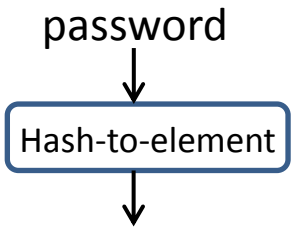
draft-harkins-tls-pwd

Dan Harkins
Aruba Networks

- What
  - Certificate-less ciphersuites
  - Authentication using a password
  - Resistance to off-line dictionary attack
  - No, it’s not patented
- Why (…not SRP)
  - EC support
  - Finite cyclic group is not fixed for each user, allowing a ciphersuite’s hash and cipher to be of commensurate strength
  - Parlay a simple passcode into a certificate using a RFC 5967–style request and a RFC 5751-style (degenerate, certs only) response
  - Same key exchange used in other protocol for data plane protection, nice to do the same thing for control plane protection– straight forward way to provide consistent, system-wide security
  - Commodity-purchased smart energy device with limited GUI
  - Misuse-resistant TLS

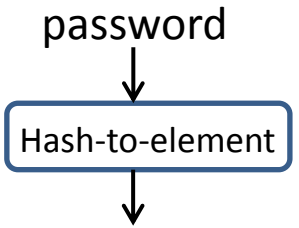
How it Works (very broadly)

Alice generates Password Element



PE = password element

Bob generates Password Element



PE = password element

Alice generates 2 random numbers

rnd-a, mask-a $\overset{\$}{<}$-- Z_q

Bob generates 2 random numbers

rnd-b, mask-b $\overset{\$}{<}$-- Z_q

Alice sends scalar and element to Bob

scalar-a = (rnd-a + mask-a) mod q -->
element-a = $PE^{-mask\text{-}a}$ mod p -->

Bob sends scalar and element to Alice

< -- scalar-b = (rnd-b + mask-b) mod q
< -- element-b = $PE^{-mask\text{-}b}$ mod p

Alice and Bob generate pre-master secret

$(PE^{scalar\text{-}b} * element\text{-}b)^{rnd\text{-}a} \bmod p = pre\text{-}master\text{-}secret = (PE^{scalar\text{-}a} * element\text{-}a)^{rnd\text{-}b} \bmod p$

How it works (changes to TLS)

```
enum { ff_pwd, ec_pwd } KeyExchangeAlgorithms;

struct {
   opaque salt<1..2^8-1>;
   opaque pwd_p<1..2^16-1>;
   opaque pwd_g<1..2^16-1>;
   opaque pwd_q<1..2^16-1>;
   opaque ff_sscalar<1..2^16-1>;
   opaque ff_selement<1..2^16-1>;
} ServerFFPWDParams;

struct {
   opaque salt<1..2^8-1>;
   ECParameters curve_params;
   opaque ec_sscalar<1..2^8-1>;
   ECPoint ec_selement;
} ServerECPWDParams;

struct {
   select (KeyExchangeAlgorithm) {
    case ec_pwd:
      ServerECPWDParams params;
    case ff_pwd:
      ServerFFPWDParams params;
   } ;
} ServerKeyExchange;
```

```
struct {
   opaque ff_cscalar<1..2^16-1>;
   opaque ff_celement<1..2^16-1>;
} ClientFFPWDParams;

struct {
   opaque ec_cscalar<1..2^8-1>;
   ECPoint ec_celement;
} ClientECPWDParams;

struct {
   select (KeyExchangeAlgorithm) {
    case ff_pwd:
      ClientFFPWDParams;
    case ec_pwd:
      ClientECPWDParams;
   }  exchange_keys;
} ClientKeyExchange;
```

- diff v00 v01
  - Salting password on server side
  - Mitigation of side channel attack on the process of hashing into an elliptic curve
  - Editorial changes: security considerations, justification/purpose

- OK, what do I want
  - Ask WG to accept document and move it forward as a Proposed Standard

  or, at the very least
  - Stable, published specification
  - Codepoints for pwd ciphersuites

CipherSuite TLS_FFCPWD_WITH_3DES_EDE_CBC_SHA = (TBD, TBD);
CipherSuite TLS_FFCPWD_WITH_AES_128_CBC_SHA = (TBD, TBD);
CipherSuite TLS_ECCPWD_WITH_AES_128_CBC_SHA = (TBD, TBD);
CipherSuite TLS_ECCPWD_WITH_AES_128_GCM_SHA256 = (TBD, TBD);
CipherSuite TLS_ECCPWD_WITH_AES_256_GCM_SHA384 = (TBD, TBD);
CipherSuite TLS_FFCPWD_WITH_AES_128_CCM_SHA = (TBD, TBD);
CipherSuite TLS_ECCPWD_WITH_AES_128_CCM_SHA = (TBD, TBD);
CipherSuite TLS_ECCPWD_WITH_AES_128_CCM_SHA256 = (TBD, TBD);
CipherSuite TLS_ECCPWD_WITH_AES_256_CCM_SHA384 = (TBD, TBD);