

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 21, 2016

S. Smyshlyaev, Ed.  
E. Alekseev  
I. Oshkin  
V. Popov  
CRYPTO-PRO  
V. Podobaev  
FACTOR-TS  
I. Ustinov  
Cryptocom  
July 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-01

## Abstract

The usage of cryptographic algorithms defined by GOST R 34.10-2012 [GOST3410-2012] and GOST R 34.11-2012 [GOST3411-2012] standards for protection of the information is carried out, as a rule, within the cryptographic protocols based on the accompanying algorithms.

This memo contains a description of the accompanying algorithms defining the pseudorandom functions, the key agreement protocols based on the Diffie-Hellman method, the parameters of elliptic curves, the key derivation functions and the algorithms used for export of keying material.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 21, 2016.

## Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1.	Introduction . . . . .	2
2.	Scope . . . . .	3
3.	Conventions used in This Document . . . . .	3
3.1.	Notation . . . . .	3
3.2.	Basic terms and definitions . . . . .	4
4.	Algorithm descriptions . . . . .	6
4.1.	HMAC functions . . . . .	6
4.2.	PRF . . . . .	6
4.3.	VKO algorithms for key agreement . . . . .	8
4.4.	The parameters of elliptic curves . . . . .	10
4.5.	Key derivation function KDF_GOSTR3411_2012_256 . . . . .	13
4.6.	Key derivation function KDF_TREE_GOSTR3411_2012_256 . . . . .	14
4.7.	Key wrap and unwrap . . . . .	15
5.	Acknowledgments . . . . .	16
6.	References . . . . .	16
6.1.	Normative References . . . . .	16
6.2.	Informative References . . . . .	17
Appendix A.	Values of parameter sets . . . . .	18
A.1.	Canonical form parameters . . . . .	18
A.2.	Twisted Edwards form parameters . . . . .	20
Appendix B.	Test examples . . . . .	23
Authors' Addresses	. . . . .	34

## 1. Introduction

The usage of cryptographic algorithms defined by the GOST R 34.10-2012 [GOST3410-2012] and GOST R 34.11-2012 [GOST3411-2012] standards for protection of the information is carried out, as a rule, within the cryptographic protocols based on the accompanying algorithms.

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

This memo contains a description of the accompanying algorithms defining the pseudorandom functions, the key derivation functions, the key agreement protocols based on the Diffie-Hellman method and the algorithms used for export of key material.

This memo does not specify the cryptographic algorithms GOST R 34.10-2012 [GOST3410-2012] and GOST R 34.11-2012 [GOST3411-2012]. These algorithms are defined by the national standards GOST R 34.10-2012 [GOST3410-2012] and GOST R 34.11-2012 [GOST3411-2012] and described in [RFC7091] and [RFC6986] (an English version of Russian national standards).

The need to ensure compatibility of the cryptographic protocol implementations based on the Russian cryptographic standards GOST R 34.10-2012 [GOST3410-2012] and GOST R 34.11-2012 [GOST3411-2012] is served as the main reason for the development of this document.

## 2. Scope

This memo is recommended for usage in encryption and protection the authenticity of the data based on the usage of the digital signature algorithms GOST R 34.10-2012 [GOST3410-2012] and hash function GOST R 34.11-2012 [GOST3411-2012] in public and corporate networks to protect information that does not contain a classified information.

## 3. 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].

### 3.1. Notation

This document uses the following notation for the sets and operations on the elements of these sets in accordance with GOST R 34.11-2012 [GOST3411-2012]:

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

$V_n$  the finite-dimensional vector space over  $GF(2)$  of dimension  $n$  with the (xor) operation, for  $n = 0$  the  $V_0$  space consists of a single empty element of size 0;

- U the element of  $V_n$ ; in the binary representation  $U = (u_{(n-1)}, u_{(n-2)}, \dots, u_1, u_0)$ , where  $u_i$  in  $\{0, 1\}$ ;
- $A|B$  concatenation of vectors  $A, B$ , i.e., if  $A$  in  $V_{n1}$ ,  $B$  in  $V_{n2}$ ,  $A = (a_{(n1-1)}, a_{(n1-2)}, \dots, a_0)$ , and  $B = (b_{(n2-1)}, b_{(n2-2)}, \dots, b_0)$ , then  $A|B = (a_{(n1-1)}, a_{(n1-2)}, \dots, a_0, b_{(n2-1)}, b_{(n2-2)}, \dots, b_0)$  is an element of  $V_{(n1+n2)}$ ;
- $V_{(8, r)}$  the set of byte strings of size  $r$ ; if  $W$  is an element of  $V_{(8, r)}$ , then  $W = (w^0, w^1, \dots, w^{(r-1)})$ , where  $w^0, w^1, \dots, w^{(r-1)}$  are elements of  $V_8$ ; if  $A$  in  $V_{(8, r1)}$ ,  $B$  in  $V_{(8, r2)}$ ,  $A = (a^0, a^1, \dots, a^{(r1-1)})$ , and  $B = (b^0, b^1, \dots, b^{(r2-1)})$ , then  $A|B = (a^0, a^1, \dots, a^{(r1-1)}, b^0, b^1, \dots, b^{(r2-1)})$  is an element of  $V_{(8, r1+r2)}$ ;
- Bit representation the bit representation of the element  $W = (w^0, w^1, \dots, w^{(r-1)})$  of  $V_{(8, r)}$ , where  $w^0 = (w_7, w_6, \dots, w_0)$ ,  $w^1 = (w_{15}, w_{14}, \dots, w_8)$ ,  $\dots$ ,  $w^{(r-1)} = (w_{(8r-1)}, w_{(8r-2)}, \dots, w_{(8r-8)})$  are elements of  $V_8$ , is an element  $(w_{(8r-1)}, w_{(8r-2)}, \dots, w_1, w_0)$  of  $V_{(8*r)}$ ;
- Byte representation if  $n$  is a multiple of 8,  $r = n/8$ , then the byte representation of the element  $W = (w_{(n-1)}, w_{(n-2)}, \dots, w_0)$  of  $V_n$  is a byte string  $(w^0, w^1, \dots, w^{(r-1)})$  of  $V_{(8, r)}$ , where  $w^0 = (w_7, w_6, \dots, w_0)$ ,  $w^1 = (w_{15}, w_{14}, \dots, w_8)$ ,  $\dots$ ,  $w^{(r-1)} = (w_{(8r-1)}, w_{(8r-2)}, \dots, w_{(8r-8)})$  are elements of  $V_8$ ;
- K (key) 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.

Note: It is proposed to interpret and edit the formulas in accordance with the above definitions.

### 3.2. Basic terms and definitions

This memo uses the following terms, abbreviations and symbols:

Symbols	Meaning
H_256	GOST R 34.11-2012 hash function, 256-bit
H_512	GOST R 34.11-2012 hash function, 512-bit
HMAC	a function for calculating a message authentication code, based on hash function in accordance with [RFC2104]
HMAC_256	an HMAC function based on the hash function H_256, intended for computing a message authentication code
HMAC_512	an HMAC function based on the hash function H_512, intended for computing a message authentication code
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 for the root key and random data using a pseudorandom function

To produce a byte sequence of the size  $r$  with functions that give a longer output the input should be taken from the output sequence of the first  $r$  bytes. This remark applies to the following functions:

- o the functions described in Section 4.2;
- o KDF\_TREE\_GOSTR3411\_2012\_256.

When  $n$  is multiple of 8, an element of  $V_n$  can be represented in the bit and byte form. The result of operation  $\ll|\gg$ , applied to the elements in the bit representation is described in the bit representation. The result of the operation  $\ll|\gg$ , applied to the same elements in byte representation is described in the byte representation. Thus, the symbol  $\ll|\gg$  is used to refer to two different operations, depending on the form of their arguments. The operation is uniquely determined by the representation of arguments.

Hereinafter all data (the elements of  $V_n$ ) are considered given in the byte representation unless otherwise specified. Operation  $\ll|\gg$  on the arguments of functions, unless explicitly stated, is performed on their byte representation.

If the function is defined outside this document (eg, H\_256) and its definition is using arguments in bit representation, it is assumed that the bit representation of the argument is formed immediately before the calculation of the function (in particular, immediately after the application of the operation  $\ll|\gg$  to the byte representation of the arguments).

If the output of another function that is defined outside of this document is used as the argument of the function defined below and has output value in bit representation, it is assumed that the output value will be translated into the byte representation before substitution in arguments.

#### 4. Algorithm descriptions

For the algorithms described in this paper the possible values of the functions are limited by the permissibility of applying them as the input parameter of the transformations and are assigned by the protocols.

##### 4.1. HMAC functions

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

###### 4.1.1. HMAC\_GOSTR3411\_2012\_256

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 given in [RFC2104]. The size of the HMAC\_256 output in bytes is equal to 32, the block size of the iterative procedure for the H\_256 compression function in bytes is equal to 64 (in the notation of [RFC2104],  $L = 32$  and  $B = 64$ , respectively).

###### 4.1.2. HMAC\_GOSTR3411\_2012\_512

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 given in [RFC2104]. The size of the HMAC\_512 output in bytes is equal to 64, the block size of the iterative procedure for the H\_512 compression function in bytes is equal to 64 (in the notation of [RFC2104],  $L = 64$  and  $B = 64$ , respectively).

##### 4.2. PRF

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 IPsec.

## 4.2.1. PRFs for the TLS protocol

## 4.2.1.1. PRF\_TLS\_GOSTR3411\_2012\_256

This is the transformation to implement the pseudorandom function of the TLS protocol; the transformation uses HMAC\_256 based on GOST R 34.11-2012 [GOST3411-2012], 256-bit output, and corresponds to the method of specifying the arguments and the output value of P\_hash data expansion function, given in Section 5 of [RFC2246] and kept in [RFC5246].

$$\begin{aligned} \text{PRF\_TLS\_GOSTR3411\_2012\_256}(\text{secret}, \text{label}, \text{seed}) &= \\ &= \text{P\_GOSTR3411\_2012\_256}(\text{secret}, \text{label} \mid \text{seed}), \end{aligned}$$

## 4.2.1.2. PRF\_TLS\_GOSTR3411\_2012\_512

This is the transformation to implement the pseudorandom function of the TLS protocol; the transformation uses HMAC\_512 based on GOST R 34.11-2012 [GOST3411-2012], 512-bit output, and corresponds to the method of specifying the arguments and the output value of P\_hash data expansion function, given in Section 5 of [RFC2246] and kept in [RFC5246].

$$\begin{aligned} \text{PRF\_TLS\_GOSTR3411\_2012\_512}(\text{secret}, \text{label}, \text{seed}) &= \\ &= \text{P\_GOSTR3411\_2012\_512}(\text{secret}, \text{label} \mid \text{seed}), \end{aligned}$$

## 4.2.2. PRFs for the IPsec protocols based on GOST R 34.11-2012, 256-bit

## 4.2.2.1. PRF\_IPSEC\_KEYMAT\_GOSTR3411\_2012\_256

This pseudorandom function used for the keying material generation is defined as follows (the arguments are the byte strings K and S):

$$\text{PRF\_IPSEC\_KEYMAT\_GOSTR3411\_2012\_256}(K, S) = T1 \mid T2 \mid T3 \mid T4 \mid \dots,$$

where

$$\begin{aligned} T1 &= \text{HMAC\_256}(K, S), \\ T2 &= \text{HMAC\_256}(K, T1 \mid S), \\ T3 &= \text{HMAC\_256}(K, T2 \mid S), \\ T4 &= \text{HMAC\_256}(K, T3 \mid S), \\ &\dots \end{aligned}$$

PRF\_IPSEC\_KEYMAT\_GOSTR3411\_2012\_256 function is similar to KEYMAT function in [RFC2409] regarding the assignment scheme for the arguments in the iterations.

#### 4.2.2.2. PRF\_IPSEC\_PRFPLUS\_GOSTR3411\_2012\_256

The pseudorandom function PRF\_IPSEC\_PRFPLUS\_GOSTR3411\_2012\_256 is similar to the prf+ function in [RFC7296], where HMAC\_256 function is used as a prf function.

#### 4.2.3. PRFs for the IPsec protocols based on GOST R 34.11-2012, 512-bit

##### 4.2.3.1. PRF\_IPSEC\_KEYMAT\_GOSTR3411\_2012\_512

This pseudorandom function for the keying material generation is defined as follows (the arguments are the byte strings K and S):

$$\text{PRF\_IPSEC\_KEYMAT\_GOSTR3411\_2012\_512 (K, S) = T1 | T2 | T3 | T4 | \dots,}$$

where

$$\begin{aligned} T1 &= \text{HMAC\_512 (K, S),} \\ T2 &= \text{HMAC\_512 (K, T1 | S),} \\ T3 &= \text{HMAC\_512 (K, T2 | S),} \\ T4 &= \text{HMAC\_512 (K, T3 | S),} \\ &\dots \end{aligned}$$

PRF\_IPSEC\_KEYMAT\_GOSTR3411\_2012\_512 is similar to KEYMAT function in [RFC2409] regarding the assignment scheme for the arguments in iterations.

##### 4.2.3.2. PRF\_IPSEC\_PRFPLUS\_GOSTR3411\_2012\_512

The pseudorandom function PRF\_IPSEC\_PRFPLUS\_GOSTR3411\_2012\_512 is similar to the prf+ function in [RFC7296], where HMAC\_512 function is used as a prf function.

#### 4.3. VKO algorithms for key agreement

This section identifies the key agreement algorithms using GOST R 34.10-2012 [GOST3410-2012].

##### 4.3.1. VKO\_GOSTR3410\_2012\_256

The VKO\_GOSTR3410\_2012\_256 transformation is used for an agreement of the VKO 256-bit keys and based on GOST R 34.11-2012 [GOST3411-2012], 256-bit. This algorithm can be applied for a key agreement using the GOST R 34.10-2012 [GOST3410-2012] 256-bit and 512-bit 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. Key or keying material KEK\_VKO (x, y, UKM) is produced by



the side of communication from his private key  $x$ , the public key  $y*P$  of the opposite side and UKM value, considered as a number.

The algorithm can be used for deriving both static and ephemeral key with the public key size  $n \geq 512$  bits including the case where one side uses a static key and the other - ephemeral.

UKM parameter is optional (the default UKM = 1) and can take any value from 1 to  $2^{(n/2)}-1$ . It is allowed to use a nonzero UKM of arbitrary size not exceeding  $n/2$  bits. UKM size of 64 bit or more is recommended for cases where the keys at least one of the parties are static.

$K$  is calculated using formula

$$K(x, y, UKM) = (m/q * UKM * x \bmod q) * (y * P),$$

where  $m$  and  $q$  are the parameters of the elliptic curve defined in the GOST R 34.10-2012 [GOST3410-2012] notation.

KEK\_VKO is calculated using formula

$$KEK\_VKO(x, y, UKM) = H_{256}(K(x, y, UKM)).$$

This algorithm is defined by analogy with 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`) and  $K(x, y, UKM)$  is calculated with public key size  $n \geq 512$  bits and UKM size up to  $n/2$  bits.

#### 4.3.2. VKO\_GOSTR3410\_2012\_512

The VKO\_GOSTR3410\_2012\_256 transformation is used for an agreement of the VKO 512-bit keys and based on GOST R 34.11-2012 [GOST3411-2012], 512-bit. This algorithm can be applied for a key agreement using the GOST R 34.10-2012 [GOST3410-2012] 512-bit keys.

The algorithm is designed to produce an encryption key or keying material of size 512 bits to be used in cryptographic protocols. Key or keying material KEK\_VKO ( $x, y, UKM$ ) is produced by the exchange participant from his private key  $x$ , the public key  $y*P$  of the opposite side and the UKM value, considered as a number.

The algorithm can be used for both static and ephemeral key with the public key size  $n \geq 1024$  bits including the case where one side uses a static key and the other uses an ephemeral one.

UKM parameter is optional (the default UKM = 1) and can take any value from 1 to  $2^{(n/2)-1}$ . It is allowed to use a nonzero UKM of arbitrary size not exceeding  $n/2$  bits. UKM size of 128 bit or more is recommended for cases where the keys at least one of the parties are static.

$$K(x, y, UKM) = (m/q * UKM * x \bmod q) * (y * P),$$

where  $m$  and  $q$  - the parameters of the elliptic curve according GOST R 34.10-2012 [GOST3410-2012] notation.

$$KEK\_VKO(x, y, UKM) = H_{512}(K(x, y, UKM)).$$

This algorithm is defined by analogy with Section 5.2 of [RFC4357], but instead of the hash function GOST R 34.11-94 [GOST3411-94] (referred as gostR3411) applies the hash function  $H_{256}$ , and  $K(x, y, UKM)$  is calculated at the public key size  $n \geq 1024$  bits and UKM size up to  $n/2$  bits.

#### 4.4. The parameters of elliptic curves

This section defines the elliptic curves parameters and identifiers that are recommended for the usage with signature and verification algorithms of digital signature in accordance with 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.

##### 4.4.1. 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 a prime 512-bit modulus in canonical (Weierstrass) form, that is given by the following equation defined in GOST R 34.10-2012 [GOST3410-2012]:

$$y^2 = x^3 + ax + b.$$

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

##### 4.4.1.1. Parameters and identifiers

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

p the elliptic curve modulus;

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

q the order of the elliptic curve;

(x, y) the coordinates of a point P (generator of the prime order group) of the elliptic curve in the canonical form.

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

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

Parameter sets have the following identifiers:

1. id-tc26-gost-3410-12-512-paramSetA, <<1.2.643.7.1.2.1.2.1.>> ,
2. id-tc26-gost-3410-12-512-paramSetB, <<1.2.643.7.1.2.1.2.2.>> .

Corresponding values of parameter sets can be found in Appendix Appendix A.1.

#### 4.4.2. Twisted Edwards form

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

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

$$e*u^2 + v^2 = 1 + d*u^2*v^2,$$

where e, d are in  $F_p$ ,  $ed(e-d) \neq 0$ .

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

$$a = s^2 - 3t^2,$$

$$b = 2t^3 - ts^2,$$

where

$$s = (e - d) / 4,$$

$$t = (e + d) / 6,$$

Coordinate transformation is defined as follows:

$$(u,v) \rightarrow (x,y) = (s(1+v) / (1-v) + t, s(1+v) / ((1-v)u)),$$

$$(x,y) \rightarrow (u,v) = ((x-t) / y, (x-t-s) / (x-t+s)).$$

#### 4.4.2.1. Parameters and identifiers

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

- p the elliptic curve modulus;
- 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 order of the elliptic curve group;
- q the order of the subgroups of prime order elliptic curve group;
- (x, y) the coordinates of a point P (generator of the prime order group) of the elliptic curve in the canonical form;
- (u, v) the coordinates of a point P (generator of the prime order group) 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
}
```

Parameter sets have the following identifiers:

1. id-tc26-gost-3410-2012-256-paramSetA, <<1.2.643.7.1.2.1.1.1>> ,
2. id-tc26-gost-3410-2012-512-paramSetC, <<1.2.643.7.1.2.1.2.3>> .

Corresponding values of parameter sets can be found in Appendix Appendix A.2.

#### 4.5. Key derivation function KDF\_GOSTR3411\_2012\_256

The key derivation function KDF\_GOSTR3411\_2012\_256 based on HMAC\_256 function is designed to generate a 256-bit keying material and is given by:

$$\text{KDF}(\text{K\_in}, \text{label}, \text{seed}) = \text{HMAC\_256}(\text{K\_in}, \text{0x01} \mid \text{label} \mid \text{0x00} \mid \text{seed} \mid \text{0x01} \mid \text{0x00}),$$

where

- o K\_in -- derivation key,
- o label, seed -- the parameters, fixed and assigned by a protocol.

The key derivation function KDF\_GOSTR3411\_2012\_256 is a special case of KDF\_TREE\_GOSTR3411\_2012 function, described in the next section.

## 4.6. Key derivation function KDF\_TREE\_GOSTR3411\_2012\_256

The key derivation function KDF\_TREE\_GOSTR3411\_2012\_256 based on HMAC\_256 and is given by:

$$\text{KDF\_TREE} (K_{in}, \text{label}, \text{seed}, R) = K(1) | K(2) | K(3) | K(4) | \dots,$$
$$K(i) = \text{HMAC\_256} (K_{in}, [i]_2 | \text{label} | 0x00 | \text{seed} | [L]_2), i \geq 1,$$

where

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

K\_in derivation key;

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

[L]\_2 byte representation of L, in network byte order;

i iteration counter;

[i]\_2 byte representation of the iteration counter (in the network byte order), the number of bytes in the representation [i]\_2 is equal to R (no more than 4 bytes);

label, seed the parameters, fixed and assigned by a protocol.

The key derivation function KDF\_TREE\_GOSTR3411\_2012\_256 is intended for generating a keying material in size of L, not exceeding  $256 \cdot (2^{8 \cdot R} - 1)$  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]. HMAC\_256 algorithm with 256-bit output described in Section 4.1 is selected as a pseudorandom function.

When  $R = 1$  and  $L = 256$  the function KDF\_TREE\_GOSTR3411\_2012\_256 is equivalent to KDF\_GOSTR3411\_2012\_256 from the previous section.

Each key derived from the keying material formed using the derivation key K\_in (0-level key) may be a 1-level diversification 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 key 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 the specific protocols.

#### 4.7. Key wrap and unwrap

Wrapped representation of the secret key  $K$  (GOST R 34.10-2012 [GOST3410-2012] key or GOST 28147-89 [GOST28147-89] key) is formed as follows by using a given export key  $K_e$  (GOST 28147-89 [GOST28147-89] key) and a random UKM vector from 8 to 16 bytes in size:

1. Generate a random UKM vector.
2. With the key derivation function, using export key  $K_e$  as a derivation key, and a UKM vector as the value of seed, generate a key, denoted by  $KEK_e$  (UKM), where

$$KEK_e \text{ (UKM)} = \text{KDF} (K_e, \text{label}, \text{UKM}),$$

where KDF function (see previous section) is used as a key derivation function for the fixed value

$$\text{label} = (0x26 \mid 0xBD \mid 0xB8 \mid 0x78).$$

3. MAC value GOST 28147-89 (4-byte) for the data  $K$  and the key  $KEK_e$  (UKM) is calculated, initialization vector (IV) in this case is equal to the first 8 bytes of UKM. The resulting value is denoted as  $CEK\_MAC$ .
4. The key  $K$  is encrypted by the GOST 28147-89 algorithm in the Electronic Codebook (ECB) mode with the key  $KEK_e$  (UKM). The encoding result is denoted as  $CEK\_ENC$ .
5. The wrapped representation of the key is considered ( $UKM \mid CEK\_ENC \mid CEK\_MAC$ ).

and the seed value that is equal to UKM.

During the key import the value of key  $K$  is restored as follows from the wrapped representation of the key (GOST R 34.10-2012 [GOST3410-2012] key or GOST 28147-89 key [GOST28147-89] key) and the export key  $K_e$ :

1. From the wrapped representation of the key selects the sets UKM,  $CEK\_ENC$ , and  $CEK\_MAC$ .

2. With the key derivation function, using the export key  $K_e$  as a derivation key, and a random UKM value as the value of seed, generates a key, denoted by  $KEK_e(UKM)$ , where

$$KEK_e(UKM) = KDF(K_e, label, UKM).$$

3. The  $CEK_{ENC}$  set is decrypted by the GOST 28147-89 algorithm in the Electronic Codebook (ECB) mode with the key  $KEK_e(UKM)$ . The unwrapped key  $K$  is assumed to be equal to the result of decryption.
4. MAC value GOST 28147-89 (4-byte) for the data  $K$  and the key  $KEK_e(UKM)$  is calculated, initialization vector (IV) in this case is equal to the first 8 bytes of UKM. If the result does not equal to  $CEK_{MAC}$ , an error is returned.

The algorithms for wrapping and unwrapping of the GOST R 34.10-2012 [GOST3410-2012] keys are modifications of the CryptoPro Key Wrap and CryptoPro Key Unwrap algorithms, described in Sections 6.3 and 6.4 of [RFC4357].

## 5. Acknowledgments

Smyslov Valery and Dmitry Belyavsky have provided useful comments and suggestions on early drafts.

## 6. References

### 6.1. Normative References

[GOST28147-89]

Gosudarstvennyi Standard of USSR, Government Committee of the USSR for Standards, "Systems of information processing. Cryptographic data security. Algorithms of cryptographic transformation", GOST 28147-89, 1989.

[GOST3410-2012]

Federal Agency on Technical Regulating and Metrology, "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, "Information technology. Cryptographic Data Security. Hashing function", GOST R 34.11-2012, 2012.



- [GOST3411-94] Federal Agency on Technical Regulating and Metrology, "Information technology. Cryptographic Data Security. Hashing function", GOST R 34.11-94, 1994.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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>>.

## 6.2. Informative References

- [NISTSP800-108] National Institute of Standards and Technology, "Recommendation for Key Derivation Using Pseudorandom Functions", NIST SP 800-108, October 2009.
- [RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, DOI 10.17487/RFC2246, January 1999, <<http://www.rfc-editor.org/info/rfc2246>>.
- [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>>.
- [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>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

- [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>>.
- [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>>.

## Appendix A. Values of 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 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 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 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 3

INTEGER

00 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  
 6C

INTEGER

00 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  
 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  
 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 2

INTEGER

00 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 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 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
    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 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 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
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\_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\_256(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 keys (512-bit output) 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  $x \cdot P$  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  $y \cdot P$  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 keys (512-bit output) 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  $x \cdot P$  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  $y \cdot P$  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<sub>in</sub> 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

13) Key wrap and unwrap with the szOID\_Gost28147\_89\_TC26\_Z\_ParamSet parameters

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

UKM value:

af 21 43 41 45 65 63 78

Label:

26 bd b8 78f

KEK\_e(UKM) = KDF(K\_e, label, UKM):

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:

38 d5 8a a3

CEK\_ENC:

b9 fb 92 42 95 0f 84 3f 0f bd 5b 9a 5e cf 9f 17  
f7 9e 6d 21 58 16 56 de 6d c5 85 dd 62 7a 44 0a

#### Authors' Addresses

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

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

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

Email: alekseev@cryptopro.ru

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

Email: oshkin@cryptopro.ru

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

Email: vpopov@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

Igor Ustinov  
Cryptocom  
14, Kedrova str., build 2  
Moscow 117218  
Russian Federation

Phone: +7 (499) 124-62-26  
Email: igus@cryptocom.ru