NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Automated Analysis for Pushing Performance Limits in Symmetric-Key Cryptography

Thomas Peyrin
NTU Singapore

Huawei Forum on Trust and Privacy for the Future Digital World 2024

Singapore
29th November 2024

Problem Statement

Cryptographic design is always a fight **performance** vs **security**

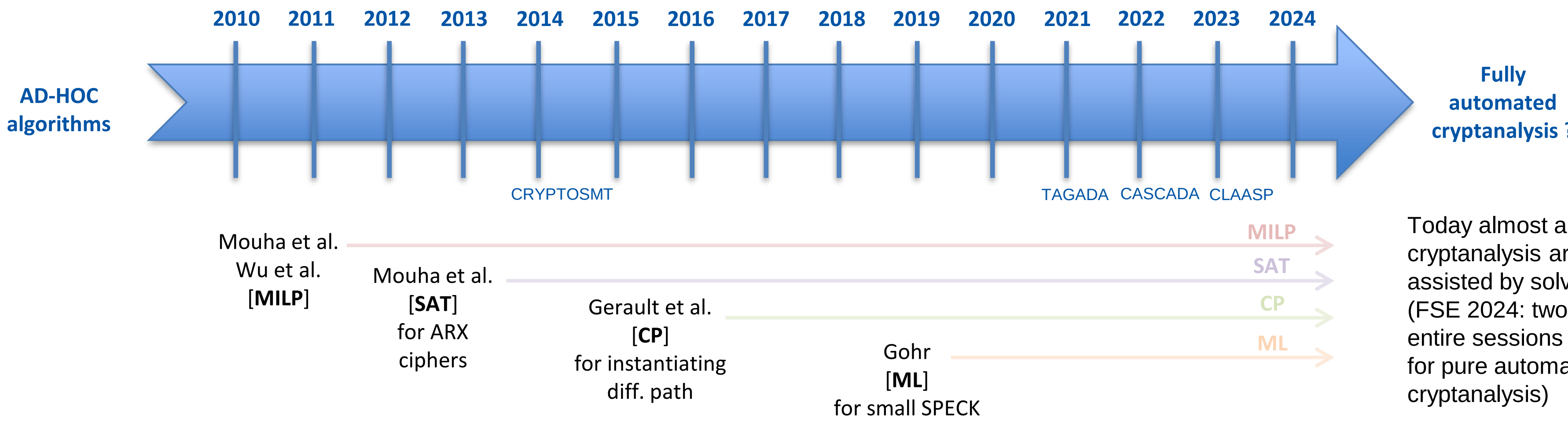
Performance is usually modeled according to some physical/technological model, and the community is now considering more and more exotic metrics (lightweight, low-latency, MPC-friendly, etc)

Security analysis was done by humans and now more and more assisted by automated tools.

Can automated tools be more integrated within the design process ?

Automated Cryptanalysis

Timeline of Automated Cryptanalysis



Today almost all cryptanalysis are assisted by solvers (FSE 2024: two entire sessions just for pure automated cryptanalysis)

Automated cryptanalysis using declarative frameworks (SAT/MILP/CP/etc.) is generally slower or at best same as ad-hoc tools, but so much **more convenient**

Mainly on **differential** and **linear cryptanalysis**, but now also on integral distinguishers, cube attacks, meet-in-the-middle attacks, etc.

Solving time is a crucial aspect and can be impacted by:

- the framework you use (SAT/MILP/CP/etc.)
- the strategy of modeling (many works on various modeling strategies)
- the solver (less contributions on that, different research field)
- the type of problem studied / scale

Automated Cryptanalysis for Designers

Classical design process: cipher's structure is pre-established by the human. The computer will brute force some components (Sbox, diffusion matrix) or parameters (rotation constant, etc.) to select the best candidate.

However:

- There is no "search" per se, it is just localized small brute force searches and taking the best candidates
- Evaluation of the cipher's security and performance is done at the end (no insight to search in a smart way)

Can we give more freedom to the computer to create good ciphers ?

Can automated cryptanalysis help us searching for good ciphers ?

Fast AES-based MAC

LeMac - PetitMac

Fast AES-Based Universal Hash Functions and MACs (Featuring LeMac and PetitMac) – **ToSC 2024-2**

Joint work with A. Bariant, J. Baudrin, G. Leurent, C. Pernot and L. Perrin

Why Fast MAC ?

- AES has globally good performances, but it is **really fast in practice** because of **hardware acceleration** widely available (AES-NI).
- The granularity of AES-NI is on the **AES round**, so it has been used to build many fast primitives:
  - Hash functions (ECHO, LANE, SHAVITE-3, VORTEX, etc.),
  - AEAD schemes (AEGIS, TIAOXIN-346, DEOXYS, ROCCA(-S), etc.),
  - Permutations (AREION, SIMPIRA, HARAKA, PHOLKOS, etc.).
- Now, not so difficult to reach throughput < 1 c/B on typical processors

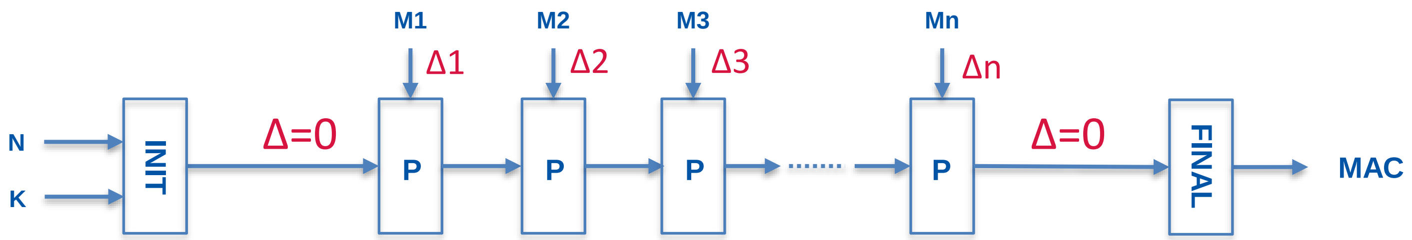
  Ex: 2 AES rounds in parallel each cycle, thus (10/2)/16 = 0.31 c/B
- But sixth-generation mobile comm. systems (6G) to deliver an amazing throughput of **100 Gbps to 1 Tbps** (0.24 to 0.024 c/B on a 3GHz CPU) !

We need to create primitives with even <u>much</u> larger throughput !

State-of-the-art of Fast AES-based MAC

Many ultra-fast AES-based collision resistant permutations:

AEGIS, TIAOXIN-346, ROCCA-(S), Jean-Nikolić [JN16] and Nikolić [Nik17a] (fastest)



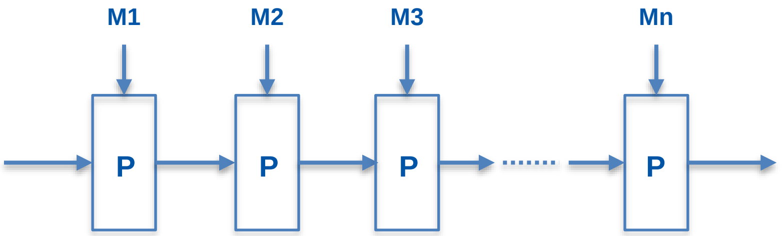
Goal: guarantee **no collision path** exist with good probability

ROCCA targets 256-bit key / 128-bit tag AEAD. Some security issues [HII+22].
ROCCA-S targets 256-bit key / 256-bit tag AEAD (under submission at IETF).

<u>Sub-optimal throughput:</u> optimal in ROCCA framework [TSI23] reaches 0.104 c/B on Tiger Lake, while theoretical max is 0.0625 c/B.

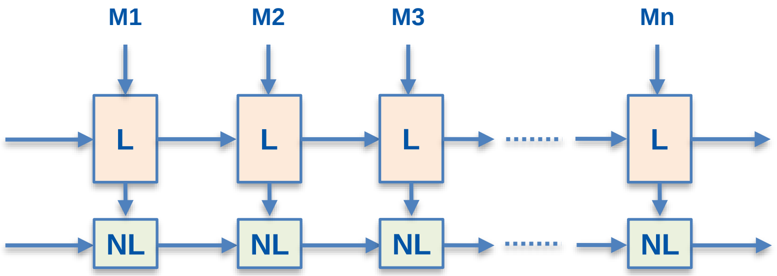
Designing a collision-resistant permutation

Classical: large state entirely updated non-linearly. Issue: costly for a large state.



Better ?: large state separated in two parts (inspired from TBC or PANAMA hash):

- **one part updated with (expensive) non-linear components** (AES round in our case)
- **one part updated with linear components** (not influenced by the first one, reducing dependencies that complicate instructions scheduling and automated security analysis).

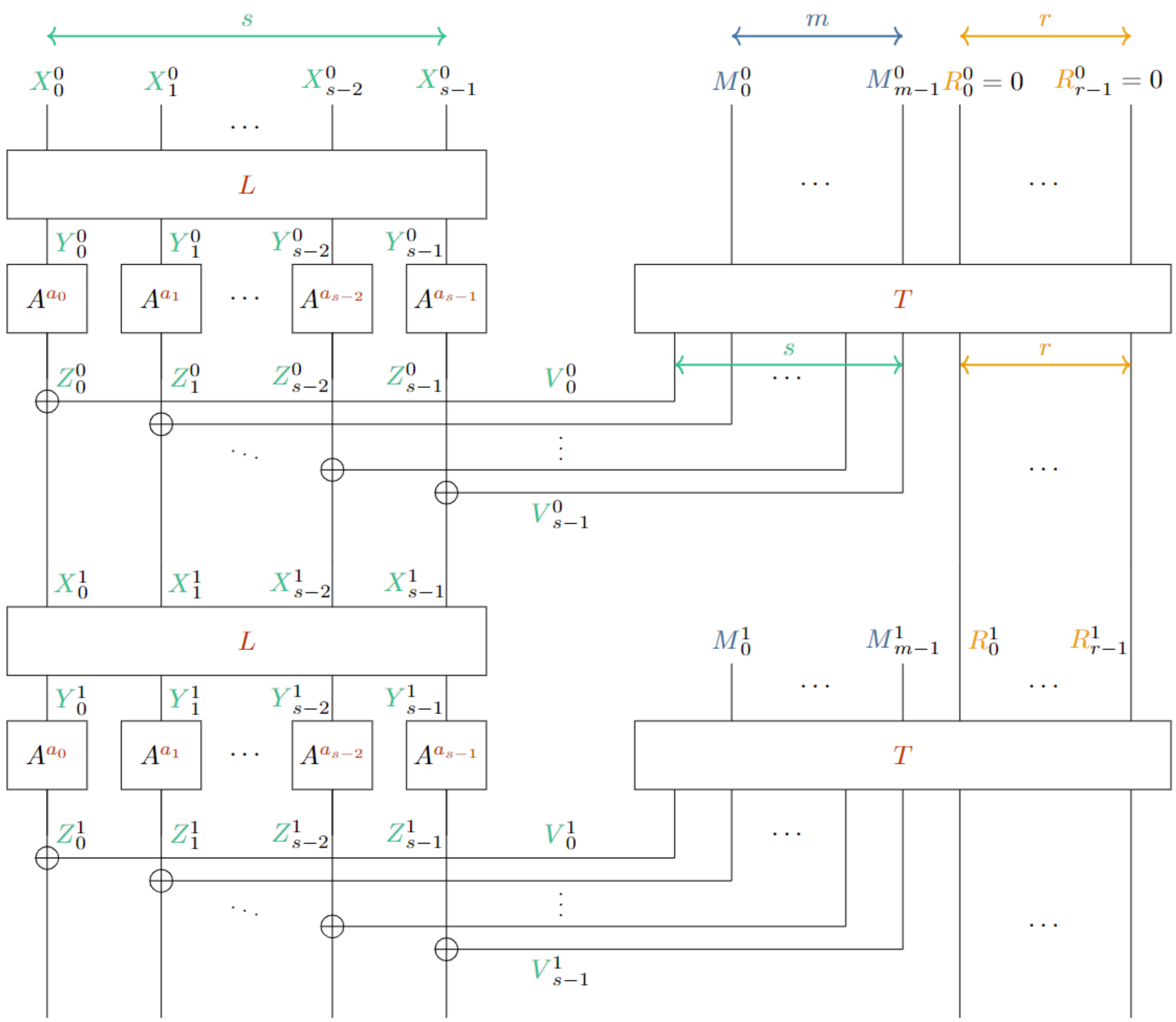


Our overall permutation structure

Goal: no differential path with Probability > 2^{-128}

AddRoundKey is free with AES-NI: we can use a free XOR after each AES round

Increasing r and s generally improves performance, but we limit to s + r <16



A is AES round, T and L are linear matrices

Automatic security and performance analysis

Automatic security analysis:

- a MILP model to evaluate diff. paths automatically without linear incompatibilities (cheap)
- another MILP model with linear incompatibilities (quite expensive)

Automatic performance benchmark: an automatic implementation is produced for each candidate (quite cheap) to benchmark them.

- so performant that XOR becomes important (carefully consider AES-NI / XOR latency, throughput, ports). For x AES rounds, make x/2 XOR max (unlike Jean-Nikolic or Rocca).
- Dependency chains are also important: Rocca in decryption has long chains (reduced perf.)
- Many other complex things to consider, so the best way is to actually benchmark directly

Architecture	Instr	Latency	Throughput	P_0	P_1	P_2	P_3	P_4	P_5	P_6
Intel Haswell	XOR	1	0.33	x	x				x	
	AESENC	7	1						x	
Intel Skylake	XOR	1	0.33	x	x				x	
	AESENC	4	1	x						
Intel Ice Lake	XOR	1	0.33	x	x				x	
	AESENC	3	0.5	x	x					
Intel Tiger Lake	XOR	1	0.33	x	x				x	
	AESENC	3	0.5	x	x					
AMD Zen 1/2/3/4	XOR	1	0.25	x	x	x	x			
	AESENC	4	0.5	x	x					

Scheduling of AESENC and XOR instructions on modern processors

Handling a large search space

Extremely large search space, so we reduce it by:

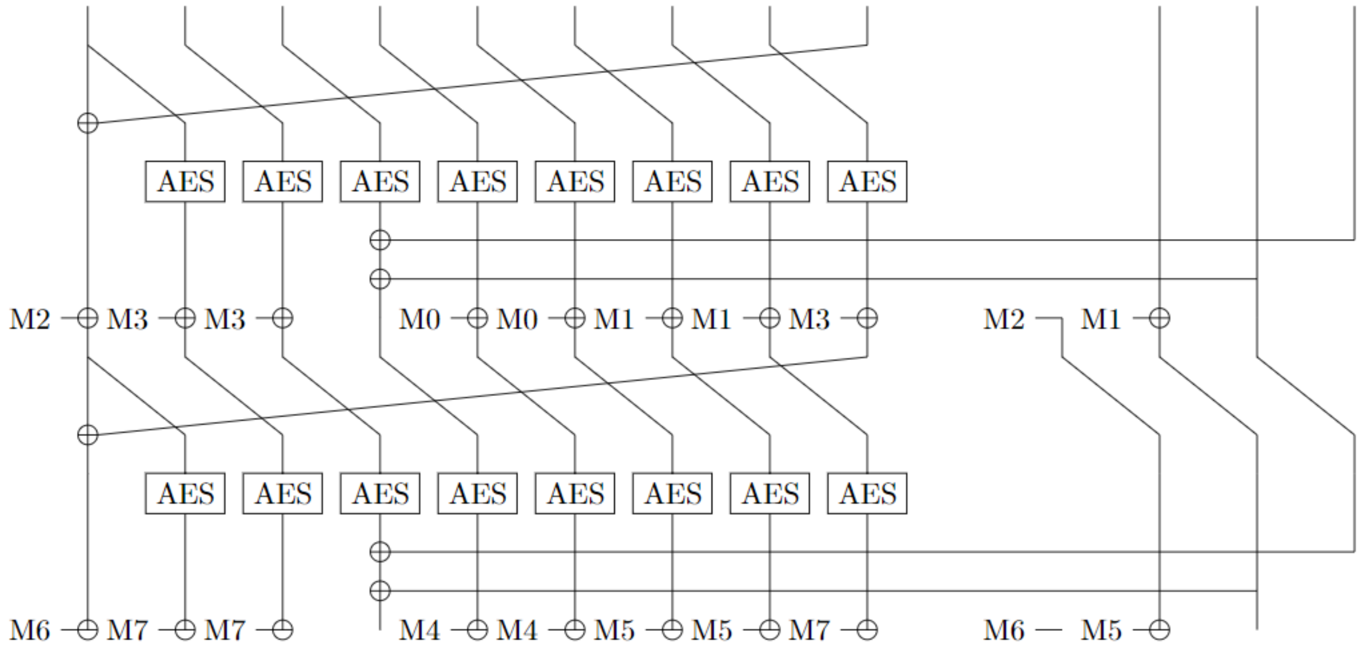
- leveraging symmetries
- select subparts that are interesting (limit #XORs, higher diffusion matrices)

Our search strategy (NEW):

LeMac (128-bit key / 128-bit tag)

- The state is composed of **13 128-bit words** (9 in non-linear part, 4 in linear)
- 8 AES rounds for 4 message blocks (rate 2), only 4 extra XORs (**perfect ratio**)
- **Security:** at least **26 active Sboxes** (diff. path probability < $2^{-6*26} = 2^{-156}$)

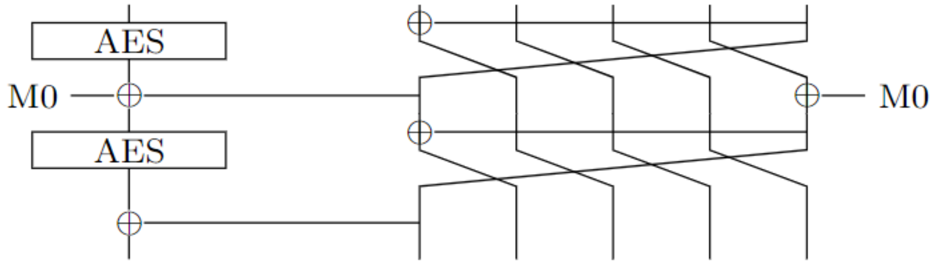
2 rounds of the UHF of LeMac



PetitMac (128-bit key / 128-bit tag)

- The state is composed of **6 128-bit words** (1 in non-linear part, 5 in linear)
- 2 AES rounds for 1 message block (rate 2), 3 extra XORs
- **Security:** at least **26 active Sboxes** (diff. path probability < $2^{-(26*6)} = 2^{-156}$)

1 round of the UHF of PetitMac



Performance results

< 0.1 c/B throughput for LeMac ! (Using only 128-bit instructions, not AVX-512).

The **fastest MAC** (by far) on medium/high-end processors.

PetitMAC aims for a better **tradeoff** on constrained devices: AES round-based MAC with rate 2, with acceptable memory footprint.

18.3 c/B on ARM Cortex-M4.

		Speed (c/B)		
CPU	Cipher	1kB	16kB	256kB
Intel Haswell (Xeon E5-2630 v3)	GCM (AD only)	1.138	0.700	0.605
	Rocca (AD only)	0.602	0.225	0.201
	Rocca-S (AD only)	0.660	0.290	0.269
	AEGIS128 (AD only)	0.809	0.578	0.564
	AEGIS128L (AD only)	0.542	0.299	0.285
	Tiaoxin-346 v2 (AD only)	0.489	0.207	0.190
	Jean-Nikolić	0.455	0.149	0.159
	LeMac	0.498	0.148	0.131
	PetitMac	1.116	0.890	0.876
Intel Skylake (Xeon Gold 6130)	GCM (AD only)	0.817	0.396	0.370
	Rocca (AD only)	0.573	0.190	0.167
	Rocca-S (AD only)	0.568	0.213	0.192
	AEGIS128 (AD only)	0.682	0.470	0.460
	AEGIS128L (AD only)	0.505	0.267	0.253
	Tiaoxin-346 v2 (AD only)	0.473	0.206	0.189
	Jean-Nikolić	0.389	0.142	0.130
	LeMac	0.422	0.144	0.126
	PetitMac	0.792	0.635	0.626
Intel Ice Lake (Xeon Gold 5320)	GCM (AD only)	0.699	0.311	0.286
	Rocca (AD only)	0.528	0.171	0.149
	Rocca-S (AD only)	0.478	0.172	0.151
	AEGIS128 (AD only)	0.619	0.401	0.389
	AEGIS128L (AD only)	0.416	0.208	0.195
	Tiaoxin-346 v2 (AD only)	0.328	0.131	0.121
	Jean-Nikolić	0.307	0.126	0.113
	LeMac	0.289	0.082	0.068
	PetitMac	0.521	0.384	0.376

Code: https://github.com/AugustinBariant/Implementations_LeMac_PetitMac

Future of LeMac / PetitMac

- What about **(Authenticated)-Encryption** ?
- What about 256-bit keys (mandated by 6G) and 256-bit tags ?
- Probably **difficult to do faster**:
  - we are at the performance theoretical limit for rate 2
  - we proposed candidates with rate < 2, but practical performance is not improved
- Consider using LeMac/PetitMac as building blocks for amazing speed ! (NIST “Accordion cipher” ?)

Low-Latency Cryptography

Under submission

Joint work with K. Hu., M. Khairallah and Q. Q. Tan

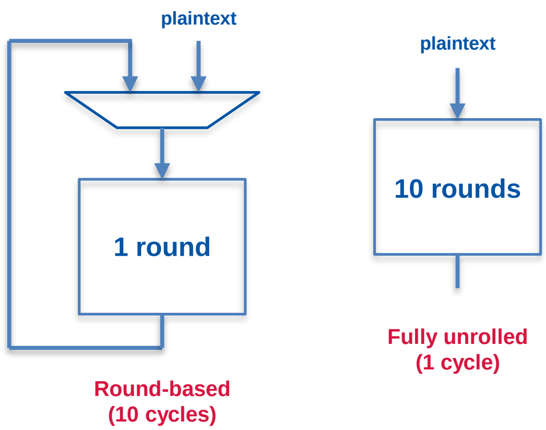
Why Low-latency

AES good for general usage, but lot of attention on lightweight cryptography in the past 15 years. NIST has standardized ASCON, **what's next ?**

In some applications, the **latency** (time it takes to produce the ciphertext byte/block of a corresponding plaintext byte/block) is very important:

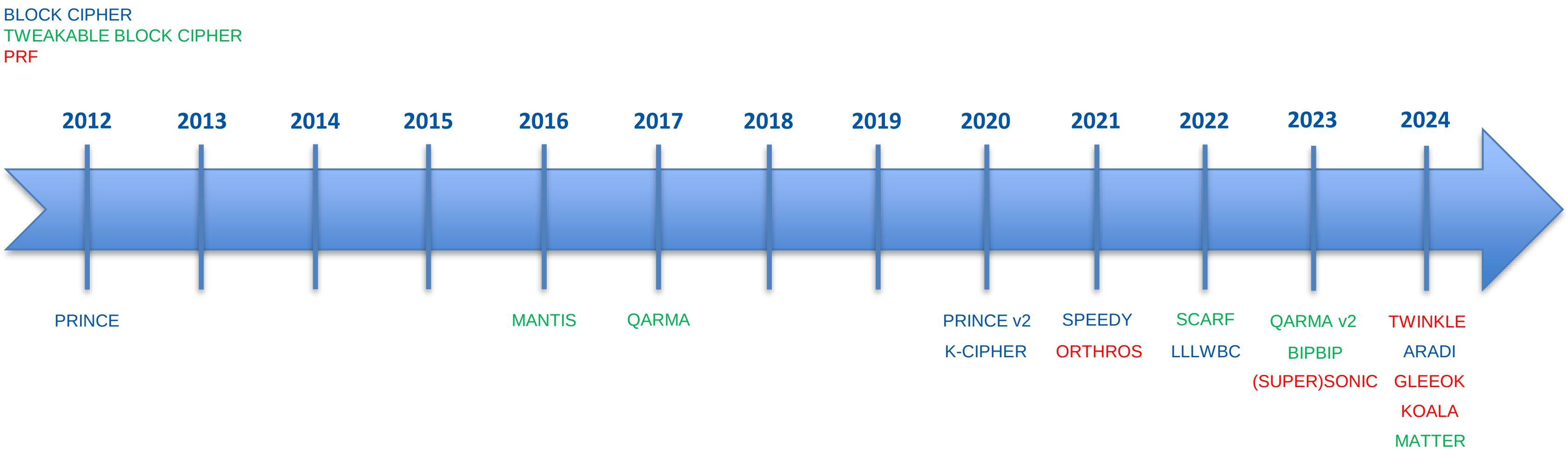
- RAM memory encryption/authentication (typically with a hardware memory encryption engine), especially with the rise of cloud computing,
- sensor data encryption/authentication (critical systems, automotive)
- system security (pointer authentication)

We talk about hardware (ASIC principally, or FPGA), with **fully unrolled implementations** (entire cipher in a single cycle, but lower freq.).



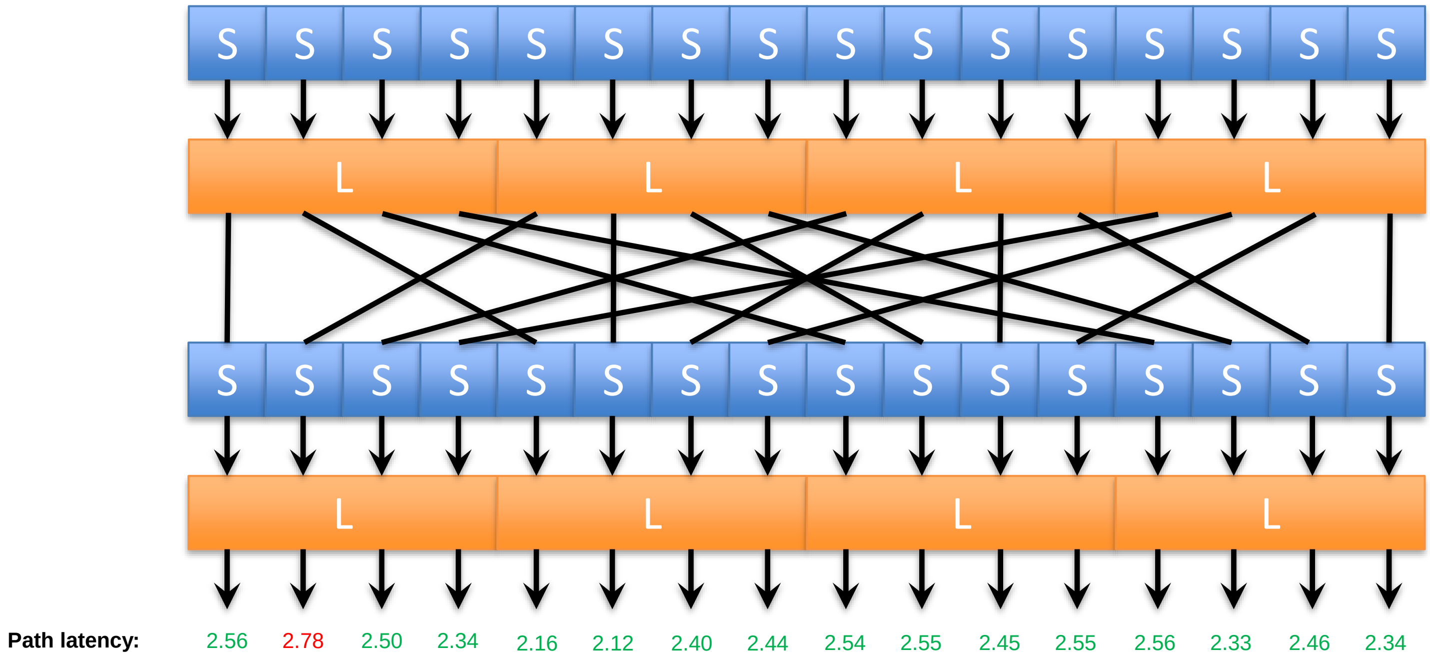
Here we consider the **internal primitive**, not the operating mode.

Low-latency cryptography timeline



- **PRINCE** was the first cipher to claim **latency** as main performance goal
- Low-latency trend is accelerating
- We now have BC, TBC, PRF candidates
- Design strategy is to use special Sboxes, linear layers, combinations of them, special structures, to reduce latency locally while maintaining security
- Special **operating modes** have also been proposed

Why Low-latency is difficult ?



In contrary to area/throughput, **it is difficult to predict the latency accurately in practice.**

It is also **difficult to know in advance the critical path** of the implementation and the impact that a change on one internal component might do to the latency.

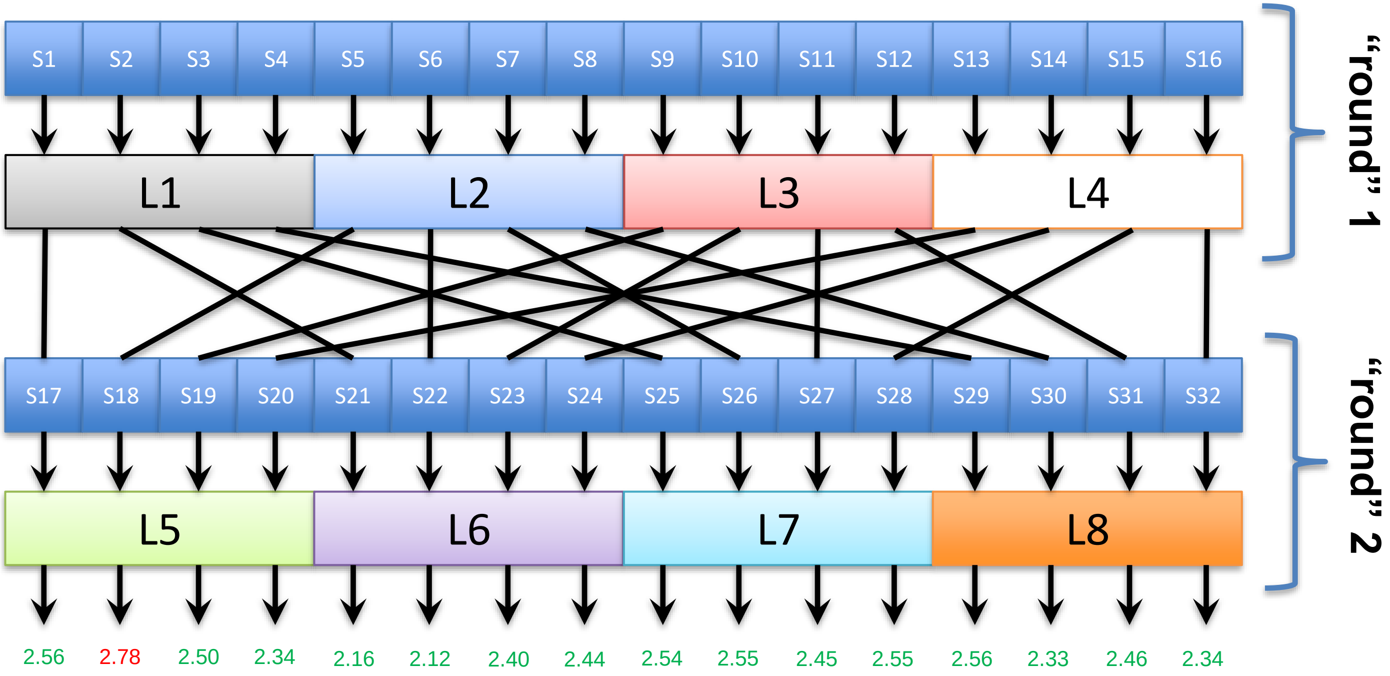
Breaking the iterative round paradigm

Low latency ciphers are used with **unrolled implementation**, so <u>**no need to follow a classical round structure anymore**</u> **(NEW)** !

Problem: the security analysis becomes difficult for humans
Solution (NEW): let automated cryptanalysis guide the design !

Two benefits:

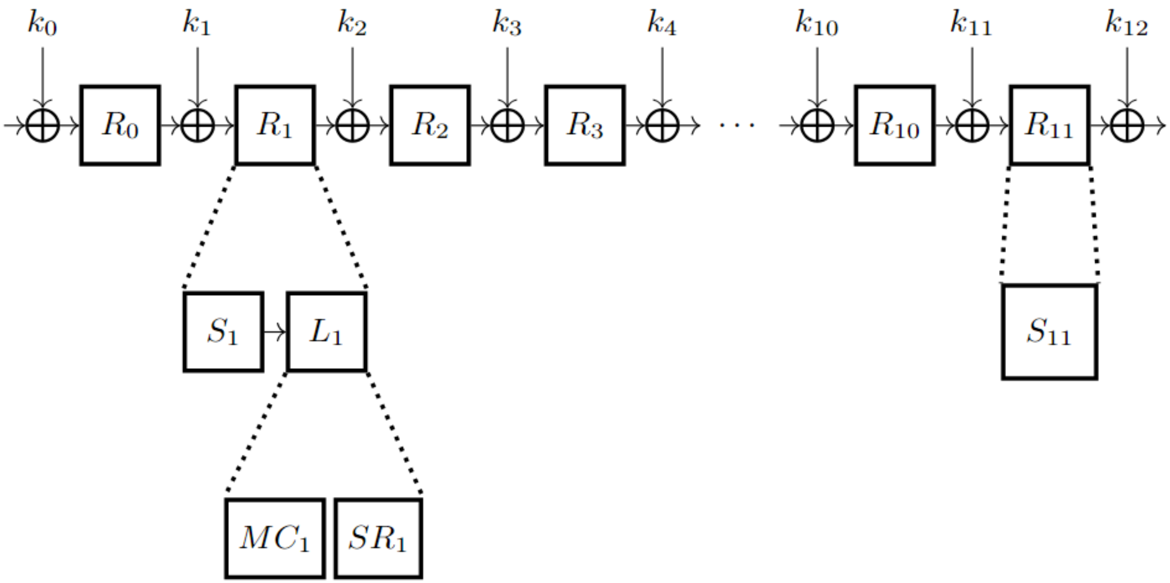
- One can create the cipher **round per round**
- We can adapt each round (and each component within a round) separately to **minimize the max path latency**



The uKNIT Cipher

The **uKNIT extremely low-latency block cipher** structure:

- Classical **64-bit SPN,** with sixteen **4-bit low-latency Sboxes**, each can be different (bit-permuted variants of the MANTIS Sbox)
- Special **low-latency linear layers**
- **Each round can be different !**
- **Key Schedule: New** generalization of the STK construction



Building the cipher: Evolutionary Algorithm

Problem: the **search space is now VERY large** (sboxes, linear layers)

Solution: we use an **evolutionary algorithm** to search in that large space, optimizing for good latency/security tradeoff.

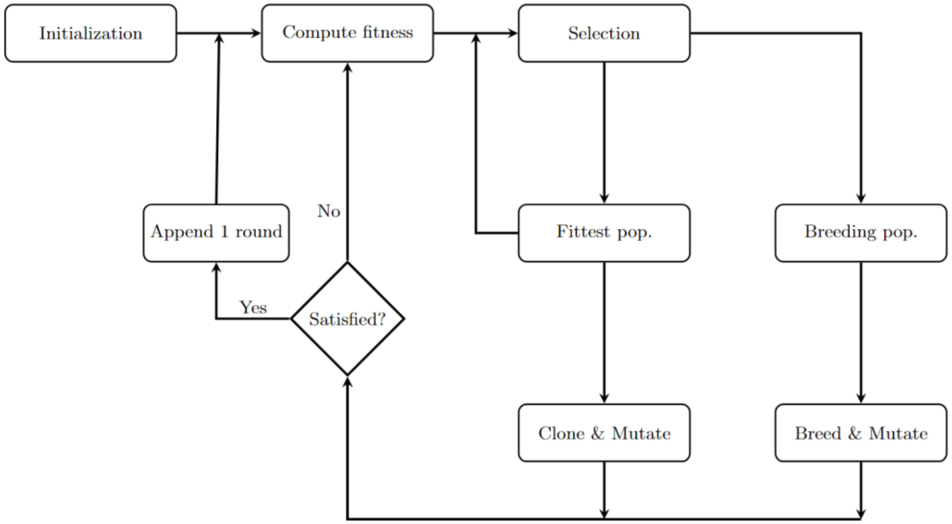
Importance of the **objective function**:

- If too latency oriented, not good
- If too security oriented, not good

$$\frac{\max[-\log_2(prob_d), -2 \cdot \log_2(bias_l)]^2}{lat}$$

We start from good candidates on 3 rounds. Then, we proceed **round per round** until reaching 12 rounds.

Our design is fully automated (almost **NEW** [Nikolić 2017])



Security of uKNIT

uKNIT has a **good resistance against differential and linear cryptanalysis**.

We also studied many other state-of-the-art cryptanalysis.

Stronger diff/linear resistance than PRINCE.

Differential probabilities for all windows of r-round

r \ i	0	1	2	3	4	5	6	7	8	9	10	11	PRINCE
1	2	2	2	2	2	2	2	2	2	2	2	2	–
2	8	8	6	6	8	8	6	8	8	6	8	–	–
3	14	12	12	12	14	14	12	14	12	12	–	–	–
4	25	23	24	26	30	26	26	24	24	–	–	–	32
5	40	40	39	40	40	39	37	37	–	–	–	–	39
6	49	48	46	46	50	47	49	–	–	–	–	–	44
7	60	58	52	61	60	59	–	–	–	–	–	–	56
8	71	70	68	71	72	–	–	–	–	–	–	–	66
9	81	82	80	82	–	–	–	–	–	–	–	–	74
10	94	87	92	–	–	–	–	–	–	–	–	–	80
11	101	99	–	–	–	–	–	–	–	–	–	–	89
12	113	–	–	–	–	–	–	–	–	–	–	–	99

Linear correlations for all windows of r-round

r \ i	0	1	2	3	4	5	6	7	8	9	10	11	PRINCE
1	1	1	1	1	1	1	1	1	1	1	1	1	–
2	4	4	3	3	4	4	3	4	4	3	4	–	–
3	7	6	6	6	7	6	6	7	6	6	–	–	–
4	13	10	11	13	14	12	12	11	12	–	–	–	16
5	19	18	19	19	19	18	17	17	–	–	–	–	19
6	24	23	22	23	25	23	21	–	–	–	–	–	22
7	29	26	26	30	29	27	–	–	–	–	–	–	27
8	35	34	34	34	34	–	–	–	–	–	–	–	32
9	39	38	37	39	–	–	–	–	–	–	–	–	34
10	45	44	43	–	–	–		–	–	–	–	–	38
11	49	50	–	–	–	–	–	–	–	–	–	–	41
12	55	–	–	–	–	–	–	–	–	–	–	–	49

Performance

uKNIT breaks new records for low-latency:

~ 10% **reduced latency** vs PRINCEv2

~ 20% **reduced area** vs PRINCEv2

~ >10% **increased security** (-$\log_2$ of differential probability) vs PRINCEv2

	Name	Block Size	Latency (ns)	Area (μm^2)
FIL-PRF	Gleeok128 [3]	128	3.45	73,078.92
		128	1.61	133,343.99
	Orthros [7]	128	2.66	40,932.36
		128	1.59	77,437.08
TBC	BipBip [12]	24	4.03	39,278.52
		24	1.45	60,630.12
	SPEEDY 7 rnds [78]	192	3.75	46,826.64
		192	1.79	88,331.04
	Qarmav1 9 rnds [4]	128	4.84	42,787.08
		128	2.74	94,944.23
Public Perm.	KoalaP [2]	64	1.46	24,104.88
		64	1.16	52,965.36
BC	PRINCEv2 [36]	64	2.90	12,006.72
		64	1.65	27,564.12
	uKNIT-BC (with side loading)	64	2.58	10,685.88
		64	1.64	14,587.92
		64	1.49	21,779.27
	uKNIT-BC	64	2.53	15,859.80
		64	1.64	22,963.67
		64	1.48	30,436.20

Hardware implementation benchmarks on TSMC 65nm

Future

- **uKNIT**: **lowest latency with good security**. Very competitive compared to the state-of-the-art
- More search can probably find a slightly better candidate, but probably not much
- Can be used as building block for larger primitives
- Our **design strategy** can be reused for other use-cases or primitives

Conclusion

Conclusion

- We will see **more automated cryptanalysis during design** phase
- Automation allows **design strategies that wouldn't be possible before**
- Performance gain is still possible in symmetric-key crypto design
- We tend to concentrate on complexity reduction to judge quality of automated cryptanalysis (i.e. $2^{20.5}$ is better than 2^{21}), but **the simplicity and ease-of-use of automated cryptanalysis is undervalued**

Thank You !