Use Identity as Raw Public Key in EAP-TLS


IETF113-2022-EMU

MeilingChen /China Mobile
Li Su /China Mobile
HaiguangWang/Huawei
Use case of the EAP-TLS-IBS:

1. Used for authentication of Internet of Things devices
2. Used for systems that do not support CA certificates
   The goal is to improve the authentication efficiency of the IoTs

Running code

1. Coding eap-tls-ibs based on eap-tls1.2 using ECCSI in 2020

Draft history
Presentations in IETF109 and IETF111

Comments received
1、What scenario is it for?
IoT, especially passive long-life devices.
2、Is it related to IBE?
only use IBS for identity authentication
3、any running code?
Simple prototype implementation
4、Any cross scope of IOT OPS?
No

Commenters
Russ Housley: would do the ASN.1 structures for pyasn1-modules when it becomes an RFC.
will review the ASN.1 portions of the specification to make sure they are clear.

Sean Turner: I am not a lover of IBS. I am okay with people exploring. WG or AD sponsor is okay.
Example:
ECCSI used for EAP-TLS-IBS

```
(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 = RawPublicKey}
 {server_certificate_type = RawPublicKey}
 {certificate = (1.3.6.1.5.5.7.6.29, hash value of ECCSIPublicParameters,
 serverID})
 {certificate_request = (eccsi_sha256)}
 {certificate_verify = {ECCSI-Sig-Value}}
 {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})
```
Authentication by IBS

**Prerequisite:** The client and server have obtained the public-private key pair from the same KMS

server to client: public key, signature, hash value of KMS public parameters
{ ID(public key)+Hash value+OID } = Certificate
{ Signature } = Certificate_verify

**Client processing:** validate hash value of KMS public parameters to prove that they belong to the same algorithms and KMS
The client verifies the identity of the server: input ID, Message, Signature, KMS’s public parameter

**Mathematical operation:** Refer to rfc6507 for the verification process.

A successful signature indicates that the authentication has passed.

vice versa
Process of initialization, signature and signature verification For ECCSI

**Signer**
- **Compute KPAK:**
  \[ KPAK = |KS|A|G| \]

**KMS**
- **Generate (SSK, PVT):**
  \[ \text{Input: ID, KSAK, KPAK} \]
  \[ \text{Output: (SSK, PVT)} \]
  \[ \text{Processing:} \]
  \[ PVT = [u]G; \]
  \[ HS = \text{hash}(G||KPAK)||ID||PVT); \]
  \[ SSK = (KSAK + HS^v) \mod q; \]

**Verifier**
- **Verify SSK:**
  \[ \text{Input: (SSK, PVT)} \]
  \[ \text{Verification process:} \]
  \[ HS = \text{hash}(G||KPAK)||ID||PVT); \]
  \[ KPAK = |SSK|G - [HS]PVT; \]

**Sign message (M):**
- **Input:** ID, SSK, KPAK, PVT
- **Output:** Signature
- **Calculation process:**
  \[ J = [H]G; \]
  \[ r = bc; \]
  \[ HE = \text{hash}(HS||r||M); \]
  \[ s = ((HE + r^*SSK)^{-1})*1 \mod q; \]
  \[ \text{Signature} = (r||s)||PVT; \]

**Signature verification process:**
- **Input:** ID, M, KPAK, Signature
- **Process:**
  \[ HS = \text{hash}(G||KPAK)||ID||PVT); \]
  \[ HE = \text{hash}(HS||r||M); \]
  \[ Y = [HS]PVT + KPAK; \]
  \[ J = s ((HEG + rY) \mod P) \]
  \[ \text{Verify Jx modulo p not 0} \]