Addition of GOST Ciphersuites to Transport Layer Security (TLS)

<draft-chudov-cryptopro-cptls-01.txt>

Abstract

This document is intended to register new cipher suites for the Transport Layer Security (TLS) protocol, according to the procedure specified in section A.5 of [TLS], and to register a new TLS extension, according to section 5 of [TLSEXT]. Those cipher suites are based on Russian national cryptographic standards - key derivation algorithms based on GOST R 3410-94 and GOST R 3410-2001 public keys, GOST 28147-89 encryption algorithm and GOST R 34.11-94 digest algorithm.

Table of Contents

1  Introduction .................................................. 2
2  Proposed Ciphersuites ....................................... 3
3  CipherSuite Definitions ..................................... 3
3.1  Key Exchange .............................................. 3
This document proposes the addition of new cipher suites and extensions to the Transport Layer Security (TLS) protocol to support GOST R 34.11-94 digest, GOST 28147-89 encryption and VKO GOST R 34.10-94/2001 key exchange algorithms. The cipher suites defined here were proposed by CRYPTO-PRO Company for "Russian Cryptographic Software Compatibility Agreement" community.

Algorithms GOST R 34.10-94, GOST R 34.10-2001, GOST 28147-89 and GOST R 34.11-94 have been developed by Russian Federal Agency of Governmental Communication and Information (FAGCI) and "All-Russian Scientific and Research Institute of Standardization". They are described in [GOSTR341094], [GOSTR34102001], [GOSTR3411] and [GOST28147]. Algorithms VKO GOST R 34.10-94/2001 and PRF_GOSTR3411 are described in [CPALGS].

This document defines two configurations:
- anonymous client - authenticated server (only server provides a certificate);
- authenticated client - authenticated server (client and server exchange certificates).

The presentation language used here is the same as in [TLS]. Since this specification extends TLS, these descriptions should be merged with those in the TLS specification and any others that extend TLS.
This means, that enum types may not specify all possible values and structures with multiple formats chosen with a select() clause may not indicate all possible cases.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119].

2 Proposed CipherSuites

The new cipher suites proposed here have the following definitions:

CipherSuite TLS_GOST341094_WITH_GOST28147_OFB_GOST28147  = {0x00,0x80}
CipherSuite TLS_GOST34102001_WITH_GOST28147_OFB_GOST28147= {0x00,0x81}
CipherSuite TLS_GOST341094_WITH_NULL_GOSTR3411           = {0x00,0x82}
CipherSuite TLS_GOST34102001_WITH_NULL_GOSTR3411         = {0x00,0x83}

Note: The above numeric definitions for CipherSuites have not yet been registered.

3 CipherSuite Definitions

3.1 Key exchange

The cipher suites defined here use the following key exchange algorithms:

<table>
<thead>
<tr>
<th>CipherSuite</th>
<th>Key Exchange Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_GOST341094_WITH_GOST28147_OFB_GOST28147</td>
<td>VKO GOST R 34.10-94</td>
</tr>
<tr>
<td>TLS_GOST34102001_WITH_GOST28147_OFB_GOST28147</td>
<td>VKO GOST R 34.10-2001</td>
</tr>
<tr>
<td>TLS_GOST341094_WITH_NULL_GOSTR3411</td>
<td>VKO GOST R 34.10-94</td>
</tr>
<tr>
<td>TLS_GOST34102001_WITH_NULL_GOSTR3411</td>
<td>VKO GOST R 34.10-2001</td>
</tr>
</tbody>
</table>

Key derivation algorithms based on GOST R 3410-94 and GOST R 3410-2001 public keys (VKO GOST R 34.10-94, VKO GOST R 34.10-2001) are described in [CPALGS].

3.2 PRF, Signature and Hash

For a PRF, described in section 5 of [TLS], the cipher suites described here use PRF_GOSTR3411 (refer to section 5.1)

GOST R 3410-94/2001 signature is used for CertificateVerify message.

GOST R 34.11 digest algorithm ([GOSTR341194]) is used for CertificateVerify.signature.gostR3411_hash and Finished.verify_data (see sections 7.4.8 and 7.4.9 of [TLS])
### 3.3 Cipher and MAC

The following cipher algorithm and MAC functions are used (for details refer to section 5.1):

<table>
<thead>
<tr>
<th>CipherSuite</th>
<th>Cipher</th>
<th>MAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_GOST341094_WITH_GOST28147_OFB_GOST28147</td>
<td>GOST28147</td>
<td>IMIT_GOST28147</td>
</tr>
<tr>
<td>TLS_GOST34102001_WITH_GOST28147_OFB_GOST28147</td>
<td>GOST28147</td>
<td>IMIT_GOST28147</td>
</tr>
<tr>
<td>TLS_GOST341094_WITH_NULL_GOSTR3411</td>
<td>-</td>
<td>HMAC_GOSTR3411</td>
</tr>
<tr>
<td>TLS_GOST34102001_WITH_NULL_GOSTR3411</td>
<td>-</td>
<td>HMAC_GOSTR3411</td>
</tr>
</tbody>
</table>

For all four cipher suites, the use of MAC is slightly different from the one, described in section 6.2.3.1 of [TLS]. In [TLS], MAC is calculated from the following data:

\[
\text{MACed}_\text{data}[\text{seq_num}] = \text{seq_num} + \\
\text{TLSCompressed.type} + \\
\text{TLSCompressed.version} + \\
\text{TLSCompressed.length} + \\
\text{TLSCompressed.fragment};
\]

These cipher suites use the same input for first record, but for each next record the input from all previous records is concatenated:

\[
\text{MACed}_\text{data}[0] + \ldots + \text{MACed}_\text{data}[n]
\]

### 4 TLS Extensions for GOST

A new TLS extension -- the Hash, HMAC and PRF Select Extension -- allows a client and server to negotiate the use of a different hash, HMAC and PRF algorithm, instead of those, defined in [TLS]. It follows the general approach outlined in [TLSEXT]. The client enumerates supported hash/HMAC/PRF combinations, by including the appropriate extension in its ClientHello message. By echoing that extension in its ServerHello, the server selects one of them for this TLS connection. Sections 5.3 and 5.4 describe the structure of this extension in further details.

### 5 Data Structures and Computations

#### 5.1 Algorithms

GOST 28147-89 [GOST28147] uses 256-bit key size and 8-byte IV. Cipher suites, defined here, use GOST 28147-89 as a stream cipher in OFB mode with S-box gost28147-89-UZ-CryptoPro-A (see [CPALGS]) and CryptoPro key meshing algorithm. This is very similar to gost28147-89-CryptoPro-A-ParamSetAI parameter set, except for different cipher mode.
IMIT_GOST28147 is GOST 28147-89 [GOST28147] in "IMITOVSTAVKA" mode (4 bytes)

HMAC_GOSTR3411(secret, data) is based on GOST R 34.11 digest and described in [CPALGS].

PRF_GOSTR3411(secret, label, seed) is based on HMAC_GOSTR3411 and described in [CPALGS].

5.2 Key Calculation

Key calculation is done according to section 6.3 of [TLS], with PRF_GOSTR3411 function used instead of PRF. The parameters are as follows:

- SecurityParameters.hash_size = 32
- SecurityParameters.key_material_length = 32
- SecurityParameters.IV_size = 8

Length of necessary key material is 144 bytes.

5.3 Client Hello Extensions

When this message is sent:

The Hash, HMAC and PRF Select Extension SHOULD be sent along with any ClientHello message that includes cipher suites, proposed in this document.

Meaning of this message:

This extension allows client and server to override algorithms, predefined in [TLS], and select the appropriate set of algorithms and parameters for the chosen cipher suite.

Structure of this message:

The general structure of TLS extensions is described in [TLSEXT] and this specification adds a new type to ExtensionType.

```c
enum { hash_alg_select(60000) } ExtensionType;

hash_alg_select: Indicates the set of hash/HMAC/PRF algorithms supported by the client. For this extension, the opaque extension_data field contains DER-encoding of the TLSGostExtensionHashHMACSelectClient structure.

TLSGostExtensionHashHMACSelect ::= 
    SEQUENCE {
        hashAlgorithm AlgorithmIdentifier,
        ... 
    }
```
hmacAlgorithm AlgorithmIdentifier,
prfAlgorithm AlgorithmIdentifier
}

TLSGostExtensionHashHMACSelectClient ::= SEQUENCE OF
  TLSGostExtensionHashHMACSelect

Actions of the sender:

A client that proposes algorithm selection in its ClientHello appends
this extension along with any others.

Actions of the receiver:

A server that receives a ClientHello containing this extension MUST
use one of the proposed algorithm sets.

If a server does not understand this extension or is unable to
perform handshake using any of the proposed algorithm sets, it MUST
NOT negotiate the use of GOST cipher suites. Depending on what other
cipher suites are proposed by the client and supported by the server,
this may result in a fatal handshake failure alert due to the lack of
common cipher suites.

If Client Hello contains GOST cipher suites, but does not contain
this extension, the server MUST assume that client proposes the use
of the following default set:

hashAlgorithm ::= id-GostR3411-94 with NULL algorithmParameters
hmacAlgorithm ::= id-Gost28147-89-MAC with NULL algorithmParameters
prfAlgorithm  ::= id-PRF-GostR3411-94 with NULL algorithmParameters

5.4 Server Hello Extensions

When this message is sent:

The ServerHello Hash, HMAC and PRF Select Extension is sent in
response to a Client Hello message containing this extension.

Meaning of this message:

This extension indicates the server’s agreement to use one of the
sets of algorithms, given by client, during handshake.

Structure of this message:

"extension_data" field of the Server Hello extension contains DER-
encoding of the TLSGostExtensionHashHMACSelectServer structure.

\[
\text{TLSGostExtensionHashHMACSelectServer ::=}
\]
\[
\text{TLSGostExtensionHashHMACSelect}
\]

This MUST be one the elements of TLSGostExtensionHashHMACSelectClient sequence.

Actions of the sender:

A server chooses one algorithm set from the client list, and makes sure that it can complete a handshake using the chosen cipher suite and this algorithm set before sending this extension.

Actions of the receiver:

A client that receives a ServerHello with this extension makes sure, that server selected one of the algorithm sets, specified in the corresponding ClientHello extension, and proceeds with a handshake, using the algorithms specified in it.

5.5 Server Certificate

For these cipher suites this message is required and it MUST contain a certificate, with a public key algorithm matching ServerHello.cipher_suite.

5.6 Server Key Exchange

This message MUST NOT be used in these cipher suites, because all the parameters necessary are present in server certificate (see [CPPK]).

5.7 Certificate Request

This message is used as described in section 7.4.4 of [TLS], and extended as follows:

\[
\text{enum } \{
\text{gost341094}(21), \text{gost34102001}(22),(255)
\} \text{ ClientCertificateType;}
\]

gost341094 and gost34102001 certificate types identify that the server accepts GOST R 34.10-94 and GOST R 34.10-2001 public key certificates.

5.8 Client Key Exchange Message

This message is used as described in section 7.4.7 of [TLS], it is
required for these suites, and contains DER-encoded
TLSGostKeyTransportBlob structure.

enum { vko_gost } KeyExchangeAlgorithm;

struct {
    select (KeyExchangeAlgorithm) {
        case vko_gost: TLSGostKeyTransportBlob;
    }
} exchange_keys;

ClientKeyExchange;

ASN1-syntax for this structure is:

TLSGostKeyTransportBlob ::= SEQUENCE {
    keyBlob GostR3410-KeyTransport,
    proxyKeyBlobs SEQUENCE OF TLSProxyKeyTransportBlob OPTIONAL
}

TLSProxyKeyTransportBlob ::= SEQUENCE {
    keyBlob GostR3410-KeyTransport,
    cert OCTET STRING
}

GostR3410-KeyTransport is defined in [CPCMS].

keyBlob.transportParameters MUST be present.

keyBlob.transportParameters.ephemeralPublicKey MUST be present if the
server didn’t request client certificate or client’s public key
algorithm and parameters do not match those of the recipient. Else it
SHOULD be omitted.

proxyKeyBlobs - (optional) contains key exchange for secondary
recipients (for example, for the firewall, which audits
connections).

cert - contains secondary recipient’s certificate.

Actions of client:

First, the client generates a random 32-byte premaster_secret.

Then ukm is calculated as first 8 bytes of digest of concatenated
client random and server random: keyBlob.transportParameters.ukm =
GOSTR3411(client_random|server_random)[0..7]

Then client chooses a sender key. If
keyBlob.transportParameters.ephemeralPublicKey is present, the
corresponding secret key MUST be used as a sender key. If it is
missing, the secret key, corresponding to the client certificate MUST be used.

Using the sender key and recipient’s public key, algorithm VKO GOST R 34.10-94 or VKO GOST R 34.10-2001 (described in [CPALGS]) is applied to produce KEK. VKO GOST R 34.10-2001 is used with keyBlob.transportParameters.ukm as IV.

Then key wrap algorithm, specified by encryptionParamSet, is applied to encrypt premaster_secret and produce CEK_ENC and CEC_MAC. Again, keyBlob.transportParameters.ukm is used as IV. keyBlob.transportParameters.encryptionParamSet is used for all encryption operations.

The resulting encrypted key (CEK_ENC) is placed in keyBlob.sessionEncryptedKey.encryptedKey field, it’s mac (CEK_MAC) is placed in keyBlob.sessionEncryptedKey.macKey field, and synchrovector (IV) is placed in keyBlob.transportParameters.ukm field.

Actions of server:

Server MUST verify, that keyBlob.transportParameters.ukm is equal to GOSTR3411(client_random|server_random){0..7}, before decrypting the premaster_secret.

Server applies VKO GOST R 34.10-94 or VKO GOST R 34.10-2001, (depending on the client public key type), and key wrap algorithm (depending on encryptionParamSet) in the similar manner to decrypt the premaster_secret.

Server MUST verify keyBlob.sessionEncryptedKey.macKey after decrypting the premaster_secret.

5.9 Certificate Verify

This message is used as described in section 7.4.8 of [TLS]. If the client have sent both a client certificate and an ephemeral public key, it MUST send a certificate verify message, as a proof of possession of the private key for provided certificate.

The TLS structures are extended as follows:

```c
enum { gost341094, gost34102001 } SignatureAlgorithm;

select (SignatureAlgorithm) {
  case gost341094:
    digitally-signed struct {
```

Chudov, Leontiev Informational [Page 9]
opaque gost341194_hash[32];
};
case gost34102001:
    digitally-signed struct {
        opaque gost341194_hash[32];
    };
} Signature;

CertificateVerify.signature.gostR3411_hash = GOSTR3411(handshake_messages)

5.10 Finished

This message is used as described in section 7.4.9 of [TLS].

Finished.verify_data = PRF_GOSTR3411(master_secret, finished_label + GOSTR3411(handshake_messages)) [0..11]

6 Security Considerations

It is RECCOMENDED, that applications verify signature values and subject public keys to conform to [GOSTR34102001], [GOSTR341094] standards prior to their use.

Use of the same key for signature and key derivation is NOT RECOMMENDED.

It is RECOMMENDED for both client and server to verify the private key usage period, if this extension is present in the certificate.

The cipher suites TLS_GOST341094_WITH_GOST28147_OFB_GOST28147 and TLS_GOST34102001_WITH_GOST28147_OFB_GOST28147 proposed hereby, have been analyzed by special certification laboratory of Scientific and Technical Centre "ATLAS" in appropriate levels of target_of_evaluation (TOE).

It is RECOMMENDED to perform an examination of cipher suites implementations by authorized agency with approved methods of cryptographic analysis.

7 Appendix ASN.1 Modules

Additional ASN.1 modules, referenced here, can be found in [CPALGS] and [CPCMS].

7.1 Gost-CryptoPro-TLS
DEFINITIONS ::= BEGIN
  -- EXPORTS All --
  -- The types and values defined in this module are exported for
  -- use in the other ASN.1 modules contained within the Russian
  -- Cryptography "GOST" & "GOST R" Specifications, and for the use
  -- of other applications which will use them to access Russian
  -- Cryptography services. Other applications may use them for
  -- their own purposes, but this will not constrain extensions and
  -- modifications needed to maintain or improve the Russian
  -- Cryptography service.
  IMPORTS
    Certificate,
    AlgorithmIdentifier
    FROM PKIX1Explicit88 {iso(1) identified-organization(3)
      dod(6) internet(1) security(5) mechanisms(5) pkix(7)
      id-mod(0) id-pkix1-explicit-88(1)}
    id-CryptoPro-algorithms, gostR3410-EncryptionSyntax
    FROM Cryptographic-Gost-Useful-Definitions
    { iso(1) member-body(2) ru(643) rans(2)
      cryptographic-Gost-Useful-Definitions(0) 1 }
  GostR3410-KeyTransport
  FROM GostR3410-EncryptionSyntax
  gostR3410-EncryptionSyntax;

  id-PRF-GostR3411-94 OBJECT IDENTIFIER ::= { id-CryptoPro-algorithms prf-gostr3411-94(23) }

  TLSProxyKeyTransportBlob ::= SEQUENCE {
    keyBlob GostR3410-KeyTransport,
    cert OCTET STRING
  }

  TLSGostKeyTransportBlob ::= SEQUENCE {
    keyBlob GostR3410-KeyTransport,
    proxyKeyBlobs SEQUENCE OF
      TLSProxyKeyTransportBlob OPTIONAL
  }

  TLSGostSrvKeyExchange ::= SEQUENCE OF
    OCTET STRING (CONSTRAINED BY {Certificate})

  TLSGostExtensionHashHMACSelect ::= SEQUENCE {
hashAlgorithm AlgorithmIdentifier,
hmacAlgorithm AlgorithmIdentifier,
prfAlgorithm AlgorithmIdentifier
}

TLSGostExtensionHashHMACSelectClient ::= SEQUENCE OF
TLSGostExtensionHashHMACSelect
TLSGostExtensionHashHMACSelectServer ::= TLSGostExtensionHashHMACSelect

END -- Gost-CryptoPro-TLS

8 References


[CPPK] "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificates and Certificate Revocation List (CRL), corresponding to the algorithms GOST R 34.10-94, GOST R 34.10-2001, GOST R 34.11-94", IETF draft, <draft-ietf-pkix-gost-cppk-00.txt>, work in progress


Acknowledgments

This document was created in accordance with "Russian Cryptographic Software Compatibility Agreement", signed by FGUE STC "Atlas", CRYPTO-PRO, Factor-TC, MD PREI, Infotecs GmbH, SPRCIS (SPbRCZI), Cryptocom, R-Alpha. The aim of this agreement is to achieve mutual compatibility of the products and solutions.

The authors wish to thank:

Microsoft Corporation Russia for provided information about company products and solutions, and also for technical consulting in PKI.

RSA Security Russia and Demos Co Ltd for active colaboration and critical help in creation of this document.

NIP Informzachita for compatibility testing of the proposed data formats while incorporating them into company products.

Citrix Inc for help in compatibility testing Citrix products for Microsoft Windows.

Russ Hously (Vigil Security, LLC, housley@vigilsec.com) and Vasilij Sakharov (DEMOS Co Ltd, svp@dol.ru) for initiative, creating this document.

This document is based on a contribution of CRYPTO-PRO company. Any substantial use of the text from this document must acknowledge CRYPTO-PRO. CRYPTO-PRO requests that all material mentioning or referencing this document identify this as "CRYPTO-PRO CPTLS".

Author’s Addresses

Serguei Leontiev
CRYPTO-PRO
38, Obraztsova,
Moscow, 127018, Russian Federation
EMail: lse@cryptopro.ru

Grigorij Chudov
CRYPTO-PRO
38, Obraztsova,
Moscow, 127018, Russian Federation
EMail: chudov@cryptopro.ru

Alexandr Afanasiev
Factor-TC
and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.