Internet-Draft
Intended status: Informational
Expires: June 21, 2010

V. Dolmatov, Ed.
Cryptocom Ltd.

December 21, 2009

GOST R 34.11-94
Hash function algorithm
draft-dolmatov-cryptocom-gost341194-07

Status of This Memo

Copyright Notice

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.

Table of Contents

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.).

Vk 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 work 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.

(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 x V256 -> 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], ..., b[1]) belonging to V256.

Let
   X = x[4]||x[3]||x[2]||x[1] = eta[16]||[eta15]||...||eta[1]

     = xi[32]||xi[31]||...||xi[1], where

     x[i] = (b[i*64],...,b[(i-1)*64+1]) belonging to V64, i = 1..4,

   eta[j] = (b[j*16],...,b[(j-1)*16+1]) belonging to V16, j = 1..16,

    xi[k] = (b[k*8],..., b[(k-1)*8+1]) belonging to V8, k = 1..32;

Yet another notation: A(X) = (x[1](xor)x[2])||x[4]||x[3]||x[2].

The transformation P : V256 -> V256 maps the word xi32||...||xi1
onto the word xi[phi(32)] || ... || xi[phi(1)],

where phi(i + 1 + 4 (k - 1)) = 8i + k , i = 0..3, k = 1..8.

For the key generation one should use the following initial data:

- words H, M belonging to V256 ,

- parameters: words C[i] (i = 2, 3, 4) , with values:

     C[2] = C[4] = 0^256;

     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, U := H , V := M .

2. Calculate:

     W = U (xor) V , 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)(xor)C[i], V := A(A(V)),
     W := U(xor)V, 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:

H = h[4]||h[3]||h[2]||h[1],

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]), where: i = 1,2,3,4

As a result of the stage the following sequence is formed

S = s[4]||s[3]||s[2]||s[1].

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) -> V256(2) transform the word

eta[16]||eta[15]||...||eta[1], eta[i] 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))) , where PSI^i(X) is the transformation PSI applied i 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:

	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

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:

```
h0 = 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000
```

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:

```
_M_   = 73657479 62203233 3D687467 6E656C20
        2C656761 7373656D 20736920 73696854
```

for hash-function:

```
H   = 00000000 00000000 00000000 00000000
      00000000 00000000 00000000 00000000
```

for the sum of text blocks:

```
SIGMA = 00000000 00000000 00000000 00000000
        00000000 00000000 00000000 00000000
```

for the length of the text:

```
L = 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000
```

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

```
L = 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000100
```

```
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
2C656761 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
626E7373 20657369 79676120 33206D54

K[2] = 110C733D 0D166568 130E7474 06417967
1D00626E 161A2065 090D326C 4D393320

K[3] = 80B111F3 730DF216 850013F1 C7E1F941
620C1DFF 3ABAE91A 3FA109F2 F513B239

K[4] = A0E2804E FF1B73F2 ECE27A00 E7B8C7E1
EE1D620C AC0CC5BA 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[2] = 00000000 00000000 is encrypted using key K[2] and
s[2] = 5203EBC8 5D9BCFFD is obtained.

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

Block h[4] = 00000000 00000000 is encrypted using key K[4] and
s[4] = E7860419 0D2A562D is obtained.

So S = E7860419 0D2A562D 8D345899 00FF0E28
5203EBC8 5D9BCFFD 42ABBCCE 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 9AA09C1C 8C3B417D 658C24E3
50428833 59DE3D15 6776A6C1 A4248734

K[2] = 8FCF68D9 809AA09C 3C8C3B41 C7658C24

```
K[3] = 4E70CF97 3C8065A0 853C8CC4 57389A8C
       CABB50BD E3D7A6DE D1996788 5CB35B24

K[4] = 584E70CF C53C8065 48853C8C 1657389A
       EDCABB50 78E3D7A6 EED19867 7F5CB35B

S    = 66B70F5E F163F461 468A9528 61D60593
       E5EC8A37 3FD42279 3CD1602D DD783E86

KSI  = 2B6EC233 C7BC89E4 2ABC2692 5FEA7285
       DD3848D1 C6AC997A 24F74E2B 09A3AEF7
```

Now assign H = KSI again and calculate chi(SIGMA, H):

```
K[1] = 5817F104 0BD45D84 B6522F27 4AF5B00B
       A531B57A 9C8FDFCA BB1EFCC6 D7A517A3

K[2] = E82759E0 C278D950 15CC523C FC72EBB6
       D2C73DA8 19A6CAC9 3E8440F5 C0DDB65A

K[3] = 77483AD9 F7C29CAA EB06D1D7 841BCAD3
       FBC3DAA0 7CB555F0 D4968080 0A9E56BC

K[4] = A1157965 2D9FBC9C 088C7CC2 46FB3DD2
       7684ADCB FA4ACA06 53EFF7D7 C0748708

S    = 2AEBFA76 A85FB57D 6F164DE9 2951A581
       C31E7435 4930FD05 1F8A4942 550A582D

KSI  = FAFF37A6 15A81669 1CFF3EF8 B68CA247
       E09525F3 9F811983 2EB81975 D366C4B1
```

Then the hash result is

```
H    = FAFF37A6 15A81669 1CFF3EF8 B68CA247
       E09525F3 9F811983 2EB81975 D366C4B1
```

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    = 00000000 00000000 00000000 00000000
          00000000 00000000 00000000 00000000

   M_s = 73616820 65676173 73656D20 6C616E69
         6769726F 20656874 2065736F 70707553

   K[1] = 73736720 61656965 686D7273 20206F6F
          656C2070 67616570 616E6875 73697453

   K[2] = 14477373 0C0C6165 1F01686D 4F002020
          4C50656C 04156761 061D616E 1D277369

   K[3] = CBFF14B8 6D04F30C 96051FFE DFFFB000
          35094CAF 72F9FB15 7CF006E2 AB1AE227

   K[4] = EBACCB00 F7006DFB E5E16905 B0B0DFFF
          BA1C3509 FD118DF9 F61B830F F8C554E5

   S    = FF41797C EEAADAC2 43C9B1DF 2E14681C
          EDDC2210 1EE1ADF9 FA67E757 DAFE3AD9

   KSI  = F0CEEA4E 368B5A60 C63D96C1 E5B51CD2
          A93BEFBD 2634F0AD CBBB69CE ED2D5D9A

   STEP 2.

   H    = F0CEEA4E 368B5A60 C63D96C1 E5B51CD2
          A93BEFBD 2634F0AD CBBB69CE ED2D5D9A

   M'   = 00000000 00000000 00000000 00007365
          74796220 3035203D 20687467 6E656C20

   K[1] = F0C6DDEB CE3D42D3 EA968D1D 4EC19DA9
          36E51683 8BB50148 5A6FD031 60B790BA

   K[2] = 16A4C6A9 F9DF3D3B E4FC96EF 5309C1BD
          FB68E526 2CDBB534 FE161C83 6F7DD2C8

   K[3] = C49D846D 1780482C 9086887F C48C9186
          9DCB0644 D1E641E5 A02109AF 9D52C7CF
```

```
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 00000190

K[1] = 95FEB83E BE3C2833 A09D7C9E BE45B6FE
       88432CF6 D56CBC57 AAE8136D 02215B39

K[2] = 8695FEB8 1BBE3C28 E2A09D7C 48BE45B6
       DA88432C EBD56CBC 7FABE813 F292215B

K[2] = 8695FEB8 1BBE3C28 E2A09D7C 48BE45B6
       DA88432C EBD56CBC 7FABE813 F292215B

K[3] = B9799501 141B413C 1EE2A062 0CB74145
       6FDA88BC D0142A6C FA80AA16 15F2FDB1

K[4] = 94B97995 7D141B41 C21EE2A0 040CB741
       346FDA88 46D0142A BDFA81AA DC1562FD

S    = D42336E0 2A0A6998 6C65478A 3D08A1B9
       9FDDFF20 4808E863 94FD9D6D F776A7AD

KSI  = 47E26AFD 3E7278A1 7D473785 06140773
       A3D97E7E A744CB43 08AA4C24 3352C745

STEP 4.

H    = 47E26AFD 3E7278A1 7D473785 06140773
       A3D97E7E A744CB43 08AA4C24 3352C745

SIGMA = 73616820 65676173 73656D20 6C61E1CE
        DBE2D48F 509A88B1 40CDE7D6 DED5E173

K[1] = 340E7848 83223B67 025AAAAB DDA5F1F2
       5B6AF7ED 1575DE87 19E64326 D2BDF236

K[2] = 03DC0ED0 F4CD26BC 8B595F13 F5A4A55E
       A8B063CB ED3D7325 6511662A 7963008D
```

```
   K[3] = C954EF19 D0779A68 ED37D3FB 7DA5ADDC
          4A9D0277 78EF765B C4731191 7EBB21B1
```

```
K[4] = 6D12BC47 D9363D19 1E3C696F 28F2DC02
       F2137F37 64E4C18B 69CCFBF8 EF72B7E3

S    = 790DD7A1 066544EA 2829563C 3C39D781
       25EF9645 EE2C05DD A5ECAD92 2511A4D1

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

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

[GOST28147] "Cryptographic Protection for Data Processing System", GOST 28147-89, Gosudarstvennyi Standard of USSR, Government Committee of the USSR for Standards, 1989. (In Russian)

[GOST3411] "Information technology. Cryptographic Data Security. Hashing function.", GOST R 34.10-94, Gosudarstvennyi Standard of Russian Federation, Government Committee of the Russia for Standards, 1994. (In Russian)

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
Leninskie gory, 1
Moscow, 119991, Russian Federation

EMail: svysh@pn.sinp.msu.ru