Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2019

H. Wang, Ed.
Y. Yang
X. Kang
Huawei International Pte. Ltd.

March 5, 2019

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-08

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 signature algorithms are extended to support Identity-based signature (IBS). A typical Identity-based signature algorithm, the ECCSI signature algorithm defined in RFC 6507, is supported in the current version.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

DISCLAIMER: This is a personal draft and a limited security analysis is provided.

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] and has been included in the TLS 1.3[RFC 8446]. With RAW public key, 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). Another name of PKG is Key Management System (KMS), which is also used in some IBC system. In this document, the terms of PKG and KMS are interchangable.

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 is added for supporting IBS algorithms as raw public key.

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. 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", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals.

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: SubjectPublicKeyInfo ASN.1 Structure

With IBS algorithm, an identity is used as the raw public key, which can be converted to an BIT string and put into the subjectPublicKey field. The algorithm field in AlgorithmIdentifier structure is the object identifier of the IBS algorithm used. Specifically, for the ECCSI signature algorithm supported in this draft, the OBJECT IDENTIFIER is described with following data structure:

```
sa-eccsiWithSHA256 SIGNATURE-ALGORITHM ::= {
    IDENTIFIER id-alg-eccsi-with-sha256
    VALUE ECCSI-Sig-Value PARAMS TYPE NULL ARE absent
    HASHES { mda-sha256 }
    SMIME-CAPS { IDENTIFIED BY id-alg-eccsi-with-sha256 }
}
```

Figure 2: ECCSI Signature Algorithm ANSI.1 Structure

Note, in a real implementation, only OID part will be transmitted over the TLS negotiation protoocols.

Beside OID, 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. In the following, a data structure for carrying ECCSI-based parameters are defined. For other IBS algorithm, it can be defined in the future. If client and server are sure that each of them knows the global parameters, this data structure can be omitted from transmission.

The structure to carry the ECCSI-based global parameters is specified in followng Figure :

```
ECCSIPublicParameters ::= SEQUENCE {
    version   INTEGER { v2(2) },
    curve     OBJECT IDENTIFIER,
    hashfcn   OBJECT IDENTIFIER,
    pointP    POINT,
    pointPpub POINT
}
```

Figure 3: ECCSI Global Parameters ANSI.1 Structure

With above data structure, pointP shall be G in RFC 6507 and pointPpub shall be KPAK in RFC 6507. The POINT structure specifies a point on an elleptic curve and is defined as follows:

```
POINT ::= SEQUENCE {
    x INTEGER,
    y INTEGER
}
```

Figure 4: POINT Structure ANSI.1 Structure

To support IBS algorithm over TLS protocol, a data structure for signature value need to be defined. A data structure for ECCSI is defined as follows(based RFC 6507):

```
ECCSI-Sig-Value ::= SEQUENCE {
    r INTEGER,
    s INTEGER,
    PVT OCTET STRING
}
```

Figure 5: ECCSI Signature Value ANSI.1 Structure

where PVT (as defined in RFC 6507) is encoded as 0x04 || x-coordinate of [v]G || y-coordinate of [v]G.

To use a signature algorithm with TLS, OID for the signature algorithm need be provided. For ECCSI algorithm, an OID has been assigned by IANA recently. 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	1.3.6.1.5.5.7.6.29

Table 1: Algorithm Object Identifiers

4. New Signature Algorithms for IBS

To using identity as raw public key, new signature algorithms corresponding to the IBS need to be defined. With TLS 1.3, the value for signature algorithm is defined in the SignatureScheme. This document specifies how to support ECCSI algorithm. As a reult, the SignatureScheme data structure has to be amended by including the ECCSI algorithm.

```
       enum {
               ...

           /* IBS ECCSI signature algorithm */
               eccsi_sha256 (TBD),

           /* Reserved Code Points */
           private_use (0xFE00..0xFFFF),
           (0xFFFF)
       } SignatureScheme;
```

Figure 6: Include ecdhe_eccsi in KeyExchangeAlgorithm

Note: The signature algorithm of eccsi_sha256 is defined in RFC6507.

Note: Other IBS signature algorithms can be added in the future.

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 8446], but with the support of the new signature algorithms 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,
        +key_share
        +signature_algorithms
        client_certificate_type,
        server_certificate_type   ->

                                <-  server_hello,
                                    + key_share
                                    {EncryptyedExtensions}
                                    {client_certificate_type}
                                    {server_certificate_type}
                                    {Certificate}
                                    {CertificateVerify}
                                    {CertificateRequest}
                                    {Finished}
                                    [Applicaiton Data]
       {Certificate}
       {CertificateVerify}
       {Finished}          -------->
       [Application Data]  <-------> [Application Data]
```

Figure 7: 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 signature algorithm 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. If ECCSI is selected, the ECCSIPublicParameters can be used to carry 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 signature value is in the CertificateVerify message and the format of signature value should be specified by each IBS algorithm. In this document, an ECCSI-Sig-Value data strcuture for ECCSI signature algorithm is defined based on the specification of 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 signature algorithm, the server need to indicate the required IBS signature algorithms in the signature_algorithm extension within the CertificateRequest.

After receiving the server hello, the client checks the CertificateRequest 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 specified by the server. Assume the IBS algorithm supported by the client is in the list, then the client response with 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 CertificateVerify message should be sepcified by each individual signature algorithm. If ECCSI is chosen, an ECCSI-Sig-Value data strcuture is used to carry the signature.

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 signature algorithm. As a result, the TLS client sets both the server_certificate_type and the client_certificate_type extensions to be raw public key; in addition, the client sets the signature algorithm in the client hello message to be eccsi_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, the identity of server, ServerID, which is the public key of server also, and PKG public parameters (ECCSIPublicParameters). 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), which requires the client to use eccsi_sha256 for signature. A signature value based on the eccsi_sha256 algorithm is carried in the CertificateVerify (6). The client, which has an ECCSI key, returns its ECCSI public key in the Certificate payload to the server (7), which includes an OID for the ECCSI signature algorithm, the PKGInfo for KMS parameters, and identity of client, ClientID, which is the public key of client also. The client also includes a signature value, ECCSI-Sig-Value, in the CertificateVerify (8) message.

When client/server receive PKG public parameters from peer, it should decide whether these parameters are acceptable or not. An exmaple way to make decision is that a whitelist of acceptable PKG public parameters are stored locally at client/server. They can simply make a decision based on the white list stored locally.

```
 client_hello,
  +key_share // (1)
  signature_algorithm = (eccsi_sha256)   // (1)
  client_certificate_type=(RawPublicKey) // (1)
  server_certificate_type=(RawPublicKey) // (1)
                        ->
                        <- server_hello,
                           + key_share
                           { server_certificate_type = RawPublicKey} // (2)
                           {certificate=((1.3.6.1.5.5.7.6.29,
                            ECCSIPublicParameters), serverID)} //(3)
                           {client_certificate_type = RawPublicKey // (4)
                           {certificate_request = (eccsi_sha256)} //(5)
                           {CertificateVerify = {ECCSI-Sig-Value} // (6)
                           {Finishaed}

{Certificate=(
 (1.3.6.1.5.5.7.6.29,
 ECCSIPublicParameters),
 ClientID)} // (7)
{CertificatVerify = (ECCSI-Sig-Value)} //(8)
{Finished }
[Applicateion Data] ---->
[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 eccsi_sha256 proceeds ecdsa_secp256r1_sha256 in the signature_algorithm payload, and the RawPublicKey value precedes the other value in the server_certificate_type payload. Furthermore, the client indicates that it has a ECCSI-based raw public key for client-side authentication. Client also indicate it supports server using either ECCSI or ecdsa for the certificate signature. This further indicates that server can also use ecdsa_secp256r1_sha256 to sign the message.

With the received client_hello, 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) and signature value after receiving and processing the TLS server hello message.

```
  client_hello,
   +key_share
   signature_algorithms =(eccsi_sha256)   // (1)
   signature_algorithms_cert =(eccsi_sha256,
    ecdsa_secp256r1_sha256)   // (1)
   {client_certificate_type=
   (RawPublicKey)}             // (1)
   {server_certificate_type=
   (RawPublicKey, X.509)       // (1)
                       ->
                       <-  server_hello,
                           +key_share
                           {server_certificate_type=X.509} // (2)
                           {Certificate = (x.509 certificate)} // (3)
                           {client_certificate_type = (RawPublicKey)} // (4)
                           {CertificateRequest} = (eccsi_sha256)} // (5)
                           {CertificateVerify}
                           {Finished}
  certificate=(
  (1.3.6.1.5.5.7.6.29,
  ECCSIPublicParameters),
  ClientID), // (6)
 {CertificatVerify =
  (ECCSI-Sig-Value)} //(7)
 { Finished }
 [Applicateion Data] ---->
 [Application Data]  <---> [Application Data]
```

Figure 9: Basic Raw Public Key TLS Exchange

7. Security Considerations

Using ECCSI-based raw public key in TLS/DTLS does not change the message flows of TLS, hence, for the most part, the security considerations involved in using the Transport Layer Security protocol with raw public key also apply here. The additional security of the resulting protocol rests on the security of the used ECCSI algorithms.

ECCSI signature algorithm has been standardized for ten years and has been adopted in real application. However, we would like to point out the difference between ECCSI and existing raw public key: the private key of ECCSI used for signature generation is generated by the Key Management System (KMS), while the private key for the existing raw public key is generated locally. Therefore, ECCSI mechanism may face a security risk of private key disclosure due to improper management of KMS system. The user of ECCSI shall be aware the above risk and a stronger key management system shall be adopted by KMS system when using ECCSI.

8. IANA Considerations

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

IANA is asked to assign an OID for ECCSI signature algorithm specified in the [RFC6507], which is used by this document. The required OID should be assigned under the registry of SMI Security for PKIX Algorithms (1.3.6.1.5.5.7.6) with following name:

- id-alg-eccsi-with-sha256.

- an OID has been assigned by IANA to ECCSI as 1.3.6.1.5.5.7.6.29.

The following TLS registries shall be updated also:

- Signature Scheme Registry: signature algorithm for ECCSI, eccsi_with_sha256, are required to be reserved.

9. Acknowledgements

10. References

10.1. Normative References

[PKIX] "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] "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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8216] Pantos, R., Ed. and W. May, "HTTP Live Streaming", RFC 8216, DOI 10.17487/RFC8216, August 2017, <https://www.rfc-editor.org/info/rfc8216>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

10.2. Informative References

[Defeating-SSL] "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