The State of the Art in Symmetric Lightweight Cryptography

Léo Perrin
Based on a joint work with Alex Biryukov

November 18, 2017
Cryptacus Workshop

La programmation des milliards de processeurs équipant tous nos objets, qui doit prendre en compte des dispositifs très bon marché mais peu sûrs, devant par exemple développer des algorithmes de cryptographie faible, constitue un défi

Taken from a document written originally in English.

The programming of billions of processors embedded in all our devices, which must take into account devices that are very cheap and poorly secured, that require for instance the implementation of weak cryptographic algorithm, is a challenge...

Translation

Weak Cryptography?

Weak ≠ Lightweight

Weak Cryptography?

Weak ≠ Lightweight

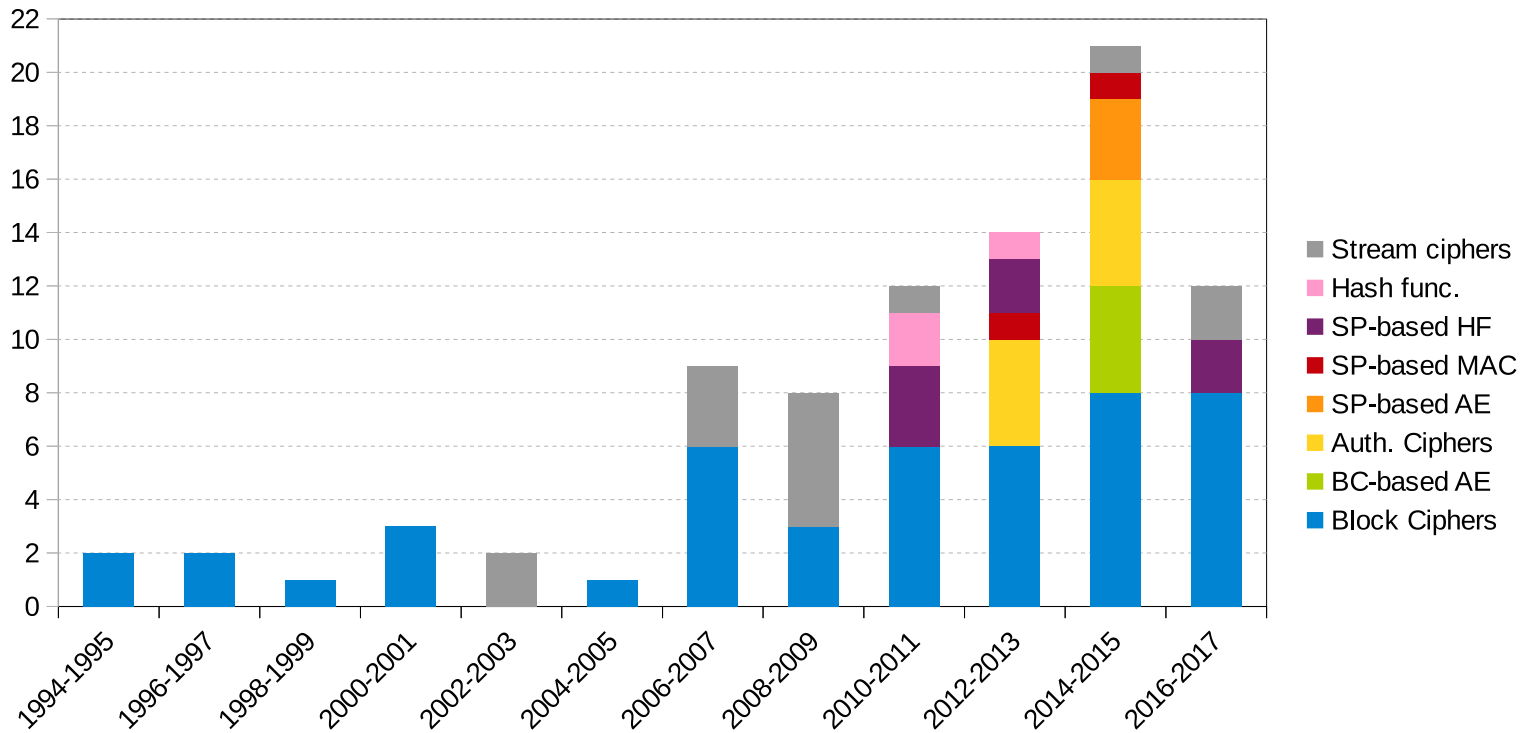
What is lightweight (symmetric) cryptography?

It is vast (1/2)

	Stream C.	Block C.	Hash F.	Auth. C.	MAC	Total
Academia	14	50	10	10	2	86
Proprietary	17	5	0	0	1	23
Government	1	5	0	0	0	6
Total	32	60	10	10	3	**115**

It is vast (1/2)

	Stream C.	Block C.	Hash F.	Auth. C.	MAC	Total
Academia	14	50	10	10	2	86
Proprietary	17	5	0	0	1	23
Government	1	5	0	0	0	6
Total	32	60	10	10	3	**115**



It is vast (2/2)

Several scattered
national/international standards,
none chosen after a competition
(apart from the AES).

It is vast (2/2)

Several scattered national/international standards, none chosen after a competition (apart from the AES).

State of the Art in Lightweight Symmetric Cryptography,

Alex Biryukov and Léo Perrin

`https://ia.cr/2017/511`

`http://cryptolux.org`

Description		Parameters			Fraction of Rounds Attacked		
Name	Ref.	Key	Block	Rounds	SK	RK	Ref.
3-Way	[DGV94]	96	96	11	0.45	**1**	[DGV94, KSW97]
RC5	[Riv95]	0–2040	32/64/128	12	**1**	**1**	[BK98]
Misty1	[Mat97]	128	64	8	**1**	**1**	[Tod17, BOK16]
XTEA	[NW97]	128	64	64	0.36	0.56	[SMVP11, Lu09]
AES	[DR98]	128/192/256	128	10/12/14	0.70	**1**	[DFJ13, BK09]
BKSQ	[DR00]	96	96/144/192	10/14/18	0.60	0.60	[DR00]
Khazad	[BR00]	128	64	8	0.75	**1**	[BKP16, BN10]
Noekeon	[DPVAR00]	128	128	16	0.75	–	[DPVAR00]
Iceberg	[SPR+04]	128	64	16	0.50	0.50	[SWJS12]
mCrypton	[LK06]	64/96/128	64	12	0.83	0.83	[DF16]
HIGHT	[HSH+06]	128	64	32	0.84	**1**	[WWBC14, KHK11]
SEA	[SPGQ06]	96	96	≥ 105	$\emptyset$	$\emptyset$	$\emptyset$
CLEFIA	[SSA+07]	128/192/256	128	18/22/26	0.72	0.72	[MDS11]
DESLX	[LPPS07]	184	64	16	$\emptyset$	–	$\emptyset$
PRESENT	[BKL+07]	80/128	64	31	0.87	0.87	[BN14]
MIBS	[ISSK09]	64/80	64	32	0.59	0.59	[BHV14]
KATAN	[CDK09]	80	32/48/64	254	0.60	0.60	[FM15]
KTANTAN	[CDK09]	80	32/48/64	254	**1**	**1**	[BR11]
GOST revisited	[PLW10]	256	64	32	**1**	**1**	[Iso13]
PRINTCipher	[KLPR10]	48/96	80/160	48/96	**1**	**1**	[LAAZ11]
EPCBC	[YKPH11]	96	48/96	32	**1**	**1**	[Bul13]
KLEIN	[GNL11]	64/80/96	64	12/16/20	**1**	**1**	[LN15]
LBlock	[WZ11]	80	64	32	0.75	0.75	[WWJ16]
LED	[GPPR11]	64/128	64	32/48	0.66	0.66	[DDKS16]
Piccolo	[SIH+11]	80/128	64	25/31	0.68	0.68	[SIH+11, Min13]
PICARO	[PRC12]	128	128	12	$\emptyset$	**1**	[CLNP16]
PRINCE	[BCG+12]	128	64	12	0.83	–	[CFG+15, DP15]
ITUbee	[KDH13]	80	80	20	0.40	0.40	[Sol15]
TWINE	[SMMK13]	80/128	64	36	0.69	0.75	[BDP15, BBR+13]
Zorro	[GGNS13]	128	128	24	1.0	1.0	[BDD+15]
PRIDE	[ADK+14]	128	64	20	0.90	0.90	[LR17]
Joltik † ‡	[JNP14]	64/80/96/128	64	24/32	0.75	0.75	[JNP14]
LEA	[HLK+14]	128/192/256	128	24/28/32	0.58	0.58	[SHY16]
iScream † ‡	[GLS+14]	128	128	12/14	**1**	**1**	[LMR15, TLS16]
LBlock-s †	[ZWW+14]	80	64	16/32	**1**	**1**	[Leu16a]
Scream † ‡	[GLS+14]	128	128	10/12	**1**	**1**	[TLS16]
Lilliput	[BFMT15]	80	64	30	0.57	0.57	[ST17]
RECTANGLE	[ZBL+15]	80/128	64	25	0.72	0.72	[ZBL+15]
Fantomas	[GLSV15]	128	128	12	$\emptyset$	–	$\emptyset$
Robin	[GLSV15]	128	128	16	**1**	–	[LMR15]
Midori	[BBI+15]	128	64/128	16/20	**1**	**1**	[TLS16]
SIMECK	[YZS+15]	64/96/128	32/48/64	32/36/44	0.80	0.80	[QHS17]
RoadRunneR	[BŞ16]	80/128	64	10/12	0.60	–	[BŞ16]
FLY	[KG16]	128	64	20	$\emptyset$	$\emptyset$	$\emptyset$
Mantis ‡	[BJK+16]	128	64	14	0.71	–	[DEKM16]
SKINNY ‡	[BJK+16]	64–384	64/128	32–56	0.59	0.59	[ABC+17, LGS17]
SPARX	[DPU+16]	128/256	64/128	24–40	0.69	0.69	[DPU+16, ATY17]
Mysterion	[JSV17]	128/256	128/256	12	$\emptyset$	–	$\emptyset$
Qarma ‡	[Ava17]	128/256	64/128	16/24	$\emptyset$	–	$\emptyset$

Outline

Goal of this Talk

We will look at several “lightweight” algorithms and see what they can tell us about lightweightness.

Outline

Goal of this Talk

We will look at several “lightweight” algorithms and see what they can tell us about lightweightness.

1. A5-GCM-1 and A5-GCM-2 — What *not* to do

Outline

Goal of this Talk

We will look at several “lightweight” algorithms and see what they can tell us about lightweightness.

1. A5-GCM-1 and A5-GCM-2 — What *not* to do
2. Plantlet and LEA — Specialized algorithms

Outline

Goal of this Talk

We will look at several “lightweight” algorithms and see what they can tell us about lightweightness.

1. A5-GCM-1 and A5-GCM-2 — What *not* to do
2. Plantlet and LEA — Specialized algorithms
3. GIMLI — Multi-purpose algorithms

Outline

Plan of this Section

Satellite Phone Encryption

GSM Protocol (regular phone)

Cell phone communications in many countries (incl. Europe) are encrypted with A5/1. A5/2 was used for products sold outside Europe (e.g. Irak).

Satellite Phone Encryption

GSM Protocol (regular phone)

Cell phone communications in many countries (incl. Europe) are encrypted with A5/1. A5/2 was used for products sold outside Europe (e.g. Irak).

Satphone Standards

For satellite phones, there are two competing standards: GMR-1 and GMR-2, each with their own crypto.

Satellite Phone Encryption

GSM Protocol (regular phone)

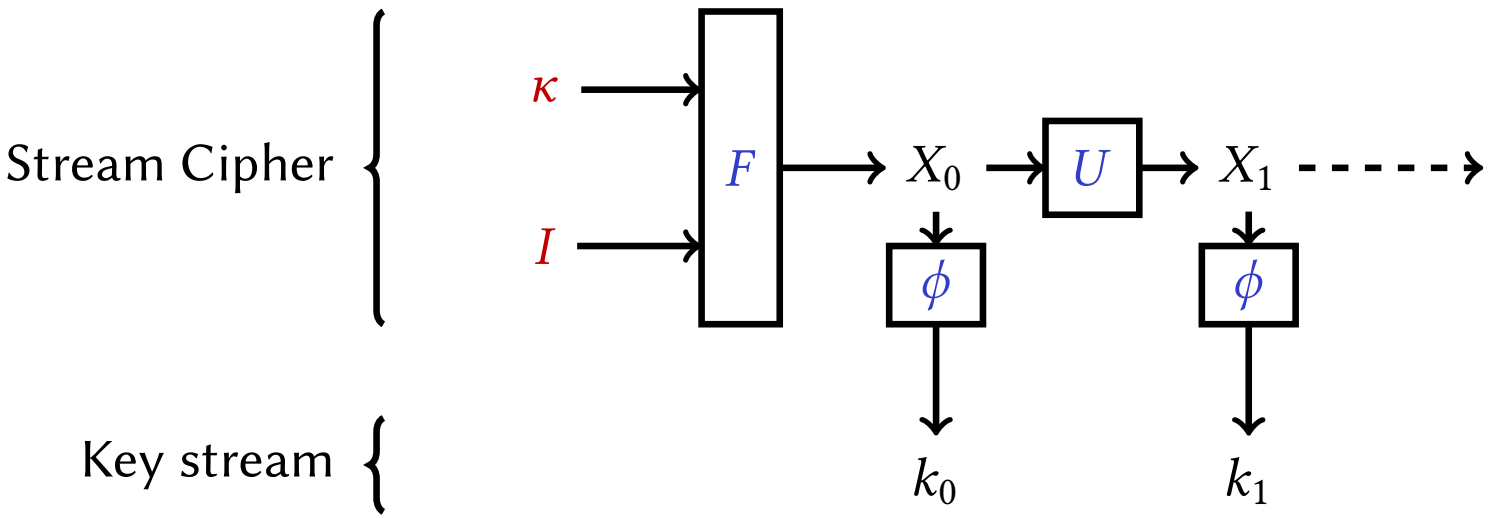
Cell phone communications in many countries (incl. Europe) are encrypted with A5/1. A5/2 was used for products sold outside Europe (e.g. Irak).

Satphone Standards

For satellite phones, there are two competing standards: GMR-1 and GMR-2, each with their own crypto.

Their crypto had to be reverse-engineered [DHW+12].

Stream Cipher



- κ: secret key
- I: IV
- X_i: internal state
- F: initialization
- U: state update function
- ϕ: filter

A5-GMR-1 (1/2)

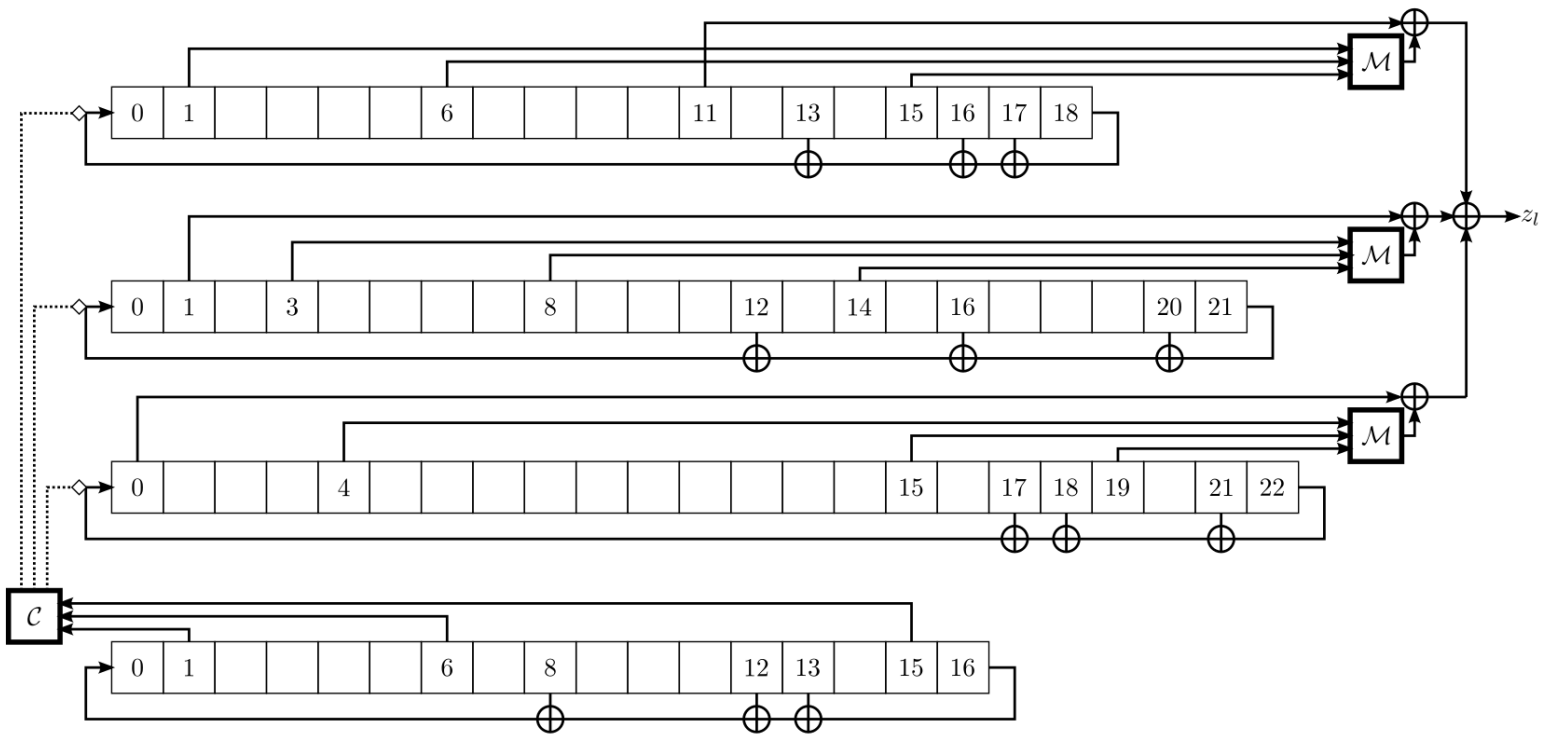


Diagram of A5-GMR-1 (from [DHW+12]).

Internal state size: 82 bits; key size: 64 bits; IV size: 19 bits.

A5-GMR-1 (2/2)

"Intuitive" characteristics of a LW algo

- Intended for low-power devices
- Very small internal state, very small key
- LFSRs → simple logic

Some operations are far cheaper than others.

Example

- LFSR: a handful of XORs
- Memory itself is expensive → small state

A5-GMR-2

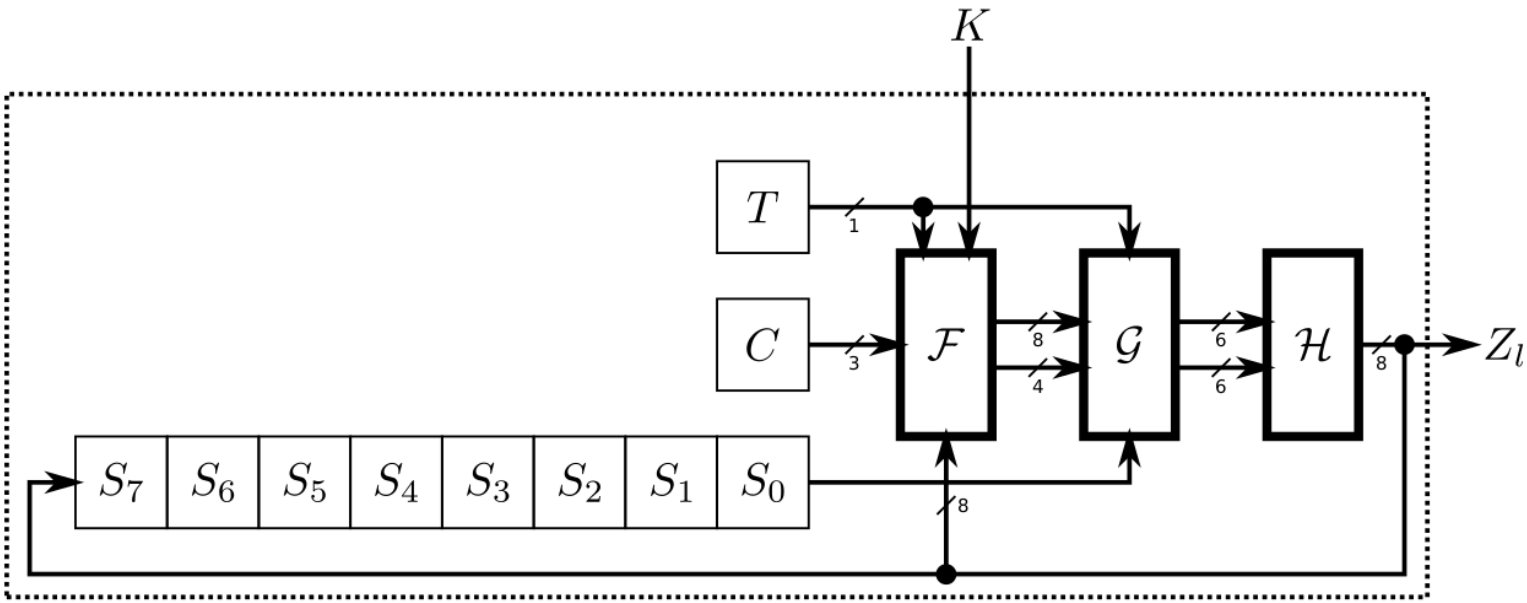


Diagram of A5-GMR-1 (from [DHW+12]).

Internal state size: 68 bits; key size: 64 bits; IV size: 22 bits.

Cryptanalysis

Are these algorithms secure?

Cryptanalysis

Are these algorithms secure?

No

In fact, A5-GMR-1 is based on A5/**2**!

Name	Things	Reference	Key	IS	IV	Att. time
A5/1	Cell phones	[And94]	64	64	22	2^{24}
A5/2		[BBK08]	64	81	22	2^{16}
CMEA †		[WSK97]	64	16–48	–	2^{32}
Oryx		[WSD+99]	96	96	–	2^{16}
A5-GMR-1	Satellite phones	[DHW+12]	64	82	19	$2^{38.1}$
A5-GMR-2		[DHW+12]	64	68	22	2^{28}
Dsc	Cordless phones	[LST+09]	64	80	35	2^{34}
SecureMem.	Atmel chips	[GvRVWS10]	64	109	128	$2^{29.8}$
CryptoMem.			64	117	128	2^{50}
Hitag2	Car key/ immobilizer	[VGB12]	48	48	64	2^{35}
Megamos		[VGE13]	96	57	56	2^{48}
Keeloq †		[BSK96]	64	32	–	$2^{44.5}$
Dst40 †		[BGS+05]	40	40	–	2^{40}
iClass	Smart cards	[GdKGV14]	64	40	–	2^{40}
Crypto-1		[NESP08]	48	48	96	2^{32}
Css	DVD players	[BD04]	40	42	–	2^{40}
Cryptomeria †		[BKLM09]	56	64	–	2^{48}
Csa-BC †	Digital televisions	[WW05]	64	64	–	2^{64}
Csa-SC			64	103	64	$2^{45.7}$
PC-1	Amazon Kindle	[BLR13]	128	152	–	2^{31}
SecurID ‡	Secure token	[BLP04]	64	64	–	2^{44}
E0	Anything	[FL01]	128	128	–	2^{38}
RC4		[Nob94]	128	2064	–	2^{32}

Why are they all broken?

- Too small key

Why are they all broken?

- Too small key

save space/export restriction

Why are they all broken?

- Too small key — save space/export restriction
- “Security through obscurity”

Why are they all broken?

- Too small key — save space/export restriction
- “Security through obscurity” — doesn’t work

Why are they all broken?

- Too small key — save space/export restriction
- "Security through obscurity" — doesn't work
- Overall bad design

Why are they all broken?

- Too small key — save space/export restriction
- "Security through obscurity" — doesn't work
- Overall bad design — not cryptographers/old

Lessons Learnt

Design

- There are cases where a dedicated lightweight algorithm is used.
- Implementation performance implies lower block/internal state size.
- Usually only one functionnality/device.

Lessons Learnt

Design

- There are cases where a dedicated lightweight algorithm is used.
- Implementation performance implies lower block/internal state size.
- Usually only one functionnality/device.

Context

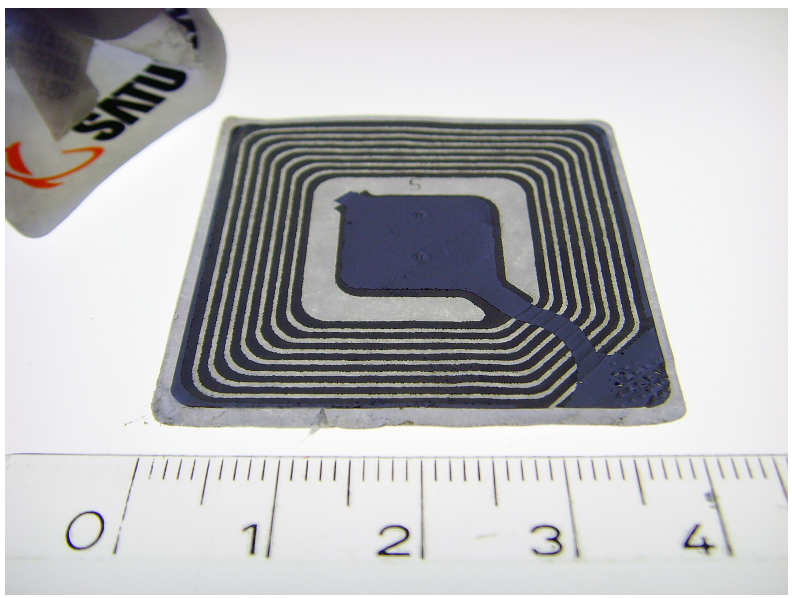
- Cryptography is hard.
- Export restrictions were a bad idea.
- Old algorithms stay for a while.

Outline

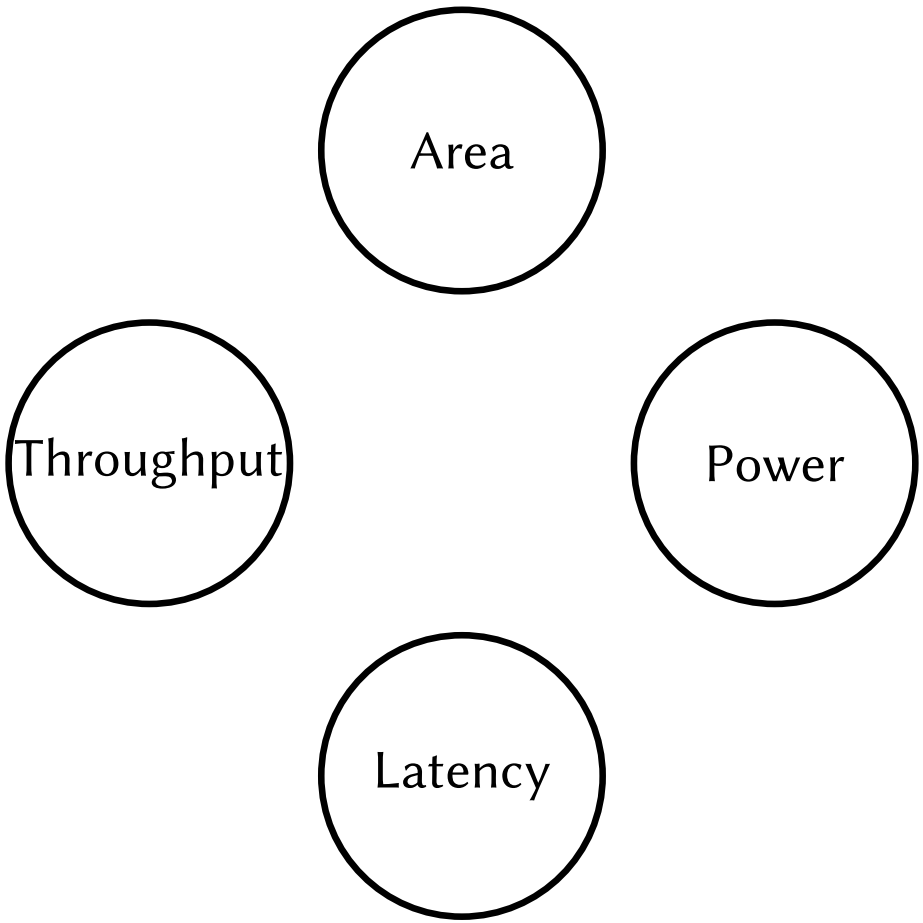
Plan of this Section

Targets

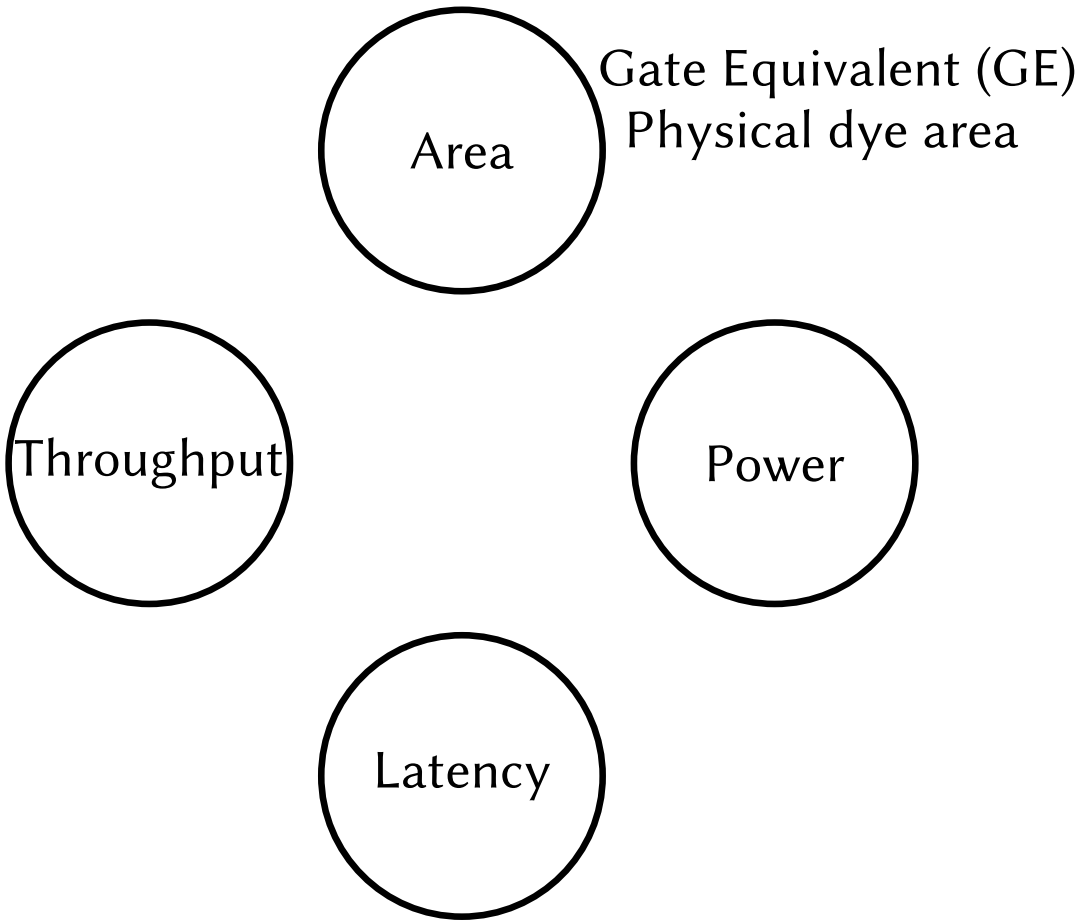
Hardware implementations are for
RFID tags, FPGA, hardware accelarators...



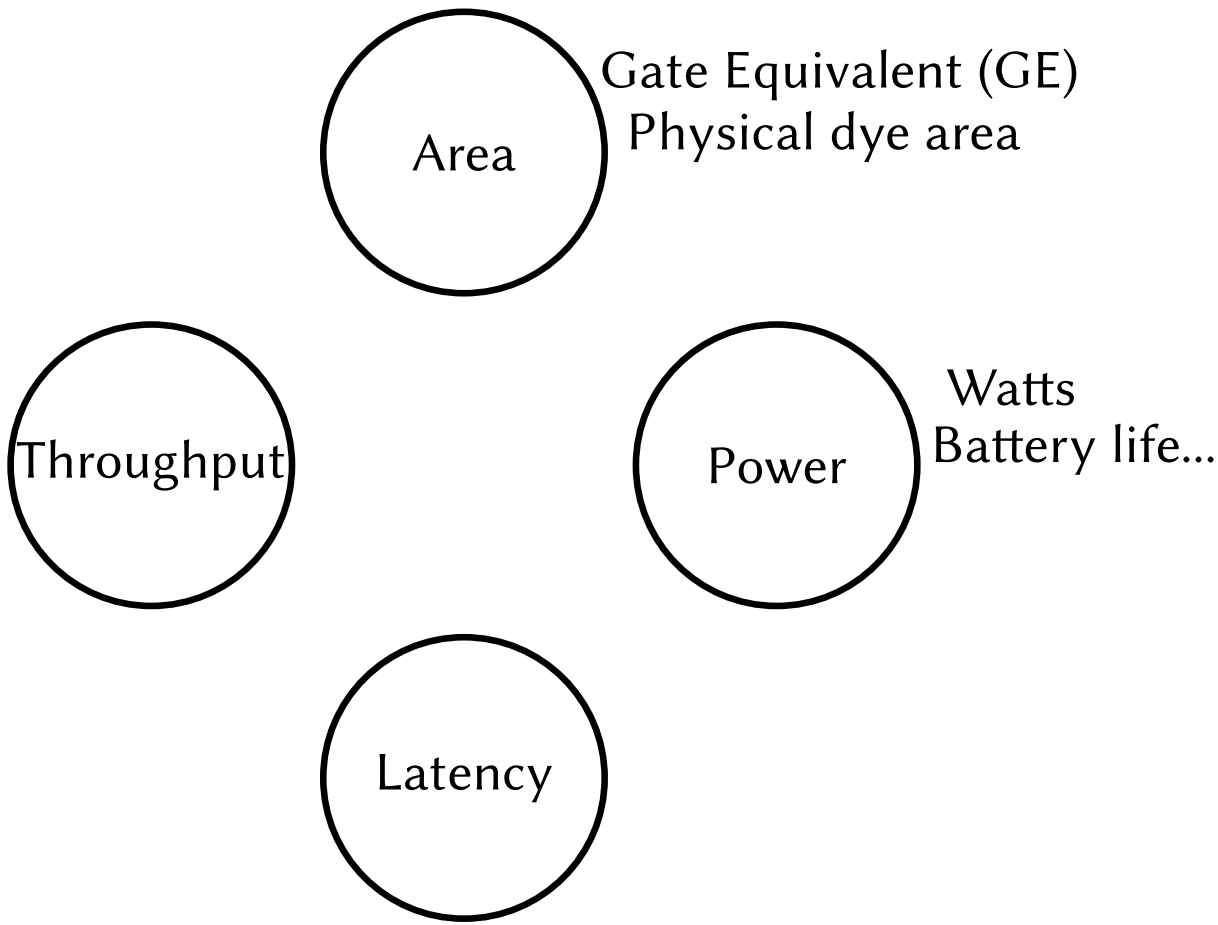
Core Trade-Off



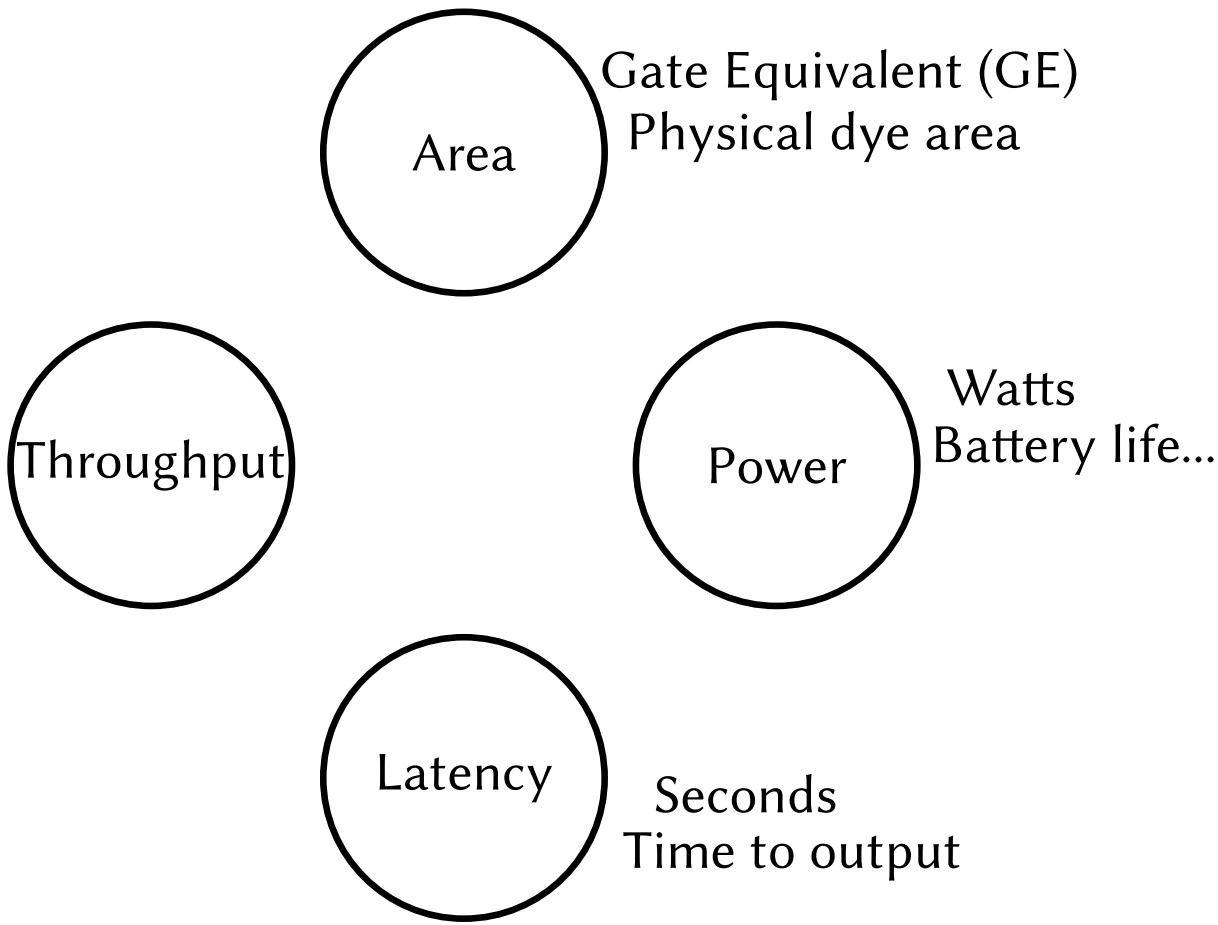
Core Trade-Off



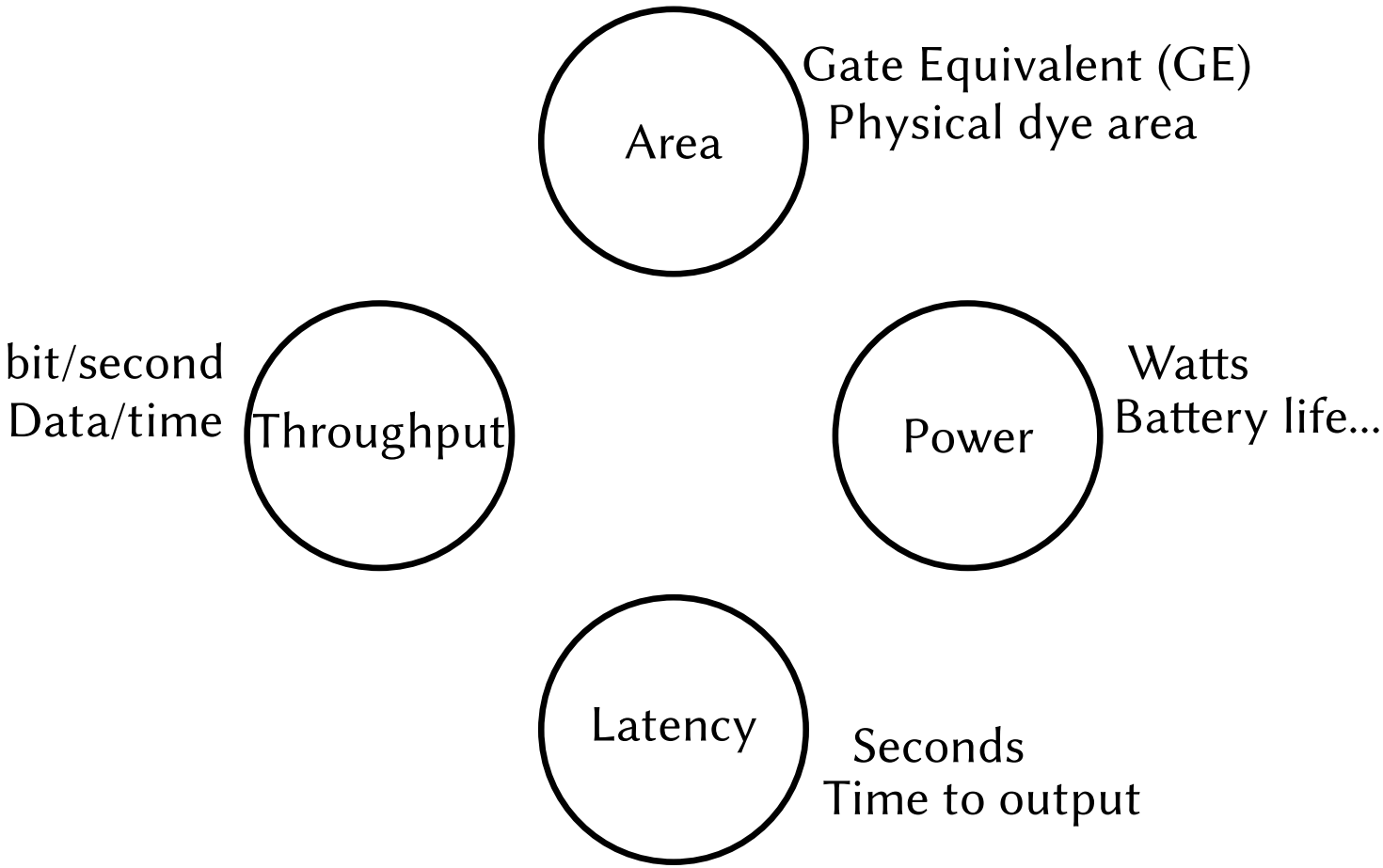
Core Trade-Off



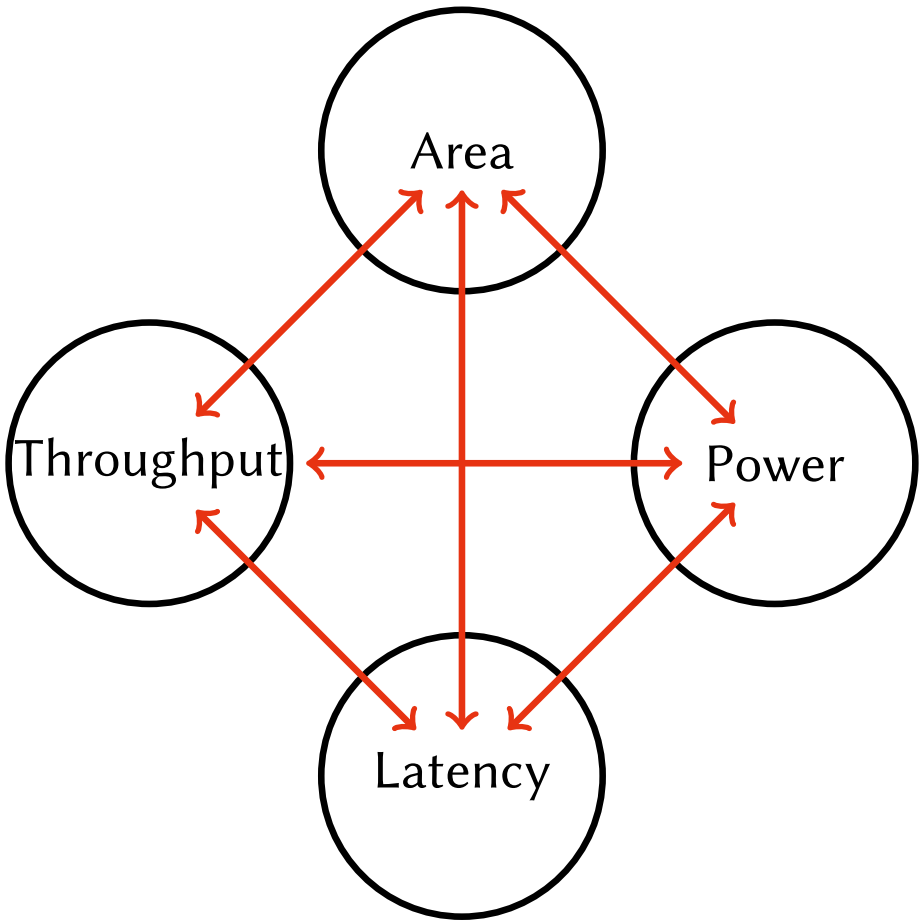
Core Trade-Off



Core Trade-Off

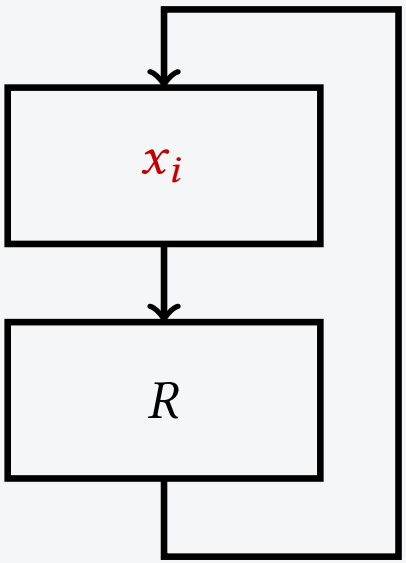


Core Trade-Off



Implementation Strategies

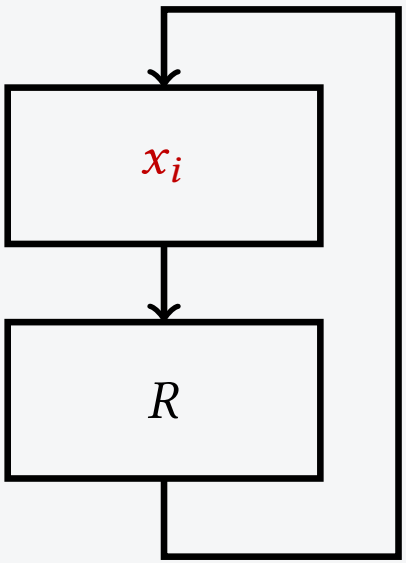
Round-based



- Low Area
- Higher Latency

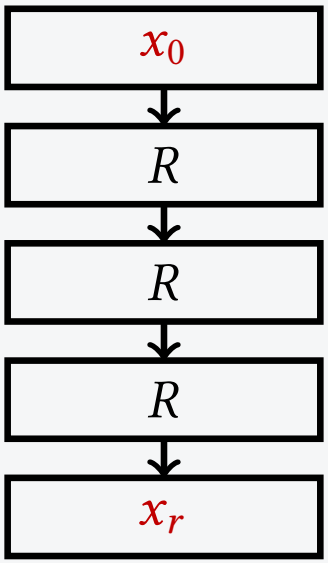
Implementation Strategies

Round-based



- Low Area
- Higher Latency

(Partially) Unrolled

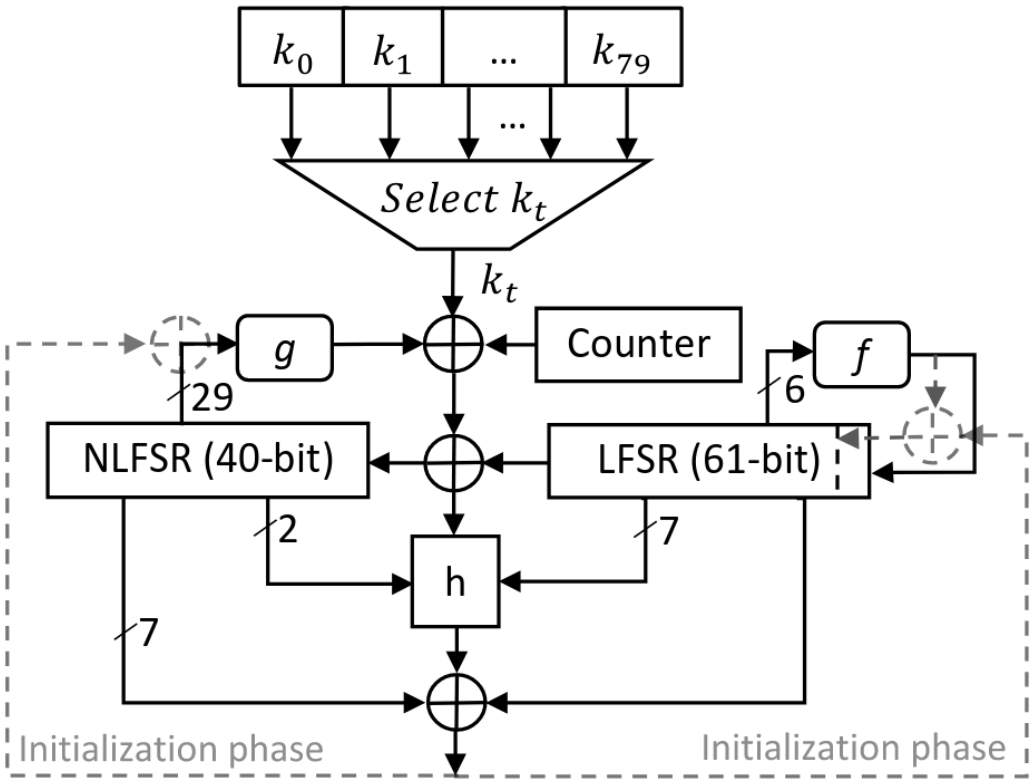


- Low latency
- High area

Specific Algorithms

Although implementation trade-offs are available, the algorithm design *itself* can facilitate some properties.

Description of Plantlet



Key size: 80 bits; Internal state size: 110 bits; IV size: 90 bits

A Cipher for Low Area

- Plantlet is a “fixed” Sprout.

A Cipher for Low Area

- Plantlet is a “fixed” Sprout.
- LFSR/NLFSR → very few gates.

A Cipher for Low Area

- Plantlet is a "fixed" Sprout.
- LFSR/NLFSR → very few gates.
- f, g, h carefully chosen

A Cipher for Low Area

- Plantlet is a "fixed" Sprout.
- LFSR/NLFSR → very few gates.
- f, g, h carefully chosen
- Small internal state (110 bits)

A Cipher for Low Area

- Plantlet is a "fixed" Sprout.
- LFSR/NLFSR → very few gates.
- f, g, h carefully chosen
- Small internal state (110 bits)
- Key state is unchanged → even fewer gates

What Plantlet Illustrates

An algorithm can be tailored for a specific implementation optimization.

The perfect algorithm would allow any implementation trade-off but in practice:

optimal for niche ≠ OK in most contexts

What Plantlet Illustrates

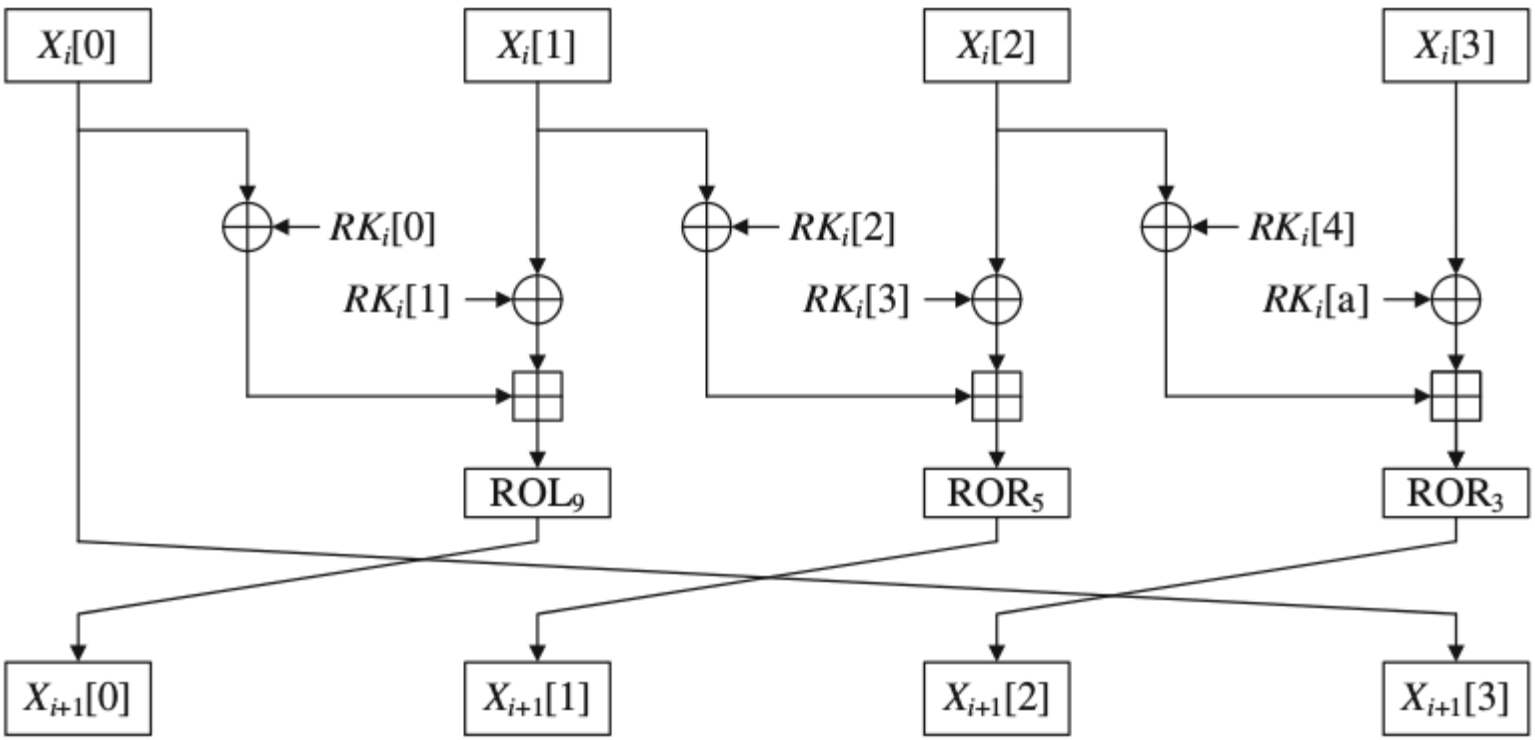
An algorithm can be tailored for a specific implementation optimization.

The perfect algorithm would allow any implementation trade-off but in practice:

optimal for niche ≠ OK in most contexts

Plantlet, SKINNY... Low area.
PRINCE, Mantis... Low latency.
Midori... Low energy.
Zorro... Easy SCA counters.

Description of LEA



Key size: 128/192/256 bits; Block size: 128 bits;

Felics framework

Table 2: The current best FELICS results for scenario 2: counter mode encryption of 128 bits.

General info			AVR (8-bit)			MSP (16-bit)			ARM (32-bit)			FoM
Name	block	key	Code	RAM	Time	Code	RAM	Time	Code	RAM	Time	
Chaskey	128	128	770	84	1597	490	86	1351	178	80	614	4.7
SIMON	64	96	448	53	2829	328	48	1959	256	56	1003	4.8
SIMON	64	128	452	53	2917	332	48	2013	276	60	972	4.9
Chaskey-LTS	128	128	770	84	2413	492	86	2064	178	80	790	5.4
SIMON	64	96	600	57	4269	460	56	2905	416	64	1335	6.6
SIMON	64	128	608	57	4445	468	56	3015	388	64	1453	6.8
Lea	128	128	906	80	4023	722	78	2814	520	112	1171	7.6
Rectangle	64	128	602	56	4381	480	54	2651	452	76	2432	8.1
Rectangle	64	80	606	56	4433	480	54	2651	452	76	2432	8.1
SPARX	64	128	662	51	4397	580	52	2261	654	72	2338	8.3
SPARX	128	128	1184	74	5478	1036	72	3057	1468	104	2935	12.4
RC5-20	64	128	1068	63	8812	532	60	15925	372	64	1919	13.5
AES	128	128	1246	81	3408	1170	80	4497	1348	124	4044	14.1
Hight	64	128	636	56	6231	636	52	7117	670	100	5532	14.8
Fantomas	128	128	1712	76	9689	1920	78	3602	2184	184	4550	18.8
Robin	128	128	2530	108	7813	1942	80	4913	2188	184	6250	22.0

ARX

Highest ranking algorithms don't use S-Boxes

Addition/Rotation/XOR (ARX)

- "better" use of CPU instructions
- not great in hardware
- hard to study

And/Rotation/XOR

- Less software oriented
- Also good in hardware
- Can be easier to study

The algorithm design will allow/prevent implementation trade-offs.

Optimizing for software ≠ Optimizing for hardware

Lessons Learnt

- Lightweight algorithms allow optimized implementations.
- Optimizations criteria compete against one another, even at the algorithm design level.
- Benchmarking is hard.
- Optimizing for software ≠ optimizing for hardware

Outline

Plan of this Section

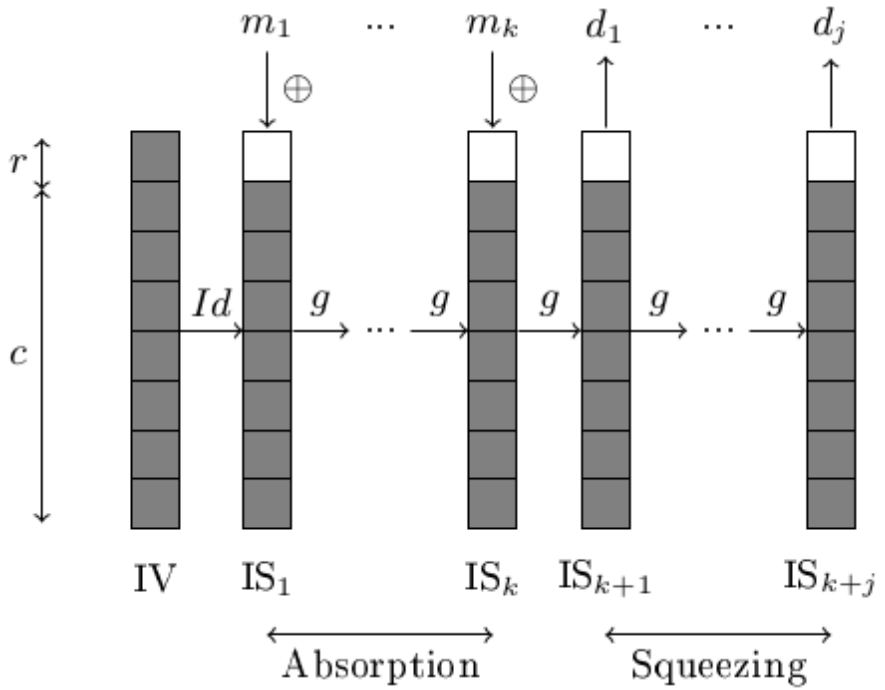
Designers' Aims

Abstract. This paper presents Gimli, a 384-bit permutation designed to achieve high security with high performance across a broad range of platforms, including 64-bit Intel/AMD server CPUs, 64-bit and 32-bit ARM smartphone CPUs, 32-bit ARM microcontrollers, 8-bit AVR microcontrollers, FPGAs, ASICs without side-channel protection, and ASICs with side-channel protection.

CHES'17 [BKL+17]

The Sponge Structure

r: rate ; c: capacity ; g: sponge permutation.



Sponge-based hash function (e.g. SHA-3).
There are many other sponge-based structures [BDPV12].

Structure of GIMLI (1/2)

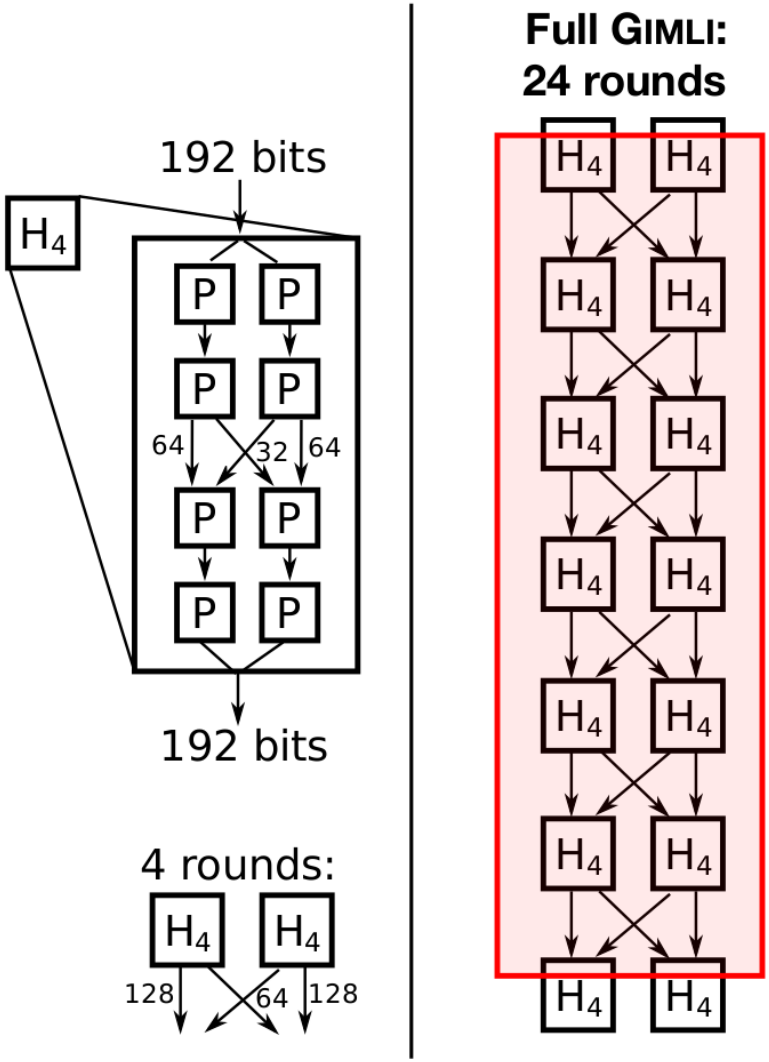
```
for (round = 24; round > 0; --round)
{
  for (column = 0; column < 4; ++column)
  {
    x = rotate(state[    column], 24);
    y = rotate(state[4 + column],  9);
    z =        state[8 + column];

    state[8 + column] = x ^ (z << 1) ^ ((y&z) << 2);
    state[4 + column] = y ^ x        ^ ((x|z) << 1);
    state[column]     = z ^ y        ^ ((x&y) << 3);
  }

  if ((round & 3) == 0) { // small swap: pattern s...s...s... etc.
    x = state[0];
    state[0] = state[1];
    state[1] = x;
    x = state[2];
    state[2] = state[3];
    state[3] = x;
  }
  if ((round & 3) == 2) { // big swap: pattern ..S...S...S. etc.
    x = state[0];
    state[0] = state[2];
    state[2] = x;
    x = state[1];
    state[1] = state[3];
    state[3] = x;
  }

  if ((round & 3) == 0) { // add constant: pattern c...c...c... etc.
    state[0] ^= (0x9e377900 | round);
  }
}
```

Structure of GIMLI (2/2)



Picture from rump session presentation corresponding to

http://ia.cr/2017/743

Distinguisher against GIMLI

Gimli has 24 rounds. If $\textsc{Gimli}_{22.5}$ is 22.5-round Gimli, then

$$x \mapsto \text{Truncate}_{192}\Big(\textsc{Gimli}_{22.5}(x \,||\, k)\Big)$$

is not a secure PRF (`http://ia.cr/2017/743`).
Unclear how it applies to sponge modes though.

Many academic designs are broken

Zorro Idea: AES with fewer S-Boxes to ease masking... Differential attacks become possible.

Many academic designs are broken

Zorro Idea: AES with fewer S-Boxes to ease masking... Differential attacks become possible.

KTANTAN Idea: build block cipher like stream cipher... Diffusion of key information can be too slow.

Many academic designs are broken

Zorro Idea: AES with fewer S-Boxes to ease masking... Differential attacks become possible.

KTANTAN Idea: build block cipher like stream cipher... Diffusion of key information can be too slow.

iScream Idea: Identical S-Boxes on columns of state, identical L-Boxes on rows... Highly structured round function + sparse round constants = invariant subspace attacks.

Lessons Learnt

- And/Rotate/XOR → way to go for versatility
- Sponge → way to go for versatility
- It is still cryptography → proper vetting by the community is needed. Practical attacks against full-round primitives do happen!

Outline

Plan of this Section

Conclusion

- Importance of publication process
- Performance vs. Security
- Versatility vs. Specialization

Conclusion

- Importance of publication process
- Performance vs. Security
- Versatility vs. Specialization

Thank you!

Ross Anderson.

A5 (Was: HACKING DIGITAL PHONES).
uk.telecom (Usenet),
https://groups.google.com/forum/?msg/uk.telecom/TkdCaytoeU4/Mroy719hdroJ#!msg/uk.telecom/TkdCaytoeU4/Mroy719hdroJ, June 1994.

Elad Barkan, Eli Biham, and Nathan Keller.

Instant ciphertext-only cryptanalysis of GSM encrypted communication.
Journal of Cryptology, 21(3):392–429, July 2008.

M. Becker and A. Desoky.

A study of the DVD content scrambling system (CSS) algorithm.
In *Proceedings of the Fourth IEEE International Symposium on Signal Processing and Information Technology, 2004.*, pages 353–356, Dec 2004.

Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche.

Duplexing the sponge: Single-pass authenticated encryption and other applications.
In Ali Miri and Serge Vaudenay, editors, *SAC 2011: 18th Annual International Workshop on Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, Heidelberg, August 2012.

Stephen C. Bono, Matthew Green, Adam Stubblefield, Ari Juels, Aviel D. Rubin, and Michael Szydlo.

Security analysis of a cryptographically-enabled RFID device.
In *Proceedings of the 14th Conference on USENIX Security Symposium - Volume 14*, SSYM'05, pages 1–1, Berkeley, CA, USA, 2005. USENIX Association.

Daniel J. Bernstein, Stefan Kölbl, Stefan Lucks, Pedro Maat Costa Massolino, Florian Mendel, Kashif Nawaz, Tobias Schneider, Peter Schwabe, François-Xavier Standaert, Yosuke Todo, and Benoît Viguier.

Gimli : A cross-platform permutation.
In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 299–320. Springer, Heidelberg, September 2017.

Julia Borghoff, Lars R. Knudsen, Gregor Leander, and Krystian Matusiewicz.

Cryptanalysis of C2.
In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 250–266. Springer, Heidelberg, August 2009.

Alex Biryukov, Joseph Lano, and Bart Preneel.

Cryptanalysis of the alleged SecurID hash function.
In Mitsuru Matsui and Robert J. Zuccherato, editors, *SAC 2003: 10th Annual International Workshop on Selected Areas in Cryptography*, volume 3006 of *Lecture Notes in Computer Science*, pages 130–144. Springer, Heidelberg, August 2004.

Alex Biryukov, Gaëtan Leurent, and Arnab Roy.

Cryptanalysis of the “kindle” cipher.
In Lars R. Knudsen and Huapeng Wu, editors, *SAC 2012: 19th Annual International Workshop on Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 86–103. Springer, Heidelberg, August 2013.

F.J. Bruwer, W. Smit, and G.J. Kuhn.

Microchips and remote control devices comprising same, May 1996.
US Patent 5,517,187.

B. Driessen, R. Hund, C. Willems, C. Paar, and T. Holz.

Don’t trust satellite phones: A security analysis of two satphone standards.
In *2012 IEEE Symposium on Security and Privacy*, pages 128–142, May 2012.

Scott R. Fluhrer and Stefan Lucks.

Analysis of the E0 encryption system.
In Serge Vaudenay and Amr M. Youssef, editors, *SAC 2001: 8th Annual International Workshop on Selected Areas in Cryptography*, volume 2259 of *Lecture Notes in Computer Science*, pages 38–48. Springer, Heidelberg, August 2001.

Flavio D. Garcia, Gerhard de Koning Gans, and Roel Verdult.

Wirelessly lockpicking a smart card reader.
International Journal of Information Security, 13(5):403–420, 2014.

Flavio D. Garcia, Peter van Rossum, Roel Verdult, and Ronny Wichers Schreur.

Dismantling SecureMemory, CryptoMemory and CryptoRF.
In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 250–259, New York, NY, USA, 2010. ACM.

Stefan Lucks, Andreas Schuler, Erik Tews, Ralf-Philipp Weinmann, and Matthias Wenzel.

Attacks on the DECT authentication mechanisms.
In Marc Fischlin, editor, *Topics in Cryptology – CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 48–65. Springer, Heidelberg, April 2009.

Karsten Nohl, David Evans, Starbug Starbug, and Henryk Plötz.

Reverse-engineering a cryptographic RFID tag.
In *USENIX security symposium*, volume 28, 2008.

Nobody.

Thank you Bob Anderson.
Mail to the cypherpunk mailing list from nobody@jpunix.com, available at https://web.archive.org/web/20010722163902/http://cypherpunks.venona.com/date/1994/09/msg00304.html, September 1994.

Roel Verdult, Flavio D. Garcia, and Josep Balasch.

Gone in 360 seconds: Hijacking with hitag2.
In *Proceedings of the 21st USENIX Conference on Security Symposium*, Security'12, pages 37–37, Berkeley, CA, USA, 2012. USENIX Association.

Roel Verdult, Flavio D Garcia, and Baris Ege.

Dismantling Megamos crypto: Wirelessly lockpicking a vehicle immobilizer.
In *Supplement to the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 703–718. USENIX Association, August 2013.

David Wagner, Leone Simpson, Ed Dawson, John Kelsey, William Millan, and Bruce Schneier.

Cryptanalysis of ORYX.
In Stafford E. Tavares and Henk Meijer, editors, *SAC 1998: 5th Annual International Workshop on Selected Areas in Cryptography*, volume 1556 of *Lecture Notes in Computer Science*, pages 296–305. Springer, Heidelberg, August 1999.

David Wagner, Bruce Schneier, and John Kelsey.

Cryptanalysis of the cellular encryption algorithm.
In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 526–537. Springer, Heidelberg, August 1997.

Ralf-Philipp Weinmann and Kai Wirt.

Analysis of the DVB Common Scrambling Algorithm.
In David Chadwick and Bart Preneel, editors, *Communications and Multimedia Security: 8th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security, Sept. 15–18, 2004, Windermere, The Lake District, United Kingdom*, volume 175 of *IFIP – The International Federation for Information Processing*, Boston, MA, 2005. Springer US.