usually operating in physically unsecured environments
- They are often manipulating data that can be sensible (user data, or critical systems data)

Solution ?

Use cryptography !

Lightweight cryptography ?

Problem(s) :

- The Internet is an insecure place
- These devices are usually operating in physically unsecured environments
- They are often manipulating data that can be sensible (user data, or critical systems data)

~~Solution~~ ?

- Cryptography on these very constrained devices is difficult
- Industry home-brewed solutions led to disasters (Ex : KeeLoq and MiFare)

Lightweight cryptography example : RFID tags

RFID tags are deployed widely (supply chain management, e-passports, contactless applications, etc.)

- we need to ensure authentication and/or confidentiality
- block ciphers are used as basic blocks for RFID device authentication and privacy-preserving protocols
- a basic RFID tag may have a total gate count of anywhere from 1000-15000 gates, with **only 200-4000 gates** budgeted for security

Standard block ciphers/hash functions were not designed with lightweight cryptography in mind (stream ciphers only provide encryption)

- ∼10k gates for `SHA-2`/`SHA-3`
- ∼6k gates for `AES-128` (without mode)
- ∼10/20k gates and ∼million of cycles for ECC multiplication

Lightweight cryptography example : RFID tags

RFID tags are deployed widely (supply chain management, e-passports, contactless applications, etc.)

- we need to ensure authentication and/or confidentiality
- block ciphers are used as basic blocks for RFID device authentication and privacy-preserving protocols
- a basic RFID tag may have a total gate count of anywhere from 1000-15000 gates, with **only 200-4000 gates** budgeted for security

~~Standard block ciphers were not designed with lightweight cryptography in mind~~

Latest `AES-128` implementations only need 1600 GE [JMPS17]

Is `AES-128` a lightweight cipher ?

Is AES-128 a lightweight cipher?

YES! Latest AES-128 implementations only need about 1600 GE

NO! This small implementation requires 1500/2000 cycles! Slow and not energy efficient.

cipher	impl. type	area (GE)	cycles	area*cycles
AES-128	1-bit serial	~1600	~1750	~2800000
AES-128	32-bit serial	~5400	54	~292000
AES-128	round based	~7200	11	~80000
SKINNY-128	1-bit serial	~1300	~7000	~9450000
SKINNY-128	round based	~2400	40	~96000
SKINNY-128	fully unrolled	~32000	1	~32000

What really matters is the flexibility of the cipher to easily offer tradeoffs

Lightweight Cryptography : a multi-dimensional problem

There are many dimensions to consider

for Lightweight Cryptography

Many different platforms

Application-Specific Integrated Circuit (ASIC)

- \+ high-performance
- \+ low power consumption
- − very expensive non-recurring cost
- − one can't change anything once produced
- − time consuming to develop

Bottom-line : for high volume production

Field-Programmable Gate Arrays (FPGA)

- \+ can be reprogrammed
- \+ simple to develop
- − more waste compared to ASIC (higher recurring cost)

Bottom-line : for low volume production

Microcontrollers and ARM

for embedded systems, mobile devices, etc.

Many different platforms : ASIC

ASIC : different cell libraries (depending on the manufacturer)

Library	Logic process	NAND NOR	NOT	XOR XNOR	AND OR	ANDN ORN	NAND3 NOR3	XOR3 XNOR3	MAOI1	MOAI1
UMC	180nm	1.00	0.67	3.00	1.33	1.67	1.33	4.67	2.67	2.00
sxlib	130nm	1.00	0.75	2.25	1.25	1.25	1.25	-	-	-
TSMC	65nm	1.00	0.50	3.00	1.50	1.50	1.50	5.50	2.50	2.50
NanGate	45nm	1.00	0.67	2.00	1.33	-	1.33	-	-	-
NanGate	15nm	1.00	0.75	2.25	1.50	-	1.50	-	-	-

TABLE – Comparisons of several standard cell libraries for typical combinatorial cells. The values are given in GE.

Gate Equivalence (GE) : area of a NAND gate

Many different platforms : FPGA, Microcontrollers and ARM

FPGA

- **Manufacturers :** Xilinx, Altera
- **Lookup table :** 4-input LUT, 6-input LUT, etc.

Microcontrollers and ARM

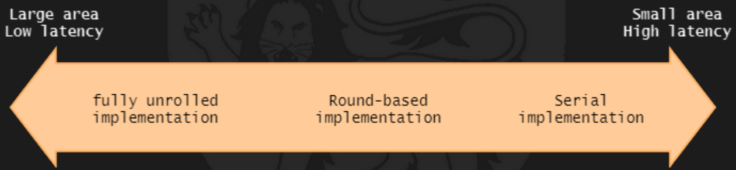
- **Word-size :** 4-bit, 8-bit, 16-bit, 32-bit
- **Memory :** ROM and RAM
- **Instructions set**

Many different implementations

Implementation tradeoffs (from smaller to bigger) :

- **bit-serial** implementation (one bit at a time)
- nibble or **byte-serial** implementation (one Sbox at a time)
- **round-based** implementation (one round at a time)
- **fully unrolled** implementation (entire cipher)

Also implementation tricks (scan flip-flops vs D flip-flops)



For lightweight applications, serial and round-based implementations are the most important

Many different goals

- **Area** (GE in ASIC, slices in FPGA, RAM/ROM on μcontrollers) : especially for very constrained devices, but a criterion to minimize anyway
- **Throughput :** not necessarily a critical aspect, but has to be not too bad
- **Energy :** for battery-powered devices
- **Power :** for passive RFID tags
- **Latency :** for disk encryption, automotive industry, etc.
- Performance for **small messages** is particularly important, for ex. Electronic Product Code (EPC)

For lightweight applications, area/energy/power are generally the most important

Other considerations to make things even more complex!

What about side-channels?

Small devices will likely be easily accessible, so more subject to SCA.

What about software implementations on the server side?

It is likely that many lightweight devices will be communicating with a single server. The cipher has to be efficient on high-end software as well. Bitsliced implementations can help.

In this talk, we will only consider ASICs for simplicity of comparison

Rough numbers to remember :

- a **NAND/NOR gate** : 1 GE
- an **XOR gate** : about 3 GE
- a **2-to-1 multiplexer** on 1 bit : about 2.75 GE
- a **memory bit** : about 6 GE

Outline

1. Lightweight cryptography : a multi-dimensional problem
2. How to design a (lightweight) cipher?
3. Lightweight cryptography competitions

How are ciphers designed

How are ciphers designed ?

How are ciphers designed

How are ciphers designed?

- designing a very secure cipher is easy
- designing a very efficient cipher is easy
- designing a secure **and** efficient cipher is difficult

How are ciphers designed

Two main properties (Shannon 1945) :

- **Diffusion** : make sure that each bit of the state will depend quickly on each bit of the plaintext and the key
- **Confusion** : make sure that the relation between each bit of the state and each bit of the plaintext and the key is very complex

General construction of a block cipher : iterated block ciphers

An iterated block cipher is composed of two parts :

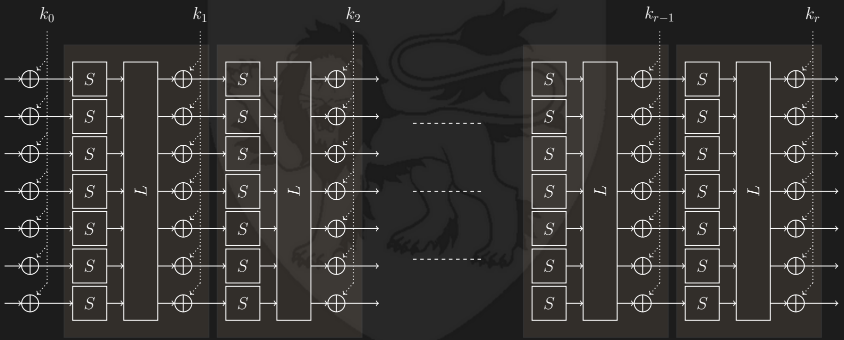
- a key schedule that generates $r+1$ subkeys $K \to (k_0, \ldots, k_r)$
- an internal permutation f repeated r times (also named round function)

An iterative design allows compact implementations (put the round function in a `for` loop) and simplicity of analysis.

General construction of a block cipher : iterated block ciphers

An iterated block cipher is composed of two parts :

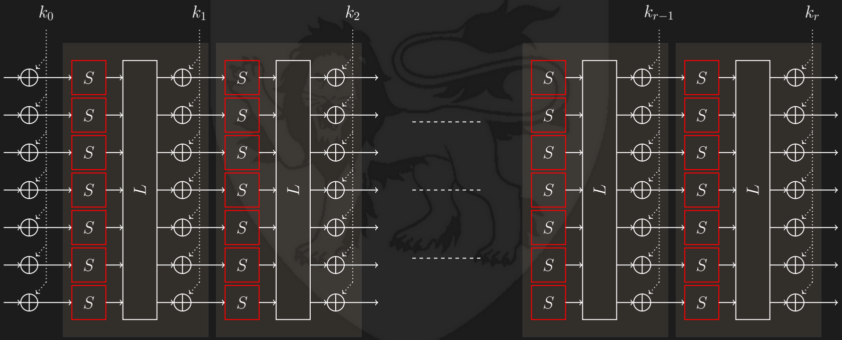
- a key schedule that generates $r+1$ subkeys $K \to (k_0, \ldots, k_r)$
- an internal permutation f repeated r times (also named round function)



General construction of a block cipher : iterated block ciphers

An iterated block cipher is composed of two parts :

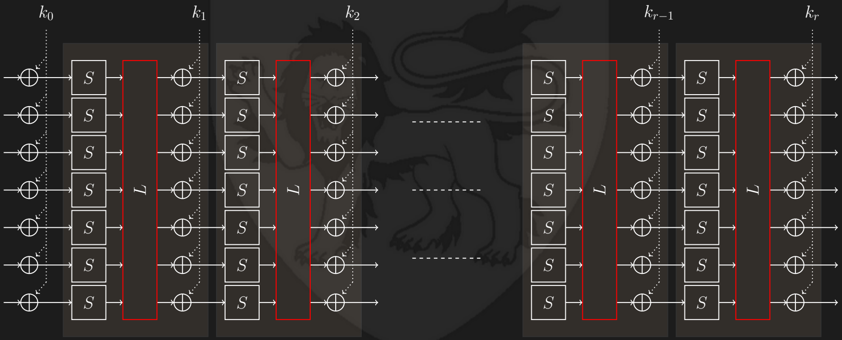
- a key schedule that generates $r+1$ subkeys $K \rightarrow (k_0, \ldots, k_r)$
- an internal permutation f repeated r times (also named round function)



General construction of a block cipher : iterated block ciphers

An iterated block cipher is composed of two parts :

- a key schedule that generates $r+1$ subkeys $K \to (k_0, \ldots, k_r)$
- an internal permutation f repeated r times (also named round function)



How to make a cipher lightweight?

How to make a cipher lightweight?

Lightweight ≃ low memory

The first minimization to aim is to reduce memory usage :
On UMC 180nm :

- one flip-flop for memory : scanFF 6 GE, DFF 4.67 GE
- one XOR gate : 3 GE
- one NAND gate : 1 GE

For lightweight cryptography, block and key sizes will tend to be small in order to avoid any waste of memory because of unwanted extra security
Block-size often 64 bits, key size often 80 bits, which can be problematic (unless your devices are extremely constrained, we should now aim for at least 128-bit block and key sizes).

Some subcomponents might help to reduce the **temporary** memory usage (e.g. recursive diffusion matrices like in LED)

Lightweight ≃ (almost) no key schedule

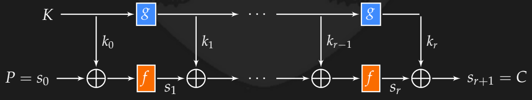
Problem :

The **key schedule** is an important part of a block cipher, and can be quite costly.

Solution :

Just get rid of it ! The current trend is to use no key schedule at all (like in LED) or just permutation of bits (which is basically free on ASICs, but can cost a bit on microcontrollers).
Such key schedule enables hard-wiring of the key when situation allows, which saves a lot of memory.

Careful : several ciphers got broken because of a too light key schedule



Lightweight ≃ small subkeys

Problem :

Incorporating a n-bit **subkey** every round requires n bitwise XORs, which is costly

Solution :

Incorporate smaller subkeys every round, and potentially use more rounds to compensate slower key /state mixing if needed.

Lightweight $\simeq$ LSFR-based constants

Problem :

One needs **constants**, to avoid slide attacks and subspace attacks, especially because cryptographic components will be very light. Constants mean more memory and more XORs.

Solution :

- Use small round-dependent constants (basically a small counter) that are dynamically generated with a very small and lightweight LFSR.
- If needed, use small fixed constants to break symmetry, that can be directly included into other parts of the scheme (for example the Sboxes)

Lightweight $\simeq$ efficient components

Of course, **using lightweight subcomponents** is crucial

Sboxes :

- use small Sboxes (4-bit Sboxes seem a good compromise between size and cryptographic quality)
- use Sboxes that are computed with few cheap operations (AND/OR/XOR)

Diffusion layer :

- use simple bit position permutation (very cheap but provides very little diffusion)
- otherwise, try to minimize the number of XORs needed (binary matrix or cheap coefficients in some finite field)
- serially computable matrices (LED) : lightweight, but slow

Lightweight cipher design approach

The design approach changed :

We used to start from very secure components, and then search for efficient ones in that set.

Now, we are starting from efficient components, and check how many you have to stack to get good-enough security (thanks to the recent improvement of automated tools for cryptanalysis)

Beyond cipher design, **operating modes** also play a very important role (sponges, tweakable block ciphers)

Outline

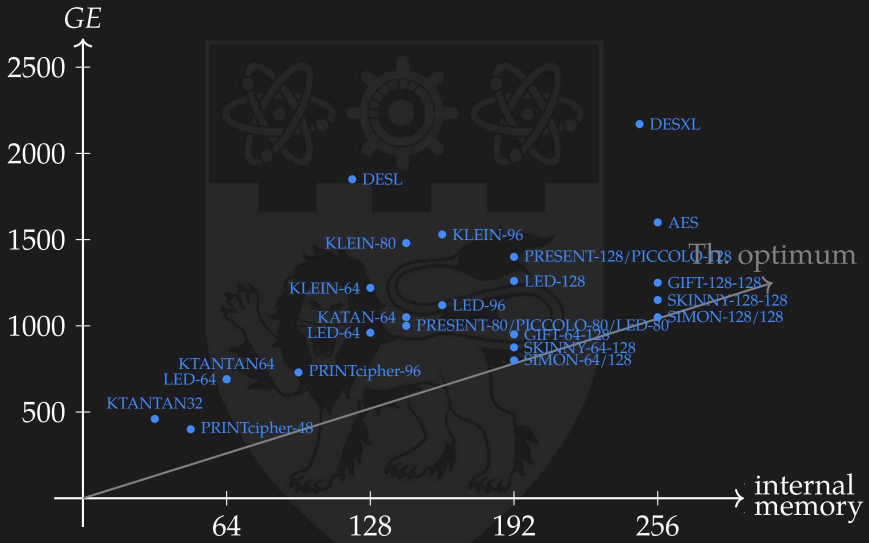
1. Lightweight cryptography : a multi-dimensional problem
2. How to design a (lightweight) cipher ?
3. Lightweight cryptography competitions

Current status of lightweight cryptography

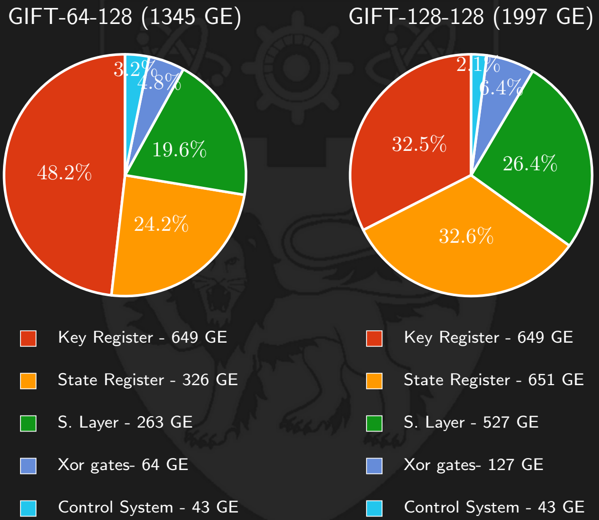
lightweight cryptography has been a very hot topic in the cryptography community in the past 15 years

- really started in 2007 with the proposal of the cipher PRESENT (though some ciphers like NOEKEON [D+00] were already "lightweight")
- a lot of research has been conducted since then, probably more then 50 ciphers have been published
- now comes standardization time (ISO, NIST), while NSA came into play with SIMON and SPECK ciphers

Serial implementations are already close to the theoretical area minimum



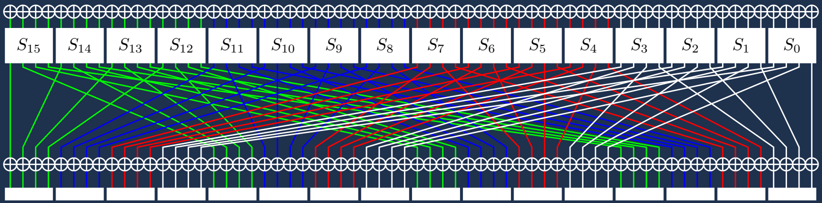
Block Cipher GIFT : round-based implementation



PRESENT [B+07]

The LW block cipher PRESENT was presented at CHES 2007 :

- first cipher to have lightweightness as main goal
- 31-round SPN block cipher with 64-bit block size
- **very simple design :** Sbox layer and bit permutation only (bit permutation is free on ASIC).
- selected in 2012 as ISO standard (ISO/IEC 29192-2)



PRESENT round

PHOTON and LED [G+11]

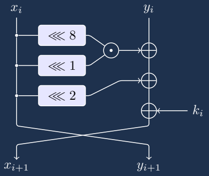
The LW block cipher LED and hash function PHOTON (CHES/CRYPTO 2011) :

- **main idea :** minimize the temporary memory and break area records for serial implementations, with new diffusion matrices. No key schedule at all for LED.
- potentially less interesting for round-based implementation
- a few NIST LWC candidates based on PHOTON (1 finalist)
- PHOTON selected in 2016 as ISO standard (ISO/IEC 29192-5)

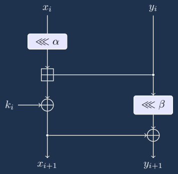
NSA's SIMON and SPECK [B+13]

The NSA's SIMON and SPECK LW block ciphers (ePrint 2013) :

- separates hardware oriented (SIMON) and software oriented (SPECK)
- very simple and efficient ciphers
- no security analysis provided, but a lot of third party analysis
- failed to become an ISO standard (backlash from the crypto community)



SIMON round

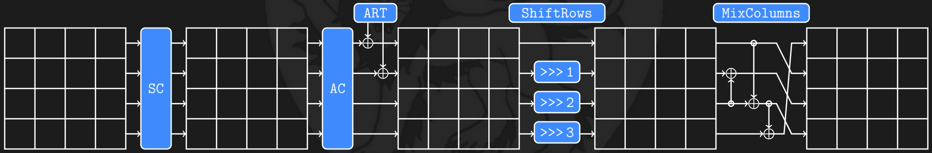


SPECK round

SKINNY [B+16]

The LW tweakable block cipher SKINNY (CRYPTO 2016) :

- **main idea :** provide an alternative to SIMON for hardware
- with better security guarantees for the same efficiency
- with tweak capability
- a few NIST LWC candidates based on SKINNY (1 finalist)
- SKINNY undergoing ISO standardisation (ISO/IEC 18033-7)

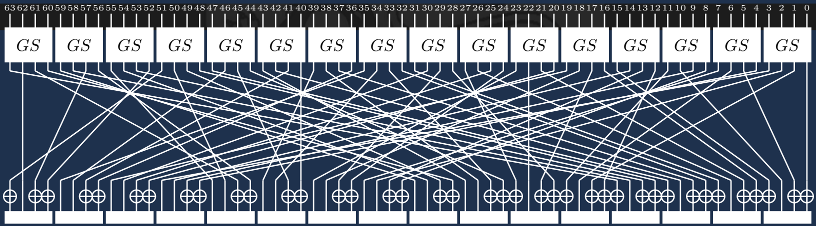


SKINNY round

GIFT [B+17]

The LW block cipher GIFT was presented at CHES 2017 :

- **"improved" version of** PRESENT, includes a 128-bit block version
- more efficient (smaller, faster) than PRESENT
- better resistance against differential/linear cryptanalysis
- several NIST LWC candidates based on GIFT (1 finalist)



GIFT round

Authenticated Encryption

Problem(s) :

- In most applications, what you want is confidentiality **AND** authentication **at the same time**
- there are a lot of unsecure ways to combine a secure encryption primitive and a secure authentication primitive
- using a single primitive doing both encryption AND authentication at the same time might be more efficient than using two separate ones

Authenticated Encryption = Authentication + Encryption

The CAESAR Competition

CAESAR Competition :

- CAESAR - Competition for **Authenticated Encryption** : Security, Applicability, and Robustness
- 5-year competition (from 2014 to 2019) organised by the community
- 57 submissions from all over the world
- several rounds to prune candidates
- a lot of cryptanalysis conducted, schemes getting broken, performance measured, etc.
- `https://competitions.cr.yp.to/caesar.html`

The CAESAR Competition results

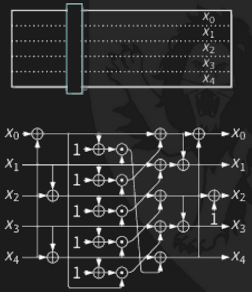
CAESAR Competition final portfolio :

- portfolio "**Lightweight applications**" :
  ASCON (1st), ACORN (2nd)
- portfolio "**High-performance applications**" :
  AEGIS and OCB (1st ex-aequo)
- portfolio "**Defense in depth**" :
  Deoxys-II (1st), COLM (2nd)

ASCON [D+14]

The LW AEAD ASCON (CAESAR 2014) :

- based on a sponge-like operating mode
- the round function of the permutation is composed of a very light Sbox and a linear layer easy to compute on software
- a few NIST LWC candidates based on ASCON (2 finalists)



$$x_0 := x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28)$$
$$x_1 := x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39)$$
$$x_2 := x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6)$$
$$x_3 := x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17)$$
$$x_4 := x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41)$$

ASCON round

The NIST LWC Competition

NIST LWC Competition :

- NIST LWC competition
  for **Authenticated Encryption** and **Hashing**
- started the 29th of March 2019, final decision for summer/end 2022 ?
- 57 submissions from all over the world
- several rounds to prune candidates (currently final stage)
- a lot of cryptanalysis conducted, schemes getting broken, performance measured, etc.
- `https://csrc.nist.gov/Projects/lightweight-cryptography`

NIST

NIST Lightweight cryptography competition

The selection of the winner(s) :

- must perform better than AES-GCM
- hashing is optional
- side-channels can also matter
- well analysed/established candidates are favored
- should they go for **one do-it-all candidate** or **2 candidates (HW/SW)** ?

It is important to push for trusted designs at NIST and ISO, to avoid issues with governmental agency-based proposals (NSA Dual EC DRBG, Russian Kuznyechik)

NIST Lightweight cryptography competition

The 10 finalists of the ongoing NIST competition

name	type	internal	SECURITY: sec. marg. internal	SECURITY: data. sec. claims	PERFORMANCES: HW	PERFORMANCES: SW	FEATURES: hash	FEATURES: side-chan. resistance	FEATURES: other
ASCON	perm.	ASCON-p	none	birthday	good	good	✓	some	
ELEPHANT	perm.	SPONGENT	?	birthday	mid	bad			parallel
GIFT-COFB	BC	GIFT	large	birthday	good	good			
Grain-128AEAD	SC	Grain	?	full	good	mid			heavy init.
ISAP	perm.	ASCON-p	none	full	bad	bad		yes	
PHOTON-Beetle	perm.	PHOTON	small	full	mid	bad	✓		ISO
Romulus	TBC	SKINNY	large	full	good	mid	✓	1 mode	misuse res. ISO
SPARKLE	perm.	ad-hoc	?	full	mid	good	✓		
TinyJambu	perm.	ad-hoc	none	birthday	good	good			
Xoodyak	perm.	Xoodoo	none	full	good	good	✓		

Thank you !