BRSKI-CLE
A Certificateless Enrollment protocol in BRSKI

draft-yan-anima-brski-cle-00

Lei YAN

IETF 117, July 2023
Use case

• The gateway cares about whether the IoT device is legitimate rather than who is the IoT device.
  > The identity information in certificates is redundant.

• Requirements of the authentication mechanism:
  > Lightweight
    - IoT devices are commonly resource-constrained.
  > Scalable
    - The amount of IoT devices is huge.

Smart medical care in the hospital
Existing asymmetric cryptography based authentication mechanisms

• **X.509 Certificate and PKI**
  > Not lightweight:
  >   - A certificate is usually associated with a certificate chain, which results in the overhead of transmitting data and validation computation.
  >   - C509 (CBOR Encoded) certificate has smaller size of a certificate, but still relies on the certificate chain.

• **Raw public key**
  > Lack of scalability:
  >   - The peer’s public key must be obtained via an out-of-band method.

• **Identity-based cryptography (IBC)**
  > Lack of security:
  >   - The device’s private key is generated by an authentication centre.
  > lightweight and scalable:
  >   - Identity is also the public key.
  >   - No need of out-of-band configure.

![Example of certificate chain](image)
Certificateless authentication mechanism

- Certificateless Public Key Cryptography was first proposed in 2003 to deal with the key escrow limitation in IBC [1].
  > A trusted third party named as KGC is responsible to generate a partial private key for the users.
  > The user obtains the full private key by combining the partial private key with a secret value, which is unknown to any other party, including the KGC.
  > Lightweight and scalable: inherited from IBC.

- Certificateless authentication mechanism in this draft:
  > The user’s public key derives from the user’s identity.

Background

- BRSKI [RFC8995] is an excellent automated bootstrap protocol for unconfigured devices called “pledges”.
- This draft focuses on the enrollment phase and the phase after enrollment.
- Existing enrollment protocols:
  - EST [RFC7030]
  - Constrained BRSKI
  - BRSKI-AE
  - ACME-integrations
- All these protocols use a CA to issue local certificates to the pledges.
- After enrollment, the pledge uses the local certificate to authenticate each other.
BRSKI-CLE

• Instead of the certificate, a **credential** is calculated by public keys.

• Instead of the CA, an authentication centre (**AC**) is used to issue credentials.

• A **mutual authentication protocol** is proposed to show how to use the **credential** in the authentication after enrollment.
Performance comparison with certificate authentication in peer-to-peer communications

**Comparison of computational overhead**

<table>
<thead>
<tr>
<th>TLS1.3 ECDHE+ECDSA, certificate chain of 3 levels (root certificate, CA certificate, device certificate)</th>
<th>Certificateless authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 sign (1 PM) + 3 verify (6 PM) + ECDHE (2 PM)</td>
<td>1 ephemeral public key (1 PM) + 1 peer’s public key (1 PM) + 1 symmetric key (2 PM)</td>
</tr>
<tr>
<td>Total: 9 PM</td>
<td>Total: 4 PM</td>
</tr>
</tbody>
</table>

PM: the operation of point multiplication in ECC

**Computational capability:** improved ~ 50%+

**Comparison of transmission overhead**

<table>
<thead>
<tr>
<th>TLS1.3 ECDHE+ECDSA with certificate chain of 3 levels (signed with ECDSA)</th>
<th>Certificateless authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>2100 Byte</td>
<td>424 Bytes</td>
</tr>
</tbody>
</table>

Improving: \((\frac{2100-424}{2100}) = 80\%\)

**Transmission capability:** improved ~ 80%
Architecture

• The only change: CA → AC

• The AC can be implemented
  > on the registrar
  > as a backend domain component

• Assumptions for the registrar and AC:
  > Their communication is protected by a security protocol, such as TLS or DTLS.
  > They can authenticate each other using the security protocol.
Enrollment protocol

• The `CredentialRequest` message is encrypted by the AC’s public key.

• The `CredentialResponse` message is encrypted by the symmetric key.
  > There is a symmetric key generated randomly by the pledge in the `CredentialRequest` message.
Mutual Authentication Protocol

• The initiating pledge acts as a client.
• The responding pledge acts as a server.
• The **Credential** exchange:
  > Calculate a symmetric key
• The **ProofofPossession** exchange:
  > Verify that the peer has got the same symmetric key
Thank you!

Questions?
Authentication and Authorization for Constrained Environments Using the OAuth 2.0 Framework (ACE-OAuth) [RFC9200]

- Oriented different scenarios
  - ACE-Oauth: A token is used by a client to request resources from a server
  - BRSKI-CLE: the usage of the credential is more general.
    - The pledge’s action after the authentication using credentials is not specified.

- Client registration and provisioning of client credentials to the client are not defined in ACE-OAuth[RFC 9200].
Enrollment protocol

- **IDevID**: An Initial Device Identifier X.509 certificate installed by the vendor on new equipment.
- **ID_P**: The local identity of the pledge.
- **CS_P**: The cipher suites list supported by the pledge.
- **ID_AC**: The identity of the AC.
- **PK_AC**: The AC's public key.
- **CS_AC**: The cipher suite choosen by the AC.
- **Enc**: The function encrypting by a public key.
- **symKey**: A symmetric key for the following communication.
- **PK_P**: The public key of the pledge.
- **Cred**: The credential of the pledge.
- **pSK**: The pledge's partial private key from the AC.
Mutual Authentication Protocol

- **ID_X**: The identity of the sender.
- **Cred_X**: The credential of the sender.
- **G_X**: The ephemeral public key of the sender.
- **CS_X**: The cipher suites list supported by the client or the cipher suite chosen by the server.
- **AuthCode_X**: The authentication code of the sender.
- **X** denotes the sending side
  - "C" for the client
  - "S" for the server
Key Derivation-Enrollment protocol

- The key derivation is based on the Schnorr signature algorithm.
- Assuming
  > "a" is a random number generated by the pledge;
  > "b" is a random number generated by the AC;
  > "G" is a elliptic curve base point;
  > "c" is a random number generated by the AC.
- The symKey in the CredentialRequest message is a random number generated by the pledge.
- SK_P: The private key of the pledge
- PK_P: The public key of the pledge
- SK_AC: the private key of AC
- PK_AC: the public key of AC
- fSK: The final private key of the pledge
- fPK: The final public key of the pledge
Key Derivation-Mutual Authentication Protocol

- Assuming
  > "x" is a random number generated by the client;
  > "y" is a random number generated by the server;
  > "G" is an elliptic curve base point.
- **G_X**: The ephemeral public key of the client
- **G_Y**: The ephemeral public key of the server
- **fPK_C**: The final public key of the client
- **fPK_S**: The final public key of the server
- **cv**: The concatenation value as the input for the hash function
- **MK**: The master key
- **AuthKey**: The key for the authentication in the proof-of-possession exchange
- **EncKey**: The symmetric key for the communication after the mutual authentication