draft-friel-tls-eap-dpp

Dan Harkins & Owen Friel

EMU WG, IETF 111

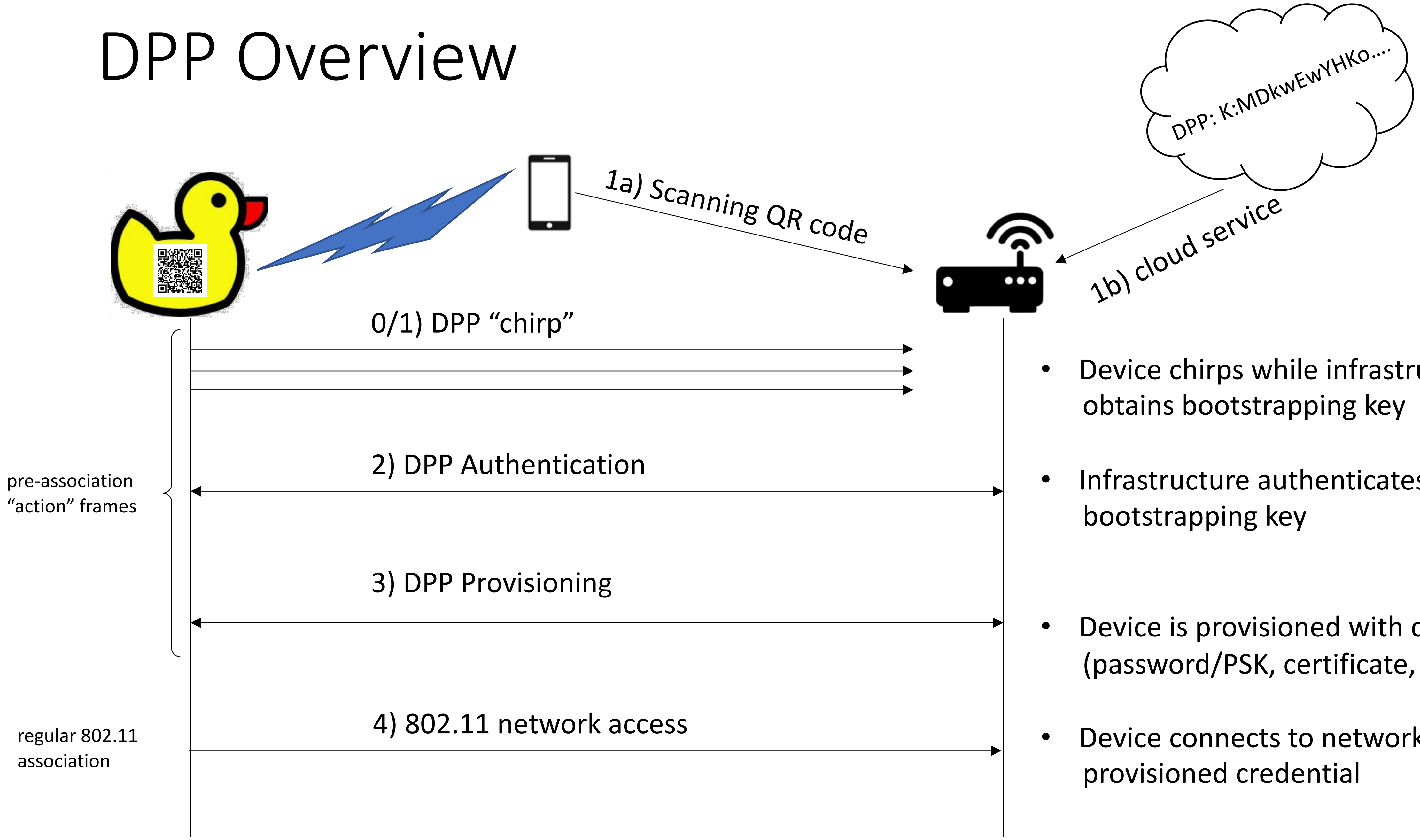
Context

- Wi-Fi alliance Device Provisioning Protocol defines how a supplicant's bootstrap keypair can be used to authenticate the supplicant and provision it for a Wi-Fi network
- Supplicant has a guarantee that it is connecting to a network that knows its bootstrap public key, network has a guarantee that the only the holder of the (private) bootstrapping key can be onboarded
  - Trust (and security) in DPP depends on integrity of bootstrapping key database
- Bootstrap Public key:
  - Encoded using the ASN.1 SEQUENCE SubjectPublicKeyInfo from RFC5280 into a URI

DPP:I:GS-803XL;K:MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgAC8YIhb0MFjXZzwIS3Ry9c4UAR+VZutTkYnjNLNWWGedE=;;

  - A raw keypair – does not have to be part of a PKI, private key can be in a TPM
  - May be static, embedded in the supplicant, and printed in a QR label, included in a BOM, etc.
  - Could be obtained from vendor cloud for true zero-touch experience
  - May be dynamically generated and displayed on a GUI
- DPP is able to provision all possible network credentials
  - PSK/password, both group and individual using SAE password identifiers
  - certificate via an EST-like exchange– CSR Attrs request, then PKCS10/PKCS7
  - connector (a signed JSON web key) used for DPP network access

DPP Overview



- Device chirps while infrastructure obtains bootstrapping key
- Infrastructure authenticates device with bootstrapping key
- Device is provisioned with credential (password/PSK, certificate, connector)
- Device connects to network with provisioned credential

OK, so what’s this to do with EMU?

- DPP solves the “catch-22” of *need a credential to get a credential*
- DPP is 802.11 only, exchange is done with pre-association 802.11 “action frames”, post association is regular Wi-Fi
- Want to use the same DPP bootstrapping (cloud, QR, BOM, NFC, etc.) to establish trust across both Wi-Fi and wired deployments
- We want to reuse the same bootstrap public key to solve the “catch-22” for wired enterprise
  - 802.1X will do EAP ID-request upon link-up, no ”chirping” necessary
  - Wired equivalent of “pre-association action frame” is EAP
  - Need to use this bootstrapping key to authenticate TEAP
  - Use TEAP’s own EST-like exchange to provision a certificate on device

TLS Authentication w/DPP Bootstrapping keys

- Bootstrapping key is used to generate two data:
  - Identifier to signal which bootstrapping key to use for authentication
  - PSK for TLS authentication
- Use RFC 8773 “external PSK”
  - PSK derived from bootstrapping key is injected into key schedule
  - Client and server prove knowledge of PSK (and therefore bootstrapping key)
- Use RFC 7250 TLS with raw public key
  - Client signs with bootstrapping key, proves possession of private key to server
- Use draft-group-tls-extensible-psks
  - Client signals the derived PSK identity and type in extended_psk extension
- No TLS changes/extensions required over and above defining new BSK type for draft-group-tls-extensible-psks

TLS authentication w/DPP bootstrapping keys

```
bskeypsk = HKDF-Expand(HKDF-Extract(<>, bskey),
                   "tls13-extended-psk-bskey", L)
 identity = HKDF-Expand(HKDF-Extract(<>, bskey),
                   "tls13-psk-identity-bskey", L)
```

```
bskeypsk = HKDF-Expand(HKDF-Extract(<>, bskey),
                   "tls13-extended-psk-bskey", L)
 identity = HKDF-Expand(HKDF-Extract(<>, bskey),
                   "tls13-psk-identity-bskey", L)
```

```
Client                                                      Server
--------                                                    --------
ClientHello
+ extended_psk=bskey_id
+ cert_with_extern_psk
+ client_cert_type=RawPublicKey
+ key_share                          -------->
                                                        ServerHello
                                            + extended_psk=bskey_id
                                             + cert_with_extern_psk
                                     + client_cert_type=RawPublicKey
                                                        + key_share
                                              {EncryptedExtensions}
                                               {CertificateRequest}
                                                      {Certificate}
                                                {CertificateVerify}
                                     <--------            {Finished}
{Certificate}
{CertificateVerify}
{Finished}                           -------->
[Application Data]                   <------->    [Application Data]
```

Legend:
- new stuff
- **present for dpp**
- existing exchange

TEAP w/DPP bootstrapping keys

no initial realm, just say "tls-pok"

```
     Authenticating Peer                              Authenticator
     -------------------- ------                       -------------------
                                              <---    EAP-Request/
                                                        Identity

       EAP-Response/
       Identity (TLS-POK)     --->
                                              <---    EAP-Request/
                                                       EAP-Type=TEAP
                                                      (TLS Start)
                                   .
                                   .
                                   .
          authenticate TEAP with TLS-DPP using bootstrapping key
                                   .
                                   .
                                   .
                                              <---    CSR Attrs TLV
       PKCS#10 TLV            --->
                                              <---    PKCS#7 TLV
```

Supplicant's subsequent connection uses provisioned certificate

Where we are and where to?

- Specification:
  - draft-friel-tls-eap-dpp-03
- Running code:
  - https://github.com/upros/mint
- Rough consensus:
  - adoption as a work item?