Use Identity as Raw Public Key in EAP-TLS

Abstract

This document specifies the use of identity as a raw public key in
EAP-TLS, the procedure of EAP-TIBS is consistent with EAP-TLS's
interactive process, identity-based signature is extended to support
EAP-TLS's signature algorithms to avoid X.509 certificates, this
authentication method can avoid the overhead of receiving and
processing certificate chains.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Extensible Authentication Protocol(EAP) defined in [RFC3748] can provide support for multiple authentication methods. Transport Layer Security(TLS) provides for mutual authentication, integrity-protected ciphersuite negotiation, and exchange between two endpoints. The EAP-TLS defined in [RFC5216] which combines EAP and TLS that apply EAP method to load TLS procedures.

Traditionally, TLS client and server public keys are obtained in PKIX containers in-band as part of the TLS handshake procedure and are validated using trust anchors based on a PKIX certification authority (CA). But there is another method, Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) are defined in [RFC7250], the document defines two TLS extensions client_certificate_type and server_certificate_type, which can be used as part of an extended TLS handshake when raw public keys are used. [RFC9190] reads certificates can be of any type supported by TLS including raw public keys. In [RFC7250] it assuming that an out-of-band mechanism is used to bind the public key to the entity presenting the key.

Digital signatures provide the functions of Sender reliability and Message integrity. A chain of trust for such signatures is usually provided by certificates, but in low-bandwidth and resource-constrained environments, the use of certificates might be undesirable. In comparison with the original certificate, the raw

public key is fairly small. This document describes a signature algorithm using identity as a raw public key in EAP-TLS, instead of transmitting a full certificate in the EAP-TLS message, only public keys are exchanged between client and server, also known as EAP-TIBS.

With the existing raw public key scheme, a public key and identity mapping table is required at server. This table usually established with offline method and may require additional efforts for establishment and maintenance, especially when the number of devices are huge. On the other hand, with IBS signature algorithm, it not only can take the advantage of raw public key, but also eliminates the efforts for the mapping table establishment and maintenance at the server side. Instead, a small table for CRL is enough for exclude revoked identity from accessing the network. A number of IBE and IBS algorithms have been standardized, such as ECCSI defined in [RFC6507].

IBC was first proposed by Adi Shamir in 1984. For an IBC system, a Key Management System (KMS) is required to generate keys for devices. The KMS choose its KMS Secret Authentication Key(KSAK) as the root of trust. A public parameter, KMS Public Authentication Key (KPAK) is derived from this secrete key and is used by others in verifying the signature. The signatures are generated by an entity with private keys obtained from the KMS. KMS is a trusted third party, users or devices can obtain private key using their identities from KMS. In IBS the private key is also known as Secret Signing Key(SSK). A sender can sign a message using SSK. The receiver can verify the signature with sender's identity and the KPAK.

This method has great advantages in internal management.

2. Terminology

The following terms are used:

  *IBC: Identity-Based Cryptograph, it is an asymmetric public key cryptosystem.

  *IBS: Identity-based Signature, such as ECCSI.

  *PKI: Public Key Infrastructure, an infrastructure built with a public-key mechanism.

  *Authenticator: The entity initiating EAP authentication.

  *Peer: The entity that responds to the authenticator.

*Backend authenticator server: A backend authentication server is
    an entity that provides an authentication service to an
    authenticator. When used, this server typically executes EAP
    methods for the authenticator.

   *EAP server: The entity that terminates the EAP authentication
    method with the peer. In the case where no backend authentication
    server is used, the EAP server is part of the authenticator. In
    the case where the authenticator operates in pass-through mode,
    the EAP server is located on the backend authentication server.

3. EAP TIBS Use Cases

3.1. IoT use case

   Used for authentication of Internet of Things devices: due to the
   limited processing power of IoT devices, resources for secure
   identity authentication are limited, especially passive, long life
   cycle devices, however, the traditional certificate authentication
   based on PKI X509, because of the complexity of certificate
   processing and certificate chain authentication, not very suitable
   for the Internet of Things scenario. Internet of Things devices
   really need a more lightweight authentication method, and EAP-TIBS
   as one of the candidates.

3.2. Non CA use case

   Used for systems that do not support CA certificates: an internal
   system with network security boundaries that can self-operate the
   Key Management System(KMS) secret key distribution center, EAP-TIBS
   can be used between internal subsystems.

4. Structure of the Raw Public Key Extension

   To support the negotiation of using raw public key between client
   and server, a new certificate structure is defined in [RFC7250]. 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, SubjectPublicKeyInfo which is a
   data structure is used to carry the raw public key and its
   cryptographic algorithm.

   The SubjectPublicKeyInfo structure is defined in Section 4.1 of
   [RFC5280] and not only contains the raw keys, such as the public
   exponent and the modulus of an RSA public key, but also an algorithm
   identifier. The algorithm identifier can also include parameters.
   The structure of SubjectPublicKeyInfo is shown in Figure 1:

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

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

         Figure 1: SubjectPublicKeyInfo ASN.1 Structure

   The algorithms identifiers are Object Identifier(OIDs),
   AlgorithmIdentifier is also data structure with two fields, OID
   represent the cryptographic algorithm used with raw public key, such
   as ECCSI, parameters are the necessary parameters associated with
   the algorithm.

   In the case of IBS algorithm, the User's identity is the raw public
   key which can be represented by "subjectPublicKey", when ECCSI is
   used as the Identity-based signature algorithm, then "algorithm" is
   for ECCSI, and "parameters" is the parameters needed in ECCSI.

   So far, IBS has the following four algorithms, the following table
   is the corresponding table of Key type and OID.

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

   In the draft draft-wang-tls-raw-public-key-with-ibc, there extend
   signature scheme with IBS algorithm which indicated in the client's

"signature_algorithms" extension. The SignatureScheme data structure also keep pace with the section 4.

5. EAP-TIBS for TLS1.2

5.1. EAP-TIBS Handshake

This section describes EAP-TIBS in the case of TLS1.2, as described in [RFC7250], the document intrudoces the use of raw public keys in TLS/DTLS, the basic raw public key TLS exchange will appear as follows, Figure 2 shows the client_certificate_type and server_certificate_type extensions added to the client and server hello messages. An extension type MUST NOT appear in the ServerHello unless the same extension type appeared in the corresponding ClientHello, defined in [RFC5246].

The server_certificate_type extension in the client hello indicates the types of certificates the client is able to process when provided by the server in a subsequent certificate payload.

The client_certificate_type and server_certificate_type extensions sent in the client hello each carry a list of supported certificate types, sorted by client preference. When the client supports only one certificate type, it is a list containing a single element. Many types of certificates can be used, such as RawPublicKey, X.509 and OpenPGP.

This section describes EAP-TLS extend using raw public keys, the procedures is as follows, In the discussion, we will use the term "EAP server" to denote the ultimate endpoint conversing with the peer.

```
Authenticating Peer       EAP server
    ------------------      -------------
                            <- EAP-Request/
                            Identity
    EAP-Response/
    Identity (MyID) ->
                            <- EAP-Request/
                            EAP-Type=EAP-TLS
                            (TLS Start)
    EAP-Response/
    EAP-Type=EAP-TLS
    (TLS client_hello
     +signature_algorithm
     server_certificate_type,
     client_certificate_type)->

                            <- EAP-Request/
                            EAP-Type=EAP-TLS
                            (TLS server_hello,
                              {client_certificate_type}
                              {server_certificate_type}
                              {TLS certificate}
                              {TLS server_key_exchange}
                              {TLS certificate_request}
                              {TLS server_hello_done}
                            )
    EAP-Response/
    EAP-Type=EAP-TLS
    (TLS certificate,
     TLS client_key_exchange,
     TLS certificate_verify,
     TLS change_cipher_spec,
     TLS finished) ->
                            <- EAP-Request/
                            EAP-Type=EAP-TLS
                            (TLS change_cipher_spec,
                             TLS finished)
    EAP-Response/
    EAP-Type=EAP-TLS ->
                            <- EAP-Success

    Figure 2: EAP-TIBS authentication procedure for TLS1.2
```

5.2. EAP-TIBS example

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.

```
Authenticating Peer                    EAP server
------------------                     ------------
                                       <- EAP-Request/
                                       Identity
    EAP-Response/
    Identity (MyID) ->
                                       <- EAP-Request/
                                       EAP-Type=EAP-TLS
                                       (TLS Start)
    EAP-Response/
    EAP-Type=EAP-TLS
    (TLS client_hello
    signature_algorithm = (eccsi_sha256)
     server_certificate_type = (RawPublicKey,...)
     client_certificate_type = (RawPublicKey,...))->

                                       <- EAP-Request/
                                       EAP-Type=EAP-TLS
                                       (TLS server_hello,
                                       {client_certificate_type = RawPublic
                                       {server_certificate_type = RawPublic
                                       {certificate = (1.3.6.1.5.5.7.6.29,
                                       value of ECCSIPublicParameters),
                                       serverID)}
                                       {certificate_request = (eccsi_sha256
                                       {server_hello_done}
                              )
    EAP-Response/
    EAP-Type=EAP-TLS
    ({certificate = ((1.3.6.1.5.5.7.6.29,
    hash value of ECCSIPublicParameters),
    ClientID)},
     {certificate_verify = (ECCSI-Sig-Value)},
     {finished}) ->
                                       <- EAP-Request/
                                       EAP-Type=EAP-TLS
                                       (TLS finished)
    EAP-Response/
    EAP-Type=EAP-TLS ->
                                       <- EAP-Success
```

        Figure 3: EAP-TIBS example

6. EAP-TIBS for TLS1.3

6.1. EAP-TIBS Handshake

TLS1.3 defined in [RFC8446], as TLS 1.3 is not directly compatible with previous versions, all versions of TLS incorporate a versioning mechanism which allows clients and servers to interoperably negotiate a common version if one is supported by both peers. when make the discussion on EAP-TLS using raw public keys we also make a different with TLS1.2, this section is for EAP-TLS1.3 handshake using raw public keys and give example for EAP-TIBS.

This section describes EAP-TLS1.3 extend using raw public keys, the procedures is as follows, both client and server have the extension "key_share", the "key_share" extension contains the endpoint's cryptographic parameters. the "signature_algorithm" extension contains the signature algorithm and hash algorithms the client and server can support for the new signature algorithms specific to the IBS algorithms. When IBS is chosen as signature algorithm, the server need to indicated the required IBS signature algorithms int the signature_algorithm extension within the CertificateRequest.

```
Authenticating Peer      EAP server
    -------------------      -------------
                             <- EAP-Request/
                             Identity
    EAP-Response/
    Identity (MyID) ->
                             <- EAP-Request/
                             EAP-Type=EAP-TLS
                             (TLS Start)
    EAP-Response/
    EAP-Type=EAP-TLS
    (TLS client_hello
     +key_share
     +signature_algorithm
     server_certificate_type,
     client_certificate_type)->

                             <- EAP-Request/
                             EAP-Type=EAP-TLS
                             (TLS server_hello,
                               +key_share
                               {EncryptedExtensions}
                               {client_certificate_type}
                               {server_certificate_type}
                               {certificate}
                               {CertificateVerify}
                               {certificateRequest}
                               {Finished}
                          )
    EAP-Response/
    EAP-Type=EAP-TLS
    ({certificate}
     {CertificateVerify}
     {Finished}
     ) ->
                          EAP-Request/
                          EAP-Type=EAP-TLS
                 <--TLS Application Data 0x00

     EAP-Response/
     EAP-Type=EAP-TLS-->
                             <- EAP-Success


    Figure 4:  EAP-TIBS authentication procedure for TLS1.3
```

6.2. EAP-TIBS example

When the EAP server receives the client hello, it processes the
message. Since it has an ECCSI raw public key from the KMS, it
indicates that it agrees to use ECCSI and provides an ECCSI key by
placing the SubjectPublicKeyInfo structure into the Certificate
payload back to the client, including the OID, the identity of
server, ServerID, which is the public key of server also, and hash
value of KMS public parameters. The client_certificate_type
indicates that the TLS server accepts raw public key. The TLS server
demands client authentication, and therefore includes a
certificate_request, which requires the client to use eccsi_sha256
for signature. A signature value based on the eccsi_sha256 algorithm
is carried in the CertificateVerify. The client, which has an ECCSI
key, returns its ECCSI public key in the Certificate payload to the
server, which includes an OID for the ECCSI signature. The example
of EAP-TLS1.3-IBS is as follows:

```
Authenticating Peer                      EAP server
  ------------------                      -------------
                                          <- EAP-Request/
                                          Identity
  EAP-Response/
  Identity (MyID) ->
                                          <- EAP-Request/
                                          EAP-Type=EAP-TLS
                                          (TLS Start)
  EAP-Response/
  EAP-Type=EAP-TLS
  (TLS client_hello
   signature_algorithm = (eccsi_sha256)
   server_certificate_type = (RawPublicKey)
   client_certificate_type = (RawPublicKey))->

                                 <- EAP-Request/
                                 EAP-Type=EAP-TLS
                                 (TLS server_hello,
                                 +key_share
                                 {client_certificate_type = RawPublic
                                 {server_certificate_type = RawPublic
                                 {certificate = (1.3.6.1.5.5.7.6.29,
                                 value of ECCSIPublicParameters,
                                 serverID)}
                                 {certificate_request = (eccsi_sha256
                                 {certificate_verify = {ECCSI-Sig-Val
                                 {Finished}

                           )
  EAP-Response/
  EAP-Type=EAP-TLS
  ({certificate = ((1.3.6.1.5.5.7.6.29,
  hash value of ECCSIPublicParameters),
  ClientID)},
   {certificate_verify = (ECCSI-Sig-Value)},
   {Finished})
              --->
                                     EAP-Request/
                                     EAP-Type=EAP-TLS
                           <----TLS Application Data 0x00)
  EAP-Response/
  EAP-Type=EAP-TLS---->
                             <---- EAP-Success


        Figure 5: EAP-TLS1.3-IBS example
```

7. IANA Considerations

This document registers the following item in the "Method Types"
registry under the "extensible Authentication Protocol(EAP)
Registry" heading.

```
+---------+------------------+
| Value   | Description      |
+---------+------------------+
| TBD     | EAP-TIBS         |
+---------+------------------+
```

8. Security Considerations

Although the identity authentication has been extended, the
generation of session key still continues the EAP-TLS method.

9. Informative References

[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/rfc/rfc6507>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H.
           Levkowetz, Ed., "Extensible Authentication Protocol
           (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004,
           <https://www.rfc-editor.org/rfc/rfc3748>.

[RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS
           Authentication Protocol", RFC 5216, DOI 10.17487/RFC5216,
           March 2008, <https://www.rfc-editor.org/rfc/rfc5216>.

[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/rfc/rfc7250>.

[RFC5280] 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", RFC 5280, DOI 10.17487/RFC5280, May
           2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
           (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/
           RFC5246, August 2008, <https://www.rfc-editor.org/rfc/
           rfc5246>.

[RFC9190]   Preuß Mattsson, J. and M. Sethi, "EAP-TLS 1.3: Using the
            Extensible Authentication Protocol with TLS 1.3", RFC
            9190, DOI 10.17487/RFC9190, February 2022, <https://
            www.rfc-editor.org/rfc/rfc9190>.

[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/rfc/rfc8446>.

Authors' Addresses

Meiling Chen (editor)
China Mobile
BeiJing
China

Email: chenmeiling@chinamobile.com

Li Su
China Mobile
BeiJing
China

Email: suli@chinamobile.com

Haiguang Wang
Huawei International Pte. Ltd.
Singapore

Email: wang.haiguang1@huawei.com