# Soviet Encryption Algorithm

from Russian translated by

Josef Pieprzyk *
Leonid Tombak

Department of Computer Science
University of Wollongong
Wollongong, NSW 2500

e-mail: josef/leo@cs.uow.edu.au

June 1, 1994

## 1 Introduction

This report describes the Soviet (now Russian) encryption algorithm. The details of the algorithm were published in 1990 as the Soviet Standard (GOST 28147-89) [1]. The algorithm provides a level of security of information that is flexible and which can be used to protect information in computer systems and computer networks. The algorithm can be implemented in both hardware and software.

## 2 The description of the algorithm

The algorithm (also called cryptosystem) is summarized in Figure 1.

The algorithm is designed using the following components:

1. The key store unit (KSU) is capable of storing a 256-bit string using eight 32-bit registers $(K_0, K_1, \ldots, K_7)$.
2. Four 32-bit registers $(R_1, \ldots, R_4)$.
3. Two 32-bit registers $(R_5, R_6)$ which contain two constants $C_2, C_1$.
4. Two 32-bit adders modulo $2^{32}$ $(CM_1, CM_3)$ (for two 32-bit words $x, y$, the adder produces $x + y \pmod{2^{32}}$).
5. The bitwise Exclusive-Or adder $(CM_2)$ for two 32-bit words.
6. The 32-bit adder modulo $(2^{32} - 1)$ (the adder $CM_4$)

*Support for this project was provided in part by the Australian Research Council under the reference number A49131885.

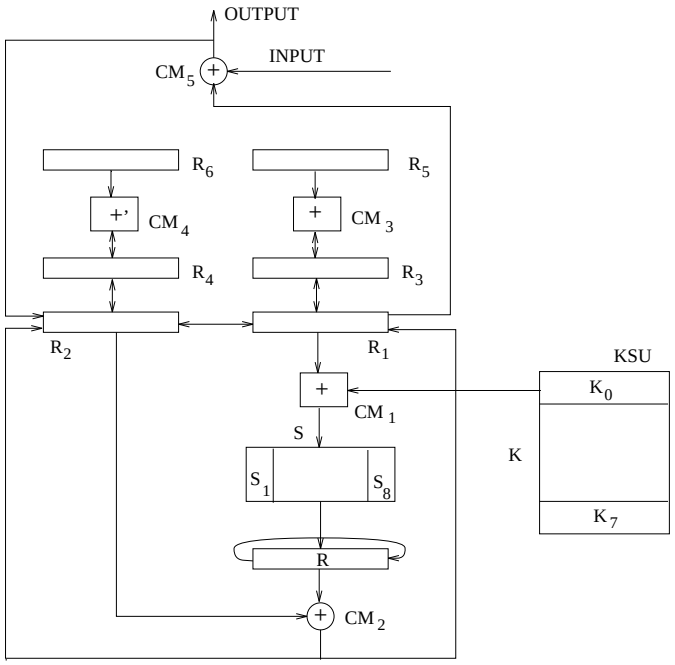


Figure 1: General scheme of the algorithm

7. The bitwise Exclusive-Or adder ($CM_5$). There is are restrictions on the number of bits for the adder ($CM_5$).

8. The substitution block (S).

9. The cyclic shift register (R) which shifts the contents 11 bits towards the highest bit.

The substitution block S consists of eight S-boxes $S_1, .., S_8$ each of size 64 bits. The 32-bit input to $S$ is divided into eight 4-bit vectors. Each 4-bit vector is transformed to another 4-bit vector by the corresponding S-box. Each S-box $S_i$ $(i = 1, \ldots, 8)$ is a permutation stored in the form of a table with 16 rows. Each row is addressable by the input vector. The output of an S-box is the value stored at the row addressed by the input. All 4-bit outputs from the S-boxes are concatenated into one 32-bit string.

Note that the high-order bits are the left bits for addition and shifting.

The algorithm applies a 256-bit key $K = (k_1, \ldots, k_{256})$ where $k_i \in \{0, 1\}$ for $i = 1, \ldots, 256$. The key is stored in the KSU where

$$\begin{aligned} (k_{32}, \ldots, k_1) &= K_0 \\ (k_{64}, \ldots, k_{33}) &= K_1 \\ &\vdots \\ (k_{256}, \ldots, k_{225}) &= K_7 \end{aligned}$$

so $k_1$ is the first bit of $K_0$, $k_2$ is the second bit of $K_0$, and so on.

When the contents of one register $\alpha = (a_1, \ldots, a_n)$ is copied to another register $\beta = (b_1, \ldots, b_n)$, then $b_i = a_i$ for $(i = 1, \ldots, n)$.

The values of constants $C_1$ and $C_2$ are: $C_1 = 2^{24} + 2^{16} + 2^8 + 2^2$ and $C_2 = 2^{24} + 2^{16} + 2^8 + 1$.

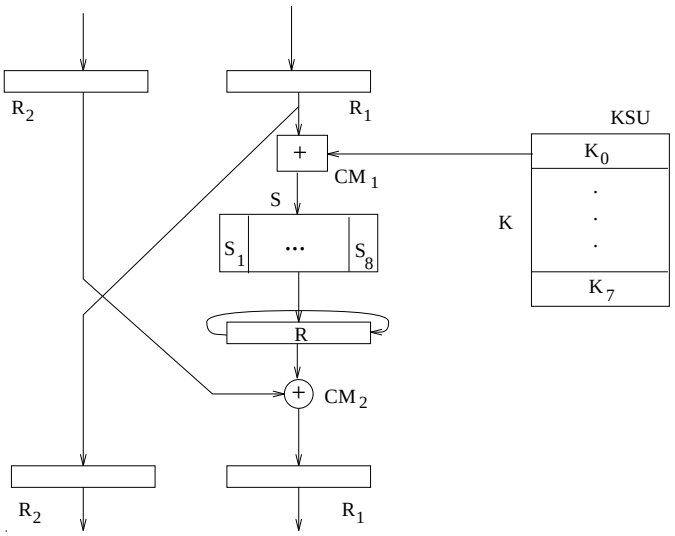


Figure 2: Information flow for a single iteration of the encryption/decryption

**Both the key $K$ which is stored in the KSU and the tables of $S$ (the S-boxes) are secret elements of the algorithm.**

When the algorithm is applied in computer networks, the substitution block $S$ is shared and is used for a long time.

The cryptosystem can work in the following four modes:

1. Simple Substitution mode (SS mode),
2. Stream mode ($\Gamma$-mode),
3. Stream mode with feedback,
4. Authentication mode.

# 3 Simple substitution mode

## 3.1 Plaintext encryption in SS mode

The cryptosystem for the Simple Substitution mode is shown in Figure 2. The plaintext is divided into 64-bit long blocks $T_0$. Each block

$$T_0 = (a_1(0), ..., a_{32}(0), b_1(0), ..., b_{32}(0))$$

is copied to the registers $R_1$ and $R_2$ in the following way. The bit $a_1(0)$ corresponds to the 1-st bit of $R_1$, the bit $a_2(0)$ corresponds to the 2-nd bit of $R_1$, and so on. The bit $a_{32}(0)$ corresponds to the 32-nd bit of $R_1$. The word $(b_1(0), \ldots, b_{32}(0))$ is copied in the same way to the register $R_2$. Thus

$$R_1 = (a_{32}(0), ..., a_1(0)) \text{ and } R_2 = (b_{32}(0), ..., b_1(0))$$

The 256-bit key $K$ is stored in the KSU and

$$\begin{aligned} K_0 &= (k_{32}, \ldots, k_1) \\ K_1 &= (k_{64}, \ldots, k_{33}) \\ &\vdots \\ K_7 &= (k_{256}, \ldots, k_{225}) \end{aligned}$$

Encryption using the algorithm consists of 32 iterations. In the first iteration, the initial value of register $R_1$ is added to the partial key $K_0$ modulo $2^{32}$. This is done by the adder $CM_1$. The value of $R_1$ is not changed. The result of modulo addition is sent to the S-boxes (the block $S$). The output from the S-boxes is put into the shift register $R$ where the contents is rotated 11 bits left (towards high-order bits). The contents of $R$ is now added bitwise Exclusive-Or to the contents of $R_2$ by the adder $CM_2$. The output from $CM_2$ is stored in $R_1$ and the old value of $R_1$ is stored in $R_2$. This concludes the first iteration.

The other iterations are similar to the first one. In the second iteration, we use the partial key $K_1$ from the KSU. The iterations 3,4,5,6,7,8 apply partial keys $K_2, K_3, K_4, K_5, K_6, K_8$, respectively. Iterations from 9 to 16 and from 17 to 24 use the same partial keys. The iterations from 25 to 32 apply the reverse order of partial keys so the 25-th iteration uses the key $K_7$, the 26-th iteration - the key $K_6$ and so on. The last iteration uses the key $K_0$. So the order of partial keys in the 32 iterations is as follows:

$$K_0, \ldots, K_7, K_0, \ldots, K_7, K_0, \ldots, K_7, K_7, \ldots, K_0$$

After 32 iterations the output from the adder $CM_2$ is in $R_2$ and $R_1$ keeps its previous value. The contents of the registers $R_1$ and $R_2$ are the 64-bit ciphertext (cryptogram) for the 64-bit plaintext.

The equations of the simple substitution mode are the following:

$$\begin{aligned} a(j) &= ((a(j-1) \boxed{+} K_{(j-1) \pmod 8}) SR \oplus b(j-1) \\ b(j) &= a(j-1) \end{aligned}$$

for $j = 1, \ldots, 24$ and

$$\begin{aligned} a(j) &= ((a(j-1) \boxed{+} K_{32-j}) SR \oplus b(j-1) \\ b(j) &= a(j-1) \end{aligned}$$

for $j = 25, \ldots, 31$. Finally, the ciphertext is (for $j = 32$):

$$\begin{aligned} a(32) &= a(31) \\ b(32) &= ((a(31) \boxed{+} K_0) SR \oplus b(31) \end{aligned}$$

Where $a(0) = (a_{32}(0), \ldots, a_1(0))$ and $b(0) = (b_{32}(0), \ldots, b_1(0))$ are the initial contents of $R_1$ and $R_2$, respectively. Vectors $a(j) = (a_{32}(j), \ldots, a_1(j))$ and $b(j) = (b_{32}(j), \ldots, b_1(j))$ are the contents of registers $R_1$ and $R_2$, respectively, after the $j$-th iteration of encryption. The signs $\oplus$ and $\boxed{+}$ stand for the bitwise Exclusive-Or addition and the sum modulo $2^{32}$, respectively. R denotes rotation a of 11 bits to the left.

The ciphertext block is $T_e = (a_1(32), ..., a_{32}(32), b_1(32), ..., b_{32}(32))$.

### 3.2 Decryption in the mode of simple substitution

The decryption algorithm is the same as for the encryption (see Figure 2). The 64-bit cryptogram is stored in two registers $R_1$ and $R_2$. The order of the bits is $T_e = (a_1(32), ..., a_{32}(32), b_1(32), ..., b_{32}(32))$.

The decryption applies the 256-bit cryptographic key stored in the KSU in the reverse order. Thus the order of partial keys for 32 iterations is as follows:

$$K_0, \ldots, K_7, K_7, \ldots, K_0, K_7, \ldots, K_0, K_7, \ldots, K_0$$

The equations for the decryption are:

$$\begin{aligned} a(32-j) &= ((a(32-j+1) \boxed{+} K_{(j-1) \pmod 8}) SR \oplus b(32-j+1) \\ b(32-j) &= a(32-j+1) \end{aligned}$$

for $j = 1, \ldots, 8$ and

$$\begin{aligned} a(32-j) &= ((a(32-j+1) \boxed{+} K_{(32-j) \pmod 8}) SR \oplus b(32-j+1) \\ b(32-j) &= a(32-j+1) \end{aligned}$$

for $j = 9, \ldots, 31$. Finally, $(a(0), b(0))$ is (for $j = 32$):

$$\begin{aligned} a(0) &= a(1) \\ b(0) &= ((a(1) \boxed{+} K_0) SR \oplus b(1) \end{aligned}$$

The registers $R_1$ and $R_2$ contain the plaintext block

$$T_0 = (a_1(0), ..., a_{32}(0), b_1(0), ..., B_{32}(0))$$

The encryption algorithm in the mode of simple substitution of the 64 bits of block $T_0$ is denoted by E, so $E(T_0) = E(a(0), b(0)) = (a(32), b(32)) = T_e$

## 4 Stream mode ($\Gamma$-mode)

### 4.1 Encryption in the stream mode

A cryptosystem which implements encryption in the stream mode is shown in Figure 3. The plaintext is first divided into 64-bit blocks

$$T_0^{(1)}, T_0^{(2)}, \ldots, T_0^{(M)}$$

So the plaintext consists of $M$ 64-bit blocks. Next they are encrypted using the adder $CM_5$ by adding them bitwise Exclusive-Or to the 64-bit running-key blocks

$$\Gamma_e = (\Gamma_e^{(1)}, ..., \Gamma_e^{(M)})$$

If the number of bits in the last plaintext block $T_0^{(M)}$ is less than 64, then we use the same number of bits from the last block of $\Gamma_e^{(M)}$ and ignore the rest of the bits.

The cryptographic key (256-bit long) is stored in the KSU. To generate the sequence of blocks $\Gamma_e$, the registers $R_1$ and $R_2$ are initialized to some value $IC = (IC_1, \ldots, IC_{64})$. The sequence $IC$ is the initial condition of the cryptosystem. Next the simple substitution mode

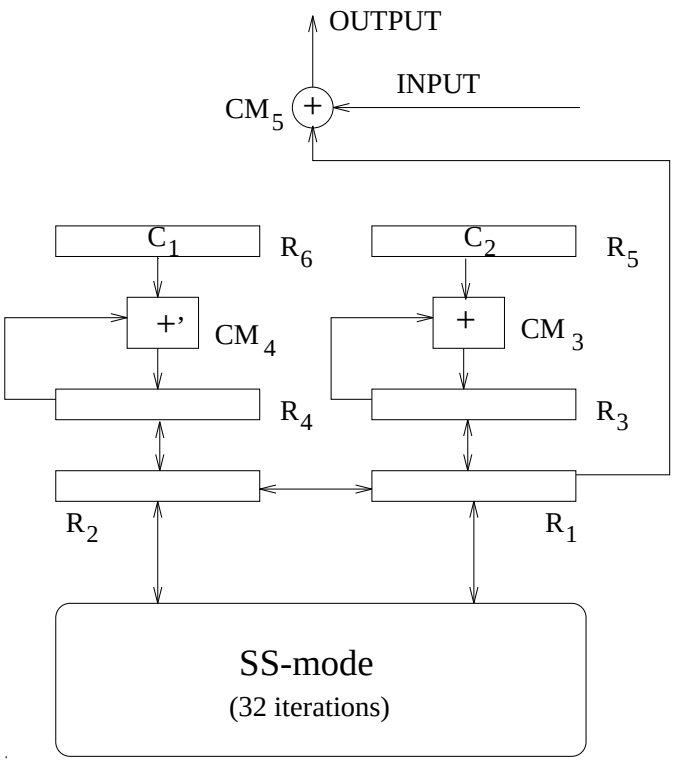


Figure 3: The encryption/decryption in the stream mode

is used to encrypt the initial condition $IC$. So $E(IC) = (Y_0, Z_0)$. We store the encrypted values $(Y_0, Z_0)$ in registers $R_3$ and $R_4$, respectively (so $R_3 = R_1$, $R_4 = R_2$).

Registers $R_6, R_5$ contain the two constants $C_1, C_2$, respectively. We add modulo $2^{32} - 1$ the contents of $R_4$ to the contents of $R_6$ in $CM_4$ and store the result back in $R_4$. We add the contents of $R_3$ and $R_5$ modulo $2^{32}$ in the adder $CM_3$ and the result is kept in $R_3$. Then we copy the contents of $R_3, R_4$ to $R_1, R_2$. Next $R_1$ and $R_2$ is encrypted in the simple substitution mode. The output (which is stored in the registers $R_1$,$R_2$) is the first block of the running-key $\Gamma_e$. The block is added bitwise Exclusive-Or to the first plaintext block in the adder $CM_5$. The result is the first block of the ciphertext $T_e^{(1)} = \Gamma_e^{(1)} \oplus T_0^{(1)}$.

To obtain the next 64 bit block of the cipher in the stream mode, the contents of $R_4$ and $R_6$ (the constant $C_1$) are added modulo $2^{32} - 1$ in the adder $CM_4$ and the contents of $R_3$ and $R_5$ are added modulo $2^{32}$ in $CM_3$. The contents of $R_3, R_4$ are copied to $R_1, R_2$, respectively. The contents of $(R_1, R_2)$ are encrypted in the SS mode. The ciphertext is the second block of the running-key which is added bitwise Exclusive-Or to the second block of the plaintext, i.e. $T_e^{(2)} = \Gamma_e^{(2)} \oplus T_0^{(2)}$. The result is the second block of the ciphertext. The process of encryption continues in the same way for the rest of the blocks.

The initial condition $IC$ and the sequence of ciphertext blocks

$$T_e^{(1)}, \ldots, T_e^{(M)}$$

are transmitted via public channels or computer networks.

The equations of the encryption in the stream mode are as follows:

$$T_e^{(i)} = E(Y_{i-1} \boxed{+} C_2, Z_{i-1} \boxed{+'} C_1) \oplus T_0^{(i)} = \Gamma_e^{(i)} \oplus T_0^{(i)}$$

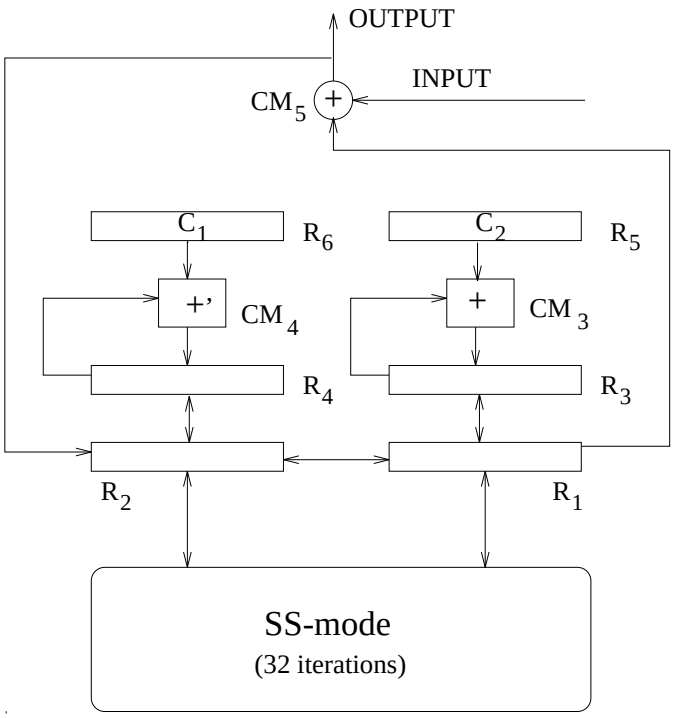


Figure 4: The encryption/decryption in the stream mode with feedback

where $\boxed{+'}$ denotes addition modulo $2^{32}-1$, $(Y_{i-1}, Z_{i-1})$ is the contents of the registers $R_1, R_2$, respectively for the $i$-th block and $i = 1, \ldots, M$.

### 4.2 Decryption of data in the stream mode

The decryption algorithm is the same as the encryption. The same 256-bit cryptographic key (stored in the KSU) is used as for the encryption. The same initial condition $IC$ must also be used. To decrypt data, the same running key $\Gamma_e$. has to be generated. If we add the running-key blocks to the ciphertext blocks, we will get the plaintext blocks back.

The decryption is described by the following equation:

$$T_0^{(i)} = E(Y_{i-1}\boxed{+}C_2, Z_{i-1}\boxed{+'}C_1) \oplus T_e^{(i)} = \Gamma_e^{(i)} \oplus T_e^{(i)}$$

where $i = 1, \ldots, M$.

## 5 The stream mode with feedback

### 5.1 Encryption

This mode is graphically illustrated in Figure 4. The plaintext is divided into 64-bit blocks $T_0^{(1)}, \ldots, T_0^{(M)}$ and encrypted by adding bitwise Exclusive-Or to the running key $\Gamma_e = (\Gamma_e^{(1)}, \ldots, \Gamma_e^{(M)})$ obtained as in the stream mode where $M$ is determined by the length of the plaintext. The 256-bit cryptographic key is stored in the KSU and the initial condition $IC$ is in registers $R_1$, $R_2$ as in the stream mode.

The contents of registers $R_1$ and $R_2$ are encrypted in the SS mode. The result is the first block of the running-key $\Gamma_e^{(1)} = E(IC)$. It is bitwise Exclusive-Or'd to the first 64-bit block of the plaintext in the adder $CM5$. The result is the first encrypted block of data, $T_e^{(1)} = (t_1^{(1)}, \ldots, t_{64}^{(1)}$. This block becomes the contents of registers $R_1$ and $R_2$ for the next encryption.

The block $T_e^{(1)}$ (stored in $R_1$ and $R_2$) gets encrypted in the SS mode and the result is the second block of the running-key. It is added bitwise Exclusive-Or to the second block of the plaintext and the sum becomes the second block of the ciphertext. The encryption proceeds in the same way for all the blocks.

If the number of bits of the last block of the plaintext is less than 64, the corresponding number of bits of the running-key is applied.

The equations for the encryption are:

$$\begin{aligned} T_e^{(1)} &= E(IC) \oplus T_0^{(1)} = \Gamma_e^{(1)} \oplus T_0^{(1)} \\ T_e^{(i)} &= E(T_e^{(i-1)}) \oplus T_0^{(i)} = \Gamma_e^{(i)} \oplus T_0^{(i)} \end{aligned}$$

for $i = 2, \ldots, M$.

The sequence of encrypted blocks $T_e^{(1)}, \ldots, T_e^{(M)}$ and the initial condition $IC$ are transmitted via a public channel to the receiver.

### 5.2 Decryption

The decryption is the inverse operation to the encryption. It applies the same cryptosystem as for encryption (it generates the same running key). The same 256-bit cryptographic key that was used for the encryption is stored in the KSU. The initial condition $IC$ is stored in registers $R_1$, $R_2$.

The contents of $R_1$ and $R_2$ are encrypted in the SS mode and the result is the first block of the running key $\Gamma_e^{(1)} = E(IC)$. If we add bitwise Exclusive-Or the first ciphertext block $T_e^{(1)}$ to $\Gamma_e^{(1)}$, we get the first plaintext block $T_0^{(1)}$.

The block $T_e^{(1)}$ becomes the contents of the registers $R_1$ and $R_2$ for the second cycle. The contents of $R_1$ and $R_2$ is encrypted in the SS mode and the result is bitwise Exclusive-Or'd to the second block of the ciphertext. The output of this is the second block of the plaintext. The decryption process goes on in the same way for the rest of the blocks.

The decryption is described by the following equations:

$$\begin{aligned} T_0^{(1)} &= E(IC) \oplus T_e^{(1)} = \Gamma_e^{(1)} \oplus T_e^{(1)} \\ T_0^{(i)} &= E(T_e^{(i-1)}) \oplus T_e^{(i)} = \Gamma_e^{(i)} \oplus T_e^{(i)} \end{aligned}$$

for $i = 2, \ldots, M$.

## 6 Authentication mode

$M$ 64-bit blocks of the plaintext ($M \geq 2$) are authenticated by producing an additional block. The process of producing this additional block is the same for all encryption modes.

The first block of the plaintext $T_0^{(1)} = (t_1^{(1)}, ..., t_{64}^{(1)})$ is stored in the registers $R_1$ and $R_2$. The first bit of the block corresponds to the first bit of $R_1$ and the last bit of the block to the last bit of $R_2$.

The contents of the registers $R_1$ and $R_2$ are encrypted using the first 16 iterations of the SS mode. The contents of the registers $R_1$ and $R_2$ are:

$$(a_1^{(1)}(16),...,a_{32}^{(1)}(16),b_1^{(1)}(16),...,b_{32}^{(1)}(16)$$

The KSU has the same key thta was used in the encryption. After 16 iterations the contents of registers $R_1$ and $R_2$ are added bitwise Exclusive-Or to the second block of the plaintext $T_0^{(2)} = (t_1^{(2)},...,t_{32}^{(2)})$ (in the adder $CM5$). The result is

$$(a_1^{(1)}(16) \oplus t_1^{(2)},\ldots,a_{32}^{(1)}(16) \oplus t_{32}^{(2)},b_1^{(1)}(16) \oplus t_{33}^{(2)},\ldots,b_{32}^{(1)}(16) \oplus t_{64}^{(2)}) = (a_1^{(2)}(0),\ldots,a_{32}^{(2)}(0),b_1^{(2)}(0),\ldots,b_{32}^{(2)}(0))$$

and it is stored in $R_1$ and $R_2$. The contents of $(R_1, R_2)$ are encrypted using the first 16 cycles of the SS mode. The result of the encryption is added bitwise Exclusive-Or to the third block of the plaintext.

The last block of the plaintext is extended to 64-bits by padding the missing part with zero bits. The resultant vector is

$$(a_1^{(M)}(0),\ldots,a_{32}^{(M)}(0),b_1^{(M)}(0),\ldots,b_{32}^{(M)}(0))$$

and is stored in the registers $R_1$ and $R_2$. Next it is encrypted using the first 16 iterations of the SS mode.

The authentication block $A_\ell$ of the plaintext is a part of the vector $(a_1^{(M)}(16),\ldots,a_{32}^{(M)}(16),b_1^{(M)}(16),\ldots,b_{32}^{(M)}($ of the length $\ell$ bits. Thus

$$A_\ell = (a_{32-\ell+1}^{(M)}(16),a_{32-\ell+2}^{(M)}(16),\ldots,a_{32}^{(M)}(16))$$

The authentication block $A_\ell$ is appended to the ciphertext blocks.

On the receiver's side, the ciphertext is decrypted and the authentication block $A_\ell$ is regenerated as described above. If the authentication block recreated is different from the one received then the ciphertext received is considered to be not genuine. The authentication block is usually produced before encryption on the sender's side and after decryption of all ciphertext blocks on the receiver's side. The first blocks of the plaintext can contain information like the address, the timestamp and the initial condition S and must not be encrypted if the header information is to be used by the computer network. The length of the authentication block $\ell$ (number of bits in the authentication block) is determined by security requirements. It is assumed that the probability of the enemy's success when he or she wants to produce the correct authentication block for an illegal ciphertext, is equal to $2^{-\ell}$.

*ACKNOWLEDGMENT*

*We would like to thank Andrew Odlyzko from AT&T Bell Laboratories for sending us the original Russian GOST documentation. Also we thank Colin Spargo and Jennie Seberry for proof-reading of this report.*

## References

[1] National Soviet Bureau of Standards. Information Processing Systems. Cryptographic Protection. Cryptographic Algorithm. GOST 28147-89, 1989.