Starting on TLS 1.3

Eric Rescorla

ekr@rtfm.com
Reminder: Objectives

• Encrypt as much of the handshake as possible

• Reduce handshake latency, with a target of 0-RTT for repeated handshakes and 1-RTT for “full” handshakes

• Reevaluate handshake contents

• Reevaluate record protection mechanisms (not discussed here)
## Rough time allocation

<table>
<thead>
<tr>
<th>Time</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>New handshake flows</td>
</tr>
<tr>
<td>7</td>
<td>Should we allow renegotiation</td>
</tr>
<tr>
<td>7</td>
<td>Should we stop supporting RSA?</td>
</tr>
<tr>
<td>7</td>
<td>Should we get rid of resumption?</td>
</tr>
<tr>
<td>7</td>
<td>Random sizes</td>
</tr>
<tr>
<td>2</td>
<td>Other?</td>
</tr>
</tbody>
</table>
New Handshake Flows

- Almost nothing here is new
- Ideas cribbed from
  - False Start
  - Snap Start
  - NPN
  - Marsh Ray’s encrypted handshake draft
  - A bunch of other people
- Writeup in: draft-rescorla-tls13-new-flows
  - Just posted (sorry about that!)
DISCLAIMER

DISCLAIMER: THIS IS A VERY ROUGH DRAFT. EVERYTHING HERE IS SUPER-HANDWAVY AND HASN’T REALLY HAD ANY SECURITY ANALYSIS. I DON’T PROMISE IT’S NOT VERY VERY WRONG BUT I WANTED TO BE ABLE TO HAVE AN EARLY DISCUSSION ABOUT DIRECTION.
Reminder: TLS 1.2 Full Handshake

ClientHello --------> ServerHello
Certificate*
ServerKeyExchange*
CertificateRequest*
<-------- ServerHelloDone
Certificate*
ClientKeyExchange
CertificateVerify*
[ChangeCipherSpec]
{Finished} --------> [ChangeCipherSpec]
<-------- {Finished}
{Application Data} <-------> {Application Data}
Reminder: TLS 1.2 Resumed Handshake

ClientHello --------> ServerHello
[ChangeCipherSpec]
<-------- {Finished}

[ChangeCipherSpec]
{Finished} -------->

{Application Data} <-------> {Application Data}
Reminder: False Start

ClientHello --------> ServerHello
Certificate*
ServerKeyExchange*
CertificateRequest*<-------- ServerHelloDone
Certificate*
ClientKeyExchange
CertificateVerify*
[ChangeCipherSpec]
{Finished}
{Application Data} -------->

[ChangeCipherSpec]
<-------- {Finished}
{Application Data} <-------> {Application Data}
Warm-up: Fast Track (sort-of)

ClientHello + CI
ClientKeyExchange --------> ServerHello + CI
Certificate*
ServerKeyExchange*
ServerHelloDone
[ChangeCipherSpec]
<-------- {Finished}

[ChangeCipherSpec]
{Finished}

{Application Data} -------->
{Application Data} <-------> {Application Data}
Warm-up: Falling back under prediction failure

ClientHello + CI
ClientKeyExchange --------> ServerHello
Certificate*
ServerKeyExchange*
CertificateRequest*
<-------- ServerHelloDone
Certificate*
ClientKeyExchange
CertificateVerify*
[ChangeCipherSpec]
{Finished} --------> [ChangeCipherSpec]
<-------- {Finished}
{Application Data} <-------> {Application Data}
Reduced RT handshake with privacy

ClientHello + CI
ClientKeyExchange --------> ServerHello[1] + CI
ServerKeyExchange*
[ChangeCipherSpec]
{ServerHello[2]}
{Certificate*}
{CertificateRequest*}
{ServerHelloDone}
<--------- {AlmostFinished}

[ChangeCipherSpec]
{Certificate*}
{CertificateVerify*}
{Finished}
{Application Data} -------->
<--------- {Finished}
{Application Data} <-------> {Application Data}
Reduced RT handshake with privacy

ClientHello[1] + CI
ClientKeyExchange --------> ServerHello[1]
<-------- ServerKeyExchange*
ClientHello[2] + CI // For consistency
ClientKeyExchange
[ChangeCipherSpec]
{ClientHello[3]} -------->
[ChangeCipherSpec]
  {ServerHello}
  {Certificate*}
  {ServerKeySignature*}
  {CertificateRequest*}
  {ServerHelloDone}
<-------- {AlmostFinished}

{Certificate*}
{CertificateVerify*}
{Finished}
{Application Data} -------->
<-------- {Finished}
{Application Data} <-------> {Application Data}
Zero RT Handshake (resumed)

ClientHello + CI + AR  
[ChangeCipherSpec]  
{Finished}  
{Application Data} ------->

ServerHello + CI + AR  
[ChangeCipherSpec]  
<--------  
{Finished}  

{Application Data}  
<-------->  
{Application Data}
Zero RT Handshake (non-resumed)

ClientHello[1] + CI + AR
ClientKeyExchange
{ClientHello[2]}
[ChangeCipherSpec]
{Certificate*}
{CertificateVerify*}
{Finished}
{Application Data} --------> 

ServerHello[1]
[ChangeCipherSpec]
{ServerHello[2]}
{ServerHelloDone}

<-------- {Finished}

{Application Data} <-------> {Application Data}
Zero-RTT Fallback Options

• How many fallback options should we have?

• Potentially
  – 0RTT resumed $\rightarrow$ 0RTT non-resumed $\rightarrow$ 1RTT Fast Track $\rightarrow$ Full handshake

• This seems awful complicated
  – Both for specification and for client
PFS just got complicated

- Resumption obviously doesn’t provide PFS
- But even the non-resumed handshake doesn’t provide it
  - Because it assumes a static server public key
- Options
  - Do a rehandshake
  - Have a two-phase handshake with the server supplying a key and client cuts over
Handwaving

ClientHello[1] + CI + AR
ClientKeyExchange
{ClientHello[2]}
[ChangeCipherSpec]
{Finished}
{Application Data} ------> ServerHello[1]

[ChangeCipherSpec]
{ServerHello[2]}
{Certificate}
{ServerKeyExchange}
{ServerHelloDone}

<-------- {{Finished}}

{{Application Data}} <-------> {{Application Data}}
Should we remove renegotiation?

• Raised by a number of people on the list

• Arguments for
  – Obvious point of complexity
  – We’ve had problems here before

• Arguments against
  – Change parameters
  – PFS refresh/rekey
  – To prevent cipher exhaustion (other ways to fix this)
  – Are we breaking people’s actual applications

• Discuss.
Should we stop supporting RSA?

- Obviously suboptimal performance characteristics
- Complexity
  - Doesn’t match the PFS pattern
  - See the handshakes above
- But everyone uses it...
  - And they have RSA certificates
  - Nice to have options
  - Discuss.
Should we remove resumption?

- Servers have gotten a lot faster
  - As have our cipher suites
- Arguments for
  - Remove complexity
- Arguments against
  - People definitely use it
  - And not everyone has gone to EC
  - Some devices have gotten much slower (DICE)
- Discuss.
Random values

- Current random values are (allegedly) 4 bytes of time and 28 bytes of randomness
- Make them shorter
  - Reduce entropy leakage from the PRNG
  - Is there an easier way to do this, e.g., separate PRNGs?
- Make them longer
  - Still waiting for a security analysis here
- Remove time
  - Potential fingerprinting service
  - But maybe useful for some stuff
  - Compatibility questions probably not a big issue
- Discuss.
Other topics?