Lightweight Cryptography: Underlying Principles and Approaches

Sergey Panasenko and Sergey Smagin

Abstract—**Lightweight cryptography is a branch of the modern cryptography, which covers cryptographic algorithms intended for use in devices with low or extremely low resources. Lightweight cryptography does not determine strict criteria for classifying a cryptographic algorithm as lightweight, but the common features of lightweight algorithms are extremely low requirements to essential resources of target devices. In this paper we propose generalized approaches to lightweight algorithms design. Also, we highlight some constraints and recommendations for implementation of lightweight algorithms. Finally, we anticipate several trends in lightweight cryptography.**

Index Terms—**Lightweight cryptography; symmetric block ciphers; RFID; matching ciphertext attacks; side-channel attacks**

I. Introduction

Information technologies widely penetrate into people's day-to-day activity. This is one of the main trends of present-day society. An average man's life cannot be imagined without various gadgets. A lot of households use devices with an embedded operating system (besides usual personal computers), which can be connected to the Internet and can even be united into a wireless network. Everywhere people are surrounded by a variety of terminals, readers, sensors etc.

Such expansion of smart technologies crucially raises data security problems. However, now it is impossible to suggest a cryptographic primitive that can be implemented in all types of target devices. We can tell that AES [1] is a really strong algorithm with good performance. It is absolutely advisable to use AES in high-end devices, in a large variety of embedded systems or in some low-end devices (with several constraints). But it is impossible to use common cryptographic algorithms in specific devices with extremely constrained resources. The examples of such devices include:

- RFIDs;
- low-end smart cards (including wireless);
- wireless sensors;
- indicators, measuring devices, custom controllers etc.

The underlying principles and approaches to the design of algorithms intended for use in devices with extremely low resources are slightly different from the design criteria of commonly used cryptographic algorithms. This very specific field is covered by a branch of modern cryptography – lightweight cryptography. Lightweight cryptography does not determine strict criteria for classifying a cryptographic algorithm as lightweight, but the common features of lightweight algorithms are extremely low requirements to essential resources of target devices, including the following:

- size required for hardware implementation;
- computational power of microprocessors or microcontrollers;
- random access memory (RAM);
- read-only memory (ROM) etc.

In this paper we propose a review of a set of lightweight block ciphers. Also, we have tried to analyze and generalize the main approaches to the design of lightweight algorithms, the constraints of their use and the trends of lightweight cryptography.

II. Examples of Lightweight Block Ciphers

This section contains a brief review of several lightweight block ciphers.

A. DESL & DESXL

DESL was proposed in [2]. DESL is based on the classical DES algorithm [3]. Unlike DES, DESL uses a single S-box instead of 8 S-boxes of DES. The design criteria of the single DESL S-box make DESL resistant to most common cryptanalytic attacks [2]. This allows to save a part of ROM for tables storage.

DESXL is a lightweight version of the DESX algorithm [4, 5], which is one of widely used variants of DES. In contrast to DES, DESX performs input and output data whitening with the specific sub keys. Like DESL, DESXL uses the same single S-box instead of 8 DESX S-boxes.

Relatively low resource requirements of DESL/DESXL are just the result of eightfold reduction of ROM requirements for tables storage (since this is the only difference between DESL/DESXL and the classical algorithms). The authors of DESL/DESXL asserted in [2] that such reduction in requirements is enough to use the proposed algorithms in devices with constrained resources with an example of passive RFIDs.

Manuscript received April 20, 2011; revised July 20, 2011.

Dr. S. Panasenko, head of software development department, ANCUD Ltd., Moscow, Russia. E-mail: serg@panasenko.ru.

S. Smagin, senior software developer, ANCUD Ltd., Moscow, Russia. E-mail: serg@ochacovo.ru.

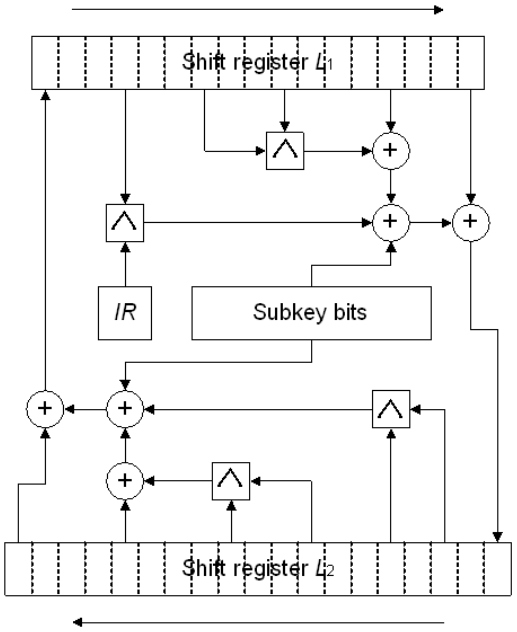

Fig. 1. Structure of KATAN/KTANTAN

B. Curupira

Curupira is a variant of the family of algorithms using the Wide Trail strategy by Joan Daemen [6]. Other examples of such algorithms are AES [1], Anubis [7], and Khazad [8]. Relatively low resource requirements of Curupira are determined by the following set of factors:

- the internal state of the algorithm is relatively small (96 bits; compared to 128 bits of AES internal state for example);
- it is possible to implement 8 X 8-bit Curupira's S-box $S()$ as a composition of two 4 X 4-bit S-boxes $P()$ and $Q()$; this possibility allows to reduce ROM requirements to store the S-boxes; S-boxes $S()$, $P()$, and $Q()$ are entirely inherited from Anubis and Khazad.

The block size of Curupira is 96 bits; it accepts several fixed key lengths: 96, 144, or 192 bits. Data block is represented as a 3 X 4 byte array (the internal state of the algorithm); every round of Curupira modifies the internal state by the following operations [9]:

1) *Nonlinear layer γ;* consists of the parallel application of the *S()* S-box to all bytes of the state.

2) *Permutation layer π;* swaps each column of the state according to the predefined rule.

3) *Linear diffusion layer θ;* performs multiplication of the state by the predefined matrix *D*.

4) *Key addition layer* $\sigma(K_r)$*;* performs bitwise addition of an *r*-round key K_r.

The number of rounds is not determined strictly: the algorithm defines the minimum and maximum numbers of rounds for each allowed key length (from 10 rounds for a 96-bit key to 23 rounds for a 192-bit one). Input whitening is performed before the first round by addition of a K_0 sub key. The final round does not perform the θ operation.

C. Katan & Ktantan

KATAN is a family of block ciphers: KATAN32, KATAN48, and KATAN64. The number in the algorithm's name represents the block size of the algorithm in bits. All the ciphers use 80-bit keys [10].

KTANTAN family also contains three algorithms with the same block sizes and key length. KTANTAN is more compact in hardware – it assumes that the key is burnt into the target device and cannot be changed (also, the key schedule of KTANTAN is much simpler compared to KATAN). Other procedures of KATAN and KTANTAN ciphers are equivalent.

The algorithms' structure is based on the structure of stream cipher trivium [11] (its variant with two registers).

The size of the internal state is equivalent to the block size of the algorithm. Each of KATAN algorithms loads a data block into two internal shift registers L1 and L2. It performs 254 rounds; each of them uses nonlinear functions which form the registers' feedback (Fig. 1).

The registers' sizes and the specific bits used by the nonlinear feedback functions are fixed for every KATAN and KTANTAN algorithm and determined in [10]. One of the nonlinear functions uses specific irregular value (IR) in addition to several register's bits. This value depends on the round's number.

KATAN48 and KATAN64 share the same nonlinear functions with KATAN32, but they use other bits of the internal registers to form the feedback. KATAN48 and KATAN64 also use the larger sizes of the registers in accordance to the block sizes. KATAN32 updates the registers once per a round; KATAN48 and KATAN64 use the nonlinear functions two or three times, correspondingly.

The key schedule of KATAN is based on the linear feedback shift register (LFSR). Another LFSR is used for counting the rounds and to stop the encryption when required. The most significant bit of the latter LFSR forms the above mentioned irregular value.

The resource requirements of KATAN and KTANTAN are extremely low because of the following collection of factors:

- they use the shift registers, which can be very easily implemented in hardware [10]; the feedback functions are also very simple, though they provide the required nonlinearity;
- they process small blocks of data – from 32 to 64 bits;
- their internal state is small and its size is equal to the block size (plus the LSFR for counting the rounds);
- KTANTAN's key schedule is extremely simple.

D. Present

PRESENT is an example of a substitution-permutation network. It performs 31 rounds on 64-bit data block and allows to use 80 or 128-bit keys [12].

Each round consists of the following operations:

- round key addition by XOR operation;
- diffusion layer (S-layer);
- mixing transformation layer (P-layer).

PRESENT is based on the transformation layers of Serpent [13] and DES [3], which have been analyzed in-depth. The diffusion layer performs non-linear substitution of 16 4-bit sub blocks of the current state using similar to Serpent 4 X 4

S-box in parallel.

The mixing transformation layer performs the predefined bit-level permutation. It is based on the mixing transformation layer of DES and seems to be the simplest one to be realized. In software the P-layer can be realized as bit operations or using the according "P-box" table.

Also PRESENT performs the output data whitening after the final round.

E. Hummingbird

Hummingbird encrypts 16-bit blocks of data using a 256-bit key.

The underlying architecture of Hummingbird is original and hybrid (with elements of block and stream ciphers). The encryption procedure can be represented as a continuously working rotor-based machine. Four identical internal block ciphers play a role of virtual rotors. They perform a set of operations on short 16-bit data blocks.

The main components of Hummingbird are (Fig. 2) [14]:

- the internal 16-bit block cipher $E()$: 4-round SP-network with the key addition, S-box application, and linear transformation layers; the final round of the internal block cipher is shortened, but it contains the output whitening procedure;
- four registers of the internal state RS_i;
- 16-bit LFSR.

Hummingbird actively uses 2^{16} modulo addition to mix the internal state registers with the data block to be processed. Alternatively, the high-level structure of the algorithm can be presented as 4-round block cipher with the feedback operations, which allow to use internal cipher blocks chaining as an additional advantage of the algorithm's structure.

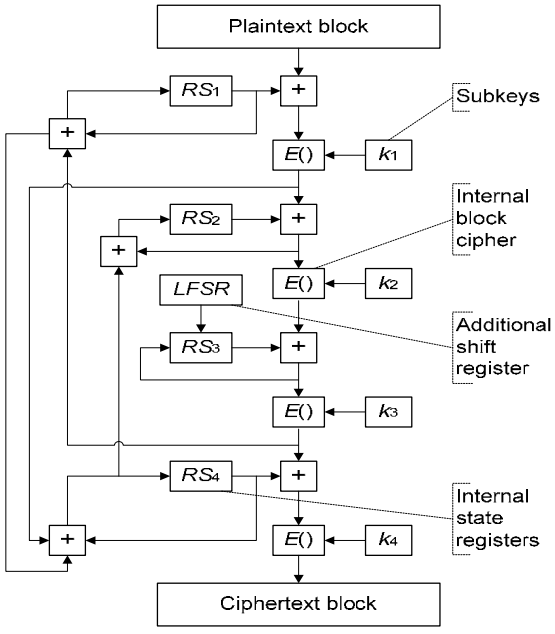

Fig. 2. High-level structure of Hummingbird

Relatively low resource requirements of Hummingbird can be achieved due to simple arithmetic and logic operations and extremely short data blocks.

III. Main Approaches to the Design of Lightweight Algorithms

One of the trends of common cryptography assumes cryptographic primitives to be more complex and "heavyweight". Main parameters of the block ciphers (block size, key length, internal state size etc.) are being increased continuously. This compensates for endless increasing of computer systems' computational power and an avalanche increase in amounts of processed data. Therefore the resource requirements (and complexity of their realization) of the block ciphers grow inevitably to make the block ciphers stronger and more efficient.

This approach is inadequate in principle for embedded systems, especially for systems with extremely low resources. Lightweight cryptography makes implementation cost the most important criterion. The security level and performance of the algorithms should also be adequate, i.e. it is highly desirable to find a compromise between these three characteristics; and such compromise highly depends on the resources of target devices [15].

Using the algorithms reviewed above as an example, we can try to summarize the main methods, which are used by the authors of lightweight algorithms to find the required balance:

- decrease of the main algorithm's parameters: block size, key length (within reasonable limits), and the algorithm's internal state;
- attempts to base the lightweight algorithms upon elements (arithmetic and logic operations, linear or nonlinear transformations etc.) which are widely in use and thoroughly analyzed; this can compensate for some forced decrease of cryptographic strength of the lightweight algorithms;
- simplifying layers of transformations, e. g. decreasing ROM requirements by using 4 X 4 S-boxes (ideally – a single 4 X 4 S-box) or by using their composition to substitute larger sub blocks;
- using low-cost (in implementation) but effective elements, such as data-dependent bit permutations, shift registers etc.;
- designing key schedules that can derive sub keys in-place (i.e. the sub keys do not need to be precomputed) forward or backward;
- using operations that allow implementation trade-offs according to the resources available on the target platform.

IV. Constraints and Compromises of Lightweight Algorithms

As stated above, the design goal of a lightweight algorithm is finding a compromise between low resource requirements, performance, and cryptographic strength of the algorithm (including secure use of the target device with the algorithm inside).

Resource constraints force cryptographers to design lightweight algorithms with small or relatively small block size and key length. In particular, this allows an adversary to

attack the lightweight algorithm using the matching cipher text attack: for an m-bit block cipher equal cipher text blocks can be expected after the encryption of $2^{m/2}$ data blocks (according to the birthday paradox); this can be concerned as a leakage of information about plaintext blocks [16]. The adversary can compose a comprehensive dictionary of the blocks for the current key after the encryption of 2^m data blocks. Thus, the matching cipher text attack is effective against, for example, 16-, 32-, and 48-bit block ciphers. Therefore it is very insecure to use algorithms with small block sizes in the modes like ECB [17] – it is required to provide some constraints for usage of such algorithms: use them in complex modes, change the key routinely etc. Unfortunately such constraints or recommendations produce an additional load to the target device; adherence to the recommendations can be impossible for a specific algorithm (e. g. KTANTAN family of algorithms does not allow to change the key by design).

Consequently, it is required to understand that lightweight cryptographic primitives are targeted at systems with relatively low or medium security requirements or systems, which can take into consideration the specifics of the algorithm to be used, that allow to find the desirable compromise.

The internal state of lightweight algorithms is also designed to be as small as possible. Designers use simpler (compared to classical algorithms) mixing and diffusion transformations. So the internal structures of lightweight algorithms are designed without adequate security margin. This allows crypt analytics to publish a lot of papers with analysis of lightweight algorithms. The analysis of lightweight algorithms is in progress: any algorithm with adequate strength can be broken fully or partially in the not-too-distant future.

Hummingbird is an illustrative example of such algorithms. It uses intensively analyzed structures and operations, but all of the structures are adopted or truncated to meet the requirements of lightweight cryptography. Hummingbird is considered a strong algorithm by its designers [14] and exterior researchers [18]. However, recently Markku-Juhani Saarinen proposed several attacks at Hummingbird's components, which can be used in complex to compromise the whole cipher [19].

Another problem is that a possibility of side-channel attacks against realizations of some lightweight algorithms is not examined at all. For example, the designers of PRESENT stated the following: "Side-channel and invasive hardware attacks are likely to be a threat to PRESENT, as they are to all cryptographic primitives. For the likely applications, however, the moderate security requirements reflect the very limited gain any attacker would make in practice. In a risk assessment, such attacks are unlikely to be a significant factor" [12]. Nevertheless, practical possibility and adaptability of side-channel attacks against RFID are shown in several papers, e. g. [20, 21]. Therefore it is required to use countermeasures against side-channel attacks (see e. g. [22]) when implementing any lightweight algorithm whose structure is potentially susceptible to side-channel attacks. The countermeasures also produce an additional load for the target device.

V. Possible Trends in Lightweight Cryptography

It can be supposed that the evolution of lightweight cryptography will be active in the nearest future, because lightweight algorithms are highly required. Also we can suppose the following trends in designing of lightweight algorithms:

1) Separation of ultra-lightweight algorithms into a specific branch of lightweight cryptography. The spectrum of devices with constrained resources is wide enough: from passive RFIDs to smart-cards with significantly higher resources. After designing a single lightweight algorithm for a whole spectrum of such devices, it can be both technically complicated to implement the algorithm in low-end devices and pointless to use it in high-end devices of the spectrum. Using the algorithms reviewed above as an example, we can see that such separation is already in progress: e. g. KATAN/KTANTAN algorithms require from 2 to 4 times less resources than DESL (when implementing in hardware) [10].

2) *Deepening the divergence between software- and hardware-oriented lightweight algorithms.* The fundamental difference between the requirements to minimize resources in software and hardware was demonstrated by Axel Poschmann in [15]. He examined PRESENT as an example: its round is extremely simple in hardware implementation (Fig. 3), but requires significant resources in software.

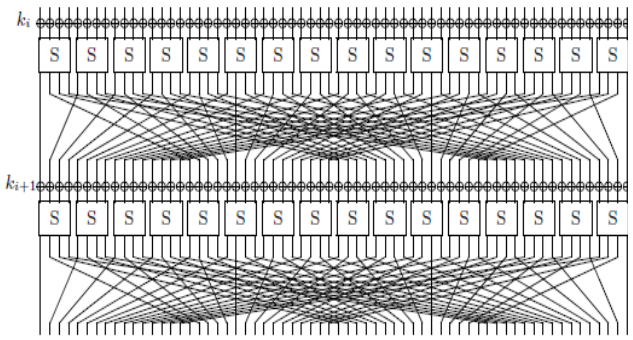

Fig. 3. Structure of PRESENT

VI. Conclusions

In this paper we consider lightweight block ciphers and propose generalized approaches to lightweight algorithms design. We highlight some constraints and recommendations for implementation of lightweight algorithms. Also, we describe compromises which should be reached by designers of lightweight cryptographic primitives. Finally, we anticipate several trends in lightweight cryptography.

Acknowledgement

We would like to thank Alexander Domoratsky for reviewing and valuable comments on this paper.

References

[1] FIPS Publication 197. Specification for the Advanced Encryption Standard. National Bureau of Standards, U. S. Department of Commerce, Washington D. C., November 26, 2001.

[2] G. Leander, C. Paar, A. Poschmann, and K. Schramm. New Lightweight DES Variants. FSE 2007, LNCS, vol. 4593, pp. 196-210. Springer, 2007.
[3] FIPS Publication 46-3. Data Encryption Standard (DES). U. S. Department of Commerce / National Institute of Standards and Technology. Reaffirmed 1999 October 25.
[4] J. Kilian and P. Rogaway. How to Protect DES against Exhaustive Key Search. CRYPTO'96, LNCS, vol. 1109, pp. 252-267. Springer, 1996.
[5] P. Rogaway. The Security of DESX. RSA Laboratories' CryptoBytes, Vol. 2, No. 2, 1996, pp. 8-11.
[6] J. Daemen. Cipher and hash function design strategies based on linear and differential cryptanalysis. Doctoral Dissertation, March 1995, K. U. Leuven.
[7] P. S. L. M. Barreto and V. Rijmen. The Anubis Block Cipher. NESSIE submission, Sept. 2000.
[8] P. S. L. M. Barreto and V. Rijmen. The Khazad Legacy-Level Block Cipher. NESSIE submission, Sept. 2000.
[9] P. S. L. M. Barreto, M. A. Simplício Jr. CURUPIRA, a block cipher for constrained platforms. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC'2007, 2007, Belém.
[10] C. De Cannière, O. Dunkelman, M. Knežević. KATAN & KTANTAN – A Family of Small and Efficient Hardware-Oriented Block Ciphers. CHES'09, LNCS, vol. 5747, pp. 272-288. Springer, 2009.
[11] C. De Cannière and B. Preneel. Trivium Specifications. Available at http://www.ecrypt.eu.org.
[12] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. CHES'07, LNCS, vol. 4727. Springer, 2007.
[13] R. J. Anderson, E. Biham, and L. R. Knudsen. Serpent: A Proposal for the Advanced Encryption Standard. Available at http://www.cl.cam.ac.uk.
[14] D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith. Hummingbird: Ultra-Lightweight Cryptography for Resource-Constrained Devices. FC 2010 Workshops, LNCS, vol. 6054, pp. 3-18.
[15] A. Poschmann. Lightweight Cryptography from an Engineers Perspective. Workshop on Elliptic Curve Cryptography (ECC 2007).
[16] S. Murphy and J. White (editors). NESSIE Public Report D13. Security Evaluation of NESSIE First Phase. 23 September 2001.
[17] FIPS Publication 81. DES Modes of Operation. National Bureau of Standards, U. S. Department of Commerce, Washington D. C., 1980 December 2.
[18] R. Frazer (editor). An Analysis of the Hummingbird Cryptographic Algorithm. Available at http://www.reveresecurity.com. 26 April 2009.
[19] M.-J. O. Saarinen. Cryptanalysis of Hummingbird-1. IACR Cryptology ePrint Archive, report 2010/612.
[20] B. Song. RFID Authentication Protocols using Symmetric Cryptography. PhD thesis, Royal Holloway, University of London, Egham, Surrey, United Kingdom, December 2009.
[21] M. R. Rieback. Security and Privacy of Radio Frequency Identification. PhD thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2008.
[22] A. Poschmann, A. Moradi, K. Khoo, C.-W. Lim, H. Wang, and S. Ling. Side-Channel Resistant Crypto for less than 2,300 GE. Journal of Cryptology (26 October 2010), pp. 1-24

Dr. S. Panasenko received his Ph.D. from Moscow Institute of Electronic Engineering, Russia (2003). Since 1996 he works at ANCUD Ltd. as a software developer and (since 1999) as the head of software development department. He is an author of two books (in Russian) in a field of cryptography. Member of IACSIT (2011). His fields of interest include cryptology and security of computer systems and networks.

S. Smagin Since 2005 he works at ANCUD Ltd. as a senior software developer. His fields of interest include cryptology and security of computer systems and networks.