Ultra-Lightweight Implementations for Smart Devices – 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. In recent years more and more security sensitive applications use passive smart devices such as contactless smart cards and RFID tags. Cost constraints imply a small hardware footprint of all components of a smart device. One particular problem of all *passive* smart devices such as RFID tags and contactless smart cards are the harsh power constraints. On the other hand, *active* smart devices have to minimize *energy* consumption. Recently, many lightweight block ciphers have been published. In this paper we present three different architecture of the ultra-lightweight algorithm PRESENT and highlight their suitability for both active and passive smart devices. Our implementation results of the serialized architecture require 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 Background

Smart cards are widely in use for authentication, access control, and payment purposes. Their applications range from access control of ski ressorts and soccer stadiums over parking lots to highly secured areas of both company and goverment buildings. MasterCard, Visa, and JCB are currently defining specifications for contact and contactless payments using smart cards in their EMV standards [17,16]. In recent years there has been an increasing trend towards contactless smart cards. In fact, contactless smart cards are a special subset of passive RFID tags [8]. With regards to the terminology of pervasive computing both can be summarised by the term *passive smart devices*. Even for very sensitive data passive smart devices are used, e.g. Visa payWave card [22], hence, security mechanism play a key role for these applications.

Many smart devices, especially commodities, are very cost sensitive. If the volumes are large enough –and this is indicated by the term *pervasive*– an application specific integrated circuit (ASIC) will nearly always be cheaper than a programmable micro computer. 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 corresponding technology. The area in GE is derived by dividing the area in μm^2 by the area of a two-input NAND gate.

Moreover, one particular problem of all *passive* smart devices such as RFID tags and contactless smart cards are the harsh *power* constraints. Depending on the transmission range of the application (close, proximity, or vincinity coupling), the power constraints can be as strict as a few μW. Therefore, the ASIC and all of its components have to be designed with special care for the total *power* consumption. *Active* smart devices, such as wireless sensor nodes, RFID reader handhelds, or contact smart cards have their own power supply, i.e. a battery, or are powered by the reading device via a physical contact. Therefore, the power constraints are more relaxed for this device class. The main design goal here is to minimize the total *energy* consumption and the overall execution time.

Block ciphers are the working horses of the cryptographic primitives. 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 quite a few cryptographic algorithms have been published that are especially optimized for ultra-constrained devices. Examples for lightweight stream ciphers are Grain and Trivium and examples for lightweight block ciphers are DESXL [13], HIGHT [5], mCrypton [14], PRESENT [3], and SEA [15]. This research area is also referred to as *low cost* or *lightweight cryptography*. Some designers kept the algorithm secret in order to gain additional security by obscurity. However, the cryptanalyses of two widely used lightweight algorithms show that this violation of the *Kerckhoff principle* [12] is prohibitive: *Keeloq* [1] and *Mifare* both were broken shortly after their algorithm was reverse-engineered [2,18].

PRESENT is an aggressively hardware optimized ultra-lightweight block cipher, first presented at CHES 2007 [3]. According to the authors PRESENT was developed with a minimal hardware footprint (1570 GE) in mind such that it is suitable for passive RFID tags. However, in this work we show that a serialized implementation can be realized with as few as 1000 GE, which make it especially interesting for all kind of low cost passive smart devices. Moreover we propose two additional architectures which are suitable for low cost and high end active smart devices.

In the remainder of this work, we first recall the specification of PRESENT in Section 2. We propose three different hardware architectures of PRESENT in Section 3 and the implementation results are evaluated 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 for the 80 bit version

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

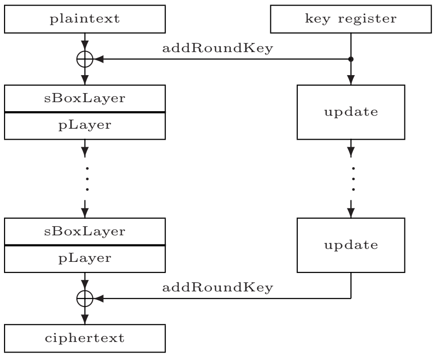

Fig. 1. A top-level algorithmic description of PRESENT

and PRESENT-128 for the 128 bit version). In the remainder of this article we focus on PRESENT, because 80-bits provide a security level which is sufficient for many RFID driven 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 in parallel 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 $P(i)$.

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 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 `round_counter` on the right. Figure 1 provides a top-level description of the PRESENT algorithm. For further details, the interested reader is referred to [3].

3 Three Different Architectures of PRESENT Implementations

For different application scenarios there exists also different demands on the implementation and the optimization goals. An implementation for a low cost passive smart device, such as RFID tags or contactless smart cards 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. Active smart devices, such as contact smart cards do not face strict power constraints but timing and sometimes energy constraints. Main key figures of the PRESENT block cipher are area, throughput, and power consumption. We propose three implementations of PRESENT, so one can choose the architecture that meets the given requirements most suitable. The first architecture is round based as described in [3]. It is optimized in terms of area, speed, and energy. The second architecture uses pipelining technique and generates a high throughput. The third architecture is serialized and is minimized in terms of area and power consumption. In order to decrease the area requirements even further, all architectures can perform encryption only. This is sufficient for encryption and decryption of data when the block cipher is operated for example in counter mode. Besides this it allows a fairer comparison with other lightweight implementations. For example the landmark implementation of Feldhofer et al. [7]. Finally, using the round based architecture of PRESENT128, we present a cryptographic co-processor with encryption and decryption capabilities. Note that the choice of an appropriate I/O interface is highly application specific, while at the same time can have a significant influence on the area, power, and timing figures. In order to have a clearer estimation of the cryptographic core's efficiency we did therefore not implement any special input or output interfaces, but rather chose a natural width of 64-bit input, 64-bit output and 80 or 128- bit key input, respectively.

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 as well. 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-Layer.

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 multiplexer to switch between the load phase and the round computation phase are required. Key register, key input multiplexer, a 5-bit XOR, one S-Box and a 61-bit shifter are merged into the component key scheduling. It computes the round key on the fly. Figure 2 presents the signal structure of the round based approach for PRESENT. 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 round key. The control logic is implemented as a Finite State Machine (FSM) and a 5-bit counter to count the rounds. The FSM also controls the multiplexers to switch between load and encryption phase.

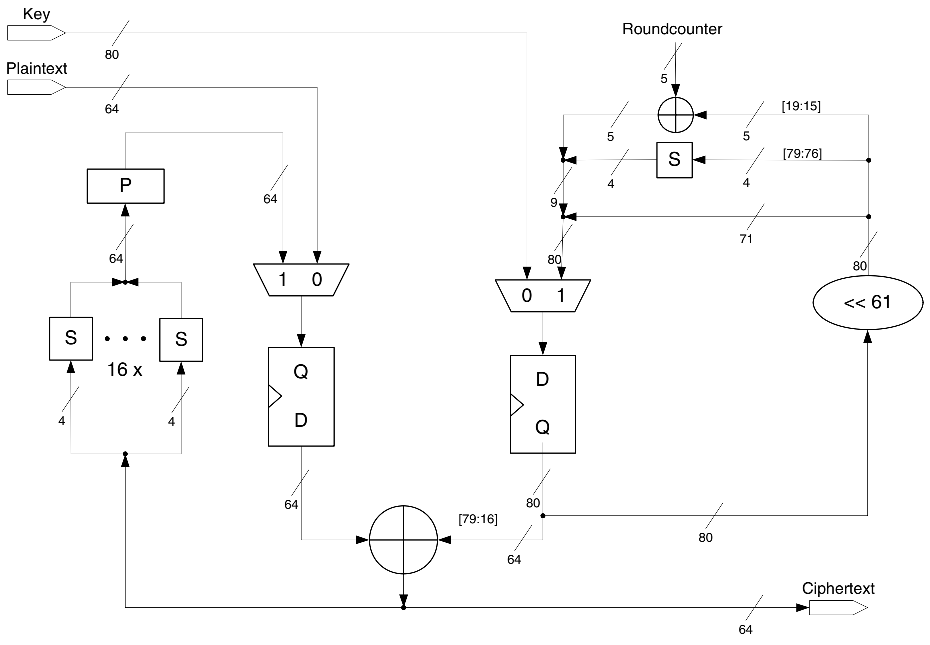

Fig. 2. Block diagram of the round-based PRESENT architecture

To reduce the used area and power 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 a 2-input 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.

3.2 Parallel Architecture

The main goal of the parallel design is to achieve a high throughput rate. Therefore the 31 time loop is unrolled, so all XORs, S-Boxes, and P-Layers are cascaded. This will lead to high area effort and power consumption, but also to high data throughput. The required round key is generated by taking the right bits from the 80-bit key and if necessary pass them through a S-Box or add a roundcounter 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-Layeres for the datapath. The keypath 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 round key 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-Layer transposes the bits at the end of each the 31 rounds. Note that, the 32th round consists only of the XOR operation.

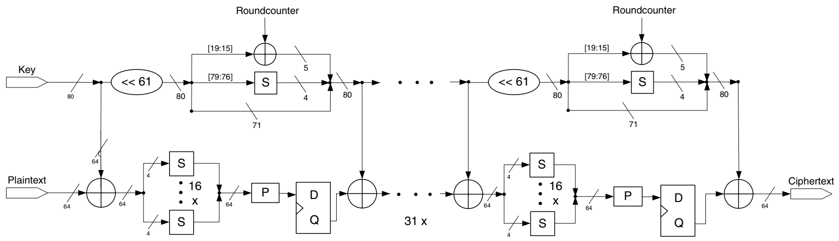

Fig. 3. Datapath of the pipelined parallel PRESENT architecture

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-Layer (see Figure 3). On the one hand this increases the chip area and power consumption, but on the other hand the maximum frequency can be raised significantly. We assume the key to be stable for many encryption operations. Thus roundkeys do not propagate through the pipeline and need not to be stored in additional FFs.

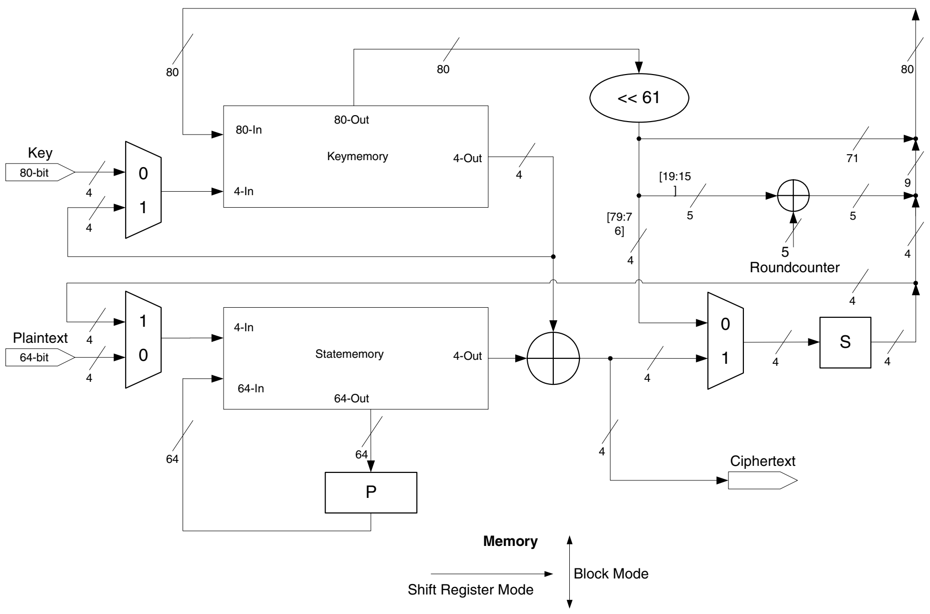

Fig. 4. Datapath of the serial PRESENT architecture

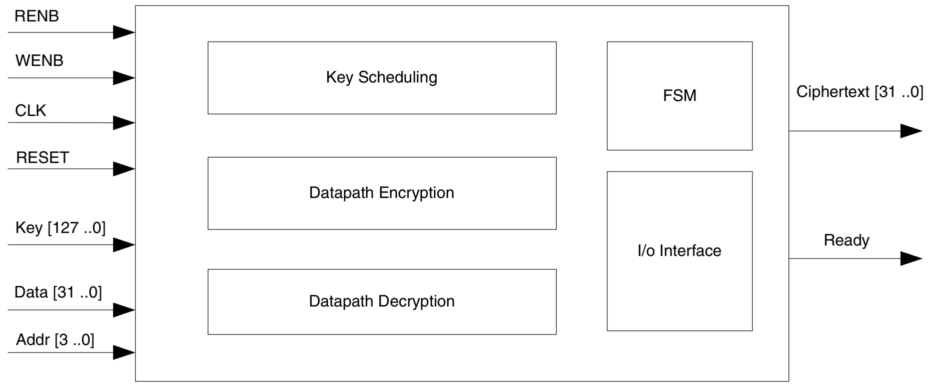

Fig. 5. Block diagramm of PRESENT-128 coprocessor with 32-bit interface

3.3 Serialized Approach

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 the data path and the key scheduling. Another power and area consuming part are the large input multiplexers. We use a 4-bit interface to read in key and 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, i.e. a D-flip-flop with integrated multiplexer, which saves area compared to one normal D-flip-flop and a seperated multiplexer. 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.

3.4 Crypto Coprocessor

To equip a smart device with cryptographic functions there are different ways to implement them. The first is to write software code. This solution requires RAM to store the program and inhibits the microcontroller while performing cryptographic algorithms. Another possibility is to implement the crypto part straight into the the microcontroller core. A more flexible way is to construct a cryptographic co-processor that is controlled by the main core. It uses a memory-like interface for communication. To get a compact and also fast solution we use the round based architecture with a modified finite state machine and added further multiplexers. Now the plaintext is loaded in 32-bit blocks. As far as we know this is the maximum bit width of microcontrollers for smart devices. The co-processor is controlled by write and read enable signals. The address signal selects the different bit blocks and encryption or decryption mode. Figure 5 illustrates the interfaces and the units.

4 Evaluation of the Results

In this section we first describe the used design flow and the metrics. Subsequently we compare our implementation results for the three scenarios low cost passive smart devices, low cost active smart devices, and high end smart devices. We considered the following optimization goals for the three scenarios: low cost and passive smart devices should be optimized for area and power constraints and low cost and active smart devices for area, energy, and time constraints. Note that in our methodology high end devices are always contact smart cards and hence should be optimized for time and energy constraints. Therefore we do not distinguish between passive and active high end smart devices.

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 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-Layer 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 of them consists of a different amount of cells and not all logical functions are implemented. 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 on the gate level by PowerCompiler[1]. 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.

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.

The interested reader can find more detailed tables with syntheses results in the appendix.

4.2 Low Cost Passive Smart Devices

Table 1 shows the synthesis results for 100 kHz clock frequency, which is a typical operating frequency of RFID tags. Smart devices with integrated

[1] Note that power estimations on the transistor level are more accurate. However, this also requires further design steps in the design flow, e.g. place&route.

contactless functionality have strict area and power constraints. For this purpose we propose a serialized implementation which will consume low area and power resources. Our serial implementation uses 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 [9]). For comparison with block ciphers we choose two AES implementations with a reduced datapath from Feldhofer et al. [6] and Hämäläinen et al. [10]. Furthermore there exists only a reduced datapath implementation of the lightweight block cipher SEA without key scheduling component and control logic. Note that a similar implementation with PRESENT would only require around 40 GE in 0.18μm UMC technology. The power consumption of our implementations show a large variation depending on the core voltage of the library, but the 0.18μm technology consumption is still the lowest compared to the other architectures. Note that power figures are highly technology dependent, therefore a fair comparison is only possible if the same technology was used.

Table 1. Implementation results of minimal datapath architectures

Cipher	**Tech.**	**Datapath**	**Freq.**	**Area**	**Throughp.**	**Cycles**	**Power**
	$[\mu m]$	$[Bit]$	$[MHz]$	$[GE]$	$[Kbps]$		$[\mu W]$
PRESENT-80	0.35	4	0.1	1,000	11.4	563	11.20
PRESENT-80	0.25	4	0.1	1,169	11.4	563	4.24
PRESENT-80	0.18	4	0.1	1,075	11.4	563	2.52
Feldhofer AES [6]	0.35	8	0.1	3400	12.4	1032	4.50
Hämäläinen AES [10]	0.13	8	80	3100	121	160	-
SEA [15]	0.13	8	0.1	449		50	3.22
better is				lower	higher	lower	lower

4.3 Low Cost Active Smart Devices

The second scenario targets standard smart cards. To reduce fabrication costs these cards are also area constraint. But in comparison to the prior scenario the crypto core draws his energy from a battery of a pervasive device or via the physical contact of the reading device. So the execution time is of major interest. The round based implementation shows a good trade off between area, time, throughput, and energy consumption. It does not consume significant more area and energy than the serial one, but needs much less clock cycles for computation. The results are compared to other known round based implementations that means a new internal state is computed every clock cycle. There are results for the ICEBERG [21] and the HIGHT [11] block cipher. Both of them use a 64-bit datapath architecture. In Mace et al. [15] different ASIC implementations of SEA had been characterized. We choose the 96-bit architecture for better comparison to the other datapaths. The results in Table 2 illustrate the very compact design of the PRESENT block cipher. Even the -normalized to 10 MHz.- throughput is only

Table 2. Implementation results of the round based datapath architectures

Cipher	**Tech.**	**Datapath**	**Freq.**	**Area**	**Tput**	**Energy/Bit**	**Power**
	$[\mu m]$	$[Bit]$	$[MHz]$	$[GE]$	$[Mbps]$	$[pJ/bit]$	$[\mu W]$
PRESENT-80	0.35	64	10	1561	20.6	170.5	3520.0
PRESENT-80	0.25	64	10	1594	20.6	21.1	436.0
PRESENT-80	0.18	64	10	1705	20.6	3.7	77.1
SEA [15]	0.13	96	250	3758	258.0	19.8	5102.0
ICEBERG [15]	0.13	64	250	7732	1000.0	9.6	9577.0
HIGHT [11]	0.25	64	80	3048	150.6	-	-
better is				lower	higher	lower	lower

Table 3. Implementation results of pipelined architecture @ 10 MHz

Library	**Area**	**Power**	**Tput/Area**	**crit. Path**	**max Freq.**	**max .Tput**
	$[GE]$	$[\mu W]$	$[kbps/\mu m^2]$	$[ns]$	$[GHz]$	$[Mbps]$
AMI 0.35 μm	24,345.87	81295.00	0.486811614	12.80	0.1	5,000.0
IHP 0.25 μm	25,193.00	11659.00	0.900080911	4.78	0.2	13,389.1
UMC 0.18 μm	27,027.69	6888.00	2.446979668	6.26	0.2	10,223.6
better is	lower	lower	higher	lower	higher	higher

Table 4. Implementation results of co-processor architectures

Cipher	**Tech.**	**Datapath**	**max Freq.**	**Area**	**Throughp.**	**Cycles**
	$[\mu m]$	$[Bit]$	$[MHz]$	$[GE]$	$[Mbps]$	
PRESENT-128	0.35	32	143	2,681	234	39
PRESENT-128	0.25	32	141	2,917	231	39
PRESENT-128	0.18	32	323	2,989	529	39
PRESENT-128	0.35	8	131	2,587	133	63
PRESENT-128	0.25	8	121	2,851	123	63
PRESENT-128	0.18	8	353	2,900	359	63
CAST AES [4]	0.18	32	300	124,000	872	44
Satoh AES [20]	0.11	32	131	54,000	311	54
Pramstaller AES [19]	0.6	32	50	85,000	70	92
better is			higher	lower	higher	lower

outperformed by the ICEBERG implementation., but again, we do not consider high throughput as highly relevant for this device class.

4.4 High End Active Smart Devices

In the third scenario there are no limitations for energy consumption. The task of the co-processor is to relieve the micro controller of the cryptographic computations. The design of this assistant should deliver results fast and consume as less area as possible to be cost-effective. One approach is to use a pipelined architecture. But Table 3 discloses that the pipelined implementation generates a very high throughput at the expense of area and power. 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 addition one has to be aware of the input/output interface. Up to now there exist only smart cards with 32-bit micro controllers. The best choice is to implement a round based architecture with an 32-bit I/O interface. In literature can be found several AES implementations that are up to the mark. We compare the PRESENT implementations to Pramstaller et al. [19] and Satoh et al. [20]. Also a commercial solution by Cast Inc. [4] is listed. Table 4 shows the results for the different implementations. As there are many smart cards equipped with 8-bit microcontrollers we list the results for an 8-bit interface, too. The PRESENT co-processor is much more compact than the other implementations and also needs less clock cycles to compute the ciphertext.

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 high end smart devices and the back end 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 low cost passive smart devices such as passive RFID tags and contactless smart cards.

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 compared to the serialized architecture. This in turns yields a good energy consumption per encryption, hence this architecture is well suited for low cost active smart devices such as wireless sensor nodes, RFID reader handhelds, and contact smart cards. Furthermore this architecture can be used to construct a cryptographic coprocessor with very low area consumption and a high throughput.

References

1. Keeloq algorithm (November 2006), `http://en.wikipedia.org/wiki/KeeLoq`
2. Bogdanov, A.: Attacks on the KeeLoq Block Cipher and Authentication Systems. In: 3rd Conference on RFID Security 2007 (RFIDSec 2007) (2007)

3. Bogdanov, A., Leander, G., Knudsen, L.R., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT - An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727 Springer, Heidelberg (2007)
4. Cast Inc. Cast aes32-c, `http://www.cast-inc.com`
5. Hong, S., Lim, J., Lee, S., Koo, B.-S., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Hong, D., Sung, J., Chee, S.: HIGHT: A New Block Cipher Suitable for Low-Resource Device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
6. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES implementation on a grain of sand. In: Information Security, IEE Proceedings, vol. 152, pp. 13–20 (October 2005)
7. Feldhofer, M., Dominikus, S., Wolkerstorfer, J.: Strong Authentication for RFID Systems Using the AES Algorithm (2004)
8. Finkenzeller, K.: RFID Handbook - Fundamentals and Applications in Contactless Smart Cards and Identification, 2nd edn. John Wiley and Sons. Ltd., Chichester (2003)
9. Good, T., Benaissa, M.: Hardware Results for selected Stream Cipher Candidates. In: State of the Art of Stream Ciphers 2007 (SASC 2007), Workshop Record (February 2007)
10. Hämäläinen, P., Alho, T., Hännikäinen, M., Hämäläinen, T.D.: Design and implementation of low-area and low-power aes encryption hardware core. In: DSD, pp. 577–583 (2006)
11. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B.-S., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A New Block Cipher Suitable for Low-Resource Device (2006)
12. Kerckhoff, A.: La cryptographie militaire. Journal des sciences militaires IX, 5–38 (1883)
13. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New Lighweight DES Variants. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 196–210. Springer, Heidelberg (2007)
14. Lim, C., Korkishko, T.: mcrypton - a lightweight block cipher for security of low-cost rfid tags and sensors. In: Kwon, T., Song, J., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 243–258. Springer, Heidelberg (2006)
15. Mace, F., Standaert, F.-X., Quisquater, J.-J.: ASIC Implementations of the Block Cipher SEA for Constrained Applications. In: Proceedings of the Third International Conference on RFID Security - RFIDSec 2007, Malaga, Spain, pp. 103–114 (2007)
16. N.A. Contactless Specifications for Payment Systems - EMV Contactless Communication Protocol Specification. Version 2.0, EMV (August 2007), `http://www.emvco.com/specifications.asp`
17. N.A. Contactless Specifications for Payment Systems - Entry Point Specification. Draft 1.0, EMV (October 2007), `http://www.emvco.com/specifications.asp`
18. Nohl, K., Ploetz, H.: Mifare - little security, despite obscurity. Talk at the 24th Chaos Communication Congress (December 2007)
19. Pramstaller, N., Mangard, S., Dominikus, S., Wolkerstorfe, J.: Efficient aes implementations on asics and fpgas. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2005. LNCS, vol. 3373, pp. 98–112. Springer, Heidelberg (2005)

20. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A compact rijndael hardware architecture with s-box optimization. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 239–254. Springer, Heidelberg (2001)
21. Standaert, F.-X., Piret, G., Gershenfeld, N., Quisquater, J.-J.: Sea: A scalable encryption algorithm for small embedded applications. In: Domingo-Ferrer, J., Posegga, J., Schreckling, D. (eds.) CARDIS 2006. LNCS, vol. 3928, pp. 222–236. Springer, Heidelberg (2006)
22. Visa. Visa payWave FAQ (accessed on 15.02.2008), `www.visa.com`

Appendix

Following abbreviations are used in the subsequent tables
Cur - Current
Tput/Area - Throughput/Area
mFreq - maximum Frequency
mTput - maximum Throughput

Table 5. 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 6. 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 7. 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 8. 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 9. 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 10. 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