JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 16, 857-861 (2000)

**Short Paper**

# Dynamic GOST

YI-SHIUNG YEH, CHU-HSING LIN[*] AND CHAN-CHI WANG
*Institute of Computer Science and Information Engineering*
*National Chiao Tung University*
*Hsinchu, Taiwan 300, R.O.C.*
*E-mail: ysyeh@csie.nctu.edu.tw*
*[*]Department of Computer Science and Information Engineering*
*Tunghai University*
*Taichung, Taiwan 407, R.O.C.*
*E-mail: chlin@mail.thu.edu.tw*

GOST is a block algorithm which was adopted as a standard by the former Soviet Union. Because it keeps a lot of secret information, including S-boxes, the algorithm is secure and easy to understand. In this article, we propose a variant of GOST, called dynamic GOST, in which permutations are applied. With the permutation information securely kept, the new version of GOST is more secure and better to withstand differential and linear attacks.

***Keywords:*** GOST, block cipher, S-boxes, DES, subkey generation, cryptanalysis, differential attack, linear attack, random permutation

## 1. INTRODUCTION

GOST is a block algorithm used as a government standard in the former Soviet Union. It is a Feistel network for cryptographically protecting data processing systems [1,2]. The important role GOST played in the former Soviet Union was not less significant than that of the Data Encryption Standard in the U. S. The algorithm processes a 64-bit message by using a 256-bit key and some secret information, including S-boxes. There are 32 rounds in GOST, as opposed to 16 rounds in DES. Therefore, it needs 32 subkeys, which are generated by a 256-bit key (or 8 32-bit subkeys), for the 32 rounds. Initially, the eight 32-bit subkeys are denoted as $\overline{k_1}, \overline{k_2}, \ldots, \overline{k_8}$, where $\overline{k_i}$ is a 32-bit subkey. Let $k_i$ be the $i^{th}$ subkey for round $i$. Then, $k_i = \overline{k}_{(i \bmod 8)}$ for $\overline{k_0} = \overline{k_8}$ and $1 \le i \le 24$, and $k_i = \overline{k}_{(33-i) \bmod 8}$ for $25 \le i \le 32$ and $\overline{k_9} = \overline{k_1}$. With these subkeys, the transformation in round $i$ of GOST can be described as:

$$L_i = R_{i-1} \text{ and } R_i = L_{i-1} \oplus f(R_{i-1}, k_i),$$

where $L_0R_0$ is a plaintext message with 32-bits in each part.

Received September 2, 1998; revised November 30, 1998; accepted March 18, 1999.
Communicated by Chi Sung Laih.

Further, there are 8 S-boxes in GOST, and each S-box is a permutation on {0, 1, ..., 15}. Let S be an ordered set of the 8 S-boxes in GOST. Then, the function *f* can be denoted as:

$$f(X, Y) = S(X \boxplus Y) <<< 11,$$

where $X \boxplus Y = X + Y \bmod 2^{32}$ and <<< *t* is a *t*-bit left circular shift.

Compared with DES, GOST needs 32 rounds with simple permutation, instead of the complex S-boxes in DES and an 11-bit left circular shift after permutation through the S-boxes. In addition to the keys, the S-boxes should be kept secret. In total, there are about 610 bits of secret information in GOST.

GOST is designed to obtain a balance between efficiency and security. The algorithm is better suited to software implementation since it modifies DES's basic design, but security is weaker. Therefore, to compensate for this weakness, the key length is very large, the round number is double in size, and the S-boxes are kept secret. This secret information may facilitate resisting differential and linear attacks. However, the same advantage does not exist when the S-boxes are known to the adversary. To solve this problem, we shall propose a new version of GOST – dynamic GOST. In dynamic GOST, S-boxes are variable and dependent on keys; thus, GOST can resist the differential and linear attacks [3, 4].

## 2. S-BOXES IN DYNAMIC GOST

To solve the problem stated previously, more effective and flexible use of S-boxes in dynamic GOST is proposed. A random permutation is applied to the S-boxes in each round, and security is significantly enhanced.

The idea is to rearrange the S-boxes in the succeeding rounds by using different permutations. Let *p*: [1..8]→[1..8] be a permutation used by the $j^{th}$ round to reorder its S-boxes. Then, the $i^{th}$ S-box in the $j^{th}$ round will be the $p(i)^{th}$ S-box in the $(j-1)^{th}$ round. For simplicity, let us explain our idea by using an example. Suppose that the S-box sequence in some round, say the $(j-1)^{th}$ round, is $S_1S_2S_3S_4S_5S_6S_7S_8$, and that the given permutation is (3,7,2,4,1,8,6,5). Then, the S-box sequence in the $j^{th}$ round will be $S_3S_7S_2S_4S_1S_8S_6S_5$. Since there are 32 rounds, we in total need 31 permutations in dynamic GOST.

Since these permutations are kept secret, the exact use of S-boxes is not explicit. Since the S-boxes are dynamic and kept secret, the difficulty of performing cryptanalysis will be increased.

## 3. THE PERMUTATIONS

In dynamic GOST, the 31 permutations must be kept secret. These permutations may be considered as being secret information in the system in addition to the S-boxes and keys. This will increase the quantity of secret information, but the system will become more secure. On the other hand, since there is more data to be kept secret, the load on users may be increased.

Alternatively, we can design the scheme such that the permutations are dependent on the existing secret materials (e.g. the key). Note that a system is considered to be more secure if the S-boxes are dependent on the keys [1]. In the worst case, the space required for the permutations will be less than 744(=24 $\times$ 31) bits, 3 bits for each of the 8 S-Boxes. To save memory space, it is possible to design permutations, e.g., by using a modular arithmetic function, though security will be somewhat degraded. The simplest example is that we can choose a small integer B, a small relative prime to 9, as the multiplier. Then, one permutations, say $p$, can be denoted as $p(i) = i \times B$ mod 9, and the others can be designed similarly.

## 4. IMPLEMENTATION OF S-BOXES

With respect to implementation, we will consider how the S-boxes are retained. They can be maintained in a table that has a two dimensional array 8 $\times$ 16 in size. Without loss of generality, let the table be denoted as M[1..8,1..16], and let the S-boxes sequence for the ($j$ – 1)$^{th}$ round be $S_1S_2S_3S_4S_5S_6S_7S_8$. Then, the $k^{th}$ word (4-bit) of $S_i$ is placed in M[i,k]. Now, if a permutation function $p$ is applied in the encryption process, then the S-box sequence in the $j^{th}$ round will become $S_{p(1)}S_{p(2)}S_{p(3)}S_{p(4)}S_{p(5)}S_{p(6)}S_{p(7)}S_{p(8)}$. In other words, the $k^{th}$ word of the $i^{th}$ S-box can be looked up from entry M[$p$(i),k] in the table.

As stated previously, in implementation, when a permutation function is included, a word for any S-box can be easily found by a table look-up. Compared with the original GOST, we can see that the extra cost in dynamic GOST lies in the permutation function. In each round, excluding the first round, we need a permutation, and a total of 31 permutations are required. From the standpoint of efficiency, the proposed new version is as good as GOST.

On the other hand, in a decryption process, the same 32 S-box sequences will be used, but in the reverse order. The computing complexity will not increase.

## 5. SECURITY ANALYSIS

It is necessary to know the exact use of S-boxes for both linear attack and differential attack to be successful. In the proposed version of GOST, the S-boxes are dynamic, are kept secret, and are dependent on the key. Neither linear or differential attacks can work [1]. Since we keep the permutations secret, it will be more difficult for an adversary to apply these two types of attacks. One way is to guess the 31 permutations correctly and then to follow the attack steps as in original GOST. However, the probability of getting the correct ordering of eigh S-boxes in one roud is $\frac{1}{40320}$ since there are 8!= 40320 different orderings.

Furthermore, in dynamic GOST, we adopt different permutations in each round for the sake of higher security. There are P(16!, 8)=(16!)!/(16!-8)! ways to assign an S-Box in GOST or in the 1st round of our dynamic GOST. The number of possible S-Boxes for the remaining 31 rounds are $(40320)^{31}$. In total, we have P(16!, 8) $\times$ $(40320)^{31}$ possible ways to assign S-Boxes for all 32 rounds in dynamic GOST.

## 6. CONCLUSIONS

By permutating the order of an S-box sequence in succeeding rounds, the usage of S-boxes can be varied. This change can help GOST resist differential and linear attacks. However, the permutations should be kept secret; otherwise, the confusion effect will no longer exist.

## ACKNOWLEDGMENT

The authors thank the National Science Council of the R.O.C. for supporting this research under contract NSC 88-2213-E-029-007. We thank the anonymous reviewers for their valuable comments and suggestions.

## REFERENCES

1. Bruce Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd edition, John Wiley & Sons, Inc., 1996.
2. Bruce Schneier, "The GOST encryption algorithm," *Dr. Dobb's Journal*, Vol. 20, No. 1, 1995, pp. 123-124.
3. M. Matsui, "Linear cryptanalysis method for DES cipher," *Advances in Cryptlogy-Eurocrypt '*93 *Proceedings*, 1994, pp. 74-87.
4. E. Biham and A. Shamir, "Differential cryptanalysis of the full 16-round DES," *Advances in Cryptology-CRYPTO '*92 *Proceedings*, 1993, pp. 486-487.

**Yi-Shiung Yeh** (葉義雄) received the B.S. degree in Math. from National Central University, Chung-Li, Taiwan, in 1975, and the M.S. and Ph.D. degrees in Computer Sciences from the University of Wisconsin-Milwaukee, U.S.A., in 1981 and 1985, respectively. He is currently an associate professor in the Institute of Computer Science and Information Engineering at Chiao-Tung University, Hsinchu, Taiwan. His interests include information security and cryptography, many-valued logical systems, reliability and performance evaluation of distributed systems, and computer networks.

**Chu-Hsing Lin** (林祝興) received his B.S. degree in applied mathematics from National Tsing Hua University in 1980, his M.S. degree, also in applied mathematics, from National Chung Hsing University in 1987, and his Ph.D. degree in computer sciences from National Tsing Hua University in 1991. He completed two year of compulsory army service at Taiwan, after he finished his university education. From 1983 to 1985, he worked for the Information Department of the Land Bank of Taiwan, and he joined the team in developing the banking system. Now he is an associate professor in the Department of Computer and Information Sciences at Tunghai University. Since 1995, he has also been the Director of the Computer Center of the University. He built the campus ATM network, finished the dorm net, and set up the information system for Academic Affairs based on a client-server architecture and the campus net. Since 1997, he has been one of the supervisors of the Chinese Information Security Association. He was the winner of the 1991 Acer Long-Term Award for Outstanding Ph.D. Dissertation. His current research interests include information security, cryptology, data engineering, and electronic commerce.

**Chan-Chi Wang**(王銓祺) received the B.S., M.S., and Ph.D. degrees in Computer Science and Information Engineering at Chiao-Tung University, Hsinchu, Taiwan, in 1991, 1993, and 1998, respectively. He is currently performing military service. His interests include information security and cryptography, computer networks, and operating systems.