Security for 1000 Gate Equivalents

Carsten Rolfes, Axel Poschmann, Gregor Leander, and Christof Paar

Horst-Görtz-Institute for IT-Security
Ruhr-University Bochum, Germany
{rolfes,poschmann,cpaar}@crypto.rub.de
leander@itsc.rub.de

Abstract. Product piracy and counterfeiting is a market with an annual turnover of hundreds of billions of US-Dollars. The application of RFID-tags is discussed widely to cope with this problem. A major obstacle for a mass deployment of RFID-tags, beside the harsh requirements on power consumption, are fabrication costs. In hardware fabrication costs are proportional to the required area. In this paper we present three different architectures of the PRESENT algorithm. Our implementation of the serialized architecture requires only 1000 GE. To the best of our knowledge this is the smallest hardware implementation of a cryptographic algorithm with a moderate security level.

1 Introduction

According to the U.S. Chamber of Commerce "counterfeiting and product piracy cost the U.S. economy between $200 billion and $250 billion per year and a total of 750.000 American Jobs" [10, p.26]. Combined with other sources, [2] estimates the global market size of counterfeited goods with US-$527 billion[1].

At the moment many products are identified with barcodes that virtually come free of cost. Even though the prices for RFID-tags continuously decreased in the past, the price for an RFID-tag is still in the range of a few cents. Furthermore, many goods, especially commodities, are very cost sensitive. In hardware the price of an ASIC is roughly equivalent to the area in silicon it requires. The area is usually measured in μm^2, but this value depends on the fabrication technology and the standard cell library. In order to compare the area requirements independently it is common to state the area as *gate equivalents* (GE). One GE is equivalent to the area which is required by the two-input NAND gate with the lowest driving strength of the appropriate technology. The area in GE is derived by dividing the area in μm^2 by the area of a two-input NAND gate.

The RFID technology is widely discussed as a promising solution for the counterfeiting issues in the literature [29, 31, 16, 20]. Many of the proposed authentication protocols use a Pseudo Random Number Generator (PRNG), a hash function, or symmetric key encryption [23, 8, 12, 11, 3, 9, 19, 26, 13]. Block

[1] Note that the value of global drug trade is estimated with US-$321.6 billion in 2005 [24, p.127].

ciphers can be used as basic building blocks for a secure identification system, for example in a challenge-response protocol. Unfortunately, a vast majority of block ciphers have been developed with good software properties in mind, which in turn means that the gate count for a hardware implementation is rather high.

In order to cope with this situation there have quite a few cryptographic algorithms been published that are especially optimized for ultra-constrained devices. Examples for lightweight stream ciphers are Grain [15] and Trivium [7] and examples for lightweight block ciphers are DESXL [18], HIGHT [6], mCrypton [21], PRESENT [5], and SEA [22]. Some designers kept the algorithm secret in order to gain also security by obscurity. However, the cryptanalysis of two widely used algorithms show that this violation of the *Kerckhoff principle* [17] is prohibitive: *Keeloq* [1] and *Mifare* both were broken shortly after their algorithm was reverse-engineered [4, 25].

PRESENT is an aggressively hardware optimized block cipher, first presented at CHES 2007 [5]. According to the authors PRESENT was developed with a minimal hardware footprint (1570 GE) in mind such that it is suitable for RFID-tags. However, in this work we show that a serialized implementation can be realized with as few as 1000 GE.

In the remainder of this work, we first recall the PRESENT algorithm in Section 2. We propose three different hardware architectures of PRESENT-80 in Section 3, and present our implementation results in Section 4. Finally, in Section 5 we conclude the paper.

2 The PRESENT Algorithm

PRESENT is a substitution-permutation network with 64-bits block size and 80 or 128-bits of key (from here on referred to as PRESENT-80 or PRESENT-128, respectively). In the remainder of this article we focus on PRESENT-80, because 80-bits provide a security level which is sufficient for many RFID-ish applications. PRESENT has 31 regular rounds and a final round that only consists of the key mixing step. One regular round consists of a key mixing step, a substitution layer, and a permutation layer.

The substitution layer consists of 16 S-Boxes that each have 4-bit input and 4-bit output (4x4): $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$. The S-Box is given in hexadecimal notation according to the following table.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

The bit permutation used in PRESENT is given by the following table. Bit i of STATE is moved to bit position $P(i)$.

```
generateRoundKeys()
for $i = 1$ to 31 do
    addRoundKey(STATE,$K_i$)
    sBoxLayer(STATE)
    pLayer(STATE)
end for
addRoundKey(STATE,$K_{32}$)
```

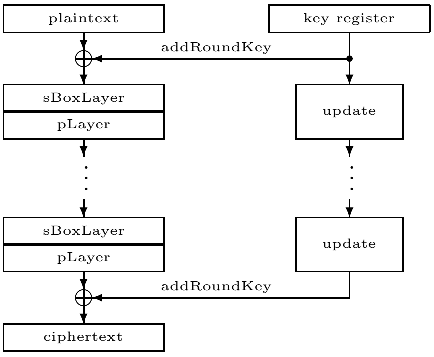

Fig. 1. A top-level algorithmic description of PRESENT.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

The key schedule of PRESENT-80 consists of a 61-bit left rotation, an S-Box, and an XOR with a round counter. Note that PRESENT uses the same S-Box for the datapath and the key schedule, which allows to share resources. The user-supplied key is stored in a key register and its 64 most significant (i.e. leftmost) bits serve as the round key. The key register is rotated by 61 bit positions to the left, the left-most four bits are passed through the PRESENT S-Box, and the round counter value i is exclusive-ored with bits $\mathtt{k}_{19}\mathtt{k}_{18}\mathtt{k}_{17}\mathtt{k}_{16}\mathtt{k}_{15}$ of K with the least significant bit of the round counter on the right. For further details, the interested reader is referred to [5].

3 Three Different Architectures of PRESENT Implementations

For different application scenarios there exists also different demands on the implementation and the optimization goals. For example, an implementation for an RFID-tag requires small area and power consumption, while the throughput is of secondary interest. On the other hand, an RFID-reader device that reads out many devices at the same time, requires a higher throughput, but area and power consumption are less important. Some key figures of the PRESENT block cipher are area, throughput, and power consumption. We characterized three architectures, so one can choose one that meets the given requirements most suitable. The first architecture is round based as described by [5]. It is optimized for area and speed. The second architecture uses pipeline technique and generates a high throughput. The third architecture is serialized and is minimized in terms of area and power consumption. All architectures can perform encryption only, which is sufficient for implementation in challenge-response protocols.

3.1 Round Based Architecture

This architecture represents the direct implementation of the PRESENT top-level algorithm description in Figure 1, i.e. one round of PRESENT is performed in one clock cycle. The focus lies on a compact solution but at the same time with an eye on the time-area product. To save power and area a loop based approach is chosen. The balance between the 64-bit datapath and the used operations per clock cycle leads to a good time-area product. Due to the reuse of several building blocks and the round structure, the design has a high energy efficiency. The architecture uses only one substitution and permutation layer. So the datapath consists of one 64-bit XOR, 16 S-Boxes in parallel, and one P-Box. To store the internal state and the key, a 64-bit state register and an 80-bit key register are introduced. Furthermore an 80-bit 2-to-1 multiplexer and a 64-bit 2-to-1 mux to switch between the load phase and the round computation phase are required. Key register, keymux, a 5-bit XOR, a single S-Box and a 61-bit shifter are merged into the component key scheduling. It computes the roundkey on the fly. Figure 2 presents the signal structure of the PRESENT round approach. At first the key and the plaintext are stored into the accordant register. After each round the internal state is stored into the state register. After 31 rounds the state is finally processed via XOR with the last roundkey. In addition to the already published PRESENT implementation [5] the control logic is implemented as a Finite State Machine (FSM). The FSM controls the multiplexers, to switch between load and encryption phase, and the round counter.

To reduce the used area we make use of clock gating. It can be applied to synchronous load enable registers, which are groups of flip-flops that are connected to the same clock and control signals. Normally a register is implemented by use of a flip-flop, a feedback loop, and a multiplexer. When this register bank maintains the same logic value through multiple clock cycles its clock network, the multiplexers, and the flip-flops unnecessarily consume power. Clock gating eliminates the feedback nets and multiplexers inserting a latch and an AND gate in the clock net of the registers. The latch prevents glitches on the enable signal. By controlling the clock signal for the register bank, the need for reloading the same value in the register through multiple clock cycles is eliminated. Clock gating reduces the clock network power dissipation, relaxes the data path timing, and reduces routing congestion by removing feedback multiplexer loops. For designs that have large multi-bit registers, clock gating can save power and further reduce the number of gates in the design. However, for smaller register banks, the overhead of adding logic to the clock tree might not compare favorably to the power saved by eliminating a few feedback nets and multiplexers.

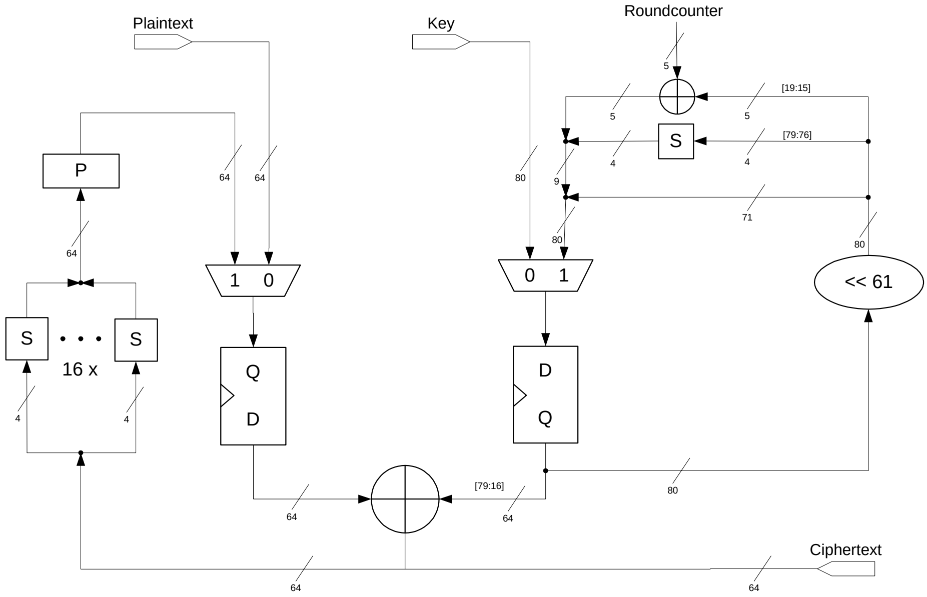

Fig. 2. Block diagram of the round based version

3.2 Parallel Architecture

The main goal of the design is to achieve a high throughput rate. Therefore the 32 time loop is unrolled, so all XORs, S-Boxes, and P-Boxes are cascaded. This will not only lead to high area effort and heavily increased power consumption, but also to high data throughput. The required roundkey is generated by taking the correct bits from the 80-bit key and if necessary pass them through a S-Box or add a round counter value. All subkeys are available in parallel and no register is needed to hold the key. Figure 3 shows the signal diagram of the pipelined architecture. It consists of 32 XORs, 496 S-Boxes, and 31 P-Boxes for the datapath. The key path consists of 31 S-Boxes and 31 XORs for key scheduling. The roundcounter input of the XOR is hard wired. First the given 64-bit plaintext and the first roundkey are xored. The result is split up into 16 4-bit blocks. Each block is processed by a 4-bit S-Box in parallel. The 64-bit P-Box transposes the bits at the end of each of the 31 rounds. However, the 32th round consists only of the XOR operation, because another S-Box and P-Box do not add security.

This straight forward approach does not achieve a high maximum operating frequency. This results from the long critical path. The input signal has to propagate through all XOR and S-Box gates. The more gates belong to the path the higher is the resulting capacitance to be switched. So the time period for a switching event is stretched. To shorten the critical path, flip-flops as pipeline stages were installed after each P-Box (see Figure 3). On the one hand

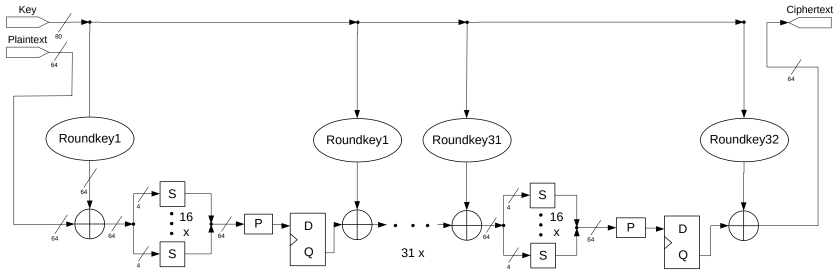

Fig. 3. Datapath of the pipelined parallel version

this increases the chip area and power consumption, but on the other hand the maximum operating frequency can be raised significantly.

3.3 Serialized Architecture

This architecture is a further modification on the round based architecture described in Section 3.1. To save more chip area, the data structure is reduced to 4-bit. One of the most area consuming parts of PRESENT are the 16 S-Boxes in parallel. So only one of them is used to represent the substitution layer, which is also shared between state scheduling and key scheduling. Another power and area consuming part are the large input multiplexers. We use a 4-bit interface to read in the key and the plaintext. This area savings come at the disadvantage of a longer computation time. Only 4-bit are processed during one clock cycle and we need 20 clock cycles for initialization. An additional 4-bit counter upgrades the FSM to control the processing of the internal state. Therefore it takes additional 15 cycles to compute the substitution layer of each round. As one can see in Figure 4 the signal diagram still shows a 64-bit wide and a 80-bit wide path. The main problem is to serialize the permutation layer. So we choose a memory structure with two different operation modes. In the first mode it behaves like a shift register. During load phase and S-Box computation phase the 4-bit input is shifted to the left. The 4-bit output is appended at the beginning. If the P-layer is computed all bits are read in parallel and the 64-bit wide or 80-bit wide input and output is used. Each memory element consists of scan flip-flops. It is a D-flip-flop with integrated multiplexer which saves area instead of one normal D-flip-flop and separated mux. One further advantage is the reduced computation time, so we need only one clock cycle for the whole P-layer. A 4-bit computation scheme would lead to much more multiplexers. All together we need 17 clock cycles per round to compute the new state.

4 Evaluation of Results

In this section we first describe the used design flow and the metrics. Subsequently we discuss our implementation results for two different frequencies,

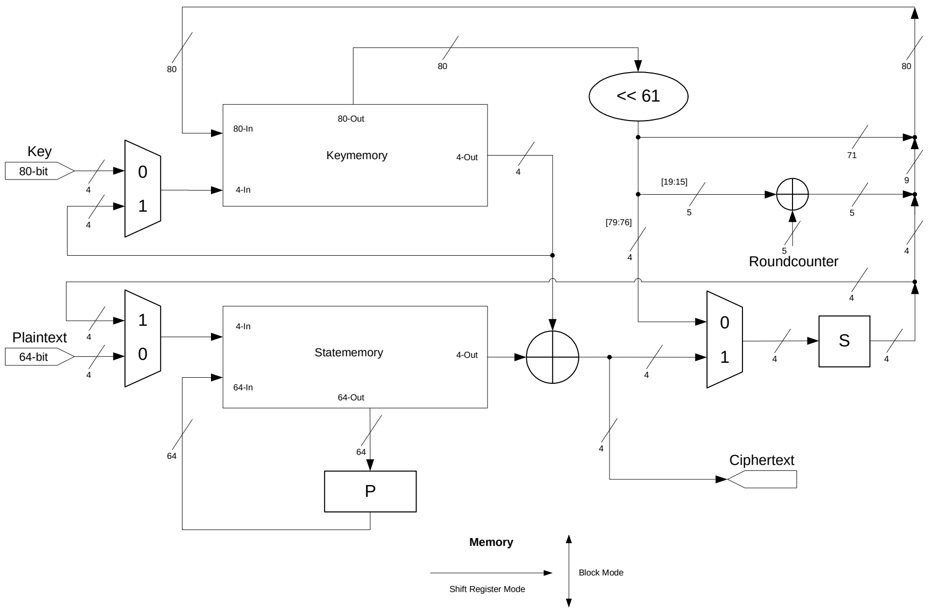

Fig. 4. Datapath of the serial version

100KHz and 10MHz, and compare them to other published hardware implementations.

4.1 Metrics and used Design Flow

All architectures were developed and synthesized by using a script based design flow. We used MentorGraphics FPGA Advantage 8.1 for HDL source code construction and functional verification. Then the RTL-level description was synthesized with Synopsys *Design Compiler* Z-2007.03-SP5, which was also used to generate the area, timing, and power estimation reports. The main effort of synthesis process was area optimization. The S-Box is described as boolean equation which leads to a combinatorial logic implementation. The P-Box is only simple wiring, which is not very costly in hardware. We used three different standard cell libraries with different technology parameters: a 350 *nm* technology MTC45000 from AMIS, a 250 *nm* technology SESAME-LP2 from IHP, and a 180 *nm* technology UMCL18G212D3 from UMC. Each library consists of a different amount of cells, because not all possible logical functions might be implemented as gates. This fact will lead to different area result expressed in GE. Following definitions of metrics were used:

Area: This metric represents the amount of area normalized to the area of one NAND gate. This ratio is expressed in GE.

Cycles: Number of clock cycles to compute and read out the ciphertext.

Throughput: The rate at which new output is produced with respect to time. The number of ciphertext bits is divided by the needed cycles and multiplied by the operating frequency. It is expressed in bits-per-second. With increasing frequency the throughput will increase, too.

Power: The power consumption is estimated by PowerCompiler. It consists of two major components: the static power which is proportional to the area and the fabrication process. The dynamic power is proportional to the switching activity (switching event probability and operating frequency). Both components also depend on the supply voltage, but the static power contributes only small percentage to the overall power dissipation.

Current: The power consumption divided by the typical core voltage of the process. These are for AMI 3.3V, for IHP 2.5V, and for UMC 1.8V.

Throughput to area ratio: This representation is used as a measure of design efficiency.

Maximum frequency: There are many connections between the input and output pins. The delay of each gate forms a timing path for the signals. The slowest path will set the upper bound of clock frequency. Note that it might be possible to increase the max. frequency, but this will also increase area and power.

4.2 Comparison of Results at 100 kHz

Table 1 shows the synthesis results at 100 kHz clock frequency, which is a typical operating frequency of RFID tags. All three architectures are synthesized with different standard cell libraries. The serial implementation consumes about 1000 GE of area. To the best of our knowledge this is the smallest implementation of a cryptographic algorithm with a moderate security level. Even implementations of the stream ciphers Grain80 and Trivium require more area (1294 GE and 1857 GE, respectively [14]). The power consumption shows a large variation depending on the core voltage of the library, but is still the lowest compared to the others architectures. The round based implementation shows a good trade-off between area, throughput, and current. It does not consume significant more area and current than the serial one, but needs much less clock cycles for computation. The pipelined implementation generates a very high throughput at the expense of area and power.

4.3 Comparison of Results at 10 MHz

Table 2 shows the synthesis results at 10 MHz clock frequency. The basic message is that scaling of operation frequency has a great impact on power consumption. The area is barely affected by this circumstance, because we chose an area optimize synthesis approach. If we get to higher frequencies the capacitances will become increasingly important. So cells with a higher driving strength must be used to drive the load and the area will increase conspicuously.

In Table 3 the results of the round based architecture, that means a new internal state is computed every clock cycle, are compared to other known round

Table 1. Implementation results of different architectures @ 100 kHz

Architecture	**Area**	**Cycles**	**Tput**	**Power**	**Current**	**Tput/Area**	**max. Freq.**
Library	**[GE]**		[$kbps$]	[μW]	[μA]	[$kbps/\mu m^2$]	[GHz]
Serial							
AMI 0.35 μm	999.52	563	11.4	11.20	3.39	0.0002	0.53
IHP 0.25 μm	1,168.75	563	11.4	4.24	1.70	0.0003	1.52
UMC 0.18 μm	1,074.98	563	11.4	2.52	1.40	0.0011	1.11
Round							
AMI 0.35 μm	1,524.77	32	200.0	33.40	10.12	0.0024	0.65
IHP 0.25 μm	1,594.25	32	200.0	4.84	1.94	0.0044	1.39
UMC 0.18 μm	1,650.30	32	200.0	3.86	2.14	0.0125	0.22
Pipeline							
AMI 0.35 μm	24,247.29	1	6,400.0	772.00	233.94	0.0049	0.07
IHP 0.25 μm	25,193.00	1	6,400.0	121.00	48.40	0.0090	0.20
UMC 0.18 μm	27,009.02	1	6,400.0	72.20	40.11	0.0245	0.15
better is	lower	lower	higher	lower	lower	higher	higher

Table 2. Implementation results of different architectures @ 10 MHz

Architecture	**Area**	**Cycles**	**Tput**	**Power**	**Current**	**Tput/Area**	**max. Freq**
Library	**[GE]**		[$kbps$]	[μW]	[μA]	[$kbps/\mu m^2$]	[GHz]
Serial							
AMI 0.35 μm	1,001.2	563	1,136.8	1123.0	340.3	0.0210	0.69
IHP 0.25 μm	1,168.8	563	1,136.8	421.0	168.4	0.0345	1.61
UMC 0.18 μm	1,075.0	563	1,136.8	247.0	137.2	0.1093	1.25
Round							
AMI 0.35 μm	1,560.5	31	20,645.2	3520.0	1066.7	0.2450	0.81
IHP 0.25 μm	1,594.2	31	20,645.2	436.0	174.4	0.4588	1.64
UMC 0.18 μm	1,706.0	31	20,645.2	77.1	42.8	1.2506	1.96
Pipeline							
AMI 0.35 μm	24,346.0	1	640,000.0	81295.0	24634.8	0.4868	0.08
IHP 0.25 μm	25,193.0	1	640,000.0	11659.0	4663.6	0.9001	0.21
UMC 0.18 μm	27,027.7	1	640,000.0	6888.0	3826.7	2.4470	0.16
better is	lower	lower	higher	lower	lower	higher	higher

based implementations of block ciphers, namely,ICEBERG [30] and HIGHT [6]. Both of them use a 64-bit datapath architecture. In Mace et. al [22] different ASIC implementations of SEA had been characterized. We choose the 96-bit architecture for better comparison to the other datapaths. Furthermore CLEFIA [28] and an compact AES implementation [27] are listed. The results illustrate the very compact design of the PRESENT block cipher. Even the throughput is only outperformed by the ICEBERG implementation, but at the expense of a significant higher energy consumption.

Table 3. Implementation results of round based datapath architectures

Cipher	**Tech.**	**Datapath**	**Freq.**	**Area**	**Tput**	**Energy/Bit**	**Power**
	$[\mu m]$	$[Bit]$	$[MHz]$	$[GE]$	$[Mbps]$	$[pJ/bit]$	$[\mu W]$
PRESENT	0.35	64	10	1561	20.6	170.5	3520.0
PRESENT	0.25	64	10	1594	20.6	21.1	436.0
PRESENT	0.18	64	10	1705	20.6	3.7	77.1
SEA [22]	0.13	96	250	3758	258.0	19.8	5102.0
ICEBERG [22]	0.13	64	250	7732	1000.0	9.6	9577.0
CLEFIA [28]	0.09	128	201	4950	715.7	-	-
AES [27]	0.13	128	131.24	5398	311.1	-	-
HIGHT [6]	0.25	64	80	3048	150.6	-	-
better is				lower	higher	lower	lower

5 Conclusions

In this paper we have pointed out that there is, due to harsh cost constraints inherent of mass deployment, a strong need for area optimized implementation of cryptographic algorithms. Furthermore, we presented the implementation results of three different architectures of the block cipher PRESENT. The pipelined version achieves a high throughput to area ratio but also consumes the most area and current compared to the other architectures. Therefore this architecture may be used in static devices in backend systems. The serial version can be implemented with as few as 1000 GE, which is to the best of our knowledge the smallest implementation of a cryptographic algorithm with a moderate security level. However, this significant area savings come at the disadvantage of a long processing time of 563 cycles. This architecture is best suited for power and area constrained devices such as passive RFID-tags. Interestingly, the round version draws for two of the three different libraries nearly the same current consumption. It requires about 50% more area but also achieves a relatively high throughput rate. This in turns yields a good energy consumption per encryption, hence this architecture is well suited for active constrained devices such as wireless sensor nodes.

References

1. Keeloq algorithm. Available online via `http://en.wikipedia.org/wiki/KeeLoq`, November 2006.
2. Havocscope illicit market news. Available online via `http://www.havocscope.com`, January 2008.
3. Daniel Bailey and Ari Juels. Shoehorning security into the EPC standard. In Roberto De Prisco and Moti Yung, editors, *International Conference on Security in Communication Networks – SCN 2006*, volume 4116 of *Lecture Notes in Computer Science*, pages 303–320, Maiori, Italy, September 2006. Springer-Verlag.

4. Andrey Bogdanov. Attacks on the KeeLoq Block Cipher and Authentication Systems. In *3rd Conference on RFID Security 2007 (RFIDSec 2007)*, 2007. Workshop Record.
5. Andrey Bogdanov, Gregor Leander, Lars R. Knudsen, Christof Paar, Axel Poschmann, Matthew J.B. Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. PRESENT - An Ultra-Lightweight Block Cipher. In *Proceedings of CHES 2007*, number 4727 in LNCS, pages 450 – 466. Springer-Verlag, 2007.
6. S. Hong J. Lim S. Lee B.-S Koo C. Lee D. Chang J. Lee K. Jeong H. Kim J. Kim D. Hong, J. Sung and S. Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device . In L. Goubin and M. Matsui, editors, *Proceedings of CHES 2006*, number 4249 in LNCS, pages 46–59. Springer-Verlag, 2006.
7. C. de Cannière and B. Preneel. Trivium. Available via `www.ecrypt.eu.org/stream`.
8. Tassos Dimitriou. A lightweight RFID protocol to protect against traceability and cloning attacks. In *International Conference on Pervasive Computing and Communications – PerCom 2006*, Pisa, Italy, March 2006. IEEE, IEEE Computer Society Press.
9. Sandra Dominikus, Elisabeth Oswald, and Martin Feldhofer. Symmetric authentication for RFID systems in practice. Handout of the Ecrypt Workshop on RFID and Lightweight Crypto, July 2005.
10. Thomas J. Donohue. The state of american business 2007. Technical report, United States Chamber of Commerce, 2007.
11. Martin Feldhofer. An authentication protocol in a security layer for RFID smart tags. In *The 12th IEEE Mediterranean Electrotechnical Conference – MELECON 2004*, volume 2, pages 759–762, Dubrovnik, Croatia, May 2004. IEEE.
12. Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In Marc Joye and Jean-Jacques Quisquater, editors, *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370, Boston, Massachusetts, USA, August 2004. IACR, Springer-Verlag.
13. Xingxin (Grace) Gao, Zhe (Alex) Xiang, Hao Wang, Jun Shen, Jian Huang, and Song Song. An approach to security and privacy of RFID system for supply chain. In *Conference on E-Commerce Technology for Dynamic E-Business – CEC-East'04*, pages 164–168, Beijing, China, September 2005. IEEE, IEEE Computer Society.
14. T. Good and M. Benaissa. Hardware Results for selected Stream Cipher Candidates. State of the Art of Stream Ciphers 2007 (SASC 2007), Workshop Record, February 2007.
15. M. Hell, T. Johansson, A. Maximov, and W. Meier. A Stream Cipher Proposal: Grain-128. In *IEEE International Symposium on Information Theory—ISIT 2006*, 2006. Also available via `www.ecrypt.eu.org/stream`.
16. A. Juels. Rfid security and privacy: a research survey. *Selected Areas in Communications, IEEE Journal on*, 24(2):381–394, Feb. 2006.
17. Auguste Kerckhoff. La cryptographie militaire. *Journal des sciences militaires*, IX:5–38, Feb. 1883. available online via `http://www.petitcolas.net/fabien/kerckhoffs/crypto_militaire_1.pdf`.
18. Gregor Leander, Christof Paar, Axel Poschmann, and Kai Schramm. New Lighweight DES Variants. In *Proceedings of Fast Software Encryption 2007 – FSE 2007*, volume 4593 of *LNCS*, pages 196–210. Springer-Verlag, 2007.

19. Sangshin Lee, Tomoyuki Asano, and Kwangjo Kim. RFID mutual authentication scheme based on synchronized secret information. In *Symposium on Cryptography and Information Security*, Hiroshima, Japan, January 2006.
20. Mikko Lehtonen, Thorsten Staake, Florian Michahelles, and Elgar Fleisch. From identification to authentication - a review of RFID product authentication techniques. Printed handout of Workshop on RFID Security – RFIDSec 06, July 2006.
21. C. Lim and T. Korkishko. mcrypton - a lightweight block cipher for security of low-cost rfid tags and sensors. In T. Kwon J. Song and M. Yung, editors, *Proceedings of the Workshop on Information Security Applications - WISA'05*, number 3786 in LNCS, pages 243–258. Springer-Verlag, 2005.
22. Franois Mace, Franois-Xavier Standaert, and Jean-Jacques Quisquater. ASIC Implementations of the Block Cipher SEA for Constrained Applications. In *Proceedings of the Third International Conference on RFID Security - RFIDSec 2007*, pages 103 – 114, Malaga, Spain, 2007.
23. David Molnar, Andrea Soppera, and David Wagner. A scalable, delegatable, pseudonym protocol enabling ownership transfer of RFID tags. Handout of the Ecrypt Workshop on RFID and Lightweight Crypto, July 2005.
24. N.A. World drug report 2005. Technical report, United Nations Office on Drugs and Crime, June 2005. available online via `http://www.unodc.org/pdf/WDR_2005/volume_1_web.pdf`.
25. Karsten Nohl and Henryk Ploetz. Mifare - little security, despite obscurity. Talk at the 24th Chaos Communication Congress, December 2007.
26. Keunwoo Rhee, Jin Kwak, Seungjoo Kim, and Dongho Won. Challenge-response based RFID authentication protocol for distributed database environment. In Dieter Hutter and Markus Ullmann, editors, *International Conference on Security in Pervasive Computing – SPC 2005*, volume 3450 of *Lecture Notes in Computer Science*, pages 70–84, Boppard, Germany, April 2005. Springer-Verlag.
27. Akashi Satoh and Sumio Morioka. Hardware-focused performance comparison for the standard block ciphers aes, camellia, and triple-des. In Colin Boyd and Wenbo Mao, editors, *ISC*, volume 2851 of *Lecture Notes in Computer Science*, pages 252–266. Springer, 2003.
28. Sony Corporation. The 128-bit Blockcipher CLEFIA - Security and Performance Evaluations. Available online via `http://www.sonet.co.uk/Products/clefia/technical/data/clefia-eval-1.0.pdf`, June 2007.
29. T. Staake, F. Thiesse, and E. Fleisch. Extending the EPC network: the potential of RFID in anti-counterfeiting. *Proceedings of the 2005 ACM symposium on Applied computing*, pages 1607–1612, 2005.
30. F.-X. Standaert, G. Piret, N. Gershenfeld, and J.-J. Quisquater. Sea: A scalable encryption algorithm for small embedded applications. In J. Domingo-Ferrer, J. Posegga, and D. Schreckling, editors, *Smart Card Research and Applications, Proceedings of CARDIS 2006*, volume 3928 of *LNCS*, pages 222–236. Springer-Verlag.
31. P. Tuyls and L. Batina. RFID-tags for Anti-Counterfeiting. *Topics in Cryptology-CT-RSA*, 3860:115–131, 2006.

6 appendix

Following abbreviations are used in the subsequent tables
Cur - Current

Tput/Area - Throughput/Area
mFreq - maximum Frequency
mTput - maximum Throughput

Table 4. Implementation results of round @ 100 kHz

Library	**Area**	**Area**	**Power**	**Cur**	**Tput/Area**	**Path**	**mFreq**	**mTput**
	$[GE]$	$[\mu m^2]$	$[\mu W]$	$[\mu A]$	$[kbps/\mu m^2]$	$[ns]$	$[GHz]$	$[Mbps]$
AMI 0.35 μm	1,524.77	82,338	33.40	10.12	0.0024	1.53	0.65	1,307.2
IHP 0.25 μm	1,594.25	44,996	4.84	1.94	0.0044	0.72	1.39	2,777.8
UMC 0.18 μm	1,650.30	15,970	3.86	2.14	0.0125	4.57	0.22	437.6
better is	lower	lower	lower	lower	higher	lower	higher	higher

Table 5. Implementation results of round @ 10 MHz

Library	**Area**	**Area**	**Power**	**Cur**	**Tput/Area**	**Path**	**mFreq**	**mTput**
	$[GE]$	$[\mu m^2]$	$[\mu W]$	$[\mu A]$	$[kbps/\mu m^2]$	$[ns]$	$[GHz]$	$[Mbps]$
AMI 0.35 μm	1,560.5	84,268	3520.0	1066.7	0.2450	1.23	0.81	1,678.5
IHP 0.25 μm	1,594.2	44,996	436.0	174.4	0.4588	0.61	1.64	3,384.5
UMC 0.18 μm	1,706.0	16,509	77.1	42.8	1.2506	0.51	1.96	4,048.1
better is	lower	lower	lower	lower	higher	lower	higher	higher

Table 6. Implementation results of pipeline @ 100 kHz

Library	**Area**	**Area**	**Power**	**Cur**	**Tput/Area**	**Path**	**mFreq**	**mTput**
	$[GE]$	$[\mu m^2]$	$[\mu W]$	$[\mu A]$	$[kbps/\mu m^2]$	$[ns]$	$[GHz]$	$[Mbps]$
AMI 0.35 μm	24,247	1,309,354	772.0	233.9	0.0049	13.84	0.07	4,624.3
IHP 0.25 μm	25,193	711,047	121.0	48.4	0.0090	4.98	0.20	12,851.4
UMC 0.18 μm	27,009	261,366	72.2	40.1	0.0245	6.78	0.15	9,439.5
better is	lower	lower	lower	lower	higher	lower	higher	higher

Table 7. Implementation results of pipeline @ 10 MHz

Library	**Area**	**Area**	**Power**	**Cur**	**Tput/Area**	**Path**	**mFreq**	**mTput**
	$[GE]$	$[\mu m^2]$	$[\mu W]$	$[\mu A]$	$[kbps/\mu m^2]$	$[ns]$	$[GHz]$	$[Mbps]$
AMI 0.35 μm	24,346	1,314,677	81295.0	24634.8	0.4868	12.8	0.08	5,000
IHP 0.25 μm	25,193	711,047	11659.0	4663.6	0.9001	4.78	0.21	13,389
UMC 0.18 μm	27,028	261,547	6888.0	3826.7	2.4470	6.26	0.16	10,224
better is	lower	lower	lower	lower	higher	lower	higher	higher

Table 8. Implementation results of serial @ 100 kHz

Library	**Area**	**Area**	**Power**	**Cur**	**Tput/Area**	**Path**	**mFreq**	**mTput**
	$[GE]$	$[\mu m^2]$	$[\mu W]$	$[\mu A]$	$[kbps/\mu m^2]$	$[ns]$	$[GHz]$	$[Mbps]$
AMI 0.35 μm	999.5	53,974	11.20	3.39	0.0002	1.89	0.5	60.1
IHP 0.25 μm	1,168.8	32,987	4.24	1.70	0.0003	0.66	1.5	172.2
UMC 0.18 μm	1,075.0	10,403	2.52	1.40	0.0011	0.9	1.1	126.3
better is	lower	lower	lower	lower	higher	lower	higher	higher

Table 9. Implementation results of serial @ 10 MHz

Library	**Area**	**Area**	**Power**	**Cur.**	**Tput/Area**	**cPath**	**mFreq.**	**mTput**
	$[GE]$	$[\mu m^2]$	$[\mu W]$	$[\mu A]$	$[kbps/\mu m^2]$	$[ns]$	$[GHz]$	$[Mbps]$
AMI 0.35 μm	1,001.19	54,064	1123.00	340.30	0.0210	1.44	0.69	78.9
IHP 0.25 μm	1,168.75	32,987	421.00	168.40	0.0345	0.62	1.61	183.3
UMC 0.18 μm	1,074.98	10,403	247.00	137.22	0.1093	0.8	1.25	142.1
better is	lower	lower	lower	lower	higher	lower	higher	higher