Abstract

This document is intended to be a source of information about the Russian Federal standard hash function (GOST R 34.11-94), which is one of the Russian cryptographic standard algorithms (called GOST algorithms). Recently, Russian cryptography is being used in Internet applications, and this document has been created as information for developers and users of GOST R 34.11-94 for hash computation.
1. Introduction

1.1. General information

1. GOST R 34.11-94 was developed by the Federal Agency for Government Communication and Information and by the All-Russia Scientific and Research Institute of Standardization.

2. GOST R 34.11-94 was accepted and activated by the Act 154 of 23.05.1994 issued by the Russian federal committee for standards.

1.2. The purpose of GOST R 34.11-94

Expanding application of information technologies when creating, processing and storing documents requires in some cases confidentiality of their contents, maintenance of completeness and authenticity.

Cryptography (cryptographic security) is one of the effective approaches for data security. It is widely applied in different areas of government and commercial activity.

Cryptographic data security methods are under serious scientific research and standardization efforts at national, regional and international levels.

GOST R 34.11-94 defines a hash function calculation procedure for an arbitrary sequence of binary symbols.

The hash function maps an arbitrary set of data represented as a sequence of binary symbols onto its image of a fixed small length.
Thus hash function can be used in procedures related to the electronic digital signature, resulting in considerable reduction of elapsed time for sign and verify stages. The effect of reduction of time is due to the fact that only a short image of initial data is actually signed.

2. Applicability

GOST R 34.11-94 defines algorithm and procedure for calculation of a hash function for an arbitrary sequence of binary symbols. These algorithm and procedure should be applied in cryptographic methods of data processing and securing, including digital signature procedures employed for data transfer and data storage in computer-aided systems.

The hash function, defined in GOST R 34.11-94, is used for digital signature systems based on the asymmetric cryptographic algorithm according to GOST R 34.10-2001 (see section 3).

3. Conventions Used in This Document

The following notations are used in GOST R 34.11-94:

- \( V_{all} \) is a set of all finite words in alphabet \( V = \{0,1\} \). The words reading and alphabet symbols numbering are performed right to left (rightmost symbol of the word has number one, second right symbol has number two etc.).

- \( V_k \) is a set of all words in alphabet \( V = \{0,1\} \) of length \( k \) bits (\( k=16,64,256 \)).

- \( |A| \) is a length of a word \( A \) belonging to \( V_{all} \).

- \( A||B \) is concatenation of words \( A, B \) belonging to \( V_{all} \). It is a word of length \( |A| + |B| \), where the left \( |A| \) symbols come from the word \( A \), and the right \( |B| \) symbols come from the word \( B \). One can also use a notation \( A||B = A * B \).

- \( A^k \) is a concatenation of \( k \) copies of the word \( A \) (\( A \) belongs to \( V_{all} \)).

- \( <N>_k \) is a word of length \( k \), containing a binary representation of \( N(mod 2^k) \) residue, with a non-negative integer \( N \).

- \( A^@$ \) is a non-negative integer with \( A \) as its binary representation.

- \((\text{xor})\) is the bitwise modulo 2 addition of the words of the same length.

- \((+)'\) is the addition according to the rule \( A (+)' B = <A^@$ + B^@$>_k \), where \( k = |A| = |B| \).
M is a binary sequence to be hashed, M belongs to V_all. M is a message in digital signature systems.

h is a hash-function which maps the sequence M belonging to V_all onto the word h(M) belonging to V_256.

E(k,A) is a result of encryption of the word A using key K with the encryption algorithm according to [GOST28147] in the electronic codebook (ECB) mode (K belongs to V256, A belongs to V64).

h0 is an initial hash value.

e := g is assignment of the value g to the parameter e.

^ is the power operator.

i = 1..8 is i being all values in interval from 1 to 8.

hUZ is the S-boxes described in [GOST28147].

4. General statements

A hash-function h is the mapping h : V_all -> V256, depending on the parameter (which is the initial hash value H, H is a word from V256). To define the hash-function it is necessary to have:

- A calculation algorithm for the step-by-step hash function

  \[ \chi : V256 \times V256 \rightarrow V256. \]

- A description of an iterative procedure for calculating the hash value h.

A hash function h depends on two parameters h0 and hUZ.

5. Step-by-step hash function

A calculation algorithm for the step-by-step hash function contains three parts, which successively do:

- keys generation, here keys are 256 bit long words;

- an encryption transformation, that is encryption of 64-bit subwords of word H using keys K[i], (i = 1, 2, 3, 4) with the algorithm according to [GOST28147] in ECB mode;

- a mixing transformation for the result of the encryption.

5.1 Key generation

Consider \( X = (b[256], b[255], \ldots, b[1]) \) belonging to V256.
Let
\[ = \xi[32]||\xi[31]||...||\xi[1], \]
where
\[ x[i] = (b[i*64],...,b[(i-1)*64+1]) \text{ belonging to } V_{64}, \quad i = 1..4, \]
\[ \eta[j] = (b[j*16],...,b[(j-1)*16+1]) \text{ belonging to } V_{16}, \quad j = 1..16, \]
\[ \xi[k] = (b[k*8],...,b[(k-1)*8+1]) \text{ belonging to } V_8, \quad k = 1..32; \]

The transformation \( P : V_{256} \to V_{256} \) maps the word \( x_{i32}||...||x_{i1} \) onto the word \( x_{i[\phi(32)]]} || ... || x_{i[\phi(1)]} \),
where \( \phi(i + 1 + 4 \cdot (k - 1)) = 8i + k, \quad i = 0..3, \quad k = 1..8. \)

For the key generation one should use the following initial data:
- words \( H, M \) belonging to \( V_{256} \),
- parameters: words \( C[i] \) (i = 2, 3, 4), with values:
\[ C[3] = 1^8||0^8||1^16||0^24||1^16||0^8||(0^8||1^8)^2||1^8||0^8\]
\[ ||(0^8||1^8)^4||(1^8||0^8)^4 . \]

The following algorithm is used for the key calculation:

1. Assign values:
\[ i := 1, \quad U := H \quad , \quad V := M . \]

2. Calculate:
\[ W = U \text{ (xor) } V, \quad K[i] = P(W). \]

3. Assign
\[ i := i + 1. \]

4. Verify condition
\[ i = 5. \]
If it is true, go to step 7. If not, go to step 5.

5. Calculate:
\[ U := A(U) \text{ (xor) } C[i], \quad V := A(A(V)), \]
\[ W := U \text{ (xor) } V, \quad K[i] = P(W). \]

6. Go to step 3.

7. End.
5.2. Encryption transformation

At this stage 64-bit subwords of the word $H$ are encrypted using keys $K[i]$ ($i = 1, 2, 3, 4$).

For the encryption transformation one should use the following initial data:


where $h[i]$ belong to V64, $i = 1,2,3,4$, and a key set is $K[1], K[2], K[3], K[4]$.

The encryption algorithm is applied and the following words are obtained

$$s[i] = E(K[i], h[i]), \text{ where: } i = 1,2,3,4$$

As a result of the stage the following sequence is formed


5.3. Mixing transformation

At this stage the obtained sequence is mixed using a shift register.

The initial data include words $H$, $M$ belonging to V256 and a word $S$ belonging to V256.

Let a mapping $PSI(X) : V256(2) \rightarrow V256(2)$ transform the word

$$\eta[16]||\eta[15]||...||\eta[1], \eta[i] \text{ belongs to V16, } i = 1..16$$

into the word

$$\eta[1](xor)\eta[2](xor)\eta[3](xor)\eta[4](xor)\eta[13](xor)\eta[16]||\eta[16]||...||\eta[2].$$

Then the value of the step-by-step hash function value is the word:

$$\chi(M, H) = PSI^61(H(xor)PSI(M(xor)PSI^12(S))), \text{ where } PSI^i(X) \text{ is the transformation } PSI \text{ applied } i \text{ times to } X.$$

6. The calculation procedure for a hash function

The calculation procedure for a hash function $h$ is assumed to be applied to a sequence $M$ belonging to V_all. Its parameter is an initial hash value $h0$ which is an arbitrarily fixed word from V256.

The calculation procedure for the function $h$ uses the following quantities at each step of iteration:

- $M$ belonging to V_all - a part of the sequence $M$, which was not hashed at previous iterations;
H belonging to V256 - the current hash value;

SIGMA belonging to V256 - the current check sum value;

L belonging to V256 - the length of the partial sequence M processed at the previous iteration step.

The calculation algorithm for function h consists of the following steps:

Step 1. Assign initial values to current quantities

1.1 _M_ := M.
1.2 H := h0.
1.3 SIGMA := 0^256.
1.4 L := 0^256.
1.5 Go to step 2.

Step 2.

2.1 Verify the condition |_M_|>256.

If it is true go to step 3.

Else make the following calculations:

2.2 L := <L^$ + |M|>_256
2.3 M’ := 0^(256 -|M|)||M
2.4 SIGMA := SIGMA (+)’ M’
2.5 H := chi (M’, H)
2.6 H := chi (L, H)
2.7 H := chi (SIGMA, H)
2.8 End.

Step 3.

3.1 Calculate a subword M_s belonging to V256 of the word _M_ (_M_ = M_p||M_s). Then make the following calculations:

3.2 H := chi (M_s, H)
3.3 L := <L^$ + 256>_256
3.4 \( \sigma := \sigma + M[s] \)

3.5 \( M_\_ = M_p \)

3.6 Go to step 2.

The quantity \( H \) obtained at step 2.7 is the value of the hash function \( h(M) \).

7. Test examples (Informative)

It is recommended to use the values for substitution units \( \pi[1], \pi[2],..., \pi[8] \) and the initial hash value \( H \) described in this appendix for the GOST R 34.11-94 test examples only.

7.1 Usage of the algorithm GOST 28147-89

The algorithm GOST 28147-89 [GOST28147] in ECB mode is used as an encryption transformation in the following examples. The following values of the substitution units \( \pi[1], \pi[2],..., \pi[8] \) have been chosen:

\[
\begin{array}{cccccccc}
8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\
0 & 1 & D & 4 & 6 & 7 & 5 & E & 4 \\
1 & F & B & B & C & D & 8 & B & A \\
2 & D & 4 & A & 7 & A & 1 & 4 & 9 \\
3 & 0 & 1 & 0 & 1 & 1 & D & C & 2 \\
4 & 5 & 3 & 7 & 5 & 0 & A & 6 & D \\
5 & 7 & F & 2 & F & 8 & 3 & D & 8 \\
6 & A & 5 & 1 & D & 9 & 4 & F & 0 \\
7 & 4 & 9 & D & 8 & F & 2 & A & E \\
8 & 9 & 0 & 3 & 4 & E & E & 2 & 6 \\
9 & 2 & A & 6 & A & 4 & F & 3 & B \\
10 & 3 & E & 8 & 9 & 6 & C & 8 & 1 \\
11 & E & 7 & 5 & E & C & 7 & 1 & C \\
12 & 6 & 6 & 9 & 0 & B & 6 & 0 & 7 \\
13 & B & 8 & C & 3 & 2 & 0 & 7 & F \\
14 & 8 & 2 & F & B & 5 & 9 & 5 & 5 \\
15 & C & C & E & 2 & 3 & B & 9 & 3 \\
\end{array}
\]
The hexadecimal value of $\pi[j](i)$ is given in a column number $j$, $j = 1..8$, and in a row number $i$, $i = 0..15$.

### 7.2 Representation of vectors

We will put down binary symbol sequences as hexadecimal digits strings, where each digit corresponds to four signs of its binary representation.

### 7.3 Examples of the hash value calculation

A zero vector, for example, can be taken as an initial hash value:

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

### 7.3.1 Hash calculation for the sample message $M$

\[
M = 73657479 62203233 3D687467 6E656C20 2C656761 7373656D 20736920 73696854
\]

Initial values are assigned for text:

\[
\begin{array}{cccccccc}
_M_ = 73657479 62203233 3D687467 6E656C20 2C656761 7373656D 20736920 73696854 \\
\end{array}
\]

for hash-function:

\[
\begin{array}{cccccccc}
H = 00000000 00000000 00000000 00000000 \\
00000000 00000000 00000000 00000000 \\
\end{array}
\]

for the sum of text blocks:

\[
\begin{array}{cccccccc}
SIGMA = 00000000 00000000 00000000 00000000 \\
00000000 00000000 00000000 00000000 \\
\end{array}
\]

for the length of the text:

\[
\begin{array}{cccccccc}
L = 00000000 00000000 00000000 00000000 \\
00000000 00000000 00000000 00000000 \\
\end{array}
\]

As a length of the message to be hashed equals 256 bits (32 bytes), then

\[
L = 00000000 00000000 00000000 00000000 \\
00000000 00000000 00000000 00001000 \\
\]

\[
M' = _M_ = 73657479 62203233 3D687467 6E656C20 2C656761 7373656D 20736920 73696854
\]

and there is no need to pad the current block with zeroes,
SIGMA=M’ = 73657479 62203233 3D687467 6E656C20 2C656766 7373656D 20736920 73696854

The step-by-step hash function chi(M, N) values are calculated.

The keys are generated:

K[1] = 733D2C20 65686573 74746769 326C6568 62673737 20673736 79676120 3206D54
K[2] = 110C733D 0D166568 130E7474 06417967 1006266E 161A2065 090D326C 4D393320
K[3] = 80B111F3 730DF216 850013F1 C7E1F941 620C1DDF 3ABAE91A 3FA109F2 F513B239
K[4] = AOE2804E FF1B73F2 ECE27A00 E7B8C7E1 EE1D620C AC0C5BA A804C05E A18B0AEC

The 64-bit subwords of block H are encrypted by the algorithm according to GOST 28147.

Block h[1] = 00000000 00000000 is encrypted using key K[1] and s[1] = 42ABBCCE 32BC0B1B is obtained.


Block h[3] = 00000000 00000000 is encrypted using key K[3] and s[3] = 8D345899 00FF0E28 is obtained.


So S = E7860419 0D2A562D 8D345899 00FF0E28 5203EBC8 5D9BCFFD 42ABBCCB 32BC0B1B

is obtained.

The mixing transformation using a shift register is performed and

KSI = chi(M, H) = CF9A8C65 505967A4 68A03B8C 42DE7624 D99C4124 883DA687 561C7DE3 3315C034

is obtained.

Assign H = KSI and calculate chi(L, H):

K[1] = CF68D956 9A09C1C8 8C3B417D 658C24E3 50428833 59DE3D15 6776A6C1 A4248734
K[2] = 8FCF68D9 809AA09C3C8C3B41 C7658C24 BB504288 2859DE3D 666676A6 B3A42487
Now assign $H = KSI$ again and calculate \( \chi(\Sigma, H) \):

\[
\begin{align*}
K[1] &= 5817F104 \ 0BD45D84 \ B6522F27 \ 4AF5B00B \\
&\quad A531B57A \ 9C8FDFFCA \ BB1E7C6 \ D7A517A3 \\
K[2] &= E82759E0 \ C278D950 \ 15CC523C \ FC72EBB6 \\
&\quad D2C73DA8 \ 19A6CACA \ 3E8440F5 \ C0DDB65A \\
K[3] &= 77483AD9 \ F7C29CAA \ EB06D1D7 \ 841BCAD3 \\
&\quad FBC3DAA0 \ 7CB555F0 \ D4968080 \ 0A9E56BC \\
K[4] &= A1157965 \ 2D9FC9C \ 088C7CC2 \ 46FB3DD2 \\
&\quad 7684ADC0 \ 44ACA06 \ 53EFF7D7 \ C0748708 \\
S &= 2AEF7A6 \ A85FB57D \ 0F164DE9 \ 2951A581 \\
&\quad C31E7435 \ 4930FD05 \ 1F8A492 \ 550A582D \\
KSI &= FAFF37A6 \ 15A81669 \ 1CFF3B8 \ B68CA247 \\
&\quad E09525F3 \ 9F811983 \ 2EB81975 \ D366C4B1 \\
\end{align*}
\]

Then the hash result is

\[
\begin{align*}
H &= FAFF37A6 \ 15A81669 \ 1CFF3B8 \ B68CA247 \\
&\quad E09525F3 \ 9F811983 \ 2EB81975 \ D366C4B1 \\
\end{align*}
\]

7.3.2 Hash calculation for the sample message $M$

Let $M = 7365 \ 74796220 \ 3035203D \ 20687467 \ 6E656C20 \\
73616820 \ 65676173 \ 73656D20 \ 6C616E69 \\
6769726F \ 20656874 \ 2065736F \ 70707553$

As the length of the message to be hashed equals 400 bits
(50 bytes), the message is divided into two blocks, and the second
(high-order) one is padded with zeroes. During the calculations the
following numbers are obtained:
STEP 1.

\[ H = \begin{array}{cccccccc}
 00000000 & 00000000 & 00000000 & 00000000 \\
 00000000 & 00000000 & 00000000 & 00000000 \\
\end{array} \]

\[ M_s = \begin{array}{cccccccc}
 73616820 & 65676173 & 73656D20 & 6C616E69 \\
 6769726F & 20656874 & 2065736F & 70707553 \\
\end{array} \]

\[ K[1] = \begin{array}{cccccccc}
 73736720 & 61656965 & 686D7273 & 20656874 \\
 656C2070 & 67616570 & 616E6875 & 73697453 \\
\end{array} \]

\[ K[2] = \begin{array}{cccccccc}
 14477373 & 0C0C6165 & 1F01686D & 4F002020 \\
 4C506566 & 04156761 & 061D616E & 1D277369 \\
\end{array} \]

\[ K[3] = \begin{array}{cccccccc}
 CBFF14B8 & 6D04F30C & 96051FFE & DFFB000 \\
 35094CAF & 72F9FB15 & 7CF006E2 & AB1AE227 \\
\end{array} \]

\[ K[4] = \begin{array}{cccccccc}
 EBACCB00 & F7006DFB & E5E16905 & B0B0DFFF \\
 BA1C3509 & FD118DF9 & F51B830F & F8C554E5 \\
\end{array} \]

\[ S = \begin{array}{cccccccc}
 FF41797C & EAAADAC2 & 43C9B1DF & 2E14681C \\
 EDDC2210 & 1E11ADF9 & FA67E757 & DAFE3AD9 \\
\end{array} \]

\[ KSI = \begin{array}{cccccccc}
 F0CEEA4E & 368B5A60 & C63D96C1 & E5B51CD2 \\
 A93BEFB0 & 2634F0AD & CBBCB9CE & ED2D5D9A \\
\end{array} \]

STEP 2.

\[ H = \begin{array}{cccccccc}
 F0CEEA4E & 368B5A60 & C63D96C1 & E5B51CD2 \\
 A93BEFB0 & 2634F0AD & CBBCB9CE & ED2D5D9A \\
\end{array} \]

\[ M' = \begin{array}{cccccccc}
 00000000 & 00000000 & 00000000 & 00007365 \\
 74796220 & 3035203D & 20687467 & 6E656C20 \\
\end{array} \]

\[ K[1] = \begin{array}{cccccccc}
 F0C6DEB6 & CE3D42D3 & EA968D1D & 4EC19DA9 \\
 36E51683 & 8BB50148 & 5A6FD031 & 60B790BA \\
\end{array} \]

\[ K[2] = \begin{array}{cccccccc}
 16A4C6A9 & F9DF3D3B & E4FC96EF & 5309C1BD \\
 FB68E526 & 2CD8534 & 1EE1AD97 & 6F7DD2C8 \\
\end{array} \]

\[ K[3] = \begin{array}{cccccccc}
 C49D846D & 1780482C & 9086887F & C48C9186 \\
 9D5CB0644 & D1E641E5 & A02109AF & 9D52C7CF \\
\end{array} \]
K[4] = BDB0C9F0 756E9131 E1F290EA 50E4CBB1 1CAD9536 F4E4B674 99F31E29 70C52AFA
S = 62A07EA5 EF3C3309 2CE1B076 173D48CC 6881EB66 F5C7959F 63FCA1F1 D33C31B8
KSI = 95BEA0BE 88D5AA02 FE3C9D45 436CE821 B8287CB6 2CBC135B 3E339EFE F6576CA9

STEP 3.
H = 95BEA0BE 88D5AA02 FE3C9D45 436CE821 B8287CB6 2CBC135B 3E339EFE F6576CA9
L = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000190
K[1] = 95FEB83E BE3C2833 A09D7C9E BE45B6FE 88432CF6 D56CBC57 AAE8136D 02215B39
K[2] = 8695FEB8 1BBE3C28 E2A09D7C 48BE45B6 DA88432C EBD56CBC 7FA8B813 F292215B
K[2] = 8695FEB8 1BBE3C28 E2A09D7C 48BE45B6 DA88432C EBD56CBC 7FA8B813 F292215B
K[3] = B9799501 141B413C 1EE2A062 0CB74145 6FDA88BC DO142A6C FA80AA16 15F2FDB1
K[4] = 94B97995 7D141B41 C21EE2A0 040CB741 346FDA88 46D0142A BDFA81AA DC1562FD
S = D42336E0 2A0A6998 6C65478A 3D08A1B9 9FDDFF20 4808E863 94FD9D6D F776A7AD
KSI = 47E26AFD 3E7278A1 7D4737B5 06140773 A3D97E7E A744CB43 08AA4C24 3352C745

STEP 4.
H = 47E26AFD 3E7278A1 7D4737B5 06140773 A3D97E7E A744CB43 08AA4C24 3352C745
SIGMA = 73616820 65676173 73656D20 6C61E1CE DBE2D48F 509A88B1 40CDE7D6 DED5E173
K[1] = 340E7848 83223B67 025AAAAB DDA5F1F2 5B6AF7ED 1575D5B8 19E64326 D2BDF236
K[2] = 03DC0ED0 F4CD26BC 8B595F13 F5A4A55E A8B063CB ED3D73C5 6511662A 7963008D
K[3] = C954EF19 D0779A68 ED37D3FB 7DA5ADDC 4A9D0277 78EF765B C4731191 7EBB21B1
Then the hash result is

H = 0852F562 3B89DD57 AEB4781F E54DF14E
    EAFBC135 0613763A 0D770AA6 57BA1A47

8. Security considerations

This entire document is about security considerations.

Current cryptographic resistance of GOST R 34.11-94 hash algorithm is estimated as $2^{128}$ operations of computations of step hash function. (There is known method to reduce this estimate to $2^{105}$ operations, but it demands padding the colliding message with 1024 random bit blocks each of 256 bit length, thus it cannot be used in any practical implementation).

9. IANA Considerations

This document has no actions for IANA.

10. Normative references


Authors’ Addresses

Vasily Dolmatov, Ed.
Cryptocom Ltd.
Kedrova st., 14, bld.2
Moscow, 117218, Russian Federation

EMail: dol@cryptocom.ru

Dmitry Kabelev
Cryptocom Ltd.
Kedrova st., 14, bld.2
Moscow, 117218, Russian Federation
EMail: kdb@cryptocom.ru
Igor Ustinov
Cryptocom Ltd.
Kedrova st., 14, bld.2
Moscow, 117218, Russian Federation
EMail: igus@cryptocom.ru

Sergey Vyshensky
Moscow State University
Leninskiye gory, 1
Moscow, 119991, Russian Federation
EMail: svysh@pn.sinp.msu.ru