A General Encryption Scheme Based on MDS Code (Extended Summary)

Lihao Xu
Department of Computer Science and Engineering
Washington University in St. Louis
lihao@cs.wustl.edu

1 Introduction

Many applications, such as wireless communications, high speed multimedia data streaming systems and sensor networks, call for strong ciphers with low computation complexity and high speed in encryption and decryption. Additive stream ciphers usually have lower computation complexity and thus provide higher encryption/decryption speed than block ciphers. An additive stream cipher usually encrypts a *plaintext* stream simply by adding it to a *key stream*, where the addition ($\oplus$) is simple bitwise XOR (exclusive OR). Additive stream ciphers though have a main disadvantage that a known plaintext-ciphertext stream pair reveals the corresponding key stream used in encryption. The encryption key used to generate the key stream can then be deduced from a long enough key stream. Even two ciphertext streams encrypted using a same key stream can be exploited to obtain the key stream itself[3]. In fact, such a disadvantage of stream ciphers has been exploited to show the insecurity of the *WEP* (Wired Equivalent Privacy) protocol used in IEEE 802.11 standard for wireless networks [2]. Block ciphers, on the other hand, can conceal statistical properties of a plaintext more easily in the corresponding ciphertext. To keep the merits of both stream cipher and block cipher but lessen their disadvantages, it has been proposed to combining stream ciphers and block ciphers together to achieve stronger security [4]. Such combining, however, does *not* lessen computation complexity of encryption and decryption. In fact, computation complexity is even higher than that of the component block ciphers and stream ciphers. In this paper, we propose to combine cryptographically strong random key stream generators (including those used in stream ciphers) with simple block encoders to achieve both high cryptographic strength and low computation complexity of encryption and decryption. In particular, we show proper error correcting codes can be used as block encoders. In general, encoding and decoding of a proper block error (or more accurately erasure) correcting code is much faster than encryption and decryption of block ciphers.

2 Basic Block Encryption using MDS Code

MDS (Maximum Distance Separable) codes are a class of block error control codes that meet the Singleton Bound, i.e., $d = n - k + 1$ for an (n, k, d) code over GF(q^l) [6]. (Here we limit our discussion to linear codes, since most practical codes are linear.) For simplicity, $q = 2$ hereafter. Encoded by an (n, k) MDS code over

GF(2^l), a k-symbol message block $m = m_1 \cdots m_k$ is expanded to an n-symbol codeword block $w = w_1 \cdots w_n$. The message block m can be perfectly recovered from *any* k symbols of the codeword w using a proper *erasure* decoding algorithm. However, the erasure decoding is possible only when both the values and the *positions* (indices) of the k codeword symbols are known. Given the values of k symbols of a codeword but without their indices in the codeword, there are $P(n,k) = n(n-1)\cdots(n-k+1)$ possible permutations of the k indices in total, and thus an erasure decoder can produce $P(n,k)$ possible outputs, only one of which is the original message. Based on this fact, an MDS encoder which takes in a k-symbol message block and produces k codeword symbols without revealing their indices is effectively a block cipher, where the correspondences are clear: the k-symbol message block m is the plaintext, the k-symbol codeword block w is the ciphertext and the corresponding k indices collectively serve as the encryption key. With the encryption key, decryption is simply an erasure decoding.

This block cipher is indeed a Hill Cipher which performs a linear transform on a plaintext block: $c = mT$, where m and c are k-dimensional vectors and T is a $k \times k$ invertible matrix over GF(2^l) [5]. T functions as the encryption key, and it is a submatrix of a *generator matrix* of the MDS code. The Hill Cipher proves to be secure against ciphertext-only attacks for 1-block messages. (It of course falls under known plaintext-ciphertext attacks.) For a 1-block message, the only possible attack against this block cipher seems to be *brute force* attack with a complexity of $P(n,k)$, which is usually computationally impractical. For example, if a $(2^{16}-1, 8)$ Reed-Solomon (RS) code is used to encrypt a 128-bit message block, in worse case the brute force attack needs $P(2^{16}-1, 8) \approx 2^{127.9}$ trials.

In practice, however, the Hill Cipher is unusable for a message with multiple blocks, since the same transform matrix T is used for encrypting multiple blocks, and T can be deduced from the redundancy in the multiple message blocks. One possible option to make Hill Cipher secure for a multi-block message is that each block uses a different transform matrix T. But that is infeasible for the Hill Cipher in practice since the key size will be no less than the message size. On the other hand, a block cipher based on an MDS code is possible since each block can use a different submatrix of the generator matrix, which is publicly known. The problem boils down to which submatrix to choose to encode (encrypt) a specific message block. One solution is to use a cryptographically strong random stream generator to choose codeword indices, i.e., the transform matrix, for encoding each block. This effectively ensures that different message blocks are not encoded using a same transform matrix in a deterministic fashion. The random codeword index stream can be controlled by the message encryption key. Thus it is feasible to use an MDS code to construct a block cipher, combining cryptographic random stream generators. Of course any good key stream generator for a stream cipher can be readily used in this general block encryption scheme.

3 A General Encryption Scheme

With the basic ideas described in the previous section, we now describe a general *block* encryption scheme based on MDS code.

A message is segmented into m-bit blocks. Each block in turn consists of k symbols (sub-blocks), each of which has l bits with m being a multiple of l and $k = \frac{m}{l}$. Choose an (n,k) MDS code over GF(2^l), where $n \gg k$. Let $E(\cdot)$ be the encoding function of the MDS code, and $w = E(p)$ is a codeword of a *plaintext* message

block p. The codeword w now has n l-bit symbols: $w = w_1 w_2 \cdots w_n$, where w_i ($1 \leq i \leq n$) is an l-bit symbol. Let $A = a_1 a_2 \cdots a_k$ be a sequence of k *distinct* integers between 1 and n, i.e., $1 \leq a_i \leq n$ and $a_i \neq a_j$ when $i \neq j$. Such a sequence is called a *k-sequence*. For a given k-sequence A and a plaintext message block p, let c be a *ciphertext* block with k symbols from the codeword w: $c = w_{a_1} w_{a_2} \cdots w_{a_k}$. We call the mapping from p to c a block encryption operation: $c = Enc(p, A)$, where the sequence A is the encryption key. Note the order of the integers of a k-sequence is as important as their values. Two sets of integers with the same elements but with different orders are two different k-sequences, since they produce two different ciphertext blocks for the same plaintext block. Given c and A, the message block can be readily decrypted by an erasure decoding of the MDS code: $p = Dec(c, A) = D(c, A)$, where $D(\cdot)$ is an erasure decoder of the MDS code.

Let R be a random number generator which generates a sequence of k-sequences, with a seed K. A general block scheme for encrypting multi-block message can be described as follows:

A General Encryption Scheme:
Input: a b-block plaintext message $P = p_1 p_2 \cdots p_b$ and an encryption key K;
Output: a b-block ciphertext $C = c_1 c_2 \cdots c_b$;
Encryption Procedure:

1. A random number generator R generates b k-sequences $A_1 A_2 \cdots A_b$ with K being the seed;
2. For $i = 1, 2, \cdots, b$: $c_i = Enc(F(p_i, A_i), G(A_i))$.

The corresponding decryption procedure is straightforward. In the encryption procedure, Step 1 can readily include other information in the seed, such as an *initial vector* IV to prevent *replay* attacks. In Step 2, $F(\cdot)$ can be a simple function which is invertible for p when A is given, e.g., $F(p, A) = p \oplus A$, and $G(\cdot)$ is a simple *one-way* function that meets the one-way property: it is computationally hard to compute x from $G(x)$.

The security strength apparently depends on the strength of the random number generator R. Any cryptographically good (pseudo)random generators can be used here, e.g., strong hash functions (e.g., MD5 or SHA-1) or key stream generators of stream ciphers (e.g., RC4). Since the random key stream is not directly XORed with a plaintext stream, currently known attacks against additive stream ciphers, such as *correlation attack* [7, Ch.6], are more difficult to launch. If the one-way function $G(\cdot)$ in Step 2 also directly or indirectly takes in certain plaintext message blocks as input, then the same encryption key generates two *different* index streams for two different plaintext messages, which are encoded to two different ciphertext streams. Thus ciphertext of XOR sum of two plaintext messages is *not* equal to the XOR sum of the two corresponding ciphertexts, and XORing attacks [3] are much more difficult to launch too. This avoids the very weakness of additive stream ciphers for which a same encryption key cannot be used to encrypt different plaintexts. A ciphertext block is *not* a block from the direct codeword corresponding to the message block, instead it is a codeword block of a modified message block effectively encrypted by a stream cipher $F(\cdot)$. Also the index stream for the MDS encoder is isolated from the message block by the one-way function $G(\cdot)$. Hence most known known-plaintext or chosen-plaintext attacks against block ciphers do not seem to directly apply here. This general encryption scheme thus combines the merits of both stream cipher and block cipher, it also avoids their respective weaknesses. In particular, this scheme should provide encryption/decryption speed comparable to stream ciphers but higher than traditional block ciphers, and allow the use of a same encryption key for different messages.

Next we provide a couple of examples of implementation of the general scheme. In these examples, both a message block and the message encryption key have 128 bits to be comparable to most widely used block ciphers. Decryption procedures are omitted since they are straightforward based on the corresponding encryption ones.

Example 1 BC (*Block Chaining*) Mode
Choose a $(2^{16}-1, 8)$ code, then a 128-bit message block consists of 8 symbols over GF(2^{16}), each of which has 16 bits. Each index (position) of a codeword symbol needs 16 bits as well. Choose a strong cryptographic hash function $H(\cdot)$ whose output has at least 128 bits, e.g., MD5.

Let $M = m_1 m_2 \cdots m_8$ be a 128-bit message block and $I = i_1 i_2 \cdots i_8$ be a 8-sequence, where i_j is of 16 bits for $1 \leq j \leq 8$. $Enc(M, I)$ is a $(2^{16}-1, 8)$ RS encoder, which takes in M as 8 original information symbols and produces the corresponding codeword symbols at positions i_1 through i_8.

For a plaintext $M = M_1 M_2 \cdots M_b$ and an encryption key K, the corresponding ciphertext $C = C_1 C_2 \cdots C_b$ is encrypted as follows for $i = 1, 2, \cdots, b$:

$$\begin{cases} K_i = H(K_{i-1}) \\ J_i = H(J_{i-1}||M_{i-1}) \\ C_i = Enc(M_i \oplus K_i, J_i) \end{cases} \tag{1}$$

with

$$\begin{cases} M_0 = IV \\ K_0 = K||IV; \\ J_0 = (K >> 8)||IV; \end{cases} \tag{2}$$

where IV is a random initial vector of 128 bits, $(x >> n)$ is a cyclic right shift of x by n bits, and $||$ is bit concatenation.

IV is transmitted in plaintext. If the output of hash function $H(\cdot)$ has more than 128 bits, then the first 128 bits are used for K_i's and J_i's. In case two or more 16-bit symbols in J_i are equal, the latter symbols are simply iteratively increased by 1 until all the symbols of J_i are distinct. The same technique applies to *all-zero* symbols of J_i. □

In this example, the index stream J_i's for MDS encoding shares a same random stream generator with the key stream K_i's XORed with the plaintext stream M_i's before MDS encoding. This is to simplify implementation, especially in hardware to reduce the number of component circuits. Of course, the two streams J_i's and K_i's can use two different random stream generators, even with two different seed keys to increase encryption key size.

The encoding and decoding speeds of an (n, k) MDS code depends on both n and k. In general, the smaller n and k lead to higher encoding and erasure decoding speeds. Thus to further speed up encryption and decryption, one $(2^{16}-1, 8)$ RS codeword can be substituted by two codewords of a $(2^{16}-1, 4)$ RS code over GF(2^{16}) or a $(2^8-1, 8)$ RS code over GF(2^8). This does *not* reduce the complexity of brute force attacks, since stream cipher used to encrypt the plaintext stream before RS encoding does not change. Similarly more short codewords, e.g., four codewords of a $(2^{16}-1, 2)$ RS code over GF(2^{16}) or a $(2^8-1, 4)$ RS code over GF(2^8) can be used. Of course, as the number of short codewords used increases, the security strength of the overall encryption algorithm relies more on the random stream generator $H(\cdot)$, and thus it shifts more toward a stream cipher.

This CB mode of course can use $J_i = H(J_{i-1}||C_{i-1})$ instead of $J_i = H(J_{i-1}||M_{i-1})$. But in either case, a current ciphertext block depends not only on the current plaintext block but also on the preceding plaintext block, thus decryption of one message block needs multiple ciphertext blocks. This mode is not suitable for applications where random access is frequent, such as database applications. Our general encryption scheme, however, is very flexible to support *counter* mode to enable high speed random access of data blocks.

Example 2 Counter Mode
A proper RS code and a hash function $H(\cdot)$ are chosen in the same way as in the previous example.

For a plaintext $M = M_1 M_2 \cdots M_b$ and an encryption key K, the corresponding ciphertext $C = C_1 C_2 \cdots C_b$ is now encrypted as follows for $i = 1, 2, \cdots, b$:

$$\begin{cases} K_i = H(i||K||IV) \\ J_i = H(i||(K >> 8i)||IV) \\ C_i = Enc(M_i \oplus K_i, J_i) \end{cases} \tag{3}$$

with

$$\begin{cases} M_0 = IV \\ K_0 = K||IV; \\ J_0 = (K >> 8)||IV; \end{cases} \tag{4}$$

where again IV is a random initial vector of 128 bits, $(x >> n)$ is a cyclic right shift of x by n bits, and $||$ is bit concatenation. □

The random initial vector is public but it should be different for different plaintext messages to overcome replay attacks and to allow using a same encryption key for different messages.

4 Applications

Besides providing a stand-alone cipher for any applications, the general encryption scheme is especially suitable for applications where MDS encoding and decoding are necessary. There are many such applications, e.g., networked distributed data storage [1] and multi-route data transmission [8].

A distributed data storage system consists of n storage nodes connected via a wired or wireless network. A user's data is encoded using an (n, k) MDS code, and then distributed to the n storage nodes with one data symbol on each node. Such a distributed storage system provides high availability and high efficiency, since a user's data can be always recovered from any k of the n storage nodes. To ensure data confidentiality, the data symbols on the storage nodes can be further encrypted so that even if some of the storage nodes are compromised, the user data is still not exposed. The general encryption scheme can thus provide high encryption/decryption speed comparable to stream ciphers but with stronger cryptographic strength.

The idea of multi-route data transmission comes from *dispersity routing* [8]. The main idea of dispersity routing was to reduce the *mean* and *variance* of data delivery time by using multiple paths sending the *same* piece of data from a *single* source to a single destination. The destination (user) is always able to recover data

from the *fastest* route, thus reducing the delays on the routing paths caused by various reasons, such as bursty cross traffics, buffer overflows, packet losses on the paths. Extending this idea to wide area networks, especially wireless networks with multiple channels or routes whose delays and packet loss rates vary greatly from time to time, a source can encoded a user data block using an (n, k) MDS code, and then send n codeword symbols to the destination through multiple channels or routes. The user data can be recovered in a timely manner from any k codeword symbols delivered over multiple channels. When the data needs to be encrypted during transmission over insecure channels, the general encryption scheme can again be applied to provide both high encryption speed and strong security strength.

5 Conclusions

A general block encryption scheme based on MDS code has been proposed. This scheme combines a random stream generator and an MDS code to obtain merits of both stream cipher and block cipher, while lessening their weakness. In particular, such an encryption scheme can provide high encryption/decryption speed comparable to a stream cipher, but offers much higher security strength than the corresponding component stream cipher. A few implementations and applications of the general encryption scheme have been briefly discussed. Future work includes detailed implementation and cryptanalysis of the general scheme, as well as comparisons to other stream ciphers and block ciphers in both performance (computation complexity) and security strength.

References

[1] V. Bohossian, C. Fan, P. LeMahieu, M. Riedel, L. Xu and J. Bruck, "Computing in the RAIN: A Reliable Array of Independent Node", *IEEE Trans. on Parallel and Distributed Systems*, Special Issue on Dependable Network Computing, 12(2), 99-114, Feb. 2001.

[2] N. Borisov, I. Goldberg and D. Wagner, "Intercepting Mobile Communications: The Insecurity of 802.11", *Proc. of ACM MOBICOM*, July 16-21, 2001, Italy.

[3] E. Dawson and L. Nielsen, "Automated cryptanalysis of XOR plaintext strings", *Cryptologia*, (2):165–181, Apr. 1996

[4] C. Ding and A. Salomaa, "Cooperatively Distributed Hashing and Ciphering", *Computers and Artificial Intelligence*, 15, 233-245, 1996.

[5] L. S. Hill, "Cryptography in An Algebraic Alphabet", *American Mathematical Monthly*, 36, 306-312, 1929.

[6] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*, Amsterdam: North-Holland, 1977.

[7] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 4th Printing, 1999.

[8] F. N. Maxemchuk, "Dispersity Routing in Store-and-Forward Networks," Ph.D. thesis, University of Pennsylvania, 1975.