

---

Stream: Internet Engineering Task Force (IETF)  
RFC: [9189](#)  
Category: Informational  
Published: February 2022  
ISSN: 2070-1721  
Authors: S. Smyshlyayev, Ed. D. Belyavskiy M-J. Saarinen E. Alekseev  
*CryptoPro* *Cryptocom* *Independent Consultant* *CryptoPro*

# RFC 9189

## GOST Cipher Suites for Transport Layer Security (TLS) Protocol Version 1.2

---

### Abstract

This document specifies three new cipher suites, two new signature algorithms, seven new supported groups, and two new certificate types for the Transport Layer Security (TLS) protocol version 1.2 to support the Russian cryptographic standard algorithms (called "GOST" algorithms). This document specifies a profile of TLS 1.2 with GOST algorithms so that implementers can produce interoperable implementations.

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, signature algorithms, supported groups, and certificate types.

### Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <https://www.rfc-editor.org/info/rfc9189>.

### Copyright Notice

Copyright (c) 2022 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 publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction
2. Conventions Used in This Document
3. Basic Terms and Definitions
4. Cipher Suite Definitions
  - 4.1. Record Payload Protection
    - 4.1.1. CTR\_OMAC
    - 4.1.2. CNT\_IMIT
  - 4.2. Key Exchange and Authentication
    - 4.2.1. Hello Messages
    - 4.2.2. Server Certificate
    - 4.2.3. CertificateRequest
    - 4.2.4. ClientKeyExchange
      - 4.2.4.1. CTR\_OMAC
      - 4.2.4.2. CNT\_IMIT
    - 4.2.5. CertificateVerify
    - 4.2.6. Finished
  - 4.3. Cryptographic Algorithms
    - 4.3.1. Block Cipher
    - 4.3.2. MAC Algorithm
    - 4.3.3. Encryption Algorithm
    - 4.3.4. PRF and HASH Algorithms
    - 4.3.5. SNMAX Parameter
5. New Values for the TLS SignatureAlgorithm Registry
6. New Values for the TLS Supported Groups Registry

## 7. New Values for the TLS ClientCertificateType Identifiers Registry

## 8. Additional Algorithms

### 8.1. TLSTREE

#### 8.1.1. Key Tree Parameters

### 8.2. Key Export and Key Import Algorithms

#### 8.2.1. KExp15 and KImp15 Algorithms

#### 8.2.2. KExp28147 and KImp28147 Algorithms

### 8.3. Key Exchange Generation Algorithms

#### 8.3.1. KEG Algorithm

#### 8.3.2. KEG\_28147 Algorithm

### 8.4. gostIMIT28147

## 9. IANA Considerations

## 10. Historical Considerations

## 11. 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 Cipher Suite

##### A.1.1.2. TLS\_GOSTR341112\_256\_WITH\_KUZNYECHIK\_CTR\_OMAC Cipher Suite

#### A.1.2. Record Examples

##### A.1.2.1. TLS\_GOSTR341112\_256\_WITH\_MAGMA\_CTR\_OMAC Cipher Suite

##### A.1.2.2. TLS\_GOSTR341112\_256\_WITH\_KUZNYECHIK\_CTR\_OMAC Cipher Suite

#### A.1.3. Handshake Examples

##### A.1.3.1. TLS\_GOSTR341112\_256\_WITH\_MAGMA\_CTR\_OMAC Cipher Suite

##### A.1.3.2. TLS\_GOSTR341112\_256\_WITH\_KUZNYECHIK\_CTR\_OMAC Cipher Suite

## [A.2. Test Examples for CNT\\_IMIT Cipher Suites](#)

### [A.2.1. Record Examples](#)

### [A.2.2. Handshake Examples](#)

[Contributors](#)

[Authors' Addresses](#)

## 1. Introduction

This document specifies three new cipher suites, two new signature algorithms, seven new supported groups, and two new certificate types for the Transport Layer Security (TLS) protocol version 1.2 [RFC5246] to support the set of Russian cryptographic standard algorithms (called "GOST" algorithms). This document specifies a profile of TLS 1.2 with GOST algorithms so that implementers can produce interoperable implementations. The profile of TLS 1.2 with GOST algorithms uses the hash algorithm GOST R 34.11-2012 [RFC6986], the signature algorithm GOST R 34.10-2012 [RFC7091], and two types of cipher suites: the CTR\_OMAC cipher suites and the CNT\_IMIT cipher suite.

The CTR\_OMAC cipher suites use the GOST R 34.12-2015 (see [RFC7801] and [RFC8891]) block ciphers.

The CNT\_IMIT cipher suite uses the GOST 28147-89 [RFC5830] block cipher.

This document specifies the profile of the TLS protocol version 1.2 with GOST algorithms. The profile of the TLS protocol version 1.3 [RFC8446] with GOST algorithms is specified in a separate document [DraftGostTLS13].

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, signature algorithms, supported groups, and certificate types.

## 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:

---

$B_t$	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 $A = (a_1, a_2, \dots, a_t)$ , where $a_1, a_2, \dots, a_t$ are in $\{0, \dots, 255\}$ .
$B^*$	the set of all byte strings of a finite length (hereinafter referred to as "strings"), including the empty string.
$A[i..j]$	the string $A[i..j] = (a_i, a_{i+1}, \dots, a_j)$ in $B_{\{j-i+1\}}$ , where $A = (a_1, \dots, a_t)$ in $B_t$ and $1 \leq i \leq j \leq t$ .
$L(A)$	the length of the byte string $A$ in bytes.
$A   C$	concatenation of strings $A$ and $C$ both belonging to $B^*$ , i.e., a string in $B_{\{L(A)+L(C)\}}$ , where the left substring in $B_{L(A)}$ is equal to $A$ and the right substring in $B_{L(C)}$ is equal to $C$ .
$A \text{ XOR } C$	bitwise exclusive-or of byte strings $A$ and $C$ both belonging to $B_t$ (both are of length $t$ bytes), i.e., a string in $B_t$ such that if $A = (a_1, a_2, \dots, a_t)$ and $C = (c_1, c_2, \dots, c_t)$ , then $A \text{ XOR } C = (a_1 \text{ (xor) } c_1, a_2 \text{ (xor) } c_2, \dots, a_t \text{ (xor) } c_t)$ , where (xor) is bitwise exclusive-or of bytes.
$i \& j$	bitwise AND of unsigned integers $i$ and $j$ .
$STR_t$	the transformation that maps an integer $i = 256^{\{t-1\}} * i_1 + \dots + 256 * i_{\{t-1\}} + i_t$ into the byte string $STR_t(i) = (i_1, \dots, i_t)$ in $B_t$ (the interpretation of the integer as a byte string in big-endian format).
$str_t$	the transformation that maps an integer $i = 256^{\{t-1\}} * i_t + \dots + 256 * i_2 + i_1$ into the byte string $str_t(i) = (i_1, \dots, i_t)$ in $B_t$ (the interpretation of the integer as a byte string in little-endian format).
$INT$	the transformation that maps a string $a = (a_1, \dots, a_t)$ in $B_t$ into the integer $INT(a) = 256^{\{t-1\}} * a_1 + \dots + 256 * a_{\{t-1\}} + a_t$ (the interpretation of the byte string in big-endian format as an integer).
$int$	the transformation that maps a string $a = (a_1, \dots, a_t)$ in $B_t$ into the integer $int(a) = 256^{\{t-1\}} * a_t + \dots + 256 * a_2 + a_1$ (the interpretation of the byte string in little-endian format as an integer).
$k$	the length of the block cipher key in bytes.
$n$	the length of the block cipher block in bytes.
$Q_c$	the public key stored in the client's certificate.
$d_c$	the private key that corresponds to the $Q_c$ key.
$Q_s$	the public key stored in the server's certificate.
$d_s$	the private key that corresponds to the $Q_s$ key.
$q_s$	an order of a cyclic subgroup of the elliptic curve points group containing point $Q_s$ .

---

- `P_s` the distinguished generator of the subgroup of order `q_s` that belongs to the same curve as `Q_s`.
- `r_c` the random string contained in the `ClientHello.random` field (see [RFC5246]).
- `r_s` the random string contained in the `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:

```
TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC = {0xC1, 0x00};
TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC = {0xC1, 0x01}.
```

The `CNT_IMIT` cipher suite has the following value:

```
TLS_GOSTR341112_256_WITH_28147_CNT_IMIT = {0xC1, 0x02}.
```

### 4.1. Record Payload Protection

The profile of TLS 1.2 with GOST algorithms requires that the compression not be used.

All of the cipher suites described in this document use such modes of operation (see [Section 4.3.3](#)) that protect the records in the same way as if they were protected by a stream cipher. 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 a certain sequence number. Note that with 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 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 the `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 `TLSCiphertext.fragment` that corresponds to the `seqnum` sequence number is calculated as follows:

1. The `MACValue_seqnum` value is generated using the Message Authentication Code (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`, and `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](#)):

```
ENCValue_seqnum = 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.

3. The fields of the `GenericStreamCipher` structure (see [Section 6.2.3.1](#) of [\[RFC5246\]](#)) for the `TLSCiphertext.fragment` value are defined by the `ENCValue_seqnum` value:

```
TLSCiphertext.fragment.content =
ENCValue_seqnum[1..length_seqnum],

TLSCiphertext.fragment.MAC = ENCValue_seqnum[length_seqnum +
1..length_seqnum + mac_length],
```

where `length_seqnum` is the `TLSCompressed.length` value of the record with the `seqnum` sequence number and `mac_length` is equal to 16 for the `TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC` cipher suite and 8 for the `TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC` cipher suite.

Note that the `CTR_OMAC` cipher suites use the authenticate-then-encrypt method (see [Appendix F.4](#) of [\[RFC5246\]](#)). Since these ciphers are functioning as stream ciphers, the authenticate-then-encrypt method is secure, and as specified by [\[RFC7366\]](#), the server that selects the `CTR_OMAC` ciphers **MUST NOT** send an `encrypt_then_mac` extension to the client.

#### 4.1.2. CNT\_IMIT

In the `CNT_IMIT` 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 `seqnum` sequence number 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`, and `i` in  $\{0, \dots, \text{seqnum}\}$  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 mode based on Cipher Block Chaining Message Authentication Code (CBC-MAC) (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 and 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.

3. The fields of the GenericStreamCipher structure (see [Section 6.2.3.1](#) of [\[RFC5246\]](#)) for the TLSCiphertext.fragment value are defined by the ENCValue\_seqnum value:

```
TLSCiphertext.fragment.content =
ENCValue_seqnum[1..length_seqnum],

TLSCiphertext.fragment.MAC = ENCValue_seqnum[length_seqnum +
1..length_seqnum + mac_length],
```

where length\_seqnum is the TLSCompressed.length value of the record with the seqnum sequence number, and mac\_length is equal to 4.

Note that the CNT\_IMIT cipher suite uses the authenticate-then-encrypt method (see [Appendix F.4](#) of [\[RFC5246\]](#)). Since this cipher is functioning as a stream cipher, the authenticate-then-encrypt method is secure, and as specified by [\[RFC7366\]](#), the server that selects the CNT\_IMIT cipher **MUST NOT** send an encrypt\_then\_mac extension to the client.

## 4.2. Key Exchange and Authentication

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

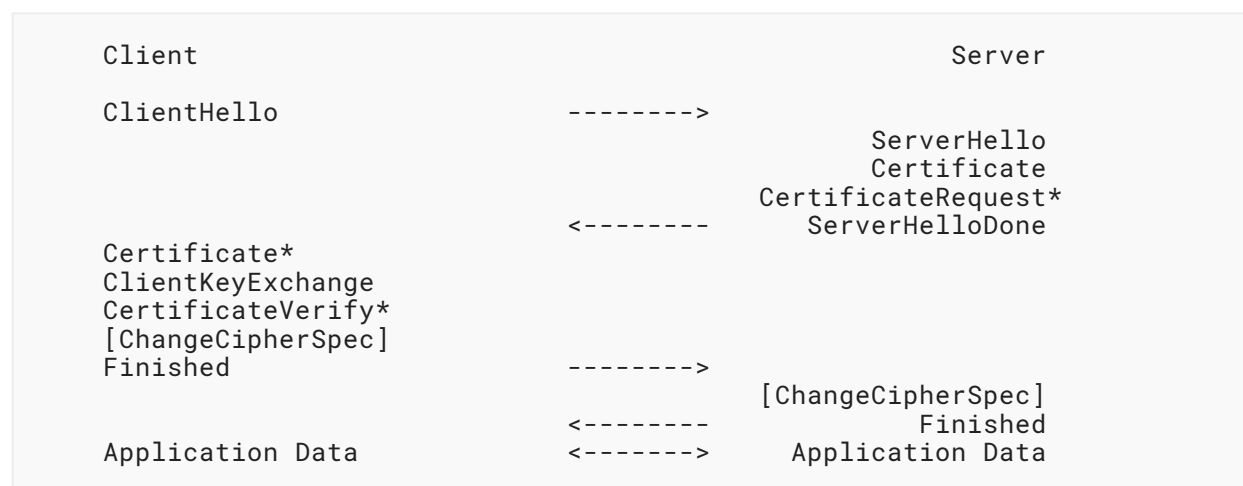


Figure 1: Message Flow for a Full Handshake

Notes for [Figure 1](#):

1. "\*" indicates optional messages that are sent for the client authentication.
2. 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 the 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 the 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 Sections 4.2.4.1 and 4.2.4.2) for the particular key export algorithm (see Sections 8.2.1 and 8.2.2) to generate the export representation PSExp of the PS value.
4. The client sends its ephemeral public key  $Q_{eph}$  and PSExp value in the ClientKeyExchange message.
5. Using its private key  $d_s$ , the server generates the import key material (see Sections 4.2.4.1 and 4.2.4.2) for the particular key import algorithm (see Sections 8.2.1 and 8.2.2) to extract the premaster secret PS from the export representation PSExp.

This section specifies the data structures and computations used by the profile of TLS 1.2 with GOST algorithms. The specifications for the ClientHello, ServerHello, Server Certificate, CertificateRequest, ClientKeyExchange, CertificateVerify, and Finished handshake messages are described in further detail below.

#### 4.2.1. Hello Messages

The ClientHello message is generated in accordance with [Section 7.4.1.2](#) 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 signature\_algorithms extension in the ClientHello message does not contain 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 7.4.1.3](#) 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 generated in accordance with [Section 7.4.4](#) of [RFC5246].

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 {8, 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.

```
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 structure corresponds to the CNT\_IMIT cipher suite and is described in [Section 4.2.4.2](#).

The DER encoding rules are used to encode the GostKeyTransport and the TLSGostKeyTransportBlob structures.

#### 4.2.4.1. CTR\_OMAC

In 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 the premaster secret PS, where PS is chosen from B\_32 at random.
3. 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_eph, Q_s, H).
```

4. 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).
```

5. 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 is as follows.

1. The following three conditions are checked. If any of these checks fail, 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. The export keys (`K_EXP_MAC` and `K_EXP_ENC`) are generated 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. The premaster secret `PS` is extracted 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. CNT\_IMIT

In the `CNT_IMIT` 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. The ephemeral key pair (`Q_eph`, `d_eph`) is generated, where:

```
d_eph is chosen from {1, ... , q_s - 1} at random;
Q_eph = d_eph * P_s.
```

2. The premaster secret PS is generated, where PS is chosen from B\_32 at random.
3. The export key (K\_EXP) is generated using the KEG\_28147 algorithm defined in [Section 8.3.2](#):

```
H = HASH(r_c | r_s);
K_EXP = KEG_28147(d_eph, Q_s, H).
```

4. An export representation PSEXP of the premaster secret PS using the KExp28147 algorithm defined in [Section 8.2.2](#) is generated:

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

5. The ClientKeyExchange message is generated using the TLSGostKeyTransportBlob structure that is defined as follows:

```
TLSGostKeyTransportBlob ::= 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, and 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 the GostR3410-TransportParameters.ukm field contains the initialization vector (IV) value.

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

Upon receiving the `ClientKeyExchange` message, the server process is as follows.

1. The following three conditions are checked. 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. The export key (`K_EXP`) is generated using the `KEG_28147` algorithm defined in [Section 8.3.2](#):

```
H = HASH(r_c | r_s);
K_EXP = KEG_28147(d_s, Q_eph, H).
```

3. The premaster secret `PS` is extracted from the export representation `PSExp` using the `KImp28147` algorithm defined in [Section 8.2.2](#):

```
PS = KImp28147(PSExp, K_EXP, H[1..8]).
```

#### 4.2.5. CertificateVerify

The client generates the value `sgn` as follows:

```
sgn = SIGN_{d_c}(handshake_messages) = str_l(r) | str_l(s)
```

where `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 the `gostr34102012_256` value of the `SignatureAndHashAlgorithm` field, and `l = 64` for the `gostr34102012_512` value of the `SignatureAndHashAlgorithm` field.

Here, "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:

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

where the `SignatureAndHashAlgorithm` structure is specified in [Section 5](#), and the `CertificateVerify.signature` field contains the `sgn` value.

#### 4.2.6. Finished

The TLS Finished message is generated in accordance with [Section 7.4.9](#) of [\[RFC5246\]](#).

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 pseudorandom function (PRF) 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 [\[RFC8891\]](#) 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 One-Key MAC (OMAC) construction defined in [\[GOST3413-2015\]](#), which can be considered as the CMAC mode defined in [\[CMAC\]](#) where the Kuznyechik or Magma block cipher (see [Section 4.3.1](#)) is used instead of the 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 MAC function `gostIMIT28147` defined in [Section 8.4](#) with the initialization vector `IV = IV0`, where `IV0` in `B_8` 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 the CTR-ACPKM encryption mode defined in [\[RFC8645\]](#) as the ENC function. The section size `N` is 4 KB for the `TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC` cipher suite and 1 KB for the `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.



Note that the counter modes used in cipher suites described in this document act as stream ciphers.

#### 4.3.4. PRF and HASH Algorithms

The 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 a 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:

Cipher Suites	SNMAX
TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC TLS_GOSTR341112_256_WITH_28147_CNT_IMIT	SNMAX = $2^{64} - 1$
TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC	SNMAX = $2^{32} - 1$

Table 1

## 5. New Values for the TLS 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]).

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

```
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 a 32-byte (256-bit) and 64-byte (512-bit) key length, respectively.

According to [RFC7091], the GOST R 34.10-2012 signature algorithm with a 32-byte (256-bit) or 64-byte (512-bit) key length uses the GOST R 34.11-2012 [RFC6986] hash algorithm with a 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]).

So, to represent gostr34102012\_256 and gostr34102012\_512 in the signature\_algorithms extension, the value shall be (8,64) and (8,65), respectively.

## 6. New Values for the TLS 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 "TLS Supported Groups" registry:

```
enum {
    GC256A(34), GC256B(35), GC256C(36), GC256D(37),
    GC512A(38), GC512B(39), GC512C(40),
} NamedGroup;
```

where the values correspond to the following curves:

Description	Curve Identifier Value	Reference
GC256A	id-tc26-gost-3410-2012-256-paramSetA	[RFC7836]
GC256B	id-GostR3410-2001-CryptoPro-A-ParamSet	[RFC4357]
GC256C	id-GostR3410-2001-CryptoPro-B-ParamSet	[RFC4357]
GC256D	id-GostR3410-2001-CryptoPro-C-ParamSet	[RFC4357]
GC512A	id-tc26-gost-3410-12-512-paramSetA	[RFC7836]
GC512B	id-tc26-gost-3410-12-512-paramSetB	[RFC7836]
GC512C	id-tc26-gost-3410-2012-512-paramSetC	[RFC7836]

Table 2

## 7. New Values for the TLS ClientCertificateType Identifiers Registry

The ClientCertificateType field of the CertificateRequest message contains a list 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 "TLS 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 34.10-2012-capable public key that corresponds to the 32-byte (256-bit) or 64-byte (512-bit) signature key, respectively.

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

The cipher suites specified in this document rely on some additional algorithms, specified below; the use of these algorithms is not confined to the use in TLS specified in this document.

### 8.1. TLSTREE

The TLSTREE function is defined as follows:

```
TLSTREE(K_root, i) = KDF_3(KDF_2(KDF_1(K_root, STR_8(i & C_1)),
STR_8(i & C_2)), STR_8(i & C_3)),
```

where

- $K\_root$  in  $B_{32}$ ;
- $i$  in  $\{0, 1, \dots, 2^{64} - 1\}$ ;
- $C_1, C_2, C_3$  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_GOSTR3411_2012_256(K, "level1", D);
KDF_2(K, D) = KDF_GOSTR3411_2012_256(K, "level2", D);
KDF_3(K, D) = KDF_GOSTR3411_2012_256(K, "level3", D).
```

#### 8.1.1. Key Tree Parameters

The `CTR_OMAC` cipher suites use the TLSTREE function for the rekeying approach. The constants for it are defined as in the table below.

Cipher Suites	C_1, C_2, C_3
TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC	C_1=0xFFFFFFFF00000000 C_2=0xFFFFFFFF80000 C_3=0xFFFFFFFFFC0
TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC	C_1=0xFFFFFC000000000 C_2=0xFFFFFE000000 C_3=0xFFFFF000

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:

```

+-----+
| KExp15(S, K_Exp_MAC, K_Exp_ENC, IV) |
+-----+
| 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_{L(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] and 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_{L(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] and 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 a key with the KImp15 algorithm, the IV value may be sent with the export key representation.

### 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 and the ECB-Encrypt(K, S) function denotes the encryption of message S on key K with the block cipher GOST 28147-89 in the electronic codebook (ECB) mode (see [RFC5830]).

The KImp28147 key import algorithm is defined as follows:

```

+-----+
| KImp28147(SExp, K, IV)
+-----+
| Input:
| - export representation SExp in B_44,
| - key K in B_32,
| - IV in B_8.
| Output:
| - imported secret S in B_32 or FAIL
+-----+
| 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](#) and 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  $\{1, \dots, q-1\}$ .

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:

```

+-----+
| KEG_512(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. return VKO_512(d, Q, UKM)
+-----+

```

where VKO\_512 is the VKO\_GOSTR3410\_2012\_512 function defined in [RFC7836].

### 8.3.2. KEG\_28147 Algorithm

The KEG\_28147 algorithm is defined as follows:

```

+-----+
| KEG_28147(d, Q, H)
+-----+
| Input:
| - private key d,
| - public key Q,
| - H in B_32.
| Output:
| - key material K in B_32.
+-----+
| 1. If  $q * Q$  is not equal to zero point
|     return FAIL
| 2. UKM = H[1..8]
| 3. R = VKO_256(d, Q, int(UKM))
| 4. return K = CPDivers(UKM, R)
+-----+

```

where the VKO\_256 function is equal to the VKO\_GOSTR3410\_2012\_256 function defined in [RFC7836] and the CPDivers function corresponds to the CryptoPro KEK Diversification Algorithm defined in [RFC4357], which takes as input the User Keying Material (UKM) value and the key value.

#### 8.4. gostIMIT28147

gost28147IMIT(IV, K, M) is a MAC algorithm with a 4-byte 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, L(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,  $m$  is the smallest, non-negative solution to the equation  $(L(M) + m) \bmod 8 = 0$ , and the MAC28147 function corresponds to the MAC generation mode defined in [RFC5830] with a 4-byte length output.

## 9. IANA Considerations

IANA has added the following values to the "TLS Cipher Suites" registry:



Value	Description	DTLS-OK	Recommended	Reference
0xC1,0x00	TLS_GOSTR341112_256_WITH_KUZYNECHIK_CTR_OMAC	N	N	RFC 9189
0xC1,0x01	TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC	N	N	RFC 9189
0xC1,0x02	TLS_GOSTR341112_256_WITH_28147_CNT_IMIT	N	N	RFC 9189

Table 4

IANA has added the following values to the "TLS SignatureAlgorithm" registry:

Value	Description	DTLS-OK	Reference
64	gostr34102012_256	Y	RFC 9189
65	gostr34102012_512	Y	RFC 9189

Table 5

IANA has added the following values to the "TLS SignatureScheme" registry:

Value	Description	Recommended	Reference
0x0840	Reserved for backward compatibility	N	RFC 9189
0x0841	Reserved for backward compatibility	N	RFC 9189

Table 6

IANA has also added the following footnote to values 64 and 65 in the "TLS SignatureAlgorithm" registry:

These values were allocated from the Reserved state due to a misunderstanding of the difference between Reserved and Unallocated that went undetected for a long time. Additional allocations from the Reserved state are not expected, and the TLS SignatureScheme registry is suitable for use for new allocations instead of this registry.

IANA has added the following values to the "TLS Supported Groups" registry:

Value	Description	DTLS-OK	Recommended	Reference
34	GC256A	Y	N	RFC 9189

Value	Description	DTLS-OK	Recommended	Reference
35	GC256B	Y	N	RFC 9189
36	GC256C	Y	N	RFC 9189
37	GC256D	Y	N	RFC 9189
38	GC512A	Y	N	RFC 9189
39	GC512B	Y	N	RFC 9189
40	GC512C	Y	N	RFC 9189

Table 7

IANA has added the following values to the "TLS ClientCertificateType Identifiers" registry:

Value	Description	DTLS-OK	Reference
67	gost_sign256	Y	RFC 9189
68	gost_sign512	Y	RFC 9189

Table 8

## 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 {0xFF, 0x85} 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 (see [\[RFC4357\]](#) and [\[R-1323565.1.024-2019\]](#)).

Description	Curve Identifier Values
GC256B	id-GostR3410_2001-CryptoPro-XchA-ParamSet id-tc26-gost-3410-2012-256-paramSetB

Description	Curve Identifier Values
GC256C	id-tc26-gost-3410-2012-256-paramSetC
GC256D	id-GostR3410-2001-CryptoPro-XchB-ParamSet id-tc26-gost-3410-2012-256-paramSetD

Table 9

Description	Curve Identifier Values
GC256B	id-GostR3410_2001-CryptoPro-XchA-ParamSet id-tc26-gost-3410-2012-256-paramSetB
GC256C	id-tc26-gost-3410-2012-256-paramSetC
GC256D	id-GostR3410-2001-CryptoPro-XchB-ParamSet id-tc26-gost-3410-2012-256-paramSetD

Table 10

The client should be prepared to handle any of these correctly if the corresponding group is included in the supported\_groups extension (see [RFC8422] and [RFC7919]).

## 11. Security Considerations

The cipher suites defined in this document do not provide Perfect Forward Secrecy.

The authenticate-then-encrypt method is crucial for the CNT\_IMIT cipher suite. Encryption of the MAC value is conducted to reduce the possibility of forgery to guessing. Here, the probability of a guess is approximately equal to  $2^{\{-32\}}$ , which is acceptable in some practical cases.

## 12. References

### 12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4357] Popov, V., Kurepkin, I., and S. Leontiev, "Additional Cryptographic Algorithms for Use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms", RFC 4357, DOI 10.17487/RFC4357, January 2006, <<https://www.rfc-editor.org/info/rfc4357>>.

- 
- [RFC4490] Leontiev, S., Ed. and G. Chudov, Ed., "Using the GOST 28147-89, GOST R 34.11-94, GOST R 34.10-94, and GOST R 34.10-2001 Algorithms with Cryptographic Message Syntax (CMS)", RFC 4490, DOI 10.17487/RFC4490, May 2006, <<https://www.rfc-editor.org/info/rfc4490>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, DOI 10.17487/RFC5746, February 2010, <<https://www.rfc-editor.org/info/rfc5746>>.
- [RFC5830] Dolmatov, V., Ed., "GOST 28147-89: Encryption, Decryption, and Message Authentication Code (MAC) Algorithms", RFC 5830, DOI 10.17487/RFC5830, March 2010, <<https://www.rfc-editor.org/info/rfc5830>>.
- [RFC6986] Dolmatov, V., Ed. and A. Degtyarev, "GOST R 34.11-2012: Hash Function", RFC 6986, DOI 10.17487/RFC6986, August 2013, <<https://www.rfc-editor.org/info/rfc6986>>.
- [RFC7091] Dolmatov, V., Ed. and A. Degtyarev, "GOST R 34.10-2012: Digital Signature Algorithm", RFC 7091, DOI 10.17487/RFC7091, December 2013, <<https://www.rfc-editor.org/info/rfc7091>>.
- [RFC7366] Gutmann, P., "Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7366, DOI 10.17487/RFC7366, September 2014, <<https://www.rfc-editor.org/info/rfc7366>>.
- [RFC7627] Bhargavan, K., Ed., Delignat-Lavaud, A., Pironti, A., Langley, A., and M. Ray, "Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension", RFC 7627, DOI 10.17487/RFC7627, September 2015, <<https://www.rfc-editor.org/info/rfc7627>>.
- [RFC7801] Dolmatov, V., Ed., "GOST R 34.12-2015: Block Cipher "Kuznyechik"", RFC 7801, DOI 10.17487/RFC7801, March 2016, <<https://www.rfc-editor.org/info/rfc7801>>.
- [RFC7836] Smyshlyaev, S., Ed., Alekseev, E., Oshkin, I., Popov, V., Leontiev, S., Podobaev, V., and D. Belyavsky, "Guidelines on the Cryptographic Algorithms to Accompany the Usage of Standards GOST R 34.10-2012 and GOST R 34.11-2012", RFC 7836, DOI 10.17487/RFC7836, March 2016, <<https://www.rfc-editor.org/info/rfc7836>>.
- [RFC7919] Gillmor, D., "Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)", RFC 7919, DOI 10.17487/RFC7919, August 2016, <<https://www.rfc-editor.org/info/rfc7919>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

- [RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <<https://www.rfc-editor.org/info/rfc8422>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8645] Smyshlyaev, S., Ed., "Re-keying Mechanisms for Symmetric Keys", RFC 8645, DOI 10.17487/RFC8645, August 2019, <<https://www.rfc-editor.org/info/rfc8645>>.
- [RFC8891] Dolmatov, V., Ed. and D. Baryshkov, "GOST R 34.12-2015: Block Cipher "Magma"", RFC 8891, DOI 10.17487/RFC8891, September 2020, <<https://www.rfc-editor.org/info/rfc8891>>.

## 12.2. Informative References

- [CMAC] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication", NIST Special Publication 800-38B, DOI 10.6028/NIST.SP.800-38B, October 2016, <<https://www.nist.gov/publications/recommendation-block-cipher-modes-operation-cmac-mode-authentication-0>>.
- [DraftGostTLS13] Smyshlyaev, S., Alekseev, E., Griboedova, E., Babueva, A., and L. Nikiforova, "GOST Cipher Suites for Transport Layer Security (TLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-smyshlyaev-tls13-gost-suites-05, 10 December 2021, <<https://datatracker.ietf.org/doc/html/draft-smyshlyaev-tls13-gost-suites-05>>.
- [GOST3413-2015] Federal Agency on Technical Regulating and Metrology, "Information technology. Cryptographic data security. Modes of operation for block ciphers", GOST R 34.13-2015, 2015.
- [IK2003] Iwata, T. and K. Kurosawa, "OMAC: One-Key CBC MAC", FSE 2003, Lecture Notes in Computer Science, Vol. 2887, DOI 10.1007/978-3-540-39887-5\_11, 2003, <[https://doi.org/10.1007/978-3-540-39887-5\\_11](https://doi.org/10.1007/978-3-540-39887-5_11)>.
- [MODES] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", NIST Special Publication 800-38A, DOI 10.6028/NIST.SP.800-38A, December 2001, <<https://csrc.nist.gov/publications/detail/sp/800-38a/final>>.
- [R-1323565.1.024-2019] Federal Agency on Technical Regulating and Metrology, "Information technology. Cryptographic data security. Elliptic curve parameters for the cryptographic algorithms and protocols", R 1323565.1.024-2019, January 2019.

## 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 Cipher Suite**

```
    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 BA 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 Cipher Suite**

```

    TLS_GOSTR341112_256_WITH_KUZNYECHIK_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 = 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 66 3E 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 D8 6A 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 Cipher Suite**



```

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

- 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

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
. . .
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

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 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: 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 0F FF

```

## TLSCiphertext:

```

00000: 17 03 03 04 08 B7 11 43 8B 16 20 1F 3C 49 33 95
00010: 21 C9 C8 CA 75 66 D4 C2 0F D3 3E 58 1F 80 07 DC
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 70 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
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
00800: 00 00 00 00 00

```

## K\_MAC\_4096:

```

00000: FB 30 EE 53 CF CF 89 D7 48 FC 0C 72 EF 16 0B 8B
00010: 53 CB BB FD 03 12 82 B0 26 21 4A B2 E0 77 58 FF

```

## MAC value:

```

00000: 50 55 A2 6A BE 19 63 81

```

## K\_ENC\_4096:

```
00000: ED F2 FD 02 47 71 60 23 83 09 00 2D 1D 57 DF 9F
00010: D2 ED 18 D6 45 66 C7 6F 4B F0 3D 3A BF 7B BB 1E

IV_4096:
00000: 00 00 10 00

TLSCiphertext:
00000: 17 03 03 08 08 99 95 26 07 03 47 1D ED A2 E6 55
00010: B6 B3 93 83 5E 33 8B 1E D0 0E DD 22 47 A2 FB 88
00020: FB B7 A8 94 80 62 08 8A F3 2C AE B6 AA 2C 4F 2A
. . .
007D0: 7F 0B 24 61 E7 5F E1 06 34 B8 4D C5 70 35 72 5A
007E0: CA 4F 0C BC A9 B0 6C B9 F7 6F BD 2F 80 46 2B 8D
007F0: 77 5E BD 41 6F 63 41 39 AC 89 C2 ED 3D F1 9F E2
00800: 4E F8 C0 5A A8 90 93 1B 01 86 FD 7D DF
```

#### A.1.2.2. TLS\_GOSTR341112\_256\_WITH\_KUZNYECHIK\_CTR\_OMAC Cipher Suite

```
TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC
```

```
*****
```

```
It is assumed that the following keys were established
during handshake:
```

```
- 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
```

```
-----
seqnum = 0
```

```
Application data:
```

```
00000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
TLSPplaintext:
```

```
00000: 17 03 03 00 0F 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 46 B3 63 57 71 13 F0 FB 56
```

```
IV_0:
```

```
00000: 00 00 00 00 00 00 00 00
```

```
TLSCiphertext:
```

```
00000: 17 03 03 00 1F 4D 1A 30 52 36 57 3B 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
```

```
-----
seqnum = 63
```

```
Application data:
```

```
00000: 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
```

```
00020: 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
```

```
00FE0: 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
```

```
TLSPplaintext:
```

```
00000: 17 03 03 10 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
00020:  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
00FE0:  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
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 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 78 F4 D1 96

-----
seqnum = 64

Application data:
00000:  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
00020:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.
.
.
01FD0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01FE0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01FF0:  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
00010:  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
.
.
.
01FD0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01FE0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01FF0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02000:  00 00 00 00 00

K_MAC_64:
00000:  AE BE 1E F4 18 71 3B F0 44 B9 FC D9 E5 72 D4 37
00010:  FB 38 B5 D8 29 56 7A 6F 79 18 39 6D 9F 4E 09 6B

```

```
MAC value:
00000:  EA C3 97 87 84 2B 1D BD 60 80 CC 3F BF AE 5C 2F

K_ENC_64:
00000:  64 F5 5A FC 37 A1 74 D9 53 3E 70 8B CD 14 FA 4A
00010:  EE C3 7B C0 E3 2B A4 99 01 B4 66 9E 96 A6 3D 96

IV_64:
00000:  00 00 00 00 00 00 00 40

TLSCiphertext:
00000:  17 03 03 20 10 E6 66 BB 98 AC 5B 0F 39 31 D8 55
00010:  1B 93 36 85 96 EE F0 EB A8 26 9C B8 BD AA E7 EB
00020:  80 C8 30 D7 5A B7 D4 6C 25 06 DC 8B 83 E1 F2 D3
. . .
01FE0:  B3 02 67 2C CB 02 86 CD 40 48 FB D5 38 1A 65 55
01FF0:  26 11 25 51 01 4F A8 ED F5 C2 1B 7D 1D B3 9D 6B
02000:  AD EC 0D 7C 07 05 34 8B 5C 55 6C 4D 50 81 69 1A
02010:  A9 EC 36 F8 B5
```

### 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 Cipher Suite**



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

```
Server public key Q_s:
x = 0x6531D4A72E655BFC9DFB94293B260702
   82FABF10D5C49B7366148C60E0BF8167

y = 0x37F8CC71DC5D917FC4A66F7826E72750
   8270B4FFC266C26CD4363E77B553A5B8
```

```
Server private key d_s:
0x5F308355DFD6A8ACAEE0837B100A3B1F
6D63FB29B78EF27D3967757F0527144C
```

```
-----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
00020: C3 1B C3 A3 45 61 65 88 96 84 CA 00 00 04 C1 00
00030: C1 01 01 00 00 13 00 0D 00 06 00 04 08 40 08 41
00040: 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
00030:  15 50 EC 78 D6 ED 51 AC 24 39 D7 E7 C1 01 00 00
00040:  09 FF 01 00 01 00 00 17 00 00

-----Server-----

Certificate message:
msg_type:      0B
length:        0001DB
body:
  certificate_list:
    length:     0001D8
    vector:
      ASN.1Cert:
        length:  0001D5
        vector:  308201D13082017EA003020102020833
                 FBB2C0E9575A46300A06082A85030701
                 010302301F311D301B06035504030C14
                 . . .
                 797990E4B5452CF82FE1F19EE237B754

```

```
CBCD5078D752A28013DFFC8224AD114B
BD7C1BB71E480AD6EEF9857A8C99C595
9053EEDFE9
```

```
00000: 0B 00 01 DB 00 01 D8 00 01 D5 30 82 01 D1 30 82
00010: 01 7E A0 03 02 01 02 02 08 33 FB B2 C0 E9 57 5A
00020: 46 30 0A 06 08 2A 85 03 07 01 01 03 02 30 1F 31
00030: 1D 30 1B 06 03 55 04 03 0C 14 74 65 73 74 5F 73
00040: 65 6C 66 73 69 67 6E 65 64 5F 63 65 72 74 30 1E
00050: 17 0D 31 39 30 36 32 37 31 35 32 34 30 38 5A 17
00060: 0D 32 30 31 32 31 38 31 35 33 34 30 38 5A 30 1F
00070: 31 1D 30 1B 06 03 55 04 03 0C 14 74 65 73 74 5F
00080: 73 65 6C 66 73 69 67 6E 65 64 5F 63 65 72 74 30
00090: 66 30 1F 06 08 2A 85 03 07 01 01 01 01 30 13 06
000A0: 07 2A 85 03 02 02 23 01 06 08 2A 85 03 07 01 01
000B0: 02 02 03 43 00 04 40 67 81 BF E0 60 8C 14 66 73
000C0: 9B C4 D5 10 BF FA 82 02 07 26 3B 29 94 FB 9D FC
000D0: 5B 65 2E A7 D4 31 65 B8 A5 53 B5 77 3E 36 D4 6C
000E0: C2 66 C2 FF B4 70 82 50 27 E7 26 78 6F A6 C4 7F
000F0: 91 5D DC 71 CC F8 37 A3 81 96 30 81 93 30 1D 06
00100: 03 55 1D 0E 04 16 04 14 E7 D0 0B B8 4D 8D 24 18
00110: 29 3E 05 C1 7C E7 77 98 D4 8D 30 16 30 0E 06 03
00120: 55 1D 0F 01 01 FF 04 04 03 02 01 C6 30 12 06 03
00130: 55 1D 13 01 01 FF 04 08 30 06 01 01 FF 02 01 01
00140: 30 4E 06 03 55 1D 23 04 47 30 45 80 14 E7 D0 0B
00150: B8 4D 8D 24 18 29 3E 05 C1 7C E7 77 98 D4 8D 30
00160: 16 A1 23 A4 21 30 1F 31 1D 30 1B 06 03 55 04 03
00170: 0C 14 74 65 73 74 5F 73 65 6C 66 73 69 67 6E 65
00180: 64 5F 63 65 72 74 82 08 33 FB B2 C0 E9 57 5A 46
00190: 30 0A 06 08 2A 85 03 07 01 01 03 02 03 41 00 E2
001A0: 88 44 F9 F1 C8 55 E2 DB 5B 19 79 90 E4 B5 45
001B0: 2C F8 2F E1 F1 9E E2 37 B7 54 CB CD 50 78 D7 52
001C0: A2 80 13 DF FC 82 24 AD 11 4B BD 7C 1B B7 1E 48
001D0: 0A D6 EE F9 85 7A 8C 99 C5 95 90 53 EE DF E9
```

Record layer message:

```
type: 16
version:
  major: 03
  minor: 03
length: 01DF
fragment: 0B0001DB0001D80001D5308201D13082
017EA003020102020833FBB2C0E9575A
46300A06082A85030701010302301F31
.
.
.
8844F9F1C855E2DB5B19797990E4B545
2CF82FE1F19EE237B754CBCD5078D752
A28013DFFC8224AD114BBD7C1BB71E48
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 1E 17 0D 31 39 30 36 32 37 31 35 32
00060: 34 30 38 5A 17 0D 32 30 31 32 31 38 31 35 33 34
00070: 30 38 5A 30 1F 31 1D 30 1B 06 03 55 04 03 0C 14
```

```

00080: 74 65 73 74 5F 73 65 6C 66 73 69 67 6E 65 64 5F
00090: 63 65 72 74 30 66 30 1F 06 08 2A 85 03 07 01 01
000A0: 01 01 30 13 06 07 2A 85 03 02 02 23 01 06 08 2A
000B0: 85 03 07 01 01 02 02 03 43 00 04 40 67 81 BF E0
000C0: 60 8C 14 66 73 9B C4 D5 10 BF FA 82 02 07 26 3B
000D0: 29 94 FB 9D FC 5B 65 2E A7 D4 31 65 B8 A5 53 B5
000E0: 77 3E 36 D4 6C C2 66 C2 FF B4 70 82 50 27 E7 26
000F0: 78 6F A6 C4 7F 91 5D DC 71 CC F8 37 A3 81 96 30
00100: 81 93 30 1D 06 03 55 1D 0E 04 16 04 14 E7 D0 0B
00110: B8 4D 8D 24 18 29 3E 05 C1 7C E7 77 98 D4 8D 30
00120: 16 30 0E 06 03 55 1D 0F 01 01 FF 04 04 03 02 01
00130: C6 30 12 06 03 55 1D 13 01 01 FF 04 08 30 06 01
00140: 01 FF 02 01 01 30 4E 06 03 55 1D 23 04 47 30 45
00150: 80 14 E7 D0 0B B8 4D 8D 24 18 29 3E 05 C1 7C E7
00160: 77 98 D4 8D 30 16 A1 23 A4 21 30 1F 31 1D 30 1B
00170: 06 03 55 04 03 0C 14 74 65 73 74 5F 73 65 6C 66
00180: 73 69 67 6E 65 64 5F 63 65 72 74 82 08 33 FB B2
00190: C0 E9 57 5A 46 30 0A 06 08 2A 85 03 07 01 01 03
001A0: 02 03 41 00 E2 88 44 F9 F1 C8 55 E2 DB 5B 19 79
001B0: 79 90 E4 B5 45 2C F8 2F E1 F1 9E E2 37 B7 54 CB
001C0: CD 50 78 D7 52 A2 80 13 DF FC 82 24 AD 11 4B BD
001D0: 7C 1B B7 1E 48 0A D6 EE F9 85 7A 8C 99 C5 95 90
001E0: 53 EE DF E9

```

-----Server-----

ServerHelloDone message:

```

msg_type: 0E
length: 000000
body: --

```

```
00000: 0E 00 00 00
```

Record layer message::

```

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

```

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

-----Client-----

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

```

Random d\_eph value:

```

0xA5C77C7482373DE16CE4A6F73CCE7F78
471493FF2C0709B8B706C9E8A25E6C1E

```

Q\_eph ephemeral key:

```

x = 0xA8F36D63D262A203978F1B3B6795CDBB
F1AE7FB8EF7F47F1F18871C198E00793

```

```

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 3D 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:  3081920428D7F0F0422367867B25FA42
                  33A954F58BDE92E9C9BBFB8816C99F15
                  E6398722A0B2B7BFE8493E9A5C306630
                  . . .
                  EFB87FAEF1BBCD95673B1B8F9703A262
                  D2636DF3A887F8141FEAC25A17CCB596
                  0461ED16B0F8B1BE93594395A10E6485
                  446B5DCA34

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 01 30
00040:  13 06 07 2A 85 03 02 02 23 01 06 08 2A 85 03 07
00050:  01 01 02 02 03 43 00 04 40 93 07 E0 98 C1 71 88
00060:  F1 F1 47 7F EF B8 7F AE F1 BB CD 95 67 3B 1B 8F

```

```

00070:  97 03 A2 62 D2 63 6D F3 A8 87 F8 14 1F EA C2 5A
00080:  17 CC B5 96 04 61 ED 16 B0 F8 B1 BE 93 59 43 95
00090:  A1 0E 64 85 44 6B 5D CA 34

```

Record layer message:

```

type:          16
version:
  major:       03
  minor:       03
length:        0099
fragment:      100000953081920428D7F0F042236786
               7B25FA4233A954F58BDE92E9C9BBFB88
               16C99F15E6398722A0B2B7BFE8493E9A
               . . .
               F1F1477FEFB87FAEF1BBBCD95673B1B8F
               9703A262D2636DF3A887F8141FEAC25A
               17CCB5960461ED16B0F8B1BE93594395
               A10E6485446B5DCA34

```

```

00000:  16 03 03 00 99 10 00 00 95 30 81 92 04 28 D7 F0
00010:  F0 42 23 67 86 7B 25 FA 42 33 A9 54 F5 8B DE 92
00020:  E9 C9 BB FB 88 16 C9 9F 15 E6 39 87 22 A0 B2 B7
00030:  BF E8 49 3E 9A 5C 30 66 30 1F 06 08 2A 85 03 07
00040:  01 01 01 01 30 13 06 07 2A 85 03 02 02 23 01 06
00050:  08 2A 85 03 07 01 01 02 02 03 43 00 04 40 93 07
00060:  E0 98 C1 71 88 F1 F1 47 7F EF B8 7F AE F1 BB CD
00070:  95 67 3B 1B 8F 97 03 A2 62 D2 63 6D F3 A8 87 F8
00080:  14 1F EA C2 5A 17 CC B5 96 04 61 ED 16 B0 F8 B1
00090:  BE 93 59 43 95 A1 0E 64 85 44 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:  B2 B7 BF E8 49 3E 9A 5C

```

HASH(r<sub>c</sub> | r<sub>s</sub>):

```

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

```

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 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
```

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 10 32 55 9D 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-----

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
```

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 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
```



-----Client-----

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

-----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  
33D5F7B967EED155F7D65592892C03C7  
885C249B1225B184AB4D5DBF

00000: 16 03 03 00 2C 0C 63 02 71 D4 DA 39 DD 8D 6B D0

00010: 40 30 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

-----Server-----

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

-----Server-----

Finished message:

msg\_type: 14

length: 000020

body:

verify\_data: 4539EC8D0AF7B1A62041AB434A437771  
D34C4719D86EBBFD0F28C3E953550CD0

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  
 00010: 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  
           7F100C4B63ADBC75F49BCBF428572D37  
           7CAED097336DB203

00000: 17 03 03 00 28 38 80 7B 6E 5E 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  
 00010: 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 Cipher Suite**

```
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  
    92E6A30A288BF447CD9FF6C3133E9724  
    7A9706B267703C9B4E239F0D7C7E3310  
    C22D2752B35BD2E4FD39B8F11DEB833A
```

```
y = 0xF305E95B36502D4E60A1059FB20AB30B  
    FC7C95727F3A2C04B1DFDDB53B0413F2  
    99F2DFE66A5E1CCB4101A7A01D612BE6  
    BD78E1E3B3D567EBB16ABE587A11F4EA
```

```
Server private key d_s:  
0x12FD7A70067479A0F66C59F9A25534AD  
    FBC7ABFD3CC72D79806F8B402601644B  
    3005ED365A2D8989A8CCAE640D5FC08D  
    D27DFBBFE137CF528E1AC6D445192E01
```

```
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
00020: C3 1B C3 A3 45 61 65 88 96 84 CA 00 00 04 C1 00
00030: C1 01 01 00 00 13 00 0D 00 06 00 04 08 40 08 41
00040: 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: 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: --

```

```

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 00 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
          ED51AC2439D7E7C100000009FF010001
          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
00030: 15 50 EC 78 D6 ED 51 AC 24 39 D7 E7 C1 00 00 00

```



00040: 09 FF 01 00 01 00 00 17 00 00

-----Server-----

Certificate message:

msg\_type: 0B

length: 00024C

body:

certificate\_list:

length: 000249

vector:

ASN.1Cert:

length: 000246

vector: 30820242308201AEA003020102020101

300A06082A850307010103033042312C

302A06092A864886F70D010901161D74

. . .  
371AF83C5BC58B366DFEFA7345D50317

867C177AC84AC07EE8612164629AB7BD

C48AA0F64A741FE7298E82C5BFCE8672

029F875391F7

```

00000: 0B 00 02 4C 00 02 49 00 02 46 30 82 02 42 30 82
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 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 65 72 35 31 32 43 40 63 72
000B0: 79 70 74 6F 70 72 6F 2E 72 75 31 12 30 10 06 03
000C0: 55 04 03 13 09 53 65 72 76 65 72 35 31 32 30 81
000D0: AA 30 21 06 08 2A 85 03 07 01 01 01 02 30 15 06
000E0: 09 2A 85 03 07 01 02 01 02 03 06 08 2A 85 03 07
000F0: 01 01 02 03 03 81 84 00 04 81 80 3A 83 EB 1D F1
00100: B8 39 FD E4 D2 5B B3 52 27 2D C2 10 33 7E 7C 0D
00110: 9F 23 4E 9B 3C 70 67 B2 06 97 7A 24 97 3E 13 C3
00120: F6 9F CD 47 F4 8B 28 0A A3 E6 92 80 F5 3F 9B 66
00130: 63 65 C6 72 D9 9A 47 DA 89 45 F1 EA F4 11 7A 58
00140: BE 6A B1 EB 67 D5 B3 E3 E1 78 BD E6 2B 61 1D A0
00150: A7 01 41 CB 1C 5E 6A E6 DF F2 99 F2 13 04 3B B5
00160: DD DF B1 04 2C 3A 7F 72 95 7C FC 0B B3 0A B2 9F
00170: 05 A1 60 4E 2D 50 36 5B E9 05 F3 A3 43 30 41 30
00180: 1D 06 03 55 1D 0E 04 16 04 14 87 9C C6 5A 0F 4A
00190: 89 CB 4A 58 49 DF 05 61 56 9B AA DC 11 69 30 0B
001A0: 06 03 55 1D 0F 04 04 03 02 03 28 30 13 06 03 55
001B0: 1D 25 04 0C 30 0A 06 08 2B 06 01 05 05 07 03 01
001C0: 30 0A 06 08 2A 85 03 07 01 01 03 03 03 81 81 00
001D0: 35 BE 38 51 EC B6 E9 2D 32 40 01 81 0F 8C 89 03
001E0: 52 42 F4 05 46 9F 4C 4E CB 05 02 7C 57 E2 71 52
001F0: 12 AF D7 CD BB 0C ED 7A 8B 4D 33 42 CC 50 1A BD
00200: 99 99 75 A5 8A DE 0E 58 4F CA 35 F5 2E 45 58 B7
00210: 31 1D 49 D0 A0 51 32 79 F7 39 37 1A F8 3C 5B C5
00220: 8B 36 6D FE FA 73 45 D5 03 17 86 7C 17 7A C8 4A

```

```

00230:  C0 7E E8 61 21 64 62 9A B7 BD C4 8A A0 F6 4A 74
00240:  1F E7 29 8E 82 C5 BF CE 86 72 02 9F 87 53 91 F7

```

Record layer message:

```

type:          16
version:
  major:       03
  minor:       03
length:        0250
fragment:      0B00024C000249000246308202423082
               01AEA003020102020101300A06082A85
               0307010103033042312C302A06092A86

               8B366DFEFA7345D50317867C177AC84A
               C07EE8612164629AB7BDC48AA0F64A74
               1FE7298E82C5BFCE8672029F875391F7

```

```

00000:  16 03 03 02 50 0B 00 02 4C 00 02 49 00 02 46 30
00010:  82 02 42 30 82 01 AE A0 03 02 01 02 02 01 01 30
00020:  0A 06 08 2A 85 03 07 01 01 03 03 30 42 31 2C 30
00030:  2A 06 09 2A 86 48 86 F7 0D 01 09 01 16 1D 74 6C
00040:  73 31 32 5F 73 65 72 76 65 72 35 31 32 43 40 63
00050:  72 79 70 74 6F 70 72 6F 2E 72 75 31 12 30 10 06
00060:  03 55 04 03 13 09 53 65 72 76 65 72 35 31 32 30
00070:  1E 17 0D 31 37 30 35 32 35 30 39 32 35 31 38 5A
00080:  17 0D 33 30 30 35 30 31 30 39 32 35 31 38 5A 30
00090:  42 31 2C 30 2A 06 09 2A 86 48 86 F7 0D 01 09 01
000A0:  16 1D 74 6C 73 31 32 5F 73 65 72 76 65 72 35 31
000B0:  32 43 40 63 72 79 70 74 6F 70 72 6F 2E 72 75 31
000C0:  12 30 10 06 03 55 04 03 13 09 53 65 72 76 65 72
000D0:  35 31 32 30 81 AA 30 21 06 08 2A 85 03 07 01 01
000E0:  01 02 30 15 06 09 2A 85 03 07 01 02 01 02 03 06
000F0:  08 2A 85 03 07 01 01 02 03 03 81 84 00 04 81 80
00100:  3A 83 EB 1D F1 B8 39 FD E4 D2 5B B3 52 27 2D C2
00110:  10 33 7E 7C 0D 9F 23 4E 9B 3C 70 67 B2 06 97 7A
00120:  24 97 3E 13 C3 F6 9F CD 47 F4 8B 28 0A A3 E6 92
00130:  80 F5 3F 9B 66 63 65 C6 72 D9 9A 47 DA 89 45 F1
00140:  EA F4 11 7A 58 BE 6A B1 EB 67 D5 B3 E3 E1 78 BD
00150:  E6 2B 61 1D A0 A7 01 41 CB 1C 5E 6A E6 DF F2 99
00160:  F2 13 04 3B B5 DD DF B1 04 2C 3A 7F 72 95 7C FC
00170:  0B B3 0A B2 9F 05 A1 60 4E 2D 50 36 5B E9 05 F3
00180:  A3 43 30 41 30 1D 06 03 55 1D 0E 04 16 04 14 87
00190:  9C C6 5A 0F 4A 89 CB 4A 58 49 DF 05 61 56 9B AA
001A0:  DC 11 69 30 0B 06 03 55 1D 0F 04 04 03 02 03 28
001B0:  30 13 06 03 55 1D 25 04 0C 30 0A 06 08 2B 06 01
001C0:  05 05 07 03 01 30 0A 06 08 2A 85 03 07 01 01 03
001D0:  03 03 81 81 00 35 BE 38 51 EC B6 E9 2D 32 40 01
001E0:  81 0F 8C 89 03 52 42 F4 05 46 9F 4C 4E CB 05 02
001F0:  7C 57 E2 71 52 12 AF D7 CD BB 0C ED 7A 8B 4D 33
00200:  42 CC 50 1A BD 99 99 75 A5 8A DE 0E 58 4F CA 35
00210:  F5 2E 45 58 B7 31 1D 49 D0 A0 51 32 79 F7 39 37
00220:  1A F8 3C 5B C5 8B 36 6D FE FA 73 45 D5 03 17 86
00230:  7C 17 7A C8 4A C0 7E E8 61 21 64 62 9A B7 BD C4
00240:  8A A0 F6 4A 74 1F E7 29 8E 82 C5 BF CE 86 72 02
00250:  9F 87 53 91 F7

```

-----Server-----

```

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:         0D000000B0243440004084008410000

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:           000000
body:             --

00000:  0E 00 00 00

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

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:    308201E03082018DA003020102020101
                  300A06082A850307010103023053312E
                  302C06092A864886F70D010901161F74

                  C1CAB43AC01AFB0F3451BDC2DB188BBC
                  B77884251CDF6037BA830F4B31D5E96F
                  DC9BC1C95ABE658266C48402E070DE1F
                  292724E8

```

```

00000:  0B 00 01 EA 00 01 E7 00 01 E4 30 82 01 E0 30 82
00010:  01 8D A0 03 02 01 02 02 01 01 30 0A 06 08 2A 85
00020:  03 07 01 01 03 02 30 53 31 2E 30 2C 06 09 2A 86
00030:  48 86 F7 0D 01 09 01 16 1F 74 6C 73 31 32 5F 63
00040:  6C 69 65 6E 74 32 35 36 41 5F 45 40 63 72 79 70
00050:  74 6F 70 72 6F 2E 72 75 31 21 30 1F 06 03 55 04
00060:  03 1E 18 00 43 00 6C 00 69 00 65 00 6E 00 74 00
00070:  32 00 35 00 36 00 41 00 5F 00 45 30 1E 17 0D 31
00080:  37 30 35 32 35 30 39 33 31 31 38 5A 17 0D 33 30
00090:  30 35 30 31 30 39 33 31 31 38 5A 30 53 31 2E 30
000A0:  2C 06 09 2A 86 48 86 F7 0D 01 09 01 16 1F 74 6C
000B0:  73 31 32 5F 63 6C 69 65 6E 74 32 35 36 41 5F 45
000C0:  40 63 72 79 70 74 6F 70 72 6F 2E 72 75 31 21 30
000D0:  1F 06 03 55 04 03 1E 18 00 43 00 6C 00 69 00 65
000E0:  00 6E 00 74 00 32 00 35 00 36 00 41 00 5F 00 45
000F0:  30 68 30 21 06 08 2A 85 03 07 01 01 01 01 30 15
00100:  06 09 2A 85 03 07 01 02 01 01 01 06 08 2A 85 03
00110:  07 01 01 02 03 43 00 04 40 75 49 0B 7B F8 79
00120:  D8 1C C5 A1 BA EA 66 65 06 DF B5 D7 BF 25 60 67
00130:  78 B7 24 F3 15 9E 8A B1 5D 0F 86 F3 48 2C B2 AB
00140:  10 8D 5A C2 DE 4E FB 2B FA E0 39 39 94 C2 DE 87
00150:  F0 D3 42 F8 61 83 F1 BB E5 9E A3 43 30 41 30 1D
00160:  06 03 55 1D 0E 04 16 04 14 74 49 1E 77 30 D3 42
00170:  A6 28 0E 72 A1 13 9D D9 90 8B FA F1 03 30 0B 06
00180:  03 55 1D 0F 04 04 03 02 07 80 30 13 06 03 55 1D
00190:  25 04 0C 30 0A 06 08 2B 06 01 05 05 07 03 02 30
001A0:  0A 06 08 2A 85 03 07 01 01 03 02 03 41 00 1C 2D
001B0:  35 22 B4 11 02 D6 20 1F 23 50 C1 CA B4 3A C0 1A
001C0:  FB 0F 34 51 BD C2 DB 18 8B BC B7 78 84 25 1C DF
001D0:  60 37 BA 83 0F 4B 31 D5 E9 6F DC 9B C1 C9 5A BE
001E0:  65 82 66 C4 84 02 E0 70 DE 1F 29 27 24 E8

```

Record layer message:

```

type:              16
version:
  major:           03
  minor:           03

```

```

length:          01EE
fragment:       0B0001EA0001E70001E4308201E03082
                018DA003020102020101300A06082A85
                0307010103023053312E302C06092A86
                . . .
                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 30 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 69 65 6E 74 32 35 36 41 5F 45
00050:  40 63 72 79 70 74 6F 70 72 6F 2E 72 75 31 21 30
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 30 35 32 35 30 39 33 31 31 38
00090:  5A 17 0D 33 30 30 35 30 31 30 39 33 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 69 65 6E 74 32
000C0:  35 36 41 5F 45 40 63 72 79 70 74 6F 70 72 6F 2E
000D0:  72 75 31 21 30 1F 06 03 55 04 03 1E 18 00 43 00
000E0:  6C 00 69 00 65 00 6E 00 74 00 32 00 35 00 36 00
000F0:  41 00 5F 00 45 30 68 30 21 06 08 2A 85 03 07 01
00100:  01 01 01 30 15 06 09 2A 85 03 07 01 02 01 01 01
00110:  06 08 2A 85 03 07 01 01 02 02 03 43 00 04 40 75
00120:  49 0B 7B F8 79 D8 1C C5 A1 BA EA 66 65 06 DF B5
00130:  D7 BF 25 60 67 78 B7 24 F3 15 9E 8A B1 5D 0F 86
00140:  F3 48 2C B2 AB 10 8D 5A C2 DE 4E FB 2B FA E0 39
00150:  39 94 C2 DE 87 F0 D3 42 F8 61 83 F1 BB E5 9E A3
00160:  43 30 41 30 1D 06 03 55 1D 0E 04 16 04 14 74 49
00170:  1E 77 30 D3 42 A6 28 0E 72 A1 13 9D D9 90 8B FA
00180:  F1 03 30 0B 06 03 55 1D 0F 04 04 03 02 07 80 30
00190:  13 06 03 55 1D 25 04 0C 30 0A 06 08 2B 06 01 05
001A0:  05 07 03 02 30 0A 06 08 2A 85 03 07 01 01 03 02
001B0:  03 41 00 1C 2D 35 22 B4 11 02 D6 20 1F 23 50 C1
001C0:  CA B4 3A C0 1A FB 0F 34 51 BD C2 DB 18 8B BC B7
001D0:  78 84 25 1C DF 60 37 BA 83 0F 4B 31 D5 E9 6F DC
001E0:  9B C1 C9 5A BE 65 82 66 C4 84 02 E0 70 DE 1F 29
001F0:  27 24 E8

```

-----Client-----

PMS value:

```

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

```

Random d\_eph value:

```

0x150ACD11B66DD695AD18418FA7A2DC63
 6B7E29DCA24536AABC826EE3175BB1FA
 DC3AA0D01D3092E120B0FCF7EB872F4B
 7E26EA17849D689222A48CF95A6E4831

```

Q\_eph ephemeral key:

```

x = 0xC941BE5193189B476D5A0334114A3E04

```

```
BBE5B37C738AE40F150B334135288664
FEBFC5622818894A07B1F7AD60E28480
B4B637B90EA7D4BA980186B605D75BC6
```

```
y = 0xA154F7B93E8148652011F4FD52C9A06A
6471ADB28D0A949AE26BC786DE874153
ABC00B35164F3214A8A83C00ECE27831
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:

```
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E
```

Export key generation. UKM value:

```
0xC3EF0428D4B7A1F4C5025F2E65DD2B2E
```

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 01 02 30 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
               04D3F65418E1D380B4CB945F0A3DCA51
               500CF3A1BEF37F76C07341A9839CCF6C
               2356845293B03178E2EC003CA8A81432
               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 01 02 30 15 06
00050:  09 2A 85 03 07 01 02 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 0E 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 51 BE 41 C9 BE 9A 7E 2A FC
000B0:  24 62 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 = SIGN\_d\_c(HM):

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

-----Client-----

CertificateVerify message:

```
msg_type:          0F
length:           000044
body:
  algorithm:
    hash:          08
    signature:     40
  signature:
    length:        0040
    vector:        F71F4362455BC55BA89A8FAF018288EC
                  00B32717482E7624B257D9797C8FF602
                  E315FDBD8DE56D085418040E1B61BBF6
                  B301AC263D50038B303113DB3617503A
```

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

Record layer message:

```
type:             16
version:
  major:          03
  minor:          03
length:           0048
fragment:         0F00004408400040F71F4362455BC55B
                  A89A8FAF018288EC00B32717482E7624
                  B257D9797C8FF602E315FDBD8DE56D08
                  5418040E1B61BBF6B301AC263D50038B
```



303113DB3617503A

```

00000: 16 03 03 00 48 0F 00 00 44 08 40 00 40 F7 1F 43
00010: 62 45 5B C5 5B A8 9A 8F AF 01 82 88 EC 00 B3 27
00020: 17 48 2E 76 24 B2 57 D9 79 7C 8F F6 02 E3 15 FD
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 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
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

```

-----Client-----

```

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

```

-----Client-----

```

HASH(HM):
00000:  C9 A4 80 DA 29 6C DD 12 3E 9A EB 26 88 8B 86 19
00010:  EA 67 78 B7 23 FA A8 B2 DC 70 6A CB A5 AB AF 11

client_verify_data:
00000:  98 7C 13 E6 FA 16 F3 D5 10 AE 83 00 23 58 72 27
00010:  32 90 09 4C 8F C7 B5 F0 C7 D7 47 C4 27 35 F8 F1

```

-----Client-----

```

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

00000:  14 00 00 20 98 7C 13 E6 FA 16 F3 D5 10 AE 83 00
00010:  23 58 72 27 32 90 09 4C 8F C7 B5 F0 C7 D7 47 C4
00020:  27 35 F8 F1

```

```

Record layer message:
type:                16
version:
  major:             03
  minor:             03
length:              0034
fragment:            4DC53D655EDFD1843AF69ADBDE989C0B
                    1F0C0A1A0FD1B3F458029D8F9989F9BF9
                    6C5C42971063A9B70714F412E4F6280F
                    7C21601B

00000:  16 03 03 00 34 4D C5 3D 65 5E DF D1 84 3A F6 9A
00010:  DB DE 98 9C 0B 1F 0C 0A 1A 0F D1 B3 F4 58 02 9D
00020:  8F 99 89 FB F9 6C 5C 42 97 10 63 A9 B7 07 14 F4
00030:  12 E4 F6 28 0F 7C 21 60 1B

```

-----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:  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
                    3D1DAF30C9A853A9871077FDD5CA063B
                    2C81BCC9D59FA6E3F5FAD9B2599BB586
                    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  
 00010: 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  
 5C09C27233006D65CFF0B5BA87504969

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

-----Server-----

Application data:

00000: 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

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:
  level:          01
  description:    00
```

```
00000:  01 00
```

```
Record layer message:
type:            15
version:
  major:         03
  minor:         03
length:          0012
fragment:        EB62E5AB78BF2A4B678920A11027EC43
                 0C3F
```

```
00000:  15 03 03 00 12 EB 62 E5 AB 78 BF 2A 4B 67 89 20
00010:  A1 10 27 EC 43 0C 3F
```

## A.2. Test Examples for CNT\_IMIT Cipher Suites

### A.2.1. Record Examples

It is assumed that the following keys were established during handshake:

```
- MAC key:
00000:  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
- Encryption key:
00000:  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
- IV:
00000:  00 00 00 00 00 00 00 00
```

-----  
seqnum = 0

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

Plaintext:  
00000: 17 03 03 00 07 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

-----  
seqnum = 1

Application data:  
00000: 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  
...  
007f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Plaintext:  
00000: 17 03 03 08 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  
...  
007f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00804: 00 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  
...  
007f0: 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

### 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
    97F36A13D5433E990B2EAC859D2E9B7A
    E054794655389158B8242923E3841B14
    24FD89F221701C89D9A3BF6A9F946795

y = 0xD01E80DEC5BD23C8BC6B85F12BBB1635
    A5AE7AD50DE24FB8FD02CB285A4AE65A
    7D6FBB99AAFFDA80629826F2F7F73282
    220444761615A06D082077C4A00FD4CF
```

```
Server private key d_s:
0x5F1E83AFA2C4CB2C5633C51380E84E37
    4B013EE7C238330709080CE914B442D4
    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:
    0100003A03036A523D6880DCC2DC75CC
    C43CFD04B616F5C3757B8077B76A9B50
    4949FD3BFDB8000002C1020100000F00
    0D0006000408410840FF01000100

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
33CD7F2C90B15E76DCC30815C19F92A6
D100915AF2DB2012AAA5E5779014711C
CD6D265BDEE5191026431C83768EE5EB
5A157F940BE9FBC102000005FF010001
00

00000: 16 03 03 00 51 02 00 00 4D 03 03 FE 92 C9 51 6D
00010: 0E 1A 67 A0 4C 33 CD 7F 2C 90 B1 5E 76 DC C3 08
00020: 15 C1 9F 92 A6 D1 00 91 5A F2 DB 20 12 AA A5 E5
00030: 77 90 14 71 1C CD 6D 26 5B DE E5 19 10 26 43 1C
00040: 83 76 8E E5 EB 5A 15 7F 94 0B E9 FB C1 02 00 00
00050: 05 FF 01 00 01 00

```

-----Server-----

```

Certificate message:
msg_type: 0B
length: 000266
body:
  certificate_list:
    length: 000263
    vector:
      ASN.1Cert:
        length: 000260
        vector:
3082025C308201C8A003020102021478
94DC9D920977809191642F1DAEDC26BA
3B5104300A06082A8503070101030330

        . . .
6C12D51F99C98A4A9904F0EA5486FED7
FF66AB8EB2425E1ACEAE8A758BDF843B
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: 80 91 91 64 2F 1D AE DC 26 BA 3B 51 04 30 0A 06

```

```

00030: 08 2A 85 03 07 01 01 03 03 30 19 31 17 30 15 06
00040: 03 55 04 03 13 0E 43 41 20 43 65 72 74 69 66 69
00050: 63 61 74 65 30 1E 17 0D 31 38 30 31 30 32 30 30
00060: 30 30 31 31 5A 17 0D 32 32 30 31 30 32 30 30 30
00070: 30 32 31 5A 30 21 31 1F 30 1D 06 03 55 04 03 13
00080: 16 53 65 72 76 65 72 20 35 31 32 20 43 65 72 74
00090: 69 66 69 63 61 74 65 30 81 AA 30 21 06 08 2A 85
000a0: 03 07 01 01 01 02 30 15 06 09 2A 85 03 07 01 02
000b0: 01 02 01 06 08 2A 85 03 07 01 01 02 03 03 81 84
000c0: 00 04 81 80 95 67 94 9F 6A BF A3 D9 89 1C 70 21
000d0: F2 89 FD 24 14 1B 84 E3 23 29 24 B8 58 91 38 55
000e0: 46 79 54 E0 7A 9B 2E 9D 85 AC 2E 0B 99 3E 43 D5
000f0: 13 6A F3 97 6D 23 24 48 99 43 41 20 C8 8A 27 C0
00100: 66 05 DB 16 CF D4 0F A0 C4 77 20 08 6D A0 15 16
00110: 76 44 04 22 82 32 F7 F7 F2 26 98 62 80 DA FF AA
00120: 99 BB 6F 7D 5A E6 4A 5A 28 CB 02 FD B8 4F E2 0D
00130: D5 7A AE A5 35 16 BB 2B F1 85 6B BC C8 23 BD C5
00140: DE 80 1E D0 A3 81 93 30 81 90 30 0C 06 03 55 1D
00150: 13 01 01 FF 04 02 30 00 30 1A 06 03 55 1D 11 04
00160: 13 30 11 82 09 6C 6F 63 61 6C 68 6F 73 74 87 04
00170: 7F 00 00 01 30 13 06 03 55 1D 25 04 0C 30 0A 06
00180: 08 2B 06 01 05 05 07 03 01 30 0F 06 03 55 1D 0F
00190: 01 01 FF 04 05 03 03 07 B0 00 30 1D 06 03 55 1D
001a0: 0E 04 16 04 14 AE 46 41 1B FD B3 08 C3 39 03 47
001b0: 57 57 2B 0F BF A3 6F 9A 99 30 1F 06 03 55 1D 23
001c0: 04 18 30 16 80 14 7F 7B 7A 15 61 A6 F2 18 A2 E3
001d0: 48 3B C6 39 D9 7F 42 DB 6D AF 30 0A 06 08 2A 85
001e0: 03 07 01 01 03 03 03 81 81 00 9C 49 78 F7 1B AB
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
00220: EB F0 AD E9 36 07 0F B2 40 E5 0D 04 37 03 7F 2A
00230: EC 99 C7 CD 23 9F 6F 20 25 A8 6C 12 D5 1F 99 C9
00240: 8A 4A 99 04 F0 EA 54 86 FE D7 FF 66 AB 8E B2 42
00250: 5E 1A CE AE 8A 75 8B DF 84 3B E1 A8 F6 FE BF 67
00260: 30 15 FE D7 AB 86 53 3D BF 20

```

Record layer message:

```

type: 16
version:
  major: 03
  minor: 03
length: 026A
fragment: 0B0002660002630002603082025C3082
01C8A00302010202147894DC9D920977
809191642F1DAEDC26BA3B5104300A06
. . .
EC99C7CD239F6F2025A86C12D51F99C9
8A4A9904F0EA5486FED7FF66AB8EB242
5E1ACEAE8A758BDF843BE1A8F6FEBF67
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 30 32 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 76 65 72 20 35 31 32
00090: 20 43 65 72 74 69 66 69 63 61 74 65 30 81 AA 30
000a0: 21 06 08 2A 85 03 07 01 01 01 01 02 30 15 06 09 2A
000b0: 85 03 07 01 02 01 02 01 06 08 2A 85 03 07 01 01
000c0: 02 03 03 81 84 00 04 81 80 95 67 94 9F 6A BF A3
000d0: D9 89 1C 70 21 F2 89 FD 24 14 1B 84 E3 23 29 24
000e0: B8 58 91 38 55 46 79 54 E0 7A 9B 2E 9D 85 AC 2E
000f0: 0B 99 3E 43 D5 13 6A F3 97 6D 23 24 48 99 43 41
00100: 20 C8 8A 27 C0 66 05 DB 16 CF D4 0F A0 C4 77 20
00110: 08 6D A0 15 16 76 44 04 22 82 32 F7 F7 F2 26 98
00120: 62 80 DA FF AA 99 BB 6F 7D 5A E6 4A 5A 28 CB 02
00130: FD B8 4F E2 0D D5 7A AE A5 35 16 BB 2B F1 85 6B
00140: BC C8 23 BD C5 DE 80 1E D0 A3 81 93 30 81 90 30
00150: 0C 06 03 55 1D 13 01 01 FF 04 02 30 00 30 1A 06
00160: 03 55 1D 11 04 13 30 11 82 09 6C 6F 63 61 6C 68
00170: 6F 73 74 87 04 7F 00 00 01 30 13 06 03 55 1D 25
00180: 04 0C 30 0A 06 08 2B 06 01 05 05 07 03 01 30 0F
00190: 06 03 55 1D 0F 01 01 FF 04 05 03 03 07 B0 00 30
001a0: 1D 06 03 55 1D 0E 04 16 04 14 AE 46 41 1B FD B3
001b0: 08 C3 39 03 47 57 57 2B 0F BF A3 6F 9A 99 30 1F
001c0: 06 03 55 1D 23 04 18 30 16 80 14 7F 7B 7A 15 61
001d0: A6 F2 18 A2 E3 48 3B C6 39 D9 7F 42 DB 6D AF 30
001e0: 0A 06 08 2A 85 03 07 01 01 03 03 03 81 81 00 9C
001f0: 49 78 F7 1B AB 54 8A 25 6D 2A 18 7C A8 4D 72 4F
00200: E1 EF A7 E5 36 67 2E 79 1F 8A 0C B6 74 1E B1 63
00210: E2 96 37 8C 5B 82 83 EE DA B4 1B A4 22 1E BC E2
00220: 05 F6 F8 79 CF EB F0 AD E9 36 07 0F B2 40 E5 0D
00230: 04 37 03 7F 2A EC 99 C7 CD 23 9F 6F 20 25 A8 6C
00240: 12 D5 1F 99 C9 8A 4A 99 04 F0 EA 54 86 FE D7 FF
00250: 66 AB 8E B2 42 5E 1A CE AE 8A 75 8B DF 84 3B E1
00260: A8 F6 FE BF 67 30 15 FE D7 AB 86 53 3D BF 20

```

-----Server-----

ServerHelloDone message:

```

msg_type: 0E
length: 000000
body: --

```

```
00000: 0E 00 00 00
```

Record layer message::

```

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

```

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

-----Client-----

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
43C9AC27D42ACF1091CE7EF67E6C3CCA
0F6C879B2DA3C1607648BAEB96471BD2
078DF5CAA4FA83ECC0FFD6D3C8E5D56
```

Q\_eph ephemeral key:

```
x = 0x4B9CB381BCC737E493E43B2D7FD95BFE
2AEF6BE8F6224882E5E559ADA08170DC
49A815B3A1B3B323D2B50195153CFC60
DD6139C3770C5762A6A7719FABF84BFB
```

```
y = 0x95CEF28392C846A5EEFCB51C84E4960A
77B77D0D85EBD22061BFDA0013C5AB6C
42DDD04973F65D2AEB8A5427A53D6872
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
00030: 4C 93 36 57 A0 81 C2 06 09 2A 85 03 07 01 02 05
00040: 01 01 A0 81 AA 30 21 06 08 2A 85 03 07 01 01 01
00050: 02 30 15 06 09 2A 85 03 07 01 02 01 02 01 06 08
00060: 2A 85 03 07 01 01 02 03 03 81 84 00 04 81 80 FB
00070: 4B F8 AB 9F 71 A7 A6 62 57 0C 77 C3 39 61 DD 60
00080: FC 3C 15 95 01 B5 D2 23 B3 B3 A1 B3 15 A8 49 DC
00090: 70 81 A0 AD 59 E5 E5 82 48 22 F6 E8 6B EF 2A FE
000A0: 5B D9 7F 2D 3B E4 93 E4 37 C7 BC 81 B3 9C 4B D0
000B0: FB 94 01 2E AF 7A 0D 64 C4 22 F7 F5 68 2D CF 72
000C0: 68 3D A5 27 54 8A EB 2A 5D F6 73 49 D0 DD 42 6C
000D0: AB C5 13 00 DA BF 61 20 D2 EB 85 0D 7D B7 77 0A
000E0: 96 E4 84 1C B5 FC EE A5 46 C8 92 83 F2 CE 95 04
000F0: 08 FB F3 9D 10 E8 00 AF 70

```

Record layer message:

```

type: 16
version:
  major: 03
  minor: 03
length: 00F9
fragment:
  100000F53081F23081EF30280420D622
  D167A5642E29525A295CB9F28F96F28B
  0EFAA7D3A2BEE149B01178C2DFD50404
  . . .
  ABC51300DABF6120D2EB850D7DB7770A
  96E4841CB5FCEEA546C89283F2CE9504
  08FBF39D10E800AF70

```

```

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
00040: 03 07 01 02 05 01 01 A0 81 AA 30 21 06 08 2A 85
00050: 03 07 01 01 01 02 30 15 06 09 2A 85 03 07 01 02
00060: 01 02 01 06 08 2A 85 03 07 01 01 02 03 03 81 84
00070: 00 04 81 80 FB 4B F8 AB 9F 71 A7 A6 62 57 0C 77
00080: C3 39 61 DD 60 FC 3C 15 95 01 B5 D2 23 B3 B3 A1
00090: B3 15 A8 49 DC 70 81 A0 AD 59 E5 E5 82 48 22 F6
000A0: E8 6B EF 2A FE 5B D9 7F 2D 3B E4 93 E4 37 C7 BC
000B0: 81 B3 9C 4B D0 FB 94 01 2E AF 7A 0D 64 C4 22 F7
000C0: F5 68 2D CF 72 68 3D A5 27 54 8A EB 2A 5D F6 73
000D0: 49 D0 DD 42 6C AB C5 13 00 DA BF 61 20 D2 EB 85
000E0: 0D 7D B7 77 0A 96 E4 84 1C B5 FC EE A5 46 C8 92
000F0: 83 F2 CE 95 04 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:
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_read_MAC|K_write_MAC|K_read_ENC|K_write_ENC|IV_read|IV_write:
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

```

```
-----Client-----
```

```
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

```

-----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

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: 7BDDBB3C0A6A4A9E302B468CCD5CF786  
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  
  description:     00
```

```
00000:  01 00
```

```
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
```

## Contributors

### Ekaterina Griboedova

CryptoPro

Email: [griboedova.e.s@gmail.com](mailto:griboedova.e.s@gmail.com)

### Grigory Sedov

CryptoPro

Email: [sedovgk@cryptopro.ru](mailto:sedovgk@cryptopro.ru)

**Dmitry Eremin-Solenikov**

Auriga

Email: [dbaryshkov@gmail.com](mailto:dbaryshkov@gmail.com)**Lidiia Nikiforova**

CryptoPro

Email: [nikiforova@cryptopro.ru](mailto:nikiforova@cryptopro.ru)

## Authors' Addresses

**Stanislav Smyshlyaev (EDITOR)**

CryptoPro

18, Sushevsky val

Moscow

127018

Russian Federation

Phone: +7 (495) 995-48-20

Email: [svs@cryptopro.ru](mailto:svs@cryptopro.ru)**Dmitry Belyavskiy**

Cryptocom

14/2, Kedrova St.

Moscow

117218

Russian Federation

Email: [beldmit@gmail.com](mailto:beldmit@gmail.com)**Markku-Juhani O. Saarinen**

Independent Consultant

Email: [mjos@iki.fi](mailto:mjos@iki.fi)**Evgeny Alekseev**

CryptoPro

18, Sushevsky val

Moscow

127018

Russian Federation

Email: [alekseev@cryptopro.ru](mailto:alekseev@cryptopro.ru)