Applied Sig Process (1999)6:177–181

Applied
Signal
Processing

A New Improved ZS Algorithm for a Synchronous Stream Cipher

H. Lee[1] and S. Moon[2]

[1]Department of Computer Engineering, Kyungwoon University, Kyungbuk, Korea; [2]School of Electronic and Electrical Engineering, Kyungpook National University, Taegu, Korea

When we apply a synchronous stream cipher to the T1 carrier system, long consecutive-zero sequences can occur in the received random ciphertext data. In this case, the sequences give difficulty in recovering the receiver clock and the zero suppression (ZS) algorithm with block detection has been proposed to prevent consecutive-zero sequences of more than $k = (\geq 2)$. In this paper we propose a new improved ZS algorithm to apply stream cipher directly using a serial detection method, and analyse its properties.

Keywords: Cryptography; Keystream synchronisation; Stream cipher; Zero suppression

1. Introduction

In a synchronous stream cipher [1–3], ciphertext bits are randomly distributed because they are the exclusive-ored value of a '0'–'1' balanced keystream (in frequency) with plaintext [1]. The probability of k consecutive zeros in ciphertext at the sender is thus 2^{-k}. In applying a synchronous stream cipher to point-to-point link encryption in a T1-carrier system [4], 15 or more consecutive zeros in ciphertext at the sender may be limited. Without limitation, the receiver clock (timing signal) may be unstable and the equipment then requires resynchronisation, since deciphering becomes impossible once synchronisation is lost. These resynchronisations, when frequent, can lead to significant deterioration of cryptographic security [5].

To solve this, a zero suppression algorithm (named ZS-1) [6] which suppressed k or more zeros between successive ones in ciphertext at the sender and recovered the original message completely at the receiver of a synchronous stream cipher system was proposed. But it is not easy to apply ZS-1 to a stream cipher system, because a stream cipher is a bit-operation principle (exclusive-ored) and ZS-1 is a block operation (block detection method) which differs from the bit operation in method. So we propose a new ZS algorithm (named ZS-2), whereas ZS-1 is operated by a block detection method, ZS-2 will be operated by a serial bit-stream detection method. With two algorithms, the probability of k consecutive zeros in ciphertext at the sender is reduced to 0. It is useful for systems which limit consecutive zeros, such as a T1-carrier system (with $k = 15$) [2].

Correspondence and offprint requests to: H. Lee, Department of Computer Engineering, Kyungwoon University, San 5-1 Indukri, Sandong-Myun, Kumi, Kyungbuk 730-850, Korea. E-mail: hjlee@kyungwoon.ac.kr

2. Zero Suppression Algorithms

The ZS algorithm consists of a detection part which detects n-bit zeros (blocks of n consecutive zeros), and the substitution part which substitutes n-bit zeros with a nonzero block. Detection methods are divided into serial detection, which detects an all-zero input block (00 … 0) by checking on a bit-by-bit serial interval with block size containing previous bits in buffer, and block detection, which detects an all-zero input block by checking on block interval with block size. Substitution methods are divided into current block substitution and mixing block substitution, which substitutes three or more blocks (preceding blocks, present block and next blocks) with non-zero random blocks.

Let:

- $\mathbf{P}_i(p_{in}, p_{in-1}, \ldots, p_{in-n+1})$ be the ith n-bit plaintext vector block.
- $\mathbf{K}_i(k_{in}, k_{in-1}, \ldots, k_{in-n+1})$ the ith n-bit keystream block.
- $\mathbf{C}_i(c_{in}, c_{in-1}, \ldots, c_{in-n+1})$ the ith n-bit ciphertext block.
- $\mathbf{Q}_i(q_{in}, q_{in-1}, \ldots, q_{in-n+1})$ the ith n-bit recovered plaintext block at the receiver.
- $\mathbf{0}$ (0, 0, …, 0) the n-bit 0 vector ($i > 0$).

The block size is

$$n = \left\lceil \frac{(k+1)}{2} \right\rceil$$

where $\lceil x \rceil$ denotes the maximum integer which is not over x.

ZS-1 is described in the following assumptions and the algorithm [6].

2.1. Assumptions

1. A redundant bit insertion or deletion at the sender is not permitted. (It is a difficult problem to control clock rate in an intermediated synchronous stream cipher system.)
2. For all i, $\mathbf{P}_i \neq \mathbf{0}$, i.e., $\mathbf{P}_i$ is a n-bit nonzero vector ($i \geq 0$, checking in block interval n).
3. Cryptographically strong keystream sequences must be used in the system.

2.2. ZS-1 Algorithm

Sender:

1. Put $\mathbf{P}_i \oplus \mathbf{K}_i$ and $\mathbf{K}_i$ to two n-stage shift registers respectively.
2. Check $\mathbf{P}_i \oplus \mathbf{K}_i = \mathbf{0}$.
3. If $\mathbf{P}_i \oplus \mathbf{K}_i = \mathbf{0}$, the output is $\mathbf{C}_i = \mathbf{K}_i$ (keystream block).
   Otherwise, the output is $\mathbf{C}_i = \mathbf{P}_i \oplus \mathbf{K}_i$ (ciphertext block).

Receiver:

1. Put $\mathbf{C}_i \oplus \mathbf{K}_i$ and $\mathbf{K}_i$ to two n-stage shift registers respectively.
2. Check $\mathbf{C}_i \oplus \mathbf{K}_i = \mathbf{0}$.
3. If $\mathbf{C}_i \oplus \mathbf{K}_i = \mathbf{0}$, the output is $\mathbf{Q}_i = \mathbf{K}_i$.
   Otherwise, the output is $\mathbf{Q}_i = \mathbf{C}_i \oplus \mathbf{K}_i$.

Theorem 1. Under the condition that a plaintext block $\mathbf{P}_i \neq \mathbf{0}$ for all i in a synchronous stream cipher, the ZS-1 algorithm limits the output of consecutive k-bit zeros at the sender and recovers plaintext exactly at the receiver.

Proof. (i) Since $\mathbf{P}_i \neq \mathbf{0}$ for all i, the output of ZS-1 at the sender cannot be permitted to contain consecutive $(2n-1)$-bit zeros.

(ii) When $\mathbf{P}_i \oplus \mathbf{K}_i = \mathbf{0}$ is detected, $\mathbf{C}_i = \mathbf{K}_i$ is transmitted to the receiver, and $\mathbf{P}_i = \mathbf{K}_i$. The receiver detects $\mathbf{C}_i \oplus \mathbf{K}_i = \mathbf{K}_i \oplus \mathbf{K}_i = \mathbf{0}$ and then outputs $\mathbf{Q}_i = \mathbf{K}_i = \mathbf{P}_i$. So plaintext is completely recovered.

(iii) Whenever $\mathbf{P}_i \oplus \mathbf{K}_i \neq \mathbf{0}$, $\mathbf{C}_i = \mathbf{P}_i \oplus \mathbf{K}_i$ is transmitted to the receiver, the receiver deciphers $\mathbf{C}_i \oplus \mathbf{K}_i = (\mathbf{P}_i \oplus \mathbf{K}_i) \oplus \mathbf{K}_i = \mathbf{P}_i \neq \mathbf{0}$ and then outputs $\mathbf{Q}_i = \mathbf{C}_i \oplus \mathbf{K}_i = \mathbf{P}_i$. So plaintext is completely recovered.

Theorem 2. In the ZS-1 algorithm, the only reversible substitution vector is a current keystream block when the detected ciphertext block is all-zero.

Proof. A random vector $\mathbf{R}_i(\neq \mathbf{K}_i)$ was substituted in the sender, and false detection in the receiver arose when the ciphertext block was matched to the random vector unexpectedly, i.e., $\mathbf{C}_i \oplus \mathbf{K}_i = \mathbf{R}_i$. In that case, ciphertext could not be recovered to the original plaintext. So the only revertible substitution vector is a current keystream vector $\mathbf{K}_i$.

Therefore ZS-1 is very meaningful in the sense that any substitution of a random block can not be reverted to the original data.

3. New Algorithm

ZS-1 is good but not useful to a stream cipher system, because a stream cipher is a bit-operation principle (exclusive-ored) and ZS-1 is a block operation (block detection method) which differs from the bit operation in the method. So we propose a new algorithm, ZS-2 using a bit-serial detection method. For serial detection we define block variables to increase bit-wise in the following.

Let:

- $\mathbf{P}_i(p_i, p_{i-1}, \ldots, p_{i-n+1})$ be the ith n-bit plaintext vector block.
- $\mathbf{K}_i(k_i, k_{i-1}, \ldots, k_{i-n+1})$ the ith n-bit keystream block.
- $\mathbf{C}_i(c_i, c_{i-1}, \ldots, c_{i-n+1})$ the ith n-bit ciphertext block.
- $\mathbf{Q}_i(q_i, q_{i-1}, \ldots, q_{i-n+1})$ the ith n-bit recovered plaintext block at the receiver.
- $\mathbf{0}$ (0, 0, …, 0) the n-bit 0 vector ($i > 2n$).

The block size is the same as in ZS-1.

ZS-2 is described in the following assumption and the algorithm.

3.1. Assumptions

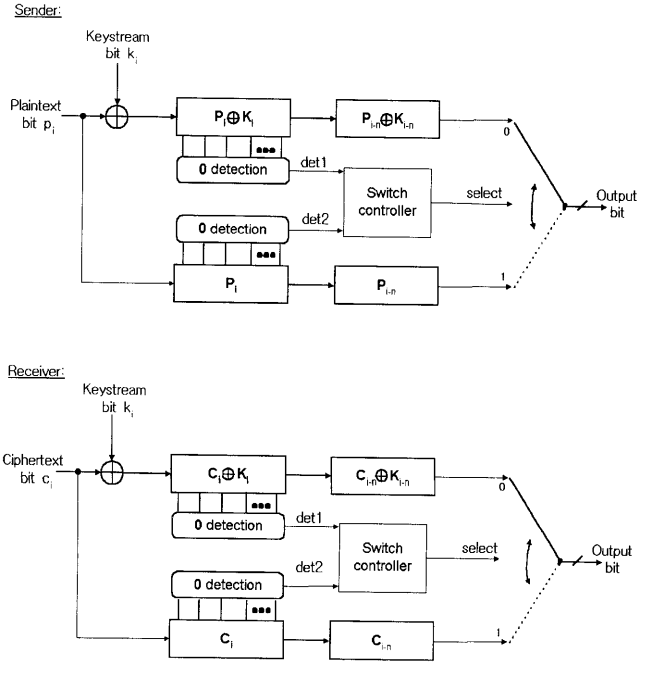
1. The same as assumption 1 and 3 in ZS-1.
2. It is not permitted to present k-bit consecutive zeros in plaintext (checking in serial time, not block interval).

3.2. ZS-2 Algorithm

Sender (see Fig. 1):

1. Put plaintext bit p_i and ciphertext bit $p_i \oplus k_i$ in two n-stage shift registers respectively, and shift four registers ($\mathbf{P}_i$, $\mathbf{P}_{i-n}$, $\mathbf{P}_i \oplus \mathbf{K}_i$, $\mathbf{P}_{i-n} \oplus \mathbf{K}_{i-n}$) (Fig. 1).
2. Check two n-bit vector $\mathbf{P}_i = \mathbf{0}$ or $\mathbf{P}_i \oplus \mathbf{K}_i = \mathbf{0}$.
3. If $\mathbf{P}_i \neq \mathbf{0}$, $\mathbf{P}_i \oplus \mathbf{K}_i \neq \mathbf{0}$, the output is the previously buffered 1-bit ciphertext,

$$c_{i-2n} = p_{i-2n} \oplus k_{i-2n}$$



※ Switch controller : if det2=1, select SW=1 and output 3n bits
else if det1=1, select SW=1 and output n bit
else, select SW=0 and output 1 bit

Fig. 1. ZS-2 algorithm.

otherwise if $\mathbf{P}_i \neq \mathbf{0}$, $\mathbf{P}_i \oplus \mathbf{K}_i = \mathbf{0}$, the output is the previously buffered n-bit vector,

$$\mathbf{P}_{i-n} (= p_{i-n}, p_{i-n-1}, p_{i-n-2}, \cdots, p_{i-2n+1})$$

otherwise if $\mathbf{P}_i = \mathbf{0}$, the output is three n-bit vectors,

$$\mathbf{P}_{i-n}, \mathbf{P}_i \text{ and } \mathbf{P}_{i+n}$$

Receiver (see Fig. 1):

1. Put ciphertext bit c_i and plaintext bit $c_i \oplus k_i$ to two n-stage shift registers respectively, and shift four registers ($\mathbf{C}_i$, $\mathbf{C}_{i-n}$, $\mathbf{C}_i \oplus \mathbf{K}_i$, $\mathbf{C}_{i-n} \oplus \mathbf{K}_{i-n}$) in Fig. 1.
2. Check two n-bit vector $\mathbf{C}_i = \mathbf{0}$ or $\mathbf{C}_i \oplus \mathbf{K}_i = \mathbf{0}$.
3. If $\mathbf{C}_i \neq \mathbf{0}$, $\mathbf{C}_i \oplus \mathbf{K}_i \neq \mathbf{0}$, the output is the previously buffered 1-bit plaintext,

$$p_{i-2n} = c_{i-2n} \oplus k_{i-2n}$$

otherwise if $\mathbf{C}_i \neq \mathbf{0}$, $\mathbf{C}_i \oplus \mathbf{K}_i = \mathbf{0}$, the output is the previously buffered n-bit vector,

$$\mathbf{C}_{i-n} = \mathbf{P}_{i-n} (= p_{i-n}, p_{i-n-1}, p_{i-n-2}, \cdots, p_{i-2n+1})$$

otherwise if $\mathbf{C}_i = \mathbf{0}$, the output is three n-bit vectors,

$$\mathbf{C}_{i-n} = \mathbf{P}_{i-n}, \mathbf{C}_i = \mathbf{P}_i \text{ and } \mathbf{C}_{i+n} = \mathbf{P}_{i+n}$$

.

Theorem 3. Under the condition that it is not permitted to present k-bit consecutive zeros for a plaintext in a synchronous stream cipher, the ZS-2 algorithm limits the output of consecutive k-bit zeros at the sender and (excluding channel error) recovers plaintext exactly at the receiver.

Proof. (i) If $\mathbf{P}_i \neq \mathbf{0}$ and $\mathbf{P}_i \oplus \mathbf{K}_i \neq \mathbf{0}$, then $c_{i-n} = p_{i-n} \oplus k_{i-n}$ is transmitted to the receiver. So the receiver output is $q_{i-n} = c_{i-n} \oplus k_{i-n} = (p_{i-n} \oplus k_{i-n}) \oplus k_{i-n} = p_{i-n}$, and so plaintext is completely recovered.

(ii) When detected, only $\mathbf{P}_i \oplus \mathbf{K}_i = \mathbf{0}$ ($\mathbf{P}_i \neq \mathbf{0}$ and $\mathbf{P}_i = \mathbf{K}_i$), and then $\mathbf{C}_{i-1} = \mathbf{P}_{i-1} \oplus \mathbf{K}_{i-1}$ and $\mathbf{C}_i = \mathbf{P}_i$ are transmitted to the receiver. So the receiver detects $\mathbf{C}_i \oplus \mathbf{K}_i = \mathbf{P}_i \oplus \mathbf{P}_i = \mathbf{0}$ and then outputs $\mathbf{Q}_{i-1} = \mathbf{C}_{i-1} \oplus \mathbf{K}_{i-1} = \mathbf{P}_{i-1} \oplus \mathbf{K}_{i-1} \oplus \mathbf{K}_{i-1} = \mathbf{P}_{i-1}$ and $\mathbf{Q}_i = \mathbf{C}_i = \mathbf{P}_i$, so plaintext is completely recovered.

(iii) When detected, $\mathbf{P}_i = \mathbf{0}$, $\mathbf{C}_{i-1} = \mathbf{P}_{i-1}$, $\mathbf{C}_i = \mathbf{P}_i$, $\mathbf{C}_{i+1} = \mathbf{P}_{i+1}$ are transmitted to the receiver. Then the receiver detects $\mathbf{C}_i = \mathbf{0}$ and outputs $\mathbf{Q}_{i-1} = \mathbf{C}_{i-1} = \mathbf{P}_{i-1}$, $\mathbf{Q}_i = \mathbf{C}_i = \mathbf{P}_i$, and $\mathbf{Q}_{i+1} = \mathbf{C}_{i+1} = \mathbf{P}_{i+1}$, so plaintext is completely recovered.

(iv) When detected, $\mathbf{P}_{i-1} = \mathbf{0}$ and $\mathbf{P}_i = \mathbf{0}$, ZS-2 outputs $\mathbf{C}_{i-1} = \mathbf{P}_{i-1}$ and $\mathbf{C}_i = \mathbf{P}_i$, so the output of ZS-2 at the sender cannot permit k (= $2n-1$ or $2n$)-bit consecutive zeros by assumption, where

$$n = \lceil \frac{k+1}{2} \rceil$$

and k is a positive odd or even number (selected from system characteristics), for example, $n = 8$ for $k = 15$ or 16.

In a synchronous stream cipher, ciphertext bits are randomly distributed, because they are the exclusive-ored value of a '0'–'1' balanced keystream with plaintext. The probability of k consecutive zeros in ciphertext at the sender is thus 2^{-k}. In applying a synchronous stream cipher to point-to-point link encryption in a T1-carrier system, 15 or more consecutive zeros in ciphertext at the sender may be limited. Without ZS, the receiver clock may be unstable and lose keystream synchronisation. The equipment then requires resynchronisation, since deciphering becomes impossible once keystream synchronisation is lost. These resynchronisations, when frequent, can lead to significant deterioration of cryptographic security [5]. Applying ZS to a T1-carrier system for link encryption with $k = 15$, $n = 8$, we can confirm improved system performance for clock recovery, ie., ZS prevents a serious deterioration of performance of keystream synchronisation.

We summarise our proposal in Table 1. ZS is a method to encrypt data with a stream cipher that reduces the occurrence of long consecutive-zero sequences. In this way, the receiver clock can more easily be recovered, while keeping cryptographic security of the underlying stream cipher. In this paper, we propose a new algorithm, ZS-2, updated from our original proposed algorithm, ZS-1. The main difference between the two is that in ZS-1, checking for consecutive zeros is done block by block, while in ZS-2, all sequences are checked. In other words, ZS-1 uses a block detection method, but ZS-2 uses a precise serial detection method; ZS-1 requires block synchronisation on the start and the end of each block, but ZS-2 does not require block synchronisation. Finally, in terms of hardware implementation, ZS-2 is simpler because of this omission of block synchronisation.

4. Conclusion

We have proposed a ZS-2 algorithm which suppresses blocks of k or more consecutive zeros of ciphertext at the sender and recovers the original message exactly at the receiver by bit-serial operation, like a stream cipher. It is updated from ZS-1, a useful algorithm for solving the difficulty of receiver clock recovery in a synchronous stream cipher which may occur due to excess zeros. The main difference between the two is that in ZS-1, checking for consecutive zeros is done block by block, while in ZS-2, all sequences are checked. In other words, ZS-1 uses a block detection method, but ZS-2 uses a precise serial detection method; ZS-1 requires block synchronisation on the start and the end of each block, but ZS-2 does not require block synchronisation. Finally, in terms of hardware implementation, ZS-2 is simpler because of this omission of block synchronisation.

Table 1. The result of comparison ZS-1 with ZS-2.

Items	ZS-1	Proposed ZS-2
Detection method	Block detection: detects all-zero input blocks by checking block interval with block size	Serial detection: detects all-zero input block (00 … 0) by checking bit-by-bit serial interval with block size containing previous bits in buffer
Requirement of block synchronisation	Needs block synchronisation that synchronises to bit position on the start and the end	Block synchronisations not required
Simplicity of implementation	Complex hardware implementation due to need to append block synchronisation	Simpler than ZS-1
On the system clock recovery	In applying a synchronous stream cipher to point-to-point link encryption in a T1-carrier system, 15 or more consecutive zeros in ciphertext at the sender may be limited	In applying a synchronous stream cipher to point-to-point link encryption in a T1-carrier system, 15 or more consecutive zeros in ciphertext at the sender may be limited

References

1. Beker HJ, Piper FC. Cipher systems: the protection of communications. Northwood Books, London, 1982
2. Tilborg HCA. An introduction to cryptology. Kluwer Academic Publishers, Boston, 1988
3. Golomb SW. Shift register sequences. Holden-Day, San Francisco, 1967
4. CCITT Recommendation. Physical/electrical characteristics of hierarchical digital interface. In: CCITT red book, 1985, vol III, Rec G 703
5. Daemen J, Govaerts R, Vandewalle J. Resynchronization weaknesses in synchronous stream ciphers. Advances in cryptology – Eurocrypto'93, Lecture Notes in Computer Science, No 765, Springer-Verlag, 1994, pp 159–167
6. Lee H, Park B, Chang B, Moon S. A zero suppression algorithms for a synchronous stream cipher. Applied Sig Process 1998; 5: 240–243