TLS-POK

Proof of Knowledge

draft-friel-tls-eap-dpp

Friel, Harkins
Context

• Wi-Fi alliance Device Provisioning Protocol defines how a supplicant’s bootstrap keypair can be used to bootstrap the supplicant against a Wi-Fi network

• DPP gives the supplicant a guarantee that it is connecting to a network that knows its bootstrap public key

• Bootstrap Public key:
  • Encoded using the ASN.1 SEQUENCE SubjectPublicKeyInfo from RFC5280
  • A raw keypair – does not have to be part of a PKI
  • May be static, embedded in the supplicant, and printed in a QR label, included in a BOM, etc.
  • May be dynamically generated and displayed on a GUI

• We want to reuse the same bootstrap public key to enable a device to securely bootstrap against a wired network using EAP-TLS via a TLS extension

• This means that if a device supports both Wi-Fi and wired networks, the same QR, BOM, etc. may be used to establish trust across both Wi-Fi and wired deployments

DPP:I:GS-803XL;K:MDkwEwYHKoZlizj0CAQYIKoZlizj0DAQcD1gAC8YIhboMFjXzwIS3Ry9c4UAR+VZutTkYnjNLNWWGedE=;;
DPP Outline

1. Public bootstrap key is provisioned in DPP Configurator
   • Configurator could be a mobile App, or could be embedded to Wi-Fi AP

2. Proof of knowledge via DH using the bootstrap key and the Configurator ephemeral key
   • Supplicant proves it knows the private key of the bootstrap keypair
   • Configurator proves it knows the public key of the bootstrap keypair
   • Secure channel established

3. Network information is securely exchanged

4. Supplicant attaches to network

Bsk: bootstrap key
pkPsk: public key
skBsk: private key

Cdhe: Configurator DHE keypair
pkCdhe: public key
skCdhe: private key
Bootstrap key reuse for wired LAN

- The pkBsk is scanned into the network and known by the AAA / EAP TLS server
- The device wants the network to prove it knows its pkBsk
- The network wants the device to prove it knows the associated skBsk
- Can be achieved by exchanging two sets of DH keys in the ClientHello / ServerHello
  1. Standard key_share where both sides generate ephemeral key pairs
  2. Bootstrap extension where client sends its H(pkBsk) instead of pkBsk. Server responds with a second ephemeral key, and uses H(pkBsk) to lookup the actual pkBsk in order to complete its key derivation
- Both DHE calculations are injected into the key schedule using the mechanism outlined in draft-jhoyla-tls-extended-key-schedule
This document defines the "bskey_share" extension.

```
struct {
    select (Handshake.msg_type) {
        case client_hello:
            opaque bskey[32];
        case server_hello:
            opaque bskey_exchange<1..2^16-1>;
    }
} BootstrapKey;
```

```
Client                                      Server
----------                                  ----------
ClientHello                                  ServerHello
+ bskey_share                                + bskey_share
+ key_share                                  + key_share
                      {EncryptedExtensions}               {Finished}
{Finished}                                  [Application Data*]
[Application Data]  <--------  [Application Data]
```

```
PSK -> HKDF-Extract = Early Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

Derive-Secret(. , "derived", """)

bskey_input || (EC)DHE -> HKDF-Extract = Handshake Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

Derive-Secret(. , "derived", """)

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Secret(...) 
  |              ---> Derive-Secret(...) 

\( \emptyset \) -> HKDF-Extract = Master Secret
  |              ---> Derive-Scret...
Everyone is Happy
Security Considerations

• Leverages TLS handshake with no esoteric cryptography
  • Existing TLS security proofs should still be applicable
  • draft-jhoyla-tls-extended-key-schedule should handle key schedule changes

• Bootstrap key security
  • TLS-POK has the same security stance as DPP with respect to Bootstrap keys
    • **DPP**: If you know the bootstrap public key, you can claim the device
    • **TLS-POK**: If you know the bootstrap public key, you can claim the device
Working TLS Code

• Golang mint TLS stack branch
• https://github.com/upros/mint/tree/tls-pok
Discussion and Next Steps

• Consensus at EMU at IETF109 to progress this work

• 3 general work areas
  • TLS extensions to transport bootstrap key identifiers and extra DHE keypairs
  • TLS key schedule enhancements: draft-jhoyla-tls-extended-key-schedule
  • EAP/TEAP extensions to leverage new TLS-POK handshake

• How many documents?
  • draft-jhoyla-tls-extended-key-schedule
  • Short TLS WG draft for TLS extensions?
  • Short EMU WG draft for leveraging new TLS-POK mechanism?
  • Single draft that covers both TLS extensions and EAP mechanisms?