Lightweight Symmetric-Key Cryptography

Thomas Peyrin

NTU - Singapore

CTCRYPT 2018

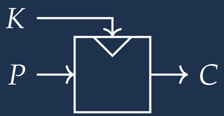
Suzdal - May 29, 2018

Outline

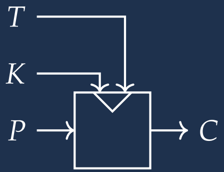
Outline

(Tweakable) Block Ciphers

A block cipher is a family of permutations parametrized by a secret key K.



A tweakable block cipher is a family of permutations parametrized by a secret key K and a tweak value T [LRW02].

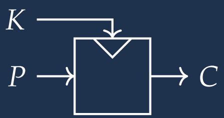


We denote

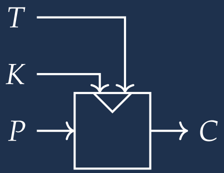
- P the n-bit plaintext
- C the n-bit ciphertext
- K the k-bit key
- T the t-bit tweak

(Tweakable) Block Ciphers

A block cipher is a family of permutations parametrized by a secret key K.



A tweakable block cipher is a family of permutations parametrized by a secret key K and a tweak value T [LRW02].

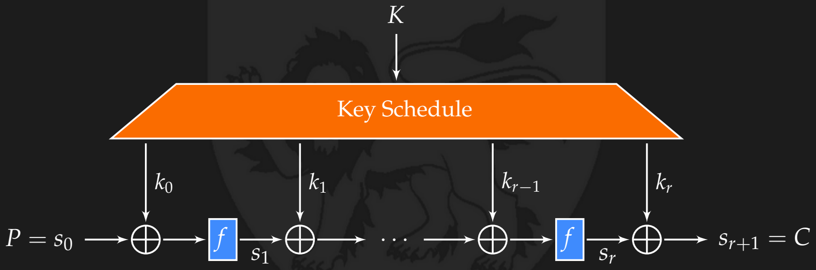


In this talk, we will only discuss about block ciphers and tweakable block ciphers
(even though you can have other lightweight primitives : stream ciphers, hash functions, operating modes)

General construction of a block cipher : iterated block ciphers

An iterated block cipher is composed of two parts :

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



General construction of a block cipher : iterated block ciphers

An iterated block cipher is composed of two parts :

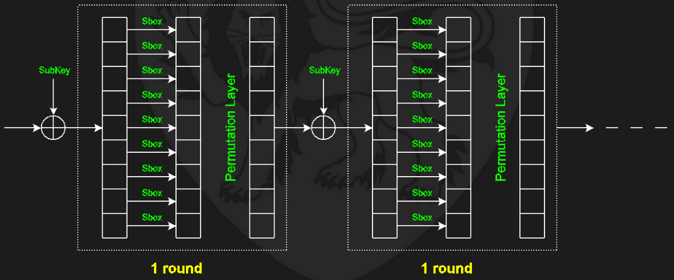
- an internal permutation f repeated r times (also named round function)
- a key schedule that generates $r+1$ subkeys $K \rightarrow (k_0, \ldots, k_r)$

Permutations

We know how to design a good permutation :

- **Feistel network**
  Example : DES (previous worldwide standard)
- **Substitution-Permutation network (SPN)**
  Example : AES (current worldwide standard)

Substitution-Permutation Network (example : AES cipher)



Permutations

We know how to design a good permutation :

- **Feistel network**
  Example : `DES` (previous worldwide standard)
- **Substitution-Permutation network (SPN)**
  Example : `AES` (current worldwide standard)

Nowadays, new ciphers MUST be secure against differential and linear (and all other known) attacks, so Sboxes, permutation layers and general structure of the cipher must be chosen very carefully !

At the same time, trying to be "lightweight" imposes many constraints, often opposite to security : **many many lightweight ciphers got broken**

Outline

Lightweight cryptography ?

We expect **RFID tags** to be 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
- ▷ it was estimated in 2005 that a basic RFID tag may have a total gate count of anywhere from 1000-10000 gates, with **only 200-2000 gates** budgeted for security

Standard block ciphers were not designed with lightweight cryptography in mind

Lightweight cryptography ?

We expect **RFID tags** to be 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
- it was estimated in 2005 that a basic RFID tag may have a total gate count of anywhere from 1000-10000 gates, with **only 200-2000 gates** budgeted for security

~~Standard block ciphers were not designed with lightweight cryptography in mind~~
Latest `AES-128` implementations only need 1600 GE (super slow though) - [JMPS07]
Is `AES-128` a lightweight cipher ?

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)

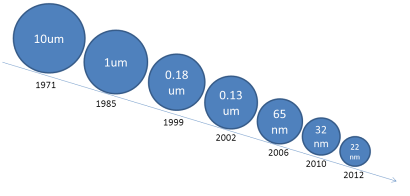
Bottom-line : for low volume production

Microcontrollers and ARM

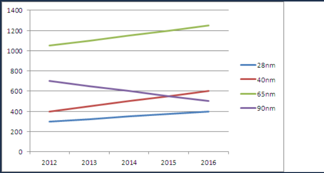
for embedded systems, mobile devices, etc.

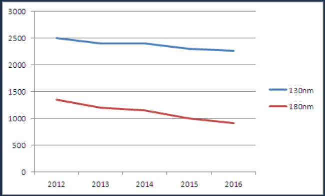
Many different platforms : ASIC

ASIC : different technology nodes



anysilicon.com





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 : area of a NAND gate

- Comparing implementations with different technologies does not make sense
- Comparing only one technology gives only a narrow view.

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**

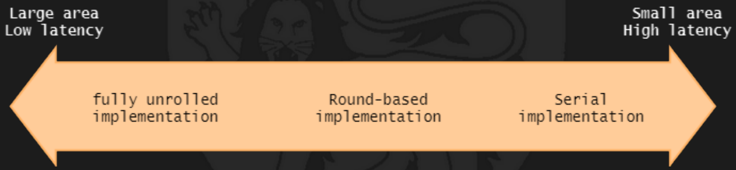
In this talk, we will only consider ASICs for simplicity

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.
- **FOM/FOAM :** a figure for taking into account the time/area/power/(security margin) tradeoffs
- 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 worse

Side-channels protection

What about **side-channels** ?

Lightweight devices will likely be easily accessible, so more subject to side-channels attacks.

Software implementation

What about 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 (it mimics hardware implementations), but then what about small messages ?

Chip production flow

There are many different stages in an hardware implementation : Synthesis, Place and Route, ... **we usually stop as the synthesis**. In theory, we should be measuring the silicon area of the final circuit (the only way to know for sure is to produce the chip).

Should we have many different ciphers for all different cases, or just one do-it-all candidate ?

It is now clear that you can't be the best everywhere (e.g. SIMON for ASIC/FPGA and SPECK for microcontrollers)

Outline

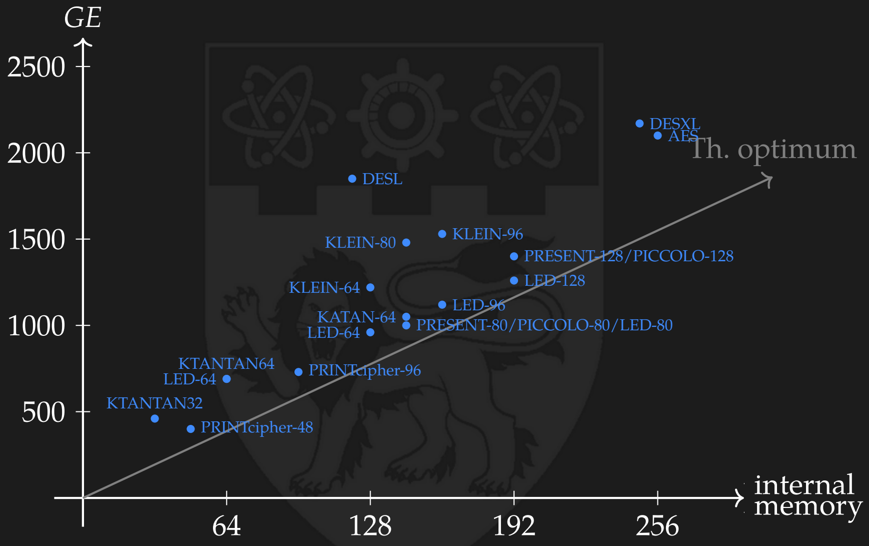
1. A Quick Introduction to Block Ciphers
2. Lightweight Cryptography : a Multi-Dimensional Problem
3. Current Design Trends
4. The Skinny tweakable block cipher
   - SKINNY description
   - SKINNY performances
5. The GIFT block cipher
   - A PRESENT and a GIFT
   - GIFT description
   - GIFT rationale
   - GIFT performances
6. Conclusion

Current status of lightweight cryptography

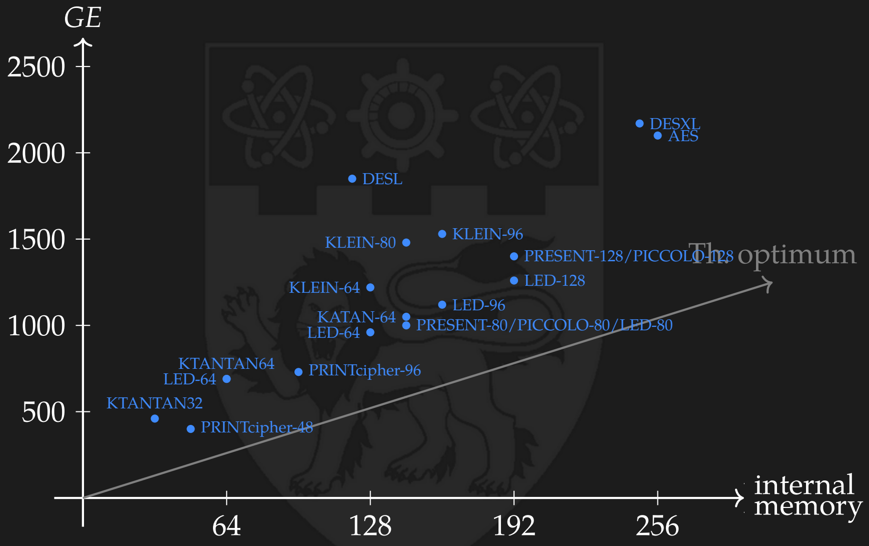
lightweight cryptography / green cryptography is a currently a very hot topic in the cryptography community

- really started in 2007 with the proposal of the cipher PRESENT (even though some ciphers like NOEKEON were already "lightweight")
- a lot of research has been conducted since then, probably more then 50 ciphers have been published (main ones listed here `https://www.cryptolux.org/index.php/Lightweight_Block_Ciphers`)
- now comes standardization time (ISO, NIST), while NSA came into play with SIMON and SPECK ciphers

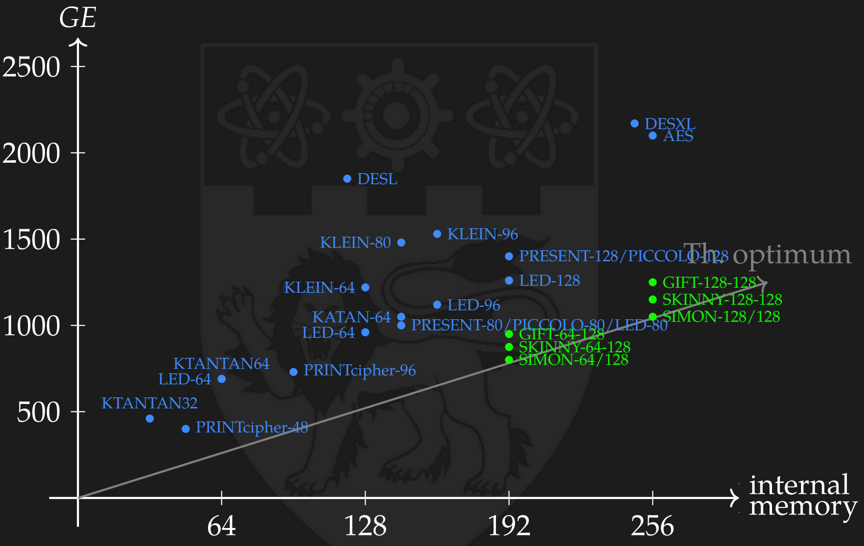
Current picture of serial implementation for block ciphers - graphically



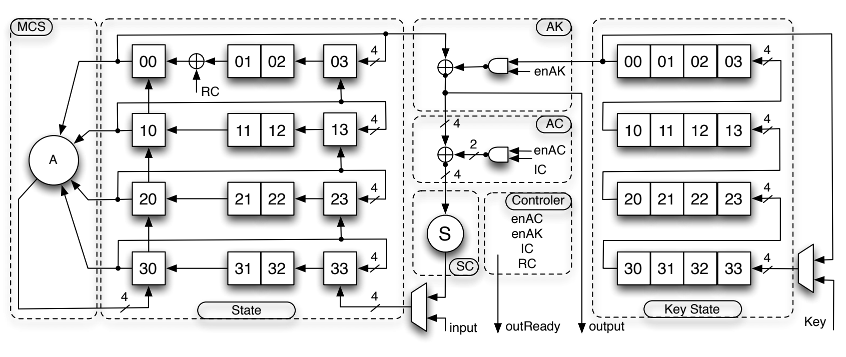
Current picture of serial implementation for block ciphers - graphically



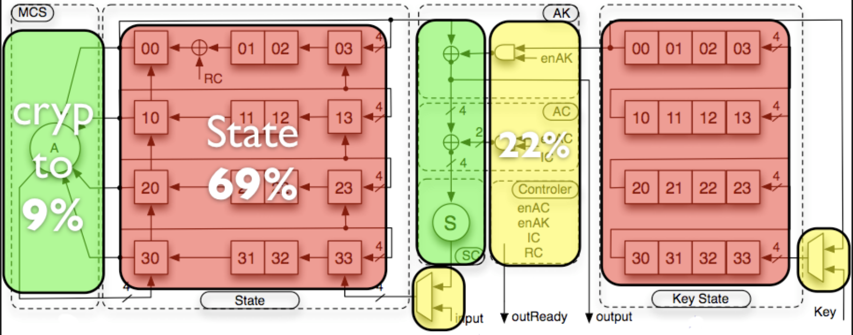
Current picture of serial implementation for block ciphers - graphically



Lightweight $\simeq$ low memory



Lightweight ≃ low memory



Lightweight ≃ low memory

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

- one flip flip 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 $\simeq$ (almost) no key schedule

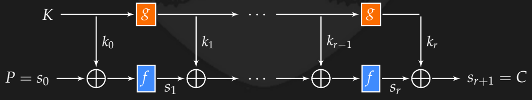
Problem :

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 $\simeq$ 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 ≃ 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 ≃ 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)

Lightweight operating modes

Beyond cipher design, **operating modes** also play a very important role

- Sponges :
  very small state, not so efficient for small message, not easily parallelisable
- Tweakable block ciphers :
  very flexible, BBB security (good for lightweight)

Outline

SKINNY

- SKINNY -

C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi,
T. Peyrin, Y. Sasaki, P. Sasdrich and S.M. Sim

CRYPTO 2016

Paper, Specifications, Results and Updates available at :
https://sites.google.com/site/skinnycipher/

SKINNY

- SKINNY -

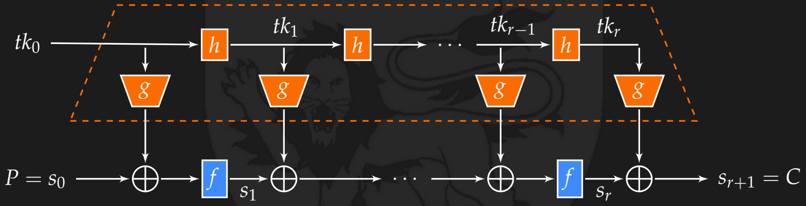
C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi,
T. Peyrin, Y. Sasaki, P. Sasdrich and S.M. Sim

CRYPTO 2016

Our goal : to propose an academy alternative to SIMON, with better security properties and tweak capability

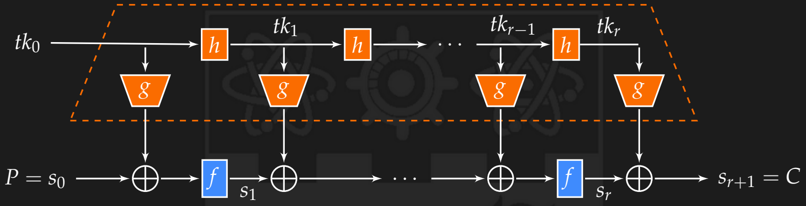
The TWEAKEY framework

The TWEAKEY framework rationale [ASIACRYPT'14] :
tweak and key should be treated the same way ⟶ tweakey



TWEAKEY **generalizes the class of key-alternating ciphers**

The TWEAKEY framework



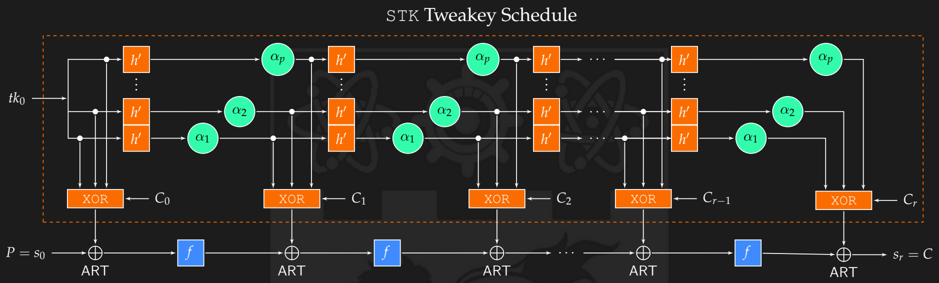
The main issue :

adding more tweakey state makes the security drop, or renders security hard to study, even for automated tools

Idea :

separate the tweakey material in several words, design a secure tweakey schedule for one word and then superpose them in a secure way

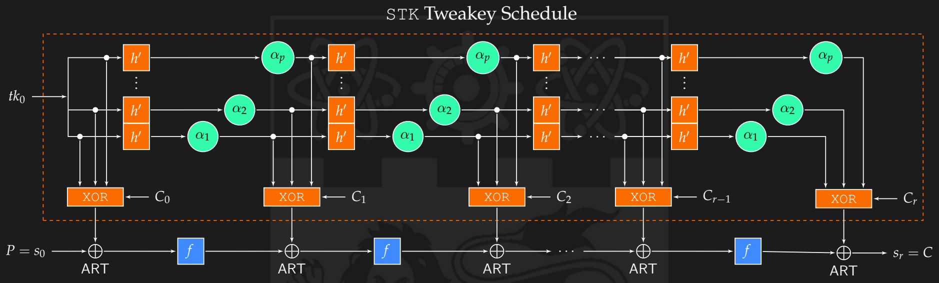
The STK construction (Superposition-TWEAKEY)



From the TWEAKEY framework to the STK construction :

- the tweakey state update function h consists in the same subfunction h' applied to each tweakey word
- the subtweakey extraction function g consists in XORing all the words together
  - reduce the implementation overhead
  - reduce the area footprint by reusing code
  - simplify the security analysis

The STK construction (Superposition-TWEAKEY)

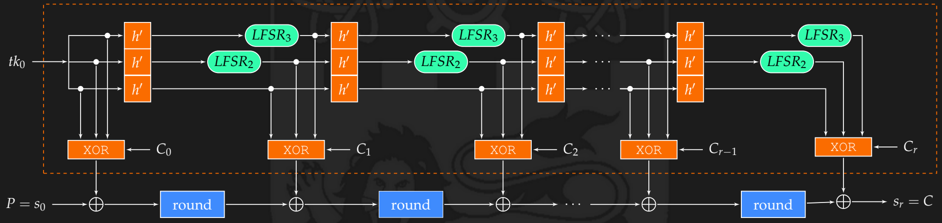


From the TWEAKEY framework to the STK construction :

- **problem** : strong interaction between the parallel branches of tweakey state
- **solution** : differentiate the parallel branches by simply using distinct small linear layers

Outline

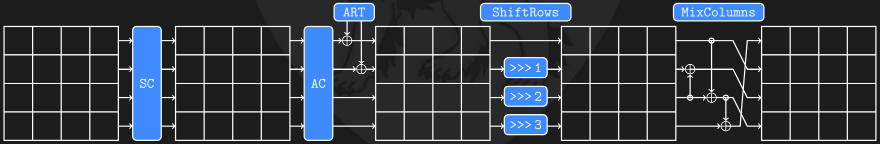
The SKINNY tweakable block ciphers



The SKINNY round function

AES-like round function

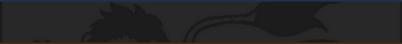
- **SubCells (SC)** : Application of a very lightweight 4-bit or 8-bit Sbox to all 16 cells
- **AddConstants (AC)** : Inject cheap round constants in the state
- **AddRoundTweakey (ART)** : Extract and inject the subtweakeys to half the state
- **ShiftRows (SR)** : Right-rotate line *i* by *i* positions
- **MixColumns (MC)** : Multiply the state by a binary matrix (only 3 XORs to apply it)



SKINNY results

SKINNY versions

Block size n	Tweakey size t: n	$2n$	$3n$
64	32 rounds	36 rounds	40 rounds
128	40 rounds	48 rounds	56 rounds



SKINNY

- A ultra-lightweight family of tweakable block ciphers
- Security guarantees for differential/linear cryptanalysis (both single and related-key)
- Efficient and competitive software/hardware implementations
- Scalable security
- Suitable for most lightweight applications
- Perform and share publicly full security analysis

Outline

Security of SKINNY and comparison with SIMON and others

Ratio of rounds required for Diff/Lin resistance

Cipher	Single Key (SK)	Related Key (RK)
SKINNY-64-128	8/36 = 22%	15/36 = 42%
SIMON-64-128	19/44 = 43%	no bound known
SKINNY-128-128	15/40 = 37%	19/40 = 47%
SIMON-128-128	37/68 = 54%	no bound known
AES-128	4/10 = 40%	6/10 = 60%

Ratio of attacked rounds

Cipher	Single Key (SK)	Related Key (RK)
SKINNY-64-128	20/36 = 55%	23/36 = 64%
SIMON-64-128	31/44 = 70%	? $\geq$ 70%
SKINNY-128-128	18/40 = 45%	19/40 = 48%
SIMON-128-128	49/68 = 72%	? $\geq$ 72%
AES-128	7/10 = 70%	7/10 = 70%

Round-based ASIC implementation results

	Area	Delay	Through. @100KHz	Through. @max
	GE	ns	KBit/s	MBit/s
SKINNY-64-128	**1696**	1.87	177.78	951.11
SKINNY-128-128	2391	2.89	320.00	1107.20
SKINNY-128-256	3312	2.89	266.67	922.67
SIMON-64-128	**1751**	1.60	145.45	870
SIMON-128-128	2342	1.60	188.24	1145
SIMON-128-256	3419	1.60	177.78	1081
LED-64-128	3036	-	133.0	-
PRESENT-64-128	1884	-	200.00	-
PICCOLO-64-128	1773	-	193.94	-

SKINNY is also very competitive for serial implementations and for implementations on microcontrollers

SKINNY cryptanalysis competition

There is currently a SKINNY cryptanalysis competition ongoing, more details here : https://sites.google.com/site/skinnycipher/cryptanalysis-competition

Outline

Outline

10 Years Ago ...

A decade ago, a lightweight block cipher, PRESENT, was presented at CHES 2007 :

- 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)

Block Cipher PRESENT

PRESENT resistance against differential cryptanalysis (DC) comes from its Sbox which has differential branching number 3

Differential branching number of an Sbox

Differential branching number is the minimum total input/output difference Hamming weight of any nonzero input/output differences. We denote BNx the set of Sbox with branching number x.

FIGURE – Hamming wt2 Example.

FIGURE – Hamming wt3 Example.

Block Cipher PRESENT

However, BN3 Sboxes are costly in general :
PRESENT Sbox (BN3) costs 21.33GE, while
SKINNY Sbox (BN2) costs 13.33GE.
This difference is multiplied in round based implementation.

Also, PRESENT is much weaker against linear cryptanalysis (LC) : paths exist with only 1 active Sbox per round, while for DC one can prove at least 2 active Sbox per round.

Now...

At CHES 2017, we presented a new lightweight block cipher, improving over `PRESENT`, we called it — `GIFT`.
Joint work with S. Banik, S.K. Pandey, S.M. Sim, Y. Todo and Y. Sasaki

By carefully crafting the bit permutation in conjunction with the Sbox properties, we can remove the constraint of BN3.

Advantages of `GIFT` compared to `PRESENT` :

- smaller area thanks to smaller Sbox and also lesser subkey additions,
- better resistance against LC thanks to good choice of Sbox and bit permutation,
- lesser rounds and higher throughput,
- simpler and faster key schedule.

Outline

Block Cipher GIFT

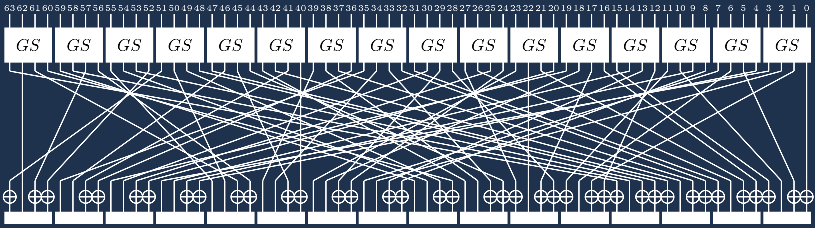
There are 2 versions of GIFT :

- GIFT−64, 28-round with 64-bit block size,
- GIFT−128, 40-round with 128-bit block size.

Both versions have 128-bit key size.

Each round of GIFT consists of 3 steps :

SubCells, PermBits and AddRoundKey



Block Cipher GIFT : SubCells

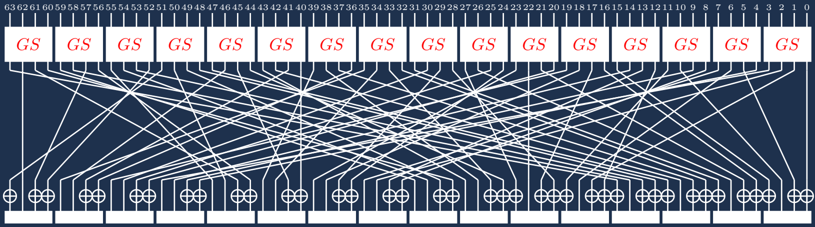
There are 2 versions of GIFT :

- GIFT−64, 28-round with 64-bit block size,
- GIFT−128, 40-round with 128-bit block size.

Both versions have 128-bit key size.

Each round of GIFT consists of 3 steps :

SubCells, PermBits and AddRoundKey



Block Cipher GIFT : PermBits

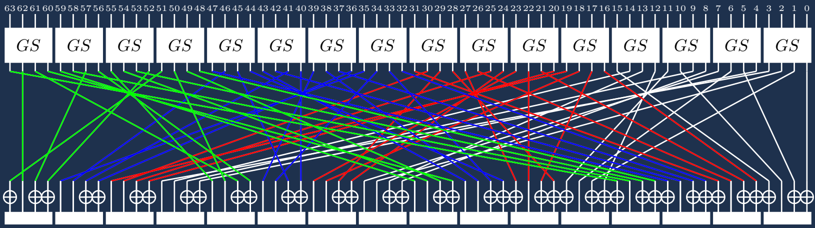
There are 2 versions of GIFT :

- GIFT−64, 28-round with 64-bit block size,
- GIFT−128, 40-round with 128-bit block size.

Both versions have 128-bit key size.

Each round of GIFT consists of 3 steps :

SubCells, PermBits and AddRoundKey



Block Cipher GIFT : AddRoundKey

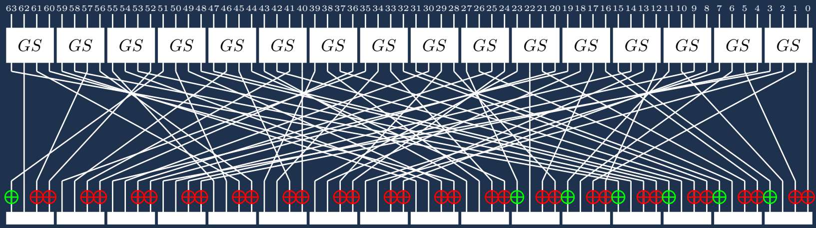
There are 2 versions of GIFT :

- GIFT−64, 28-round with 64-bit block size,
- GIFT−128, 40-round with 128-bit block size.

Both versions have 128-bit key size.

Each round of GIFT consists of 3 steps :

SubCells, PermBits and AddRoundKey



Block Cipher GIFT : Round Key and Key Schedule

The 128-bit key is split into eight 16-bit words.

$$K = k_7 \| k_6 \| \ldots \| k_1 \| k_0, \text{ where } k_i \text{ is 16-bit words.}$$

k_1 and k_0 are extracted as the round key $RK = U \| V$.

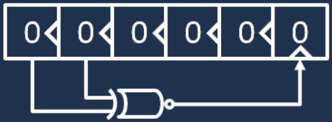
k_7	k_6	k_5	k_4	k_3	k_2	k_1	k_0

Key state is updated after key extraction :

$k_1 \ggg 2$	$k_0 \ggg 12$	k_7	k_6	k_5	k_4	k_3	k_2

Block Cipher GIFT : Round Constants

Round constants are generated using a 6-bit affine LFSR with 1 XNOR gate (same as SKINNY's).



Initialised to zero, and updated before using as round constants.

Outline

GIFT rationale : Bad Output must go to Good Input (BOGI)

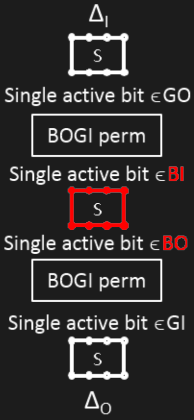
Table – 1 – 1 bit DDT Example

Δx \ Δy	bit 3	bit 2	bit 1	bit 0
bit 3	0	2	4	0
bit 2	0	0	0	0
bit 1	0	0	0	0
bit 0	0	2	2	0

Let GI, GO, BI, BO denote the set of good inputs, good outputs, bad inputs and bad outputs respectively.

$$GI = \{\text{bit } 2, \text{bit } 1\}, \qquad GO = \{\text{bit } 3, \text{bit } 0\},$$
$$BI = \{\text{bit } 3, \text{bit } 0\}, \qquad BO = \{\text{bit } 2, \text{bit } 1\}.$$

GIFT Core Idea



Observation :

If a single active bit transition occurs, the input and output active bit must be in *BI* and *BO*.

Core idea :

We send the bit from *BO* to *GI* so that single bit transition does not happen continuously. Same for backward direction.

Both Δ_I and Δ_O have at least 2 active bits.

≥ 7 active Sboxes in 5 rounds !

BOGI Permutation in GIFT

Let $\pi_1 : BO \to GI$ and $\pi_2 : GO \to (\pi_1(BO))^c$.
BOGI permutation π is the union of π_1 and π_2.

$$GI = \{\text{bit } 2, \text{bit } 1\}, \qquad GO = \{\text{bit } 3, \text{bit } 0\},$$
$$BI = \{\text{bit } 3, \text{bit } 0\}, \qquad BO = \{\text{bit } 2, \text{bit } 1\}.$$

For this example, π can be an identity mapping.
I.e. $\pi : \text{bit } j \mapsto \text{bit } j$.

Necessary and sufficient condition :

$$|BO| \leq |GI| \implies |GI| + |GO| \geq 4$$

Denote $|GI| + |GO|$ the score of an Sbox.

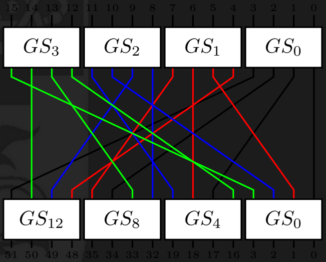
This can be extended to the $1 - 1$ bit LAT and linear cryptanalysis, which is the Achilles' heel of PRESENT.

GIFT-64 Group Mapping

New bit permutation based on BOGI group mapping.

TABLE – GIFT-64 group mapping

Q0 \ R0	GS_0	GS_4	GS_8	GS_{12}
GS_0	$(0,0)$	$(1,1)$	$(2,2)$	$(3,3)$
GS_1	$(1,1)$	$(2,2)$	$(3,3)$	$(0,0)$
GS_2	$(2,2)$	$(3,3)$	$(0,0)$	$(1,1)$
GS_3	$(3,3)$	$(0,0)$	$(1,1)$	$(2,2)$



Select an Sbox with score 4 and has BOGI identity permutation.

We found one such Sbox with also good cryptographic properties (MDP, MLP, algebraic degree, etc) and that is much cheaper than the PRESENT Sbox (16 GE instead of 21.33 GE)

Block Cipher GIFT : Differential and Linear Bounds

TABLE – Lower bounds for number of active Sboxes.

Cipher	DC/LC	Rounds 1	2	3	4	5	6	7	8	9
GIFT-64	DC	1	2	3	5	7	10	13	16	18
	LC	1	2	3	5	7	9	12	15	18
PRESENT	DC	1	2	4	6	10	12	14	16	18
	LC	1	2	3	4	5	6	7	8	9
GIFT-128	DC	1	2	3	5	7	10	13	17	19
	LC	1	2	3	5	7	9	12	14	18

GIFT matches the differential bound of PRESENT : an average of 2 active Sboxes per round.

In addition, GIFT achieves the same ratio for linear bound at 9-round where PRESENT could not.

Block Cipher GIFT : Differential and Linear Probabilities

Table – 9-round Differential/Linear Probabilities

Cipher	No. of Rounds	Differential Probability	Linear Hull Effect	Est. Rounds Needed
GIFT-64	28	$2^{-44.415}$	$2^{-49.997}$	14
PRESENT	31	$2^{-40.702}$	$2^{-27.186}$	22
GIFT-128	40	$2^{-46.99}$	$2^{-45.99}$	27

Outline

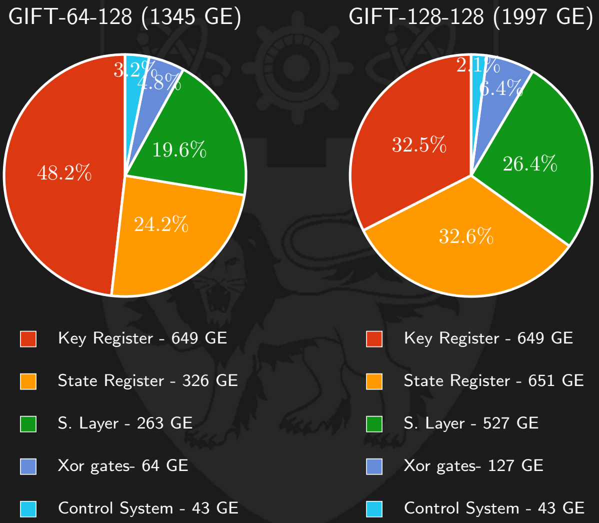
Block Cipher GIFT : Round-based Implementation

GIFT currently has the most efficient round-based implementation.

Table – Round-based implementations synthesized with STM 90nm Standard cell library

Cipher	Area (GE)	Delay (ns)	Cycles	TP_{MAX} (MBit/s)	Power (μW) (@10MHz)	Energy (pJ)
GIFT-64-128	1345	1.83	29	1249.0	74.8	216.9
SKINNY-64-128	1477	1.84	37	966.2	80.3	297.0
PRESENT 64/128	1560	1.63	33	1227.0	71.1	234.6
SIMON 64/128	1458	1.83	45	794.8	72.7	327.3
GIFT-128-128	1997	1.85	41	1729.7	116.6	478.1
SKINNY-128-128	2104	1.85	41	1729.7	132.5	543.3
SIMON 128/128	2064	1.87	69	1006.6	105.6	728.6
AES 128	7215	3.83	11	3038.2	730.3	803.3

Block Cipher GIFT : Round-based Implementation



Block Cipher GIFT : Bit-slice Implementation

GIFT is also very efficient in software.

TABLE – Bitslice software implementations, results given in cycles per byte, with messages composed of 2000 64-bit blocks.

Cipher	Speed (c/B)	Cipher	Speed (c/B)
GIFT-64-128	2.10	GIFT-128-128	2.57
SKINNY-64-128	2.88	SKINNY-128-128	4.70
SIMON-64-128	1.74	SIMON-128-128	2.55

Outline

Table – Total number of operations of various lightweight block ciphers. N denotes a NOR gate, A denotes a AND gate, X denotes a XOR gate.

Cipher	nb. of rds	gate cost (per bit per round) int. cipher	key sch.	total	nb. of op. w/o key sch.	nb. of op. w/ key sch.	round-based impl. area
GIFT -64-128	28	1 N 2 X		1 N 2 X	3×28 $= 84$	3×28 $= \mathbf{84}$	$1 + 2.67 \times 2$ $= \mathbf{6.34}$
SKINNY -64-128	36	1 N 2.25 X	0.625 X	1 N 2.875 X	3.25×36 $= 117$	3.875×36 $= \mathbf{139.5}$	$1 + 2.67 \times 2.875$ $= \mathbf{8.68}$
SIMON -64/128	44	0.5 A 1.5 X	1.5 X	0.5 A 3.0 X	2×44 $= 88$	3.5×44 $= \mathbf{154}$	$0.67 + 2.67 \times 3$ $= \mathbf{8.68}$
PRESENT -128	31	1 A 3.75 X	0.125 A 0.344 X	1.125 A 4.094 X	4.75×31 $= 147.2$	5.22×31 $= \mathbf{161.8}$	$1.5 + 2.67 \times 4.094$ $= \mathbf{12.43}$
GIFT -128-128	40	1 N 2 X		1 N 2 X	3.0×40 $= 120$	3.0×40 $= \mathbf{120}$	$1 + 2.67 \times 2$ $= \mathbf{6.34}$
SKINNY -128-128	40	1 N 2.25 X		1 N 2.25 X	3.25×40 $= 130$	3.25×40 $= \mathbf{130}$	$1 + 2.67 \times 2.25$ $= \mathbf{7.01}$
SIMON -128/128	68	0.5 A 1.5 X	1 X	0.5 A 2.5 X	2×68 $= 136$	3×68 $= \mathbf{204}$	$0.67 + 2.67 \times 2.5$ $= \mathbf{7.34}$
AES -128	10	4.25 A 16 X	1.06 A 3.5 X	5.31 A 19.5 X	20.25×10 $= 202.5$	24.81×10 $= \mathbf{248.1}$	$7.06 + 2.67 \times 19.5$ $= \mathbf{59.12}$

This is the end ?

We are probably reaching the limits of performance for lightweight encryption on ASIC.

Challenge : can you design a cipher with sufficient security margin that requires less operation per bit than `GIFT` ?

There might still be room for improvement for getting better software performances at the same time
(this will be less and less true in future, as bitslice implementations mimic ASIC implementations)

Standardization time

Now is time for standardization :

- **NIST** is preparing a competition for lightweight encryption that will start soon (currently waiting for public comments on the draft call for submissions)
- **ISO** already has a lightweight cryptography section (for example PRESENT, PHOTON, SPONGENT, etc.). There is currently a study period regarding the inclusion of `SKINNY` in ISO standards.

Thank you !