Encrypted Client Hello

I-D: https://datatracker.ietf.org/doc/draft-ietf-tls-esni/
Editor’s Copy: https://github.com/tlswg/draft-ietf-tls-esni

IETF TLS Virtual Interim
April 27, 2020

Overview

Clients encrypt a “private” ClientHello and carry it in an outer “public” ClientHello.

- SNI, ALPN, and any future sensitive extensions get protection.
- Total ClientHello encryption binds all of its contents -- including PSK binders -- to the ClientHello ciphertext.

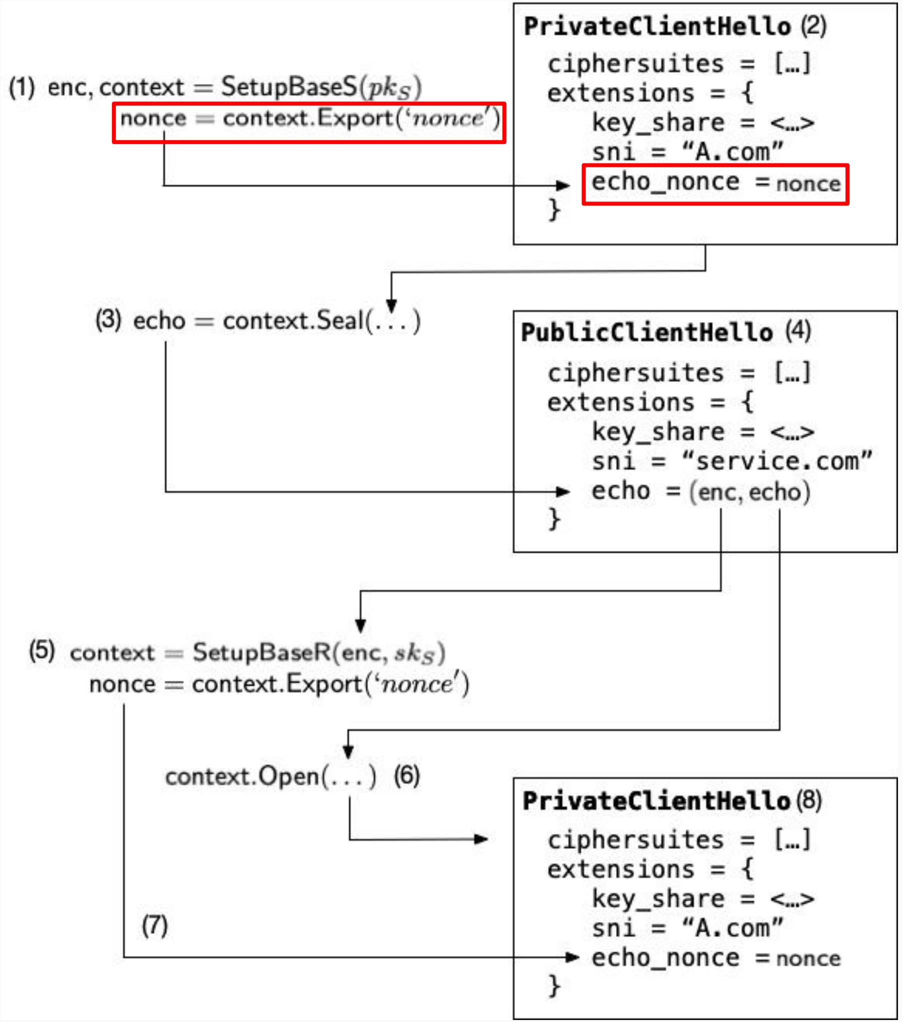
Servers use either the private or public ClientHello in the handshake (transcript).

- On decryption success, use the “private” ClientHello in the handshake.
- Otherwise*, use the “public” ClientHello in the handshake.

**Decryption failures currently trigger failure. GREASE may need to relax that requirement.*

Construction

1. Setup sender HPKE context and export a nonce.
2. Construct the “private” ClientHello, carrying the true SNI and nonce.
3. Encrypt the private ClientHello using the HPKE context.
4. Construct the “public” ClientHello, carrying the encrypted ClientHello.
5. Setup HPKE receiver HPKE context and export a nonce.
6. Decrypt the encrypted ClientHello (if possible).
7. Check the private ClientHello nonce against the derived value.



(1) enc, context = SetupBaseS(pk_S)
nonce = context.Export(‘$nonce$’)

```
PrivateClientHello (2)
  ciphersuites = […]
  extensions = {
     key_share = <…>
     sni = “A.com”
     echo_nonce = nonce
  }
```

(3) echo = context.Seal(. . .)

```
PublicClientHello (4)
  ciphersuites = […]
  extensions = {
     key_share = <…>
     sni = “service.com”
     echo = (enc, echo)
  }
```

(5) context = SetupBaseR(enc, sk_S)
nonce = context.Export(‘$nonce$’)

context.Open(. . .) (6)

(7)

```
PrivateClientHello (8)
  ciphersuites = […]
  extensions = {
     key_share = <…>
     sni = “A.com”
     echo_nonce = nonce
  }
```

Analysis Status

Security goal:

> *Adversary cannot distinguish between connections to A, B, or F, even in the event of origin server and long-term private key compromise. (Compromise of client-facing server private key leads to SNI leakage.)*

ProVerif model for TLS 1.3 handshake, including: core handshake, (fallback) public name authentication, and HRR*. *PSK support in progress.*

Updated report will come when analysis complete.

**Model is still running...*

Open Issues

Padding #209

ECHOConfig and HTTPSSVC #219, #216

HPKE code points #218

ECHOConfigContents.extensions #217

GREASE indistinguishability #177

Tunneling TLS 1.2 and below #214

Dependency: HPKE

Editor's copy is stable and nearly feature complete.

Several implementations exist: Go, Rust, C, Swift.

Corresponding CryptoVerif proof and analysis update from INRIA in progress.

Dependency: HTTPSSVC

In the process of converging with ECHO updates.

DNSOP indicates desire to WGLC before IETF 108.

Open question:

- Which draft should define the “echo” SvcParam?

Next Steps

Submit PRs to address open issues.

Update implementations for another round of experiments.

Questions?