draft-smyshlyaev-tls12-gost-suites-11

Abstract

This document specifies three new cipher suites for the Transport Layer Security (TLS) protocol Version 1.2 to support the Russian cryptographic standard algorithms (called GOST algorithms).

This specification is developed to facilitate implementations that wish to support the GOST algorithms. This document does not imply IETF endorsement of the cipher suites.

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 https://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 December 2, 2021.

Copyright Notice

Copyright (c) 2021 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 (https://trustee.ietf.org/license-info) in effect on the date of
Table of Contents

1. Introduction ....................................................... 3
2. Conventions Used in This Document ................................. 4
3. Basic Terms and Definitions ...................................... 4
4. Cipher Suite Definitions .......................................... 5
   4.1. Record Payload Protection ................................... 6
       4.1.1. CTR_OMAC ............................................. 7
       4.1.2. CNT_IMIT ............................................. 8
   4.2. Key Exchange and Authentication ................................ 9
       4.2.1. Hello Messages ....................................... 10
       4.2.2. Server Certificate .................................. 11
       4.2.3. CertificateRequest .................................. 11
       4.2.4. ClientKeyExchange .................................. 12
           4.2.4.1. CTR_OMAC ..................................... 12
           4.2.4.2. CNT_IMIT ..................................... 14
       4.2.5. CertificateVerify ..................................... 16
       4.2.6. Finished ............................................. 17
   4.3. Cryptographic Algorithms ..................................... 17
       4.3.1. Block Cipher .......................................... 17
       4.3.2. MAC algorithm ....................................... 17
       4.3.3. Encryption algorithm ................................. 18
       4.3.4. PRF and HASH algorithms ............................. 18
       4.3.5. SNMAX parameter ..................................... 18
5. New Values for the SignatureAlgorithm Registry ................. 18
6. New Values for the Supported Groups Registry .................. 19
7. New Values for the ClientCertificateType Identifiers Registry 20
8. Additional Algorithms ............................................ 21
   8.1. TLSTREE ..................................................... 21
       8.1.1. Key Tree Parameters ................................. 21
   8.2. Key export and key import algorithms ....................... 22
       8.2.1. KEKp15 and KImp15 Algorithms ....................... 22
       8.2.2. KEKp28147 and KImp28147 Algorithms ................ 23
   8.3. Key Exchange Generation Algorithms .......................... 24
       8.3.1. KEK Algorithm ....................................... 24
       8.3.2. KEK_28147 Algorithm ............................... 26
   8.4. gostIMIT28147 ............................................. 27
9. IANA Considerations .............................................. 27
10. Historical considerations ....................................... 29
11. Security Considerations ....................................... 30
12. References ..................................................... 30
1. Introduction

This document specifies three new cipher suites for the Transport Layer Security (TLS) Protocol Version 1.2 [RFC5246] to support the set of Russian cryptographic standard algorithms (called GOST algorithms). These cipher suites use the same hash algorithm GOST R 34.11-2012 [GOST3411-2012] (the English version can be found in [RFC6986]) and the same signature algorithm GOST R 34.10-2012 [GOST3410-2012] (the English version can be found in [RFC7091]) but use different encryption and MAC algorithms, so they are divided into two types: the CTR_OMAC cipher suites and the CNT_IMIT cipher suite.

The CTR_OMAC cipher suites use the GOST R 34.12-2015 [GOST3412-2015] block ciphers (the English version can be found in [RFC7801]).

The CNT_IMIT cipher suite uses the GOST 28147-89 [GOST28147-89] block cipher (the English version can be found in [RFC5830]).

This document specifies cipher suites only for the TLS protocol version 1.2. The cipher suites for the TLS protocol version 1.3 [RFC8446] to support the set of Russian cryptographic standard algorithms are specified in a separate document [DraftGostTLS13].
2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Basic Terms and Definitions

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

- \( B_t \) is the set of byte strings of length \( t \), \( t \geq 0 \), for \( t = 0 \) the \( B_t \) set consists of a single empty string of zero length. If \( A \) is an element of \( B_t \), then \( A = (a_1, a_2, \ldots, a_t) \), where \( a_1, a_2, \ldots, a_t \) are in \{0, \ldots, 255\};

- \( B^* \) is the set of all byte strings of a finite length (hereinafter referred to as strings), including the empty string;

- \( A[i..j] \) is the string \( A[i..j] = (a_i, a_{i+1}, \ldots, a_j) \) in \( B_{(j-i+1)} \) where \( A = (a_1, \ldots, a_t) \) in \( B_t \) and \( 1 \leq i \leq j \leq t \);

- \( |A| \) is the length of the byte string \( A \) in bytes;

- \( A \ | \ C \) is the concatenation of strings \( A \) and \( C \) both belonging to \( B^* \), i.e., a string in \( B_{(|A|+|C|)} \), where the left substring in \( B_{|A|} \) is equal to \( A \), and the right substring in \( B_{|C|} \) is equal to \( C \);

- \( A \ \text{XOR} \ C \) is bitwise exclusive-or of byte strings \( A \) and \( C \) both belonging to \( B_t \) (i.e. both are of length \( t \) bytes), i.e., a string in \( B_t \) such that if \( A = (a_1, a_2, \ldots, a_t) \), \( C = (c_1, c_2, \ldots, c_t) \) then \( A \ \text{XOR} \ C = (a_1 \ \text{xor} \ c_1, a_2 \ \text{xor} \ c_2, \ldots, a_t \ \text{xor} \ c_t) \) where \( \text{xor} \) is bitwise exclusive-or of bytes;

- \( i \ & \ j \) is bitwise AND of integers \( i \) and \( j \);

- \( \text{STR}_t \) is the transformation that maps an integer \( i = 256^{(t-1)} \times i_1 + \ldots + 256 \times i_{(t-1)} + i_t \) into the byte string \( \text{STR}_t(i) = (i_1, \ldots, i_t) \) in \( B_t \) (the interpretation of the integer as a byte string in big-endian format);

- \( \text{str}_t \) is the transformation that maps an integer \( i = 256^{(t-1)} \times i_t + \ldots + 256 \times i_2 + i_1 \) into the byte string \( \text{str}_t(i) = (i_1, \ldots, i_t) \) in \( B_t \) (the interpretation of the integer as a byte string in little-endian format);
INT    the transformation that maps a string \( a = (a_1, \ldots, a_t) \) in \( B_t \) into the integer \( \text{INT}(a) = 256^{t-1} \times a_1 + \ldots + 256 \times a_{t-1} + a_t \) (the interpretation of the byte string in big-endian format as an integer);

\text{int}    the transformation that maps a string \( a = (a_1, \ldots, a_t) \) in \( B_t \) into the integer \( \text{int}(a) = 256^{t-1} \times a_t + \ldots + 256 \times a_2 + a_1 \) (the interpretation of the byte string in little-endian format as an integer);

\( k \)    the length of the block cipher key in bytes;
\n\( n \)    the length of the block cipher block in bytes;
\n\( Q_c \)    the public key stored in the client's certificate;
\n\( d_c \)    the private key that corresponds to the \( Q_c \) key;
\n\( Q_s \)    the public key stored in the server's certificate;
\n\( d_s \)    the private key that corresponds to the \( Q_s \) key;
\n\( q_s \)    an order of a cyclic subgroup of elliptic curve points group containing point \( Q_s \);
\n\( P_s \)    the point of order \( q_s \) that belongs to the same curve as \( Q_s \);
\n\( r_c \)    the random string contained in ClientHello.random field (see [RFC5246]);
\n\( r_s \)    the random string contained in ServerHello.random field (see [RFC5246]).

4. Cipher Suite Definitions

This document specifies the CTR_OMAC cipher suites and the CNT_IMIT cipher suite.

The CTR_OMAC cipher suites have the following values:

\[
\begin{align*}
\text{TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC} &= \{0x\text{C1}, 0x00\}; \\
\text{TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC} &= \{0x\text{C1}, 0x01\}.
\end{align*}
\]

The CNT_IMIT cipher suite has the following value:

\[
\text{TLS_GOSTR341112_256_WITH_28147_CNT_IMIT} = \{0x\text{C1}, 0x02\}.
\]
4.1. Record Payload Protection

The compression method (see Section 6.2.2 of [RFC5246]) MUST be "null" in all of the cipher suites described in this document. This compression method is negotiated according to Section 4.2.1. Note that the CompressionMethod.null operation is an identity operation; no fields of the TLSCompressed structure are altered.

All of the cipher suites described in this document use the stream cipher (see Section 4.3.3) to protect records. The TLSCipherText structure for the CTR_OMAC and CNT_IMIT cipher suites is specified in accordance with the Standard Stream Cipher case (see Section 6.2.3.1 of [RFC5246]):

```
struct {
    ContentType type;
    ProtocolVersion version;
    uint16 length;
    GenericStreamCipher fragment;
} TLSCipherText;
```

where TLSCipherText.fragment is generated in accordance with Section 4.1.1 when the CTR_OMAC cipher suite is used and Section 4.1.2 when the CNT_IMIT cipher suite is used.

The connection key material is a key material that consists of the sender_write_key (either the client_write_key or the server_write_key), the sender_write_MAC_key (either the client_write_MAC_key or the server_write_MAC_key) and the sender_write_IV (either the client_write_IV or the server_write_IV) parameters that are generated in accordance with Section 6.3 of [RFC5246].

The record key material is a key material that is generated from the connection key material and is used to protect a record with the certain sequence number. Note that in some cipher suites defined in this document the record key material can be equal to the connection key material.

In this section the TLSCipherText.fragment generation is described for one particular endpoint (server or client) with the corresponding connection key material and record key material.
4.1.1.  CTR_OMAC

In case of the CTR_OMAC cipher suites the record key material differs from the connection key material, and for the sequence number seqnum consists of:

- K_ENC_seqnum in B_k;
- K_MAC_seqnum in B_k;
- IV_seqnum in B_{n/2}.

The K_ENC_seqnum and K_MAC_seqnum values are calculated using the TLSTREE function defined in Section 8.1, the connection key material and the sequence number seqnum. IV_seqnum is calculated by adding seqnum value to sender_write_IV modulo 2^{(n/2)*8}:

- K_ENC_seqnum = TLSTREE(sender_write_key, seqnum);
- K_MAC_seqnum = TLSTREE(sender_write_MAC_key, seqnum);
- IV_seqnum = STR_{n/2}((INT(sender_write_IV) + seqnum) mod 2^{(n/2)*8}).

The TLSCompressed.fragment that corresponds to the sequence number seqnum is equal to the ENCVAlu__se_num.value that is calculated as follows:

1. The MACValue_seqnum value is generated using the MAC algorithm (see Section 4.3.2) similar to Section 6.2.3.1 of [RFC5246] except the sender_write_MAC_key is replaced by the K_MAC_seqnum key:

   MACValue_seqnum = MAC(K_MAC_seqnum, STR_8(seqnum) | type_seqnum | version_seqnum | length_seqnum | fragment_seqnum),

   where type_seqnum, version_seqnum, length_seqnum, fragment_seqnum are the TLSCompressed.type, TLSCompressed.version, TLSCompressed.length and TLSCompressed.fragment values of the record with the seqnum sequence number.

2. The entire data with the MACValue is encrypted with the ENC stream cipher (see Section 4.3.3):

   ENCVAlu__se_num.value = ENC(K_ENC_seqnum, IV_seqnum, fragment_seqnum | MACValue_seqnum),

   where fragment_seqnum is the TLSCompressed.fragment value of the record with the seqnum sequence number.
4.1.2. CNTIMIT

In case of the CNTIMIT cipher suite the record key material is equal to the connection key material and consists of:

- sender_write_key in B_k;
- sender_write_MAC_key in B_k;
- sender_write_IV in B_n.

The TLSCiphertext.fragment that corresponds to the sequence number seqnum is equal to the ENCValue_seqnum value that is calculated as follows:

1. The MACValue_seqnum value is generated by the MAC algorithm (see Section 4.3.2) as follows:

   \[
   MACValue_seqnum = MAC(sender_write_MAC_key, STR_8(0) | type_0 | version_0 | length_0 | fragment_0 | ... | STR_8(seqnum) | type_seqnum | version_seqnum | length_seqnum | fragment_seqnum),
   \]

   where type_i, version_i, length_i, fragment_i, i in \{0, ..., seqnum\}, are the TLSCompressed.type, TLSCompressed.version, TLSCompressed.length and TLSCompressed.fragment values of the record with the i sequence number.

   Due to the use of the CBC-MAC based mode (see Section 4.3.2) producing the MACValue_seqnum value does not mean processing all previous records. It is enough to store only an intermediate internal state of the MAC algorithm.

2. The entire data with the MACValue is encrypted with the ENC stream cipher (see Section 4.3.3):

   \[
   ENCValue_0 | ... | ENCValue_seqnum = ENC(sender_write_key, sender_write_IV, fragment_0 | MACValue_0 | ... | fragment_seqnum | MACValue_seqnum),
   \]

   where the length of the byte string ENCValue_i in bytes is equal to the length of the byte string (fragment_i | MACValue_i) in bytes, i in \{0, ..., seqnum\}.

   Due to the use of the stream cipher (see Section 4.3.3) producing the ENCValue_seqnum value does not mean processing all previous records. It is enough to store only an intermediate internal state of the ENC stream cipher.
4.2. Key Exchange and Authentication

All of the cipher suites described in this document use a key encapsulation mechanism based on Diffie-Hellman to share the TLS premaster secret.

```
Client                                               Server
ClientHello                                        <-------->
                                               ServerHello
                                               Certificate
                                               CertificateRequest*
<--------      ServerHelloDone
                                               Certificate*
                                               ClientKeyExchange
                                               CertificateVerify*
[ChangeCipherSpec]
Finished                                        <-------->
                                               [ChangeCipherSpec]
                                               Finished
Application Data                                Application Data
```

Figure 1: Message flow for a full handshake.

* Indicates optional messages that are sent for the client authentication.

Note: To help avoid pipeline stalls, ChangeCipherSpec is an independent TLS protocol content type, and is not actually a TLS handshake message.

Figure 1 shows all messages involved in the TLS key establishment protocol (full handshake). A ServerKeyExchange MUST NOT be sent (the server's certificate contains enough data to allow client to exchange the premaster secret).

The server side of the channel is always authenticated; the client side is optionally authenticated. The server is authenticated by proving that it knows the premaster secret that is encrypted with the public key Q_s from the server's certificate. The client is authenticated via its signature over the handshake transcript.

In general the key exchange process for both CTR_OMAC and CNT_IMIT cipher suites consists of the following steps:
1. The client generates the ephemeral key pair \((d_{eph}, Q_{eph})\) that corresponds to the server's public key \(Q_s\) stored in its certificate.

2. The client generates the premaster secret \(PS\). The \(PS\) value is chosen from \(B_{32}\) at random.

3. Using \(d_{eph}\) and \(Q_s\) the client generates the export key material (see Section 4.2.4.1 and Section 4.2.4.2) for the particular key export algorithm (see Section 8.2.1 and Section 8.2.2) to generate the export representation \(PS_{Exp}\) of the \(PS\) value.

4. The client sends its ephemeral public key \(Q_{eph}\) and \(PS_{Exp}\) value in the ClientKeyExchange message.

5. Using its private key \(d_s\) the server generates the import key material (see Section 4.2.4.1 and Section 4.2.4.2) for the particular key import algorithm (see Section 8.2.1 and Section 8.2.2) to extract the premaster secret \(PS\) from the export representation \(PS_{Exp}\).

The cipher suites specified in this document define the ClientHello, ServerHello, server Certificate, CertificateRequest, ClientKeyExchange, CertificateVerify and Finished handshake messages, that are described in further detail below.

### 4.2.1. Hello Messages

The ClientHello message is generated in accordance with Section 7.4.1.1. of [RFC5246] and must meet the following requirements:

- The ClientHello.compression_methods field MUST contain exactly one byte, set to zero, which corresponds to the "null" compression method.

- The ClientHello.extensions field MUST contain the signature_algorithms extension (see [RFC5246]).

   If the negotiated cipher suite is one of CTR_OMAC/CTR_IMIT and the client implementation does not support generating the signature_algorithms extension with the values defined in Section 5, the server MUST either abort the connection or ignore this extension and behave as if the client had sent the signature_algorithms extension with the values \{8, 64\} and \{8, 65\}. 

The ServerHello message is generated in accordance with Section 7.4.1.2. of [RFC5246] and must meet the following requirements:

- The ServerHello.compression_method field MUST contain exactly one byte, set to zero, which corresponds to the "null" compression method.
- The ServerHello.extensions field MUST NOT contain the encrypt_then_mac extension (see [RFC7366]).

4.2.2. Server Certificate

This message is used to authentically convey the server's public key $Q_s$ to the client and is generated in accordance with Section 7.4.2 of [RFC5246].

Upon receiving this message the client validates the certificate chain, extracts the server's public key, and checks that the key type is appropriate for the negotiated key exchange algorithm. (A possible reason for a fatal handshake failure is that the client's capabilities for handling elliptic curves and point formats are exceeded).

4.2.3. CertificateRequest

This message is sent by the server when requesting client authentication and is specified in accordance with [RFC5246] as follows.

struct {
    ClientCertificateType certificate_types<1..2^8-1>;
    SignatureAndHashAlgorithm
        supported_signature_algorithms<2..2^16-2>;
    DistinguishedName certificateAuthorities<0..2^16-1>;
} CertificateRequest;

If the CTR_OMAC or CNT_IMIT cipher suite is negotiated, the CertificateRequest message MUST meet the following requirements:

- the CertificateRequest.supported_signature_algorithm field MUST contain only signature/hash algorithm pairs with the values {8, 64} or {0, 65} defined in Section 5;
the CertificateRequest.certificate_types field MUST contain only
the gost_sign256 (67) or gost_sign512 (68) values defined in
Section 7.

4.2.4. ClientKeyExchange

The ClientKeyExchange message is defined as follows.

```c
enum { vko_kdf_gost, vko_gost } KeyExchangeAlgorithm;

struct {
    select (KeyExchangeAlgorithm) {
        case vko_kdf_gost: GostKeyTransport;
        case vko_gost: TLSGostKeyTransportBlob;
    } exchange_keys;
} ClientKeyExchange;
```

The body of the ClientKeyExchange message consists of a
GostKeyTransport/TLSGostKeyTransportBlob structure that contains an
export representation of the premaster secret PS.

The GostKeyTransport structure corresponds to the CTR_OMAC cipher
suites and is described in Section 4.2.4.1 and the
TLSGostKeyTransportBlob corresponds to CNT_IMIT cipher suite and is
described in Section 4.2.4.2.

4.2.4.1. CTR_OMAC

In case of the CTR_OMAC cipher suites the body of the
ClientKeyExchange message consists of the GostKeyTransport structure
that is defined below.

The client generates the ClientKeyExchange message in accordance with
the following steps:

1. Generates the ephemeral key pair (Q_eph, d_eph), where:
   
   d_eph is chosen from \{1, ..., q_s - 1\} at random;

   Q_eph = d_eph * P_s.

2. Generates export keys (K_EXP_MAC and K_EXP_ENC) using the KEG
   algorithm defined in Section 8.3.1:

   \( H = \text{HASH}(r_c \mid r_s) \);
K_EXP_MAC | K_EXP_ENC = KEG(d_eph, Q_s, H).

3. Generates an export representation PSExp of the premaster secret PS using the KExp15 algorithm defined in Section 8.2.1:

\[ IV = H[25..24 + n / 2]; \]

\[ PSExp = KExp15(PS, K_EXP_MAC, K_EXP_ENC, IV). \]

4. Generates the ClientKeyExchange message using the GostKeyTransport structure that is defined as follows:

```
GostKeyTransport ::= SEQUENCE {
  keyExp               OCTET STRING,
  ephemeralPublicKey   SubjectPublicKeyInfo,
  ukm                  OCTET STRING OPTIONAL
}

SubjectPublicKeyInfo ::= SEQUENCE {
  algorithm            AlgorithmIdentifier,
  subjectPublicKey     BIT STRING
}

AlgorithmIdentifier ::= SEQUENCE {
  algorithm            OBJECT IDENTIFIER,
  parameters           ANY OPTIONAL
}
```

where the keyExp field contains the PSExp value, the ephemeralPublicKey field contains the Q_eph value and the ukm field MUST be ignored by the server.

Upon receiving the ClientKeyExchange message, the server process it as follows.

1. Checks the following three conditions. If either of these checks fails, then the server MUST abort the handshake with an alert.
   - Q_eph belongs to the same curve as server public key Q_s;
   - Q_eph is not equal to zero point;
   - q_s * Q_eph is equal to zero point.

2. Generates export keys (K_EXP_MAC and K_EXP_ENC) using the KEG algorithm defined in Section 8.3.1:
H = HASH(r_c | r_s);

K_EXP_MAC | K_EXP_ENC = KEG(d_s, Q_eph, H).

3. Extracts the premaster secret PS from the export representation PSExp using the KImp15 algorithm defined in Section 8.2.1:

IV = H[25..24 + n / 2];

PS = KImp15(PSExp, K_EXP_MAC, K_EXP_ENC, IV).

### 4.2.4.2. CNTIMIT

In case of the CNTIMIT cipher suite the body of the ClientKeyExchange message consists of a TLSGostKeyTransportBlob structure that is defined below.

The client generates the ClientKeyExchange message in accordance with the following steps:

1. Generates the ephemeral key pair (Q_eph, d_eph), where:
   - d_eph is chosen from \(\{1, \ldots, q_s - 1\}\) at random;
   - \(Q_{eph} = d_{eph} \cdot P_s\).

2. Generates export key (K_EXP) using the KEG28147 algorithm defined in Section 8.3.2:

   H = HASH(r_c | r_s);

   K_EXP = KEG28147(d_eph, Q_s, H).

3. Generates an export representation PSExp of the premaster secret PS using the KExp28147 algorithm defined in Section 8.2.2:

   PSExp = IV | CEK_ENC | CEK_MAC = KExp28147(PS, K_EXP, H[1..8]).

4. Generates the ClientKeyExchange message using the TLSGostKeyTransportBlob structure that is defined as follows:
Internet-Draft       GOST Cipher Suites for TLS 1.2             May 2021

TL SGostKeyTransportBlob ::= SEQUENCE {
    keyBlob               GostR3410-KeyTransport,
}

GostR3410-KeyTransport ::= SEQUENCE {
    sessionEncryptedKey  Gost28147-89-EncryptedKey,
    transportParameters  [0] IMPLICIT GostR3410-TransportParameters
                   OPTIONAL
}

Gost28147-89-EncryptedKey ::= SEQUENCE {
    encryptedKey         Gost28147-89-Key,
    maskKey              [0] IMPLICIT Gost28147-89-Key OPTIONAL,
    macKey               Gost28147-89-MAC
}

GostR3410-TransportParameters ::= SEQUENCE {
    encryptionParamSet   OBJECT IDENTIFIER,
    ephemeralPublicKey   [0] IMPLICIT SubjectPublicKeyInfo OPTIONAL,
    ukm                  OCTET STRING
}

where GostR3410-KeyTransport, Gost28147-89-EncryptedKey and
GostR3410-TransportParameters are defined according to Section 4.2.1.
of [RFC4490].

In the context of this document the
GostR3410-KeyTransport.transportParameters field is always used, the
Gost28147-89-EncryptedKey.maskKey field is omitted, the
GostR3410-KeyTransport.transportParameters.ephemeralPublicKey field
is always used.

The Gost28147-89-EncryptedKey.encryptedKey field contains the CEK_ENC
value, the Gost28147-89-EncryptedKey.macKey field contains the
CEK_MAC value, and GostR3410-TransportParameters.ukm field contains
the IV value.

The keyBlob.transportParameters.ephemeralPublicKey field contains the
client ephemeral public key Q_eph. The encryptionParamSet contains
value 1.2.643.7.1.2.5.1.1 that corresponds to the id-tc26-gost-
28147-param-Z parameters set defined in [RFC7836].

Upon receiving the ClientKeyExchange message, the server process it
as follows.

1. Checks the following three conditions. If either of these checks
   fails, then the server MUST abort the handshake with an alert.
   
   o Q_eph belongs to the same curve as server public key Q_s;
2. Generates export key (K_EXP) using the KEG_28147 algorithm defined in Section 8.3.2:

\[ H = \text{HASH}(r_c | r_s); \]

\[ K_{\text{EXP}} = \text{KEG}_{28147}(d_s, Q_{\text{eph}}, H). \]

3. Extracts the premaster secret PS from the export representation PSExp using the KImp28147 algorithm defined in Section 8.2.2:

\[ PS = \text{KImp28147}(PSExp, K_{\text{EXP}}, H[1..8]). \]

### 4.2.5. CertificateVerify

Client generates the value sgn as follows:

\[ sgn = \text{SIGN}_{d_c}(\text{handshake_messages}) = \text{str}_l(r) \mid \text{str}_l(s) \]

where \( \text{SIGN}_{d_c} \) is the GOST R 34.10-2012 [RFC7091] signature algorithm, \( d_c \) is a client long-term private key that corresponds to the client long-term public key \( Q_c \) from the client's certificate, \( l = 32 \) for gostr34102012_256 value of the SignatureAndHashAlgorithm field and \( l = 64 \) for gostr34102012_512 value of the SignatureAndHashAlgorithm field.

Here \( \text{handshake_messages} \) refers to all handshake messages sent or received, starting at ClientHello and up to CertificateVerify, but not including the last message, including the type and length fields of the handshake messages.

The TLS CertificateVerify message is specified as follows.

```c
struct {
    SignatureAndHashAlgorithm algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

where SignatureAndHashAlgorithm structure is specified in Section 5 and CertificateVerify.signature field contains sgn value.
4.2.6. Finished

The TLS Finished message is specified as follows.

```c
struct {
    opaque verify_data[verify_data_length];
} Finished;
```

```
verify_data = PRF(master_secret, finished_label, HASH(handshake_messages))[0..verify_data_length-1];
```

where the `verify_data_length` value is equal to 32 for the CTR_OMAC cipher suites and is equal to 12 for the CNT_IMIT cipher suite. The PRF function is defined in Section 4.3.4.

4.3. Cryptographic Algorithms

4.3.1. Block Cipher

The cipher suite TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC MUST use Kuznyechik [RFC7801] as a base block cipher for the encryption and MAC algorithm. The block length `n` is 16 bytes and the key length `k` is 32 bytes.

The cipher suite TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC MUST use Magma [GOST3412-2015] as a base block cipher for the encryption and MAC algorithm. The block length `n` is 8 bytes and the key length `k` is 32 bytes.

The cipher suite TLS_GOSTR341112_256_WITH_28147_CNT_IMIT MUST use GOST 28147-89 as a base block cipher [RFC5830] with the set of parameters id-tc26-gost-28147-param-Z defined in [RFC7836]. The block length `n` is 8 bytes and the key length `k` is 32 bytes.

4.3.2. MAC algorithm

The CTR_OMAC cipher suites use the OMAC message authentication code construction defined in [GOST3413-2015], which can be considered as the CMAC mode defined in [CMAC] where Kuznyechik or Magma block cipher (see Section 4.3.1) are used instead of AES block cipher (see [IK2003] for more detail) as the MAC function. The resulting MAC length is equal to the block length and the MAC key length is 32 bytes.

The CNT_IMIT cipher suite uses the message authentication code function gostIMIT28147 defined in Section 8.4 with the initialization
vector IV = IV₀, where IV₀ in B₈ is a string of all zeros, with the CryptoPro Key Meshing algorithm defined in [RFC4357]. The resulting MAC length is 4 bytes and the MAC key length is 32 bytes.

### 4.3.3. Encryption algorithm

The CTR_OMAC cipher suites use the block cipher in CTR-ACPKM encryption mode defined in [RFC8645] as the ENC function. The section size N is 4 KB for TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC cipher suite and 1 KB for TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC cipher suite.

The CNT_IMIT cipher suite uses the block cipher in counter encryption mode (CNT) defined in Section 6 of [RFC5830] with the CryptoPro Key Meshing algorithm defined in [RFC4357] as the ENC function.

### 4.3.4. PRF and HASH algorithms

The pseudorandom function (PRF) for all the cipher suites defined in this document is the PRF_TLS_GOSTR3411_2012_256 function defined in [RFC7836].

The hash function HASH for all the cipher suites defined in this document is the GOST R 34.11-2012 [RFC6986] hash algorithm with 32-byte (256-bit) hash code.

### 4.3.5. SNMAX parameter

The SNMAX parameter defines the maximal value of the sequence number seqnum during one TLS 1.2 connection and is defined as follows:

```
+---------------------------------------------+--------------------+
|               CipherSuites                  |        SNMAX       |
+---------------------------------------------+--------------------+
|TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC | SNMAX = 2^64 - 1   |
|TLS_GOSTR341112_256_WITH_28147_CNT_IMIT      |                    |
+---------------------------------------------+--------------------+
|TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC      | SNMAX = 2^32 - 1   |
+---------------------------------------------+--------------------+
```

Table 1

### 5. New Values for the SignatureAlgorithm Registry

The signature/hash algorithm pairs are used to indicate to the server/client which algorithms can be used in digital signatures and
are defined by the SignatureAndHashAlgorithm structure (see Section 7.4.1.4.1 of [RFC5246]) as follows:

```c
struct {
    HashAlgorithm hash;
    SignatureAlgorithm signature;
} SignatureAndHashAlgorithm;
```

This document defines new values for the "SignatureAlgorithm Registry" that can be used in the SignatureAndHashAlgorithm.signature field for the particular signature/hash algorithm pair:

```c
enum {
    gostr34102012_256(64),
    gostr34102012_512(65),
} SignatureAlgorithm;
```

where the gostr34102012_256 and gostr34102012_512 values correspond to the GOST R 34.10-2012 [RFC7091] signature algorithm with 32-byte (256-bit) and 64-byte (512-bit) key length respectively.

According to [RFC7091] the GOST R 34.10-2012 signature algorithm with 32-byte (256-bit) or 64-byte (512-bit) key length use the GOST R 34.11-2012 [RFC6986] hash algorithm with 32-byte (256-bit) or 64-byte (512-bit) hash code respectively (the hash algorithm is intrinsic to the signature algorithm). Therefore, if the SignatureAndHashAlgorithm.signature field of a particular hash/signature pair listed in the Signature Algorithms Extension is equal to the 64 (gostr34102012_256) or 65 (gostr34102012_512) value, the SignatureAndHashAlgorithm.hash field of this pair MUST contain the "Intrinsic" value 8 (see [RFC8422]).

6. New Values for the Supported Groups Registry

The Supported Groups Extension indicates the set of elliptic curves supported by the client and is defined in [RFC8422] and [RFC7919].

This document defines new values for the "Supported Groups" registry:
enum {
    GC256A(34), GC256B(35), GC256C(36), GC256D(37),
    GC512A(38), GC512B(39), GC512C(40),
} NamedGroup;

Where the values corresponds to the following curves:

<table>
<thead>
<tr>
<th>Description</th>
<th>Curve Identifier Value</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>GC256A</td>
<td>id-tc26-gost-3410-2012-256-paramSetA</td>
<td>RFC 7836</td>
</tr>
<tr>
<td>GC256B</td>
<td>id-GostR3410-2001-CryptoPro-A-ParamSet</td>
<td>RFC 4357</td>
</tr>
<tr>
<td>GC256C</td>
<td>id-GostR3410-2001-CryptoPro-B-ParamSet</td>
<td>RFC 4357</td>
</tr>
<tr>
<td>GC256D</td>
<td>id-GostR3410-2001-CryptoPro-C-ParamSet</td>
<td>RFC 4357</td>
</tr>
<tr>
<td>GC512A</td>
<td>id-tc26-gost-3410-12-512-paramSetA</td>
<td>RFC 7836</td>
</tr>
<tr>
<td>GC512B</td>
<td>id-tc26-gost-3410-12-512-paramSetB</td>
<td>RFC 7836</td>
</tr>
<tr>
<td>GC512C</td>
<td>id-tc26-gost-3410-2012-512-paramSetC</td>
<td>RFC 7836</td>
</tr>
</tbody>
</table>

Table 2

7. New Values for the ClientCertificateType Identifiers Registry

The ClientCertificateType field of the CertificateRequest message contains a list of the types of certificate types that the client may offer and is defined in Section 7.4.4 of [RFC5246].

This document defines new values for the "ClientCertificateType Identifiers" registry:

enum {
    gost_sign256(67),
    gost_sign512(68),
} ClientCertificateType;

To use the gost_sign256 or gost_sign512 authentication mechanism, the client MUST possess a certificate containing a GOST R
The client proves possession of the private key corresponding to the certified key by including a signature in the CertificateVerify message as described in Section 4.2.5.

8. Additional Algorithms

8.1. TLSTREE

The TLSTREE function is defined as follows:

$$\text{TLSTREE}(K_{\text{root}}, i) = KDF_3(KDF_2(KDF_1(K_{\text{root}}, \text{STR}_8(i \& C_1)), \text{STR}_8(i \& C_2)), \text{STR}_8(i \& C_3)),$$

where

- $K_{\text{root}}$ in B_{32};
- $i$ in \{0, 1, ..., 2^{64} - 1\};
- $C_1$, $C_2$, $C_3$ are constants defined by the particular cipher suite (see Section 8.1.1);
- $KDF_j(K, D)$, $j = 1, 2, 3$, $K$ in B_{32}, $D$ in B_8, is the key derivation function based on the KDF_GOSTR3411_2012_256 function defined in [RFC7836]:

$$KDF_1(K, D) = KDF\text{_GOSTR3411\_2012\_256}(K, "level1", D);$$
$$KDF_2(K, D) = KDF\text{_GOSTR3411\_2012\_256}(K, "level2", D);$$
$$KDF_3(K, D) = KDF\text{_GOSTR3411\_2012\_256}(K, "level3", D).$$

8.1.1. Key Tree Parameters

The CTR_OMAC cipher suites use the TLSTREE function for the re-keying approach. The constants for it are defined as in the table below.
### 8.2. Key export and key import algorithms

#### 8.2.1. KExp15 and KImp15 Algorithms

Algorithms KExp15 and KImp15 use the block cipher determined by the particular cipher suite.

The KExp15 key export algorithm is defined as follows.

```plaintext
+--------------------------------------------+----------------------+
|               CipherSuites                |    C_1, C_2, C_3     |
+--------------------------------------------+----------------------+
|TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC|C_1=0xFFFFFFFF00000000|
|                                            |C_2=0xFFFFFFFFFFFFF80000|
|                                            |C_3=0xFFFFFFFFFFFFFC0|
+--------------------------------------------+----------------------+
|TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC    |C_1=0xFFFFFFC000000000|
|                                            |C_2=0xFFFFFFFFFE000000|
|                                            |C_3=0xFFFFFFFFFFFFF000|
+--------------------------------------------+----------------------+
```

Table 3

#### 8.2. Key export and key import algorithms

#### 8.2.1. KExp15 and KImp15 Algorithms

Algorithms KExp15 and KImp15 use the block cipher determined by the particular cipher suite.

The KExp15 key export algorithm is defined as follows.

```plaintext
+------------------------------------------------------------+----------------------+
|   KExp15(S, K_Exp_MAC, K_Exp_ENC, IV)                      |    C_1, C_2, C_3     |
+------------------------------------------------------------+----------------------+
|  Input:                                                    |                      |
|  - secret S to be exported, S in B*,                        |                      |
|  - key K_Exp_MAC in B_k,                                   |                      |
|  - key K_Exp_ENC in B_k,                                   |                      |
|  - IV in B_{n/2}                                           |                      |
|  Output:                                                   |                      |
|  - export representation SExp in B_{|S|+n}                 |                      |
+------------------------------------------------------------+----------------------+

1. CEK_MAC = OMAC(K_Exp_MAC, IV | S), CEK_MAC in B_n
2. SExp = CTR-Encrypt(K_Exp_ENC, IV, S | CEK_MAC)
3. return SExp

where the OMAC function is defined in \([MODES]\), the CTR-Encrypt(K, IV, S) function denotes the encryption of message S on key K and nonce IV in the CTR mode with s = n (see \([MODES]\)).

The KImp15 key import algorithm is defined as follows.
KImp15(SExp, K_Exp_MAC, K_Exp_ENC, IV)

Input:
- export representation SExp in B *
- key K_Exp_MAC in B_k,
- key K_Exp_ENC in B_k,
- IV in B_{n/2}

Output:
- secret S in B_{|SExp|-n} or FAIL

1. S | CEK_MAC = CTR-Decrypt(K_Exp_ENC, IV, SExp), CEK_MAC in B_n
2. If CEK_MAC = OMAC(K_Exp_MAC, IV | S)
   then return S; else return FAIL

where the OMAC function is defined in [MODES], the CTR-Decrypt(K, IV, S) function denotes the decryption of message S on key K and nonce IV in the CTR mode (see [MODES]).

The keys K_Exp_MAC and K_Exp_ENC MUST be independent. For every pair of keys (K_Exp_ENC, K_Exp_MAC) the IV values MUST be unique. For the import of key K with the KImp15 algorithm every IV value MUST be sent with the export key representation or be a preshared value.

8.2.2. KExp28147 and KImp28147 Algorithms

The KExp28147 key export algorithm is defined as follows.

KExp28147(S, K, IV)

Input:
- secret S to be exported, S in B_32,
- key K in B_32,
- IV in B_8.

Output:
- export representation SExp in B_44

1. CEK_MAC = gost28147IMIT(IV, K, S), CEK_MAC in B_4
2. CEK_ENC = ECB-Encrypt(K, S), CEK_ENC in B_32
3. return SExp = IV | CEK_ENC | CEK_MAC
where the gost28147IMIT function is defined in Section 8.4, the ECB-Encrypt(K, S) function denotes the encryption of message S on key K with the block cipher GOST 28147-89 in the ECB mode (see [RFC5830]).

The KImp28147 key import algorithm is defined as follows:

```
+----------------------------------------------------------------+
<table>
<thead>
<tr>
<th>KImp28147(SExp, K, IV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
</tr>
<tr>
<td>- export representation SExp in B_44,</td>
</tr>
<tr>
<td>- key K in B_32,</td>
</tr>
<tr>
<td>- IV in B_8.</td>
</tr>
<tr>
<td>Output:</td>
</tr>
<tr>
<td>- imported secret S in B_32 or FAIL</td>
</tr>
</tbody>
</table>
+----------------------------------------------------------------|

1. extract from SExp
   IV' = SExp[1..8],
   CEK_ENC = SExp[9..40],
   CEK_MAC = SExp[41..44]

2. if IV' != IV then return FAIL; else

3. S = ECB-Decrypt(K, CEK_ENC), S in B_32

4. If CEK_MAC = gost28147IMIT(IV, K, S)
   then return S; else return FAIL
```

where the gost28147IMIT function is defined in Section 8.4, the ECB-Decrypt(CEK_ENC, M) function denotes the decryption of ciphertext CEK_ENC on key K with a block cipher GOST 28147-89 in the ECB mode (see [RFC5830]).

### 8.3. Key Exchange Generation Algorithms

#### 8.3.1. KEG Algorithm

The KEG algorithm is defined as follows:
KEG(d, Q, H)

Input:
- private key d,
- public key Q,
- H in B_32.

Output:
- key material K in B_64.

1. If q * Q is not equal to zero point
   return FAIL
2. If 2^{254} < q < 2^{256}
   return KEG_256(d, Q, H)
3. If 2^{508} < q < 2^{512}
   return KEG_512(d, Q, H)
4. return FAIL

where q is an order of a cyclic subgroup of elliptic curve points group containing point Q, d in {1, ..., q - 1}.

The KEG_256 algorithm is defined as follows:

KEG_256(d, Q, H)

Input:
- private key d,
- public key Q,
- H in B_32.

Output:
- key material K in B_64.

1. r = INT(H[1..16])
2. If r = 0
   UKM = 1; else UKM = r
3. K_EXP = VKO_256(d, Q, UKM)
4. seed = H[17..24]
5. return KDFTREE_256(K_EXP, "kdf tree", seed, 1)

where VKO_256 is the function VKO_GOSTR3410_2012_256 defined in [RFC7836] and KDFTREE_256 is the KDF_TREE_GOSTR3411_2012_256 function defined in [RFC7836] with the parameter L equal to 512.

The KEG_512 algorithm is defined as follows:
8.3.2.  KEG_28147 Algorithm

The KEG_28147 algorithm is defined as follows:

1. If \( q * Q \) is not equal to zero point
   return FAIL
2. UKM = \( H[1..8] \)
3. \( R = \text{VKO}_256(d, Q, \text{int(UKM)}) \)
4. return \( K = \text{CPDivers(UKM, R)} \)

where the \( \text{VKO}_256 \) function is equal to the \( \text{VKO}_{\text{GOSTR3410}_2012}_256 \) function defined in [RFC7836], the \( \text{CPDivers} \) function corresponds to the CryptoPro KEK Diversification Algorithm defined in [RFC4357], which takes as input the UKM value and the key value.
8.4. gostIMIT28147

gost28147IMIT(IV, K, M) is a MAC algorithm with 4 bytes output and is defined as follows:

```
+----------------------------------------------------------------+
| gost28147IMIT(IV, K, M)                                      |
+----------------------------------------------------------------|
| Input:                                                        |
| - initial value IV in B_8,                                    |
| - key K in B_32,                                              |
| - message M in B*.                                            |
| Output:                                                       |
| - MAC value T in B_4.                                         |
+----------------------------------------------------------------|
| 1. M' = PAD(M)                                                |
| 2. M' = M'_0 | ... | M'_r, |M'_i| = 8, i in {0, ... , r}      |
| 3. M'' = (M'_0 XOR IV) | M'_1 | ... | M'_r |
| 4. return T = MAC28147(K, M'')                                |
+----------------------------------------------------------------|
```

where the PAD function is the padding function that adds m zero bytes to the end of the message, where m is the smallest, non-negative solution to the equation \(|M| + m\) mod 8 = 0, the MAC28147 function corresponds to Message Authentication Code Generation Mode defined in [RFC5830] with 4 byte length output.

9. IANA Considerations

IANA is asked to update the registry entries to reference this document when it is published as an RFC.

IANA has added numbers {0xC1, 0x00}, {0xC1, 0x01} and {0xC1, 0x02} with the names TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC, TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC, TLS_GOSTR341112_256_WITH_28147_CNT_IMIT to the "TLS Cipher Suite" registry with this document as reference, as shown below.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>DTLS-OK</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xC1, 0x00</td>
<td>TLS_GOSTR341112_256_ _WITH_KUZNYECHIK_CTR_OMAC</td>
<td>N</td>
<td>this RFC</td>
</tr>
<tr>
<td>0xC1, 0x01</td>
<td>TLS_GOSTR341112_256_ _WITH_MAGMA_CTR_OMAC</td>
<td>N</td>
<td>this RFC</td>
</tr>
<tr>
<td>0xC1, 0x02</td>
<td>TLS_GOSTR341112_256_ _WITH_28147_CNTIMIT</td>
<td>N</td>
<td>this RFC</td>
</tr>
</tbody>
</table>

Table 4

IANA has added numbers 64, 65 with the names gostr34102012_256, gostr34102012_512, to the "TLS SignatureAlgorithm" registry, as shown below.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>DTLS-OK</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>gostr34102012_256</td>
<td>Y</td>
<td>this RFC</td>
</tr>
<tr>
<td>65</td>
<td>gostr34102012_512</td>
<td>Y</td>
<td>this RFC</td>
</tr>
</tbody>
</table>

Table 5

IANA has added numbers 34, 35, 36, 37, 38, 39, 40 with the names GC256A, GC256B, GC256C, GC256D, GC512A, GC512B, GC512C to the "TLS Supported Groups" registry, as shown below.
Table 6

IANA has added numbers 67, 68 with the names gost_sign256, gost_sign512 to the "ClientCertificateType Identifiers" registry, as shown below.

Table 7

10. Historical considerations

Note that prior to the existence of this document implementations could use only the values from the Private Use space in order to use the GOST-based algorithms. So some old implementations can still use the old value \{0x00, 0x81\} instead of the \{0xC1, 0x02\} value to indicate the TLS_GOSTR341112_256_WITH_28147_CNT.IMIT cipher suite; one old value 0xEE instead of the values 64, 8 and 67 (to indicate the gostr34102012_256 signature algorithm, the Intrinsic hash algorithm and the gost_sign256 certificate type respectively); one old value 0xEF instead of the values 65, 8 and 68 (to indicate the gostr34102012_512 signature algorithm, the Intrinsic hash algorithm and the gost_sign512 certificate type respectively).
Due to historical reasons in addition to the curve identifier values listed in Table 2 there exist some extra identifier values that correspond to the curves GC256B, GC256C and GC256D as follows.

<table>
<thead>
<tr>
<th>Description</th>
<th>Curve Identifier Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>GC256B</td>
<td>id-GostR3410_2001-CryptoPro-XchA-ParamSet</td>
</tr>
<tr>
<td></td>
<td>id-tc26-gost-3410-2012-256-paramSetB</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>GC256C</td>
<td>id-tc26-gost-3410-2012-256-paramSetC</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>GC256D</td>
<td>id-GostR3410-2001-CryptoPro-XchB-ParamSet</td>
</tr>
<tr>
<td></td>
<td>id-tc26-gost-3410-2012-256-paramSetD</td>
</tr>
</tbody>
</table>

Table 8

Client should be prepared to handle any of them correctly if corresponding group is included in the supported_groups extension (see [RFC8422] and [RFC7919]).

11. Security Considerations

This entire document is about security considerations.

12. References

12.1. Normative References


12.2. Informative References


Appendix A.  Test Examples

A.1.  Test Examples for CTR_OMAC cipher suites

A.1.1.  TLSTREE Examples

A.1.1.1.  TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC ciphersuite

TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC
**************************************************
Root Key K_root:
00 11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A
11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00

seqnum = 0
First level key from Divers_1:
F3 55 89 F0 9B F8 01 B1 CA 11 42 73 B9 5F D6 C1
39 2E 78 F9 FB 81 4D A0 5A 7C CA 08 9E C8 65 42

Second level key from Divers_2:
51 37 D5 C4 A6 E6 BE 42 C4 40 D1 0A 95 EE A0 7F
08 9E 74 0D 38 90 EB 52 65 2C 0C B9 3F 20 7B B4
The resulting key from Divers 3:
19  A7  6E  D3  0F  4D  6D  1F  5B  72  63  EC  49  1A  D8  38
17  C0  B5  7D  8A  03  56  12  71  40  FB  4F  74  25  49  4D

seqnum = 4095
First level key from Divers_1:
F3  55  89  F0  9B  F8  01  B1  CA  11  42  73  B9  5F  D6  C1
39  2E  78  F9  FB  81  4D  A0  5A  7C  CA  08  9E  C8  65  42

Second level key from Divers_2:
51  37  D5  C4  A6  E6  BE  42  C4  40  D1  0A  95  EE  A0  7F
08  9E  74  0D  38  90  EB  52  65  2C  0C  B9  3F  20  7B  B4

The resulting key from Divers 3:
19  A7  6E  D3  0F  4D  6D  1F  5B  72  63  EC  49  1A  D8  38
17  C0  B5  7D  8A  03  56  12  71  40  FB  4F  74  25  49  4D

seqnum = 4096
First level key from Divers_1:
F3  55  89  F0  9B  F8  01  B1  CA  11  42  73  B9  5F  D6  C1
39  2E  78  F9  FB  81  4D  A0  5A  7C  CA  08  9E  C8  65  42

Second level key from Divers_2:
51  37  D5  C4  A6  E6  BE  42  C4  40  D1  0A  95  EE  A0  7F
08  9E  74  0D  38  90  EB  52  65  2C  0C  B9  3F  20  7B  B4

The resulting key from Divers 3:
FB  30  EE  53  CF  CF  89  D7  48  FC  0C  72  EF  16  0B  8B
53  CB  BB  FD  03  12  82  B0  26  21  4A  B2  E0  77  58  FF

seqnum = 33554431
First level key from Divers_1:
F3  55  89  F0  9B  F8  01  B1  CA  11  42  73  B9  5F  D6  C1
39  2E  78  F9  FB  81  4D  A0  5A  7C  CA  08  9E  C8  65  42

Second level key from Divers_2:
51  37  D5  C4  A6  E6  BE  42  C4  40  D1  0A  95  EE  A0  7F
08  9E  74  0D  38  90  EB  52  65  2C  0C  B9  3F  20  7B  B4

The resulting key from Divers 3:
B8  5B  36  DC  22  82  32  6B  C0  35  C5  72  DC  93  F1  8D
83  AA  01  74  F3  94  20  9A  51  3B  B3  74  DC  09  35  AE

seqnum = 33554432
First level key from Divers_1:
F3  55  89  F0  9B  F8  01  B1  CA  11  42  73  B9  5F  D6  C1
39  2E  78  F9  FB  81  4D  A0  5A  7C  CA  08  9E  C8  65  42
Second level key from Divers_2:
3F EA 59 38 DA 2B F8 DD C4 7E C1 DC 55 61 89 66
79 02 BE 42 0D F4 C3 7D AF 21 75 3B CB 1D C7 F3

The resulting key from Divers 3:
0F D7 C0 9E FD F8 E8 15 73 EE CC F8 6E 4B 95 E3
AF 7F 34 DA B1 17 7C FD 7D B9 7B 6D A9 06 40 8A

seqnum = 274877906943
First level key from Divers_1:
F3 55 89 F0 9B F8 01 B1 CA 11 42 73 B9 5F D6 C1
39 2E 78 F9 FB 81 4D A0 5A 7C CA 08 9E C8 65 42

Second level key from Divers_2:
AB F3 A5 37 98 3A 1B 98 40 06 6D E6 8A 49 BF 25
97 7E E5 C3 F5 2D 33 3E 3C 22 0F 1D 15 C5 08 93

The resulting key from Divers 3:
48 0F 99 72 B4 F2 5D 4C 36 9A 96 AF 91 BC A4 55
3F 79 D8 F0 C5 61 8B 19 FD 44 CF DC 57 FA 37 33

seqnum = 274877906944
First level key from Divers_1:
15 60 0D 9E 8F A6 85 54 CF 15 2D C7 4F BC 42 51
17 B0 3E 09 76 BB 28 EA 98 24 C3 B7 0F 28 CB D8

Second level key from Divers_2:
6C C2 8E B0 93 24 72 12 5C 7A D3 F8 09 73 B3 C8
C4 13 7D A5 73 BC 17 1A 24 ED D4 A3 71 F1 F8 73

The resulting key from Divers 3:
25 28 C1 C6 A8 F0 92 7B F2 BE 27 BB 78 D2 7F 21
46 D6 55 93 B0 C7 17 3A 06 CB 9D 88 DF 92 32 65

### A.1.1.2. TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC ciphersuite

<table>
<thead>
<tr>
<th>TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Key K_root:</td>
</tr>
<tr>
<td>00 11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A</td>
</tr>
<tr>
<td>11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00</td>
</tr>
</tbody>
</table>

seqnum = 0
First level key from Divers_1:
F3 55 89 F0 9B F8 01 B1 CA 11 42 73 B9 5F D6 C1
Second level key from Divers_2:
51 37 D5 C4 A6 E6 BE 42 C4 40 D1 0A 95 EE A0 7F
08 9E 74 0D 38 90 EB 52 65 2C 0C B9 3F 20 7B B4

The resulting key from Divers 3:
19 A7 6E D3 0F 4D 6D 1F 5B 72 63 EC 49 1A D8 38
17 C0 B5 7D 8A 03 56 12 71 40 FB 4F 74 25 49 4D

seqnum = 63
First level key from Divers_1:
F3 55 89 F0 9B F8 01 B1 CA 11 42 73 B9 5F D6 C1
39 2E 78 F9 FB 81 4D A0 5A 7C CA 08 9E C8 65 42

Second level key from Divers_2:
51 37 D5 C4 A6 E6 BE 42 C4 40 D1 0A 95 EE A0 7F
08 9E 74 0D 38 90 EB 52 65 2C 0C B9 3F 20 7B B4

The resulting key from Divers 3:
19 A7 6E D3 0F 4D 6D 1F 5B 72 63 EC 49 1A D8 38
17 C0 B5 7D 8A 03 56 12 71 40 FB 4F 74 25 49 4D

seqnum = 64
First level key from Divers_1:
F3 55 89 F0 9B F8 01 B1 CA 11 42 73 B9 5F D6 C1
39 2E 78 F9 FB 81 4D A0 5A 7C CA 08 9E C8 65 42

Second level key from Divers_2:
51 37 D5 C4 A6 E6 BE 42 C4 40 D1 0A 95 EE A0 7F
08 9E 74 0D 38 90 EB 52 65 2C 0C B9 3F 20 7B B4

The resulting key from Divers 3:
AE BE 1E F4 18 71 3B F0 44 B9 FC D9 E5 72 D4 37
FB 38 B5 D8 29 56 7A 6F 79 18 39 6D 9F 4E 09 6B

seqnum = 524287
First level key from Divers_1:
F3 55 89 F0 9B F8 01 B1 CA 11 42 73 B9 5F D6 C1
39 2E 78 F9 FB 81 4D A0 5A 7C CA 08 9E C8 65 42

Second level key from Divers_2:
51 37 D5 C4 A6 E6 BE 42 C4 40 D1 0A 95 EE A0 7F
08 9E 74 0D 38 90 EB 52 65 2C 0C B9 3F 20 7B B4

The resulting key from Divers 3:
6F 18 D4 00 3E A2 CB 30 F5 FE C1 93 A2 34 F0 7D
7C 43 94 98 7F 50 75 8D E2 2B 22 0D 8A 10 51 06
seqnum = 524288
First level key from Divers_1:
F3 55 89 F0 9B F8 01 B1 CA 11 42 73 B9 5F D6 C1
39 2E 78 F9 FB 81 4D A0 5A 7C CA 08 9E C8 65 42

Second level key from Divers_2:
F6 59 EB 85 EE BD 2A 8D CC 1B B3 F7 C6 00 57 FF
6D 33 B6 0F 74 65 DD 42 B5 11 2C F3 A6 B1 AB 66

The resulting key from Divers 3:
E5 4B 16 41 5B 3B 6E 78 0B 06 2D 24 F7 36 C4
49 54 63 C3 A8 91 E1 FA 46 F7 AE 99 FF F9 F3 78

seqnum = 4294967295
First level key from Divers_1:
F3 55 89 F0 9B F8 01 B1 CA 11 42 73 B9 5F D6 C1
39 2E 78 F9 FB 81 4D A0 5A 7C CA 08 9E C8 65 42

Second level key from Divers_2:
F4 BC 10 1A BB 68 86 2A 8C E3 1E A0 0D DF A7 FE
B8 29 10 F1 24 F4 B1 E2 9E A8 3B E0 06 C2 26 8D

The resulting key from Divers 3:
CF 60 09 04 C7 1E 7B 88 A4 9A C8 E2 45 77 4B 3D
BE ED FB 81 DE 9A 0E 2F 4E 46 C3 56 07 BC 2F 04

seqnum = 4294967296
First level key from Divers_1:
55 CC 95 E0 D1 FB 54 85 AF 8E F6 9A CD 72 B2 32
79 7C D2 E8 5D 86 CD FD 1D E5 5B D1 FA 14 37 78

Second level key from Divers_2:
72 16 91 E1 01 C4 28 96 A6 40 AE 18 3F BB 44 5B
76 37 9C 57 E1 FD 8A 7D 49 A6 23 E4 23 8C 0E 1D

The resulting key from Divers 3:
16 18 0B 24 64 54 00 B8 36 14 38 37 D6 AC 93
95 2A E3 EB 82 44 D5 EC 2A B0 2C FF 30 78 11 38

A.1.2.  Record Examples

A.1.2.1.  TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC ciphersuite

TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC
*************************************************************************
It is assumed that during Handshake following keys were established:

- MAC key:
  00000: 00 11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A
  00010: 11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00
- Encryption key:
  00000: 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00 11
  00010: 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00 11 22
- IV:
  00000: 00 00 00 00

---

seqnum = 0

Application data:
  00000: 00 00 00 00 00 00 00

TLSPlaintext:
  00000: 17 03 03 00 07 00 00 00 00 00 00 00 00 00 00 00

K_MAC_0:
  00000: 19 A7 6E D3 0F 4D 6D 1F 5B 72 63 EC 49 1A D8 38
  00010: 17 C0 B5 7D 8A 03 56 12 71 40 FB 4F 74 25 49 4D

MAC value:
  00000: F3 3E B6 89 6F EC E2 86

K_ENC_0:
  00000: 58 AF BE 9A 4C 31 98 AA AB AA 26 92 C4 19 F1 79
  00010: 7C 9B 92 DE B3 CC 74 46 B3 63 57 71 13 F0 FB 56

IV_0:
  00000: 00 00 00 00

TLSCiphertext:
  00000: 17 03 03 00 0F 9B 42 0D A8 6F AF 36 7F 05 14 43
  00010: CE 9C 10 72

---

seqnum = 4095

Application data:
  00000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  00010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  00020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  ...
TLSPlaintext:
00000:   17 03 03 04 00 00 00 00 00 00 00 00 00 00 00 00
00010:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00020:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. . .
003D0:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
003E0:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
003F0:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00400:   00 00 00 00 00

K_MAC_4095:
00000:   19 A7 6E D3 0F 4D 6D 1F 5B 72 6C EC 49 1A D8 38
00010:   17 C0 B5 7D 8A 03 56 12 71 40 FB 4F 74 25 49 4D

MAC value:
00000:   58 D3 BB 60 8F BC 98 B8

K_ENC_4095:
00000:   58 AF BE 9A 4C 31 98 AA AB AA 26 92 C4 19 F1 79
00010:   7C 9B 92 DE B3 CC 74 46 B3 63 57 71 13 F0 FB 56

IV_4095:
00000:   00 00 00 00

TLSCiphertext:
00000:   17 03 03 08 B7 11 43 8B 16 20 1F 3C 49 33 95
00010:   21 C9 C8 CA 75 66 D4 C2 0F D3 3E 26 92 C4 19 F1 79
00020:   76 04 3E 2B 35 C8 E8 4B B2 55 08 27 66 13 59 6F
. . .
003D0:   E7 77 70 BF 45 17 E1 F8 DD 1B 2C 05 64 AD 68 FC
003E0:   4A 88 9A 48 B8 B1 FF 0E A4 E1 BB 79 4D 56 A4 75
003F0:   2F 51 A5 82 CC 54 1A 80 8F 8C 8B 62 97 68 88 C8
00400:   10 59 DE 41 27 63 A3 E0 99 9A CD DA 77

-------------------------------------------------------------------------------------
seqnum = 4096

Application data:
00000:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00010:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00020:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. . .
007D0:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
007E0:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
007F0:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

TLSPlaintext:
00000:   17 03 03 08 00 00 00 00 00 00 00 00 00 00 00 00
A.1.2.2. TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC ciphersuite

It is assumed that during Handshake following keys were established:

- MAC key:
  00000: 00 11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A
  00010: 11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00
- Encryption key:
  00000: 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00 11
  00010: 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00 11 22
- IV:
00000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

seuqnum = 0

Application data:
00000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00010: 00 00 00 00

TLSPlaintext:
00000: 17 03 03 00 0F 00 00 00 00 00 00 00 00 00 00 00
00010: 00 00 00 00

K_MAC_0:
00000: 19 A7 6E D3 0F 4D 6D 1F 5B 72 63 EC 49 1A D8 38
00010: 17 C0 B5 7D 8A 03 56 12 71 40 FB 4F 74 25 49 4D

MAC value:
00000: FD 17 19 DD 95 08 37 EB 7C 7B B8 F5 00 37 99 81

K_ENC_0:
00000: 58 AF BE 9A 4C 31 98 AA AB AA 26 92 C4 19 F1 79
00010: 7C 9B 92 DE B3 CC 74 4E B3 63 57 71 13 F0 FB 56

IV_0:
00000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

TLSCiphertext:
00000: 17 03 03 00 1F 4D 1A 30 52 36 7B FF C1 4E 46
00010: DC BE 74 6D B6 C9 9A 17 5A 81 C4 71 1E 2F 84 C3
00020: 92 C5 40 7C

seuqnum = 63

Application data:
00000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. . .
00FD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00FE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00FF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

TLSPlaintext:
00000: 17 03 03 10 00 00 00 00 00 00 00 00 00 00 00 00
00010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. . .
Internet-Draft       GOST Cipher Suites for TLS 1.2             May 2021

00FD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00FE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00FF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01000: 00 00 00 00 00

K_MAC_63:
00000: 19 A7 6E D3 0F 4D 6D 1F 5B 72 63 EC 49 1A D8 38
00010: 17 C0 B5 7D 8A 03 56 12 71 40 FB 4F 74 25 49 4D

Mac value:
00000: 98 46 27 61 D0 26 24 4A 2C 0B 7D 1B CC CB E7 B0

K_ENC_63:
00000: 58 AF BE 9A 4C 31 98 AA AB AA 26 92 C4 19 F1 79
00010: 7C 9B 92 DE B3 CC 74 46 B3 63 57 71 13 F0 FB 56

IV_63:
00000: 00 00 00 00 00 00 00 3F

TLSCiphertext:
00000: 17 03 03 10 10 12 93 51 D2 6E 14 07 13 A2 1B 37
00010: 68 24 A2 23 17 CD C0 D8 8E 01 CF A3 FE 21 41 5F
00020: 5C 5E 05 86 9C CF 38 A5 1B C2 E0 ED 68 94 46 A8

00FE0: 19 AD 99 8C 06 25 21 E6 7B 63 59 A4 F5 C8 16 F9
00FF0: 47 6B A7 13 26 82 BB A8 CE 0B ED AD 65 E4 20 A2
01000: 97 B6 E2 C6 1F A4 06 D9 B8 CA 36 FD 9F CD 3A EE
01010: 24 7F F4 D1 96

---------------------------------------------------------------------
seqnum = 64

Application data:
00000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

TLSPlaintext:
00000: 17 03 03 20 00 00 00 00 00 00 00 00 00 00 00 00
00010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

---------------------------------------------------------------------
A.1.3. Handshake Examples

The ClientHello.extensions and the ServerHello.extensions fields contain the extended_master_secret extension (see [RFC7627]) and the renegotiation_info extension (see [RFC5746]) in the following examples.

A.1.3.1. TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC ciphersuite

Server certificate curve OID:
id-GostR3410-2001-CryptoPro-A-ParamSet, "1.2.643.2.2.35.1"

Server public key Q_s:
\[
x = 0x6531D4A72E655BFC9DFB94293B26070282FABF10D5C49B7366148C60E0BF8167
y = 0x37F8CC71DC5D917FC4A66F7826E72750
\]
Server private key $d_s$:
$$0x5F308355DFD6A8AACAEE0837B100A3B1F$$
$$6D63FB29B78EF27D3967757F0527144C$$

ClientHello message:
- msg_type: 01
- length: 000040
- body:
  - client_version:
    - major: 03
    - minor: 03
  - random: 933EA21EC3802A561550EC78D6ED51AC
    2439D7E749C31BC3A3456165889684CA
  - session_id:
    - length: 00
    - vector: --
  - cipher_suites:
    - length: 0004
    - vector:
      - CipherSuite: C100
      - CipherSuite: C101
  - compression_methods:
    - length: 01
    - vector:
      - CompressionMethod: 00
  - extensions:
    - length: 0013
    - vector:
      - Extension: /* signature_algorithms */
        - extension_type: 000D
        - extension_data:
          - length: 0006
          - vector:
            - supported_signature_algorithms:
              - length: 0004
              - vector:
                - /* 1 pair of algorithms */
                  - hash: 08
                  - signature: 40
                - /* 2 pair of algorithms */
                  - hash: 08
                  - signature:
Extension: /* renegotiation_info */
extension_type: FF01
extension_data:
  length: 0001
  vector:
    renegotiated_connection:
      length: 00
      vector: --

Extension: /* extended_master_secret */
extension_type: 0017
extension_data:
  length: 0000
  vector: --

00000: 01 00 00 40 03 03 93 3E A2 1E C3 80 2A 56 15 50
00010: EC 78 D6 ED 51 AC 24 39 D7 E7 49 C3 1B C3 A3 45
00020: 61 65 88 96 84 CA 00 00 04 C1 00 C1 01 01 00 00
00030: 13 00 0D 00 06 00 04 08 40 08 41 FF 01 00 01 00
00040: 00 17 00 00

Record layer message:
type: 16
version:
  major: 03
  minor: 03
length: 0044
fragment: 010000400303933EA21EC3802A561550
          EC78D6ED51AC2439D7E749C31BC3A345
          6165889684CA000004C100C101010000
          13000D0006000408400841FF01000100
          00170000

00000: 16 03 03 00 44 01 00 00 40 03 03 93 3E A2 1E C3
00010: 80 2A 56 15 50 EC 78 D6 ED 51 AC 24 39 D7 E7
00020: 49 C3 1B C3 A3 45 61 65 88 96 84 CA 00 00 04 C1
00030: C1 01 01 00 00 13 00 0D 00 06 00 04 08 40 08
00040: 41 FF 01 00 01 00 00 17 00 00

--------------------------- Server ---------------------------

ServerHello message:
msg_type: 02
length: 000041
body:
  server_version:
    major: 03
minor: 03
random: 933EA21E49C31BC3A3456165889684CA
       A5576CE7924A24F58113808DBD9EF856
session_id:
  length: 10
  vector: C3802A561550EC78D6ED51AC2439D7E7
cipher_suite:
  CipherSuite: C101
compression_method:
  CompressionMethod: 00
extensions:
  length: 0009
  vector:
    Extension: /* renegotiation_info */
    extension_type: FF01
    extension_data:
      length: 0001
      vector:
        renegotiated_connection:
          length: 00
          vector: --
    Extension: /* extended_master_secret */
    extension_type: 0017
    extension_data:
      length: 0000
      vector: --

00000: 02 00 00 41 03 03 93 3E A2 1E 49 C3 1B C3 A3 45
00010: 61 65 88 96 84 CA A5 57 6C E7 92 4A 24 F5 81 13
00020: 80 8D BD 9E F8 56 10 C3 80 2A 56 15 50 EC 78 D6
00030: ED 51 AC 24 39 D7 E7 C1 01 00 00 09 FF 01 00 01
00040: 00 00 17 00 00

Record layer message:
  type: 16
  version:
    major: 03
    minor: 03
  length: 0045
  fragment: 020000410303933EA21E49C31BC3A345
           6165889684CAA5576CE7924A24F58113
           808DBD9EF85610C3802A561550EC78D6
           ED51AC2439D7E7C101000009FF010001
           0000170000

00000: 16 03 03 00 45 02 00 00 41 03 03 93 3E A2 1E 49
00010: C3 1B C3 A3 45 61 65 88 96 84 CA A5 57 6C E7 92
00020: 4A 24 F5 81 13 80 8D BD 9E F8 56 10 C3 80 2A 56
Certificate message:

msg_type: 0B
length: 0001DB

body:

certificate_list:
length: 0001D8

vector:
ASN.1Cert:
length: 0001D5
vector: 308201D13082001D7E7C1010000

... 
797990E4B5452CF82FE1F9EE237B754
C0D5787D752A28013DFFC8224AD114B
BD7C1BB7E480AD6EEF9857A8C99C595
9053EEDEFE9
Record layer message:

type: 16
version:
major: 03
minor: 03
length: 01DF
fragment: 0B0001DB0001D80001D5308201D13082
017EA003020102020833FBB2C0E9575A
46300A06082A85030701010302301F31
. . .
8844F9F1C855E2DB5B19797990E4B545
2CF8F2FE1F19EE237B754C6CD507B0D752
A28013DFFC8224AD114BB7C1BB71E48
0AD6EEF9857A8C99C5959053EEDFE9

00000: 16 03 03 01 DF 0B 00 01 DB 00 01 D8 00 01 D5 30
00010: 82 01 D1 30 82 01 7E A0 03 02 01 02 02 08 33 FB
00020: B2 C0 E9 57 5A 46 30 0A 06 08 2A 85 03 07 01 01
00030: 03 02 30 1F 31 1D 30 1B 06 03 55 04 03 0C 14 74
00040: 65 73 74 5F 73 65 6C 66 73 69 67 6E 65 64 5F 63
00050: 65 72 74 30 6F 30 1F 06 08 2A 85 03 07 01 01
00060: 01 01 30 13 06 07 2A 85 03 02 02 23 01 06 08 2A
00070: 85 03 07 01 01 02 02 03 43 00 04 40 67 81 BF E0
00080: 60 8C 14 66 73 9B C4 D5 10 BF FA 82 02 07 26 3B
00090: 29 94 FB 9D FC 5B 65 2E A7 D4 31 65 B8 A5 53 B5
000A0: 77 3E 36 D4 6C C2 66 C2 FF B4 70 82 50 27 E7 26
000B0: 78 6F A6 C4 7F 91 5D DC 71 CC F8 37 A3 81 96 30
000C0: 81 93 30 1D 06 03 55 1D 0E 04 16 04 14 E7 D0 0B
000D0: B8 4D 8D 24 18 29 3E 05 C1 7C E7 77 98 D4 8D 30
000E0: 16 30 0E 06 03 55 1D 0F 01 01 FF 04 04 03 02 01
000F0: C6 30 12 06 03 55 1D 13 01 01 FF 04 04 08 30 06 01
00100: 01 FF 02 01 01 30 4E 06 03 55 1D 23 04 47 30 45
00110: 80 14 E7 D0 0B B8 4D 8D 24 18 29 3E 05 C1 7C E7
00120: 77 98 D4 8D 30 16 A1 23 A4 21 30 1F 31 1D 30 1B
00130: 06 03 55 04 03 0C 14 74 65 73 74 5F 73 65 6C 66
00140: 73 69 67 6E 65 64 5F 63 65 72 74 82 08 33 FB B2
00150: C0 E9 57 5A 46 30 0A 06 08 2A 85 03 07 01 01 03
ServerHelloDone message:
msg_type:                0E
length:                  000000
body:                    --

Record layer message::
type:                    16
version:
  major:                 03
  minor:                 03
length:                  0004
fragment:                0E000000

---------------------------Client---------------------------
PMS:

Random d_eph value:
0xA5C77C7482373DE16CE4A6F73CCE7F78
  471493FF2C0709B8B706C9E8A25E6C1E

Q_eph ephemeral key:
\[ x = 0xA8F36D63D262A203978F1B3B6795CDBB \]
\[ F1AE7FB8EF74F1F18871C198E00793 \]
\[ y = 0x34CA5D6B4485640EA195435993BEB1F8 \]
\[ B016ED610496B5CC175AC2EA1F14F887 \]

HASH (r_c | r_s):
  00000:   C3 EF 04 28 D4 B7 A1 F4 C5 02 5F 2E 65 DD 2B 2E
  00010:   A5 83 AE EF DB 67 C7 F4 21 4A 6A 29 8E 99 E3 25
Export key generation. r value:
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E

Export key generation. UKM value:
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E

seed:
00000: A5 83 AE EF DB 67 C7 F4
K_EXP:
00000: 1E 58 54 90 E8 65 FF D1 8F 18 D7 C0 A0 4D 0E E8
00010: 4F 1A 5D 79 7C EF AD A0 1B 1E 3B 7F DB 90 E0 29

Export keys K_Exp_MAC | K_Exp_ENC used in KExp15 algorithm:
00000: 2D 8B A8 C8 4C B2 32 FF 41 F1 0C 3A D9 24 13 42
00010: 23 25 4F 71 E5 69 6D 21 29 C3 E4 C9 DA A6 B2 93
00020: 84 9E B6 34 0B FF AE 69 28 A3 C3 E4 FF 92 EC CB
00030: 1E 8F 0C F7 A1 88 36 8E 6B 74 8E 52 EA 37 8B 0C

IV:
00000: 21 4A 6A 29
PMSEXP:
00000: D7 F0 F0 42 23 67 86 7B 25 FA 42 33 A9 54 F5 8B
00010: DE 92 E9 C9 BB FB 88 16 C9 9F 15 E6 39 87 22 A0
00020: B2 B7 BF E8 49 3E 9A 5C

------------------------Client------------------------

ClientKeyExchange message:
msg_type: 10
length: 000095
body:
exchange_keys:
00000: 10 00 00 95 30 81 92 04 28 D7 F0 F0 42 23 67 86
00010: 7B 25 FA 42 33 A9 54 F5 8B DE 92 E9 C9 BB FB 88
00020: 16 C9 9F 15 E6 39 87 22 A0 B2 B7 BF E8 49 3E 9A
00030: 5C 30 66 30 1F 06 08 2A 85 03 07 01 01 01 30
00040: 13 06 07 2A 85 03 02 02 23 01 06 08 2A 85 03 07
Record layer message:

- **type**: 16
- **version**: 03
- **major**: 03
- **minor**: 03
- **length**: 0099
- **fragment**: 100000953081920428D7F0F042236786

PMSEXP extracted:

- 00000: D7 F0 F0 42 23 67 86 7B 25 FA 42 33 A9 54 F5 8B DE 92
- 00020: BF E8 49 3E 9A 5C 30 66 30 1F 06 08 2A 85 03 07
- 00030: 01 01 01 01 30 13 06 08 2A 85 03 07
- 00040: 08 2A 85 03 07 01 01 02 02 03 43 00 04 40 93
- 00050: A1 0E 64 85 64 85 6B 5D CA 34

--- Server ---

PMSEXP extracted:

- 00000: D7 F0 F0 42 23 67 86 7B 25 FA 42 33 A9 54 F5 8B
- 00010: DE 92 E9 C9 BB FB 88 16 C9 9F 15 E6 39 87 22 A0
- 00020: BF E8 49 3E 9A 5C

**HASH(r_c | r_s):**

- 00000: C3 EF 04 28 D4 B7 A1 F4 C5 02 5F 2E 65 DD 2B 2E
- 00010: A5 83 AE EF DB 67 C7 F4 21 4A 6A 29 8E 99 E3 25

Export key generation. r value:

0xC3EF0428D4B7A1F4C5025F2E65DD2B2E

Export key generation. UKM value:
seed:
00000:   A5 83 AE EF DB 67 C7 F4

K_EXP:
00000:   1E 58 54 90 E8 65 FF D1 18 D7 C0 A0 4D 0E E8
00010:   4F 1A 5D 79 7C EF AD A0 1B 1E 3B 7F DB 90 E0 29

Import keys K_Imp_MAC | K_Imp_ENC used in KImp15 algorithm:
00000:   2D 8B A8 C8 4C B2 32 FF 41 F1 0C 3A D9 24 13 42
00010:   23 25 4F 71 E5 69 6D 3D 29 C3 E4 C9 DA A6 B2 93
00020:   84 9E B6 34 08 FF AE 69 28 A3 C3 E4 FF 92 EC CB
00030:   1E 8F 0C F7 A1 88 36 8E 6B 74 8E 52 EA 37 8B 0C

IV:
00000:   21 4A 6A 29

PMS:
00000:   A5 57 6C E7 92 4A 24 F5 81 13 80 8D BD 9E F8 56
00010:   F5 BD C3 B1 83 CE 5D AD CA 36 A5 3A A0 77 65 1D

---------------------------Client---------------------------

HASH(HM):
00000:   7E 1F 59 D3 64 9D B6 09 00 EA 4F 8A 58 5A 65 7A
00010:   92 77 B3 04 50 58 4C F5 43 51 19 8C DE A3 0C 49

MS:
00000:   FD D2 7C B4 04 AD 4E 44 49 68 4F 7C 55 90 E9 E7
00010:   02 EF 41 01 93 3B 52 77 A4 A9 6D F5 00 B0 7C C3
00020:   32 4F D8 A6 D9 07 CB B0 3D F3 FB 33 1F 1C 4D 0C

Client connection key material
K_write_MAC|K_read_MAC|K_write_ENC|K_read_ENC|IV_write|IV_read:
00000:   DD 4E 10 17 E3 09 1F FD 86 75 65 8A 78 00 90 09
00010:   3B BE 69 EC A6 93 31 5C A8 5B E0 A6 14 3D C9 F8
00020:   1D 64 D0 23 46 5F 8B EA 17 F8 12 F8 C2 D8 BF C0
00030:   D9 BB AB A7 B4 DF D3 A1 7C E0 E1 3B 2D 63 65 F3
00040:   FC 8B 34 59 CF 54 FE 44 9A 04 07 64 53 73 08 00
00050:   75 18 32 55 90 07 B6 C4 EA C6 75 48 71 BC 97 8A
00060:   B9 0E 2A EE 98 77 14 BB D8 F7 57 AE F7 84 FF 24
00070:   47 B3 94 2E B4 3E 26 35 73 1C 4C 28 22 D0 2D 79
00080:   2B 6A 81 3F 93 ED A6 FA

---------------------------Server---------------------------
Server connection key material
K_read_MAC|K_write_MAC|K_read_ENC|K_write_ENC|IV_read|IV_write:
00000:  DD 4E 10 17 E3 09 1F FD 86 75 65 8A 78 00 90 09
00010:  3B BE 69 EC A6 93 31 5C A8 5B E0 A6 14 3D C9 F8
00020:  1D 64 D0 23 46 5F 8B EA 17 F8 12 F8 C2 D8 BF C0
00030:  D9 BB AB A7 B4 DF D3 A1 7C E0 E1 3B 2D 63 65 F3
00040:  FC 8B 34 59 CF 54 FE 44 9A 04 07 64 53 73 08 00
00050:  75 10 32 55 9D 07 B6 C4 EA C6 75 48 71 BC 97 8A
00060:  B9 6E 2A EE 98 77 14 BB D8 F7 57 AE F7 84 FF 24
00070:  47 B3 94 2E B4 3E 26 35 73 1C 4C 28 22 D0 2D 79
00080:  2B 6A 81 3F 93 ED A6 FA

---------------------------Client---------------------------
ChangeCipherSpec message:
type:                      01
00000:  01

Record layer message:
type:                      14
type:                      14
version:                   03
major:                     03
minor:                     03
length:                    0001
fragment:                  01
00000:  14 03 03 00 01 01

---------------------------Client---------------------------
HASH(HM):
00000:  7E 1F 59 D3 64 9D B6 09 00 EA 4F 8A 58 5A 65 7A
00010:  92 77 B3 04 50 58 4C F5 43 51 19 8C DE A3 0C 49
client_verify_data:
00000:  B4 61 C5 AD 25 EA 1E 62 B3 70 BD 1F 1B CB 16 91
00010: FC CC BA 37 8B BC 13 43 BE 54 B3 8D F5 53 B7 A5

---------------------------Client---------------------------

Finished message:
msg_type: 14
length: 000020
body:
  verify_data: B461C5AD25EA1E62B370BD1F1BCB1691
  FCCCBA378BBC1343BE54B38DF553B7A5

00000: 14 00 00 20 B4 61 C5 AD 25 EA 1E 62 B3 70 BD 1F
00010: 1B CB 16 91 FC CC BA 37 8B BC 13 43 BE 54 B3 8D
00020: F5 53 B7 A5

Record layer message:
  type: 16
  version:
    major: 03
    minor: 03
  length: 002C
  fragment: 0C630271D4DA39DD8D6BD040302D9B8F
            33D5F7B967EED155F7D6592892C03C7
            885C249B1225B184AB4D5DBF

00000: 16 03 03 00 2C 0C 63 02 71 D4 DA 39 DD 8D 6B D0
00010: 40 39 2D 9B 8F 33 D5 F7 B9 67 EE D1 55 F7 D6 55
00020: 92 89 2C 03 C7 88 5C 24 9B 12 25 B1 84 AB 4D 5D
00030: BF

---------------------------Server---------------------------

ChangeCipherSpec message:
  type: 01

00000: 01

Record layer message:
  type: 14
  version:
    major: 03
    minor: 03
  length: 0001
  fragment: 01

00000: 14 03 03 00 01 01
HASH(HM):
00000:   DB D7 D8 93 82 4A ED FD D5 FB 7B 75 4B 47 E1 E6
00010:   AF E0 77 DA E6 D1 13 63 42 07 C7 EE 0F C6 F3 B1

server_verify_data:
00000:   45 39 EC 8D 0A F7 B1 A6 20 41 AB 43 4A 43 77 71
00010:   D3 4C 47 19 D8 6E BB FD 0F 28 C3 E9 53 55 0C D0

Finished message:
msg_type:                14
length:                  000020
body:
    verify_data:           4539EC8D0AF7B1A62041AB434A437771
                          D34C4719D86EBBFD0F28C3E9S3550CD0
00000:   14 00 00 20 45 39 EC 8D 0A F7 B1 A6 20 41 AB 43
00010:   4A 43 77 71 D3 4C 47 19 D8 6E BB FD 0F 28 C3 E9
00020:   53 55 0C D0

Record layer message:
type:                    16
version:
    major:                 03
    minor:                 03
length:                  002C
fragment:                E6A94A4BF70886566A2316811E57B483
                          BB1E47950A1FF820A80DCA77A4DF9954
                          2DAB6953F3ED03D95CCA4748
00000:   16 03 03 00 2C E6 A9 4A 4B F7 08 86 56 6A 23 16
00010:   81 1E 57 B4 83 BB 1E 47 95 0A 1F F8 20 A8 0D CA
00020:   77 A4 DF 99 54 2D AB 69 53 F3 ED 03 D9 5C CA 47
00030:   48

---------------------------Client---------------------------

Application data:
00000:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00010:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Record layer message:
type:                    17
version:
  major: 03
  minor: 03
length: 0028
fragment: 38807B6E5E0C3F4F7E0DBF7758031BF0
100C4B63ADBC75F49BCBF428572D37
7CAED097336DB203

00000: 17 03 03 00 28 38 80 7B 6E 0C 3F 4F 7E 0D BF
00010: 77 58 03 1B F0 7F 10 0C 4B 63 AD BC 75 F4 9B CB
00020: F4 28 57 2D 37 7C AE D0 97 33 6D B2 03

---------------------------Server---------------------------

Application data:
00000: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00010: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

Record layer message:
type: 17
version:
  major: 03
  minor: 03
length: 0028
fragment: 05B869E5C979C3B9D4837B8E39D9BBEE
1BBD0052D3D48340D0CDE082B33BC07F
4E742D1113249AD8

00000: 17 03 03 00 28 05 B8 69 E5 C9 79 C3 B9 D4 83 7B
00010: 8E 39 D9 BB EE 1B BD 00 52 D3 D4 83 40 D0 CD E0
00020: 82 B3 3B C0 7F 4E 74 2D 11 13 24 9A D8

---------------------------Client---------------------------
close_notify alert:
Alert:
  level: 01
  description: 00

00000: 01 00

Record layer message:
type: 15
version:
  major: 03
  minor: 03
length: 000A
fragment: 4F2A0807A0374E28C632

00000: 15 03 03 00 0A 4F 2A 08 07 A0 37 4E 28 C6 32

---------------------------Server---------------------------

close_notify alert:
Alert:
  level: 01
  description: 00

00000: 01 00

Record layer message:
type: 15
version:
  major: 03
  minor: 03
length: 000A
fragment: 999468B49AC5B0DE512C

00000: 15 03 03 00 0A 99 94 68 B4 9A C5 B0 DE 51 2C

A.1.3.2. TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC ciphersuite

Server certificate curve OID:
  id-tc26-gost-3410-2012-512-paramSetC, "1.2.643.7.1.2.1.2.3"

Server public key Q_s:
x = 0xF14589DA479AD972C66563669B3FF580
  92E6A30A288BF447C09F6C3133E9724
  7A9706B267703C9B4E239F0D7C7E3310
  C22D2752B35BD2E4FD39B8F11DEB833A

y = 0xF305E95B36502D4E60A1059FB20413F2
  FC7C95727F3A2C04B1DFDB53B0413F2
  99F2DE66A5E1CCB4101A7A01D612BE6
  BD78E1E3B3D567EBB16ABE587A11F4EA

Server private key d_s:
Client certificate curve OID:
id-tc26-gost-3410-2012-256-paramSetA, "1.2.643.7.1.2.1.1.1"

Client public key Q_c:
x = 0x0F5DB18A9E15F324B778676025BFD7B5
   DF066566EABAA1C51CD879F87B0B4975
y = 0x9EE5BBF18361F842D3F087DEC2943939
   E0FA2BFB4EDEC25A8D10ABB22C48F386

Client private key d_c:
0x0918AD3F7D209ABF89F1E8505DA894CE
   E10DA09D32E72E815D9C0ADA30B5A103

---------------------------Client---------------------------

ClientHello message:
msg_type:                01
length:                  000040
body:
   client_version:
      major:               03
      minor:               03
      random:                933EA21EC3802A561550EC78D6ED51AC
         2439D7E749C31BC3A3456165889684CA
   session_id:
      length:              00
      vector:              --
   cipher_suites:
      length:              0004
      vector:
         CipherSuite:       C100
         CipherSuite:       C101
   compression_methods:
      length:              01
      vector:
         CompressionMethod:  00
   extensions:
      length:              0013
      vector:
         Extension: /* signature_algorithms */
            extension_type: 000D
extension_data:
  length: 0006
  vector:
    supported_signature_algorithms:
      length: 0004
      vector:
        /* 1 pair of algorithms */
        hash: 08
        signature:
          40
        /* 2 pair of algorithms */
        hash: 08
        signature:
          41
  Extension: /* renegotiation_info */
    extension_type: FF01
    extension_data:
      length: 0001
      vector:
        renegotiated_connection:
          length: 00
          vector: --
    Extension: /* extended_master_secret */
      extension_type: 0017
      extension_data:
        length: 0000
        vector: --

00000: 01 00 00 40 03 03 93 3E A2 1E C3 80 2A 56 15 50
00010: EC 78 D6 ED 51 AC 24 39 D7 E7 49 C3 1B C3 A3 45
00020: 61 65 88 96 84 CA 00 00 04 C1 00 C1 01 01 00 00
00030: 13 00 0D 00 06 00 04 08 40 08 41 FF 01 00 01 00
00040: 00 17 00 00

Record layer message:
type: 16
version:
  major: 03
  minor: 03
length: 0044
fragment: 010000400303933EA21EC3802A561550
          EC78D6ED51AC2439D7E749C31BC3A345
          6165889684CA000004C100C101010000
          13000D0006000408400841FF01000100
          00170000

00000: 16 03 03 00 44 01 00 00 40 03 03 93 3E A2 1E C3
00010: 80 2A 56 15 50 EC 78 D6 ED 51 AC 24 39 D7 E7 49
ServerHello message:
msg_type: 02
length: 000041
body:
  server_version:
    major: 03
    minor: 03
  random: 933E49C31BC3A3456165889684CA
          A5576CE7924A24F58113808DBD9EF856
  session_id:
    length: 10
    vector: C3802A561550EC78D6ED51AC2439D7E7
  cipher_suite:
    CipherSuite: C100
  compression_method:
    CompressionMethod: 00
  extensions:
    length: 0009
    vector:
      Extension: /* renegotiation_info */
      extension_type: FF01
      extension_data:
        length: 0001
        vector:
          renegotiated_connection:
            length: 00
            vector: --
      Extension: /* extended_master_secret */
      extension_type: 0017
      extension_data:
        length: 0000
        vector: --
version:
  major:      03
  minor:      03
length:       0045
fragment:     020000410303933EA21E49C31BC3A3456165889684CAA5576CE7924A24F58113808DBD9EF85610C3802A561550EC78D6ED51AC2439D7E7C100000000FF010001
0000170000

---------------------------Server---------------------------

Certificate message:
msg_type:       0B
length:         00024C
body:
  certificate_list:
    length:       000249
    vector:
      ASN.1Cert:
        length:       000246
        vector:
          30820242308201AEA08302010201001
          300A06082A850307010103033042312C
          302A06092A864886F57D0109011161D74
          . . .
          371AF83C5BC58B36DFEFA7345D50317
          867C177AC84AC07EE8612164629AB7BD
          C4BAA0F64A741FE7298E825BFCE8672
          029F875391F7
00000:   0B 00 02 4C 00 02 49 00 02 46 30 82 02 42 30 02
00010:   01 AE A0 03 02 01 02 02 01 01 30 0A 06 08 2A 85
00020:   03 07 01 01 03 03 30 42 31 2C 30 2A 06 09 2A 86
00030:   48 86 F7 0D 01 09 01 16 1D 74 6C 73 31 32 5F 73
00040:   65 72 76 65 72 35 31 32 43 40 63 72 79 70 74 6F
00050:   70 72 6F 2E 72 75 31 12 30 10 06 03 55 04 03 13
00060:   09 53 65 72 76 65 72 76 35 31 32 30 1E 17 0D 31 37
00070:   30 35 32 35 30 39 32 35 31 38 5A 17 0D 33 30 30
00080:   35 30 31 30 39 32 35 31 38 5A 30 42 31 2C 30 2A
00090:   06 09 2A 86 48 86 F7 0D 01 09 01 16 1D 74 6C 73
000A0:   31 32 5F 73 65 72 76 76 32 35 31 32 43 40 63 72

---------------------------Server---------------------------
Record layer message:

type: 16
version:
  major: 03
  minor: 03
length: 0250
fragment: 0B00024C000024900029570248302423082 01AE003020102010101300A06082A85 030701010303042312C302A06092A86 8B366DFEF7A7345D50317867C177AC84A C07EE8612164629AB7BDC48AA0F64A74 1FE7298E825BFC8E8672029F875391F7
CertificateRequest message:

msg_type: 0D
length: 00000B

body:
certificate_types:
  length: 02
  vector:
    /* gost_sign256 */
    43
    /* gost_sign512 */
    44

supported_signature_algorithms:
  length: 0004
vector:
/* 1 pair of algorithms */
hash: 08
signature: 40
/* 2 pair of algorithms */
hash: 08
signature: 41
certificate_authorities:
  length: 0000
vector: --

00000: 0D 00 00 0B 02 43 44 00 04 08 40 08 41 00 00

Record layer message:
type: 16
version:
  major: 03
  minor: 03
length: 000F
fragment: 0D00000B0243440004084008410000

00000: 16 03 03 00 0F 0D 00 00 0B 02 43 44 00 04 08 40
00010: 08 41 00 00

---------------------------Server---------------------------

ServerHelloDone message:
msg_type: 0E
length: 00000
body: --

00000: 0E 00 00 00

Record layer message:
type: 16
version:
  major: 03
  minor: 03
length: 0004
fragment: 0E00000

00000: 16 03 03 00 04 0E 00 00 00

---------------------------Client---------------------------

Certificate message:
msg_type: 0B
length: 0001EA
body:
  certificate_list:
    length: 0001E7
    vector:
      ASN.1Cert:
        length: 0001E4
        vector: 308201E03082018DA003020102020101300A06082A850307010103023053312E302C06092A864886F70D010901161F74 . . . C1CAB43AC01AFB0F3451BDC2DB188BBCB77884251C9DF6037BA830F4B3105E96FDCC9BC1C95ABE658266C48402E070DE1F292724E8
Internet-Draft       GOST Cipher Suites for TLS 1.2             May 2021

Record layer message:
  type:                    16
  version:
    major:                 03
    minor:                 03
  length:                  01EE
  fragment:                0B0001EA0001E70001E4308201E03082
                          018DA003020102020101300A06082A85
                          030701010302305312E302C06092A86
                          . . .
                          3522B41102D6201F2350C1CAB43AC01A
                          FB0F3451BDC2DB188BBCB77884251CDF
                          6037BA830F4B31D5E96FDC9BC1C95ABE
                          658266C48402E070DE1F292724E8

00000:   16 03 03 01 EE 0B 00 01 EA 00 01 E7 00 01 E4 30
00010:   82 01 E0 30 82 01 8D A0 03 02 01 02 02 01 01 30
00020:   0A 06 08 2A 85 03 07 01 01 03 02 39 53 31 2E 30
00030:   2C 06 09 2A 86 48 86 F7 0D 01 09 01 16 1F 74 6C
00040:   73 31 32 5F 63 6C 6F 6E 65 6E 67 74 32 35 36 41
00050:   5F 45 40 63 72 79 70 74 6F 70 72 6F 2E 72 75 31
00060:   1F 06 03 55 04 03 1E 18 00 43 00 6C 00 69 00 65
00070:   00 6E 00 74 00 32 00 35 00 36 00 41 00 5F 00 45
00080:   30 1E 17 0D 31 37 39 35 32 35 33 31 31 38
00090:   5A 17 0D 33 30 30 35 30 31 39 03 31 31 38 5A
000A0:   30 53 31 2E 30 2C 06 09 2A 86 48 86 F7 0D 01 09
000B0:   01 16 1F 74 6C 73 31 32 5F 63 6C 6F 6E 65 6E
000C0:   74 32 35 36 41 5F 45 40 63 72 79 70 74 6F 70
000D0:   72 75 31 21 30 1F 06 03 55 04 03 1E 18 00 43
000E0:   00 6C 00 69 00 65 00 74 00 32 00 35 00 36 00
000F0:   41 00 5F 00 45 04 30 30 21 06 08 2A 85 03 07
00100:   01 01 01 30 15 06 09 2A 85 03 07 01 01 01
00110:   06 08 2A 85 03 07 01 01 02 02 03 43 00 04 40
00120:   01 49 0B 7B F8 79 D8 1C C5 A1 BA EA 66 65 06
00130:   DF B5 67 BF 25 60 67 78 B7 24 F3 15 9E 8A B1
00140:   5D 0F 86 01 40 01 48 2C B2 AB 10 8D 5A C2 DE
00150:   4E FB 2B FA E0 39 39 94 C2 DE 87 F0 D3 42 F8
00160:   61 83 F1 BB E5 9E A3 43 39 41 30 1D 06 03 55
00170:   1D 0E 04 16 04 14 74 49 1E 77 30 D3 42 A6 28
00180:   0E 72 A1 13 9D D9 90 8B FA F1 03 30 0B 06 03
00190:   55 1D 0F 04 04 03 02 07 80 30 13 06 03 55 1D
001A0:   25 04 0C 30 0A 06 08 2B 06 01 05 0F 07 03 02
001B0:   30 0A 06 08 2A 85 03 07 01 01 03 02 03 41 00
001C0:   1C 2D 35 22 B4 11 02 D6 20 1F 23 50 C1 CA B4
001D0:   3A C0 1A FB 0F 34 51 BD C2 DB 18 8B BC B7 78
001E0:   84 25 1C DF 60 37 BA 83 0F 4B 31 D5 E9 6F DC
001F0:   9B C1 C9 5A BE 65 82 66 C4 84 02 E0 70 DE 1F

PMS value:
00000: A5 57 6C E7 92 4A 24 F5 81 13 80 BD BD 9E F8 56
00010: F5 BD C3 B1 83 CE 5D AD CA 36 A5 3A A0 77 65 1D

Random d_eph value:
0x150ACD11B66DD695AD18418FA7A2DC63
6B7E29DCA24536AAABC826EE3175BB1FA
DC3AA0D01D3092E120B0FCF7EB872F4B
7E26EA17849D689222A48CF95A6E4831

Q_eph ephemeral key:
x = 0xC941BE5193189B476D5A0334114A3E04
BBE5B37C738AE40F150B334135288664
FEFBC5622818894A07B1F7AD60E28480
B4B637B90EA7D4BA980186B605D75BC6

y = 0xA154F7B93E8148652011F4FD52C9A06A
6471ADB28D0A949AE26BC786DE874153
ABC0B35164F3214AB83C00ECE27831
B093528456234EFE766224FC2A7E9ABE

HASH (r_c | r_s):
00000: C3 EF 04 28 D4 B7 A1 F4 C5 02 5F 2E 65 DD 2B 2E
00010: A5 83 AE EF DB 67 C7 F4 21 4A 6A 29 8E 99 E3 25

Export key generation. r value:
0xC3EF0428D04B7A1F4C5025F2E65DD2B2E

Export key generation. UKM value:
0xC3EF0428D04B7A1F4C5025F2E65DD2B2E

Export keys K_Exp_MAC | K_Exp_ENC used in KExp15 algorithm:
00000: 7D AC 56 E4 8A 4D C1 70 FA A8 FC BA E2 0D B8 45
00010: 45 0C CC C4 C6 32 8B DC 8D 01 15 7C EF A2 A5 F1
00020: 1F 1C BA D8 86 61 66 F0 1F FA AB 01 52 E2 4B F4
00030: 60 9D 5F 46 A5 C8 99 C7 87 90 0D 08 B9 FC AD 24

IV:
00000: 21 4A 6A 29 8E 99 E3 25

PMSEXP:
00000: 25 0D 1B 67 A2 70 AB 04 D3 F6 54 18 E1 D3 80 B4
00010: CB 94 5F 0A 3D CA 51 50 0C F3 A1 BE F3 7F 76 C0
00020: 73 41 A9 83 9C CF 6C BA 71 89 DA 61 EB 67 17 6C
### Client

**ClientKeyExchange message:**

| msg_type: | 10 |
| length: | 0000E2 |
| body: |
| exchange_keys: |
| 3081DF0430250D1B67A270AB04D3F654 |
| 18E1D380B4CB945F0A3DCA51500CF3A1 |
| BEF37F76C07341A9839CCF6CBA7189DA |
| . . |
| 93B03178E2EC003CA8A814324F16350B |
| C0AB534187DE86C76BE29A940A8DB2AD |
| 71646AA0C952FDF411206548813EB9F7 |
| 54A1 |

| 00000: | 10 00 00 E2 30 81 DF 04 30 25 0D 1B 67 A2 70 AB |
| 00010: | 04 D3 F6 54 18 E1 D3 80 B4 CB 94 5F 0A 3D CA 51 |
| 00020: | 50 0C F3 A1 BE F3 7F 76 C0 73 41 A9 83 9C CF 6C |
| 00030: | BA 71 89 DA 61 EB 67 17 6C 30 81 AA 30 21 06 08 |
| 00040: | 2A 85 03 07 01 01 02 03 15 06 09 2A 85 03 07 |
| 00050: | 01 02 01 02 03 06 08 2A 85 03 07 01 01 02 03 03 |
| 00060: | 81 84 00 04 81 80 C6 5B D7 05 B6 86 01 98 BA D4 |
| 00070: | A7 0E B9 37 B6 B4 80 84 E2 60 AD F7 B1 07 4A 89 |
| 00080: | 18 28 62 C5 BF FE 64 86 28 35 41 33 0B 15 0F E4 |
| 00090: | 8A 73 7C B3 E5 BB 04 3E 4A 11 34 03 5A 6D 47 9B |
| 000A0: | 18 93 51 BE 41 C9 BE 9A 7E 2A FC 24 62 76 FE 4E |
| 000B0: | 23 56 84 52 93 B0 31 78 E2 EC 00 3C A8 A8 14 32 |
| 000C0: | 4F 16 35 0B C0 AB 53 41 87 DE 86 C7 6B E2 9A 94 |
| 000D0: | 0A 8D B2 AD 71 64 6A A0 C9 52 FD F4 11 20 65 48 |
| 000E0: | 81 3E B9 F7 54 A1 |

**Record layer message:**

| type: | 16 |
| version: |
| major: | 03 |
| minor: | 03 |
| length: | 00E6 |
| fragment: |
| 100000E23081DF0430250D1B67A270AB |
| 64D3F65418E1D380B4 CB945F0A3DCA51 |
| 500CF3A1BEF37F76C07341A9839CCF6C |
| . . |
| 2356B45293B03178E2EC003CA8A81432 |
| 4F16350BC0AB534187DE86C76BE29A94 |
| 0A8DB2AD71646AA0C952FDF411206548 |
| 813EB9F754A1 |

| 00000: | 16 03 03 00 E6 10 00 00 E2 30 81 DF 04 30 25 0D |
00010:  1B 67 A2 70 AB 04 D3 F6 54 18 E1 D3 80 B4 CB 94
00020:  5F 0A 3D CA 51 50 0C F3 A1 BE F3 7F 76 C0 73 41
00030:  A9 83 9C CF 6C BA 71 89 DA 61 EB 67 17 6C 30 81
00040:  AA 30 21 06 08 2A 85 03 07 01 01 02 30 15 06
00050:  09 2A 85 03 07 01 02 03 06 08 2A 85 03 07
00060:  01 01 02 03 03 81 84 00 04 81 80 C6 5B D7 05 B6
00070:  86 01 98 BA D4 A7 9E B9 37 B6 B4 80 84 E2 60 AD
00080:  F7 B1 07 4A 89 18 28 62 C5 BF FE 64 86 28 35 41
00090:  33 0B 15 0F E4 8A 73 7C B3 E5 BB 04 3E 4A 11 34
000A0:  03 5A 6D 47 9B 18 93 1B 41 C9 BE 9A 7E 2A FC
000B0:  24 45 76 FE 4E 23 56 84 52 93 B0 31 78 E2 EC 00
000C0:  3C A8 A8 14 32 4F 16 35 0B C0 AB 53 41 87 DE 86
000D0:  C7 6B E2 9A 94 0A 8D B2 AD 71 64 6A A0 C9 52 FD
000E0:  F4 11 20 65 48 81 3E B9 F7 54 A1

---------------------------Server---------------------------

PMSEXP extracted:

00000:  25 0D 1B 67 A2 70 AB 04 D3 F6 54 18 E1 D3 80 B4
00010:  CB 94 5F 0A 3D CA 51 50 0C F3 A1 BE F3 7F 76 C0
00020:  73 41 A9 83 9C CF 6C BA 71 89 DA 61 EB 67 17 6C

HASH(r_c | r_s):

00000:  C3 EF 04 28 D4 B7 A1 F4 C5 02 5F 2E 65 DD 2B 2E
00010:  A5 83 AE EF DB 67 C7 F4 21 4A 6A 29 8E 99 E3 25

Export key generation. r value:
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E

Export key generation. UKM value:
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E

Export keys K_Exp_MAC | K_Exp_ENC used in KImp15 algorithm:

00000:  7D AC 56 E4 8A 4D C1 70 FA A8 FC BA E2 0D B8 45
00010:  45 0C CC C4 C6 32 8B DC 8D 01 15 7C EF A2 A5 F1
00020:  1F 1C BA D8 86 61 66 F0 1F FA AB 01 52 E2 4B F4
00030:  60 9D 5F 46 A5 C8 99 C7 87 90 0D 08 B9 FC AD 24

IV:

00000:  21 4A 6A 29 8E 99 E3 25

PMS:

00000:  A5 57 6C E7 92 4A 24 F5 81 13 80 8D BD 9E F8 56
00010:  F5 BD C3 B1 83 CE 5D AD CA 36 A5 3A A0 77 65 1D

---------------------------Client---------------------------
Random value \( k \) used in signature generation:
\[ 0x163962EEA268203E7C6B3F70BF8D4A36 \]
\[ 34CE6E2CFC424687951D70ACE0B4292A \]

Signature value \( sgn_c = \text{SIGN}_d_c(HM) \):

<table>
<thead>
<tr>
<th>Byte Range</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>F7 1F 43 62 45 5B C5 5B A8 9A 8F AF 01 82 88 EC</td>
</tr>
<tr>
<td>00010</td>
<td>00 B3 27 17 48 2E 76 24 B2 57 D9 79 7C 8F F6 02</td>
</tr>
<tr>
<td>00020</td>
<td>E3 15 FD BD 8D E5 6D 08 54 18 04 0E 1B 61 BB F6</td>
</tr>
<tr>
<td>00030</td>
<td>B3 01 AC 26 3D 50 03 8B 30 31 13 DB 36 17 50 3A</td>
</tr>
</tbody>
</table>

CertificateVerify message:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg_type</td>
<td>0F</td>
</tr>
<tr>
<td>length</td>
<td>00044</td>
</tr>
<tr>
<td>body:</td>
<td></td>
</tr>
<tr>
<td>algorithm:</td>
<td>08</td>
</tr>
<tr>
<td>hash:</td>
<td>40</td>
</tr>
<tr>
<td>signature:</td>
<td></td>
</tr>
<tr>
<td>length:</td>
<td>0040</td>
</tr>
<tr>
<td>vector:</td>
<td>F71F4362455BC55BA89A8FAF018288EC 00B32717482E7624B257D9797C8FF602 E315FDBD8DE56D085418040E1B61BBF6 B301AC263D50038B303113DB3617503A</td>
</tr>
</tbody>
</table>

Record layer message:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>16</td>
</tr>
<tr>
<td>version:</td>
<td></td>
</tr>
<tr>
<td>major:</td>
<td>03</td>
</tr>
<tr>
<td>minor:</td>
<td>03</td>
</tr>
<tr>
<td>length:</td>
<td>0048</td>
</tr>
<tr>
<td>fragment:</td>
<td>6FO00004408400040F71F4362455BC55BA89A8FAF018288EC00B32717482E7624B257D9797C8FF602E315FDBD8DE56D085418040E1B61BBF6B301AC263D50038B303113DB3617503A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte Range</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>16 03 03 00 48 0F 00 00 44 08 40 00 40 F7 1F 43 62 45 5B C5 5B A8 9A 8F AF 01 82 88 EC 00 B3 27</td>
</tr>
<tr>
<td>00010</td>
<td>62 45 5B C5 5B A8 9A 8F AF 01 82 88 EC 00 B3 27</td>
</tr>
<tr>
<td>00020</td>
<td>17 48 2E 76 24 B2 57 D9 79 7C 8F F6 02 E3 15 FD</td>
</tr>
</tbody>
</table>
00030: BD 8D E5 6D 08 54 18 04 0E 1B 61 BB F6 B3 01 AC
00040: 26 3D 50 03 8B 30 31 13 DB 36 17 50 17 3A

---------------------------Client---------------------------

HASH(HM):
00000: 9D 64 0D D8 B2 54 6B 87 05 CC 3E 67 F3 BB 83 2F
00010: 89 2A 5B D5 D4 5C A0 44 85 01 14 C2 E6 56 02 69

MS:
00000: E3 18 17 B0 EC 7F 3B C9 4A 8B C4 5F 89 12 DE C5
00010: 71 2A 7A 34 78 56 31 C0 4B AE 81 43 EE 17 90 B4
00020: C9 D3 68 0F 6C 9D E1 70 74 58 C8 75 62 4D B6 ED

Client connection key material
K_write_MAC|K_read_MAC|K_write_ENC|K_read_ENC|IV_write|IV_read:
00000: 50 52 5D 33 4E F7 00 6C 1D ED B8 B8 08 EA 03 CC
00010: CF 1F CB 3D 33 65 F9 72 E1 7C 7C 31 4E DD 97 90
00020: 6C 74 35 22 0A A1 B0 C6 DE 6A 1B 0F AC 29 B6 17
00030: 9E B3 23 86 62 25 E0 7F 30 4C A1 D1 27 75 86 29
00040: 7B 97 20 5D 7A 08 C2 CD 7F 60 3C 09 46 75 E6 C4
00050: CC 15 F2 84 0D 9A EC 63 F0 2A FF 51 DB D5 74 D2
00060: 76 6C 77 2B 83 2F CE 58 CB 4D E5 49 88 77 A6 7A
00070: A4 51 40 B2 ED 52 6E 61 65 0A 28 1B 32 56 35 BC
00080: CB 8E F9 4C 5B DF 5B 9F 47 48 B9 5B F1 B0 E0 BF

---------------------------Server---------------------------

HASH(HM):
00000: 9D 64 0D D8 B2 54 6B 87 05 CC 3E 67 F3 BB 83 2F
00010: 89 2A 5B D5 D4 5C A0 44 85 01 14 C2 E6 56 02 69

MS:
00000: E3 18 17 B0 EC 7F 3B C9 4A 8B C4 5F 89 12 DE C5
00010: 71 2A 7A 34 78 56 31 C0 4B AE 81 43 EE 17 90 B4
00020: C9 D3 68 0F 6C 9D E1 70 74 58 C8 75 62 4D B6 ED

Server connection key material
K_read_MAC|K_write_MAC|K_read_ENC|K_write_ENC|IV_read|IV_write:
00000: 50 52 5D 33 4E F7 00 6C 1D ED B8 B8 08 EA 03 CC
00010: CF 1F CB 3D 33 65 F9 72 E1 7C 7C 31 4E DD 97 90
00020: 6C 74 35 22 0A A1 B0 C6 DE 6A 1B 0F AC 29 B6 17
00030: 9E B3 23 86 62 25 E0 7F 30 4C A1 D1 27 75 86 29
00040: 7B 97 20 5D 7A 08 C2 CD 7F 60 3C 09 46 75 E6 C4
00050: CC 15 F2 84 0D 9A EC 63 F0 2A FF 51 DB D5 74 D2
00060: 76 6C 77 2B 83 2F CE 58 CB 4D E5 49 88 77 A6 7A
ChangeCipherSpec message:
  type: 01

Record layer message:
  type: 14
  version:
    major: 03
    minor: 03
  length: 0001
  fragment: 01

client_verify_data:

Finished message:
  msg_type: 14
  length: 000020
  body:
    verify_data: 987C13E6FA16F3D510AE830023587227
                 3290094C8FC7B5F0C7D747C42735F8F1

Record layer message:
  type: 16
version:
  major: 03
  minor: 03
length: 0034
fragment: 4DC53D655EDFD1843AF69ADBDE989C0B
  1F0C0A1A0FD1B3F458029D8F9989F8F9
  6C5C42971063A9B70714F412E4F6280F
  7C21601B

---------------------------Server---------------------------
ChangeCipherSpec message:
type: 01

Record layer message:
type: 14
version:
  major: 03
  minor: 03
length: 0001
fragment: 01

---------------------------Server---------------------------
HASH(HM):
00000: 4A 41 4C AD 20 F8 46 D8 F5 D1 05 26 10 A5 9D ED
00010: 6D 2B 1B B2 A8 9E 13 51 01 FC 9E 49 ED A8 0F B4

server_verify_data:
00000: 1E 93 7D A4 77 EE 1F 23 0A 41 D6 E9 D4 14 46 B7
00010: F2 1C A1 B2 E2 32 4A 55 2D 52 B3 25 5E B4 3D DF

---------------------------Server---------------------------
Finished message:
msg_type: 14
length: 000020
body:
  verify_data: 1E937DA477EE1F230A41D6E9D41446B7
                F21CA1B2E2324A552D52B3255EB43DDF

00000:  14 00 00 20 1E 93 7D A4 77 EE 1F 23 0A 41 D6 E9
00010:  D4 14 46 B7 F2 1C A1 B2 E2 32 4A 55 2D 52 B3 25
00020:  5E B4 3D DF

Record layer message:
type: 16
version:
  major: 03
  minor: 03
length: 0034
fragment: F9887C3654B6CCC6AE7D7B18A46C663F
          3D1DAF30C9A853A9871077FD05CA063B
          2C81BCC9D59FA6E3F5FAD9B2599BB5B6
          854A2D76

00000:  16 03 03 00 34 F9 88 7C 36 54 B6 CC C6 AE 7D 7B
00010:  18 A4 6C 66 3F 3D 1D AF 30 C9 A8 53 A9 87 10 77
00020:  FD D5 CA 06 3B 2C 81 BC C9 D5 9F A6 E3 F5 FA D9
00030:  B2 59 9B B5 86 85 4A 2D 76

---------------------------Client---------------------------

Application data:
00000:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00010:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Record layer message:
type: 17
version:
  major: 03
  minor: 03
length: 0030
fragment: F14F06FB8557408846080690E7A5525D
          1C6E9C901D24025486AB79728BF63D06
          5C09C2723306D65CFF0B5BA87504969

00000:  17 03 03 00 30 F1 4F 06 FB 85 57 40 88 46 08 06
00010:  90 E7 A5 52 5D 1C 6E 9C 90 1D 24 02 54 86 AB 79
00020:  72 8B F6 3D 06 5C 09 C2 72 33 00 6D 65 CF F0 B5
00030:  BA 87 50 49 69
Internet-Draft       GOST Cipher Suites for TLS 1.2             May 2021

---------------------------Server---------------------------

Application data:
00000:   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00010:   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

Record layer message:
type:                    17
version:
  major:                 03
  minor:                 03
length:                  0030
fragment:                1561E52A8B6DB258746FFE18F3CDCB11
                       1D0173AF2E5C13741C99BFF13B47CD32
                       B3CED856A9506E706A2340D5841AB114

00000:   17 03 03 00 30 15 61 E5 2A 8B 6D B2 58 74 6F FE
00010:   18 F3 CD CB 11 1D 01 73 AF 2E 5C 13 74 1C 99 BF
00020:   F1 3B 47 CD 32 B3 CE D8 56 A9 50 6E 70 6A 23 40
00030:   D5 84 1A B1 14

---------------------------Client---------------------------

close_notify alert:
Alert:
  level:                 01
  description:           00

00000:   01 00

Record layer message:
type:                    15
version:
  major:                 03
  minor:                 03
length:                  0012
fragment:                E530C164642A078CEF528CB465E9DA7E
                       AD4D

00000:   15 03 03 00 12 E5 30 C1 64 64 2A 07 8C EF 52 8C
00010:   B4 65 E9 DA 7E AD 4D

---------------------------Server---------------------------

close_notify alert:
Alert:
A.2. Test Examples for CNT_IMIT cipher suites

A.2.1. Record Examples

It is assumed that during Handshake following keys were established:

- MAC key:
  00000: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
  00010: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

- Encryption key:
  00000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  00010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

- IV:
  00000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

seqnum = 0

Application data:
  00000: 00 00 00 00 00 00 00 00 00

Plaintext:
  00000: 17 03 03 00 07 00 00 00 00 00 00 00 00 00 00

MAC:
  00000: 30 01 34 a1

Ciphertext:
  00000: 17 03 03 00 0b 86 71 cd bf 3c 1a ae 0f 62 4b 04
A.2.2. Handshake Examples

The ClientHello.extensions and the ServerHello.extensions fields contain the renegotiation_info extension (see [RFC5746]) in the following examples.

Server certificate curve OID:
id-tc26-gost-3410-12-512-paramSetA, "1.2.643.7.1.2.1.2.1"

Server public key Q_s:
x = 0x16DB0566C0278AC8204143994824236D
   97F36A13D5433EE990B2EAC859D2E9B7A
   E054794655389158B824923E3841B14
   24FD89F221701C89D9A3BF6A9F946795

y = 0x0D01E80DEC5BD23C8BC6885F12BEB1635
   A5AE7AD50DE24FB8FD02CB285A4AEF5A

plaintext:
00000: 17 03 03 08 00 00 00 00 00 00 00 00 00 00 00 00
00010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
....
007ff: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00804: 00 00 00 00

MAC:
00000: f7 c3 8b 8a

Ciphertext:
00000: 17 03 03 08 04 cf aa 0c b4 2f a5 a4 7a 13 3d 73
00010: b9 f2 c0 b0 4f 8c a2 55 52 f8 56 bc be 6a 58 fa
....
007ff: 3e e2 c7 6f a2 30 a0 44 be 21 dc 8e 1a 96 f9 a8
00804: 88 1f ad 83 45 96 96 84 47
Server private key \( d_s \):
\[
0x5F1E83AFA2C4CB2C5633C51380E84E37
4B013EE7C23833079080CE914B442D4
34EB016D23FB63FEDC18B62D9DA93D26
B3B9CE6F663B383303BD5930ED41608B
\]

---------------------------Client---------------------------

ClientHello message:

msg_type: 01
length: 00003a

body:

client_version:
  major: 03
  minor: 03
random: 6A523D6880DCC2DC75CCC43CFD04B616
  F5C3757B8077B76A9B504949FD3BFDB8

session_id:
  length: 00
  vector: --

cipher_suites:
  length: 0002
  vector:
    CipherSuite: C102

compression_methods:
  length: 01
  vector:
    CompressionMethod: 00

extensions:
  length: 000F
  Extension: /* signature_algorithms */
    extension_type: 000D
    extension_data:
      length: 0006
      vector:
        supported_signature_algorithms:
          length: 0004
          vector:
            /* 1 pair of algorithms */
            hash: 08
            signature: 41
            /* 2 pair of algorithms */
            hash: 08
            signature:
40
Extension: /* renegotiation_info */
extension_type: FF01
extension_data:
  length: 0001
  vector:
    renegotiated_connection:
      length: 00
      vector: --

00000: 01 00 00 3A 03 03 6A 52 3D 68 80 DC C2 DC 75 CC
00010: C4 3C FD 04 B6 16 F5 C3 75 7B 80 77 B7 6A 9B 50
00020: 49 49 FD 3B FD B8 00 00 02 C1 02 01 00 00 0F 00
00030: 0D 00 06 00 04 08 41 08 40 FF 01 00 01 00

Record layer message:
type: 16
version:
  major: 03
  minor: 03
length: 003e
fragment: 0100003A03036A523D6880DC2DC75CC
          C43CFD04B616F5C3757B8077B76A9B50
          4949FD3BFDB8000002C1020100000FF00
          0D0006000408410840FF01000001

00000: 16 03 03 00 3E 01 00 00 3A 03 03 6A 52 3D 68 80
00010: DC C2 DC 75 CC C4 3C FD 04 B6 16 F5 C3 75 7B 80
00020: 77 B7 6A 9B 50 49 49 FD 3B FD B8 00 00 02 C1 02
00030: 01 00 00 0F 00 0D 00 06 00 04 08 41 08 40 FF 01
00040: 00 01 00

--------------------------------------- Server --------------------------------------

ServerHello message:
msg_type: 02
length: 00004D
body:
  client_version:
    major: 03
    minor: 03
  random: FE92C9516D0E1A67A04C33CD7F2C90B1
          5E76DCC30815C19F92A6D100915AF2DB
  session_id:
    length: 20
    vector: 12AAA5E5779014711CCD6D265BDEE519
            1026431C83768EE5EB5A157F940BE9FB
cipher_suite:
    CipherSuite: C102
compression_method:
    CompressionMethod: 00
extensions:
    length: 0005
    Extension: /* renegotiation_info */
    extension_type: FF01
    extension_data:
        length: 0001
        vector:
            renegotiated_connection:
                length: 00
                vector: --

00000: 02 00 00 4D 03 03 FE 92 C9 51 6D 0E 1A 67 A0 4C
00010: 33 CD 7F 2C 90 B1 5E 76 DC C3 08 15 C1 9F 92 A6
00020: D1 00 91 5A F2 DB 20 12 AA A5 E5 77 90 14 71 1C
00030: CD 6D 26 5B DE E5 19 10 26 43 1C 83 76 8E E5 EB
00040: 5A 15 7F 94 0B E9 FB C1 02 00 00 05 FF 01 00 01
00050: 00

Record layer message:
    type: 16
    version:
        major: 03
        minor: 03
    length: 0051
    fragment: 0200004D0303FE92C9516D0E1A67A04C
            33CD7F2C90B15E76DC30815C19F92A6
            D100915AF2DB2012AA5E5779014711C
            CD6D265BDEE5191026431C83768EE5EB
            5A157F940BE9FBC102000005FF010001
            00

---------------------------Server---------------------------

Certificate message:
    msg_type: 0B
    length: 000266
body:
  certificate_list:
    length:        000263
    vector:
      ASN.1Cert:
        length:        000260
        vector:        3082025C308201C8A00302010202147894DC92097789191642F1DAEDC26BA3B5104300A06082A85030701010303306C12D51F99C98A4A9904F0EA5486FED7...
          6C12D51F99C98A4A9904F0EA5486FED7
FF66AB8EB2425E1ACEAE8A758BD843B
E1A8F6FEBF673015FED7AB86533DBF20

  00000:    0B 00 02 66 00 02 63 00 02 60 30 82 02 5C 30 82
  00010:    01 C8 A0 03 02 01 02 02 14 78 94 DC 9D 92 09 77
  00020:    08 2A 85 03 07 01 03 03 30 19 31 17 30 15 06
  00030:    03 55 04 03 13 0E 43 41 20 43 65 72 74 69 66 69
  00040:    63 61 74 65 30 1E 7D 31 30 31 30 32 30 30 30
  00050:    03 30 31 31 5A 17 0D 32 32 30 31 30 32 30 30 30
  00060:    30 32 31 5A 30 21 31 1F 30 1D 06 03 55 04 03 13
  00070:    16 53 65 72 65 72 74 20 35 31 32 20 43 65 72 74
  00080:    69 66 69 63 61 74 65 30 81 AA 30 21 06 08 2A 85
  00090:    03 07 01 01 01 02 30 15 06 09 2A 85 03 07 01 02
  000a0:    01 02 01 06 08 2A 85 03 07 01 01 02 03 03 81 84
  000b0:    00 04 81 80 95 67 94 9F 6A BF A3 D9 89 1C 70 21
  000c0:    F2 89 FD 24 14 1B 84 E3 23 29 24 B8 58 91 38 55
  000d0:    46 79 54 E0 7A 9B 2E 9D 85 AC 2E 0B 99 3E 43 D5
  000e0:    13 6A F3 97 6D 23 24 48 99 43 41 20 C8 8A 27 C0
  000f0:    66 05 DB 16 CF D4 23 24 48 99 43 41 20 C8 8A 27 C0
  00100:    76 44 04 22 82 32 F7 F7 F2 26 98 62 8D DA FF AA
  00110:    99 BB 6F 7D 5A E6 4A 5A 28 CB 02 FD B8 4F E2 0D
  00120:    D5 7A AE A5 35 16 BB 2F B1 85 6B BC C8 23 BD C5
  00130:    DE 80 1E D0 A3 81 93 30 81 90 30 0C 06 03 55 1D
  00140:    13 01 01 FF 04 02 30 00 30 1A 06 03 55 1D 11 04
  00150:    13 30 11 02 09 0C 6F 63 61 6C 66 6F 73 74 87 04
  00160:    7F 00 00 01 30 13 06 03 55 1D 25 04 0C 30 0A 06
  00170:    0E 2B 06 01 05 05 07 03 01 30 0F 06 03 55 1D 0F
  00180:    01 01 FF 04 05 03 03 07 B0 00 30 0D 06 03 55 1D
  00190:    0E 04 16 04 14 AE 46 41 1B FD B3 08 C3 39 03 47
  001a0:    57 57 2B 0F BF A3 6F 9A 99 30 1F 06 03 55 1D 23
  001b0:    04 18 30 16 80 14 7F 7B 7A 15 61 A6 F2 18 A2 E3
  001c0:    48 3B C6 39 D9 7F 42 DB 6D AF 30 0A 06 08 2A 85
  001d0:    00 05 05 05 07 03 01 30 0F 06 03 55 1D 0F
  001e0:    04 18 30 16 80 14 7F 7B 7A 15 61 A6 F2 18 A2 E3
  001f0:    54 8A 25 6D 2A 18 7C A8 4D 72 4F E1 EF A7 E5 36
  00200:    67 2E 79 1F 8A 0C B6 74 1E B1 63 E2 96 37 8C 5B
  00210:    82 83 EE DA B4 1B A4 22 1E BC E2 05 F6 F8 79 CF
Record layer message:

type:                    16
version:                 03
    major:                 03
    minor:                 03
length:                  026A
fragment:                0B0002660002630002603082025C3082
                        01C8A00302010202147894DC9D920977
                        809191642F1DAEDC26BA3B5104300A06
                        . . .
                        EC99C7CD239F6F2025A86C12D51F99C9
                        8A4A9904F0EA5486FED7FF6AB8EB242
                        5E1ACEAE8A758BDF843BE1A8F6E6BF67
                        3015FED7AB86533DBF20

00000:   16 03 03 02 6A 0B 00 02 66 00 02 63 00 02 60 30
00010:   82 02 5C 30 82 01 C8 A0 03 02 01 02 02 14 78 94
00020:   DC 9D 92 09 77 80 91 91 64 2F 1D AE DC 26 BA 3B
00030:   51 04 30 0A 06 08 2A 85 03 07 01 01 03 03 30 19
00040:   31 17 30 15 06 03 55 04 03 13 0E 43 41 20 43 65
00050:   72 74 69 66 69 63 61 74 65 30 1E 17 0D 31 38 30
00060:   31 38 32 30 30 30 30 30 31 31 5A 17 0D 32 32 30 31
00070:   30 32 30 30 30 30 32 31 5A 30 21 31 1F 30 1D 06
00080:   03 55 04 03 13 16 53 65 72 66 63 61 74 65 30 81 AA 30
00090:   21 06 08 2A 85 03 07 01 01 01 02 30 15 06 09 2A
000a0:   85 03 07 01 02 01 02 01 06 08 2A 85 03 07 01 01
000b0:   02 03 03 81 84 00 04 81 80 95 67 94 9F 6A BF A3
000c0:   D9 89 1C 70 21 F2 89 FD 24 14 1B 84 E3 23 29 24
000d0:   B9 58 91 38 55 46 79 54 E0 7A 9B 2E 9D 85 AC 2E
000e0:   0B 99 3E 43 D5 13 6A F3 97 6D 23 24 48 99 43 41
000f0:   20 C8 8A 27 C0 66 65 DB 16 CF D4 6F A0 C4 77 20
00100:   08 6D A0 15 16 76 44 04 22 82 32 F7 F7 F2 26 98
00110:   62 80 DA FF AA 99 BB 6F 7D 5A E6 4A 5A 28 CB 02
00120:   FD BB 4F E2 0D D5 7A AE A5 35 16 BB 2B F1 85 6B
00130:   BC C8 23 BD C5 DE 80 1E D0 A3 81 93 30 81 90 30
00140:   0C 06 03 55 1D 13 01 01 FF 04 02 30 00 30 1A 06
00150:   03 55 1D 11 04 13 30 11 82 09 6C 6F 63 61 6C 68
00160:   6F 73 74 87 04 7F 00 00 01 30 13 06 03 55 1D 25
00170:   04 0C 30 0A 06 08 2B 06 01 05 05 07 03 01 30 0F
00180:   06 03 55 1D 0F 01 01 FF 04 05 03 03 07 B0 00 30
00190:   1D 06 03 55 1D 0E 04 16 04 14 AE 46 41 1B FD B3
ServerHelloDone message:
`msg_type:` 0E
`length:` 000000
`body:` --

Record layer message:
`type:` 16
`version:`
  `major:` 03
  `minor:` 03
`length:` 0004
`fragment:` 0E000000

PMS:
`00000: CE 0D D6 B6 70 42 12 15 2B E4 69 5A 7E 89 F6 4C`
`00010: 89 29 A4 0D BF 0A 5A 55 C2 CE 00 2B 06 BA B6 2F`

Random d_eph value:
0xC96486B1A3732389A162F5AD0145D537
43C9AC27D42ACF1091CE7EF676C3CCA
0FC879B2DA3C1607648BAEB96471BD2
078DF5CAA4FA83ECC8FFD63C8E5D56

Q_eph ephemeral key:
\[ x = 0x4B9CB381BCC737E493E43B2D7FD95BFE \
\] 2AEF6BE8F6224882E5E559ADA08170DC
y = 0x95CEF28392C846A5EEFCB51C84E4960A
77B77D0D85EBD22061BFDA0013C5AB6C
42DD04973F65D2AE8A5427A536872
CF2D68F5F722C4640D7AAF2E0194FBD0

HASH(r_c | r_s):
00000:   FB F3 9D 10 E8 00 AF 70 E7 AA 22 C1 10 DA 94 A9
00010:   9A 58 98 D8 45 27 C7 CB DE C1 1E 53 39 90 6A 1A

K_EXP:
00000:   3F D9 99 D1 68 4A 15 CC 9B DD 5A 35 06 7A F6 98
00010:   17 15 00 22 E0 95 54 AC 79 1A 60 F1 61 F5 53 49

IV:
00000:   FB F3 9D 10 E8 00 AF 70

CEK_ENC:
00000:   D6 22 D1 67 A5 64 2E 29 52 5A 29 5C B9 F2 8F 96
00010:   F2 8B 0E FA A7 D3 A2 BE E1 49 B0 11 78 C2 DF D5

CEK_MAC:
00000:   4C 93 36 57

PMSEXP:
00000:   FB F3 9D 10 E8 00 AF 70 D6 22 D1 67 A5 64 2E 29
00010:   52 5A 29 5C B9 F2 8F 96 F2 8B 0E FA A7 D3 A2 BE
00020:   E1 49 B0 11 78 C2 DF D5 4C 93 36 57

---------------------------Client---------------------------

ClientKeyExchange message:
msg_type:                10
length:                  0000F5
body:
  exchange_keys:         3081F23081EF30280420D622D167A564
                          2E29525A295CB9F28F96F28B0EFAA7D3
                          A2BEE149B01178C2DFD504044C933657
                          ... DABF6120D2EB850D7DB7770A96E4841C
                          B5FCEEA546C89283F2CE950408FBF39D
                          10E800AF70

 00000:   10 00 00 F5 30 81 F2 30 81 EF 30 28 04 20 D6 22
 00010:   D1 67 A5 64 2E 29 52 5A 29 5C B9 F2 8F 96 F2 8B
 00020:   0E FA A7 D3 A2 BE E1 49 B0 11 78 C2 DF D5 04 04
Record layer message:

type:          16
version:       03
  major:       03
  minor:       03
length:        00F9
fragment:

00000: 16 03 03 00 F9 10 00 00 F5 30 81 F2 30 81 EF 30
00010: 28 04 20 D6 22 D1 67 A5 64 2E 29 52 5A 29 5C B9
00020: F2 8F 96 F2 8B 0E FA A7 D3 A2 BE E1 49 B0 11 78
00030: C2 DF D5 04 04 4C 93 36 57 A0 81 C2 06 09 2A 85 03 07 01 02 05
00040: 02 38 15 06 09 2A 85 03 07 01 02 01 02 01 06 08
00050: 2A 85 03 07 01 01 02 03 03 81 84 00 04 81 80 FB
00060: 4B F8 AB 9F 71 A7 A6 62 57 0C 77 C3 39 61 DD 60
00070: FC 3C 15 95 01 B5 D2 23 B3 A1 B3 15 A8 49 DC
00080: 70 81 A0 AD 59 E5 E5 48 22 F6 E8 6B EF 2A FE
00090: 5B D9 7F 2D 3B E4 93 E4 37 C7 BC 81 B3 9C 4B D0
000A0: FB 94 01 2E AF 7A 00 64 C4 22 F7 F5 68 2D CF 72
000B0: 68 3D A5 27 54 8A EB 2A 5D F6 73 49 D0 DD 42 6C
000C0: AB C5 13 00 DA BF 61 20 D2 EB 85 0D 7D B7 77 0A
000D0: 9E E4 84 1C B5 FC EE A5 46 C8 92 83 F2 CE 95 04
000E0: 08 FB F3 9D 10 E8 00 AF 70

---------------------------Client---------------------------
HASH(HM):
00000:  F8 D6 FE EB 17 64 4D 17 B0 38 36 A6 51 EB 87 69
00010:  BD EA A2 D3 EB 18 47 F6 91 91 42 7C 30 D0 17 8E

MS:
00000:  BE 57 46 C8 BB B7 84 7E 97 8F D4 C9 4F 52 34 52
00010:  44 2C 8E B1 72 FD E6 28 1C 18 C5 44 63 B1 F9 4C
00020:  2B D9 81 40 05 41 6D BB 0F 90 A5 7E A4 E0 6B 50

Client connection key material
K_write_MAC|K_read_MAC|K_write_ENC|K_read_ENC|IV_write|IV_read:
00000:  F3 37 F6 A8 6F F3 1F CA 52 EA 64 7C DE E3 B7 83
00010:  34 AB 77 B5 7F E0 DB 2F C0 C8 71 EC DC AC A5 A8
00020:  FB A0 4C 21 32 82 3A 24 96 EF 93 6F 0E BC F3 0E
00030:  A0 CB 7E AF 6C A7 94 75 4F 1F 45 B1 77 22 DE B4
00040:  4E 5B C3 2D 44 30 AF 58 93 11 6A CF 81 A3 BE 0C
00050:  90 D2 EA 8E 76 E0 84 07 28 BA F5 E2 B2 F9 40 C0
00060:  AE 18 26 7B B6 34 C1 6A 1D 1A C1 24 73 50 95 4B
00070:  2F EE 9B 77 F3 0D 18 D5 54 01 2B 43 78 60 87 0A
00080:  D9 21 A8 4B 07 FF 98 AF 8C 82 38 6B 91 FB BA 64

---------------------------Server---------------------------
PMSEXP extracted:
00000:  FB F3 9D 10 E8 00 AF 70 D6 22 D1 67 A5 64 2E 29
00010:  52 5A 29 5C B9 F2 8F 96 F2 8B 0E FA A7 D3 A2 BE
00020:  E1 49 B0 11 78 C2 DF D5 4C 93 36 57

HASH(r_c | r_s):
00000:  FB F3 9D 10 E8 00 AF 70 E7 AA 22 C1 10 DA 94 A9
00010:  9A 58 98 D8 45 27 C7 CB DE C1 1E 53 39 90 6A 1A

K_EXP:
00000:  3F D9 99 D1 68 4A 15 CC 9B DD 5A 35 06 7A F6 98
00010:  17 15 00 22 E0 95 54 AC 79 1A 60 F1 61 F5 53 49

PMS:
00000:  CE 0D D6 B6 70 42 12 15 2B E4 69 5A 7E 89 F6 4C
00010:  89 29 A4 0D BF 0A 5A 55 C2 CE 00 2B 06 BA B6 2F

---------------------------Server---------------------------
HASH(HM):
00000:  F8 D6 FE EB 17 64 4D 17 B0 38 36 A6 51 EB 87 69
00010:  BD EA A2 D3 EB 18 47 F6 91 91 42 7C 30 D0 17 8E

MS:
Client connection key material

K_read_MAC|K_write_MAC|K_read_ENC|K_write_ENC|IV_read|IV_write:

000000:   F3 37 F6 A8 6F F3 1F CA 52 EA 64 7C DE E3 B7 83
000010:   FB A0 4C 21 32 82 3A 24 96 EF 93 6F 0E BC F3 0E
000020:   A0 CB 7E AF 6C A7 94 75 4F 1F 45 B1 77 22 DE B4
000030:   4E 5B C3 2D 44 30 AF 58 93 11 6A CF 81 A3 BE 0C
000040:   90 D2 EA 8E 76 E0 84 07 28 BA F5 E2 B2 F9 40 C0
000050:   AE 18 26 7B B6 34 C1 6A 1D 1A C1 24 73 50 95 4B
000060:   2F EE 9B 77 F3 0D 18 D5 54 01 2B 43 78 60 87 0A
000070:   D9 21 A8 4B 07 FF 98 AF 8C 82 38 6B 91 FB BA 64

---------------------------Client---------------------------

ChangeCipherSpec message:
type:                    01

00000:   01

Record layer message:
type:                    14
version:                 03
length:                  00001
fragment:                01

00000:   14 03 03 00 01 01

---------------------------Client---------------------------

HASH(HM):                

00000:   F8 D6 FE EB 17 64 4D 17 B0 38 36 A6 51 EB 87 69
000010:   BD EA A2 D3 EB 18 47 F6 91 91 42 7C 30 D0 17 8E

Finished message:
msg_type:                14
length:                  00000C
body:                    
   verify_data:           D3EE1DEA725CD7080C744311

00000:   14 00 00 0C D3 EE 1D EA 72 5C D7 08 0C 74 43 11
Record layer message:
type: 16
version:
  major: 03
  minor: 03
length: 0014
fragment: 8854A0ED0CCBDAE076FA7D22D763A8D1
  AF701BBB

00000: 16 03 03 00 14 88 54 A0 ED 0C CB DA E0 76 FA 7D
00010: 22 D7 63 A8 D1 AF 70 1B BB

---------------------------Server---------------------------

ChangeCipherSpec message:
type: 01

00000: 01

Record layer message:
type: 14
version:
  major: 03
  minor: 03
length: 0001
fragment: 01

00000: 14 03 03 00 01 01

---------------------------Server---------------------------

HASH(HM):
00000: 9C 9F C4 E3 32 5B 5F B3 70 B9 94 2A 71 D2 6E F0
00010: 10 71 D8 A5 A1 8F 69 E8 C2 0B 70 CC 90 E9 A9 46

Finished message:
msg_type: 14
length: 00000C
body:
  verify_data: D6A2A697E9F23DB0F9017A79

00000: 14 00 00 0C D6 A2 A6 97 E9 F2 3D B0 F9 01 7A 79

Record layer message:
type: 16
version:
  major: 03
  minor: 03
length: 0014
fragment: 7BDDDBB3C0A6A4A9E302B468CCD5CF786
665FFEBC

00000:  16 03 03 00 14 7B DD BB 3C 0A 6A 4A 9E 30 2B 46
00010:  8C CD 5C F7 86 66 5F FE BC

---------------------------Client---------------------------

Application data:
00000:  48 45 4C 4F 0A

Record layer message:
type: 17
version:
  major: 03
  minor: 03
length: 0009
fragment: A8951D9389D1AEFE3B

00000:  17 03 03 00 09 A8 95 1D 93 89 D1 AE FE 3B

---------------------------Server---------------------------

Application data:
00000:  48 45 4C 4F 0A

Record layer message:
type: 17
version:
  major: 03
  minor: 03
length: 0009
fragment: 0F368E5CEC86B4F8D7

00000:  17 03 03 00 09 0F 36 8E 5C EC 86 B4 F8 D7

---------------------------Client---------------------------

close_notify alert:
Alert:
  level: 01
Record layer message:
type: 15
version:
  major: 03
  minor: 03
length: 0006
fragment: F91FCD98F309

00000: 15 03 03 00 06 F9 1F CD 98 F3 09

---------------------------Server---------------------------
close_notify alert:
Alert:
  level: 01
description: 00

00000: 01 00

Record layer message:
type: 15
version:
  major: 03
  minor: 03
length: 0006
fragment: 117B57AD5FED

00000: 15 03 03 00 06 11 7B 57 AD 5F ED

Appendix B. Contributors

- Evgeny Alekseev
  CryptoPro
  alekseev@cryptopro.ru

- Ekaterina Smyshlyaeva
  CryptoPro
  ess@cryptopro.ru
Appendix C. Acknowledgments

Authors' Addresses

Stanislav Smyshlyaev (editor)
CryptoPro
18, Suschevsky val
Moscow 127018
Russian Federation

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

Dmitry Belyavsky
Cryptocom
14/2 Kedrova st
Moscow 117218
Russian Federation

Email: beldmit@gmail.com

Markku-Juhani O. Saarinen
Independent Consultant
Email: mjos@iki.fi