# Use Identity as Raw Public Key in EAP-TLS

https://datatracker.ietf.org/doc/draft-chen-emu-eap-tls-ibs/

## IETF116-2023-EMU

Meiling Chen /China Mobile
Li Su /China Mobile

**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

**Draft history**

Presentations in IETF109 and IETF111

Adoption Call(2022), still waiting for more interst.

**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 .

**Alexander Clouter**: I thought the hold up was OpenSSL does not support RFC7250 without patching?

**Heikki Vatiainen**: Related to implementations; for me OpenSSL support would be required for an implementation. While this seems to be progressing, a client implementation would be needed for testing since I work on the EAP server side. Is there any news on the client side? For example, would wpa_supplicant be a possible client? The IOT use case might be something where a client could be based on wpa_supplicant.

**updates from 04 to 05**

1. Algorithm name modification: EAP-TLS-IBS hase been changed to EAP-TIBS;
2. Removed the part for Basic Raw Public Key TLS Exchange;
3. Removed the part for EAP-TLS mutual authentication with TLS1.3 handshake;
4. Updated references to RFCs: RFC 9190 and RFC 8446;
5. Security Considerations: although the identity authentication has been extended, the generation of session key still continues the EAP-TLS method.

**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

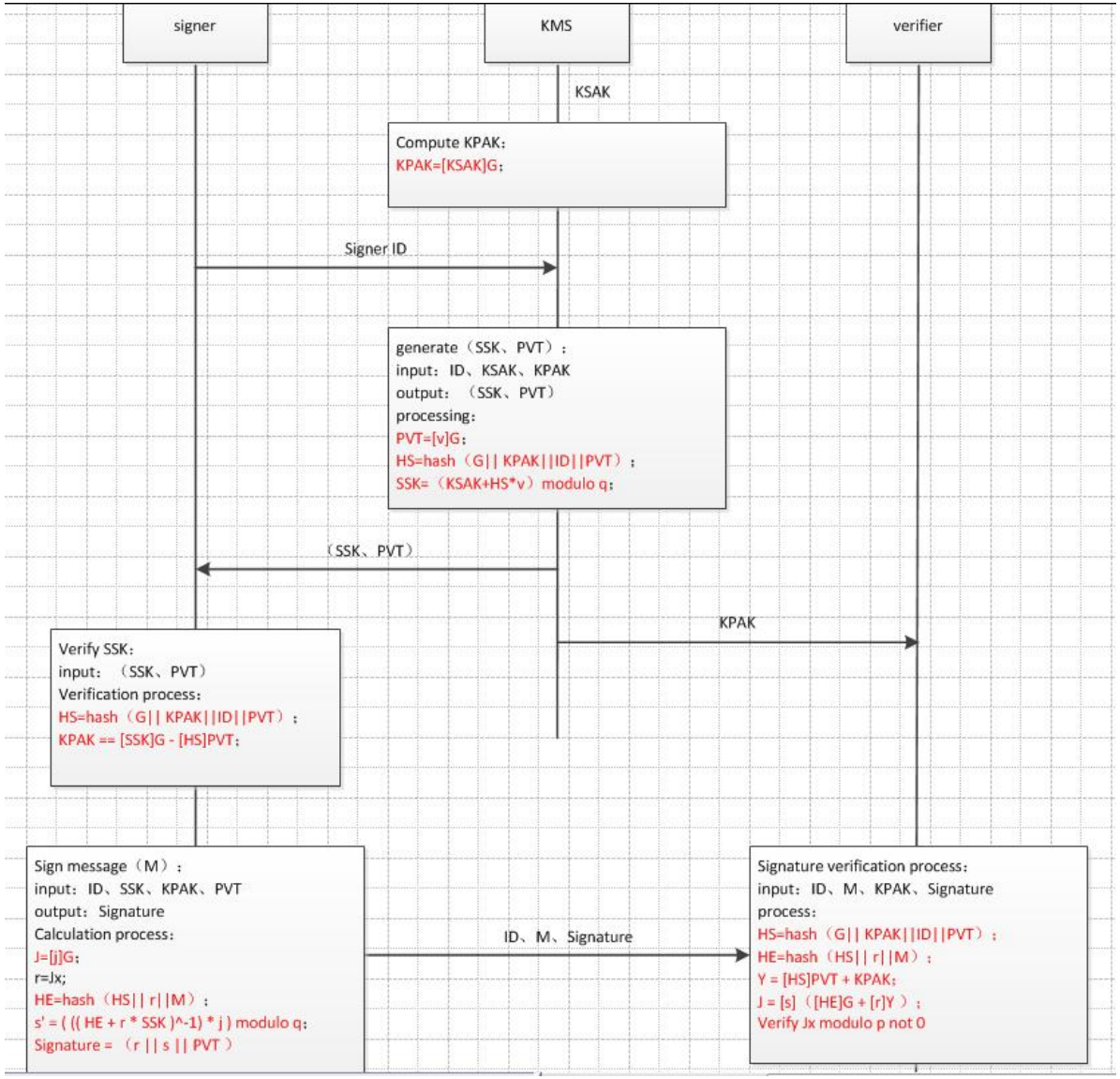
**server to client**: public key, signature, hash value of KMS public parameters
{ ID(public key)+Hash value+OID } = Certificate
{ Signature } = Certificate_verify

**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



1. We continue to wait for more people to be interested, we are very happy to see if someone willing to do code implementation.
2. We are also waiting for the release of a new version of openssl which support RPK.