The LED Block Cipher

Jian Guo, Thomas Peyrin, Axel Poschmann and Matt Robshaw

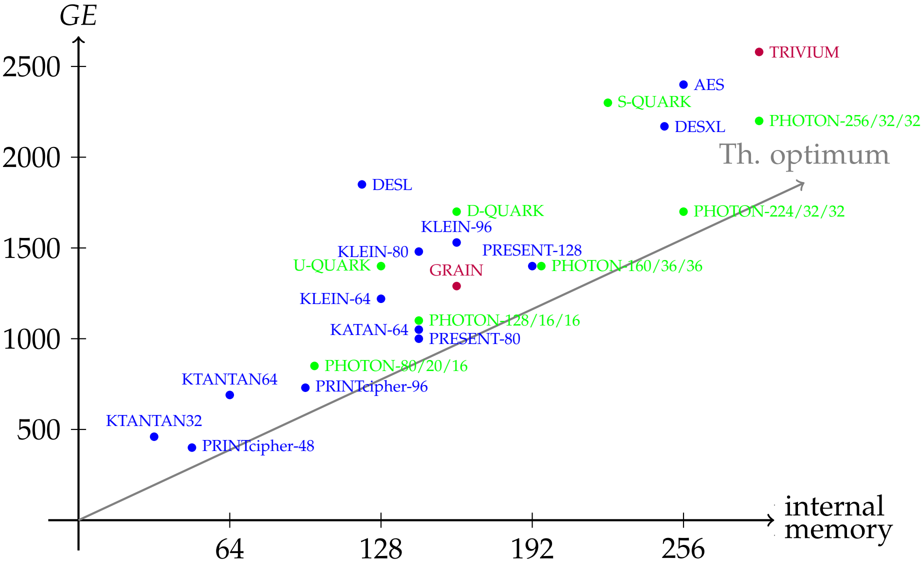
I2R, NTU and Orange Labs

CHES 2011

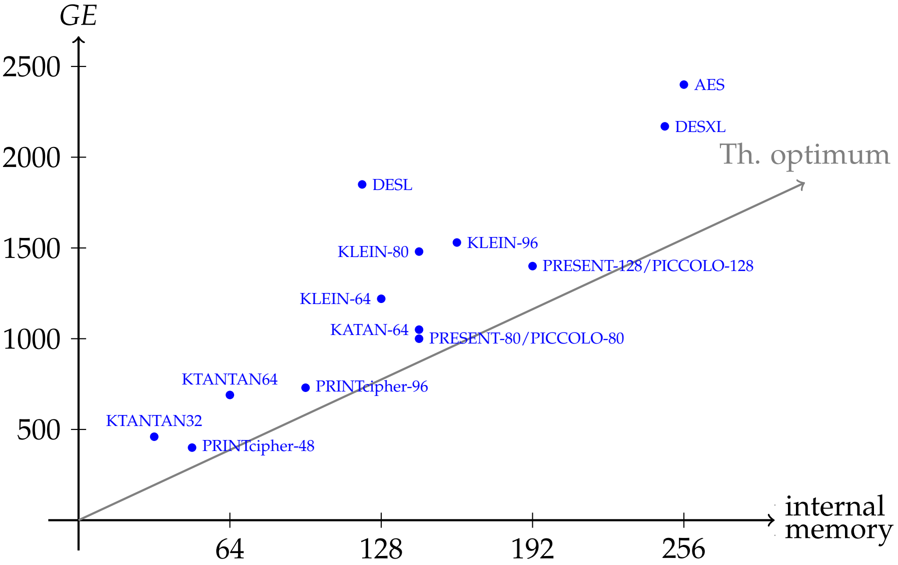
Nara, Japan

Outline

Current picture of lightweight primitives - graphically



Current picture of lightweight block ciphers - graphically



Lightweight block ciphers are too provocative ?

- `ARMADILLO`: key-recovery attacks [A+-2011]
- `HIGHT`: related-key attacks [K+-2010]
- `Hummingbird-1`: practical related-IV attacks [S-2011]
- `KTANTAN`: practical related-key attacks [Å-2011]
- `PRINTcipher`: large weak-keys classes [ÅJ-2011]

`PRESENT` is still unbroken.

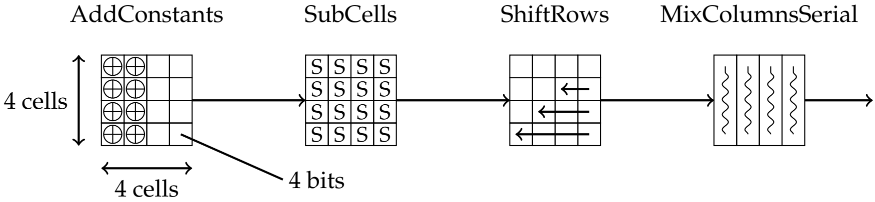
Light Encryption Device

We propose a new **64-bit block cipher** LED:

- as **small** as PRESENT
- **faster** than PRESENT in software (and slower in hardware)
- significant security margin
- can take **any key size** from 64 to 128 bits
- **key can be directly hardwired** (without any modification)
- **provable resistance** to classical differential and linear attacks ...
- ... both in the **single-key** and **related-key** models

Outline

A single round of LED



The 64-bit round function is an SP-network:

- **AddConstants:** xor round-dependent constants to the two first columns
- **SubCells:** apply the PRESENT 4-bit Sbox to each cell
- **ShiftRows:** rotate the i-th line by i positions to the left
- **MixColumnsSerial:** apply the special MDS matrix to each columns independently

Efficient Serially Computable MDS Matrices

MDS Matrices ("Maximum Distance Separable") have **excellent diffusion properties**: for a d-cell vector, we are ensured that at least $d + 1$ input / output cells will be active.

We use the same trick as in PHOTON (CRYPTO 2011): **implement an MDS matrix that can be efficiently computed in a serial way**. We keep the **same good diffusion properties** and **good software performances** as the classical MDS constructions, but the **hardware is improved since no additional memory cell is needed** (for both ciphering and deciphering).

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\ Z_0 & Z_1 & Z_2 & Z_3 & \cdots & Z_{d-4} & Z_{d-3} & Z_{d-2} & Z_{d-1} \end{pmatrix}$$

Efficient Serially Computable MDS Matrices

MDS Matrices ("Maximum Distance Separable") have **excellent diffusion properties**: for a d-cell vector, we are ensured that at least $d+1$ input / output cells will be active.

We use the same trick as in PHOTON (CRYPTO 2011): **implement an MDS matrix that can be efficiently computed in a serial way**. We keep the **same good diffusion properties** and **good software performances** as the classical MDS constructions, but the **hardware is improved since no additional memory cell is needed** (for both ciphering and deciphering).

$$\begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\ Z_0 & Z_1 & Z_2 & Z_3 & \cdots & Z_{d-4} & Z_{d-3} & Z_{d-2} & Z_{d-1} \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{d-4} \\ v_{d-3} \\ v_{d-2} \\ v_{d-1} \end{pmatrix} =$$

Efficient Serially Computable MDS Matrices

MDS Matrices ("Maximum Distance Separable") have **excellent diffusion properties**: for a d-cell vector, we are ensured that at least $d+1$ input / output cells will be active.

We use the same trick as in PHOTON (CRYPTO 2011): **implement an MDS matrix that can be efficiently computed in a serial way**. We keep the **same good diffusion properties** and **good software performances** as the classical MDS constructions, but the **hardware is improved since no additional memory cell is needed** (for both ciphering and deciphering).

$$\begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\ Z_0 & Z_1 & Z_2 & Z_3 & \cdots & Z_{d-4} & Z_{d-3} & Z_{d-2} & Z_{d-1} \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{d-4} \\ v_{d-3} \\ v_{d-2} \\ v_{d-1} \end{pmatrix} = \begin{pmatrix} v_1 \\ \\ \vdots \\ \\ \\ \\ \\ \end{pmatrix}$$

Efficient Serially Computable MDS Matrices

MDS Matrices ("Maximum Distance Separable") have **excellent diffusion properties**: for a d-cell vector, we are ensured that at least $d+1$ input / output cells will be active.

We use the same trick as in PHOTON (CRYPTO 2011): **implement an MDS matrix that can be efficiently computed in a serial way**. We keep the **same good diffusion properties** and **good software performances** as the classical MDS constructions, but the **hardware is improved since no additional memory cell is needed** (for both ciphering and deciphering).

$$\begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\ Z_0 & Z_1 & Z_2 & Z_3 & \cdots & Z_{d-4} & Z_{d-3} & Z_{d-2} & Z_{d-1} \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{d-4} \\ v_{d-3} \\ v_{d-2} \\ v_{d-1} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ \\ \\ \\ \\ \end{pmatrix}$$

Efficient Serially Computable MDS Matrices

MDS Matrices ("Maximum Distance Separable") have **excellent diffusion properties**: for a d-cell vector, we are ensured that at least $d+1$ input / output cells will be active.

We use the same trick as in PHOTON (CRYPTO 2011): **implement an MDS matrix that can be efficiently computed in a serial way**. We keep the **same good diffusion properties** and **good software performances** as the classical MDS constructions, but the **hardware is improved since no additional memory cell is needed** (for both ciphering and deciphering).

$$\begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\ Z_0 & Z_1 & Z_2 & Z_3 & \cdots & Z_{d-4} & Z_{d-3} & Z_{d-2} & Z_{d-1} \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{d-4} \\ v_{d-3} \\ v_{d-2} \\ v_{d-1} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{d-3} \\ \\ \\ \\ \end{pmatrix}$$

Efficient Serially Computable MDS Matrices

MDS Matrices ("Maximum Distance Separable") have **excellent diffusion properties**: for a d-cell vector, we are ensured that at least $d+1$ input / output cells will be active.

We use the same trick as in PHOTON (CRYPTO 2011): **implement an MDS matrix that can be efficiently computed in a serial way**. We keep the **same good diffusion properties** and **good software performances** as the classical MDS constructions, but the **hardware is improved since no additional memory cell is needed** (for both ciphering and deciphering).

$$\begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\ Z_0 & Z_1 & Z_2 & Z_3 & \cdots & Z_{d-4} & Z_{d-3} & Z_{d-2} & Z_{d-1} \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{d-4} \\ v_{d-3} \\ v_{d-2} \\ v_{d-1} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{d-3} \\ v_{d-2} \\ \\ \end{pmatrix}$$

Efficient Serially Computable MDS Matrices

MDS Matrices ("Maximum Distance Separable") have **excellent diffusion properties**: for a d-cell vector, we are ensured that at least $d+1$ input / output cells will be active.

We use the same trick as in PHOTON (CRYPTO 2011): **implement an MDS matrix that can be efficiently computed in a serial way**. We keep the **same good diffusion properties** and **good software performances** as the classical MDS constructions, but the **hardware is improved since no additional memory cell is needed** (for both ciphering and deciphering).

$$\begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\ Z_0 & Z_1 & Z_2 & Z_3 & \cdots & Z_{d-4} & Z_{d-3} & Z_{d-2} & Z_{d-1} \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{d-4} \\ v_{d-3} \\ v_{d-2} \\ v_{d-1} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{d-3} \\ v_{d-2} \\ v_{d-1} \\ \end{pmatrix}$$

Efficient Serially Computable MDS Matrices

MDS Matrices ("Maximum Distance Separable") have **excellent diffusion properties**: for a d-cell vector, we are ensured that at least $d+1$ input / output cells will be active.

We use the same trick as in PHOTON (CRYPTO 2011): **implement an MDS matrix that can be efficiently computed in a serial way**. We keep the **same good diffusion properties** and **good software performances** as the classical MDS constructions, but the **hardware is improved since no additional memory cell is needed** (for both ciphering and deciphering).

$$
\begin{pmatrix}
0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\
 & \vdots & & & & & & \vdots & \\
0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\
Z_0 & Z_1 & Z_2 & Z_3 & \cdots & Z_{d-4} & Z_{d-3} & Z_{d-2} & Z_{d-1}
\end{pmatrix}
\cdot
\begin{pmatrix}
v_0 \\ v_1 \\ \vdots \\ v_{d-4} \\ v_{d-3} \\ v_{d-2} \\ v_{d-1}
\end{pmatrix}
=
\begin{pmatrix}
v_1 \\ v_2 \\ \vdots \\ v_{d-3} \\ v_{d-2} \\ v_{d-1} \\ v'_0
\end{pmatrix}
$$

The MixColumnsSerial matrix for LED

The serial decomposition of our MixColumnsSerial matrix is very lightweight (the matrix $(B)^4$ is MDS):

$$(B)^4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 4 & 1 & 2 & 2 \end{pmatrix}^4 = \begin{pmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{pmatrix}$$

So is its inverse:

$$(B^{-1})^4 = \begin{pmatrix} 1 & 2 & 2 & 4 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}^4 = \begin{pmatrix} C & C & D & 4 \\ 3 & 8 & 4 & 5 \\ 7 & 6 & 2 & E \\ D & 9 & 9 & D \end{pmatrix}$$

Outline

The Key Schedule of LED

Recent lessons learned in block ciphers design:

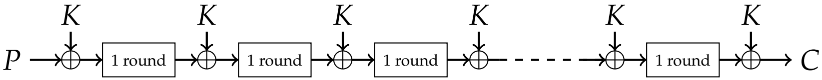
- **designing key schedules is hard** (see recent attacks on AES), same for message expansions in hash functions (look at the SHA-3 competition)
- obtaining **security proofs** when also considering differences in the key schedule is not trivial ...
- either you use the very same function (can be bad, see attacks on Whirlpool)
- either you use a purposely different function in order to make cryptanalysis hard (see AES, PRESENT, ...)

Our rationale: use NO key schedule

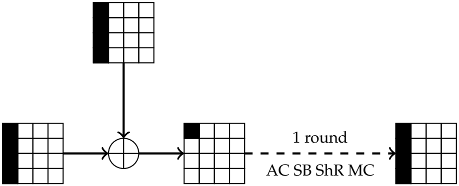
- much **simpler for cryptanalysts**, not relying on the difficulty to analyze
- **only leverages the quality of the permutation** and we DO know how to build good permutations
- **you can directly hardwire the key** in some particular scenarios

First attempt

Key repeated every round

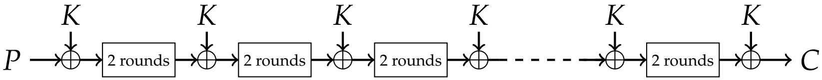


But paths exist with only **1 active Sbox per round** on average

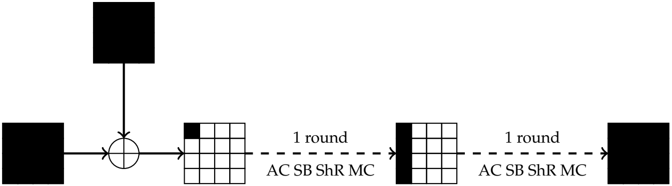


Second attempt

Key repeated every two rounds

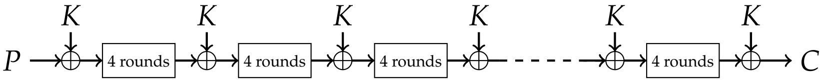


But paths exist with only **2.5 active Sboxes per round** on average

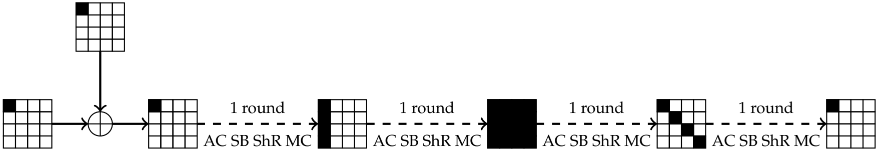


Third attempt

Key repeated every four rounds

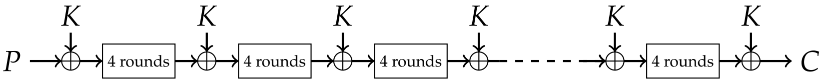


The best path has **3.125 active Sboxes per round** on average

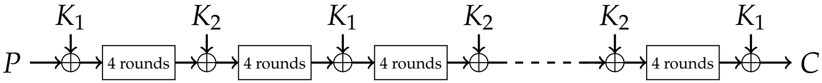


LED key schedule

For **64-bit key**, we xored it to the internal state **every four rounds**. We apply a total of **8 steps (or 32 rounds)**:



For **up to 128-bit key**, we divide it into **two equal chunks** K_1 and K_2 that are alternatively xored to the internal state **every four rounds**. We apply a total of **12 steps (or 48 rounds)**:



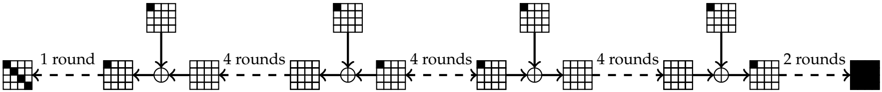
Outline

Differential/linear attacks

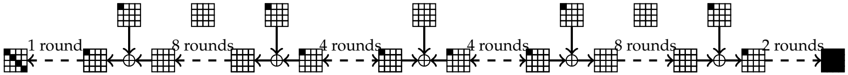
- AES**-like permutations** are simple to understand, well studied, provide very good security
- **In single-key model:** one can easily derive proofs on the minimal number of active Sboxes for 4 rounds of the permutation:
  $(d+1)^2 = 25$ **active Sboxes for 4 rounds of** LED
- **In related-key model:** we have at least half of the 4-round steps active, using the same reasoning we obtain:
  $(d+1)^2 = 25$ **active Sboxes for 8 rounds of** LED

	LED-64 SK	LED-64 RK	LED-128 SK	LED-128 RK
minimal no. of active Sboxes	200	100	300	150
differential path probability	2^{-400}	2^{-200}	2^{-600}	2^{-300}
linear approx. probability	2^{-400}	2^{-200}	2^{-600}	2^{-300}

Rebound attack and improvements



In the **chosen-related-key model**, one can distinguish **15 rounds** (over 32) of LED-64 with complexity 2^{16}



In the **chosen-related-key model**, one can distinguish **27 rounds** (over 48) of LED-128 with complexity 2^{16}

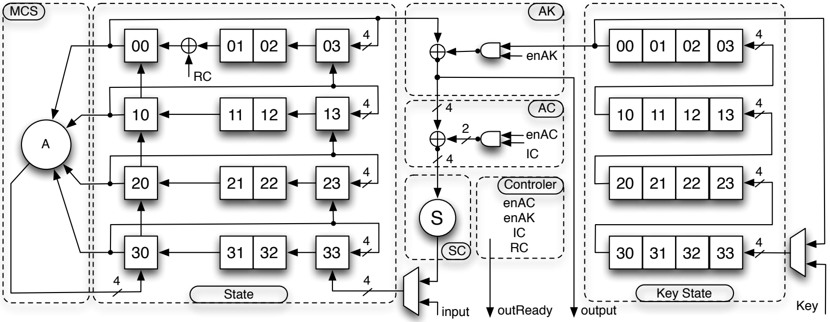
Improvements are unlikely since no key is used during four rounds of the permutation, so **the amount of freedom degrees given to the attacker is limited to the minimum**.

Other cryptanalysis techniques

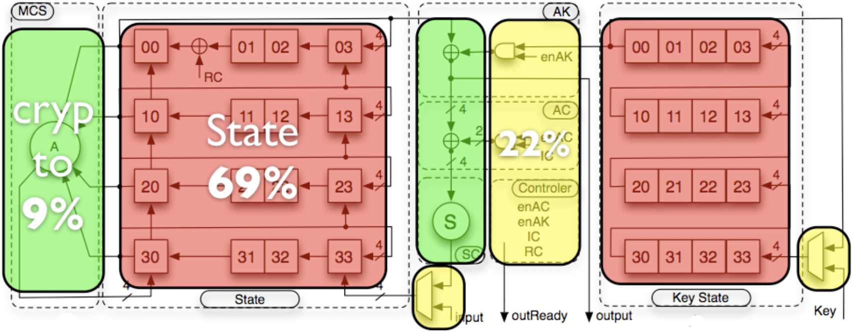
- **cube testers:** the best we could find within practical time complexity is at most 3 rounds
- **zero-sum partitions:** distinguishers for at most 12 rounds with 2^{64} complexity in the known-key model
- **algebraic attacks:** the entire system for a 64-bit fixed-key `LED` permutation consists of 10752 quadratic equations in 4096 variables
- **slide attacks:** all rounds are made different thanks to the round-dependent constants addition
- **rotational cryptanalysis:** any rotation property in a cell will be directly removed by the application of the Sbox layer
- **integral attacks:** currently can't even break 2 steps

Outline

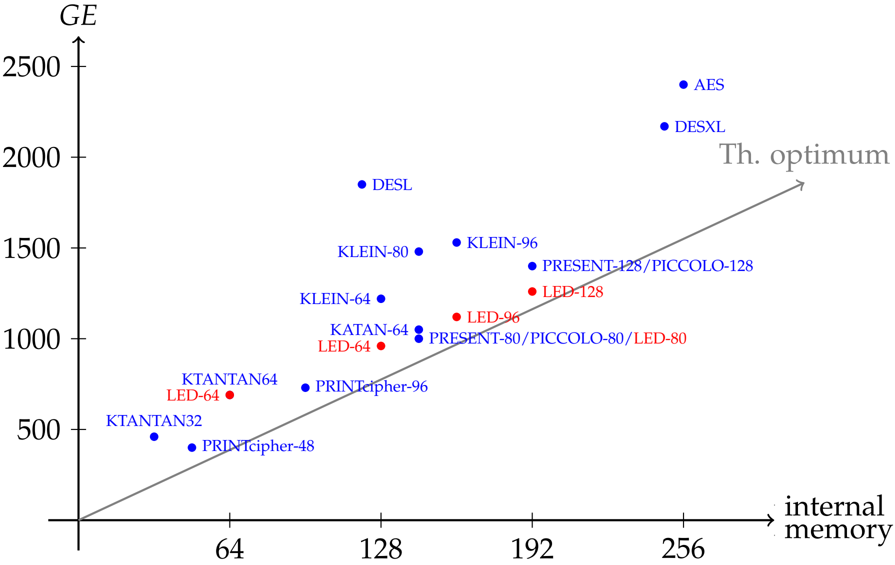
Hardware implementation



Hardware implementation



Hardware implementation results



Software implementation results

Table: Software implementation results of LED.

	table-based implementation
LED-64	57 cycles/byte
LED-128	86 cycles/byte

One can use **"Super-Sbox" implementations** (ongoing work).

Conclusion

The LED block cipher:

- is very **simple** and **clean**
- is as **small** as PRESENT
- **faster** than PRESENT in software (and slower in hardware)
- **key can be hardwired** without modification of the algorithm
- provides **provable security** against classical linear/differential cryptanalysis **both in the single-key and related-key models**
- extremely large security margin in the single-key model
- security analysis done in the very optimistic known/chosen-keys model

Latest results on https://sites.google.com/site/ledblockcipher/