Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: October 13, 2021

H. Wang, Ed.
Y. Yang
X. Kang
Huawei International Pte. Ltd.
Z. Cheng
Shenzhen Olym Info. Security Tech. Ltd.
M. Chen
China Mobile

April 11, 2021

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-14

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 few Identity-based signature algorithms from different standard organizations are supported in the current version.

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 October 13, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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]. To simplify certificates exchange, using RAW public key with TLS/DTLS has been specified in [RFC 7250] and has been included in the TLS 1.3 [RFC 8446]. 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 verify the purported public key to the claimed entity.

Recently, 3GPP has adopted the EAP as a unified uthentication for 5G and included both EAP-APA' and EAP-TLS as authentication methods. It is specified in the 5G specification that EAP-TLS can be used for private networks, especially for networks with a large number of IoT devices [TS33.501]. For IoT networks, EAP-TLS 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 cost, e.g. devices have to pre-register to the server.

To simplify the binding between the public key and the entity, 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 existing 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, private key generator (PKG), which is responsible for issuing private keys to the users. A 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 interchangeable.

A number of IBE and IBS algorithms have been standardized by different standardization bodies, such as IETF, IEEE, ISO, etc. For example, IETF has specified several RFCs such as [RFC 5091], [RFC 6507] and [RFC6508] for both IBE and IBS algorithms. ISO and IEEE also have a few standards on IBC algorithms, such as IBS1, IBS2, and ChineseIBS [ISO IEC-IBS].

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 eliminate the binding between public keys and identities, this further simplifies the using of raw public key with TLS. Therefore, in this document, an amendment is added for supporting IBS algorithms when using raw public key.

With IBS algorithms, a PKG generates private keys for entities based on identities from requestors. Global parameters such as PKG's Master Public Key (MPK) are 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 2 defines the terms used in thsi document, Section 3 defines the data structure required when identity is used as raw public key. Section 4 specifies 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 when 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 message exchange 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 that 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 carries 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, 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

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. On the other hand, when IBS algorithm is used for authentication, normally the global parameters in use are known to client and server, hence, instead of transmitting a full set of PKG public parameters, a hash value of them is transmitted, which is put in the parameters field of AlgorithmIdentifier data structure.

The data structure used to carry the hash value of public parameters is defined as follows:

```
IBSPublicParametersHash ::= SEQUENCE {
        HASHES { mda-sha256 }
}
```

Figure 3: IBS Global Parameters Hash ANSI.1 Structure

The hash value of the global parameters is generated by taking in the DER encoded PKG public parameters of each individual IBS algorithms as input. The data structure for each IBS algorithms supported in this draft are defined in the following.

For the ECCSI IBS signature algorithms, its PKG public parameters is specified in following Figure :

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

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

Figure 4: ECCSI Global Parameters ANSI.1 Structure

The structure to carry the ISO-IBS1/ISO-IBS2 PKG public parameters are the same and is specified in following Figure :

```
ISOIBSPublicParameters ::= SEQUENCE {
    version   INTEGER { v3(3) },
    curve     OBJECT IDENTIFIER,
    hashfcn   OBJECT IDENTIFIER,
    pairing   PAIRING OPTIONAL,
    p         INTEGER OPTIONAL,
    q         INTEGER OPTIONAL,
    pointP    FpPoint,
    pointPpub FpPoint
}

PAIRING ::= ENUMERATED{
    weil  (1)  --Weil pairing
    tate  (2)  --Tate pairing
    optimalAte (3)  --Optimal Ate pairing
}
```

Figure 5: ISO-IBS1/IBS2 Global Parameters ANSI.1 Structure

The structure to carry the ISO-SM9 PKG public parameters is specified in following Figure :

```
SM9PublicParameters ::= SEQUENCE {
    version   INTEGER { v3(3) },
    curve     OBJECT IDENTIFIER,
    hashfcn   OBJECT IDENTIFIER,
    pairing   PAIRING OPTIONAL,
    p         INTEGER OPTIONAL,
    q         INTEGER OPTIONAL,
    pointP2   FpxPoint,
    pointP2pub FpxPoint,
    v          FpxElement
}

FpxPoint ::= CHOICE{
    fpPoint FpPoint,
    fp2Point [2] EXPLICIT Fp2Point,
}

Fp2Point ::= SEQUENCE{
    x  Fp2Element,
    y  Fp2Element
}

Fp2Element ::= SEQUENCE{
    a  INTEGER,
    b  INTEGER
}

FpxElement ::= CHOICE{
    fp2Elemt  Fp2Element,
    fp12Elemt Fp12Element,
}

Fp12Element ::= SEQUENCE{
    a  Fp6Element,
    b  Fp6Element
}

Fp6Element ::= SEQUENCE{
    a  Fp2Element,
    b  Fp2Element,
    c  Fp2Element
}
```

Figure 6: ISO-ChineseIBS Global Parameters ANSI.1 Structure

For ECCSIPublicParameters data structure, pointP shall be G in RFC 6507 and pointPpub shall be KPAK in RFC 6507. For

ISOIBSPublicParameters data structure, pointP and pointPpub shall be the same as defined in RFC 5091, and the pairing field shall be weil (1) or tate (2). The pairing field in SM9PublicParameters should be optimalAte (3) and the choice of v should be determined by the curve identifier. For example, for supersingular curves [RFC 5901], v shall be of type Fp2Element and for BN curves or BLS12-curves [FST10], v shall be of type Fp12Element.

To support IBS algorithm over TLS protocol, a data structure for signature value need to be defined.

Data structure for ECCSI is defined as follows(based RFC 6507):

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

Figure 7: 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.

Data structure for ISO-IBS1 is defined as follows:

```
ISO-IBS1-Sig-Value ::= SEQUENCE {
    r INTEGER,
    s ECPoint
}
```

Figure 8: ISO-IBS1 Signature Value ANSI.1 Structure

Data structure for ISO-IBS2 is defined as follows:

```
ISO-IBS2-Sig-Value ::= SEQUENCE {
    r INTEGER,
    s ECPoint
}
```

Figure 9: ISO-IBS2 Signature Value ANSI.1 Structure

Data structure for ISO-ChineseIBS (SM9) is defined as follows:

```
SM9-Sig-Value ::= SEQUENCE {
    r INTEGER,
    s ECPoint
}
```

Figure 10: ISO-ChineseIBS Signature Value ANSI.1 Structure

The definition of ECPoint can be found in section 2.2 of RFC 5480.

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
ISO/IEC 14888-3 IBS-1	ISO/IEC 14888-3: IBS-1 mechanism	1.0.14888.3.0.7
ISO/IEC 14888-3 IBS-2	ISO/IEC 14888-3: IBS-2 mechanism	1.0.14888.3.0.8
ISO/IEC 14888-3 ChineseIBS(SM9)	ISO/IEC 14888-3: ChineseIBS mechanism	1.2.156.10197.1.302.1
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 IBS algorithm. As a result, the SignatureScheme data structure has to be amended by including the presented IBS algorithms.

```
enum {
        ...

    /* IBS ECCSI signature algorithm */
    eccsi_sha256 (0x0704),
    iso_ibs1 (0x0705),
    iso_ibs2 (0x0706),
    iso_chinese_ibs (0x0707),

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

Figure 11: Include IBS 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. Identity Format and Key Revocation

With the raw public scheme proposed in TLS 1.3 [RFC 8446], the server maintains a whitelist to bind raw public key and identity. When a raw public key is revoked, then the server removes the binding record from the whitelist. On the other hand, when using IBS algorithms, it is not necessary to maintain a whitelist at the server's side. Instead, the server can simply maintain a blacklist, which is much shorter than the whitelist. However, if we simply use the identifier as a raw public key, the revocation list may keep on increasing with the time going on. Hence, to prevent the revocation list from increasing continuously, it is recommended to include a timestamp for the automatic expiration of key material. With the timestamp included in the identifier, i.e. the raw public key, server can remove the revoked raw public key from the revocation list when it is expired.

Based on the above analysis, it is necessary to include an expiration time in the identifiers for the purpose of public key management. Therefore, in this draft, we recommend both client and server take following format for the identifiers used for TLS session setup:

```
Identifier ::= SEQUENCE {
    version INTEGER {v1 (1)},
            identity String,
            expiration UTCTime
}
```

Figure 12: Identifier Format ANSI.1 Structure

Both the client and server should check the validity of the expiration field of the raw public key before verify the signature. If the expiration time is invalid, the client or the server should abort the handshake procedure.

The identities of client or server shall be unique within the domain managed by one PKG. There are many different identities domains such as email address, telephone number, Network Access Identifier (NAI), International Mobile Subscriber Identity (IMSI) etc. It is up to network operators' choice to determine which name domain the device and server take.

6. TLS Client and Server Handshake Behavior

When RAW public is used with IBS 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 the 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]).

Figure 13: Basic Raw Public Key TLS Exchange

The client hello message tells the server the types of certificate or raw public key supported by the client, and also the certificate types that the client expects to receive from the server. When raw public with IBS algorithm from the 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, the 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, the server uses the SubjectPublicKeyInfo structure to carry the raw public key, OID for IBS algorithm and public parameters or the hash value of public parameters. Assuming that ECCSI is selected, the ECCSIPublicParameters data structure is used to carry global public parameters. With this 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 is specified by the selected IBS algorithm. The data structures for PKG public parameters and signature values have been specified in the previous section of this document.

When the sever specifies that RAW public key should be used by the client to authenticate with the 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 the 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 responds 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 specified by each individual signature algorithm.

The server verifies the signature based on the chosen IBS algorithm and the relevant PKG parameters specified by the client.

7. Examples

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

7.1. 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 provides an ECCSI key by placing the SubjectPublicKeyInfo structure into the Certificate payload back to the client (3), including the OID, the identity of the server, ServerID, which is the public key of the server also, and the hash value of PKG public 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), 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 the 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 receives PKG public parameters from peer, it should decide whether these parameters are acceptable or not. An example 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, hash
                          value of 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,
 hash value of ECCSIPublicParameters),
 ClientID)}                            //(7)
{CertificatVerify = (ECCSI-Sig-Value)} //(8)
{Finished }
[Applicateion Data] ---->
[Application Data]  <--->   [Application Data]
```

Figure 14: Basic Raw Public Key TLS Exchange

7.2. 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 with ECCSI signature algorithm since eccsi_sha256 precedes the ecdsa_secp256r1_sha256. Furthermore, the client indicates that it has a ECCSI-based raw public key for client-side authentication. The client also indicates that it supports the server using either ECCSI or ecdsa_secp256r1_sha256 for the certificate signature. This further indicates that the server can 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,
               ecdsa_secp256r1_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 15: Basic Raw Public Key TLS Exchange

Handshake for other IBS algorithms can be completed similarly by including different data structures for public parameters and signature values respectively.

8. Security Considerations

Using IBS-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 IBS algorithms.

IBS signature algorithm has been standardized for ten years and has been adopted in real applications. However, we would like to point out the differences between IBS signature algorithm and the existing raw public key based algorithms: the private key of IBS used for signature generation is generated by the PKG centre, while the private key for the existing raw public key algorithms can be generated locally. Therefore, IBS mechanism may face a security risk of private key disclosure due to improper management of KMS system. The entity using IBS with TLS protocol shall be aware the above risk and an enforced key management system shall be adopted by the organization.

When using IBS algorithm, key escrow is an concern as the private key of user or devices normally is generated by PKG. PKG in the system which could generate each device's private key. However, when IBS is used in TLS1.3, passive attacks to recover the session key is not possible. Actively man-in-the-middle attack by replacing exchanged DH tokens and signatures would certainly leave traces even transiently. Similarly, a PKG could impersonate an entity to conduct a TLS session, just as the KMS in the symmetric key solution, but forensic traces could be also collected in this situation. It would be hugely risky for a PKG, which would usually be a trusted party, to launch such attacks. If such an attack is caught in red-handed, no one would trust the PKG's service anymore.

Another worry of using IBS is about the compromising of PKG. The PKG could become operationally compromised and an attacker may obtain master secrets of a PKG. However, this security risk can be solved by protect the PKG with HSM, which is often used by CA to protect the root signing key.

Private key compromising is one security risk that need to be considered when using public key technology. When using raw public key with IBS algorithm, as we have suggested in this document, a revocation list shall be maintained at the server side. At the same

time, a timestamp shall be included in the public key to prevent the revocation list from keeping on increasing. With the revocation list, the server can prevent following attacks:

1) when a device use a revoked identifier for authentication, which has not expired yet, then the server can reject the TLS session by checking the revocation list maintained at the server-side. As it is on the list, then the server aborts the TLS handshake.

2) When a device using a identifier which has been expired, the server can simply verify the timestamp contained in the identifier and abort the handshake procedure immediately.

3) If the attacker changes the timestamp within the identifier, then it will cause signature verification error when the server verify the signature contained in the signature_verify from client.

9. IANA Considerations

IANA has assigned 4 code points from the TLS SignatureScheme registry for the four IBS algorithms used in this document. The code points are listed as follows:

- eccsi_sha256

- iso_ibs1

- iso_ibs2

- iso_chinese_ibs

For all of these entries the Recommended field should be N, and the Reference field should be this document.

10. Acknowledgements

11. References

11.1. 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>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <https://www.rfc-editor.org/info/rfc5480>.

[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>.

11.2. 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>.

[FST10] "A Taxonomy of Pairing-Friendly Elliptic Curves. Journal of Cryptology. Volume 23, Issue 2, pp 224-280,", Apr 2010.

[ISO_IEC-IBS] "ISO/IEC 14888-3:2018 IT Security techniques - Digital signatures with appendix - Part 3: Discrete logarithm based mechanisms", Sept 2019.

[TS33.501] "TS 33.501, Security architecture and procedures for 5G System, v.g.0.0", Sept 2019, <https://www.3gpp.org/ftp/Specs/archive/33_series/33.501/>.

Authors' Addresses

Haiguang Wang (editor)
Huawei International Pte. Ltd.
9 North Buona Vista Dr, #13-01
Singapore 138589
SG

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

Yanjiang Yang
Huawei International Pte. Ltd.
9 North Buona Vista Dr, #13-01
Singapore 138589
SG

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

Xin Kang
Huawei International Pte. Ltd.
9 North Buona Vista Dr, #13-01
Singapore 138589
SG

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

Zhaohui Cheng
Shenzhen Olym Info. Security Tech. Ltd.
Futong Haowangjiao Podium Building
Shenzhen, Guang Dong Province 518101
CN

Phone: +86 755 8618 2108
Email: chenzh@myibc.net

Meiling Chen
China Mobile
32, Xuanwumen West
BeiJing, BeiJing 100053
China

Email: chenmeiling@chinamobile.com