TLS PRF Considered Harmful

Issues with implementing Hardware Security Module Support for TLS

Multi-use PRF

- Issues came up with defining TLS mechanisms for PKCS11
- PRF is used in functions which produce both public (e.g. finish message tag, IVs) and private (key material) data
- The Key Derivation Function based on the PRF produces both public and private data
  - PRF (master_key, “key expansion”, otherpublicdata) => 2 mac, 2 encryption, 2 ivs.
  - Single key stream sliced into pieces
  - But what happens if we re-run the KDF with same params, but request 0 length mac, 0 length encryption keys? Key material leaks to IVs.
- This makes implementing hardware protection difficult
- If you can call the TLS_PRF directly with the same key, it’s even more trivial to extract key secrets.

Proposed solution

- Cryptographically isolate the three types of data produced by PRF
  - PRF is no longer a directly callable construct, but a part of the other constructs
  - Build a TLS_MAC construct which only produces public data (integrity tag for finished message)..
  - Build a TLS_KDF construct which only produces private key data
    - Mix-in length of keys and possibly types of keys. Provide a fixed label string for the construct that's mixed in with any user provided label.
  - Build a TLS_RANDOM_DATA construct which only produces pseudo-random data. (E.g. use this for IV's)
    - Provide a fixed label string for the construct that's mixed in with any user provided label.
- Ensure that these three constructs are cryptographically isolated from each other. I.e., even if same user label, key and random data are used in multiple constructs, the output streams can't be replicated by a different construct.

Details

- TLS_MAC ::= PRF (secret,”mac tls”, finished_label, Hash(handshake_messages))
  - Produces public integrity tag
- TLS_KDF ::= PRF (secret, “kdf tls”, label, $keyLength_1$[,$keyLength_N$*…],public_data)
  - Key stream is $\sum_{1..N}$ keyLength bytes split into N keys.
  - No mixing of key and IV’s on output.
- TLS_RANDOM_DATA ::= PRF (secret, “rand tls”, label, public_data)
  - Produces generic public data.

Threat Model

- Insider has access to HSM token credentials.
- Can reuse the keys in the token, but can't necessarily extract them.
- Prevent back door extraction of TLS session keys.