TLS 1.3

draft-ietf-tls-tls13-18

Eric Rescorla

Mozilla

ekr@rtfm.com

Agenda

- Status
- WGLC issues
- Timeline

Status

- In WGLC with: draft-ietf-tls-tls13-18
- Quite a few interoperable implementations
  - draft-16 in Firefox Nightly, Chrome Dev/Canary, Cloudflare live
  - draft-18 in NSS, BoringSSL (under review), TLS-Tris (Cloudflare), Mint, Fizz (Facebook)
  - Other implementations under development

Interop Matrix

draft-ietf-tls-tls13-18 interop								
client ↓ | server →	NSS	BoringSSL	mint	BoGo	TLS-tris	Fizz	miTLS	ProtoTLS
NSS	1RZCH @ekr	1R @ekr	1RZ @ekr	1 @ekr	1 @ekr	1 @subodh		
BoringSSL	1R @svaldez	1RCKH @sva	1R @svaldez	1RCKH @sva	1 @svaldez	1 @subodh		
mint	1 @ekr	1 @svaldez	1RZK	1 @nharper		1 @subodh		
BoGo	1 @nharper	1RCKH @sva	1R @nharper	1RKH @nhar	1 @nharper			
TLS-tris								
Fizz								
miTLS								
ProtoTLS								
	Legend:							
	self-test	interop	known broker	unknown	N/A			
To Test:	1=1-RTT							
	R=Resumption							
	Z=0-RTT							
	C=Client Auth							
	K=KeyUpdate							
	H=HelloRetryRequest							

PR#748: Forbid negotiating $<$ TLS 1.2 with "supported_versions"

- Draft says that if "supported_versions" is present, it's the sole version negotiation mechanism
  - But you should list all the versions you support
  - In principle possible to negotiate TLS 1.1 via this mechanism
- Alternate design: require at least TLS 1.2 if you offer TLS 1.3
  - Forbid listing any value $<$ TLS 1.2 as client
  - Forbid negotiating any value $<$ TLS 1.2 on server

Issue#758: Exporters should call Hash() before HKDF-Expand-Label()

```
HKDF-Expand-Label(Secret, Label, HashValue, Length) =
     HKDF-Expand(Secret, HkdfLabel, Length)

struct {
    uint16 length = Length;
    opaque label<9..255> = "TLS 1.3, " + Label;
    opaque hash_value<0..255> = HashValue;
} HkdfLabel;
```

- Exporters are defined as;

```
HKDF-Expand-Label(Secret, label, context_value, key_length)
```

- This means you pass “context_value” as “hash”
- Confusing and imposes a 255-byte limit.
- Proposal:

```
HKDF-Expand-Label(Secret, label, Hash(context_value), key_length)
```

Issue#760: Certificate extension rules and client certs

- In draft-18 we put extensions in Certificate
  - Gated on ClientHello extensions
  - This doesn't make any sense for the cert for client authentication
- We have extensions in CertificateRequest
  - But they just filter on OID/value pair
  - Proposed resolution: add real extensions to CertificateRequest

Issue#760: CertificateRequest

```
struct {
      opaque certificate_request_context<0..2^8-1>;
      SignatureScheme
        supported_signature_algorithms<2..2^16-2>;
      DistinguishedName certificate_authorities<0..2^16-1>;
      Extension certificate_extensions<0..2^16-1>;
  } CertificateRequest;

  struct {
      opaque certificate_extension_oid<1..2^8-1>;
      opaque certificate_extension_values<0..2^16-1>;
  } OIDFilter;

  struct {
    OIDFilter filters<0..2^16-1>;
  } OIDFilterExtension;
```

- Previous CertificateRequest.extensions now are OID extensions

Issue#760: CertificateRequest extension variations

- Replace OIDs with extension IDs and flattten list
- Have two lists (OIDs and usual extensions)

- We should also make certificate_authorities an extension

Record Header

```
struct {
    ContentType opaque_type = 23; /* application_data */
    ProtocolVersion legacy_record_version = 0x0301; /* TLS v1.x */
    ...
} TLSCiphertext;
```

- This is three bytes of waste.
  - Would like to get rid of it
  - Questions about interop (with passive inspection middleboxes)
- Subtle point about 0-RTT failure transition
  - Steal a bit from the header
- Proposal in PR#762
  - We will take compat measurements in the next month or two
  - WG can then decide

Longer key lifetimes

Regardless of the actual record size, each 128-bit block encryption is performed with a unique 128-bit counter which is formed by the 96-bit IV and the 32-bit counter_block value called CB in NIST SP 800-38D under a given key as long as the number of encrypted records is not more than 2^64.

Assuming a user would like to limit the probability of a collision among 128-bit ciphertext-blocks under 1/2^32, the data limit of the ciphertext (or plaintext) is 2^(96/2) (= 2^48) 128-bit blocks which is 2^64 bytes.

Reading the 2nd paragraph of section 5.5, a user might feel that he/she needs to rekey a lot more quicker than he/she needs. Putting an unnecessarily low data limit of 2^24.5 full-size records (2^38.5 bytes) also creates an incorrect negative impression (in my opinion) about GCM.

I would like to request the working group to consider to revise the text.

- Anyone persuaded?

Timeline

Nov 20	WGLC Ends
Dec 1	draft-19 with all WGLC comments
Dec 31	Results of record header experiment
Jan 15	draft-20
Jan 31	End of cryptographic review period
Feb 10	Draft-20 (if needed) and pub request