Shared Key Encryption by the State Machine with Two-Dimensional Random Look-up Table

Michael Lifliand

Cisco Systems Israel, HaMelacha st, P.O.Box 8735, Netanya, 42504, Israel
mlifliand@gmail.com

Abstract. This paper describes a new approach to creation of fast encryption methods with symmetric (shared) key. The result solution is intermediate one between block and stream ciphers. The main advantages of both types of ciphers are realized, while most of disadvantages are eliminated. The new approach combines encryption with built-in calculation of the hash for the data integrity, pseudo-random generator, and option for dual shared keys. These properties are pivotal for secure applications. These new methods may be designed for different size of shared secret key. Both software and hardware implementations of the new methods are fast and simple and may be used in various security applications. Presented cryptanalysis proves basic features and gives an option to design actual provable security methods.

Keywords: Shared key encryption, block cipher, stream cipher, chaining, initialization vector, look-up table, pseudo-random generator, hash, data integrity, message authentication code.

1 Introduction

Security applications use different ciphers to solve a variety of security problems. However, faster and more secure methods of shared key encryption combined with authentication for use in cheap and simple client devices are still needed.

Most known methods of shared key encryption may be classified as either block or stream ciphers [1, 2]. Block ciphers are mostly considered to be more secure and are used in network IPSec and other applications. Block ciphers are based on dozens of shift and exclusive-OR (XOR) conversions of 64-256 bit data units and their parameters are defined by the shared secret. Stream ciphers are generally faster and cheaper to implement. They mostly use XOR operations of the input data with the pseudo-random sequence. The latter must be unique and be generated for each encrypted packet of data. Pseudo-random generators of both the sender and receiver are synchronized by the shared key and an initialization vector. The latter is mostly located and sent as a clear text in each packet of data after encryption.

Security in networking defines the message authentication code (MAC) calculated for the encrypted payload (cipher text) and optionally for an additional header of the packet. MAC is calculated in a separate way. There is an important challenge [2, 7] in usage of encryption as a basic part of the MAC calculations. This may be done by dependency of the cipher state and the input data [7].

The main goal of this work is to combine generic features of block and stream ciphers. The next issue is a strengthening encryption by involving more elements of security protocols in the encryption primitives. The basic targets are chaining and initialization vectors.

This work describes a new approach and new methods along with their cryptanalysis. It is organized as follows. Section 2 describes the basic encryption block and its properties. Sections 3-6 define and analyse the simplest and enhanced schemes of encryption. Rest sections elaborate and summarise the features of designed encryption methods.

2 Two-input basic encryption block with chaining

All the data is represented as sequence of data units of **n** bit. The value **n** is defined as a parameter of the encryption method. Input data for the encryption is usually called plain text [1] and is represented as data units **p[i]** where $\mathbf{0 \leq i \leq N_p - 1}$, and $\mathbf{N_p}$ defines the length of the plain text. Random initialization vector (IV) **r[i]** of $\mathbf{N_v}$ data units is to be prepended as securely invisible by encrypting prior to the plain text. The input data **d[i]** for the further encryption process is defined by

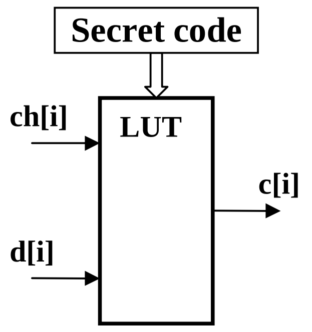
$$\mathbf{d[i] = r[i],\quad 0 \le i < N_v}$$
$$\mathbf{d[i] = p[i - N_v],\quad N_v \le i < N_p + N_v} \qquad (2.1)$$

Suggested encryption methods are defined as a stream-like sequential processing of input data units **d[i]** by a state machine (combinational function with a memory). This stateful object creates chaining and randomness of the result codes (ciphertext). Combinatorial conversion is to be implemented in a fast look-up table (LUT). Memory consumption for the LUT decreases by means of decomposition of the LUT into several sequentially connected basic encryption blocks. Each of them is a two-input LUT function **L** with input data **d[i]** and chaining **ch[i]**, and an output encrypted result code **c[i]** with the **n** bit width of each data unit:

$$\mathbf{c[i] = L(ch[i], d[i])} \qquad (2.2)$$

The content of the LUT is defined by the shared secret code. The full two-input LUT is of $\mathbf{u^2}$ entries, where

$$\mathbf{u = 2^n} \qquad (2.3)$$



Size of data unit is **n** bits

LUT: c[i] = f (ch[i], d[i])

ch[i]	d[i]	c[i]
0	0	x
0	1	y
0	$2^n - 1$	z
i	0	z
i	1	y
i	$2^n - 1$	x
$2^n - 1$	0	x
$2^n - 1$	1	z
$2^n - 1$	$2^n - 1$	y

Figure 1. Dual entry LUT-encryption block with chaining

Figure 1 shows the basic LUT-encryption block and the content of the look-up table. The LUT may be represented as **u** partial look-up tables with **u** entries per value of chaining **ch[i]**. Each partial LUT may be calculated by the initialization procedure of permutations of values **[0,…,u - 1]**. The pseudo-random values for permutations are to be defined by the shared key. The length of pseudo-random sequence for LUT generation is $\mathbf{2^{2n}}$.

Decryption at the level of a single basic encryption block is fulfilled in a similar way by using the decryption look-up table (dLUT).

The basic decryption block uses similar logic with **ch[i]**, **c[i]** as inputs and **d[i]** as the output. The dLUT is calculated straightforward from the encryption LUT.

Actually, each couple **(ch[i], d[i])** matches the unique code **c[i]**, while each couple **(ch[i], c[i])** matches the value **d[i]** in the same way. This is a basic requirement for the valid decryption. Multiple couples **(ch[i], d[i])** with the same **d[i]** may match the same code **c[i]**.

Main properties of the basic encryption block are elaborated by combinatorial analysis of the LUT.

Theorem 2.1. There is $(\mathbf{u}!)^{\mathbf{u}}$ different available combinations for the encryption look-up table.

Proof. The unique decryption is available if and only if all the values of **c** in the partial LUTs are different. Therefore, any permutation of all the **u** values of data unit, i.e. $\mathbf{[0,\ldots,2^n - 1]}$, is a valid set for **c** in a partial LUT. The number of possible permutations is $(\mathbf{u}!)$. Then, there are **u** partial LUT, all independent one of another. Any valid values of sets of the partial LUTs are admissible. Hence, the number of possible sets for the whole LUT of **u** partial LUTs is $(\mathbf{u}!)^{\mathbf{u}}$, which completes the proof.

Besides that, the LUT may be represented as a square matrix with entries **c** were rows and columns are values of **d** and **ch**. There is a special case of restricted subset with matrixes that have each of the values **c** only once in any row and in any column. This is the well known case of Latin Squares for the LUT that may be useful in specific applications.

The basic encryption block is an element of the methods described below rather than a stand alone encryption primitive. Its strength against brute force attacks is estimated as the number of possible encryptions, i.e. valid LUTs, assigned in Theorem 2.1.

The size **n** of the data unit characterizes the strength of encryption against brute forth attacks. On the other hand, it defines the size of LUT that should be available for in the memory. Some trade-off should be done here to get a brute force attack practically impossible and keep a price of solution in available range. For the data units between 4 and 10 bit the size of the LUT is between 128B and 1.5MB, while the number of combinations for the brute force attack is incredibly huge, between 2^{708} and $2^{8,979,454}$, to match needs of different applications.

3 Deep chaining and scheme of encryption methods

Basic encryption block is a core element for decomposition of the combinatorial function of the encryption state machine. Several basic encryption blocks are connected in the encryption scheme. A simplest scheme that provides invisible chaining **ch** and the IV may be represented by the following equations:

$$\mathbf{ch[i] = L(ch[i - 1], d[i])}$$
$$\mathbf{c[i] = L(ch[i - 3], ch[i])} \qquad (3.1)$$

The fixed values derived from the shared key are used for the chaining with negative indices of the first input data units.

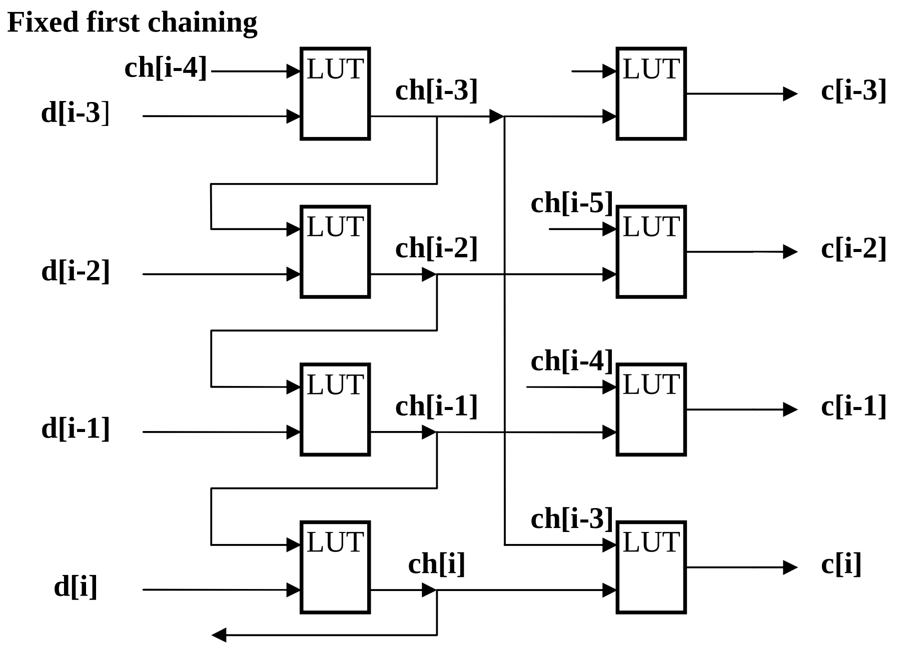


Figure 2. Encryption with two basic elements per data unit

Figure 2 shows the scheme by (3.1). This scheme uses two sequentially connected basic encryption blocks and represents a state machine with 3 memory data units for 3 chaining values. Decryption is done in the reverse way by the similar scheme using the decryption LUT (dLUT).

4 Security of sequentially connected encryption blocks

The goal of attack against suggested methods is a disclosure of the secret LUT by analysis of encrypted codes for a known encryption scheme. Randomness of the init vector as a first encrypted data yields the randomness of the encryption state machine, and then the result encrypted codes. Regarding the (2.1), any plain texts are encrypted into different codes. Variation of the random invisible IV generated within encryption tool cause changes in the state of the encryption state machine.

Considering the size of IV, the latter changes make impossible to distinguish further changes defined by chosen plain text used for attack. This approach makes impossible to apply the linear, differential, and boomerang attacks [6] against the suggested methods.

Potential weakness of the simple scheme (3.1) may be found in encryption of the long sequential data. Relatively small memory of 3 data units of this state machine may cause repeated sequences in the output encryption codes. Further schemes with a bigger internal state withstand this weakness. Usage of the method for a fixed maximum size of encrypted sequence is a reasonable restriction as well.

Cryptanalysis of the scheme (3.1) is important for estimation of parameters of the LUT, IV. Attacks against suggested methods should use a known and chosen plain text and specific features of the encryption scheme. In general, sequential processing defines two basic types of attacks: at the beginning and at the middle of the encrypted sequences.

4.1 Attack at the beginning of the encrypted sequence

Attack at the beginning of the encrypted sequence uses a fact that the chaining for the first encryption blocks is constant over the multiple encrypted sequences. Analysis of probability of a successful attack will estimate the encryption parameters particularly the needed size of IV.

Lemma 4.1. If one entry of the basic encryption block, for example **d**, is known, the second one, lets say **ch**, is invisible and **P(ch)** stands for the probability of guess of the value **ch**, then the random hit of the third entry **c** and the line of the LUT may be estimated as an independent probability

$$\mathbf{P(c) = P(ch) \cdot 2^{-n}} \qquad (4.1)$$

Proof. The LUT is created by pseudo-random permutations and all the values are expected with the same probability. Then the probability of the random hit of an output value **c** of **n** bit is $\mathbf{2^{-n}}$. The probability of the hit of two independent events is the product of their probabilities, and (4.1) follows.

It is worth emphasizing that for the certain value **d[i]**, all the $\mathbf{2^{n}}$ values **c[i]** have equal probability $\mathbf{2^{-n}}$ if and only if for the Latin Square LUTs. Other LUTs cause repeatability of some values. However, the probability of the random guessing of **c[i]** for any distribution is $\mathbf{2^{-n}}$.

Theorem 4.2. If the scheme (3.1) has an init vector as a random value of $\mathbf{N_v}$ data units then the maximum probability of the guess the chaining values and first lines of the LUT in attack at the beginning of encrypted sequence, is for the first data unit of the plain text **p[0]** and it equals

$$\mathbf{P_1 = 2^{-n(2Nv+4)}} \qquad (4.2)$$

Proof. Known information is the plain text **p** and the encrypted codes **c**. The IV and the fixed initial values of chaining are invisible. Therefore, any way of attack may use only relation between known **d[i]** and **c[i]** for some $\mathbf{i \geq N_v}$. Concerning to lemma 4.1 the probability of the hit of any internal chaining in the scheme (3.1) is a product of the probabilities of the previous chaining. So, the probability of successful hit is bigger for smaller **i**. Therefore, the maximum probability may be achieved for the first data unit of the plain text **p[0]** or **d[i]** for $\mathbf{i = N_v}$. The probability of the guess of each independent unknown chaining and the IV is $\mathbf{2^{-n}}$. Then the probability of discovering the first lines of the LUT by guessing all the previous values of chaining and the IV is a product of probabilities for all the following elements, and (4.2) follows:

$$\mathbf{P_1 = 2^{-n(Nv+1)} * 2^{-3n} * 2^{-nNv}} \qquad (4.3)$$

Theorem (4.2) estimates the probability of successful guess in a single test of encrypted method. The way of distinguishing the correct invisible internal values from any other on-going random is an unknown hypothetic means. Disregarding the latter existence and probability the minimum number of tests may be estimated using (4.2) as following.

The random IV causes the independence of different tests. Therefore, if $\mathbf{P_1}$ stays for probability of the hit of supposed values in a single test, then probability $\mathbf{P_T}$ of the hit even once in **T** tests equals

$$\mathbf{P_T = 1 - (1 - P_1)^T}$$

The number of needed tests **T** for desired probability $\mathbf{P_T}$ of the hit is

$$\mathbf{T = \log(1 - P_T) / \log(1 - P_1)} \qquad (4.4)$$

Table 1 shows the number of combinations (tests) that should be checked in order to succeed an attack with probability 0.1 and 0.9 and for using a hypothetic detecting facility of the certain values within the state machine (the real implementation should make this option impossible even with significant probability!). Estimation below refers to different sizes of data unit.

Table 1. Estimation of hypothetic attack at the beginning of encrypted sequence with IV of 8 byte

n, bit	Size of LUT (encryption and decryption)	Number of combinations to analyze for attack with probability of success	
		probability 0.1	probability 0.9
4	256 B	2^{140}	2^{145}
6	8 KB	2^{148}	2^{153}
8	128 KB	2^{156}	2^{161}
10	2 MB	2^{164}	2^{169}

Thus, the scheme (3.1) with all the intermediate chaining values invisible and random init vector causes huge number of combinations for attack at the beginning of encrypted sequence and makes this attack fruitless.

4.2 Attack at the middle of the encrypted sequence

Analysis of encrypted known and chosen plain text from any middle point is a more efficient attack. Multiple fragments of the same sequence may be used rather than a single fragment at the beginning.

As it is defined by the scheme (3.1) for each data unit the only previous values are **ch[i-1]** and **ch[i-3]**. There are only $\mathbf{2^{2n}}$ combinations for the intermediate chaining values for any data unit. The goal of attacker is to guess the **ch[i]**, **ch[i-3]**, and certain lines of the LUT, and, therefore, compromise this encryption. Concerning the lemma (4.1) the probability of such a hit is

$$\mathbf{P_1 = 2^{-3n}} \tag{4.5}$$

This hit may define the first two lines of the LUT.

Though, the simple guess of internal chaining in the middle is not viable because all internal values are not visible. There is no way to consider some suspected values of chaining match to a single visible pair **(d[i], c[i])**. The estimation (4.5) defines a probability that certain invisible values will be in a single test for any pair of an input and output **(d, c)**.

The only way for success of such attack may be a collection of multiple fragments with certain pairs **(d, c)** from encrypted data with chosen plaintext. Further analysis of duplication cases in some neighbour data units by hypothetical method (if any may be designed) may make assumptions about intermediate chaining values and LUT.

For any middle fragment of **f** data units the probability of the hit of intermediate chaining values may be estimated regarding the lemma (4.1) and the scheme (3.1):

$$\mathbf{P_f = 2^{-n(3+2(f-1))}, \quad 0<f<4,} \tag{4.6}$$
$$\mathbf{P_f = 2^{-n(f+4)}, \quad f>3}$$

Attack is more efficient for **f>3** while all the intermediate chaining values are in the set of the hit values. On the other hand bigger values of **f** define lower probability of the hit. Let's estimate condition and success probability of attack for fragments of **f=4** pairs **(d, c)**.

From the (4.6) a single test for a fragment of **f=4** pairs **(d, c)** contains supposed values of internal chaining values with probability

$$\mathbf{P_f = 2^{-8n}} \tag{4.7}$$

The pseudo-random nature of encryption causes occasional result of encrypted codes and desired set of pairs **(d, c)** for the fragment may be received with probability $\mathbf{P_{out}}$

$$\mathbf{P_{out} = 2^{-fn}} \qquad (4.8)$$

These events are independent and the probability $\mathbf{P_h}$ of the hit for a single test case is a product of probabilities:

$$\mathbf{P_h = P_f * P_{out}} \qquad (4.9)$$

For **f=4** the it equals to

$$\mathbf{P_h = 2^{-12n}} \qquad (4.10)$$

Thus, the number of the needed fragments for the hit with a desired probability for a hypothetic analysis may be estimated by (4.4). This estimation defines a minimal condition for attack at the middle with a chosen plain text. For example, the worst case is an illegal access to the encryption tool with a possibility to encrypt desired packets. Then use repeated fragment of 4 chosen data units. At than time it needed more than 2^{44} fragments in order to get desired set even once with a probability of 0.1. Such amount of encrypted data is about a lifetime of the device or any long-time shared secret. Besides that this estimation is done for a hypothetical attack. Any real method of analysis should significantly increase the number of needed tests.

The estimations above and possibilities of repeated values of encrypted data show that this scheme theoretically may be vulnerable and its strengthening is a challenge for the further development. This scheme may be regarded as an intermediate research method or a solution for certain specific application of encryption, for example, for multimedia random unpredictable data with limited size of the encrypted data.

5 Internal memory and built-in pseudo-random generator

Addition of internal memory increases the number of states of the encryption state machine. This is the major solution for the higher security and prevention of cycles in the long encrypted sequences. Additional internal memory is updated with processed data. So, it creates a built-in pseudo-random generator [1]. Figure 3 shows three versions of enhanced schemes with additional external memory.

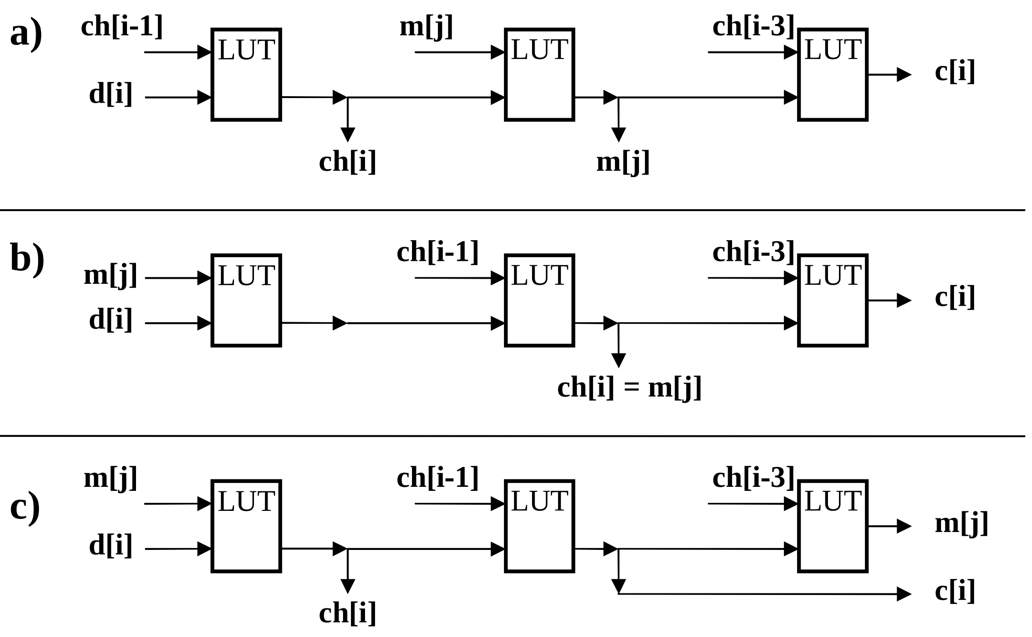


Figure 3. Three examples of encryptions with internal memory

The enhanced scheme on the Figure 3c may be defined by the following equations:

ch[i] = L(m[j], d[i]);
c[i] = L(ch[i-1], ch[i]);
m[j] = L(ch[i-3[, c[i]); (5.1)
j = (j + 1) mod M;

- **i** defines the index of the input data element;
- **j** defines the index of the data unit **m[j]** of the internal memory;
- **M** defines the number of data units in the internal memory.

Initial value of internal memory may be a part of the generating sequence for calculated LUT. Another option is to use it as a second shared secret key for encryption. This gives a new option for dual secret cipher.

Described above methods of encryption use sequentially connected LUTs. The strength of the whole encryption depends on the number of possible values in these LUTs. Usage of several different LUT initialized by different pseudo-random sequences increases the number of possible combinations for all types of attacks in power of the number of LUTs. The amount of needed memory increases linearly only in **k** times. Several sets of internal memory may be used in the same way.

In generic case there is **k** LUTs and **q** memory sets (sequences)**.** Each time certain elements are chosen by indexes, for example, defined by certain bits of on-going data unit of the memory sequences **m[j]**.

6 Analysis of enhanced schemes of encryption

The methods of analysis of possible attacks against enhanced schemes are similar to described above. The most important is the strength against attack at the middle of encrypted sequence.

The probability of the hit of the certain internal state (**m[j]**, **ch[i-1]**, **ch[i]**, **ch[i-3]**, and the new value ***m*[j]** for the analysed pair (**d[i]**, **c[i]**) is $\mathbf{2^{-5n}}$. For the generic case of **k** LUTs the hit of the correct set of the three randomly defined LUTs for (5.1) is estimated by probability $\mathbf{k^{-3}}$. Result probability of the hit of the full internal state for the certain known pair (**d**, **c**) is

$$\mathbf{P_1 = (k^{-3}) * (2^{-5n})} \tag{6.1}$$

Probability of the hit for the fragment of **f** data units may be estimated similar to the (4.6) concerning to additional invisible values of a previous and a new memory states:

$$\mathbf{P_f = (k^{-3}) * 2^{-n(5+4(f-1))}\ ,\quad 0<f<4,} \tag{6.2}$$
$$\mathbf{P_f = (k^{-3}) * 2^{-n(3f+4)}\ ,\quad f>3}$$

For the similar fragment of **f=4** data units

$$\mathbf{P_f = (k^{-3}) * 2^{-16n}} \tag{6.3}$$

Following to (4.9) and estimations for the fragment **f=4** and **k=4** LUTs, the value of probability of the hit in a single test may be calculated as

$$\mathbf{P_h = 2^{-(20n+6)}} \tag{6.4}$$

Then, the number of the needed fragments for the hit with a desired probability for a hypothetic analysis may be estimated by (4.4).

Table 2 shows the comparison of estimations for the basic scheme by (3.1) with the enhanced one by (5.1). The number of combinations **T** is defined for probabilities of success of attack for different sizes of data unit **n**. These numbers refer to the hypothetic ideal method to distinguish the certain line of the LUT.

Table 2. Estimation of hypothetic attack at the middle of encrypted sequence: basic scheme (3.1) / enhanced scheme (5.1).

n, bit	Size of LUT (encryption and decryption)	Number of combinations to analyze for attack with probability of success:	
		probability 0.1	probability 0.9
4	256 B / 1 KB	2^{44} / 2^{70}	2^{49} / 2^{75}
6	8 / 32 KB	2^{68} / 2^{110}	2^{73} / 2^{115}
8	128 / 512 KB	2^{92} / 2^{150}	2^{97} / 2^{155}
10	2 / 8 MB	2^{116} / 2^{190}	2^{121} / 2^{195}

7 Provable statement of security for the LUT encryption

The main problem of most encryption methods is the threat to discover a method of attack that compromises the shared keys and causes the loss of security in a certain application. Heuristic complex algorithms of encryption cause difficulties to prove that the method is reliable and there are no flaws in the logic.

Analysis of generic types of attack as described above evaluates parameters for encryption with a desired level of security. So, the LUT encryption brings the option of provable state for security applications.

8 Features of the LUT encryption and modes of operation

Described methods may be efficiently used without padding as it is done in block ciphers. The last part of the plaintext that is less than the data unit may be XOR-ed with encrypted fixed code that is a part of the shared secret. The result bits are secure at the same level as encryption of the known plain text.

Described methods implement calculating of the hash values for check data integrity by recurring encryption of the data with continuous chaining values. Additional header may be authenticated with minimum additional operations.

Continuous encryption generates pseudo-random values that may be used for the init vectors and various elements of security protocols.

Suggested approach may be used to create a classic stream cipher based of the XOR between the plaintext and the result of a cyclical encryption of the shared secure sequence. The IV is encrypted prior to the secure sequence and the result is sent to the receiver. The latter provides synchronization between sender and receiver and prevents all types of known plain text attack.

Special mode of operation may be used for implementation length-preserving encryption when the length of the encrypted codes is the same as an input plain text without any addition of the initialization vector. Equal plain texts are encrypted into the same ciphertext. This mode may be implemented by two sequential encryptions with different shared secret values of the chaining, while the second encryption does processing in reverse direction (from the end to the beginning). Each of the data unit of result ciphertext depends on the whole plain text.

Described methods use minimum simplest operations. Processing of each data unit is as simple as 2-3 accesses to look-up tables and save intermediate chaining in temporary memory. It may be efficiently implemented in hardware and software for any processor.

9 Conclusions

Described approach defines a new option for shared key encryption. Suggested methods are intermediate between block and stream ciphers and use conversion of input data by a state machine. Main elements are as simple as two-entry look-up tables sequentially connected by the certain scheme of encryption. The result features are speed of a stream cipher and direct mixing of bits of several data units like a block cipher. Built-in chaining with on-going pseudo-random generator calculates the hash for the message authentication during encryption.

Suggested algorithms may be implemented as in software as in hardware for different data unit sizes of 4-10 bits that provides different levels of security and memory consumptions.

The work describes new methods of cryptanalysis and their application to the encryption methods. Suggested analysis estimates the probability of success for a worst case of hypothetic attack against encryption methods. All together these results bring an opportunity of design actual provable security methods.

References

1. A. Menezes, P. Van Oorschot, and S. Vanstone. Handbook of applied cryptography. The CRC Press series on discrete mathematics and its applications. CRC-Press, 1997.
2. A.Biryukov, "Block Ciphers and Stream Ciphers. The state of the Art", (survey), Lecture Notes in Computer Science, to appear in proceedings of the COSIC Summer course, 2003.
3. RFC 3686 (R. Housley), Using AES Counter Mode With IPsec ESP, 2004.
4. RFC 4106 (J. Viega, D. McGrew), The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP), 2005.
5. M. Dworkin, Recommendation for Block Cipher Modes of Operation. Methods and Techniques. NIST Special Publication 800-38A 2001 Edition.
6. P. Gunod, Statistical Cryptanalysis of Block Cipher , These N3179, Lausanne, EPFL 2005
7. D. Whiting, B. Schneier, S. Lucks, and F. Muller, Phelix: Fast Encryption and Authentication in a Single Cryptographic Primitive.