11-2007

Light-Weight Encryption Schemes for Multimedia Data

Feng BAO
Singapore Management University, fbao@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1109/GLOCOM.2007.43

Light-Weight Encryption Schemes for Multimedia Data and High-Speed Networks

Feng Bao
Cryptography and Security Department
Institute for Inforcomm Research
Singapore
baofeng@i2r.a-star.edu.sg

Robert H. Deng
School of Information Systems
Singapore Management University
Singapore
robertdeng@smu.edu.sg

Abstract—**Due to the pervasiveness of high-speed networks and multimedia communications and storage, the demand for high-speed cryptosystems is ever increasing. It is widely believed that there is a tradeoff between speed and security in cryptosystem design. No existing encryption algorithms are both fast enough for high-speed operation and sufficiently secure to withstand powerful cryptanalysis. In this paper, we propose and analyze a generic construction of high-speed encryption schemes. Our solution is based on the fact that there exist secure but relatively slow block ciphers, e. g. AES, and super-fast but relatively weaker stream ciphers. We then combine a secure block cipher with a super-fast stream cipher such that the resulting encryption scheme possesses both the speed of the stream cipher and the security of the block cipher. We show the results our security analysis as well as our experiment on a 2.1 GHz Pentium VI processor.**

Keywords- block ciphers; stream ciphers; high-speed encryption.

I. INTRODUCTION

A. The High Speed Requirments

Speed-hungry computer and communication applications are on the rise for a variety of services, such as multimedia electronic mail, video conferencing, and high-definition televisions. Several types of high-speed networks exist. One example of such networks is the Asynchronous Transfer Mode (ATM) technology which provides data rates from tens of Mb/s to Gb/s [1]. As new networks/multimedia services become more capable and user friendly, high-speed networks will continue to attract more traffic and more sensitive information (such as corporate propriety information). The content of such communication represents an increasingly lucrative target for eavesdropping and tampering. The need to protect multimedia data over high-speed network has been long recognized. For example, encryption of IP datagrams in the new Internet Protocol IPv6 is mandatory [2].

High-speed networks are expected to integrate a variety of multimedia applications with different traffic characteristics and quality of service (QoS) requirements [3]. Protection of the traffic, whether at the application level or at the network level, must not introduce degradation in QoS. This demands the availability of efficient encryption algorithms so that encryption speed does not become a performance bottleneck in bandwidth-hungry applications.

B. Speed of Existing Ciphers

Since we are concerned with traffic encryption, our focus in the present paper is on symmetric key ciphers. Such ciphers can be classified into block ciphers and stream ciphers.

Intensive study on cryptanalysis to block ciphers has been carried out during the last decade and several very powerful methods, such as differential attack [4], truncated differential attack [5], linear attack [6] and algebraic attack [7], have been developed. A block cipher is regarded as secure if it is able to withstand all these known attacks. Examples of such block ciphers are DES, AES, IDEA, SAFER, TWOFISH and CAST. Typical speeds of these ciphers are around 20 to 50 MB/s on a Pentium IV 2.1GHz processor.

Stream ciphers (see [8] and the references therein) are normally much faster than block ciphers. However, stream ciphers are perceived to be weaker in security than block ciphers. Many stream ciphers that used to be believed very secure were broken later. Even a carefully designed stream cipher might not withstand very careful and intensive attacks. An example is the stream cipher TWOPRIMES [9], which is proved to possess high linear complexity, large cycle length, good resistance to LSFR-synthesis attacks and many other desirable attributes of good stream ciphers; however, it was broken in with only a complexity of 2^{32} [10].

C. Existing Solutions to High Speed Encryption

One way to meet the high-speed requirement is to design a fast cipher without revealing its algorithm and to implement it by hardware. This approach is normally not considered as prudent practice in the security community. One thing is that this puts a harsh limit on the use of the cipher – the implementation of the algorithm must be carefully guarded in trusted environment or tamper resistant hardware. Another and probably more serious problem is that once the algorithm is revealed, its weakness may be found under various powerful attacks. We have seen such examples as SkipJack and CMEA (cellular message encryption algorithm) [11]. The secret encryption algorithms were found to have various weaknesses a short time after they were released to the cryptographic community.

We also notice that some companies have designed very fast stream ciphers (e.g., [12], [13]); they publicly announce that their ciphers are secure because no one are able to break their ciphers and claim the big price awards they put up. In [13], ten million Japanese yen is put as the award to the people who can break their chaos-based cipher. But all these encryption schemes have not received sufficient and careful cryptanalyses. In crypto community, it is well-known that an unanswered challenge to break a cipher does not mean anything about the security of the underlying cipher. Therefore, these fast encryption schemes cannot give sufficient confidence to their users.

In this paper we aim at designing super-fast and secure symmetric key encryption schemes. However, we do not propose new encryption algorithms. What we do is to combine existent strong algorithms (relatively slow) with fast algorithms to form new encryption schemes, which are fast enough to satisfy the high-speed requirement for various applications. Meanwhile, we can confirm the security of our schemes under certain reasonable assumptions.

II. Outline of Our Schemes

As we mentioned in the previous section, symmetric key ciphers can be roughly classified into stream ciphers and block ciphers. Block ciphers usually have stronger security than stream ciphers. Here we refer to the traditional stream cipher based on the LFSR. Block-cipher-derived stream ciphers, such as RC4, have similar security as the underlying block cipher.

A block cipher is typically an iterated function on a fixed block size. The security mainly comes from the iteration. For a block cipher, it is impossible to express the output and the input in an explicit algebraic formula. This is because the iteration function makes the formula expand into an overwhelmingly large size. Block ciphers are usually designed to resist various kinds of attacks, including linear attack and differential attack and so on. Most of the attacks to block cipher are aimed at deriving the key. *So a secure block cipher should be able to keep the key forever secret, even under attacks where attackers can arbitrarily choose plaintext/ ciphertext pairs as they want.* Although it is controversial whether chosen plaintext or ciphertext attacks are practical, any news of possible breaking may decrease people's faith in a cipher. Hence, resistance to chosen plaintext/ciphertext attacks are regarded as a fundamental requirement of block ciphers.

A stream cipher is usually a pseudo-random generator. The pseudo-random sequence (key stream) generated by the pseudo-random generator is XORed with the plaintext to obtain the ciphertext. The pseudo-random generator may or may not depend on the plaintext and ciphertext.

Our generic construction of fast encryption schemes are based on the following observation: we combine a secure block cipher with a fast stream cipher such that the encryption scheme can have both the security of the block cipher and the speed of the stream cipher. The secret key of our encryption schemes is protected by the block cipher while the large plaintext is encrypted by the stream cipher with segment keys generated from the block cipher. The principle of this kind of combination has been adopted in the physical world for a long time in order to design strong but light-weight objects, for example wooden boats with steel framework.

Let **BE**(K, m) denote a block cipher encryption algorithm (such as AES) on message m using key K and **SE**(k, M) denote a stream cipher encryption algorithm on message M using key k. Here m has fixed size, i.e., the block size of **BE** while M has arbitrary size. We divided a plaintext into segments with equal size (padding may be applied to the last segment):

$$plaintext = pseg_1, pseg_2, pseg_3, \ldots, pseg_t$$

The segment size is the number of bits of seg_1. Let K be the secret key of the scheme, which is a key of **BE**. The encryption is performed as follows.

First we randomly choose a number r of the block size of **BE**, then generate the segment keys as k_1 = **BE**(K, r), k_2 = **BE**(K, k_1), …, k_t = **BE**(K, k_{t-1}). The corresponding ciphertext is given by

$$r, cseg_1, cseg_2, cseg_3, \ldots, cseg_t$$

where

$$cseg_i = \mathbf{SE}(k_i, pseg_i).$$

Note that r precedes the ciphertext so that decryption can be carried out at the receiver. We require that r never be reused. In our schemes, r is set to be 128-bit; therefore, the probability that two randomly selected r's happen to be the same is as small as correctly guessing the key K. (Note. The r can also be generated from the plaintext, say, let r = **BE**(the first 128 bits of the plaintext, K). There are actually many ways to generate r.).

It is easy to see that such an encryption scheme can protect the secret key K if the block cipher **BE** can. That is, a secure block cipher can guarantee that our scheme's secret key will not be discovered by an attacker no matter what cryptanalysis means he exploits. The stream cipher **SE** we use here can be allowed to be weaker in the security. *It is enough for our usage if SE can resist the attack of known plaintext/ciphertext pair of length over the segment size.* The segment size can depend on how strong the **SE** is. The segment size can be taken larger if the **SE** has better security. The segment size also affects the speed of our scheme. The larger the segment size, the closer our speed is to **SE**'s speed. The segment size in our schemes is usually taken from 2^{15} to 2^{18}. This is a very small number from the point of view of cryptanalysis. Therefore, the security principle of our schemes is based on the following observations: When we talk about a cipher, it may be strong under some attacks but weak under some more powerful attacks. The most powerful attacks are chosen plaintext/ciphertext attacks, which are towards deriving the secret key. However, the secret key of our scheme cannot be derived by chosen plaintext/ciphertext attacks due to using a secure **SE**. Chosen plaintext/ciphertext attacks make no sense in attacking segment keys since segment keys are never repeatedly used in our scheme.

III. FAST ENCRYPTION SCHEME VARIANT I

In this Section we deign a stream cipher that is about multiple times faster than a typical block cipher. When we take suitable segment size, the combined encryption scheme is almost as fast as the stream cipher. The experiment results will be shown in Section 6.

The notations used here are the same as in Section 2. Let **BE** be the secure block cipher and **SE** be the fast stream cipher. The plaintext is written as

$$\begin{matrix} p_1 & p_2 & \cdots & p_n \\ p_{n+1} & p_{n+2} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{tn+1} & p_{tn+2} & \cdots & p_{(t+1)n} \end{matrix}$$

where $p_{tm+2}, \cdots, p_{(t+1)m}$ may be empty. The first block p_1 is not in the original plaintext. It may be a random head appended to the original plaintext before the encryption. It may also be a number used as a counter to indicate the *i*-th encryption. It is fine as long as p_1 satisfies a) p_1 is different for different plaintext or b) p_1 is not required to be confidential. Assume that

$$\begin{matrix} p_1 & p_2 & \cdots & p_n \\ p_{n+1} & p_{n+2} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{tn+1} & p_{tn+2} & \cdots & p_{(t+1)n} \end{matrix}$$

is encrypted into

$$\begin{matrix} c_1 & c_2 & \cdots & c_n \\ c_{n+1} & c_{n+2} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{tn+1} & c_{tn+2} & \cdots & c_{(t+1)n} \end{matrix}$$

We have

1. $c_1 = p_1$

2. $c_2 c_3 \cdots c_n = \mathbf{SE}(\mathbf{BE}(K, p_1),\ p_2 p_3 \cdots p_n)$

3. $c_{in+1} c_{in+2} \cdots c_{(i+1)n} =$
$\mathbf{SE}(\mathbf{BE}(K, p_{in}),\ p_{in+1} p_{in+2} \cdots p_{(i+1)n})$ for $i = 1, 2, \ldots, t$.

The decryption is easy to see: just calculate $\mathbf{BE}(K, p_1)$ from $c_1 = p_1$, then decrypt $c_2 c_3 \cdots c_n$ to get $p_2 p_3 \cdots p_n$. Then from p_n to get the next segment key and decrypt $c_{n+1} c_{n+2} \cdots c_{2n}$, and so on. In this scheme, the segment size is $128|n|$.

A. Description of SE

Let the plaintext be

$$b_1 b_2 \cdots b_m \cdots$$

where each b_i is a 32-bit string. Let *F* be a function defined as

$$F(k, x) = ((((x + k_1) \oplus k_2) \times k_3) \oplus k_4) >>>$$

where *k* is the 128-bit key and $k = k_1 k_2 k_3 k_4$ for 32-bit k_i, x is a 32-bit string, $\oplus$ is the bit-wise exclusive-or, + and $\times$ are mod 2^{32} addition and multiplication, $>>>$ is to reverse the 32 bits into opposite ranking. The encryption of $b_1 b_2 \cdots b_m \cdots$ is

$$d_i = b_i \oplus F(k, F(k, F(k, d_{i-1}) \oplus b_{i-1}) \oplus d_{i-2})$$

where $d_1 d_2 \cdots d_m \cdots$ is the ciphertext. The d_{-1}, d_{-2}, d_{-3} can be set k_2, k_3, k_4.

B. Security Discusison of the Secheme

Security of the key: The key of the encryption/decryption is *K* that is protected by the **BE** block cipher. On the assumption of the security of **BE** (or say on the assumption of the security of AES), *K* will never be derived by any attacker no matter what kind of attacks are used.

Meet in the middle attack (the attack to the segment key): This is a kind of attack of brutal force. By meet one or more bits in the middle, exhaustively search the key bits relevant to these middle bits. Since we take 3 rounds of *F*, the meet in the middle attack does not work. This is because at least one way to the middle goes through two rounds of *F*, therefore, the number of key bits that affect a single bit is large: for two round of *F*, at least 96 bits of the key will effect a middle bit. This is enough to resist brute force searching.

Chosen ciphertext attack (the attack to the segment key): We know that all the stream ciphers that have ciphertext feedback are weak to the chosen ciphertext. For example, if our stream cipher was defined by

$$d_i = b_i \oplus F(k, F(k, F(k, d_{i-1}) \oplus d_{i-2}) \oplus d_{i-3})$$

Then the cipher would be weak to chosen ciphertext attack. By choosing $d_{i-1} = d'_{i-1}$, $d_{i-2} = d'_{i-2}$ and d_{i-3}, d'_{i-3} different at only one bit, the attacker can ask for the decryption of d_i, d'_i and apply the differential attack. But our stream cipher is defined by

$$d_i = b_i \oplus F(k, F(k, F(k, d_{i-1}) \oplus b_{i-1}) \oplus d_{i-2})$$

where the formula has both ciphertext and plaintext feedback. In such case, if the attacker choose both plaintext and ciphertext, the decrypted plaintext has very small chance to meet the plaintext chosen by the attacker. If the attacker tries to pick such meeting formats from the known plaintext/ciphertext (instead of chosen ciphertext), the number of known plaintext/ciphertext (block) pairs required is estimated 2^{48} (like the birthday attack to $2^{3\times 32}$). But our segment can never be so large.

IV. FAST ENCRYPTION SCHEME VARIANT II

The idea of combination of a fast cipher with a secure (but maybe slow) cipher can also be applied to two block ciphers. For example, Serpent has 32 round, which can resist the

cryptanalysis of over 2^{120} known plaintext-ciphertext pairs [14]. It is also estimated in [14] that a 6-round Serpent can resist cryptanalysis of over 2^{25} blocks known plaintext-ciphertext pairs(i.e., 2^{32} bits). Now we can use the 32-round Serpent to get the segment keys and use each segment key to encrypt *n* blocks by the 6-round Serpent. Here we take *n* to be much smaller than 2^{25}. In this case, the security of the scheme is guaranteed. The secret key is protected by the full round Serpent so that it can never be derived even in face of chosen plaintext/ciphertext attacks. The segment keys are also safe since each segment key is used to encrypt at most *n* blocks, while attacking the segment key requires 2^{25} blocks of known plaintext/ ciphertext pairs.

Of course, the key setup procedure may slow done the speed of the scheme. But if *n* is large enough, the speed of the scheme is about 32/6 times that of the original Serpent. In our implementation, we take *n* to be 2^8, 2^9, 2^{10} and 2^{11}. The speeds are listed in Section 6.

V. Fast Encryption Scheme Variant III

This fast encryption scheme consists of a very fast stream cipher and a block cipher as described in Section 2. The block cipher is used to generate segment keys and the stream cipher is used to generate key streams from segment keys. The block cipher we used is AES. The design of the stream cipher is given below.

This stream cipher is used to extend a 128-bit key into a key stream of segment size. Before we illustrate the detailed design of the stream cipher, we give the notations below:

&: bit-wise AND. ⊕: bit-wise XOR. >>> : right rotation. T : a table containing 32 elements, each element is with 32 bits. F : Feedback function. G: Output function. K : The 128-bit secret key. It consists of four 32-bit words: k_1, k_2, k_3 and k_4.

C: $C_i (0 \le i \le 31)$, each one is a 32-bit constant. They are generated from the constant e in the following way:

for $i = 0$ to 31 $\quad C_i = (e \times 2^{32(i+1)}) \,\&\, 0\text{xFFFFFFFF}$

r: $r_i (0 \le i \le 63)$, each one is between 3 and 14.

They are generated from the constant π in the following way:

for $i = 0$ to 63 $\quad r_i = ((\pi \times 2^{8(i+1)}) \,\&\, 0\text{xFF}) \bmod 12 + 3;$

The functions F and G are defined as follows:

Definition of F: The input of function F is the table T and two rotation constants r_1 and r_2. The output of function F is denoted as *feedback*. F operates in the following way:

$$tem = (((((T[0] \oplus T[22]) >>> r_1) + (T[10]) \oplus T[27]) >>> r_2) + T[15]$$

$$feedback = tem \oplus T[tem \,\&\, 31]$$

Definition of G*: The input of function G is the table* T. *The output of function of G is denoted as* output *output. And G operates in the following way:*

$$tem = (((T[29] \oplus T[23]) >>> r_1) + (T[20]) \oplus T[13]) >>> r_2) + T[2]$$

$$output = tem \oplus T[tem \,\&\, 31]$$

The operation of this stream cipher consists of two stages: an initial setup stage and an output stage.

Initial Setup

1. Initialize the table T
   for $i = 0$ to 31 $\quad T[i] = C_i + k_{i \bmod 4}$
2. Run the main algorithm (given below in this section) for 64 cycles and prepare for the output.
3.

The Main Algorithm

For the ith cycle, the cipher operates in the following way:

1. Run the F function with $r_{2i \bmod 32}$ and $r_{2i+1 \bmod 32}$, obtain the value of *feedback*.
2. for $j = 0$ to 30 $\quad T[j] = T[j+1];$ $\quad T[31] = feedback$
3. Run the function G and generate the *output*.

The Security

First, as we addressed previously, the secret K of the encryption scheme is protected by AES. Therefore, to attack the key is as hard as attack AES. Second, each segment key generated by AES is used to encrypt message of a very limited length by the stream cipher.

To resist known plaintext/ciphertext attack: Since in this stream cipher the key stream is generated independent of input/output, the known plaintext/ciphertext attack is just like to know the key stream of the same length. In the stream cipher, both the linear operation and non-linear operation are well combined. The relationships among the output words are extremely complicated. Although the *G* function is relative simple comparing to the traditional shift register-based ciphers, finding the relationship between the output blocks becomes more difficult due to the introduction of *F*. At this stage, we believe that the stream cipher cannot be broken within reasonable time for given key stream of short length, say 2^{30} bits. We think this is a conservative estimation. Anyhow, the security of this stream cipher is being investigated.

VI. Fast Encryption Scheme Variant III

We implemented the variants of our scheme described in this paper on a 2.1GHz Pentium-IV processor. AES is the block cipher we use for all these variants. The encryption speed of AES on our platform is about 50 MB/s. The average encryption speed for the variants of our scheme is given in the following tables based on 4 tests for each variant.

TABLE I. RESULT OF VARIANT I ENCRYPTION SPEED TEST

Data Size Mb	Seg. Size bits	Test 1 MB/s	Test 2 MB/s	Test 3 MB/s	Test 4 MB/s	AVG MB/s
5,242	32,768	297	296	297	297	296
5,242	65,536	304	302	304	304	303
5,242	131,072	307	307	307	308	307
5,242	262,144	309	309	309	309	309

TABLE II. RESULT OF VARIANT II ENCRYPTION SPEED TEST (8-ROUND SERPENT)

Data Size Mb	Seg Size bits	Test 1 MB/s	Test 2 MB/s	Test 3 MB/s	Test 4 MB/s	AVG MB/s
1,048	32,768	116	116	116	116	116
1,048	65,536	118	119	118	118	118
1,048	131,072	119	120	119	119	120
1,048	262,144	120	120	120	120	120

TABLE III. RESULT OF VARIANT III ENCRYPTION SPEED TEST

Data Size Mb	Seg Size bits	Test 1 MB/s	Test 2 MB/s	Test 3 MB/s	Test 4 MB/s	AVG MB/s
24,903	38,912	1338	1338	1342	1343	1340
24,903	79,824	1465	1470	1470	1465	1467
24,903	159,648	1543	1537	1543	1543	1542
24,903	319,296	1577	1584	1577	1584	1580

VII. CONCLUDING REMARKS

In this paper we proposed a general principle to combine a strong but slow cipher with a weak but fast cipher such that the combined encryption scheme is as strong as the former and as fast as the latter. We also gave some concrete variant schemes under this principle.

Our design is based on the observation on the available attacks to symmetric key ciphers. When we talk about a cipher, it may be weak under some powerful attacks, i.e., the chosen plaintext/ciphertext attacks, which are targeting to drive the secret key. Such attacks make no sense towards deriving segment keys. Therefore, many "weak" ciphers become "strong" when being used under our principle.

Although our schemes are very fast, we would like to point out that the encryption schemes constructed under our principle have speed advantage only when they are used to encrypt large amount of data.

REFERENCES

[1] M. de Prycker, Asynchronous Transfer Mode: Solution for BroadcastISDN, Ellis Horwood, New York, 2nd edition, 1993.

[2] C. Huitema, Ipv6: The New Internet Protocol, Prentice Hall PTR, Upper Saddle River, NJ, 1996.

[3] E. Gelenbe, X. Mang, and R. Onvural, "Bandwidth allocation and call adminission control in high-speed networks", IEEE Communications Magazine, pp. 122-129, May 1997.

[4] E. Biham and A. Shamir, "Differential cryptanalysi of DES-like cryptosystems", Journal of Cryptology, Vol. 4, No. 1, pp. 3-72, 1991.

[5] L. R. Knudsen, "Truncated and higher order differentials", In B. Preneel, editor, Fast Software Encryption, LNCS 1008, pages 196-211, Springer Verlag, 1995.

[6] M. Matsui, "Linear cryptanalysis method for DES cipher", Proceedings of Eurocrypt'93, LNCS 765, Springer-Verlag, pp. 386-397, 1994.

[7] N. T. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology - Asiacrypt'02*, LNCS 2501, pages 267–287. Springer-Verlag, 2002.

[8] R. A. Ruepple, Analysis and Design of Stream Ciphers, Springer-Verlag, 1986.

[9] C. Ding, V. Niemi, A. Renvall and A. Salomaa, "TWOPRIME: A fast stream ciphering algorithm", Proceedings of FSE'97, LNCS 1267, Springer-Verlag, pp. 88-102, 1997.

[10] D. Coppersmith, D. Wagner, B. Schneier and J. Kelsey, "Cryptanalysis of TWOPRIMES", Proceedings of FSE'98, LNCS, Springer-Verlag, 1998

[11] D. Wagner, B. Schneier and J. Kelsey, "Cryptanalysis of cellular message encryption algorithm", Proceedings of Crypto'97, LNCS, Springer-Verlag, 1997.

[12] http://www.nyber.com.

[13] http://www.iisi.co.jp/index_e.html.

[14] E. Biham, R. Anderson and L. Knudsen, "Serpent: a proposal for the advanced encryption standard", ttp://www.cl.cam.ac.uk/~rja14/serpent.html.