Internet Engineering Task Force H. Wang, Ed.
Internet-Draft Y. Yang
Intended status: Standards Track X. Kang
Expires: June 29, 2019 Huawei International Pte. Ltd.
December 26, 2018

Using Identity as Raw Public Key in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)
draft-wang-tls-raw-public-key-with-ibc-03

Abstract

This document specifies the use of identity as a raw public key in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). The TLS protocol procedures are kept unchanged, but cipher suites are extended to support Identity-based signature (IBS). The example OID tables in the [RFC 7250] are expanded with OIDs specific to IBS algorithms.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 29, 2019.

Copyright Notice

Table of Contents

1. Introduction

DISCLAIMER: This is a personal draft and has not yet seen significant security analysis.

Traditionally, TLS client and server exchange public keys endorsed by PKIX [PKIX] certificates. It is considered complicated and may cause security weaknesses with the use of PKIX certificates Defeating-SSL [Defeating-SSL]. To simplify certificates exchange, using RAW public key with TLS/DTLS has been spcified in [RFC 7250]. That is, instead of transmitting a full certificate or a certificate chain in the TLS messages, only public keys are exchanged between client and server. However, using RAW public key requires out-of-band mechanisms to bind the public key to the entity presenting the key.

Recently, 3GPP has adopted the EAP authentication framework for 5G and EAP-TLS is considered as one of the candidate authentication methods for private networks, especially for networks with a large number of IoT devices. For IoT networks, TLS/DTLS with RAW public key is particularly attractive, but binding identities with public keys might be challenging. The cost to maintain a large table for identity and public key mapping at server side incurs additional maintenance cost. e.g. devices have to pre-register to the server.

To simplify the binding between the public key and the entity presenting the public key, a better way could be using Identity-Based Cryptography(IBC), such as ECCSI public key specified in [RFC 6507], for authentication. Different from X.509 certificates and raw public keys, a public key in IBC takes the form of the entity's identity. This eliminates the necessity of binding between a public key and the entity presenting the public key.

The concept of IBC was first proposed by Adi Shamir in 1984. As a special class of public key cryptography, IBC uses a user's identity as public key, avoiding the hassle of public key certification in public key cryptosystems. IBC broadly includes IBE (Identity-based Encryption) and IBS (Identity-based Signature). For an IBC system to work, there exists a trusted third party, PKG (private key generator) responsible for issuing private keys to the users. In particular, the PKG has in possession a pair of Master Public Key and Master Secret Key; a private key is generated based on the user's identity by using the Master Secret key, while the Master Public key is used together with the user's identities for encryption (in case of IBE) and signature verification (in case of IBS).

A number of IBE and IBS algorithms have been standardized by different standardization bodies, such as IETF, IEEE, ISO/IEC, etc. For example, IETF has spcified several RFCs such as [RFC 5091], [RFC 6507] and [RFC6508] for both IBE and IBS algorithms. ISO/JTC and IEEE also have a few standards on IBC algorithms.

RFC 7250 has specified the use of raw public key with TLS/DTLS handshake. However, supporting of IBS algorithms has not been included therein. Since IBS algorithms are efficient in public key transmission and also eliminate the binding between public keys and identities, in this document, an amendment to RFC 7250 is added for supporting IBS algorithms.

IBS algorithm exempts client and server from public key certification and identity binding by checking an entity's signatures and its identity against the master public key of its PKG. With an IBS algorithm, a PKG generates private keys for entities based on their identities. Global parameters such as PKG's Master Public Key (MPK) need be provisioned to both client and server. These parameters are not user specific, but PKG specific.

For a client, PKG specific parameters can be provisioned at the time PKG provisions the private key to the client. For the server, how to get the PKG specific parameters provisioned is out of the scope of this document, and it is deployment dependent.

The document is organized as follows: Section 3 defines the data structure required when identity is used as raw public key, and a list of OIDs for IBS algorithms. Section 4 defines the cipher suites required to support IBS algorithm over TLS/DTLS. Section 5 explains how client and server authenticate each other when using identity as raw public key. Section 6 gives examples for using identity as raw public key over TLS/DTLS handshake procedure. Section 7 discusses the security considerations.

2. Terms

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].

3. Extension of RAW Public Key to IBC-based Public Key

To support the negotiation of using raw public between client and server, a new Certificate structure is defined in RFC 7250. It is used by the client and server in the hello messages to indicate the types of certificates supported by each side.

When RawPublicKey type is selected for authentication, a data structure, subjectPublicKeyInfo, is used to carry the raw public key and its cryptographic algorithm. Within the subjectPublicKeyInfo structure, two fields, algorithm and subjectPublicKey, are defined. The algorithm is a data structure specifies the cryptographic algorithm used with raw public key, which is represented by an object Identifiers (OID); and the parameters field provides necessary parameters associated with the algorithm. The subjectPublicKey field within the subjectPublicKeyInfo carry the raw public itself.

```
    subjectPublicKeyInfo  ::=  SEQUENCE  {
             algorithm                AlgorithmIdentifier,
             subjectPublicKey         BIT STRING
        }

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

Figure 1: SubjectECCSIPublicKeyInfo ASN.1 Structure

When using an IBS algorithm, an identity is used as the raw public key, which can be converted to an OCTET string and put into the subjectPublicKey field. The algorithm field in Algorithm Identifier structure is the object identifier of the IBS algorithm used. Beside

that, it is necessary to tell the peer the set of global parameters used by the signer. The information can be carried in the payload of the parameters field in AlgorithmIdentifier. However, the global public parameters can be large. Instead of carrying the full set of global public parameters of a PKG, an URI or IRI of a PKG is put in the parameter field. The URI/IRI allows the peer know which set of public parameters shall be used to verify the signature.

The structure to carry the PKGInfo is specified in Figure 2:

```
opaque DistinguishedName<1..2^16-1>;
struct {
   DistinguishedName pkg_addr<1..2^16-1>;
} PKGInfo;
```

Figure 2: PKGInfo ANSI.1 Structure

The pkg_addr field is a string of an URI or IRI of a PKG, indicating the PKG where public parameters of the IBC algorithm identified by the OBJECT IDENTIFIER are available.

When using an IBS algorithm, an identity is used as raw public key, which can be converted to an OCTET string. Therefore, the Certificate and subjectPublicKey structure can be reused without changes.

To use a signature algorithm with TLS, OID for the signature need to be provided. However, no OID for for the IBS algorithm specified in RFC 6507 has been given. Thus, OID should be allocated to the ECCSI algorithm specified in [RFC 6507] before it can be used for TLS. The following table shows the basic information needed for the ECCSI signature algorithm to be used for TLS.

Key Type	Document	OID
Elliptic Curve-Based Signatureless For Identitiy-based Encryption (ECCSI)	Section 5.2 in RFC 6507	1.3.6.1.5.x (need to apply)

Table 1: Algorithm Object Identifiers

4. New Key Exchange Algorithms and Cipher Suites

To support using identity as raw public key, new key exchange algorithms corresponding to the IBS algorithms need to be defined. The existing key exchange algorithms making use of ephemeral DH are

extended to support IBS algorithms. Considering the performance and the compatibility with the use of ECDSA in TLS (see [RFC 4492]), this specification proposes to support the IBS algorithm, ECCSI, defined in [RFC 6507]. As a reult, the table below summarizes the new key exchange algorithm, which mimics DHE_DSS, ECDHE_ECDSA, respectively (see [RFC 5246] and [RFC 4492]).

Key Exchange Algorithm	Description
ECDHE_ECCSI	Ephemeral ECDH with ECCSI signatures

Table 2: Algorithm Object Identifiers

To include the new key exhange algorithm, the data structure KeyExchangeAlgorithm need to be expanded with a new value ecdhe_eccsi as follows:

```
enum {
        ecdhe_eccsi
} KeyExchangeAlgorithm;
```

Figure 3: Include ecdhe_eccsi in KeyExchangeAlgorithm

Note: The specification of ECDHE_ECCSI can follow ECHDE_ECDSA by substituting ECDSA with ECCSI[RFC6507]. The detailed specification will be provided in the future

Note: Other key exchange algorithms with other IBS algorithms may be added in the future.

Accordingly, below defines new cipher suites that use above new key exchange algorithms:

CipherSuite TLS_ECDHE_ECCSI_WITH_AES_128_CBC_SHA256 = { TBD, TBD }

CipherSuite TLS_ECDHE_ECCSI_WITH_AES_256_CBC_SHA256 = { TBD, TBD }

5. TLS Client and Server Handshake Behavior

When IBS is used as RAW public for TLS, signature and hash algorithms are negotiated during the handshake.

The handshake between the TLS client and server follows the procedures defined in [RFC 7250], but with the support of the new key exchange algorithm and cipher suites specific to the IBS algorithms. The high-level message exchange in the following figure shows TLS

handshake using raw public keys, where the client_certificate_type and server_certificate_type extensions added to the client and server hello messages (see Section 4 of [RFC 7250]).

```
    client_hello,
    client_certificate_type,
    server_certificate_type   ->

                              <-  server_hello,
                                  client_certificate_type,
                                  server_certificate_type,
                                  certificate,
                                  server_key_exchange,
                                  certificate_request,
                                  server_hello_done
    certificate,
    client_key_exchange,
    certificate_verify,
    change_cipher_spec,
    finished                  ->

                              <- change_cipher_spec,
                                 finished

Application Data        <------->     Application Data
```

Figure 4: Basic Raw Public Key TLS Exchange

The client hello messages tells the server the types of certificate or raw public key supported by the client, and also the certificate types that client expects to receive from server. When raw public with IBS algorithm from server is supported by the client, the client includes desired IBS cipher suites in the client hello message based on the order of client preference.

After receiving the client hello message, server determines the client and server certificate types for handshakes. When the selected certificate type is RAW public key and IBS is the chosen signature algorithm, server uses the SubjectPublicKeyInfo structure to carry the raw public key, OID for IBS algorithm and URI/IRI for global public parameters. With these information, the client knows the signature algorithm and the public parameters that should be used to verify the signature. The format of signature in the server_key_exhange message is defined in the corresponding specification. For example, when ECCSI is used, the format of signature is defined in [RFC 6507].

When sever specifies that RAW public key should be used by client to authenticate with server, the client_certificate_type in the server hello is set to RawPublicKey. Besides that, the server also sends Certificate Request, indicating that client should use some specific signature and hash algorithms. When IBS is chosen as raw public key signature algorithm, the server need to indicate the supporting of IBS signature algorithms in the CertificateRequest.

The Certificate Request is a structure defined in TLS1.2 as follows :

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

Figure 5: ANSI.1 structure for CertificateRequest

To support IBS algorithms, values of the ClientCertificateType and SignatureAlgorithm need to be amended. To support ECCSI defined in IETF RFC 6507, eccsi_sign type is added to ClientCertificateType as follows:

```
     enum {
         eccsi_sign, (255)
      } ClientCertificateType;
```

Figure 6: Value of ECCSI in ClientCertificateType

eccsi_sign: the subsequent client certificate is a raw public key certificate containing an ECCSI public key.

Moreover, an eccsi type needs to be added to the SignatureAlgorithm structure, which is in turn used in the SignatureAndHashAlgorithm structure:

```
     enum {
             eccsi, (255)
     } SignatureAlgorithm.
```

Figure 7: Value of ECCSI for SignatureAlgorithm

No new hash function type is required. RFC 6507 does not specify any specific hash function to use for ECCSI. As a result, SHA256 suffices to instantiate ECCSI.

To support more IBS signature algorithms, additional values can be added to the ClientCertificateType and SignatureAlgorithm in the future.

If raw public key is selected by server for client authentication, the client checks the CertificateRequest received for signature algorithms. If client wants to use an IBS algorithm for signature, then the signature algorithm it intended to use must be in the list of supported signature algorithms by the server. Assume the IBS algorithm supported by the client is in the list, then the client specifies the IBS signature algorithm and PKG information with SubjectPublicKeyInfo structure in the certificate structure and provide signatures in the certificate verify message. The format of signature in the certificate_verify message is defined in the corresponding specification.

The server verifies the signature based on the algorithm and PKG parameters specified by the messages from client.

6. Examples

In the following, examples of handshake exchange using IBS algorithm under RawPublicKey are illustrated.

6.1. TLS Client and Server Use IBS algorithm

In this example, both the TLS client and server use ECCSI for authentication, and they are restricted in that they can only process ECCSI keys. As a result, the TLS client sets both the server_certificate_type extension and the client_certificate_type extension to be raw public key; in addition, the client sets the ciphersuites in the client hello message to be TLS_ECDHE_ECCSI_WITH_AES_256_CBC_SHA256.

When the TLS server receives the client hello, it processes the message. Since it has an ECCSI raw public key from the PKG, it indicates in (2) that it agrees to use ECCSI and provided an ECCSI key by placing the SubjectPublicKeyInfo structure into the Certificate payload back to the client (3), including the OID and URI/IRI of global public key parameters. The client_certificate_type in (4) indicates that the TLS server accepts raw public key. The TLS server demands client authentication, and therefore includes a certificate_request (5) for ECCSI raw public. The client, which has an ECCSI key, returns its ECCSI certificate in the Certificate payload to the server (6).

```
client_hello,
cipher_suites=(TLS_ECDHE_ECCSI_WITH_AES_256_CBC_SHA256) // (1)
client_certificate_type=(RawPublicKey) // (1)
server_certificate_type=(RawPublicKey) // (1)
                         ->
                         <- server_hello,
                            server_certificate_type= RawPublicKey // (2)
                            certificate=((1.3.6.1.5.x,
                                          pkgx.org/1.html), KEY) // (3)
                            client_certificate_type=RawPublicKey // (4)
                            certificate_request= (eccsi_sign, (eccsi,
                                                 SHA256)), // (5)
                             server_key_exchange,
                             server_hello_done

certificate=(
   (1.3.6.1.5.x,
   pkgx.org/1.html),
   KEY), // (6)
client_key_exchange,
change_cipher_spec,
finished                   ->

                         <- change_cipher_spec,
                            finished

Application Data        <------->     Application Data
```

Figure 8: Basic Raw Public Key TLS Exchange

6.2. Combined Usage of Raw Public Keys and X.509 Certificates

This example combines the uses of an ECCSI key and an X.509 certificate. The TLS client uses an ECCSI key for client authentication, and the TLS server provides an X.509 certificate for server authentication.

The exchange starts with the client indicating its ability to process a raw public key, or an X.509 certificate, if provided by the server. It prefers a raw public key, since TLS_ECDHE_ECSSI_WITH_AES_256_CBC_SHA256 proceeds TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA256 in the cipher_suites payload, and the RawPublicKey value precedes the other value in the server_certificate_type payload. Furthermore, the client indicates that it has a raw public key for client-side authentication.

The server chooses to provide its X.509 certificate in (3) and indicates that choice in (2). For client authentication, the server

indicates in (4) that it has selected the raw public key format and requests an ECCSI certificate from the client in (4) and (5). The TLS client provides an ECSSI certificate in (6) after receiving and processing the TLS server hello message.

```
client_hello,
cipher_suites=(
TLS_ECDHE_ECSSI_WITH_AES_256_CBC_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA256), // (1)
client_certificate_type=(RawPublicKey), // (1)
server_certificate_type=
                    (RawPublicKey, X.509) // (1)
                       ->
                       <-  server_hello,
                           server_certificate_type=X.509, // (2)
                           certificate, // (3)
                           client_certificate_type = RawPublicKey // (4)
                           certificate_request= (eccsi_sign, (eccsi,
                                                SHA256)), // (5)
                              server_key_exchange,
                              server_hello_done
certificate=(KEY,
   (1.3.6.1.5.x,
   pkgx.org/1.html)), // (6)
client_key_exchange,
change_cipher_spec,
finished                    ->

                            <- change_cipher_spec,
                               finished

Application Data        <------->     Application Data
```

Figure 9: Basic Raw Public Key TLS Exchange

7. Security Considerations

Using IBS-enabled raw public key in TLS/DTLS will not change the information flows of TLS, so the security of the resulting protocol rests on the security of the used IBS algorithms. The example IBS algorithms mentioned above are all standardized and open, and thus the security of these algorithms is supposed to have gone through wide scrutinization.

8. IANA Considerations

Existing IANA references have not been updated yet to point to this document.

With TLS protocol, an OID is required to identify the signature algorithm used by client or server. For example, the RSA signature algorithm used in the TLS is identified by 1.2.840.113549.1.1 (Page 5 of RFC 7250). However, the ECCSI signature algorithm specified in the RFC 6507 and used in this document has not been assigned an OID yet. Therefore, an OID should be assigned to the ECCSI signature algorithm.

The following TLS registries shall be updated also:

- TLS Cipher Suite Registry: Future values with the first byte in the range 0-191 (decimal) inclusive are assigned via Standards Action [RFC2434]. Values with the first byte in the range 192-254 (decimal) are assigned via Specification Required [RFC2434]. Values with the first byte 255 (decimal) are reserved for Private Use [RFC2434].

- TLS KeyExchangeAlgorithm Registry: Future values are allocated via Standards Action [RFC2434]

- TLS ClientCertificateType Registry: Future values are allocated via Standards Action [RFC2434]

- TLS SignatureAlgorithm Registry: Future values are allocated via Standards Action [RFC2434]

9. Acknowledgements

10. References

10.1. Normative References

[PKIX] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List(CRL) Profile", June 2008.

[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>.

[RFC2434] Narten, D., Alvestrand, H., "Guidelines for Writing an IANA Consideration Section in RFCs", October 1998.

[RFC5091] Boyen, X. and L. Martin, "Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems", RFC 5091, DOI 10.17487/RFC5091, December 2007, <https://www.rfc-editor.org/info/rfc5091>.

[RFC6507] Groves, M., "Elliptic Curve-Based Certificateless Signatures for Identity-Based Encryption (ECCSI)", RFC 6507, DOI 10.17487/RFC6507, February 2012, <https://www.rfc-editor.org/info/rfc6507>.

[RFC6508] Groves, M., "Sakai-Kasahara Key Encryption (SAKKE)", RFC 6508, DOI 10.17487/RFC6508, February 2012, <https://www.rfc-editor.org/info/rfc6508>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

10.2. Informative References

[Defeating-SSL] Marlinspike, M.,, "New Tricks for Defeating SSL in Practice", Feb 2009, <http://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf>.

Appendix A. Examples

Authors' Addresses

Haiguang Wang (editor)
Huawei International Pte. Ltd.
11 North Buona Vista Dr, #17-08
Singapore 138589
SG

Phone: +65 6825 4200
Email: wang.haiguang1@huawei.com

Yanjiang Yang
Huawei International Pte. Ltd.
11 North Buona Vista Dr, #17-08
Singapore 138589
SG

Phone: +65 6825 4200
Email: yang.yanjiang@huawei.com

Xin Kang
Huawei International Pte. Ltd.
11 North Buona Vista Dr, #17-08
Singapore 138589
SG

Phone: +65 6825 4200
Email: xin.kang@huawei.com