TLS Cached Info

Hannes Tschofenig
10th March 2015

Size of TLS Exchange: ECC Example

Client	Server	Size
ClientHello		121 bytes
	ServerHello	87 bytes
	Certificate	557 bytes
	Server Key Exchange	215 bytes
	Certificate Request	78 bytes
	Server Hello Done	4 bytes
Certificate		570 bytes
Client Key Exchange		138 bytes
Certificate Verify		80 bytes
Change Cipher Spec Protocol		1 byte
TLS Finished		40 bytes
	Change Cipher Spec	1 byte
	TLS Finished	40 bytes

- Example assumes a ECC-based ciphersuite with a 256 bit key.
- Only a single certificate is exchanged in the Certificate message.
- Result: 1932 bytes

- TLS exchanges lots of fairly static information.
  - Certificates
  - List of acceptable certificate authorities
  - OCSP responses

ARM

draft-ietf-tls-cached-info-18: Basic Concept

- Client starts with full exchange and caches information (e.g., server provided certificate)
- In a subsequent exchange the client indicates support for the "CachedInfo" extension in the ClientHello
  - Also includes the fingerprint of what it has cached.
- Server confirms that it supports "CachedInfo", what objects it supports, and omits transmission of specific payloads.
- Previous versions of the document only sent empty (or almost empty) payloads.
- State requirements: only at client side

ARM

Caching Client Certificates

- Server now also has to keep state and it has to be indexed.
  - Similarity to session resumption and session ticket.
- Option #1:
  - Server provides an *cache-id* in the full exchange.
  - Client repeats it again in a future exchange.
  - Server uses the session id to select the client certificate.
- Option #2:
  - Client indicates *hash of public key* to allow server to select the cached client certificate.
  - Similar to client_certificate_url (and could even re-use the structure with a new type).
- Problem: Linkability as with session ticket and session resumption

ARM