Binding Security Tokens to TLS Channels

A. Langley, Google Inc.

D. Balfanz, Google Inc.

A. Popov, Microsoft Corp.

The Problem: Bearer Tokens

- Web services generate various security tokens (HTTP cookies, OAuth tokens) for web applications to access protected resources.
- Currently these are bearer tokens, i.e. any party in possession of such token gains access to the protected resource.
- Attackers export bearer tokens from the user’s machine, present them to web services, and impersonate authenticated users.
- The idea of token binding is to prevent such attacks by creating a concept of long-lived, client-authenticated TLS channels, and cryptographically binding security tokens to these TLS channels.

Establishing a TLS Channel

- The user agent generates a private-public key pair (possibly within a secure hardware module, such as TPM) per target server.
- The user agent proves possession of the private key on every TLS connection to the target server.
- The proof of possession involves signing the tls_unique value for the TLS connection with the private key.
- The ID of such TLS channel is the corresponding public key.
- TLS channels are long-lived, i.e. they encompass multiple TLS connections and TLS sessions between a given client and server.
    - Privacy: users can reset TLS channel IDs at any time, e.g. when clearing cookies.

Preventing Token Theft

- When issuing a security token to a client that supports token binding, a server includes the ID of the client’s TLS channel in the token.
- Later on, when a client presents a security token containing a TLS channel ID, the server verifies that the TLS channel ID in the token matches the ID of the TLS channel established with the client.
- In the case of a mismatch, the server discards the token.
- In order to successfully export and replay a TLS channel-bound security token, the attacker needs to also be able to export the client’s private key, which is hard to do in the case of e.g. TPM-generated hardware backed key.

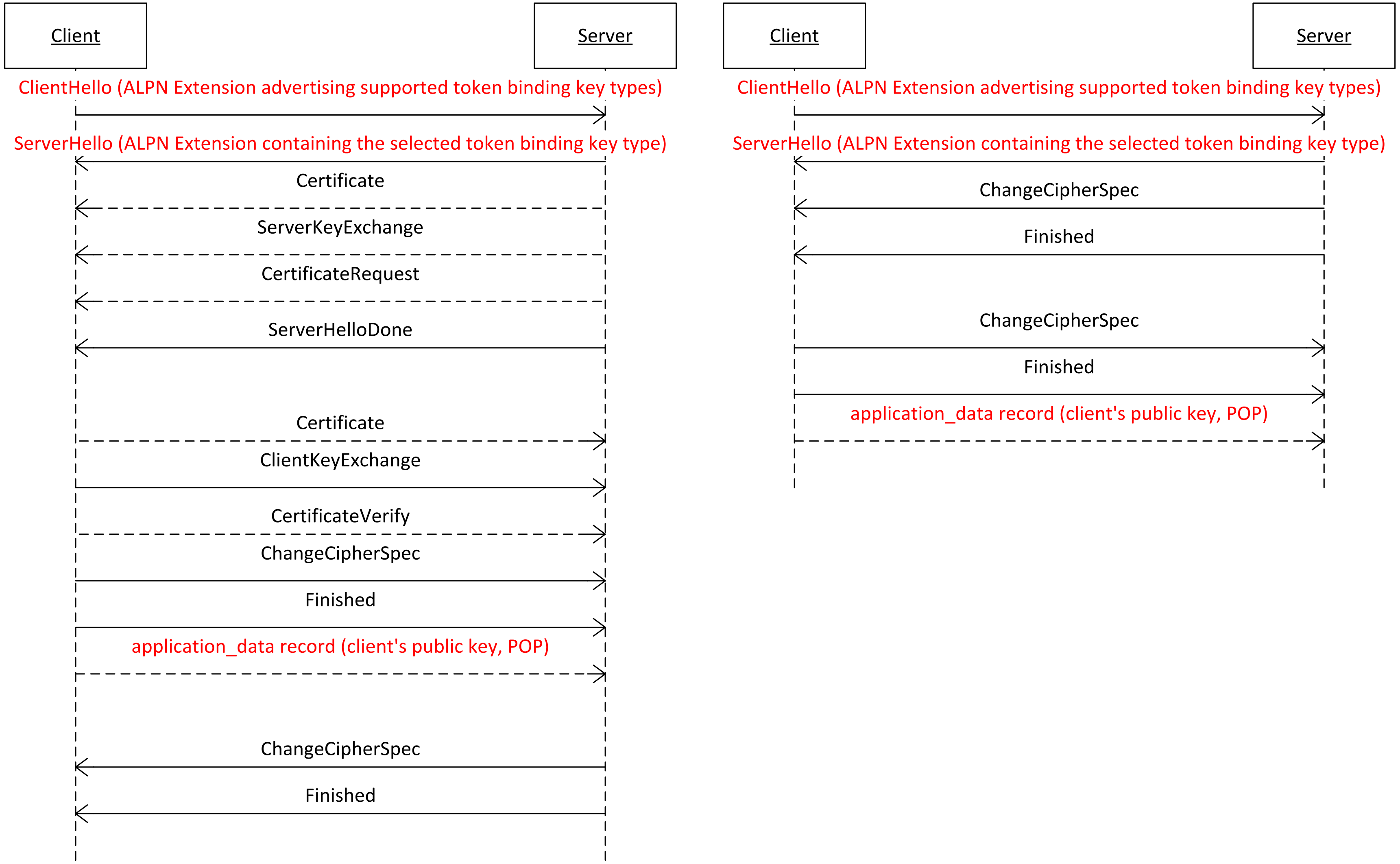
Token Binding Protocol

- We are introducing token binding as a new protocol, layered between TLS and the application protocols (such as HTTP and SMTP).
- The client and server use ALPN protocol IDs to negotiate the use of the token binding protocol, in addition to the actual application protocol.
- ALPN IDs are also used to negotiate the type of token binding key (ECDSA, RSA).
- This negotiation does not require TLS protocol changes, or additional round-trips.

Token Binding Protocol

- The token binding protocol consists of one message containing the proof of possession of a client-generated asymmetric key.
- This message is only sent if the client and server agree on the use of the token binding protocol and the token binding key type.
- The token binding message is sent within a TLS application_data record.
- When the parameters of the TLS handshake allow the use of FalseStart, this token binding message is sent immediately following (in the same round-trip with) the client’s Finished message.
- The token binding message can be followed by the messages of the negotiated application protocol (e.g. HTTP/2), and does not add network round-trips.

TLS Handshake And Token Binding Protocol



Client

Server

ClientHello (ALPN Extension advertising supported token binding key types)

ServerHello (ALPN Extension containing the selected token binding key type)

Certificate

ServerKeyExchange

CertificateRequest

ServerHelloDone

Certificate

ClientKeyExchange

CertificateVerify

ChangeCipherSpec

Finished

application_data record (client's public key, POP)

ChangeCipherSpec

Finished

Client

Server

ClientHello (ALPN Extension advertising supported token binding key types)

ServerHello (ALPN Extension containing the selected token binding key type)

ChangeCipherSpec

Finished

ChangeCipherSpec

Finished

application_data record (client's public key, POP)

Links And Contact Information

- Token binding Internet-Draft will be submitted after IETF 90.
- More background information: http://www.browserauth.net/

- Adam Langley agl@google.com
- Dirk Balfanz balfanz@google.com
- Andrei Popov andreipo@microsoft.com