TLS Working Group   Ari Singer
INTERNET-DRAFT   NTRU Cryptosystems
Expires: January 3, 2002   July 3, 2001

NTRU Cipher Suites for TLS
<draft-ietf-tls-ntru-00.txt>

Status of this Memo

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Abstract

This document defines a group of new TLS cipher suites that utilize the NTRU encryption algorithm and the NSS signature algorithm. These cipher suites are designed to maximize computational efficiency on both the client and server sides and ease deployment of the TLS protocol on constrained and embedded devices. The document assumes the reader is familiar with the TLS protocol.

Table of Contents

1. Overview

The TLS protocol was designed with the purpose of enabling private and authenticated communication over the Internet, typically between high bandwidth and computationally rich entities. This goal is achieved through the use of a combination of public-key techniques, which provide authentication and key agreement between two parties that may not have a-priori knowledge of each other, and symmetric key techniques, which provide data privacy, continued authentication and efficiencies in bandwidth and computational effort.

The purpose of this document is to specify new cipher suites that significantly reduce the computational cost of public-key operations and that are easy to implement on constrained devices. In practice, the computational cost of the public-key computations in TLS causes the most latency in the handshake and is often the limiting factor for server capacity of running multiple TLS sessions at once. In addition, TLS, especially with client authentication, may be difficult to implement on wireless devices and other constrained devices due to memory and computational limitations. The use of the NTRU encryption algorithm in the TLS handshake and the use of the NSS signature algorithm in the handshake and digital certificates may allow for greater server scalability, decreased latency in the handshake and deployment of the TLS protocol on a larger population of devices.

This document specifies only cipher suites and the aspects of the TLS protocol that need to be modified in order to implement the cipher suites. No other changes to the TLS protocol are required. For use of this Internet Draft, familiarity with the TLS protocol is assumed. For full details of the TLS protocol, see RFC 2246 [RFC2246].

In this document, the terms Client Hello, Server Hello and Client Response refer to the grouping of consecutive messages sent in the initial handshake by either the server or client. The terms ClientHello, ServerKeyExchange, etc. refer to the specific messages defined in the TLS specification [RFC2246].

2. NTRU Key Exchange Algorithms

This document defines two new key exchange algorithms based on NTRU for use within TLS. These algorithms all utilize the NTRU encryption algorithm, but utilize different authentication mechanisms.

The key exchange algorithms are as follows:

Key Exchange Algorithm	Description
NTRU_NSS	NTRU encryption with NSS signatures
NTRU_RSA	NTRU encryption with RSA signatures

In both of the above key exchange algorithms, the server SHALL send an X.509 certificate in the Certificate field that is included in the Server Hello. This certificate SHALL contain an NTRU public key and be signed with the specified signature algorithm. The client SHALL verify that the certificate is valid and, if it is valid, generate a pre-master secret and encrypt it with the server public key. After successful verification of the server certificate, the client SHALL send the encrypted pre-master secret in the ClientKeyExchange field that is included in the Client Response. If the server desires for the client to be authenticated, the server MAY request a client certificate. Client certificate types for client authentication are defined in this document and specified in section 3.

The key strength of the NTRU public key determines the size of the pre-master secret. The following table shows the required sizes of the pre-master secret with the corresponding NTRU key strength. NTRU 251, 347 and 503 provide roughly equivalent security to RSA 1024, RSA 2048 and RSA 4096 respectively.

Key Strength	Pre-master Secret Size
NTRU 251	20 Bytes
NTRU 347	32 Bytes
NTRU 503	48 Bytes

2.1 NTRU_NSS

For key exchanges using NTRU encryption with NSS signatures, the server certificate SHALL be an X.509 certificate that includes an NTRU public key and is signed by the NSS signature algorithm. The basic X.509 certificate structure may be found in RFC 2459 [RFC2459] and the ASN.1 syntax for NTRU public keys and NSS digital signatures may be found in Efficient Embedded Security Standard (EESS) #1 [EESS#1].

The NSS signature verification on the certificate and the NTRU encryption and decryption of the pre-master secret SHALL be performed as specified in EESS #1 [EESS#1]. The exact data structure for NTRU and NSS public keys, NTRU encrypted data and NSS signatures are defined in EESS #1 [EESS#1] and explicitly specified in section 4.

2.2 NTRU_RSA

For key exchanges using NTRU encryption with RSA signatures, the server certificate SHALL be an X.509 certificate that includes an NTRU public key and is signed using the RSA signature algorithm.

The approved methods for computing and verifying RSA signatures are listed in RFC 2246 [RFC2246], which references PKCS #1 [PKCS1]. The NTRU encryption and decryption of the pre-master secret SHALL be performed as specified in EESS #1 [EESS#1]. The data structures for RSA signatures are specified in RFC 2246 [RFC2246], which references

PKCS #1 [PKCS1]. The exact data structure for NTRU public keys and NTRU encrypted data are defined in EESS #1 [EESS#1] and explicitly specified in section 4.

3. NTRU Client Authentication

This document defines two new methods of client authentication based on NTRU for the TLS protocol. Both techniques utilize the NSS signature algorithm and the certificates MAY be signed using either RSA or NSS. If client authentication is desired during a TLS handshake, the client MAY present a certificate in either of the formats defined below or in any format permitted by RFC 2246 [RFC2246].

The client authentication mechanisms are as follows:

Client Certificate Type	Description
nss_sign	NSS signature key certificate signed by NSS
rsa_nss	NSS signature key certificate signed by RSA

During the initial handshake between the server and client, the server MAY send a certificate request. The certificate request SHALL include an indication of the types of certificates that the server accepts and the certificate authorities that the server trusts.

When a client certificate is requested by the server, if an appropriate certificate is available, the client SHALL include a client certificate in the Certificate field of the Client Response. In addition, the client SHALL include a digital signature of the handshake messages in the CertificateVerify field of the Client Response.

3.1 nss_sign

For client authentication using NSS signatures, the client certificate SHALL be an X.509 certificate that includes an NSS public key and is signed by the NSS signature algorithm. The ASN.1 syntax for NSS public keys and NSS digital signatures may be found in EESS #1 [EESS#1].

The NSS signature generation on the handshake messages and the NSS signature verification of the certificate and handshake messages SHALL be performed as specified in EESS #1 [EESS#1]. The exact data structure for NSS public keys and NSS signatures are defined in EESS #1 [EESS#1] and explicitly specified in section 4.

3.2 rsa_nss

For client authentication using NSS signatures and RSA signed certificates, the client certificate SHALL be an X.509 certificate

that includes an NSS public key and is signed using the RSA signature algorithm.

The approved methods for computing and verifying RSA signatures are listed in RFC 2246 [RFC2246], which references PKCS #1 [PKCS1]. The NSS signature and verification of the handshake messages SHALL be performed as specified in EESS #1 [EESS#1]. The data structures for RSA signatures are specified in RFC 2246 [RFC2246], which references PKCS #1 [PKCS1]. The exact data structure for NSS public keys and NSS signatures are defined in EESS #1 [EESS#1] and explicitly specified in section 4.

4. Message Structures

This section defines the specific message data structures necessary for implementation of the TLS protocol using the NTRU and NSS cryptographic algorithms. These definitions should be taken within the context of the TLS protocol as defined in RFC 2246 [RFC2246] and used where appropriate. The naming conventions for the specific fields are consistent with RFC 2246 [RFC2246]. The cryptographic computations and encoding of NTRU and NSS cryptographic data items are specified in section 5.

The following data structures are defined in this section:

Data Structure	Description
Server Certificate	The signed data structure that is sent during the Server Hello for server authentication and key exchange.
Server Certificate Request	The data structure that is used to request a client certificate for client authentication.
Client Certificate	The signed data structure that is sent

	during the Client Response for client authentication.
Client Key Exchange	The data structure that includes the encrypted pre-master secret for the upcoming secure session.
Client Certificate Verify	The signature that is sent during the Client Response that authenticates the client for that handshake.

4.1 Server Certificate

When this message will be sent:

In all non-anonymous TLS handshakes, the server sends a server certificate to the client during the Server Hello. This message will always immediately follow the ServerHello message.

Meaning of this message:

The certificate type SHALL be appropriate for the selected cipher suite's key exchange algorithm and is generally an X.509v3 certificate. The public key may be of any length. The

certificate is used to authenticate the server to the client and provide the encryption key for key exchange.

Key Exchange Algorithm	Certificate Key Type
NTRU_NSS	NTRU encryption key; the certificate SHALL allow the key to be used for encryption. The algorithm used to sign the certificate SHALL be NSS.
NTRU_RSA	NTRU encryption key; the certificate SHALL allow the key to be used for encryption. The algorithm used to sign the certificate SHALL be RSA.

The certificate profiles are defined by the IETF PKIX working group [RFC2459]. NTRU and NSS key and cryptographic formats are defined by the CEES [EESS#1]. RSA key and cryptographic formats are defined by PKCS #1 [PKCS#1]. When a key usage extension is present, the keyEncipherment bit must be present to allow encryption.

Structure of this message:

```
     opaque ASN.1Cert<1..2^24-1>;

     struct {
            ASN.1Cert certificate_list<0..2^24-1>;
        } Certificate;
```

certificate_list
     This is a sequence of X.509v3 certificates as specified in RFC 2246 [RFC2246] except that the key, signature and parameter fields for NTRU and NSS are as specified in EESS #1 [EESS#1].

4.2 Server Certificate Request

When this message will be sent:
     A non-anonymous server MAY request a certificate from the client if client authentication is desired. This message, if sent, SHALL be sent in the Server Hello immediately following the ServerKeyExchange message, if it is sent, or the Server Certificate message.

Meaning of this message:
     This message informs the client that the server requests the use of a client certificate. It informs the client of the types of certificate accepted by the server and

Structure of this message:
     For consistency with draft-ietf-tls-ecc-01.txt, the TLS CertificateRequest message is extended as follows:

```
     enum { nss_sign (8), nss_rsa (9), (255)
     } ClientCertificateType;
```

nss_sign

     The server requests a client certificate that contains an NSS key and is signed by an NSS signature.

rsa_nss
     The server requests a client certificate that contains an NSS key and is signed by an RSA signature.

4.3 Client Certificate

When this message will be sent:
     If the server has requested a client certificate and the client wishes to send a certificate to the server, the client MAY send a Client Certificate message. This message, if sent, SHALL be the first message in the Client Response.

Meaning of this message:
The certificate type SHOULD be selected among the certificate types requested by the server and is generally an X.509v3 certificate. The public key may be of any length. The certificate is used to authenticate the client to the server.

The certificate profiles are defined by the IETF PKIX working group [RFC2459]. NSS keys and cryptographic formats are defined by the CEES [EESS#1]. RSA keys and cryptographic formats are defined by PKCS #1 [PKCS#1]. When a key usage extension is present, the digitalSignature bit must be set for the key to be eligible for signing.

Structure of this message:

```
opaque ASN.1Cert<1..2^24-1>;

struct {
        ASN.1Cert certificate_list<0..2^24-1>;
} Certificate;
```

certificate_list
This is a sequence of X.509v3 certificates as specified in RFC 2246 [RFC2246] except that the key, signature and parameter fields for NTRU and NSS are as specified in EESS #1 [EESS#1].

4.4 Client Key Exchange

When this message will be sent:
This message is always sent in the Client Response. It SHALL immediately follow the client certificate message, if it is sent. Otherwise, it SHALL be the first message in the Client Response.

Meaning of this message:
This message contains the NTRU encrypted pre-master secret. The pre-master secret is used in TLS along with the other data to calculate the master secret. Depending on the NTRU key size, the pre-master secret will have different sizes. For the

table of permitted key sizes and pre-master secret sizes, see section 2.

Structure of this message:
The structure of the message depends on the selected key

exchange method. The KeyExchangeAlgorithm and ClientKeyExchange from RFC 2246 [RFC2246] are extended to include NTRU.

NOTE: The operation public-key-encrypted is defined in RFC 2246 [RFC2246] section 4.7 and specifies that the length is represented as an opaque vector <0..2^16-1>, where the length is specified by the encryption algorithm (e.g. NTRU) and key.

```
enum { ntru } KeyExchangeAlgorithm;
```

ntru
: The KeyExchangeAlgorithm message contains an NTRU public key.

```
struct {
        select (KeyExchangeAlgorithm) {
                case ntru: NTRUEncryptedPreMasterSecret;
        } exchange_keys;
} ClientKeyExchange

enum { NTRU251 (1), NTRU347 (2), NTRU503 (3), (255)
} NTRUKeyStrength
```

NTRU251, NTRU347, NTRU503
: The number in the NTRU key strength name represents the size of the NTRU degree N (e.g. NTRU251 has degree N equal to 251).

```
select (NTRUKeyStrength) {
        case NTRU251: opaque random [20];
        case NTRU347: opaque random [32];
        case NTRU503: opaque random [48];
} PreMasterSecret
```

random
: The random variable is a securely generated random value to be used as the pre-master secret.

```
struct {
        public-key-encrypted PreMasterSecret pre_master_secret;
} NTRUEncryptedPreMasterSecret;
```

pre_master_secret
: The value of the pre-master secret, which is encrypted with the NTRU encryption key provided by the server to obtain the NTRUEncryptedPreMasterSecret.

4.5 Client Certificate Verify

When this message will be sent:

This message SHALL only be sent following a client certificate that contains a key that has singing capability (e.g. an NSS signing key). When sent, it SHALL immediately follow the client key exchange message in the Client Response.

Meaning of this message:
This message is the digital signature of all of the preceding handshake messages and is used to provide explicit verification of a client certificate.

Structure of this message:
The SignatureAlgorithm and Signature from RFC 2246 [RFC2246] are extended to include NSS.

NOTE: The operation digitally-signed is defined in RFC 2246 [RFC2246] section 4.7 and specifies that the length is represented as an opaque vector <0..2^16-1>, where the length is specified by the signature algorithm (e.g. NSS) and key.

```
enum { nss } SignatureAlgorithm;

select (SignatureAlgorithm) {
        case nss: digitally-signed struct {
                opaque sha_hash[20];
        };
} Signature;
```

sha_hash
This is the SHA-1 hash of all of the preceding handshake messages. The SHA-1 algorithm is defined in FIPS 180-1 [FIPS180-1].

```
struct {
        Signature signature;
} CertificateVerify;
```

5. Cryptographic Computations and Encoding

This section specifies the exact cryptographic computations and encoding of NTRU and NSS data structures that are needed in order to implement TLS with the NTRU and NSS cipher suites. When not explicitly stated, all cryptographic encoding and computations SHALL be as specified in RFC 2246 [RFC2246]. Note that in particular, RSA signature verification on certificates SHALL be computed as specified in RFC 2246 [RFC2246].

The following cryptographic computations and descriptions of their use in TLS are defined in this section.

Computation	Description
NSS Digital Signing	The operation of computing the NSS digital signature on the specified hash value and encoding the signature as an opaque object.
NSS Signature Verification	The operation of verifying an NSS digital signature on a certificate or on the handshake messages.
NTRU Encryption	The operation of computing the NTRU encryption of the pre-master secret and encoding the encrypted data as an opaque object.
NTRU Decryption	The operation of decrypting the pre-master secret.

5.1 NSS Digital Signing

When this operation is performed:

Whenever client authentication is performed, the client generates an NSS digital signature on all of the preceding handshake messages and places this signature in the CertificateVerify message in the Client Response.

NSS digital signatures on certificates are outside of the scope of TLS, however, it is assumed that the NSS certificate signatures are performed as specified by the SVSSA signature scheme as defined in EESS #1 [EESS#1] and encoded in the certificate according to EESS #1 [EESS#1].

Operation:

For all NSS digital signature operations in this document, the signature SHALL be performed as specified by the SVSSA signature scheme as defined in EESS #1 [EESS#1]. The parameter values SHALL be included in the client certificate and SHALL be interpreted according to EESS #1 [EESS#1]. For the ClientVerify message, the input to the signature scheme SHALL be the concatenation of all of the previous handshake messages and the hash function for creating the hash of the message SHALL be SHA-1 [FIPS180-1] and SHALL NOT be MD5.

Encoding:
The structure of NSS signatures, written in TLS as the function digitally-signed, SHALL be encoded as a type 1 vector as defined in EESS #1 [EESS#1]. (This is essentially the polynomial written as a byte string of coefficients ordered from lowest degree to highest, with each byte representing a single coefficient.)

5.2 NSS Signature Verification

When this operation is performed:
The client performs NSS signature verification to verify the server certificate in all NTRU cipher suites. The server performs NSS signature verification in all client authenticated handshakes to verify the client certificate and to verify the ClientVerify message.

Operation:
For all NSS verification operations in this document, the verification SHALL be performed as specified by the SVSSA signature scheme as defined in EESS #1 [EESS#1]. The server SHALL verify that the parameter values be included in the client certificate and interpret them according to EESS #1 [EESS#1]. For the ClientVerify message verification, the input to the verification process SHALL be the concatenation of all of the previous handshake messages and the hash function for creating the hash of the message SHALL be SHA-1 [FIPS180-1] and SHALL NOT be MD5. If all of the above checks pass, the server SHALL accept the signature as valid, otherwise the server SHALL reject the signature as invalid.

5.3 NTRU Encryption

When this operation is performed:
The client performs NTRU encryption on the pre-master secret in all cipher suites defined in this document. The encrypted pre-master secret is included in the ClientKeyExchange message in the Client Response.

Operation:
For all NTRU encryption operations in this document, the encryption SHALL be performed as specified by the SVES encryption scheme as defined in EESS #1 [EESS#1]. The parameter values SHALL be included in the server certificate

(or ServerKeyExchange message) and SHALL be interpreted according to EESS #1 [EESS#1]. For the ClientKeyExchange message, the input to the encryption scheme SHALL be the pre-master secret as the leftmost (first) bytes, padded on the right (end) by any byte string that makes the total length equal to the input length of the encryption function (e.g. for NTRU 251, the 20-byte pre-master secret will be padded with 1 byte on the right to obtain a 21-byte input to the encryption function). The padding SHOULD consist of all '0' bytes.

Encoding:

The structure of NTRU encryptions, written in TLS as the function public-key-encrypted, SHALL be encoded as a type 1 vector as defined in EESS #1 [EESS#1]. (This is essentially the polynomial written as a byte string of coefficients ordered from lowest degree to highest (left to right), with each byte representing a single coefficient.)

5.4 NTRU Decryption

When this operation is performed:

The server performs NTRU decryption on the encrypted pre-master secret in all cipher suites defined in this document. The encrypted pre-master secret is included in the ClientKeyExchange message in the Client Response.

Operation:

For all NTRU decryption operations in this document, the decryption SHALL be performed as specified by the SVES encryption scheme as defined in EESS #1 [EESS#1]. The server SHALL use the parameter values that are included in the server certificate and interpret them according to EESS #1 [EESS#1]. For the decryption of the pre-master secret, the input to the decryption process SHALL be the encrypted pre-master secret included in the ClientKeyExchange message. The output of the NTRU decryption operation SHALL be truncated to obtain the pre-master secret by taking the leftmost (first) n bytes of the plaintext, where n is the length of the pre-master secret.

6. Cipher Suites

The table below defines the cipher suites specified in this document. They are interpreted according to their names in the same manner as in RFC 2246 [RFC2246]. The key agreement methods

specified in this standard are NTRU_NSS and NTRU_RSA. The cipher
suite numbers are subject to change depending on the numbering
conventions and the numbers that are already used.

```
CipherSuite TLS_NTRU_NSS_WITH_RC4_128_SHA           = { 0x00, 0x61 }
CipherSuite TLS_NTRU_NSS_WITH_3DES_EDE_CBC_SHA      = { 0x00, 0x62 }
CipherSuite TLS_NTRU_NSS_WITH_AES_128_CBC_SHA       = { 0x00, 0x63 }
CipherSuite TLS_NTRU_NSS_WITH_AES_256_CBC_SHA       = { 0x00, 0x64 }
CipherSuite TLS_NTRU_RSA_WITH_RC4_128_SHA           = { 0x00, 0x65 }
CipherSuite TLS_NTRU_RSA_WITH_3DES_EDE_CBC_SHA      = { 0x00, 0x66 }
CipherSuite TLS_NTRU_RSA_WITH_AES_128_CBC_SHA       = { 0x00, 0x67 }
CipherSuite TLS_NTRU_RSA_WITH_AES_256_CBC_SHA       = { 0x00, 0x68 }
```

Ciphers other than AES ciphers and hash algorithms are specified in
RFC 2246 [RFC2246]. AES ciphers are specified in [TLS-AES].

Implementations supporting NTRU cipher suites SHOULD support the
following cipher suite. Implementations MAY support any of the
other cipher suites.

TLS_NTRU_NSS_WITH_AES_128_CBC_SHA

7. Security Considerations

This document is entirely concerned with security mechanisms. It is
based on the TLS specification [RFC 2246], IEEE P1363.1 [P1363.1]
and EESS #1 [EESS#1] and the appropriate security considerations
from those documents apply.

8. Intellectual Property Rights

NTRU Cryptosystems, Inc. has been granted U.S. Patent No. 6,081,597,
which covers aspects of the NTRU public-key encryption scheme, and
has applied for a patent (or patents) that covers the NSS public-key
signature scheme. In addition, NTRU Cryptosystems may have applied
for additional patent coverage on implementation techniques related

to the use of NTRU or NSS. This and any additional patent
information will be sent to the IETF.

The IETF takes no position regarding the validity or scope of any
intellectual property or other rights that might be claimed to
pertain to the implementation or use of the technology described in
this document or the extent to which any license under such rights
might or might not be available; neither does it represent that it

9. References

[EESS#1] Efficient Embedded Security Standards (EESS) #1: Implementation Aspects of NTRU and NSS, Draft Version 2, May 18, 2001, Consortium for Efficient Embedded Security Standards, Available at http://www.ceesstandards.org.

[FIPS180-1] FIPS PUB 180-1, Secure Hash Standard, Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/National Institute of Standards and Technology, National Technical Information Service, Springfield, Virginia, April 17, 1995 (supersedes FIPS PUB 180). Available at http://www.itl.nist.gov/div897/pubs/fip180-1.htm.

[P1363.1] IEEE Draft Standard P1363.1 D2: IEEE Standard Specifications for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices, Draft 2, May 2001, Available at http://grouper.ieee.org/groups/1363.

[PKCS#1] RSA Laboratories. RSA Encryption Standard. Version 2.0, October 1, 1998.

[RFC2026] S. Bradner, "The Internet Standards Process", IETF RFC 2026, October 1996

[RFC2119] S. Bradner, "Key Words for Use in RFCs to Indicate Requirement Levels", IETF RFC 2119, March 1997

[RFC2246] T. Dierks and C. Allen, "The TLS Protocol - Version 1.0," IETF RFC 2246, January 1999

[RFC2459] R. Housley, W. Ford, W. Polk and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", IETF RFC 2459, January 1999

[TLS-AES] P. Chown, "AES Ciphersuites for TLS", draft-ietf-tls-ciphersuite-03.txt, January 22, 2001

Authors' Addresses

```
Ari Singer
NTRU
5 Burlington Woods            Phone:  1-781-418-2515
Burlington, MA  01803, USA    Email:  asinger@ntru.com
```