0-RTT Key Establishment with Full Forward Secrecy

Felix Günther\textsuperscript{1} Britta Hale\textsuperscript{2} Tibor Jager\textsuperscript{3} Sebastian Lauer\textsuperscript{4}

\textsuperscript{1}Technischen Universität Darmstadt
\textsuperscript{2}NTNU Norwegian University of Science and Technology
\textsuperscript{3}Paderborn University
\textsuperscript{4}Ruhr-Universität Bochum

Eurocrypt 2017
Yes, it is possible!
Round-Trip Time (RTT)

1-RTT

2-RTT
KEY EXCHANGE LATENCY

TLS+TCP:

Client Hello

Server Hello

Enc. Extensions

Server Finished

Client Finished

Session Key: K

Session Key: K
KEY EXCHANGE LATENCY

TLS+TCP:

1-RTT

Client

TCP SYN

ClientHello

1-RTT

Server

Session Key: K

TCP SYN+ACK

ServerHello

Enc.Extensions

Server Finished

Client Finished

Session Key: K

Client Finished

Server Finished
**Key Exchange Latency**

**TLS + UDP:**

**Client:**
- 1-RTT
- ClientHello
- Client Finished

**Server:**
- TCP SYN
- TCP SYN+ACK
- ServerHello
- Enc.Extensions
- Server Finished

**Session Key:** $K$
Why not send cryptographically protected payload \textit{immediately}?
Zero Round-Trip Time (0-RTT)

Client

0-RTT

payload

Server
• **QUIC** by ...
• QUIC by ...

Google

(Quick UDP Internet Connections)
QUIC Protocol

Client

\[
\text{(prior session)}
\]

Server \((pk_{\text{sig}}, sk_{\text{sig}}), \ sk\)

\[
\text{config: } g^{sk}, \ \text{Sig}(sk_{\text{sig}}, g^{sk})
\]
QUIC Protocol

**Client**

\[ a \in \mathbb{Z}_q, \quad \mathbf{k} = g^a \cdot \mathbf{sk} \]

\[
\begin{array}{|c|}
\hline
\mathbf{g}^a \\
\hline
\end{array}
\]

\[
\text{Enc} (\mathbf{k}, \text{payload})
\]

\[ \mathbf{K} = g^{ab} \]

**Server \( (\mathbf{pk}_{\text{sig}}, \mathbf{sk}_{\text{sig}}), \mathbf{sk} \)**

\[
\begin{array}{|c|}
\hline
\text{config: } g^{\mathbf{sk}}, \text{Sig}(\mathbf{sk}_{\text{sig}}, g^{\mathbf{sk}}) \\
\hline
\end{array}
\]

\[ b \in \mathbb{Z}_q \]

\[
\begin{array}{|c|}
\hline
\text{Enc}(\mathbf{k}, g^b) \\
\hline
\end{array}
\]

\[ \mathbf{K} = g^{ab} \]
QUIC Protocol Issues: **Replay**

Client

\[ k = g^a \cdot sk \]

\[ g^a \]

\[ \text{Enc}(k, \text{payload}) \]

Server \( (pk_{\text{sig}}, sk_{\text{sig}}), sk \)
QUIC Protocol Issues: **Replay**

Client

$\text{Client}$

$\mathbf{k} = g^a \cdot \text{sk}$

$g^a$

$\text{Enc}(k, \text{payload})$

Server $(p_{\text{sig}}, s_{\text{sig}}), \text{sk}$

$g^a$

$\text{Enc}(k, \text{payload})$
QUIC Protocol Issues: Forward Secrecy

Client

\[ k = g^a \cdot sk \]

\[ g^a \]

Enc(k, payload)

Server \((pk_{\text{sig}}, sk_{\text{sig}}), sk\)
QUIC Protocol Issues:  

Forward Secrecy

\[ k = g^a \cdot sk \]

Client

\[ g^a \]

Enc\((k, \text{payload})\)

Server  \((pk_{\text{sig}}, sk_{\text{sig}}), sk\)
Forward Secrecy Threat Landscape:
Forward Secrecy Threat Landscape:

\[ K_0 \]
Forward Secrecy Threat Landscape:

- $K_0$
- $k_1$
Forward Secrecy Threat Landscape:

- $K_0$
- $k_1$
- $K_1$

Learn long-term key
Are past session keys secure?

Perfect Forward Secrecy:
Long-term key compromised
Past session keys remain secure
Forward Secrecy Threat Landscape:

- $K_0$
- $k_1$
- $K_1$
Forward Secrecy Threat Landscape:

- $K_0$
- $k_1$
- $K_1$

Learn long-term key
Forward Secrecy Threat Landscape:

Learn long-term key

$K_0$, $k_1$, $K_1$, $k_i$
Forward Secrecy Threat Landscape:

Learn long-term key

$K_0$, $k_1$, $K_1$, $k_i$, $K_i$
Forward Secrecy Threat Landscape:

Learn long-term key

\[ K_0 \quad k_1 \quad K_1 \]

Are past session keys secure?
Forward Secrecy Threat Landscape:

Are past session keys secure?

Perfect Forward Secrecy:
Long-term key compromised
Past session keys remain secure
QUIC

Learn long-term key

medium-lived

$K_0$, $k_1$, $K_1$, $k_i$, $K_i$
Is Perfect Forward Secrecy even possible for 0-RTT?
Yes!

• Full Forward Secrecy
• Replay protection
• Based on hierarchical ID-based key encapsulation mechanism (with selective security) and one-time signatures
• Flexible to different instantiations/assumptions
• post-quantum
• pairings
• etc...
Yes!

Our design:

- Full Forward Secrecy
Yes!

Our design:

- **Full Forward Secrecy**
- **Replay protection**
Yes!

Our design:

- **Full Forward Secrecy**
- **Replay protection**
- Based on hierarchical ID-based key encapsulation mechanism (with selective security) and one-time signatures
Yes!

Our design:

- Full Forward Secrecy
- Replay protection
- Based on hierarchical ID-based key encapsulation mechanism (with selective security) and one-time signatures
- Flexible to different instantiations/assumptions
  - post-quantum
  - pairings
  - etc...
Core idea:

Server: static public key – private key can be updated

→ Forward Secret KEM

→ Forward Secret 0-RTT KE
**Forward Secure 0-RTT KE**

Core idea:

\[
(C, K) \leftarrow \text{Enc}(pk) \\
K \leftarrow \text{Dec}(sk, C) \\
sk \leftarrow \text{Punct}(sk, C) \approx sk/C
\]
Hierarchical ID-Based KEM

\[ \text{sk} \]
\[ \text{sk}_0 \]
\[ \text{sk}_00 \]
\[ \text{sk}_000 \]
\[ \text{sk}_001 \]
\[ \text{sk}_0010 \]
\[ \text{sk}_0011 \]

\[ \text{sk}_01 \]
\[ \text{sk}_010 \]
\[ \text{sk}_011 \]

\[ \text{sk}_1 \]
\[ \text{sk}_10 \]
\[ \text{sk}_100 \]
\[ \text{sk}_101 \]

\[ \text{sk}_11 \]
\[ \text{sk}_110 \]
\[ \text{sk}_111 \]
Hierarchical ID-Based KEM
Puncturing private key $sk$
Puncturing private key \( sk \)
Puncturing private key $sk$

- Private key size $\approx \#\text{punctures} \times \log(\text{max #punctures/timeslot}) + \log(\#\text{timeslots})$
- $\#\text{punctures} = \#\text{sessions}$
Puncturing private key $sk$

- private key size $\approx \#\text{punctures} \times \log \left( \text{max \#punctures/timeslot} \right) + \log(\#\text{timeslots})$
- $\#\text{punctures} = \#\text{sessions}$
Purging the private key: time sync intervals $t_0, t_1, \ldots$
Purging the private key: time sync intervals $t_0, t_1, \ldots$
**Evaluation:**
Barreto-Naehrig elliptic curve P256, bilinear pairing, $pk$ 128bits, one-time sig $pk$ 256bits, timeslot length 30bits, avg. clock rate 3.2GHz

- Enc: ms
- Dec: seconds
- Puncturing: seconds
Evaluation:
Barreto-Naehrig elliptic curve P256, bilinear pairing, $pk$ 128bits, one-time sig $pk$ 256bits, timeslot length 30bits, avg. clock rate 3.2GHz

- Enc: ms
- Dec: seconds
- Puncturing: seconds $\rightarrow$ need only selective security

...Room for improvement?
Evaluation:
Barreto-Naehrig elliptic curve P256, bilinear pairing, \( pk \) 128bits, one-time sig pk 256bits, timeslot length 30bits, avg. clock rate 3.2GHz

- Enc: ms
- Dec: seconds
- Puncturing: seconds \( \rightarrow \) need only selective security
  
  ...Room for improvement?

... vs. Green and Myers S&P ’15:

- Any HIBE vs. specific bilinear groups
- CCA-secure in standard model vs. ROM
Summary

Now:

- FS 0-RTT key exchange + security model
- Generic construction + security proof
  (from one-time signatures and any hierarchical ID-based KEM with selective security)

Future:

- Optimize KEM key delegation
- Make it practical!
Summary

Now:

- FS 0-RTT key exchange + security model
- Generic construction + security proof
  (from one-time signatures and any hierarchical ID-based KEM with selective security)

Future:

- Optimize KEM key delegation
- Make it practical!
Questions
ACKNOWLEDGEMENTS

Some slide designs are based on presentations of the same work by co-authors Felix Günther and Tibor Jager