Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 23, 2016

S. Smyshlyaev, Ed.
E. Alekseev
I. Oshkin
V. Popov
S. Leontiev
CRYPTO-PRO
V. Podobaev
FACTOR-TS
D. Belyavsky
TCI
November 20, 2015

Guidelines on the Cryptographic Algorithms, Accompanying the Usage of Standards GOST R 34.10-2012 and GOST R 34.11-2012
draft-smyshlyaev-gost-usage-17

Abstract

The purpose of this document is to make the specifications of the cryptographic algorithms defined by the Russian national standards GOST R 34.10-2012 and GOST R 34.11-2012 available to the Internet community for their implementation in the cryptographic protocols based on the accompanying algorithms.

These specifications define the pseudorandom functions, the key agreement algorithm based on the Diffie-Hellman algorithm and a hash function, the parameters of elliptic curves, the key derivation functions and the key export functions.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The accompanying algorithms are intended for the cryptographic protocols implementation. This memo contains a description of the accompanying algorithms based on the Russian national standards GOST R 34.10-2012 [GOST3410-2012] and GOST R 34.11-2012 [GOST3411-2012].

The English versions of these standards can be found in [RFC7091] and [RFC6986]; the English version of the encryption standard GOST 28147-89 [GOST28147-89] (which is used in the key export functions) can be found in [RFC5830].

The specifications of algorithms and parameters proposed in this memo are provided on the basis of experience in the development of the cryptographic protocols, as described in [RFC4357], [RFC4490] and [RFC4491].

This memo describes the pseudorandom functions, the key agreement algorithm based on the Diffie-Hellman algorithm and a hash function, the parameters of elliptic curves, the key derivation functions, and the key export functions necessary to ensure interoperability of security protocols that make use of the Russian cryptographic standards GOST R 34.10-2012 [GOST3410-2012] digital signature algorithm and GOST R 34.11-2012 [GOST3411-2012] cryptographic hash function.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Basic terms, definitions and notations

This document uses the following terms and definitions for the sets and operations on the elements of these sets:

(xor)   exclusive-or of two binary vectors of the same length;

V_n   the finite vector space over GF(2) of dimension n, n >= 0, with the (xor) operation; for n = 0 the V_0 space consists of a single empty element of size 0;
if U is an element of V_n, then U = (u_(n-1), u_(n-2), ..., u_1, u_0), where u_i in {0, 1};

V_(8, r) the set of byte vectors of size r, r >= 0, for r = 0 the V_(8, r) set consists of a single empty element of size 0; if W is an element of V_(8, r), r > 0, then W = (w^0, w^1, ..., w^(r-1)), where w^0, w^1, ..., w^(r-1) are elements of V_8;

Bit representation the bit representation of the element W = (w^0, w^1, ..., w^(r-1)) of V_(8, r) is an element (w_(8r-1), w_(8r-2), ..., w_1, w_0) of V_(8*r), where w^0 = (w_7, w_6, ..., w_0), w^1 = (w_15, w_14, ..., w_8), ..., w^(r-1) = (w_(8r-1), w_(8r-2), ..., w_(8r-8)) are elements of V_8;

Byte representation if n is a multiple of 8, r = n/8, then the byte representation of the element W = (w_(n-1), w_(n-2), ..., w_0) of V_n is a byte vector (w^0, w^1, ..., w^(r-1)) of V_(8, r), where w^0 = (w_7, w_6, ..., w_0), w^1 = (w_15, w_14, ..., w_8), ..., w^(r-1) = (w_(8r-1), w_(8r-2), ..., w_(8r-8)) are elements of V_8;

A|B   concatenation of byte vectors A and B, i.e., if A in V_(8, r1), B in V_(8, r2), A = (a^0, a^1, ..., a^(r1-1)) and B = (b^0, b^1, ..., b^(r2-1)), then A|B = (a^0, a^1, ..., a^(r1-1), b^0, b^1, ..., b^(r2-1)) is an element of V_(8, r1+r2);

K (key) an arbitrary element of V_n; if K in V_n, then its size (in bits) is equal to n, where n can be an arbitrary natural number.

This memo uses the following abbreviations and symbols:

Symbols	Meaning
H_256	GOST R 34.11-2012 hash function with 256-bit output
H_512	GOST R 34.11-2012 hash function with 512-bit output
HMAC	a function for calculating a message authentication code, based on a hash function in accordance with [RFC2104]
PRF	a pseudorandom function, i.e., a transformation that allows to generate pseudorandom sequence of bytes
KDF	a key derivation function, i.e., a transformation that allows to derive keys and keying material from the root key and additional input using a pseudorandom function
VKO	a key agreement algorithm based on the Diffie-Hellman algorithm and a hash function.

To generate a byte sequence of the size r with functions that give a longer output the output is truncated to the first r bytes. This remark applies to the following functions:

- the functions described in Section 4.2;
- KDF_TREE_GOSTR3411_2012_256 described in Section 4.4;

- o KDF_GOSTR3411_2012_256 described in Section 4.5.

Hereinafter all data are provided in byte representation unless otherwise specified.

If a function is defined outside this document (e.g., H_256) and its definition requires arguments in bit representation, it is assumed that the bit representations of the arguments are formed immediately before the calculation of the function (in particular, immediately after the application of the operation (|) to the byte representation of the arguments).

If the output of another function defined outside of this document is used as an argument of the functions defined below and it has the bit representation then it is assumed that an output MUST have length that is a multiple of 8 and that it will be translated into the byte representation in advance.

When an elliptic point is given to an input of a hash function, affine coordinates for short Weierstrass form are used (see Section 5): an x coordinate value is fed first, an y coordinate value is fed second, both in little-endian format.

4. Algorithm descriptions

4.1. HMAC functions

This section defines the HMAC transformations based on the GOST R 34.11-2012 [GOST3411-2012] algorithm.

4.1.1. HMAC_GOSTR3411_2012_256

This HMAC transformation is based on the GOST R 34.11-2012 [GOST3411-2012] hash function with 256-bit output. The object identifier of this transformation is shown below:

```
id-tc26-hmac-gost-3411-12-256::= {iso(1) member-body(2) ru(643)
rosstandart(7) tc26(1) algorithms(1) mac(4) hmac-gost-
3411-12-256(1)}.
```

This algorithm uses H_256 as a hash function for HMAC, described in [RFC2104]. The method of forming the values of ipad and opad is also specified in [RFC2104]. The size of HMAC_GOSTR3411_2012_256 output is equal to 32 bytes, the block size of the iterative procedure for the H_256 compression function is equal to 64 bytes (in the notation of [RFC2104], L = 32 and B = 64, respectively).

4.1.2. HMAC_GOSTR3411_2012_512

This HMAC transformation is based on the GOST R 34.11-2012 [GOST3411-2012] hash function with 512-bit output. The object identifier of this transformation is shown below:

```
id-tc26-hmac-gost-3411-12-512::= {iso(1) member-body(2) ru(643)
rosstandart(7) tc26(1) algorithms(1) mac(4) hmac-gost-
3411-12-512(2)}.
```

This algorithm uses H_512 as a hash function for HMAC, described in [RFC2104]. The method of forming the values of ipad and opad is also specified in [RFC2104]. The size of HMAC_GOSTR3411_2012_512 output is equal to 64 bytes, the block size of the iterative procedure for the H_512 compression function is equal to 64 bytes (in the notation of [RFC2104], L = 64 and B = 64, respectively).

4.2. Pseudorandom functions

This section defines six HMAC-based PRF transformations recommended for usage. Two of them are designed for the TLS protocol and four are designed for the IPsec protocol.

4.2.1. PRFs for the TLS protocol

4.2.1.1. PRF_TLS_GOSTR3411_2012_256

This is the transformation providing the pseudorandom function for the TLS protocol (1.0 and higher versions) in accordance with GOST R 34.11-2012 [GOST3411-2012]. It uses the P_GOSTR3411_2012_256 function that is similar to the P_hash function defined in Section 5 of [RFC5246], where HMAC_GOSTR3411_2012_256 function (defined in Section 4.1.1 of this document) is used as the HMAC_hash function.

```
PRF_TLS_GOSTR3411_2012_256 (secret, label, seed) =
= P_GOSTR3411_2012_256 (secret, label | seed).
```

Label and seed values MUST be assigned by a protocol, their lengths SHOULD be fixed by a protocol in order to avoid possible collisions.

4.2.1.2. PRF_TLS_GOSTR3411_2012_512

This is the transformation providing the pseudorandom function for the TLS protocol (1.0 and higher versions) in accordance with GOST R 34.11-2012 [GOST3411-2012]. It uses the P_GOSTR3411_2012_512 function that is similar to the P_hash function defined in Section 5 of [RFC5246], where HMAC_GOSTR3411_2012_512 function (defined in Section 4.1.2 of this document) is used as the HMAC_hash function.

```
PRF_TLS_GOSTR3411_2012_512 (secret, label, seed) =
= P_GOSTR3411_2012_512 (secret, label | seed).
```

Label and seed values MUST be assigned by a protocol, their lengths SHOULD be fixed by a protocol in order to avoid possible collisions.

4.2.2. PRFs for the IPsec protocols based on GOST R 34.11-2012

IPsec family protocols use the pseudorandom functions for the purposes of keying material generation and authentication. The specifications for the version 1 (IKEv1) [RFC2409] and version 2 (IKEv2) [RFC7296] of the Internet Key Exchange protocol define the usage of PRF in various parts of the protocols.

4.2.2.1. PRF in the IKEv1 protocol

According to the Section 4 of [RFC2409], if a PRF is not negotiated, the HMAC based on the negotiated hash algorithm is used. So, when the 256-bit version of GOST R 34.11-2012 [GOST3411-2012] is used as a hash function in IKEv1, HMAC_GOST3411_2012_256 is used as a PRF; when the 512-version of GOST R 34.11-2012 [GOST3411-2012] is used as a hash function in IKEv1, HMAC_GOST3411_2012_512 is used as a PRF.

4.2.2.2. PRF in the IKEv2 protocol

IKEv2 has no default PRF. This document specifies that HMAC_GOST3411_2012_256 and HMAC_GOST3411_2012_512 may be used as a PRF for the IKEv2 protocol.

4.3. VKO algorithms for key agreement

This section specifies the key agreement algorithms based on GOST R 34.10-2012 [GOST3410-2012].

4.3.1. VKO_GOSTR3410_2012_256

The VKO_GOSTR3410_2012_256 transformation is used for agreement of 256-bit keys and is based on the 256-bit version of GOST R 34.11-2012 [GOST3411-2012]. This algorithm can be applied for a key agreement using GOST R 34.10-2012 [GOST3410-2012] with 256-bit or 512-bit private keys.

The algorithm is designed to produce an encryption key or a keying material of size 256 bits to be used in various cryptographic protocols. A key or a keying material KEK_VKO (x, y, UKM) is produced from the private key x of one side, the public key y*P of the opposite side and the UKM value, considered as an integer.

The algorithm can be used for static and ephemeral keys with the public key size n >= 512 bits including the case where one side uses a static key and the other uses an ephemeral one.

The UKM parameter is optional (the default UKM = 1) and can take any integer value from 1 to 2^(n/2)-1. It is allowed to use a nonzero UKM of an arbitrary size not exceeding n/2 bits. If at least one of the parties uses static keys, the RECOMMENDED length of UKM is 64 bits or more.

KEK_VKO (x, y, UKM) is calculated using the formulas

    KEK_VKO (x, y, UKM) = H_256 (K (x, y, UKM)),

    K (x, y, UKM) = (m/q*UKM*x mod q)*(y*P),

where m and q are the parameters of an elliptic curve defined in the GOST R 34.10-2012 [GOST3411-2012] standard (m is an elliptic curve points group order, q is an order of a cyclic subgroup), P is a nonzero point of the subgroup; P is defined by a protocol.

This algorithm is defined similar to the one specified in Section 5.2 of [RFC4357], but applies the hash function H_256 instead of the hash function GOST R 34.11-94 [GOST3411-94] (referred as gostR3411). In addition, K(x, y, UKM) is calculated with public key size n >= 512 bits and UKM has a size up to n/2 bits.

4.3.2. VKO_GOSTR3410_2012_512

The VKO_GOSTR3410_2012_512 transformation is used for agreement of 512-bit keys and is based on the 512-bit version of GOST R 34.11-2012 [GOST3411-2012]. This algorithm can be applied for a key agreement using GOST R 34.10-2012 [GOST3410-2012] with 512-bit private keys.

The algorithm is designed to produce an encryption key or a keying material of size 512 bits to be used in various cryptographic protocols. A key or a keying material KEK_VKO (x, y, UKM) is produced from the private key x of one side, the public key y*P of the opposite side and the UKM value, considered as an integer.

The algorithm can be used for static and ephemeral keys with the public key size n >= 1024 bits including the case where one side uses a static key and the other uses an ephemeral one.

The UKM parameter is optional (the default UKM = 1) and can take any integer value from 1 to 2^(n/2)-1. It is allowed to use a nonzero UKM of an arbitrary size not exceeding n/2 bits. If at least one of the parties uses static keys, the RECOMMENDED length of UKM is 128 bits or more.

KEK_VKO (x, y, UKM) is calculated using the formulas

```
KEK_VKO (x, y, UKM) = H_512 (K (x, y, UKM)),

K (x, y, UKM) = (m/q*UKM*x mod q)*(y*P),
```

where m and q are the parameters of an elliptic curve defined in the GOST R 34.10-2012 [GOST3411-2012] standard (m is an elliptic curve points group order, q is an order of a cyclic subgroup), P is a nonzero point of the subgroup; P is defined by a protocol.

This algorithm is defined similar to the one specified in Section 5.2 of [RFC4357], but applies the hash function H_512 instead of the hash function GOST R 34.11-94 [GOST3411-94] (referred as gostR3411). In addition, K(x, y, UKM) is calculated with public key size n >= 1024 bits and UKM has a size up to n/2 bits.

4.4. The key derivation function KDF_TREE_GOSTR3411_2012_256

The key derivation function KDF_TREE_GOSTR3411_2012_256 based on the HMAC_GOSTR3411_2012_256 function is given by:

```
KDF_TREE_GOSTR3411_2012_256 (K_in, label, seed, R) = K(1) | K(2) |
K(3) | K(4) |...,

K(i) = HMAC_GOSTR3411_2012_256 (K_in, [i]_b | label | 0x00 |
seed | [L]_b), i >= 1,
```

where

K_in   derivation key;

label, seed the parameters that MUST be assigned by a protocol, their lengths SHOULD be fixed by a protocol;

R   a fixed external parameter, with possible values of 1, 2, 3 or 4;

i   iteration counter;

[i]_b   byte representation of the iteration counter (in the network byte order), the number of bytes in the representation [i]_b is equal to R (no more than 4 bytes);

L   the required size (in bits) of the generated keying material (an integer, not exceeding 256*(2^(8*R)-1));

[L]_b   byte representation of L, in network byte order (variable length: no leading zero bytes added).

The key derivation function KDF_TREE_GOSTR3411_2012_256 is intended for generating a keying material of size L, not exceeding 256*(2^(8*R)-1) bits, and utilizes general principles of the input and output for the key derivation function outlined in Section 5.1 of NIST SP 800-108 [NISTSP800-108]. The HMAC_GOSTR3411_2012_256 algorithm described in Section 4.1.1 is selected as a pseudorandom function.

Each key derived from the keying material formed using the derivation key K_in (0-level key) may be a 1-level derivation key and may be used to generate a new keying material. The keying material derived from the 1-level derivation key can be split down into the 2nd level derivation keys. The application of this procedure leads to the construction of the key tree with the root key and the formation of the keying material to the hierarchy of the levels, as described in Section 6 of NIST SP 800-108 [NISTSP800-108]. The partitioning procedure for keying material at each level is defined in accordance with a specific protocol.

4.5. The key derivation function KDF_GOSTR3411_2012_256

The KDF_GOSTR3411_2012_256 function is equivalent to the function KDF_TREE_GOSTR3411_2012_256, when R = 1, L = 256, and is given by:

```
KDF_GOSTR3411_2012_256 (K_in, label, seed) =
HMAC_GOSTR3411_2012_256 (K_in, 0x01 | label | 0x00 | seed | 0x01 |
0x00),
```

where

K_in   derivation key,

label, seed the parameters that MUST be assigned by a protocol, their lengths SHOULD be fixed by a protocol.

4.6. Key wrap and key unwrap

Wrapped representation of a secret key K (256-bit GOST 28147-89 [GOST28147-89] key, 256-bit or 512-bit GOST R 34.10-2012 [GOST3410-2012] private key) is formed as follows by using a given export key K_e (GOST 28147-89 [GOST28147-89] key) and a random seed vector:

1. Generate a random seed vector from 8 up to 16 bytes.

2. With the key derivation function, using an export key K_e as a derivation key, produce a key KEK_e (K_e, seed), where

   KEK_e (K_e, seed) = KDF_GOSTR3411_2012_256 (K_e, label, seed),

   where the KDF_GOSTR3411_2012_256 function (see Section 4.5) is used as a key derivation function for the fixed label value

   label = (0x26 | 0xBD | 0xB8 | 0x78).

3. GOST 28147-89 [GOST28147-89] MAC value (4-byte) for the data K and the key KEK_e (K_e, seed) is calculated, initialization vector (IV) in this case is equal to the first 8 bytes of seed. The resulting value is denoted as CEK_MAC.

4. The key K is encrypted with the GOST 28147-89 [GOST28147-89] algorithm in the Electronic Codebook (ECB) mode with the key KEK_e (K_e, seed). The result is denoted as CEK_ENC.

5. The wrapped representation of the key is (seed | CEK_ENC | CEK_MAC).

The value of key K is restored from the wrapped representation of the key and the export key K_e as follows:

1. Obtain the seed, CEK_ENC and CEK_MAC values from the wrapped representation of the key.

2. With the key derivation function, using the export key K_e as a derivation key, produce a key KEK_e(K_e, seed), where

   KEK_e (K_e, seed) = KDF_GOSTR3411_2012_256 (K_e, label, seed),

   where the KDF_GOSTR3411_2012_256 function (see section Section 4.5) is used as a key derivation function for the fixed label value

   label = (0x26 | 0xBD | 0xB8 | 0x78).

3. The CEK_ENC field is decrypted with the GOST 28147-89 [GOST28147-89] algorithm in the Electronic Codebook (ECB) mode with the key KEK_e(K_e, seed). The unwrapped key K is assumed to be equal to the result of decryption.

4. GOST 28147-89 [GOST28147-89] MAC value (4-byte) for the data K and the key KEK_e(K_e, seed) is calculated, initialization vector

(IV) in this case is equal to the first 8 bytes of seed. If the result is not equal to CEK_MAC, an error is returned.

The GOST 28147-89 [GOST28147-89] algorithm is used with the parameter set defined in Appendix C of this document.

5. The parameters of elliptic curves

This section defines the elliptic curves parameters and object identifiers that are RECOMMENDED for the usage with the signature and verification algorithms of the digital signature in accordance with the GOST R 34.10-2012 [GOST3410-2012] standard and with the key agreement algorithms VKO_GOSTR3410_2012_256 and VKO_GOSTR3410_2012_512.

This document does not negate the use of other parameters of elliptic curves.

5.1. Canonical form

This section defines the elliptic curves parameters of the GOST R 34.10-2012 [GOST3410-2012] standard for the case of elliptic curves with prime 512-bit moduli in canonical (short Weierstrass) form, that is given by the following equation defined in GOST R 34.10-2012 [GOST3410-2012]:

$$y^2 = x^3 + ax + b \pmod p.$$

In case of elliptic curves with 256-bit prime moduli the parameters defined in [RFC4357] are proposed to use.

5.1.1. Parameters and object identifiers

The parameters for each elliptic curve are represented by the following values which are defined in GOST R 34.10-2012 [GOST3410-2012]:

p the characteristic of the underlying prime field;

a, b the coefficients of the equation of the elliptic curve in the canonical form;

m the elliptic curve group order;

q the elliptic curve subgroup order;

(x, y) the coordinates of the point P (generator of the subgroup of order q) of the elliptic curve in the canonical form.

Both sets of the parameters are presented as ASN structures of the form:

```
SEQUENCE {
p    INTEGER,
a    INTEGER,
b    INTEGER,
m    INTEGER,
q    INTEGER,
x    INTEGER,
y    INTEGER
}
```

The parameter sets have the following object identifiers:

1. id-tc26-gost-3410-12-512-paramSetA::= {iso(1) member-body(2) ru(643) rosstandart(7) tc26(1) constants(2) sign-constants(1) gost-3410-12-512-constants(2) paramSetA(1)};

2. id-tc26-gost-3410-12-512-paramSetB::= {iso(1) member-body(2) ru(643) rosstandart(7) tc26(1) constants(2) sign-constants(1) gost-3410-12-512-constants(2) paramSetB(2)}.

The corresponding values of the parameter sets can be found in Appendix A.1.

5.2. Twisted Edwards form

This section defines the elliptic curves parameters and object identifiers of the GOST R 34.10-2012 [GOST3410-2012] standard for the case of elliptic curves that have a representation in the Twisted Edwards form with prime 256-bit and 512-bit moduli.

A Twisted Edwards curve E over a finite prime field F_p, p > 3, is an elliptic curve defined by the equation:

e*u^2 + v^2 = 1 + d*u^2*v^2 (mod p),

where e, d are in F_p, ed(e-d) != 0.

A Twisted Edwards curve has an equivalent representation in the short Weierstrass form defined by parameters a, b. The parameters a, b, e and d are related as follows:

a = s^2 - 3*t^2 (mod p),
b = 2*t^3 - t*s^2 (mod p),

where

```
s = (e - d)/4 (mod p),
t = (e + d)/6 (mod p).
```

Coordinate transformations are defined as follows:

```
(u,v) --> (x,y) = (s(1 + v)/(1 - v) + t, s(1 + v)/((1 - v)u)),
(x,y) --> (u,v) = ((x - t)/y, (x - t - s)/(x - t + s)).
```

5.2.1. Parameters and object identifiers

The parameters for each elliptic curve are represented by the following values which are defined in GOST R 34.10-2012 [GOST3410-2012]:

p — the characteristic of the underlying prime field;

a, b — the coefficients of the equation of the elliptic curve in the canonical form;

e, d — the coefficients of the equation of the elliptic curve in the Twisted Edwards form;

m — the elliptic curve group order;

q — the elliptic curve subgroup order;

(x, y) — the coordinates of the point P (generator of the subgroup of order q) of the elliptic curve in the canonical form;

(u, v) — the coordinates of the point P (generator of the subgroup of order q) of the elliptic curve in the Twisted Edwards form.

Both sets of the parameters are presented as ASN structures of the form:

```
SEQUENCE {
p       INTEGER,
a       INTEGER,
b       INTEGER,
e       INTEGER,
d       INTEGER,
m       INTEGER,
q       INTEGER,
x       INTEGER,
y       INTEGER,
u       INTEGER,
v       INTEGER
}
```

The parameter sets have the following object identifiers:

1. id-tc26-gost-3410-2012-256-paramSetA ::= {iso(1) member-body(2) ru(643) rosstandart(7) tc26(1) constants(2) sign-constants(1) gost-3410-12-256-constants(1) paramSetA(1)};

2. id-tc26-gost-3410-2012-512-paramSetC ::= {iso(1) member-body(2) ru(643) rosstandart(7) tc26(1) constants(2) sign-constants(1) gost-3410-12-512-constants(2) paramSetC(3)}.

The corresponding values of the parameter sets can be found in Appendix A.2.

6. Acknowledgments

We thank Valery Smyslov, Igor Ustinov, Basil Dolmatov, Russ Housley, Dmitry Khovratovich, Oleksandr Kazymyrov, Ekaterina Smyshlyaeva, Vasily Nikolaev and Lolita Sonina for their careful readings and useful comments.

7. Security Considerations

This entire document is about security considerations.

8. References

8.1. Normative References

[GOST28147-89]
Gosudarstvennyi Standard of USSR, Government Committee of the USSR for Standards (In Russian), "Systems of information processing. Cryptographic data security. Algorithms of cryptographic transformation", GOST 28147-89, 1989.

[GOST3410-2012]
Federal Agency on Technical Regulating and Metrology (In Russian), "Information technology. Cryptographic data security. Signature and verification processes of [electronic] digital signature", GOST R 34.10-2012, 2012.

[GOST3411-2012]
Federal Agency on Technical Regulating and Metrology (In Russian), "Information technology. Cryptographic Data Security. Hashing function", GOST R 34.11-2012, 2012.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <http://www.rfc-editor.org/info/rfc2104>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998, <http://www.rfc-editor.org/info/rfc2409>.

[RFC4357] Popov, V., Kurepkin, I., and S. Leontiev, "Additional Cryptographic Algorithms for Use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms", RFC 4357, DOI 10.17487/RFC4357, January 2006, <http://www.rfc-editor.org/info/rfc4357>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <http://www.rfc-editor.org/info/rfc7296>.

8.2. Informative References

[GOST3411-94] Federal Agency on Technical Regulating and Metrology (In Russian), "Information technology. Cryptographic Data Security. Hashing function", GOST R 34.11-94, 1994.

[NISTSP800-108] National Institute of Standards and Technology, "Recommendation for Key Derivation Using Pseudorandom Functions", NIST SP 800-108, October 2009.

[RFC4490] Leontiev, S., Ed. and G. Chudov, Ed., "Using the GOST 28147-89, GOST R 34.11-94, GOST R 34.10-94, and GOST R 34.10-2001 Algorithms with Cryptographic Message Syntax (CMS)", RFC 4490, DOI 10.17487/RFC4490, May 2006, <http://www.rfc-editor.org/info/rfc4490>.

[RFC4491] Leontiev, S., Ed. and D. Shefanovski, Ed., "Using the GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms with the Internet X.509 Public Key Infrastructure Certificate and CRL Profile", RFC 4491, DOI 10.17487/RFC4491, May 2006, <http://www.rfc-editor.org/info/rfc4491>.

[RFC5830] Dolmatov, V., Ed., "GOST 28147-89: Encryption, Decryption, and Message Authentication Code (MAC) Algorithms", RFC 5830, DOI 10.17487/RFC5830, March 2010, <http://www.rfc-editor.org/info/rfc5830>.

[RFC6986] Dolmatov, V., Ed. and A. Degtyarev, "GOST R 34.11-2012: Hash Function", RFC 6986, DOI 10.17487/RFC6986, August 2013, <http://www.rfc-editor.org/info/rfc6986>.

[RFC7091] Dolmatov, V., Ed. and A. Degtyarev, "GOST R 34.10-2012: Digital Signature Algorithm", RFC 7091, DOI 10.17487/RFC7091, December 2013, <http://www.rfc-editor.org/info/rfc7091>.

Appendix A. Values of the parameter sets

A.1. Canonical form parameters

Parameter set: id-tc26-gost-3410-12-512-paramSetA

```
SEQUENCE
{
    OBJECT IDENTIFIER
    id-tc26-gost-3410-12-512-paramSetA
    SEQUENCE
    {
     INTEGER
      00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FD
      C7
     INTEGER
      00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FD
      C4
     INTEGER
      00 E8 C2 50 5D ED FC 86 DD C1 BD 0B 2B 66 67 F1
      DA 34 B8 25 74 76 1C B0 E8 79 BD 08 1C FD 0B 62
```

```
         65 EE 3C B0 90 F3 0D 27 61 4C B4 57 40 10 DA 90
         DD 86 2E F9 D4 EB EE 47 61 50 31 90 78 5A 71 C7
         60
        INTEGER
         00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
         FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
         FF 27 E6 95 32 F4 8D 89 11 6F F2 2B 8D 4E 05 60
         60 9B 4B 38 AB FA D2 B8 5D CA CD B1 41 1F 10 B2
         75
        INTEGER
         00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
         FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
         FF 27 E6 95 32 F4 8D 89 11 6F F2 2B 8D 4E 05 60
         60 9B 4B 38 AB FA D2 B8 5D CA CD B1 41 1F 10 B2
         75
        INTEGER
         03
        INTEGER
         75 03 CF E8 7A 83 6A E3 A6 1B 88 16 E2 54 50 E6
         CE 5E 1C 93 AC F1 AB C1 77 80 64 FD CB EF A9 21
         DF 16 26 BE 4F D0 36 E9 3D 75 E6 A5 0E 3A 41 E9
         80 28 FE 5F C2 35 F5 B8 89 A5 89 CB 52 15 F2 A4
       }
   }
```

Parameter set: id-tc26-gost-3410-12-512-paramSetB

```
   SEQUENCE
   {
       OBJECT IDENTIFIER
       id-tc26-gost-3410-12-512-paramSetB
       SEQUENCE
       {
        INTEGER
         00 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00
         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
         6F
        INTEGER
         00 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00
         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
         6C
        INTEGER
         68 7D 1B 45 9D C8 41 45 7E 3E 06 CF 6F 5E 25 17
         B9 7C 7D 61 4A F1 38 BC BF 85 DC 80 6C 4B 28 9F
```

```
         3E 96 5D 2D B1 41 6D 21 7F 8B 27 6F AD 1A B6 9C
         50 F7 8B EE 1F A3 10 6E FB 8C CB C7 C5 14 01 16
        INTEGER
         00 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00
         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
         01 49 A1 EC 14 25 65 A5 45 AC FD B7 7B D9 D4 0C
         FA 8B 99 67 12 10 1B EA 0E C6 34 6C 54 37 4F 25
         BD
        INTEGER
         00 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00
         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
         01 49 A1 EC 14 25 65 A5 45 AC FD B7 7B D9 D4 0C
         FA 8B 99 67 12 10 1B EA 0E C6 34 6C 54 37 4F 25
         BD
        INTEGER
         02
        INTEGER
         1A 8F 7E DA 38 9B 09 4C 2C 07 1E 36 47 A8 94 0F
         3C 12 3B 69 75 78 C2 13 BE 6D D9 E6 C8 EC 73 35
         DC B2 28 FD 1E DF 4A 39 15 2C BC AA F8 C0 39 88
         28 04 10 55 F9 4C EE EC 7E 21 34 07 80 FE 41 BD
       }
   }
```

A.2. Twisted Edwards form parameters

Parameter set: id-tc26-gost-3410-2012-256-paramSetA

```
   SEQUENCE
   {
       OBJECT IDENTIFIER
       id-tc26-gost-3410-2012-256-paramSetA
       SEQUENCE
       {
           INTEGER
           00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
           FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FD
           97
           INTEGER
           00 C2 17 3F 15 13 98 16 73 AF 48 92 C2 30 35 A2
           7C E2 5E 20 13 BF 95 AA 33 B2 2C 65 6F 27 7E 73
           35
           INTEGER
           29 5F 9B AE 74 28 ED 9C CC 20 E7 C3 59 A9 D4 1A
           22 FC CD 91 08 E1 7B F7 BA 93 37 A6 F8 AE 95 13
           INTEGER
           01
           INTEGER
```

```
            06 05 F6 B7 C1 83 FA 81 57 8B C3 9C FA D5 18 13
            2B 9D F6 28 97 00 9A F7 E5 22 C3 2D 6D C7 BF FB
            INTEGER
            01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
            00 3F 63 37 7F 21 ED 98 D7 04 56 BD 55 B0 D8 31
            9C
            INTEGER
            40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
            0F D8 CD DF C8 7B 66 35 C1 15 AF 55 6C 36 0C 67
            INTEGER
            00 91 E3 84 43 A5 E8 2C 0D 88 09 23 42 57 12 B2
            BB 65 8B 91 96 93 2E 02 C7 8B 25 82 FE 74 2D AA
            28
            INTEGER
            32 87 94 23 AB 1A 03 75 89 57 86 C4 BB 46 E9 56
            5F DE 0B 53 44 76 67 40 AF 26 8A DB 32 32 2E 5C
            INTEGER
            0D
            INTEGER
            60 CA 1E 32 AA 47 5B 34 84 88 C3 8F AB 07 64 9C
            E7 EF 8D BE 87 F2 2E 81 F9 2B 25 92 DB A3 00 E7
        }
    }
```

Parameter set: id-tc26-gost-3410-2012-512-paramSetC

```
    SEQUENCE
    {
        OBJECT IDENTIFIER
        id-tc26-gost-3410-2012-512-paramSetC
        SEQUENCE
        {
            INTEGER
            00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
            FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
            FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
            FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FD
            C7
            INTEGER
            00 DC 92 03 E5 14 A7 21 87 54 85 A5 29 D2 C7 22
            FB 18 7B C8 98 0E B8 66 64 4D E4 1C 68 E1 43 06
            45 46 E8 61 C0 E2 C9 ED D9 2A DE 71 F4 6F CF 50
            FF 2A D9 7F 95 1F DA 9F 2A 2E B6 54 6F 39 68 9B
            D3
            INTEGER
            00 B4 C4 EE 28 CE BC 6C 2C 8A C1 29 52 CF 37 F1
            6A C7 EF B6 A9 F6 9F 4B 57 FF DA 2E 4F 0D E5 AD
            E0 38 CB C2 FF F7 19 D2 C1 8D E0 28 4B 8B FE F3
```

```
          B5 2B 8C C7 A5 F5 BF 0A 3C 8D 23 19 A5 31 25 57
          E1
          INTEGER
          01
          INTEGER
          00 9E 4F 5D 8C 01 7D 8D 9F 13 A5 CF 3C DF 5B FE
          4D AB 40 2D 54 19 8E 31 EB DE 28 A0 62 10 50 43
          9C A6 B3 9E 0A 51 5C 06 B3 04 E2 CE 43 E7 9E 36
          9E 91 A0 CF C2 BC 2A 22 B4 CA 30 2D BB 33 EE 75
          50
          INTEGER
          00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
          FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
          FF 26 33 6E 91 94 1A AC 01 30 CE A7 FD 45 1D 40
          B3 23 B6 A7 9E 9D A6 84 9A 51 88 F3 BD 1F C0 8F
          B4
          INTEGER
          3F FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
          FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
          C9 8C DB A4 65 06 AB 00 4C 33 A9 FF 51 47 50 2C
          C8 ED A9 E7 A7 69 A1 26 94 62 3C EF 47 F0 23 ED
          INTEGER
          00 E2 E3 1E DF C2 3D E7 BD EB E2 41 CE 59 3E F5
          DE 22 95 B7 A9 CB AE F0 21 D3 85 F7 07 4C EA 04
          3A A2 72 72 A7 AE 60 2B F2 A7 B9 03 3D B9 ED 36
          10 C6 FB 85 48 7E AE 97 AA C5 BC 79 28 C1 95 01
          48
          INTEGER
          00 F5 CE 40 D9 5B 5E B8 99 AB BC CF F5 91 1C B8
          57 79 39 80 4D 65 27 37 8B 8C 10 8C 3D 20 90 FF
          9B E1 8E 2D 33 E3 02 1E D2 EF 32 D8 58 22 42 3B
          63 04 F7 26 AA 85 4B AE 07 D0 39 6E 9A 9A DD C4
          0F
          INTEGER
          12
          INTEGER
          46 9A F7 9D 1F B1 F5 E1 6B 99 59 2B 77 A0 1E 2A
          0F DF B0 D0 17 94 36 8D 9A 56 11 7F 7B 38 66 95
          22 DD 4B 65 0C F7 89 EE BF 06 8C 5D 13 97 32 F0
          90 56 22 C0 4B 2B AA E7 60 03 03 EE 73 00 1A 3D
      }
  }
```

Appendix B. Test examples

1)   HMAC_GOSTR3411_2012_256

Key K:

```
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
```

T:

```
01 26 bd b8 78 00 af 21 43 41 45 65 63 78 01 00
```

HMAC_GOSTR3411_2012_256(K, T) value:

```
a1 aa 5f 7d e4 02 d7 b3 d3 23 f2 99 1c 8d 45 34
01 31 37 01 0a 83 75 4f d0 af 6d 7c d4 92 2e d9
```

2)   HMAC_GOSTR3411_2012_512

Key K:

```
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
```

T:

```
01 26 bd b8 78 00 af 21 43 41 45 65 63 78 01 00
```

HMAC_GOSTR3411_2012_512(K, T) value:

```
a5 9b ab 22 ec ae 19 c6 5f bd e6 e5 f4 e9 f5 d8
54 9d 31 f0 37 f9 df 9b 90 55 00 e1 71 92 3a 77
3d 5f 15 30 f2 ed 7e 96 4c b2 ee dc 29 e9 ad 2f
3a fe 93 b2 81 4f 79 f5 00 0f fc 03 66 c2 51 e6
```

3)   PRF_TLS_GOSTR3411_2012_256

Key K:

```
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
```

Seed:

```
18 47 1d 62 2d c6 55 c4 d2 d2 26 96 91 ca 4a 56
0b 50 ab a6 63 55 3a f2 41 f1 ad a8 82 c9 f2 9a
```

Label:

```
11 22 33 44 55
```

Output T1:

```
ff 09 66 4a 44 74 58 65 94 4f 83 9e bb 48 96 5f
15 44 ff 1c c8 e8 f1 6f 24 7e e5 f8 a9 eb e9 7f
```

Output T2:

```
c4 e3 c7 90 0e 46 ca d3 db 6a 01 64 30 63 04 0e
c6 7f c0 fd 5c d9 f9 04 65 23 52 37 bd ff 2c 02
```

4)   PRF_TLS_GOSTR3411_2012_512

Key K:

```
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
```

Seed:

```
18 47 1d 62 2d c6 55 c4 d2 d2 26 96 91 ca 4a 56
0b 50 ab a6 63 55 3a f2 41 f1 ad a8 82 c9 f2 9a
```

Label:

```
11 22 33 44 55
```

Output T1:

```
f3 51 87 a3 dc 96 55 11 3a 0e 84 d0 6f d7 52 6c
5f c1 fb de c1 a0 e4 67 3d d6 d7 9d 0b 92 0e 65
ad 1b c4 7b b0 83 b3 85 1c b7 cd 8e 7e 6a 91 1a
62 6c f0 2b 29 e9 e4 a5 8e d7 66 a4 49 a7 29 6d
```

Output T2:

```
e6 1a 7a 26 c4 d1 ca ee cf d8 0c ca 65 c7 1f 0f
88 c1 f8 22 c0 e8 c0 ad 94 9d 03 fe e1 39 57 9f
72 ba 0c 3d 32 c5 f9 54 f1 cc cd 54 08 1f c7 44
02 78 cb a1 fe 7b 7a 17 a9 86 fd ff 5b d1 5d 1f
```

5)   PRF_IPSEC_KEYMAT_GOSTR3411_2012_256

Key K:

```
c9 a9 a7 73 20 e2 cc 55 9e d7 2d ce 6f 47 e2 19
2c ce a9 5f a6 48 67 05 82 c0 54 c0 ef 36 c2 21
```

Data of S:

```
01 26 bd b8 78 00 1d 80 60 3c 85 44 c7 27 01 00
```

Output T1:

```
21 01 d8 0c 47 db 54 bc 3c 82 9b 8c 30 7c 47 55
50 88 83 a6 d6 9e 60 1b f7 aa fb 0a bc a4 ed 95
```

Output T2:

```
33 b8 4e d0 8f 93 56 f8 1d f8 d2 79 f0 79 c9 02
87 cb 45 2c 81 d4 1e 80 38 43 08 86 c1 92 12 aa
```

6) PRF_IPSEC_PRFPLUS_GOSTR3411_2012_256

Key K:

```
c9 a9 a7 73 20 e2 cc 55 9e d7 2d ce 6f 47 e2 19
2c ce a9 5f a6 48 67 05 82 c0 54 c0 ef 36 c2 21
```

Data of S:

```
01 26 bd b8 78 00 1d 80 60 3c 85 44 c7 27 01 00
```

Output T1:

```
2d e5 ee 84 e1 3d 7b e5 36 16 67 39 13 37 0a b0
54 c0 74 b7 9b 69 a8 a8 46 82 a9 f0 4f ec d5 87
```

Output T2:

```
29 f6 0d da 45 7b f2 19 aa 2e f9 5d 7a 59 be 95
4d e0 08 f4 a5 0d 50 4d bd b6 90 be 68 06 01 53
```

7) PRF_IPSEC_KEYMAT_GOSTR3411_2012_512

Key K:

```
c9 a9 a7 73 20 e2 cc 55 9e d7 2d ce 6f 47 e2 19
2c ce a9 5f a6 48 67 05 82 c0 54 c0 ef 36 c2 21
```

Data of S:

```
01 26 bd b8 78 00 1d 80 60 3c 85 44 c7 27 01 00
```

Output T1:

```
b9 55 5b 29 91 75 4b 37 9d a6 8e 60 98 f5 b6 0e
df 91 8a 56 20 4b ff f3 a8 37 6d 1f 57 ed b2 34
a5 12 32 81 23 cd 6c 03 0b 54 14 2e 1e c7 78 2b
03 00 be a5 7c c2 a1 4c a3 b4 f0 85 a4 5c d6 ca
```

Output T2:

```
37 b1 e0 86 52 43 a4 fb 29 14 8d 27 4d 30 63 fc
bf b0 f2 f4 68 d5 27 e4 3b ca 41 fa 6b b5 3e c8
df 21 bf c4 62 3a 2e 76 8b 64 54 03 3e 09 52 32
d1 8c 86 a6 8f 00 98 d3 31 81 75 f6 59 05 ae db
```

8)   PRF_IPSEC_ PRFPLUS_GOSTR3411_2012_512

Key K:

```
c9 a9 a7 73 20 e2 cc 55 9e d7 2d ce 6f 47 e2 19
2c ce a9 5f a6 48 67 05 82 c0 54 c0 ef 36 c2 21
```

Data of S:

```
01 26 bd b8 78 00 1d 80 60 3c 85 44 c7 27 01 00
```

Output T1:

```
5d a6 71 43 a5 f1 2a 6d 6e 47 42 59 6f 39 24 3f
cc 61 57 45 91 5b 32 59 10 06 ff 78 a2 08 63 d5
f8 8e 4a fc 17 fb be 70 b9 50 95 73 db 00 5e 96
26 36 98 46 cb 86 19 99 71 6c 16 5d d0 6a 15 85
```

Output T2:

```
48 34 49 5a 43 74 6c b5 3f 0a ba 3b c4 6e bc f8
77 3c a6 4a d3 43 c1 22 ee 2a 57 75 57 03 81 57
ee 9c 38 8d 96 ef 71 d5 8b e5 c1 ef a1 af a9 5e
be 83 e3 9d 00 e1 9a 5d 03 dc d6 0a 01 bc a8 e3
```

9)   VKO_GOSTR3410_2012_256 with 256-bit output on the GOST
R 34.10-2012 512-bit keys with id-tc26-gost-3410-12-512-paramSetA

UKM value:

```
1d 80 60 3c 85 44 c7 27
```

Private key x of A:

```
c9 90 ec d9 72 fc e8 4e c4 db 02 27 78 f5 0f ca
c7 26 f4 67 08 38 4b 8d 45 83 04 96 2d 71 47 f8
c2 db 41 ce f2 2c 90 b1 02 f2 96 84 04 f9 b9 be
6d 47 c7 96 92 d8 18 26 b3 2b 8d ac a4 3c b6 67
```

Public key x*P of A (curve point (X, Y)):

```
aa b0 ed a4 ab ff 21 20 8d 18 79 9f b9 a8 55 66
54 ba 78 30 70 eb a1 0c b9 ab b2 53 ec 56 dc f5
d3 cc ba 61 92 e4 64 e6 e5 bc b6 de a1 37 79 2f
24 31 f6 c8 97 eb 1b 3c 0c c1 43 27 b1 ad c0 a7
91 46 13 a3 07 4e 36 3a ed b2 04 d3 8d 35 63 97
1b d8 75 8e 87 8c 9d b1 14 03 72 1b 48 00 2d 38
46 1f 92 47 2d 40 ea 92 f9 95 8c 0f fa 4c 93 75
64 01 b9 7f 89 fd be 0b 5e 46 e4 a4 63 1c db 5a
```

Private key y of part B:

```
48 c8 59 f7 b6 f1 15 85 88 7c c0 5e c6 ef 13 90
cf ea 73 9b 1a 18 c0 d4 66 22 93 ef 63 b7 9e 3b
80 14 07 0b 44 91 85 90 b4 b9 96 ac fe a4 ed fb
bb cc cc 8c 06 ed d8 bf 5b da 92 a5 13 92 d0 db
```

Public key y*P of B (curve point (X, Y)):

```
19 2f e1 83 b9 71 3a 07 72 53 c7 2c 87 35 de 2e
a4 2a 3d bc 66 ea 31 78 38 b6 5f a3 25 23 cd 5e
fc a9 74 ed a7 c8 63 f4 95 4d 11 47 f1 f2 b2 5c
39 5f ce 1c 12 91 75 e8 76 d1 32 e9 4e d5 a6 51
04 88 3b 41 4c 9b 59 2e c4 dc 84 82 6f 07 d0 b6
d9 00 6d da 17 6c e4 8c 39 1e 3f 97 d1 02 e0 3b
b5 98 bf 13 2a 22 8a 45 f7 20 1a ba 08 fc 52 4a
2d 77 e4 3a 36 2a b0 22 ad 40 28 f7 5b de 3b 79
```

KEK_VKO value:

```
c9 a9 a7 73 20 e2 cc 55 9e d7 2d ce 6f 47 e2 19
2c ce a9 5f a6 48 67 05 82 c0 54 c0 ef 36 c2 21
```

10)   VKO_GOSTR3410_2012_512 with 512-bit output on the GOST R 34.10-2012 512-bit keys with id-tc26-gost-3410-12-512-paramSetA

UKM value:

```
1d 80 60 3c 85 44 c7 27
```

Private key x of A:

```
c9 90 ec d9 72 fc e8 4e c4 db 02 27 78 f5 0f ca
c7 26 f4 67 08 38 4b 8d 45 83 04 96 2d 71 47 f8
c2 db 41 ce f2 2c 90 b1 02 f2 96 84 04 f9 b9 be
6d 47 c7 96 92 d8 18 26 b3 2b 8d ac a4 3c b6 67
```

Public key x*P of A (curve point (X, Y)):

```
aa b0 ed a4 ab ff 21 20 8d 18 79 9f b9 a8 55 66
54 ba 78 30 70 eb a1 0c b9 ab b2 53 ec 56 dc f5
d3 cc ba 61 92 e4 64 e6 e5 bc b6 de a1 37 79 2f
24 31 f6 c8 97 eb 1b 3c 0c c1 43 27 b1 ad c0 a7
91 46 13 a3 07 4e 36 3a ed b2 04 d3 8d 35 63 97
1b d8 75 8e 87 8c 9d b1 14 03 72 1b 48 00 2d 38
46 1f 92 47 2d 40 ea 92 f9 95 8c 0f fa 4c 93 75
64 01 b9 7f 89 fd be 0b 5e 46 e4 a4 63 1c db 5a
```

Private key y of B:

```
48 c8 59 f7 b6 f1 15 85 88 7c c0 5e c6 ef 13 90
cf ea 73 9b 1a 18 c0 d4 66 22 93 ef 63 b7 9e 3b
80 14 07 0b 44 91 85 90 b4 b9 96 ac fe a4 ed fb
bb cc cc 8c 06 ed d8 bf 5b da 92 a5 13 92 d0 db
```

Public key y*P of B (curve point (X, Y)):

```
19 2f e1 83 b9 71 3a 07 72 53 c7 2c 87 35 de 2e
a4 2a 3d bc 66 ea 31 78 38 b6 5f a3 25 23 cd 5e
fc a9 74 ed a7 c8 63 f4 95 4d 11 47 f1 f2 b2 5c
39 5f ce 1c 12 91 75 e8 76 d1 32 e9 4e d5 a6 51
04 88 3b 41 4c 9b 59 2e c4 dc 84 82 6f 07 d0 b6
d9 00 6d da 17 6c e4 8c 39 1e 3f 97 d1 02 e0 3b
b5 98 bf 13 2a 22 8a 45 f7 20 1a ba 08 fc 52 4a
2d 77 e4 3a 36 2a b0 22 ad 40 28 f7 5b de 3b 79
```

KEK_VKO value:

```
79 f0 02 a9 69 40 ce 7b de 32 59 a5 2e 01 52 97
ad aa d8 45 97 a0 d2 05 b5 0e 3e 17 19 f9 7b fa
7e e1 d2 66 1f a9 97 9a 5a a2 35 b5 58 a7 e6 d9
f8 8f 98 2d d6 3f c3 5a 8e c0 dd 5e 24 2d 3b df
```

11)   Key derivation function KDF_GOSTR3411_2012_256

K_in key:

```
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
```

Label:

```
26 bd b8 78
```

Seed:

```
af 21 43 41 45 65 63 78
```

KDF(K_in, label, seed) value:

a1 aa 5f 7d e4 02 d7 b3 d3 23 f2 99 1c 8d 45 34
01 31 37 01 0a 83 75 4f d0 af 6d 7c d4 92 2e d9

12) Key derivation function KDF_TREE_GOSTR3411_2012_256

Output size of L:

512

K_in key:

00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f

Label:

26 bd b8 78

Seed:

af 21 43 41 45 65 63 78

Value of K1:

22 b6 83 78 45 c6 be f6 5e a7 16 72 b2 65 83 10
86 d3 c7 6a eb e6 da e9 1c ad 51 d8 3f 79 d1 6b

Value of K2:

07 4c 93 30 59 9d 7f 8d 71 2f ca 54 39 2f 4d dd
e9 37 51 20 6b 35 84 c8 f4 3f 9e 6d c5 15 31 f9

Value of R:

1

13) Key wrap and unwrap with the szOID_Gost28147_89_TC26_Z_ParamSet parameters

Key K_e:

00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f

Key K:

```
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
```

seed value:

```
af 21 43 41 45 65 63 78
```

Label:

```
26 bd b8 78
```

KEK_e(seed) = KDF_GOSTR3411_2012_256(K_e, label, seed):

```
a1 aa 5f 7d e4 02 d7 b3 d3 23 f2 99 1c 8d 45 34
01 31 37 01 0a 83 75 4f d0 af 6d 7c d4 92 2e d9
```

CEK_MAC:

```
be 33 f0 52
```

CEK_ENC:

```
d1 55 47 f8 ee 85 12 1b c8 7d 4b 10 27 d2 60 27
ec c0 71 bb a6 e7 2f 3f ec 6f 62 0f 56 83 4c 5a
```

Appendix C. GOST 28147-89 parameter set

The parameter set has the following object identifier:

1. id-tc26-gost-28147-param-Z::= {iso(1) member-body(2) ru(643) rosstandart(7) tc26(1) constants(2) cipher-constants(5) gost-28147-constants(1) param-Z(1)}

The parameter set is defined below:

x	K1(x)	K2(x)	K3(x)	K4(x)	K5(x)	K6(x)	K7(x)	K8(x)
0	c	6	b	c	7	5	8	1
1	4	8	3	8	f	d	e	7
2	6	2	5	2	5	f	2	e
3	2	3	8	1	a	6	5	d
4	a	9	2	d	8	9	6	0
5	5	a	f	4	1	2	9	5
6	b	5	a	f	6	c	1	8
7	9	c	d	6	d	a	c	3
8	e	1	e	7	0	b	f	4
9	8	e	1	0	9	7	4	f
a	d	4	7	a	3	8	b	a
b	7	7	4	5	e	1	0	6
c	0	b	c	3	b	4	d	9
d	3	d	9	e	4	3	a	c
e	f	0	6	9	2	e	3	b
f	1	f	0	b	c	0	7	2

Authors' Addresses

Stanislav Smyshlyaev (editor)
CRYPTO-PRO
18, Suschevsky val
Moscow 127018
Russian Federation

Phone: +7 (495) 995-48-20
Email: svs@cryptopro.ru

Evgeny Alekseev
CRYPTO-PRO
18, Suschevsky val
Moscow 127018
Russian Federation

Phone: +7 (495) 995-48-20
Email: alekseev@cryptopro.ru

Igor Oshkin
CRYPTO-PRO
18, Suschevsky val
Moscow 127018
Russian Federation

Phone: +7 (495) 995-48-20
Email: oshkin@cryptopro.ru

Vladimir Popov
CRYPTO-PRO
18, Suschevsky val
Moscow 127018
Russian Federation

Phone: +7 (495) 995-48-20
Email: vpopov@cryptopro.ru

Serguei Leontiev
CRYPTO-PRO
18, Suschevsky val
Moscow 127018
Russian Federation

Phone: +7 (495) 995-48-20
Email: lse@cryptopro.ru

Vladimir Podobaev
FACTOR-TS
11A, 1st Magistralny proezd
Moscow 123290
Russian Federation

Phone: +7 (495) 644-31-30
Email: v_podobaev@factor-ts.ru

Dmitry Belyavsky
TCI
8, Zoologicheskaya st
Moscow 117218
Russian Federation

Phone: +7 (499) 254-24-50
Email: beldmit@gmail.com