GM

CRYPTOGRAPHIC INDUSTRY STANDARD

OF THE PEOPLE'S REPUBLIC OF CHINA

ICS 35.040

L 80

Reference No.: 44625-2014

GM/T 0024-2014

SSL VPN specification

SSL VPN 技术规范

Issued on: February 13, 2014 **Implemented on: February 13, 2014**

Issued by: National Cryptography Authority of China

Table of Contents

Foreword

This Standard was drafted in accordance with the rules given in GB/T 1.1-2009.

Attention is drawn to the possibility that some of the elements of this Standard may be the subject of patent rights. The issuing authority shall not be held responsible for identifying any or all such patent rights.

This Standard was proposed by and shall be under the jurisdiction of Cryptographic Industry Standardization Technical Committee.

Main drafting organizations of this Standard: Shanghai Geer Software Co., Ltd., Huawei Technologies Co., Ltd., Shenzhen Shen Xinfu Electronics Technology Co., Ltd., Shenzhen Austrian Union Technology Co., Ltd., Net Yu Shenzhou Technology (Beijing) Co., Ltd., Chengdu Weishitong Information Industry Co., Ltd., Beijing Oriental Huaxin Information Technology Co., Ltd., China International Electronic Commerce Limited, Lenovo Network Technology (Beijing) Limited, Shanghai Anderson Information Security Co., Ltd., Wuxi Jiangnan Information Security Engineering Technology Center, Beijing Tianrong letter Network Security Technology Co., Ltd.

Main drafters of this Standard: Liu Ping, Tan Wuzheng, Huang Min, Zeng Jianfa, Dan Bo, Liu Jianfeng, Luo Jun, Li Zhichao, Li Feibo, He Zhiyu, Chen Kai, Zhu Zhengchao, Ni Yongnian, Han Lin.

SSL VPN specification

1 Scope

This Standard specifies the technical agreement, product functionality, performance and management, and inspection of SSL VPN.

This Standard is applicable to the development of SSL VPN products. It can be also used to guide the inspection, management and use of SSL VPN products.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

GM/T 0005, *Randomness testing specification*

GM/T 0009, *SM2 cryptography algorithm usage specification*

GM/T 0010, *SM2 cryptography algorithm encryption signature message syntax specification*

GM/T 0014, *Digital certificate authentication system password protocol specification*

GM/T 0015, *Digital certificate format specification based on SM2 cryptography algorithm*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1 digital certificate

also known as a public key certificate, a data structure that contains public key owner information, public key, issuer information, expiration date, and extended information signed by the certificate authority (CA); it can be divided into personal certificate, institutional certificate and equipment certificate according to category OR signature certificate and encryption certificate

according to use

3.2 identity based cryptography algorithm

the IBC algorithm is also known as the identity cipher algorithm; it is an asymmetric cryptography algorithm that can be arbitrarily identified as a public key and does not require a digital certificate to prove the public key

3.3 IBC identity

an IBC identity is a string that represents an entity's identity or attribute

3.4 IBC public parameter

IBC public parameter contains the public parameter information such as name, calculation curve, identification coding method and key generation algorithm of IBC key management center; the information is used to convert the entity ID to public key

3.5 initialization vector / initialization value; IV

initial data used for data transformation and introduced to increase security or synchronize cryptographic devices during password conversion

3.6 secure sockets layer protocol

a transport layer security protocol used to build a secure channel between client and server

3.7 payload

the data format of the ISAKMP communication exchange message, the basic unit of the ISAKMP message

3.8 session

the association between client and server created by the handshake protocol; a session can be shared by multiple connections

3.9 SM1 algorithm

a packet cipher algorithm with a packet length of 128 bits and a key length of 128 bits

3.10 SM2 algorithm

an elliptic curve public key cryptography algorithm with key length of 256 bits

3.11 SM3 algorithm

A (0) = seed;

A (i) = HMAC (secret, A (i-1));

P_hash can iterate iteratively until the data of required length is generated.

5.1.5 Pseudo-random function PRF

The calculation method of PRF is as follows:

PRF (secret, label, seed) = P_SM3 (secret, label + seed)

5.2 Key type

5.2.1 Overview

In this Standard, it uses asymmetric cryptography algorithm to carry out identity authentication and key exchange. Identify the pre-master key after post-negotiation. Each party calculates the master key, and then derives the work key. Use the work key for encryption and decryption and integrity verification.

5.2.2 Server key

When server key is the key of asymmetric cryptography algorithm, including signature key pair and encryption key pair. The signature key pair is generated by the VPN self-password module. The encryption key is applied to KMC through the CA Certification Center, used for negotiation of server-side identity authentication and pre-master key during handshaking.

5.2.3 Client key

The client key is the key pair of the asymmetric cryptography algorithm, including signature key pair and encryption key pair. The signature key pair is generated by the VPN self-password module. The encryption key is applied to KMC through the CA Certification Center, used for negotiation of server-side identity authentication and pre-master key during handshaking.

5.2.4 Pre_master_secret

Pre_master_secret is the key material generated by both parties, used to generate master secret.

5.2.5 Master_secret

The master secret is the key material consisted of pre-master secret, client random number, server random number, constant string after calculation, used to generate work key.

lower limit and y represents the upper limit; if it only needs to express the upper limit, use <y>. The length of all vectors is in bytes. The variable length vector represents the actual length of the vector whose header size is the minimum number of bytes that can accommodate the maximum length of the variable length vector.

6.2.3 Enumerateds

Enumerateds are a field collection of a set of specific values. Usually each field includes a name and a value. If it contains an unnamed value, this value shall represent the specified maximum value. If it only includes a name without defining a value, it shall only be used to refer to a state value and can not be used in actual encoding. For example, enum {red (0), green (1), (255)} color. Enumerated variable size is the minimum number of bytes that can hold the maximum enumeration value.

6.2.4 Constructed types

Constructed Types are defined by struct, similar to the struct syntax of C language. The fields in the struct are concatenated in sequence. If a struct is included in another struct, it shall omit the name of the struct.

6.2.5 Variants

Variables types are defined by select, case, used to define structures that depend on external information, similar to union or ASN.1 CHOICE in C language.

6.3 Record layer protocol

The record layer protocol is hierarchical, and each layer includes length field, description field, and content field. The record layer protocol receives messages that shall be transmitted, blocks the data, compresses (optional), computes HMAC, encrypts, and then transfers. The received data is decrypted, verified, decompressed (optional), re-encapsulated and then passed to high-level applications. Record layer protocol includes: handshake, alarm, password specification change and gateway to gateway and other types. To support protocol extensions, the record layer protocol can support other record types. Any new record types must be allocated outside of the content type assigned to the above types. If an unrecognized record type is received, it shall be ignored.

6.3.1 Connection state

The connection state is the operating environment of the record layer protocol. It includes four typical connection states: current read and write states, pending read and write states. The read indicates the received data, and the

d) MACAlgorithm:

cryptographic hash algorithm used to calculate and verify the message integrity; defined as:

enum {sha_1, sm3} MACAlgorithm

e) hash_size:

hash length

f) CompressionMethod

an algorithm used for data compression; defined as:

enum {null(O), (255)} CompressionMethod

g) master_secret

the 48-byte key calculated by the pre-master key, the client random number, and the server-side random number during the negotiation

h) client_random

32 bytes of random data generated by the client

i) server_random

32 bytes of random data generated by the server.

j) record_iv_length

IV length

k) mac_length

MAC length

The record layer shall use the above security parameters to generate the following:

- client writes verification key “client write MAC secret”;
- server writes verification key “server write MAC secret”;
- client writes key “client write key”;
- server writes key “server write key”.

The server uses the client write parameters when receiving and processing

used for both parties to provide the security parameters used by the record layer, authenticate, and report errors to each other, etc.

The handshake agreement family is responsible for negotiating a session that contains:

session ID: random byte sequence selected by the server to identify active or recoverable sessions;

certificate: X.509 v3 format digital certificate, in accordance with GM/T 0015;

compression method: an algorithm for compressing data;

password specification: the specified cryptography algorithm;

mater secret: a 48-byte key shared by the client and the server;

reuse ID: indicating whether the identity of a new connection can be initiated with the session.

Use the above data to generate security parameters. With the reuse feature of the handshake protocol, it can use the same session to establish multiple connections.

6.4.1 Password specification change protocol

The password specification change protocol is used to inform the password specification of the change, i.e., inform the other party to use the newly negotiated security parameters to protect the next data. This protocol consists of a message. The message is compressed with the current compression algorithm and encrypted with the current password specification. If it is the first consultation, the message shall be plaintext.

The length of the message is one byte and its value is 1. Both the client and the server shall send this message before the handshake message ends after the security parameters have been negotiated.

For a key that has just been negotiated, the write key shall be enabled immediately after this message is sent. The read key is enabled immediately after receiving this message.

The password specification change message structure is defined as follows:

```
struct {
    enum { change_cipher_spec(1), (255) } type;
} ChangeCipherSpec;
```

identity_need	205	fatal	Missing the other party's ibc identifier

For those who do not explicitly indicate the level of false alarm, the sender can decide whether it is fatal. If the sender thinks that is fatal, it shall inform the receiver to close the notice, and finally close the connection. If a warning level is received, the receiver can decide whether it is fatal. If the receiver thinks that is fatal, it shall inform the sender to close the notice, and finally close the connection.

6.4.3 Handshake protocol overview

The handshake protocol involves the following processes:

- exchange the hello messages to negotiate a cipher suite, exchange random numbers, and decide whether to reuse the session;
- exchange the necessary parameters, negotiate the pre-master secret;
- exchange certificate or IBC information to verify each other;
- generates the master secret by using the pre-master secret and the exchanged random number;
- provide security parameters to the record layer;
- verify the consistency of the security parameters calculated by both parties, the authenticity and integrity of the handshake protocol.

The handshake process is as follows: the client sends the client-side Hello message to the server, the server should respond to the hello message, otherwise a fatal error is generated and the connection is disconnected. Client-side Hello and server-side Hello are used for client and server based on RSA, ECC or IBC algorithm negotiation, as well as determinations of secure transmission capability, including protocol version, session ID, password suite and other attributes, and to generate and exchange random numbers.

The client-side Hello and server-side Hello messages are followed by the authentication and key exchange process, including server certificate, server key exchange, client certificate, client key exchange. After sending the hello message on the server, it shall send its own certificate message, the server key exchange message. If the server needs to verify the identity of the client, send a certificate request message to the client. And then send a server-side Hello completion message to indicate that the hello message phase has ended, the server waiting for the client to return the message. If the server sends a certificate request message, the client must return a certificate message. The client then sends a key exchange message, which depends on

```
struct {
    _unix_time;
 opaque random_bytes[28];
 } Random;
```

gmt_unix_time is the sender clock represented by the standard UNIX 32-bit format, which is the number of seconds from January 1, 1970 to the current time from Greenwich Mean Time.

random_bytes is a random number of 28 bytes.

c) session_id:

the session identifier used by the client in the connection; defined as:

opaque SessionID<0..32>

session_id is a variable-length field whose value is determined by the server. If there is no reusable session ID or want to negotiate security parameters, the field shall be empty, otherwise the client wants to reuse the session. This session ID may be the previous connection identifier, the current connection identifier, or other connection identifier. After the session ID is generated, it shall remain until the timeout is deleted or the connection associated with this session encounters a fatal error that is closed. When a session fails or is closed, the connection associated with it shall be forcedly off.

d) cipher_suites:

a list of cipher suites supported by the client; the clients shall follow the priority order used by the cipher suite, and the highest priority cipher suite shall be in the first place. If the session ID field is not null, this field shall contain at least the cipher suite used to reuse the session.

The cipher suite is defined as follows:

uint8 CipherSuitc[2].

Each cipher suite includes a key exchange algorithm, an encryption algorithm, and a verification algorithm. The server shall select a matching cipher suite in the cipher suite list. If there is no matching cipher suite, it shall return a handshake failure alarm message handshake_failure and close the connection.

The list of cipher suites supported by this Standard is shown in Table 2.

directly from the server's encryption certificate.

```
struct {
    ECParameters curve_params;
    ECPoint public;
} ServerECDHParams;
```

If it uses the SM2 algorithm, the first number shall not be verified.

b) ServerIBSDHParams:

when using the IBSDH algorithm, the server's key shall exchange parameters. The key exchange parameter format is shown in the SM9 algorithm.

c) ServerIBCParams:

when using the IBC algorithm, the server's key shall exchange parameters. The key exchange parameter format is shown in the SM9 algorithm.

d) IBCEncryptionKey:

when using the IBC algorithm, the length of the server's encrypted public key is 1024 bytes.

e) signed_params:

when the key exchange modes are ECDHE, IBSDH and IBC, signed_params shall be the signature of both server's random number and server key exchange parameters. When the key exchange modes are ECC and RSA, signed_params shall be the signature of server random number and server encryption certificate.

6.4.4.4 Certificate Request message

This message is a certificate request message.

If the client requires an authentication client, this message shall be sent to require the client to send its own certificate.

This message follows the server key exchange message.

The structure of the certificate request message is defined as follows:

```
struct {
    ClientCertificateType certificate_types<1..2^8-1> ;
    DistinguishedName certificate_authorities<0..2^16-1> ;
} CertificateRequest;
```

```
struct {

  ECParameters    curve_params;
  ECPoint         public;
} ClientECDHParams;
```

If it uses the SM2 algorithm, the first parameter shall not be verified.

b) ClientIBSDHParams:

when using the IBSDH algorithm, the client's key shall exchange parameters.

c) ECCEncryptedPreMasterSecret:

when using the ECC encryption algorithm, it shall use the pre-master secret encrypted with the server encryption public key.

d) IBCEncryptedPreMasterSecret:

when using the IBC encryption algorithm, it shall use the pre-master secret encrypted with the server encryption public key.

e) RSAEncryptedPreMasterSecret:

when using the RSA encryption algorithm, it shall use the pre-master key encrypted with the server encryption public key.

The data structure of the pre-master secret is as follows:

```
struct {
  ProtocolVersion client_version;
  opaque random[46];
} PreMasterSecret;
```

where,

1) client_version:

the version number supported by the client. The server shall check whether the value matches the value sent in the client-side Hello message.

2) random:

random number of 46 bytes.

6.4.4.8 Certificate Verify message

6.4.4.9 Finished message

The message is a handshake end message.

The server and the client shall respecctively transmit the message after the password specification change message so as to verify that the key exchange process is successful and verify the integrity of the handshake process.

This message is protected by algorithms and keys negotiated with this handshake process.

The receiver of this message must match the correctness of the message content. Once a party sends a handshake end message and receives a handshake end message from each other and passes the check, the connection shall be used for secure transmission of data.

The handshake end message data structure is as follows:

```
struct {
  opaque verify_data[12];
    } Finished;
```

where,

verify_data is the check data, and the data is generated as follows:

```
PRF(master_secret, finished_label, SM3(handshake_messages)) [0..11]
```

where,

a) finished_label:

for the end message sent by the client, the tag is the string "client finished". For the server, the tag is the string "server finished".

b) handshake_messages:

referring to all messages related to handshaking, including the type and length field of the handshake message, from the client-side Hello message until this message (excluding this message, password specification change message).

6.5 Key calculation

6.5.1 Master secret calculation

The master secret consists of 48 bytes, generated by the pre-master secret, the client random number, the server random number, the constant string,

fixed and the remaining bits are wildcard values.

- ipv6_addr_range:

  indicating the address range of ipv6. It consists of two 128-bit values. The first value represents the starting address of the address range, and the second value represents the end address of the address range. All IP addresses that fall within these two address ranges are considered to be within the address range.

d) Site2SiteData:

gateway-to-gateway data packets; the contents are the original complete IP packets.

6.6.3 Control message exchange process

The exchange process of the control message is initiated by either party on both sides of the communication to inform the other party of the network information. The control message can be sent at any time after the handshake protocol is completed, using the security parameter protection negotiated in the handshake protocol.

If the control packet of the new transmission protection domain is received during the transmission of the gateway-to-gateway data message, the reuse process of the connection shall be triggered. For details, see 6.4.3.

6.6.4 Data packet outbound-inbound process

SSL VPN gateways establish SSL connection through the handshake protocol and bind the connection to the local and peer protection domains. The incoming and outgoing IP packets need to match the peer and local protection domains. Local and peer protection domains can be obtained by controlling packets or manually.

6.6.4.1 Outbound message processing

When the IP packets in the local protection domain are forwarded through the SSL VPN, the SSL VPN shall search for the matching peer protection domain and the local protection domain according to the destination IP address and source IP address of the packet to obtain the corresponding valid SSL connection. And then encapsulate the entire IP packet as the data content of the recording layer on the connected record layer. Transmit through the record layer. The encapsulation format is detailed in 6.3.2. If there is no corresponding valid connection, the new handshake process shall be triggered or the process of the connection shall be reused. See 6.4.3. Negotiate to establish or update the corresponding connection. If the

authentication of the server is a prerequisite function. The authentication of the client is optional. It shall support the authentication mechanism based on digital certificates (RSA or ECC) or based on identity algorithm. Any kind of authentication method needs to ensure the integrity and validity of the identification.

7.1.6 Access control

SSL VPN products should have fine-grained access control, effective control of resources based on user or user groups. At least the network access should be controlled to the IP address, port. The access to the Web resource should be controlled at least to the URL, and can be controlled according to the access time.

7.1.7 Key update

SSL VPN products should have the ability to perform key updates based on time periods or message traffic. The update according to the time period is a prerequisite function. The update according to the message flow is an optional function. When the update is based on the time period, the maximum length of the client-server mode shall be no more than 8 h; the maximum length of the gateway- gateway mode shall be no more than 1 h.

7.1.8 Client host security inspection

SSL VPN products should have client host security inspection function. When the client connects to the server, the client shall check the security of the user's operating system according to the client's security policy. Users who do not meet the security policy shall not be able to use SSL VPN.

The client security policy shall include at least one of the following conditions:

- whether anti-virus software is installed and enabled;
- whether a personal firewall is installed and enabled;
- whether the latest operating system security patch has been installed;
- whether the login password has been set for the system.

7.2 Product performance parameters

7.2.1 The maximum number of concurrent users

It refers to the maximum number of online users at the same time. This indicator reflects the maximum number of users who can deliver the product at the same time.

7.3.1.2 Configure data security

All configuration data shall ensure its integrity and reliability in the equipment. The management interface should be configured and managed. The administrator should enter the management interface through identity authentication.

7.3.1.2.1 Hardware security

SSL VPN products should provide security measures to ensure the storage security of cryptography algorithm, key, key data.

All cryptographic operations should be performed in separate cryptographic components.

In addition to the necessary communication interface and management interface, it shall not provide any external interface for debugging, tracking. Internal debugging, the detection interface should be closed after the product is stereotyped.

7.3.1.2.2 Software security

All security protocols and management software should be implemented independently.

The operating system should be securely secured. Shut down all unneeded ports and services.

Any operational instructions and any combination thereof can not disclose key and sensitive information.

7.3.1.2.3 Client security

SSL VPN client products should have integrity of the self-inspection function, including the vendor's signature of the client software so as to protect the integrity.

7.3.2 Management requirements

7.3.2.1 Log management

SSL VPN products should provide log recording, viewing, and export capabilities.

The log content includes:

- administrator operations, including login authentication, system configuration, key management and other operations;

7.1.1.

8.1.2 Random number function

Extract samples in accordance with the requirements of GM/T 0005. Carry out the detection according to relevant requirements of this specification. The detection results shall be qualified.

8.1.3 Key exchange

The key exchange protocol shall be in accordance with 6.4 requirements. The detection results shall comply with the requirements of 7.1.3.

8.1.4 Secure packet transmission

The security packet encapsulation protocol shall be in accordance with 6.3 requirements. The detection results shall comply with the requirements of 7.1.4.

8.1.5 Identification

The identification shall be carried out according to requirements of 6.4. The detection results shall comply with the requirements of 7.1.5.

8.1.6 Access control

It shall only access to the authorized resources when it visits the server-protected intranet server from the client. The detection results shall comply with the requirements of 7.1.6.

8.1.7 Key update

The key update shall be carried out according to requirements of 6.4. The detection results shall comply with the requirements of 7.1.7.

8.2 Product performance detection

8.2.1 The maximum number of concurrent users

Simulate multiple client behavior on the detection platform. Establish an SSL session with the server. Download 512 bytes of data from the intranet server. And set the page delay on the intranet server to ensure that every session is maintained and data is passed during the entire load increase process. Then, increase the client and repeat this process. Take the average number of concurrent sessions of the load stabilization period as the test results.

8.2.2 The maximum number of concurrent connections

Simulate multiple client behavior on the detection platform. Connect with the

server and keep it. And then continue to add the client, and repeat this process until it is unable to establish and maintain the connection. The number of SSL connections that have been accessed shall be the test results.

8.2.3 The number of new connections per second

Simulate multiple client behavior on the detection platform. Concurrency establish an SSL session with the server. Repeat this process for a while. Take the average of the number of SSL sessions per second as the test results.

8.2.4 Throughput rate

Simulate multiple client behavior on the detection platform. Establish an SSL session with the server. In this session, download 1 MB of data from the intranet server. Repeat the above steps until each user successfully downloads 20 MB of data. And then upload 1 MB data to the intranet server. Repeat the above steps until each user successfully uploads 20 MB of data. Take the average rate of the data sent and received by the intranet server as the test results.

8.3 Security management detection

8.3.1 Security detection

8.3.1.1 Key security

8.3.1.1.1 Server key

The detection results shall comply with the requirements of 7.3.1.1.1.

8.3.1.1.2 Work key

The detection results shall comply with the requirements of 7.3.1.1.2.

8.3.1.2 Configure data security

The detection results shall comply with the requirements of 7.3.1.2.

8.3.1.2.1 Hardware security

The detection results shall comply with the requirements of 7.3.1.2.1.

8.3.1.2.2 Software security

The detection results shall comply with the requirements of 7.3.1.2.2.

8.3.1.2.3 Client security

This is an excerpt of the PDF (Some pages are marked off intentionally)

------ The End ------