International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-2, July 2019

Design Considerations of Lightweight Block Ciphers for Low-Cost Embedded Devices

Deepti Sehrawat, Nasib Singh Gill

Abstract*: In the 21st century, Internet of Things (IoT) has become pivotal technology and has attracted worldwide attention in the smart computing environment. In order to enjoy the benefits of this new environment, security is considered as the main aspect in constrained end nodes. Ordinary cryptographic solutions are not sufficient for resource constraint devices. To fulfill this gap, a relatively new field of cryptography called lightweight cryptography came into existence. Furthermore, designing lightweight ciphers especially software oriented poses some significant limits and inherent conditions. Some design considerations of a lightweight encryption algorithm and tradeoffs for minimizing resource requirements are presented in this paper. Also, for a cipher, it is necessary to provide strong resistance against attacks. In this paper different countermeasures to prevent from attacks is also presented. By considering the performance-enhancing ideas presented in this paper, a new GFN based lightweight block cipher is proposed. A cipher design has two main parts; internal structure and key schedule. The internal structure of the proposed design has three layers in which operations are so arranged that it utilizes small code size and has fast diffusion. Besides, design of newly proposed cipher is such that it thwarts the effect of most of the attacks. In the nutshell, this paper serves as a base for designing good lightweight cipher for resource constraint smart IOT environment.

***Index Terms*: design considerations, lightweight block ciphers, performance enhancement, prevention from attacks.**

I. INTRODUCTION

IoT provides a virtual view of real-life things in a resource-constrained environment where security and privacy are considered most important. Everything in Internet of Things (IoT) environment becomes virtual having their own locatable and addressable counterparts on the network. The virtual entities know their physical state and behave accordingly on their own. This view has extended the concept of IoT to "Internet of Everything" by including “anything, anyone, any service”. IoT is also referred to as "Smart Objects Networks" where a huge number of constraint devices are connected to each other in the network [1]. However, embedded devices have some constraints in the form of low processing power, limited memory, low connectivity, and low energy consumption. Although ordinary cryptographic solutions focus on high-level security but due to limited resources, it is not so easy to successfully apply ample cryptographic functions on IoT based resource-constrained devices. A relatively new field of cryptography known as “Lightweight cryptography” is tailored for resource-constrained environments [2]. In lightweight cryptography huge variety of resource constraint devices uses different communication technologies. They have some limitations in the form of implementation memory size, speed, power, security, performance and as a result it is not possible to implement standard cryptographic solutions. So there is a need of some tradeoffs in the above parameters in resource-constrained devices. In this era of pervasive computing, lightweight cryptography expects simple and fast security solutions [3]. Memory consumption (RAM and ROM size), speed and throughput are among the main primitives which are considered towards finding good lightweight algorithms [3]. These are specific to the hardware or software implementation of a security algorithm. Authors in [4] mentioned 6 metrics for evaluation of software implementation of lightweight ciphers, these are:

1. Code size is given in bytes,
2. RAM usage is given in bytes,
3. Cycle count in encryption, one block Cycles/byte
4. Cycle count in decryption, one block Cycles/byte
5. Energy consumed given in µJ, and
6. A combined metric (code size X cycle count product) which is given after the normalization, by block size.

There is a strong correlation between energy consumption and cycle count [4]. Lightweight algorithms implemented in software, targeted for micro-controllers usually consider some more performance evaluation parameters like:

1. Throughput measured in bytes per CPU cycle.
2. Power Consumption

Among all the primitives, memory is considered to be the most prevailing part. Optimized software implementations result in fast speed thereby utilizing low power consumption. The rest of the paper is organized as follows: In section 2 performance enhancement ideas are enlightened which can be considered for the design of a software implementation of LBC (lightweight block ciphers) for their performance enhancement. Section 3 presents newly proposed lightweight block cipher and finally section 4 concludes the paper.

Revised Manuscript Received on 30 July 2019.
***** Correspondence Author
Deepti Sehrawat*, Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak, Haryana, India.
Nasib Singh Gill, Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak, Haryana, India.

II. ENHANCING PERFORMANCE THROUGH TRADE-OFFS

It is really a challenging task to select the best algorithm satisfying both memory requirements and energy efficiency [1].

This section presents some useful performance enhancement implementation ideas to improve the software based primitives.

A. S-Boxes

S-box provides nonlinearity in a cipher as a result, security of a cryptosystem can be significantly influenced by their choice. Some performance enhancement criteria for optimized S-box implementations are:

1. Considering the energy consumption per cycle, the S-boxes which are having lower signal delay must be favored over others. 4-bit S-box is better and proficient than that of 8-bit S-boxes because these are having a lower signal delay. The ratio of energy consumption by 4-bit S-boxes and 8-bit S-boxes is 1:2. Conversely, 8-bit S-boxes provides more nonlinearity and lower values for DP/LP coefficient. A greater number of rounds are required by an implementation using 4-bit S-boxes to attain the same level of security margin as that by using 8-bit S-boxes [5]. Considering energy efficient design, using 4-bit S-boxes is favored over using 8-bit S-boxes and finally, to judge for the best S-box it is required to examine clustering of linear and differentials trails [6].
2. 8-bit S-boxes require around forty times more area than that required by 4-bit S-boxes [7]. 4-bit S-boxes are cryptographically weaker than 8-bit S-boxes. Moreover, S-box of LBlock lightweight cipher provides the best nonlinearity, good differential probability, and good algebraic order [8].
3. Performance enhancement and optimization are done at the algorithmic level and, it is good to store S-boxes in Flash or RAM [2, 9].
4. Constructing 8-bit S-boxes by combining 4-bit S-boxes results in decreased memory requirements by implementing S-boxes as look-up tables. But, this results in increased execution time [10].

B. Key Schedule

Attacker's main goal is to break the cipher under a particular key. To limit the data available to the attacker, changing key frequently provides resistance to the related key attacks [11]. So, in this direction, the use of long keys is encouraged because the use of short keys makes exhaustive key search become feasible. Most of the lightweight security algorithms differ in the key schedule from their non-lightweight counterparts. A complex key schedule results in the increase in RAM size or gate area. For lightweight algorithms, an increase in memory size is not acceptable so it is common to not consider related key attacks in lightweight cryptography and allows to use simple key scheduling. Encryption and decryption of Feistel ciphers are the same which results in low memory usage [12]. If keys can be chosen independently and randomly then it provides resistance against related-key attacks [3].

C. Internal State

To minimize the constraints posed by IoT, most of the lightweight block algorithms considered to use smaller internal states in the form of block sizes and smaller key sizes. As a result, most of the block ciphers in IoT use a block size of 64-bits. This results in low memory footprints in software implementation as well as in hardware implementation of the security algorithm. This reduced block size, as a result, proves to be problematic because by doing so the security level of some modes of operation like CBC (Cipher Block Chaining) erodes rapidly when the encrypted number of blocks of size n-bit reaches $2^{n/2}$ [13]. Small key sizes usually 80 bits or smaller offers slight security margin against brute-force search. Time-memory-data tradeoff can turn out to be an issue if in multi-key setting key size is very small [14]. XORing actual key prior to its use thwarts the weak key attacks in a cipher. Same was realized by the authors in [15] in which a lightweight algorithm SIT was proposed in which actual key is not used, the key is first XORed before its use.

D. Prevention from Attacks

After a deep analysis of attacks on Lightweight Block ciphers in a smart environment, countermeasures to prevent from these attacks are concluded in the publication in [16].

1. XOR function is mostly used in ARX-based ciphers. Using addition mod 2b is more useful in place of bitwise XOR because for several reasons. Using a mod 2b key addition improves the diffusion layer by introducing sufficient non-linearity. This is done at the same cost and same speed as by bitwise XOR. It also helps to provide enough resistance against structural attacks [17].
2. Moreover, a bit-slice implementation in comparison to table-based implementation proves to be safer for cache and timing attacks (implementation attacks) [18]. When bitwise operations like AND and XOR are applied on w-bits words, it results in parallel evaluation of S-box w-times [19].
3. To provide provable security to a cipher against two most widely used statistical analysis i.e. linear and differential cryptanalysis, some methodology proposed by the authors in [20] can be adopted in cipher's design like:
   - Change in location of round function to enable parallel computation without compromising security.
   - Introduction of different sizes substitution boxes in a round or in different rounds which also provides resistance against algebraic attacks. A similar study in [20] gives CLEFIA cipher which uses two different 8-bit S-boxes, first 8-bit S-box is created by randomly choosing 4-bit S-boxes and second is given by an inverse function using GF (2^8). With this mixing CLEFIA provides resistance against algebraic attack.
   - We can recursively construct the round functions which also results in the reduction of the size of substitution boxes.

E. Minimizing Resource Requirements in Software Implementation

Some tradeoffs for minimizing resource requirements in IoT enabled smart environment are described here, it includes carefully considering RAM/ ROM sizes of software implementations.

1. To lower the decryption implementation cost of a cipher, components used in lightweight block ciphers are required to be involutions. S-boxes or Feistel structures can be used to introduce non-linear involutions. An alternate approach is by introducing those operations in which decryption is not required. For instance, it is possible to save the ROM size by using a block cipher in counter mode rather than storing the instructions for the inverse operations of the block cipher. Ciphers having involution components have overhead due to the implementation of inverse key-scheduling. This overhead can be lowered by introducing certain symmetry in the sub-key sequence however simultaneously it results in weak keys and has potential slide attacks [19].
2. To minimize the constraints posed by IoT, most of the lightweight block algorithms considered to use smaller internal states in the form of block sizes.
3. The code size and the RAM size requirement are considered towards inclusion in the standard. (Less resource utilization in comparison to the prevailing standards on the same platform are well-thought-out as possibly lightweight for software environments).
4. Loading information from RAM to CPU registers and vice versa is considered as a costly operation. So it is required to limit such operations so as to decrease the RAM consumption and time complexity of the algorithm [20].
5. Resource requirements in software implementations can be measured by counting registers required along with the RAM and ROM size requirement in bytes. Functions using a lower number of registers have comparably low calling overhead. This is because fewer variables are placed on the stack before overwriting registers. Program code, S-boxes, hardcoded round keys are stored in ROM while for intermediate values of computations, RAM is used. A trade-off between on the fly calculations versus lookup tables is another issue to be considered [23].
6. To speed up the operations in a cipher, copying constant ROM data to RAM is a good option. One cycle is required to read a RAM byte to the register and four cycles are required to read from ROM. Copying constant ROM data to RAM before execution improves the overall performance of a routine [3].
7. For optimization, using same size bitwise operations as the platform is, provides good efficiency because these can be directly mapped to the ARM instructions. For example, using 32-bitwise ARX operations for a 32-bit platform can be mapped without additional effort to 32-bit ARM instruction [24]. It is good to define the integer type length explicitly which are available in inttypes.h like int8_t in place of char, int32_t in place of int. this gives consistent behavior and when defined for local variables, it improves register usage.
8. Using lookup tables (LUTs) in place of substitution tables results in small code size. This also speeds up the processing, as a result, low power consumption. To realize higher throughput, many ciphers use larger LUTs. Larger LUTs results in an increase in code size. So maintaining a balance is a design issue which is to be considered wisely [7]. The maximum information leakage operation is table look-up operation [25]. Furthermore, computing large tables of constants on the fly in place of lookup tables is also desirable when there is limited memory available. It also offers security implications if the indices are data dependent and in cases where processors contain a cache because of timing differences due to varying access times can be used as a side channel [1].
9. For efficient register usage and avoiding branching in programming, loop unrolling can be used. This speed up the execution but simultaneously increases the code size. Partial loop unrolling is another technique which helps to achieve good execution speed without increasing the code size too much. LEA lightweight block cipher also implemented this technique for speed optimization [7, 24]. On the other hand, code rolling into a loop can be used for limiting the ROM size [26, 27]. This is all according to the requirement of a particular application.
10. A smaller code size can be achieved by doing function calls for a common operation like n-bit XORs, but this results in affected throughput as with each function call overhead is associated which adversely affect the throughput. SIMON and SPECK low-flash implementations make use of this concept. They also proposed that additional code size savings can be made for both SPECK and SIMON but only by significantly reducing the throughput [28]. The memory size of less than 1 KB is considered as a good choice for lightweight block cipher implementations [29].
11. Most effective operations are modular arithmetic operations like addition-modulo and subtraction-modulo. Rotations are not considered good operations, especially rotations of the same amount on neighboring large operands. Furthermore, different rotation amounts offer a reasonable amount of different trade-offs between efficiency and security (especially linear and differential probabilities) [30]. So choosing carefully best rotation amounts is considered as a good decision in a cipher design [31].
12. For the most efficient implementation choosing the word size is an important decision, it is advisable to choose the word size identical to the microcontroller's register size. Furthermore, if the register size of the microcontroller is greater than that of word/ operand size, the worst results are obtained. Consequently, normal implementation efficiency is achieved when the size of word is multiple of microcontroller's register size [30].

As in IOT applications, a variety of devices are used, so a cryptographic solution must perform well on wide-ranging microcontrollers and supporting different register sizes. To attain this, the cipher's word size and largest register size supported by microcontroller should be equal.

F. Some other Performance Enhancement Trade-Offs

Structure of a cipher has its impact on its performance, here some considerations are given which is related to the internal structure of a cipher and plays an important part in performance enhancement.

1. Symmetric key algorithms use mathematical functions to achieve significant confusion and diffusion in each round. More rounds improve security but resulting in more energy consumption [31].
2. Using type-2 generalized Feistel structure has advantages like parallelism can be achieved in the execution of F-function, it allows smaller size F-functions to be used. But this also requires more number of rounds and usage of small F-functions results in a slow diffusion rate. Authors in [32] used the DSM (Diffusion Switching Mechanism) technique to get rid of these limitations. In DSM, two different matrices are used for better diffusion and thus improves resistance against linear and differential attacks.
3. Attacks that depend on the slow diffusion in Feistel ciphers and highly controllable by round keys as a result of a simple key schedule can be avoided by using more rounds in Feistel ciphers. For instance, the Feistel cipher has fewer round function than SPNs, as Feistel ciphers update only half of the data in every round. Besides, decryption in Feistel ciphers can be implemented without much effort and cost, as it is done by simply reversing the order of operations [33].
4. LTS (Long Trail Strategy) advocates large S-boxes through ARX-based implementation and sparse linear layers. With this strategy, it is possible to bound the linear and differential characteristics of a cipher for any number of rounds [12].
5. For performance enhancement and improved security, authors in [26] proposed a family of 4 x 4 diffusion layers called recursive diffusion layers. Proposed diffusion layers use linear Feistel structure rounds. SPN structures can deploy these diffusion layers because these are having simple inverses. The proposed diffusion layers have provable security against linear and differential cryptanalysis.

III. PROPOSED DESIGN

Lightweight block ciphers have comparably smaller internal states in the form of key size and block size. Lightweight ciphers vary from traditional cryptographic algorithms. These can be implemented either through software implementation or through hardware implementations. Software oriented cipher designs provide more flexibility at lower costs on manufacturing and maintenance as compared to hardware implementations. Even providing strong resistance against mathematical attacks could not protect hardware-oriented block ciphers from side channel attacks thereby losing its keys. So a good software design is required which has low memory utilization and enhanced efficiency along with enough security guard against attacks. ARX (modular Addition-Rotation-bitwise XOR) designs have even more efficient software implementations. So, considering this in mind, a lightweight Feistel type block cipher is developed which have low decryption overhead. Description of a newly proposed lightweight block cipher is given into two parts: the first part is data processing part and the second part is the key scheduling part. The proposed design employs a 4-branch Generalized Feistel Network (GFN) that reduces restriction on the designing part of inner auxiliary functions. The round function of a GFN is comparatively lighter than those of SPN structures, as decryption can be easily implemented from encryption without much effort. For $k \in GF(4)^n$, $4n$ round function is a key-dependent function which is a map given by (1):

$$R_k = GF(4)^n \times GF(4)^n \rightarrow GF(4)^n \quad (1)$$

This map is defined by various operations performed on it like key whitening, ARX operations and round permutation in three layers, these are given in “Fig. 1”.

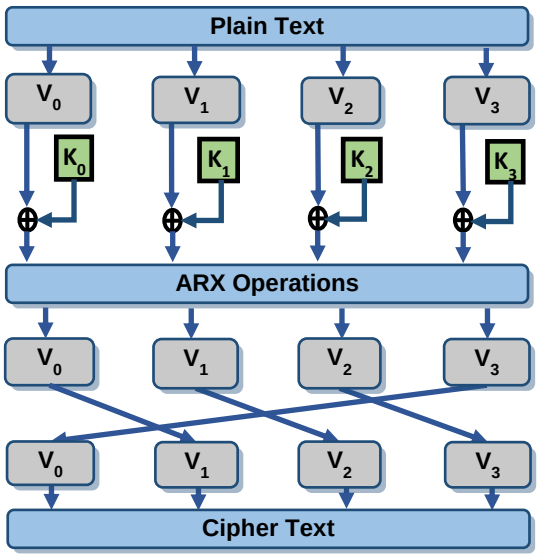

Figure 1: Layers in the proposed ciphers

Key whitening is applied as a first step in the newly proposed cipher because it is relatively easy to map the block cipher with detectable weakness where actual key drives the non-linear operations of the cipher. So, keeping this point in mind pre key whitening is applied i.e. the key is XORed prior to its use in the round function. ARX designs have very efficient software implementations and efficiently implements the encryption, decryption and key schedule procedures in a parallel way. Modular addition, XOR and circular shift operations are then arranged in the second layer in a way that provides fast diffusion and low code size. Usually, GFN has slow diffusion property and round permutation provides faster diffusion.

Fast diffusion in a cipher thwarts those attacks that are made possible because of the slow diffusion property of block ciphers. Therefore, in the last third layer, a round permutation is applied at the end of each round that increases confusion and diffusion to a great extent. By considering the performance-enhancing ideas and countermeasures to protect from attacks, design of newly proposed cipher is such that it thwarts the effect of most of the attacks. In decryption, the same round function is used but here the order of the round keys, constants, and operations are reversed. Key scheduling, being a crucial component of a block cipher can break down entire design if otherwise not designed cleverly.

Weak key scheduling gives classes of weak keys i.e. keys for which it is easy to break the cipher. In key scheduling part, a user-defined master key is stored in the register key, represented by (2):

$$Key = k^n k^{n-1} k^{n-2} \dots k^2 k^1 k^0 \quad (2)$$

Instead of using the entire key, a sub-key is derived from the master key which is fed to the rounds, each round uses different keys. More rounds introduce more confusion and diffusion and hence as a result provides more security.

IV. CONCLUSION

Designing a security algorithm for IoT enabled devices must consider some constraints posed by resource constraint devices. Design and implementation of a lightweight cipher go simultaneously and this has revealed some significant limits and inherent conditions. For minimizing resource requirements, some design considerations and tradeoffs are presented in this paper. The main aim of this paper is to present ideas to achieve efficient and optimal software implementation of LBC for IoT-enabled low-resource embedded devices. Besides, countermeasures to prevent from attacks is also presented. Based on these design considerations and countermeasures, a security model for IoT enabled devices is presented in this paper. The proposed design employs a 4-branch GFN defined by various operations performed on it in three layers like key whitening, ARX operations and round permutation. The operations in the proposed design are so arranged that it not only provides fast diffusion it also utilizes low code size. This work helps the researchers in the area of IoT security. This paper set a base for further research work and in the near future, we will evaluate the performance of proposed LBC for IoT enabled environment.

REFERENCES

1. M. Katagi, S. Moriai, “Lightweight cryptography for the Internet of Things,” Sony Corp., 2008, pp. 7–10. https://pdfs.semanticscholar.org/9595/b5b8db9777d5795625886418d38864f78bb3.pdf
2. I. K. Dutta, B. Ghosh, M. Bayoumi, “Lightweight Cryptography for Internet of Insecure Things: A Survey,” In 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0475-0481. IEEE, 2019. 10.1109/CCWC.2019.8666557
3. M. Matsui, Y. Murakami, “Minimalism of software implementation,” In: International Workshop on Fast Software Encryption, Springer, Berlin, Heidelberg, 2013, pp. 393-409. https://doi.org/10.1007/978-3-662-43933-3_20
4. T. Eisenbarth, Z. Gong, T. G¨uneysu, S. Heyse, S. Indesteege, S. Kerckhof, F. Koeune, T. Nad, T. Plos, F. Regazzoni, F. X. Standaert, L. O. Oldenzeel, "Compact implementation and performance evaluation of block ciphers in AT tiny devices,” In: International Conference on Cryptology in Africa, Springer, Berlin, Heidelberg, 2012, pp. 172-187. https://doi.org/10.1007/978-3-642-31410-0_11
5. B. Subhadeep, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, F. Regazzoni, “Midori: A block cipher for low energy,” In: In International Conference on the Theory and Application of Cryptology and Information Security, Springer, Berlin, Heidelberg, 2014, pp. 411-436. https://doi.org/10.1007/978-3-662-48800-3_17{83}
6. W. Zhang, Z. Bao, V. Rijmen, M. Liu, “A New Classification of 4-bit Optimal S-boxes and its Application to PRESENT , RECTANGLE and SPONGENT,” In: International Workshop on Fast Software Encryption, Springer, Berlin, Heidelberg, 2015, pp. 494-515. https://doi.org/10.1007/978-3-662-48116-5_24
7. T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, L. Uhsadel, “A survey of lightweight-cryptography implementations,” Design & Test of Computers, IEEE, vol. 24, no. 6, 2007, pp. 522–533. 10.1109/MDT.2007.178
8. W. Wu, L. Zhang, “LBlock: A Lightweight Block Cipher,” In: International Conference on Applied Cryptography and Network Security, Springer, Berlin, Heidelberg, 2011, pp. 327-344. https://doi.org/10.1007/978-3-642-21554-4_19
9. C. Alippi, A. Bogdanov, F. Regazzoni, “Lightweight cryptography for constrained devices,” In: Proceedings of the 14th International Symposium on Integrated Circuits, ISIC 2014, pp. 144–147. 10.1109/ISICIR.2014.7029580
10. B. Gérard, V. Grosso, M. N. Plasencia, F. X. Standaert, “Block ciphers that are easier to mask: How far can we go?,” In: International Workshop on Cryptographic Hardware and Embedded Systems, Springer, Berlin, Heidelberg, 2013, pp. 383-399. https://doi.org/10.1007/978-3-642-40349-1_22
11. N. Mouha, “The Design Space of Lightweight Cryptography,” In: NIST Lightweight Cryptography Workshop, Gaithersburg, United States, 2015. https://hal.inria.fr/hal-01241013/document
12. D. Dinu, L. Perrin, A. Udovenko, V. Velichkov, J. Großschädl, A. Biryukov, “Design strategies for ARX with provable bounds: Sparx and LAX,” In: International Conference on the Theory and Application of Cryptology and Information Security, Springer, Berlin, Heidelberg, 2016, pp. 484-513, https://doi.org/10.1007/978-3-662-53887-6_18
13. K. Bhargavan, G. Leurent, “On the practical (in-) security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN,” In: ACM CCS 2016 - 23rd ACM Conference on Computer and Communications Security, Vienna, Austria, ACM, 2016, pp. 456-467. 10.1145/2976749.2978423
14. A. Biryukov, S. Mukhopadhyay, P. Sarkar, “Improved time-memory trade-offs with multiple data,” In: International Workshop on Selected Areas in Cryptography, Springer, Berlin, Heidelberg, 2005, pp. 110-127. https://doi.org/10.1007/11693383_8
15. U. Muhammad, I. Ahmed, M. I. Aslam, S. Khan, U. A. Shah, “Sit: A lightweight encryption algorithm for secure internet of things,” International Journal of Advanced Computer Science and Applications, vol. 8, no. 1, 2017. 10.14569/IJACSA.2017.080151
16. D. Sehrawat, N. S. Gill, “Attacks on Lightweight Block Ciphers and their Countermeasures,” Journal of Engineering and Applied Sciences, vol. 13, Issue: 20, 2018, pp. 8439-8447. 10.3923/jeasci.2018.8439.8447
17. F. X. Standaert, G. Piret, N. Gershenfeld, J. J. Quisquater, “SEA: A scalable encryption algorithm for small embedded applications,” In: In International Conference on Smart Card Research and Advanced Applications, Springer, Berlin, Heidelberg, 2006, pp. 222-236. https://doi.org/10.1007/11733447_16
18. M. Bellare, T. Kohno, “A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications,” In: In International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, Heidelberg, 2003, pp. 491-506. https://doi.org/10.1007/3-540-39200-9_31
19. A. Biryukov, L. Perrin, “State of the Art in Lightweight Symmetric Cryptography,” IACR Cryptology ePrint Archive (2017):511, 2017. http://eprint.iacr.org/2017/511
20. M. Matsui, “How far can we go on the x64 processors?,” In: International Workshop on Fast Software Encryption, Springer, Berlin, Heidelberg, 2006, pp. 341-358. https://doi.org/10.1007/11799313_22

21. T. Shirai1, K. Shibutani, T. Akishita, S. Moriai, T. Iwata, "The 128-bit blockcipher CLEFIA," In: International Workshop on Fast Software Encryption, Springer, Berlin, Heidelberg, 2007, pp. 181-195. https://doi.org/10.1007/978-3-540-74619-5_12
22. A. Biryukov, L. Perrin, A. Udovenko, "Reverse-engineering the S-box of Streebog, Kuznyechik and STRIBOBr1," In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, Heidelberg, 2016, pp. 372-402. https://doi.org/10.1007/978-3-662-49890-3_15
23. K. A. McKay, L. Bassham, M. S. Turan and N. Mouha, "Report on lightweight cryptography," US Department of Commerce, National Institute of Standards and Technology, DRAFT NISTIR, 2017, pp. 1–29. https://doi.org/10.6028/NIST.IR.8114
24. S. Hwajeong, I. Jeong, J. Lee, W. H. Kim, "Compact Implementations of ARX-Based Block Ciphers on IoT Processors," ACM Transactions on Embedded Computing Systems (TECS), vol. 17, no. 3, Article No. 60, 2018. 10.1145/3173455
25. A. Biryukov, L. Perrin, "State of the Art in Lightweight Symmetric Cryptography," IACR Cryptology ePrint Archive, 2017, pp. 511, 2017. http://eprint.iacr.org/2017/511
26. M. Sajadieh, M. Dakhilalian, H. Mala, P. Sepehrdad, "Recursive diffusion layers for block ciphers and hash functions," Fast Software Encryption, Springer, Berlin, Heidelberg, 2012, pp. 385-401. https://doi.org/10.1007/978-3-642-34047-5_22
27. G. Leander, "On linear hulls, statistical saturation attacks, PRESENT and a cryptanalysis of PUFFIN," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6632 LNCS, 2011, pp. 303–322. https://doi.org/10.1007/978-3-642-20465-4_18
28. R. Beaulieu, D. Shors, J. Smith, S. T. Clark, B. Weeks, L. Wingers, "The SIMON and SPECK lightweight block ciphers," In: Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE, 2015, pp. 1-6. 10.1145/2744769.2747946
29. A. Baysal, S. Şahin, "Roadrunner: A small and fast bitslice block cipher for low cost 8-bit processors," In: International Workshop on Lightweight Cryptography for Security and Privacy, Springer, Cham, 2015, pp. 58-76. https://doi.org/10.1007/978-3-319-29078-2_4
30. D. D. Dinu, Efficient and secure implementations of lightweight symmetric cryptographic primitives (Doctoral dissertation, University of Luxembourg, Luxembourg), 2017. http://hdl.handle.net/10993/33803
31. C. Rajarathnam, S. Bapatla, K. P. Subbalakshmi, R. N. Uma, "Battery power-aware encryption," ACM Transactions on Information and System Security (TISSEC), vol. 9, no. 2, 2006, pp. 162-180. 10.1145/1151414.1151417
32. D. Dinu, A. Biryukov, J. Großsch¨adl, D. Khovratovich, Y. L. Corre, L´. Perrin, "Felics–fair evaluation of lightweight cryptographic systems," In: NIST Workshop on Lightweight Cryptography, vol. 128, 2015. https://www.cryptolux.org/index.php/FELICS
33. F. Karakoç, H. Demirci, A. E. Harmancı, "AKF: A key alternating Feistel scheme for lightweight cipher designs," Information Processing Letters, vol. 115, no. 2, 2015, pp. 359-367. https://doi.org/10.1016/j.ipl.2014.10.010 10.1145/1151414.1151417

AUTHORS PROFILE

Ms. Deepti Sehrawat has passed Master of Science and Applications from Department of Computer Science & Applications, M. D. University, Rohtak, India in 2008 and Master of Philosophy from C. D. L. University, Sirsa in 2009. She is currently pursuing Ph. D under the supervision of renowned academician and researcher – Professor Nasib Singh Gill of M. D. University. She has published more than 25 research papers in reputed National and International Journals and Conference Proceedings including IEEE. Her main research work focuses on IoT, Lightweight Cryptography Algorithms, Network Security and Privacy, Big Data Analytics and Data Mining. She has 8 years of teaching experience.

Dr. Nasib Singh Gill is at present senior most Professor of Department of Computer Science & Applications, M. D. University, Rohtak, India and is working in the Department since 1990. He has earned his Doctorate in Computer Science in the year 1996 and carried out his Post-Doctoral research at Brunel University, West London during 2001-2002. He is a recipient of Commonwealth Fellowship Award of British Government for the Year 2001. Besides, he also has earned his MBA degree. He has published more than 245 research papers in reputed National & International Journals, Conference Proceedings, Bulletins, Edited Books, and Newspapers. He has authored seven books. He is a Senior Member of IACSIT as well as a fellow of several professional bodies including IETE and CSI. He has been serving as Editorial Board Member, Guest Editor, Reviewer of International/National Journals and a Member of Technical Committee of several International/National Conferences. He has guided so far 8 Ph.D. scholars as well as guiding about 7 more scholars presently in the areas – IoT, Information and Network Security, Computer Networks, Measurement of Component-based Systems, Complexity of Software Systems, Decision Trees, Component-based Testing, Data mining & Data warehousing, and NLP.