Performance Analysis of Contemporary Light-Weight Block Ciphers on 8-bit Microcontrollers

Sören Rinne, Thomas Eisenbarth, and Christof Paar

Horst Görtz Institute for IT Security
Ruhr University Bochum
44780 Bochum, Germany
soeren.rinne@rub.de,{eisenbarth,cpaar}@crypto.rub.de

Abstract. This work presents a performance analysis of software implementations of ciphers that are specially designed for the domain of ubiquitous computing. The analysis focuses on the special properties of embedded devices that need to be taken into account like cost (given by memory consumption) and energy requirements. The discussed ciphers include DESL, HIGHT, SEA, and TEA/XTEA. Assembler implementations of the ciphers for an 8-bit AVR microcontroller platform were analyzed and compared with a byte-oriented AES implementation. While all ciphers fail to outperform AES on the discussed 8-bit platform, TEA/XTEA and SEA at least consume significantly less memory than the AES.

Keywords: microcontroller, software performance, embedded security, ubiquitous computing, SEA, TEA, XTEA, DESL, AES.

1 Motivation

As ubiquitous computing evolves, we have recently seen new ciphers being proposed specifically for the domain of ubiquitous computing. The target of these newly proposed ciphers is not a higher levels of security, as many of these have a shorter key length than the AES. These ciphers rather aim at providing sufficient security in the environment of restricted resources as can be found in many ubiquitous devices. Neither is their focus on a higher maximum performance, but primarily on a smaller footprint, needing less resources such as energy and computing power, and though giving ubiquitous devices a longer lifetime, smaller outline etc. At the same time a security goal of medium to high security is still achievable.

Two of these newly proposed ciphers, namely DESL [17] and HIGHT [11], were designed for a small hardware footprint rather than outstanding software performance. Yet many ubiquitous devices such as MICA Motes [3] and most low cost embedded devices ship with a contemporary low-power 8-bit microcontroller. Hence a software implementation of the cipher might be cheaper after all. Consequently we see a high necessity for a performance analysis of software implementations of these newly proposed light-weight ciphers.

This work features a performance analysis of light-weight ciphers targeted at highly constrained devices. The analysis focuses on the special properties of embedded devices that need to be taken into account like cost (given by memory consumption) and energy needs.

2 Overview of Ciphers

This Section provides a short description of each cipher. An overview of the ciphers' parameters is given in Table 1. Since the parameters of SEA can be chosen, the values that fit our implementation are given in this Table.

Table 1. Characteristic sizes of the focused ciphers

Cipher	AES	DES	DESL	DESX	HIGHT	SEA	TEA	XTEA
Block length	128	64	64	64	64	96	64	64
Key length	128	56	56	184	128	96	128	128
Rounds	10	16	16	16	32	141	32	32

The range of the ciphers is quite huge, starting with DESL comprising a 56-bit key providing only medium security. Other ciphers like HIGHT use a 128-bit key to provide high security but use a smaller block size than AES [4] to meet the needs of a restricted environment. Ciphers like SEA [19] are kept flexible in key size so each user may configure it for the security goal and performance needed.

2.1 AES

The Advanced Encryption Standard (AES) [4], also known as Rijndael, is the successor of the Data Encryption Standard (DES). It was announced by National Institute of Standards and Technology (NIST) as a U.S. FIPS in 2001. The cipher developed by J. Daemen and V. Rijmen was the winner of a 5-year standardization process. It has been deployed widely in many crypto applications, being the de-facto standard symmetric block cipher.

AES is a block cipher using an 128 bit block with an 128, 192 or 256 bit key as input. It operates on a 4×4 array of bytes. Each round of AES consists of four stages, namely `AddRoundKey`, `SubBytes`, `ShiftRows`, and `MixColumns`. The AES is known to be quite efficient, especially on 8-bit architectures, owing to its byte-oriented design. Our assembler implementation of the AES is inspired by the AES implementation of B. Gladman [9].

2.2 DES

The Data Encryption Standard (DES) [8] is a cipher selected as an official Federal Information Processing Standard (FIPS) for the United States in 1976. As a block cipher DES operates on blocks with a size of 64 bits. The key also consists of 64 bits; only 56 of these are actually used by the algorithm, the other ones are parity check bits.

The overall structure consists of a so-called Feistel network with 16 identical base rounds with 8 substitution boxes (S-Boxes), an initial permutation, a final permutation, and a separate key schedule. The whole cipher consists only of bit operations, namely shifts, bit-permutations and exclusive-or operations.

DES is not considered as secure anymore. Thanks to Moore's Law, DES can be broken by exhaustive key search in reasonable time. There are several confirmed DES crackers such as the EFF DES Cracker [7] or the COPACOBANA [14]. Furthermore attacks like differential cryptanalysis, linear cryptanalysis, and Davies' attack [2] have been published.

Yet for some applications where security is not as critical, DES and variants of it are still in use.

DESX The block cipher DESX (or DES-X) [13] is an extension to DES to improve some weaknesses of its predecessor. It is defined by $DESX_{K,K_1,K_2}(M) = K_2 \oplus DES_K(M \oplus K_1)$. It was originally suggested by Ron Rivest in 1984 to protect the cipher DES against exhaustive key-search attacks. DESX is said to be substantially more resistant than DES.

DESL Like the above mentioned DESX DESL (DES Lightweight Extension) [17] is an extension to DES to comply with the requirements of small computational devices like RFID devices or Smart Cards. It was suggested by A. Poschmann et al. in 2006 as a new alternative for ultra-low-cost encryption. To decrease chip size requirements it uses only one S-Box repeated eight times. It therefore requires 38% less transistors than the smallest DES implementation published.

2.3 HIGHT

HIGHT is another block cipher proposed by Deukjo Hong et al. [11] which is working on a 64-bit block length and a 128-bit key length. It was proposed to be used for ubiquitous computing devices such as a sensor in USN or a RFID tag at CHES '06 due to its low-resource hardware implementation. Like many of the discussed ciphers, HIGHT makes use of simple operations such as exclusive-or, addition mod 2^8, and bitwise rotation.

The cipher is a variant of generalized Feistel network. It consists of an initial transformation, 32 rounds using 4 subkeys at a time, a final transformation and a key schedule producing 128 subkeys. HIGHTs key schedule algorithm is designed to keep the original value of the master key after generating all whitening keys and all subkeys. Therefore the subkeys are generated on the fly in encryption and decryption.

2.4 SEA

The Scalable Encryption Algorithm ($SEA_{n,b}$) [19] is designed to be parametric in plaintext/key and processor size. In dependence on given hardware parameters like processor word size, the SEA parameters will be chosen. The main advantages of SEA are efficient combination of encryption/decryption and "on-the-fly" key derivation. It was designed to be an algorithm for small embedded applications like RFID or Smart Cards.

$SEA_{n,b}$ parameters in our case are plaintext/key size $n = 96$, processor word size $b = 8$, and number of words per Feistel branch $n_b = \frac{n}{2b} = 6$. Therefore we have a suggested number of cipher rounds of $n_r = \frac{3n}{4} + 2 \cdot (n_b + \lfloor b/2 \rfloor) = 92$. As we used the standard implementation provided by the author we have 94 rounds.

The cipher is targeted for processors with a limited instruction set and therefore uses only bit operations such as exclusive-or, word rotation, bit rotation, addition mod 2^b, and a substitution box.

2.5 TEA

The Tiny Encryption Algorithm (TEA) was first presented at the Fast Software Encryption workshop in 1994 by David Wheeler and Roger Needham [21]. The focus of the design was simplicity of description as well as implementation.

TEA is a block cipher operating on 64-bit blocks with a 128-bit key. The Feistel structure is dominated by suggested 64 identical rounds consisting of bit-operations like shifts, addition/substraction mod 2^8 and exclusice-or operations.

A number of revisions of TEA has been designed in order to obliterate some weaknesses of the original version. The revisions of the cipher, Block TEA (often referred to as XTEA) and XXTEA (published in 1998), were needed to secure the cipher.

TEA suffers from equivalent keys - each key is equivalent to three others, which means that the effective key size is only 126 bits. Due to this weakness a method for hacking the Microsoft's Xbox game console, where TEA was used as a hash function, has been developed [18]. The cipher is also vulnerable to a related-key attack which requires 2^{23} chosen plaintexts under a related-key pair, with 2^{32} time complexity [12].

2.6 XTEA

As mentioned before the effective key length of TEA is 126 bits not 128. So in 1996 Needham and Wheeler made two adjustments [15]. The first was to adjust the key schedule and the second was to introduce the key material more slowly. With these adjustments the weaknesses should be repaired and the simplicity is almost retained.

3 Framework Set-Up and Tools

In this Section we will show how the ciphers were implemented for the 8-bit AVR architecture. A short introduction to the software development tools and how to measure clock cycles and throughput are given as well.

3.1 Platform Specification

AVR microprocessors are a family of 8-bit RISC microcontrollers. Their memory is organized as a Harvard architecture with a 16-bit word program memory and an 8-bit word data memory. Due to its easy usage, its low power consumption and its comparatively low price, the AVR microcontrollers have reached a high popularity in embedded system design. Most of the AVR instructions working on the 32 registers are handled in one clock cycle. Reading from the program memory, where our look-up tables are stored, costs 3 clock cycles, whereas reading an writing from an to the Flash memory can be performed in 2 cycles.

Since we aimed to implement the ciphers for ubiquitous devices, we took MICA Motes as an adequate target platform. MICA motes (e.g. MICAz, MICA2, MICA2DOT [3]) are designed for development of wireless sensor networks and use an ATmega128 or ATmega128L microcontroller as CPU. The ATmega128(L) is equipped with 128 kbyte of Flash memory and 4 kbytes of SRAM. Yet our implementations of the ciphers, as presented in Section 4, are also executable on smaller AVR devices comprising less SRAM and Flash memory.

3.2 Porting to the AVR Microcontroller

The authors of a cipher usually provide a reference implementation. We had many different programming languages, e.g. C or Java. In order to reduce the size of the code and to reach a maximum performance on our hardware, all of the ciphers were reimplemented in AVR-Assembly language. We implemented them ourselves except for the AES, which is an implementation of the Chair for Communication Security at the Ruhr-University of Bochum, and SEA, which is an existing implementation in assembly language available at [5]. Other implementations of AES on an AVR platform can be found at [16][10]. For other TEA and XTEA implementations on AVR see [6]. To ensure a fair benchmarking process, we used these reference implementations as a starting point for the assembly implementations. The implementations were neither solely optimized for performance only nor for extremely small code size. Instead we tried to yield a good trade-off between both. For the implementation of SEA we made use of an existing version of the author available at [5]. We only made slight changes with respect to our compiler.

In our performance analysis in Section 4 we will use the AES as a reference implementation for the other ciphers.

3.3 Development Tools

For the software development we used the tool Programmer's Notepad 2 [20]. This is a open source text editor with special features for coders hosted on the Windows platform. Programmer's Notepad 2 contains an automatic makefile execution. It compiles the C program, assembles the assembly language program, links it to an ELF file, and then converts it to a COF file. After this procedure has run without errors we used the output file from Programmer's Notepad 2 to execute and debug the code in AVR Studio 4 [1] and simulate it on an ATmega128 device. AVR Studio 4 is an Integrated Development Environment (IDE) for writing and debugging AVR applications on the Windows platform. The tool provides a full-scale debugger which was used to obtain cycle counts of the execution of the ciphers. Cycle counts were used to benchmark the throughput. Code size was measured using the Programmer's Notepad 2 and GCC compiler.

4 Results

In this Section we present the results of our implementations. The results are compared to an implementation of the AES that was optimized for the 8-bit AVR microcontroller environment as well. The comparison focuses on code size, because memory is an important for size and price of an embedded or ubiquitous device, and on execution time, i.e. throughput, as execution time corresponds to the power consumption of a device.

4.1 Memory Usage

As embedded systems development is strongly price-driven, there are high restrictions in the size of available Flash memory and SRAM. This applies even more to applications like ubiquitous computing or even RFIDs, where power consumption is an important issue, too. The Flash (program) memory of the device is used to store program code and look-up tables, if applicable. The smaller SRAM is used for dynamic access during program execution.

Table 2 shows the memory allocation in flash memory of every cipher. Figure 1 visualizes the results ordered by size.

Table 2. Memory allocation of program code in Flash in bytes

Cipher	TEA	XTEA	SEA	DESL	AES	DES	DESX	HIGHT
Code size	1140	1160	2132	3098	3410	4314	4406	5672

As shown in Figure 1, TEA is the smallest cipher followed by XTEA and SEA. DESL is only slightly smaller than AES. The two implementations of HIGHT are using the highest amount of program memory. Yet all of the ciphers could be run on smaller engines than the used ATmega128.

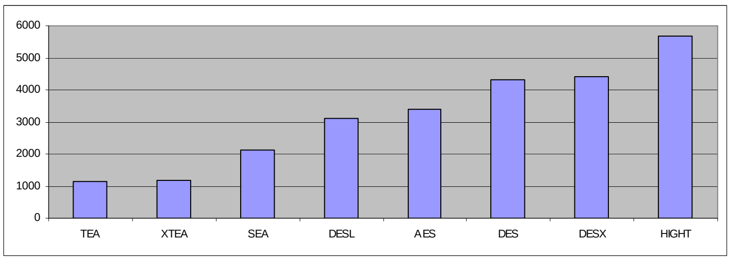

Fig. 1. Code size of ciphers in bytes

4.2 Performance

In the following performance benchmark input and output arrays are of the size of the block size of each cipher. That is to say that we encrypt or decrypt one block with each cipher.

Table 3 shows the number of cycles needed for encryption and decryption for each cipher.

Table 3. Performance of encryption and decryption in measured CPU cycles

Cipher	HIGHT	AES	TEA	XTEA	DESL	DES	DESX	SEA
Encryption	2449	3766	6271	6718	8365	8633	8699	9654
Decryption	2449	4558	6299	6718	7885	8154	8220	9654

Recall that the implementation of HIGHT is the one with the highest use of flash memory. Though in this benchmark it achieves the lowest number of cycles for encryption and decryption of one block of data.

Table 4 and table 5 focus on the throughput of encryption and decryption of each cipher. Column 2 in Table 4 and Table 5 shows the block size in bytes, column 3 replicates the count of cycles from table 3. Column 4 is the quotient of column 3 and 2 and column 5 shows the throughput of encryption/decryption in cycles per byte. The throughput in column 5 is computed assuming the CPU being clocked at 4 MHz.

Figures 2 and 3 reprints the values of Tables 3, 4 and 5 ordered by cycles and respectively by throughput.

Table 4. Throughput of encryption

Cipher	block size [bit]	Encryption [cycles]	Encryption [cycles/bit]	Throughput [bit/sec]
AES	128	3766	29,42	135953
HIGHT	64	3188	49,81	80301
TEA	64	6271	97,98	40823
SEA_96,8	96	9654	100,56	39776
XTEA	64	6718	104,97	38107
DESL	64	8365	130,70	30604
DES	64	8633	134,89	29654
DESX	64	8699	135,92	29429

Table 5. Throughput of decryption

Cipher	block size [bit]	Decryption [cycles]	Decryption [cycles/bit]	Throughput [bit/sec]
AES	128	4558	35,61	112330
HIGHT	64	3188	49,81	80301
TEA	64	6299	98,42	40641
SEA_96,8	96	9654	100,56	39776
XTEA	64	6718	104,97	38107
DESL	64	7886	123,22	32463
DES	64	8154	127,41	31396
DESX	64	8220	128,44	31144

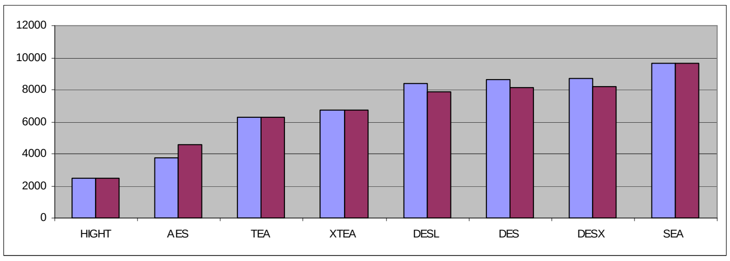

Fig. 2. Cycle count of ciphers

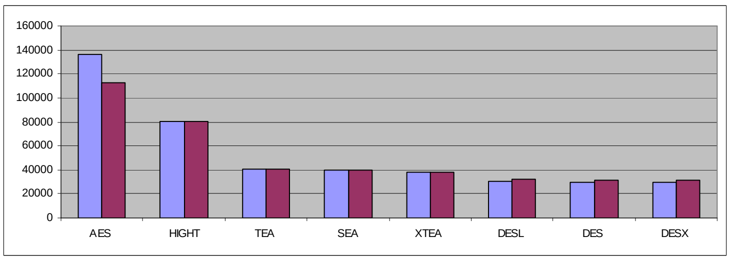

Fig. 3. Throughput of encryption and decryption

Figure 3 shows that the reference AES implementation has a higher throughput than all of the newly proposed ciphers. This is in some cases due to the design objectives of the ciphers. The DES family for example, including DESL, relies on bit permutations which are almost for free in hardware but very expensive in software. This is even true on an 8-bit microcontroller.

4.3 Discussion

Since we did not want to focus solely on code size or on performance, we introduced an additional metric. The ratio of throughput and code size was computed to visualize the combined metric. This metric is given in Figure 4.

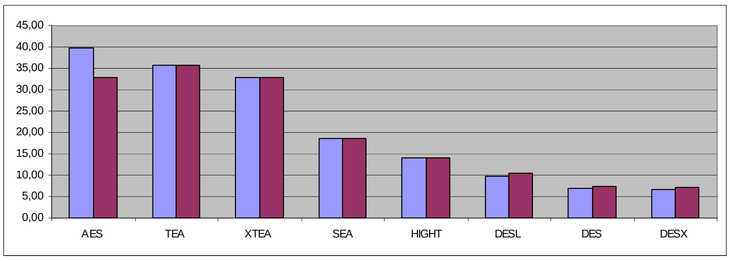

Fig. 4. Throughput-code size ratio of encryption and decryption

Still AES is doing quite well compared to the other ciphers. In this metric the TEA family is at least able to outperform AES in decryption. It can be seen that the ciphers designed for 8-bit software platforms, namely TEA/XTEA and SEA (and AES, of course) outperform the hardware-oriented ciphers HIGHT and the DES family, as expected.

5 Conclusion

We have presented a performance analysis of newly proposed light-weight block ciphers. Target architecture was the 8-bit AVR microcontroller family that can be found in many embedded devices and many applications of ubiquitous computing like wireless sensor networks.

We have shown that many (but not all) of the newly proposed ciphers outperform AES in code size. Especially TEA and XTEA have an extremely small footprint in memory consumption. Yet all of the implemented ciphers were outperformed by the AES in terms of throughput. This might be a disadvantage in wireless devices where computation time means power consumption. The HIGHT was even outperformed by the AES in both, code size and throughput. Though DESL is slightly smaller than AES in code size, it has a worse performance and does not provide comparable security.

As an overall summary one should consider well before using one of these light-weight block ciphers on an 8-bit microcontroller. Only if memory is highly critical, some of the Ciphers might be an alternative to be considered. Usually they just provide a worse performance at comparable or worse security target.

References

[1] Atmel Corporation. Avr studio 4.12, build 498. Available from: http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725.

[2] E. Biham and A. Biryukov. An Improvement of Davies' Attack on DES. In *Proceedings of EUROCRYPT '94*, pages 461–467. EUROCRYPT '94, 1994.

[3] Crossbow Technology Incl. *MPR-MIB User Manual*. Revision B, June 2006. Available from: http://www.xbow.com/Support/Support_pdf_files/MPR-MIB_Series_Users_Manual.pdf.

[4] J. Daemen and V. Rijmen. *The design of Rijndael, the Advanced Encryption Standard*. Springer-Verlag, 2003.

[5] Efton s.r.o. Implementing SEA on x51 and AVR. Available from: http://www.efton.sk/crypt/sea.htm.

[6] Efton s.r.o. TEA (Tiny Encryption Algoritm) a jeho implementacia v 8051 a AVR, 2007. Available from: http://www.efton.sk/crypt/tea_s.htm.

[7] Electronic Frontier Foundation. *Cracking DES*. O'Reilly & Associates, 1998.

[8] Federal Information Processing Standards Publication 46-3. Data encryption standard (des). Technical report, FIPS, 1999.

[9] Brian Gladman. Byte Oriented AES Implementation. Available from: http://fp.gladman.plus.com/AES/.

[10] H.C. Roepke. AVR Implementation of AES. on website. Available from: http://www.christianroepke.de/studium_praktikumB.html.

[11] D. Hong et al. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In *Proceedings of CHES 2006*, 2006.

[12] J. Kelsey et al. Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X New DES, RC2, and TEA. In *First International Conference on Information and Communication Security*, pages 233–246, 1997.

[13] J. Kilian and P. Rogaway. How to Protect DES Against Exhaustive Key Search (an Analysis of DESX). *Journal of Cryptology*, Volume 14:17–35, 2001.

[14] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler. Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker. In *Conference on Special-purpose Hardware for Attacking Cryptographic Systems*, 2006.

[15] R.M. Needham and D.J. Wheeler. Tea extensions. Computer Laboratory, Cambridge, 1997.

[16] B. Poettering. AVRAES: The AES block cipher on AVR controllers. published on website, 2003,2006. Available from: http://point-at-infinity.org/avraes/.

[17] A. Poschmann, G. Leander, K. Schramm, and C. Paar. New Light-Weight DES Variants Suited for RFID Applications. In *Proceedings of FSE 2007*. FSE 2007, 2007.

[18] Matthew D. Russell. Tinyness: An Overview of TEA and Related Ciphers. Draft v0.3, February 2004.

[19] F.X. Standaert, G. Piret, N. Gershenfeld, and J.J. Quisquater. SEA: A Scalable Encryption Algorithm for Small Embedded Applications. Workshop on RFIP and Lightweight Crypto, Graz, Austria, 2005.

[20] Simon Steele. Programmer's Notepad 2, Version v2.0.6.1-ella. Available from: http://www.pnotepad.org/.

[21] D. Wheeler and R. Needham. TEA, a Tiny Encryption Algorithm. In *Lecture Notes in Computer Science*, 1994.