Code-based cryptography for secure communication

Tanja Lange

Eindhoven University of Technology & Academia Sinica

27 April 2022

Bonus slide: What is a KEM?

- When we explain public-key crypto we explain encryption,
  but in practice we do not use public-key crypto to encrypt a message.
- We encrypt a short symmetric key and use that to encrypt the actual message.

Bonus slide: What is a KEM?

- When we explain public-key crypto we explain encryption,
  but in practice we do not use public-key crypto to encrypt a message.
- We encrypt a short symmetric key and use that to encrypt the actual message.
- Symmetric keys are just 128-256 bits, this means padding

Bonus slide: What is a KEM?

- When we explain public-key crypto we explain encryption, but in practice we do not use public-key crypto to encrypt a message.
- We encrypt a short symmetric key and use that to encrypt the actual message.
- Symmetric keys are just 128-256 bits, this means padding ... and padding attacks.
- A key-encapsulation mechanism requires 3 algorithms:
  1. Key generation, generating a public-key private-key pair.
  2. Encapsulation, taking a public key, producing key k and ciphertext. k is then used in symmetric crypto.
  3. Decapsulation, taking a private key and a ciphertext, producing key.

Bonus slide: What is a KEM?

- When we explain public-key crypto we explain encryption, but in practice we do not use public-key crypto to encrypt a message.
- We encrypt a short symmetric key and use that to encrypt the actual message.
- Symmetric keys are just 128-256 bits, this means padding ... and padding attacks.
- A key-encapsulation mechanism requires 3 algorithms:
  1. Key generation, generating a public-key private-key pair.
  2. Encapsulation, taking a public key, producing key k and ciphertext. k is then used in symmetric crypto.
  3. Decapsulation, taking a private key and a ciphertext, producing key.
- Can think of DH as a KEM:

$$\mathrm{KEM} - \mathrm{Enc}(g^a) = (g^{ra}, g^r) = (k, c)$$

- Anna-Lena Horlemann explained Niederreiter for encryption.
- Niederreiter as KEM takes public key, picks random vector of length n, weight t.

How does TLS (`https`) work?

Client | **Server**

$(\mathsf{sk}_C, \mathsf{pk}_C) \leftarrow\$ \mathsf{KGen}$

$$\xrightarrow{\mathsf{pk}_C}$$

$(\mathsf{sk}_S, \mathsf{pk}_S) \leftarrow\$ \mathsf{KGen}$

$k \leftarrow \mathsf{DH}(\mathsf{sk}_S, \mathsf{pk}_C)$

$$\xleftarrow{\mathsf{pk}_S}$$

$k \leftarrow \mathsf{DH}(\mathsf{sk}_C, \mathsf{pk}_S)$

$$\xrightarrow[\text{proves } C \text{ knows } k]{\text{stuff encrypted using } k}$$

$\Sigma \leftarrow \mathsf{Sig}(\text{everything sent so far})$

this uses a long-term signing key

$$\xleftarrow[\text{stuff encrypted using } k]{\Sigma}$$

How does PQC affect protocols?

- Length fields don't fit.

How does PQC affect protocols?

- Length fields don't fit.
  $\Rightarrow$ Restrict to systems that fit, if any,
  or keep pre-quantum algorithm next to PQC one,
  putting PQC part into the payload.

How does PQC affect protocols?

- Length fields don't fit.
  $\Rightarrow$ Restrict to systems that fit, if any,
  or keep pre-quantum algorithm next to PQC one,
  putting PQC part into the payload.
- Speed, resources.
  Combined schemes take about twice the time.

How does PQC affect protocols?

- Length fields don't fit.
  $\Rightarrow$ Restrict to systems that fit, if any,
  or keep pre-quantum algorithm next to PQC one,
  putting PQC part into the payload.
- Speed, resources.
  Combined schemes take about twice the time.
  Most experiments don't look so devastating.
- Interface mismatch – KEM instead of DH,

How does PQC affect protocols?

- Length fields don't fit.
  $\Rightarrow$ Restrict to systems that fit, if any,
  or keep pre-quantum algorithm next to PQC one,
  putting PQC part into the payload.
- Speed, resources.
  Combined schemes take about twice the time.
  Most experiments don't look so devastating.
- Interface mismatch – KEM instead of DH,
  $\Rightarrow$ Shoehorning PQC into current systems may
  prioritize weaker systems.

How does PQC affect protocols?

- Length fields don't fit.
  $\Rightarrow$ Restrict to systems that fit, if any,
  or keep pre-quantum algorithm next to PQC one,
  putting PQC part into the payload.
- Speed, resources.
  Combined schemes take about twice the time.
  Most experiments don't look so devastating.
- Interface mismatch – KEM instead of DH,
  $\Rightarrow$ Shoehorning PQC into current systems may
  prioritize weaker systems.
- Validation and certification schemes are not updated.

How does PQC affect protocols?

- Length fields don't fit.
  $\Rightarrow$ Restrict to systems that fit, if any,
  or keep pre-quantum algorithm next to PQC one,
  putting PQC part into the payload.
- Speed, resources.
  Combined schemes take about twice the time.
  Most experiments don't look so devastating.
- Interface mismatch – KEM instead of DH,
  $\Rightarrow$ Shoehorning PQC into current systems may
  prioritize weaker systems.
- Validation and certification schemes are not updated.
  $\Rightarrow$ Combine pre-and post-quantum schemes,
  certification only applies to pre-quantum scheme.

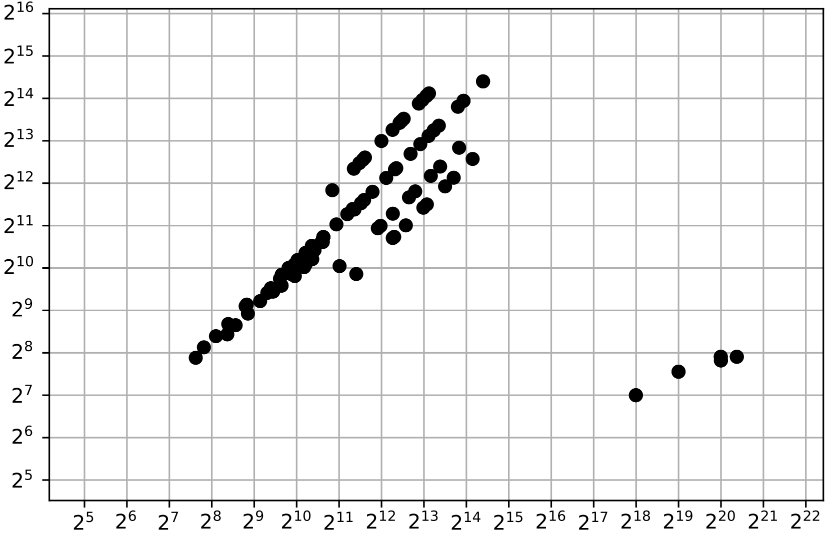
How does PQC affect protocols?

- Length fields don't fit.
  ⇒ Restrict to systems that fit, if any,
  or keep pre-quantum algorithm next to PQC one, putting PQC part into the payload.
- Speed, resources.
  Combined schemes take about twice the time.
  Most experiments don't look so devastating.
- Interface mismatch – KEM instead of DH,
  ⇒ Shoehorning PQC into current systems may prioritize weaker systems.
- Validation and certification schemes are not updated.
  ⇒ Combine pre-and post-quantum schemes,
  certification only applies to pre-quantum scheme. For such *hybrid* schemes, ensure that as strong as strongest not as weak as weakest.

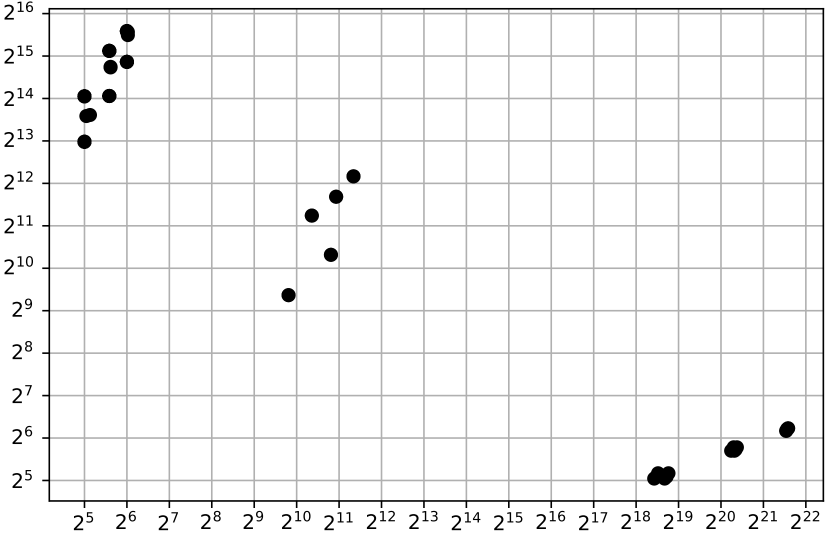
How does PQC affect protocols?

- Length fields don't fit.
  ⇒ Restrict to systems that fit, if any,
  or keep pre-quantum algorithm next to PQC one,
  putting PQC part into the payload.
- Speed, resources.
  Combined schemes take about twice the time.
  Most experiments don't look so devastating.
- Interface mismatch – KEM instead of DH,
  ⇒ Shoehorning PQC into current systems may
  prioritize weaker systems.
- Validation and certification schemes are not updated.
  ⇒ Combine pre-and post-quantum schemes,
  certification only applies to pre-quantum scheme. For such *hybrid* schemes,
  ensure that as strong as strongest not as weak as weakest.
- New security assumptions, new proofs, lots of new code.

Encryption (KEM): ciphertext size (vertical) vs. public-key size (horizontal)



Signatures: signature size (vertical) vs. public-key size (horizontal)



Deployment issues & solutions

- Different recommendations for rollout in different risk scenarios:
  - Use most efficient systems with ECC or RSA, to ease usage and gain familiarity.
  - Use most conservative systems (possibly with ECC), to ensure that data really remains secure.
- Protocol integration and implementation problems:
  - Key sizes or message sizes are larger for post-quantum systems, but IPv6 guarantees only delivery of $\leq$ 1280-byte packets, TLS software has length limits, etc.
  - Google experimented with larger keys and noticed delays and dropped connections.
  - Long-term keys require extra care (reaction attacks).
- Some libraries exist, quality is getting better.
- Google and Cloudflare are running some experiments of including post-quantum systems into TLS.

Deployment issues & solutions

- Different recommendations for rollout in different risk scenarios:
  - Use most efficient systems with ECC or RSA,
    to ease usage and gain familiarity.
  - Use most conservative systems (possibly with ECC),
    to ensure that data really remains secure.
- Protocol integration and implementation problems:
  - Key sizes or message sizes are larger for post-quantum systems,
    but IPv6 guarantees only delivery of $\leq$ 1280-byte packets,
    TLS software has length limits, etc.
  - Google experimented with larger keys and noticed delays and dropped connections.
  - Long-term keys require extra care (reaction attacks).
- Some libraries exist, quality is getting better.
- Google and Cloudflare are running some experiments of including post-quantum systems into TLS.
- These all use lattice based schemes.

Deployment issues & solutions

- Different recommendations for rollout in different risk scenarios:
  - Use most efficient systems with ECC or RSA, to ease usage and gain familiarity.
  - Use most conservative systems (possibly with ECC), to ensure that data really remains secure.
- Protocol integration and implementation problems:
  - Key sizes or message sizes are larger for post-quantum systems, but IPv6 guarantees only delivery of $\leq$ 1280-byte packets, TLS software has length limits, etc.
  - Google experimented with larger keys and noticed delays and dropped connections.
  - Long-term keys require extra care (reaction attacks).
- Some libraries exist, quality is getting better.
- Google and Cloudflare are running some experiments of including post-quantum systems into TLS.
- These all use lattice based schemes. How about the code-based finalist?

NIST PQC submission Classic McEliece

No patents.
Shortest ciphertexts.
Fast open-source constant-time software implementations.
Very conservative system, expected to last; has strongest security track record.

Sizes with similar post-quantum security to AES-128, AES-192, AES-256:

Metric	**mceliece348864**	**mceliece460896**	**mceliece6960119**
Public-key size	261120 bytes	524160 bytes	1047319 bytes
Secret-key size	6452 bytes	13568 bytes	13908 bytes
Ciphertext size	128 bytes	188 bytes	226 bytes
Key-generation time	52415436 cycles	181063400 cycles	417271280 cycles
Encapsulation time	43648 cycles	77380 cycles	143908 cycles
Decapsulation time	130944 cycles	267828 cycles	295628 cycles

See `https://classic.mceliece.org` for authors, details & parameters.

Key issues for McEliece

Key issues for McEliece

BIG PUBLIC KEYS.

Key issues for McEliece

Users send big data anyway. We have lots of bandwidth. Maybe 1MB keys are okay.

Each client spends a small fraction of a second generating new ephemeral 1MB key.

Key issues for McEliece

Users send big data anyway. We have lots of bandwidth. Maybe 1MB keys are okay.

Each client spends a small fraction of a second generating new ephemeral 1MB key.

But: If any client is allowed to send a new ephemeral 1MB McEliece key to server, an attacker can easily flood server's memory. **This invites DoS attacks.**
(DoS = Denial of Service)

Key issues for McEliece

Users send big data anyway. We have lots of bandwidth. Maybe 1MB keys are okay.

Each client spends a small fraction of a second generating new ephemeral 1MB key.

But: If any client is allowed to send a new ephemeral 1MB McEliece key to server,
an attacker can easily flood server's memory. **This invites DoS attacks.**
(DoS = Denial of Service)

Our goal: Eliminate these attacks by eliminating all per-client storage on server.

Goodness, what big keys you have!

Public keys look like this:

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix}$$

Left part is $(n-k) \times (n-k)$ identity matrix (no need to send).
Right part is random-looking $(n-k) \times k$ matrix.
E.g. $n = 6960$, $k = 5413$, so $n - k = 1547$.

Goodness, what big keys you have!

Public keys look like this:

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix}$$

Left part is $(n-k) \times (n-k)$ identity matrix (no need to send).
Right part is random-looking $(n-k) \times k$ matrix.
E.g. $n = 6960$, $k = 5413$, so $n - k = 1547$.
Encryption xors secretly selected columns, e.g.

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Can servers avoid storing big keys?

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix} = (I_{n-k}|K')$$

Encryption xors secretly selected columns.

With some storage and trusted environment:
Receive columns of K' one at a time, store and update partial sum.

Can servers avoid storing big keys?

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix} = (I_{n-k}|K')$$

Encryption xors secretly selected columns.

With some storage and trusted environment:
Receive columns of K' one at a time, store and update partial sum.

On the real Internet, without per-client state:

Can servers avoid storing big keys?

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix} = (I_{n-k}|K')$$

Encryption xors secretly selected columns.

With some storage and trusted environment:
Receive columns of K' one at a time, store and update partial sum.

On the real Internet, without per-client state:
Don't reveal intermediate results!
Which columns are picked is the secret message!
Intermediate results show whether a column was used or not.

McTiny

Partition key

$$K' = \begin{pmatrix} K_{1,1} & K_{1,2} & K_{1,3} & \dots & K_{1,\ell} \\ K_{2,1} & K_{2,2} & K_{2,3} & \dots & K_{2,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ K_{r,1} & K_{r,2} & K_{r,3} & \dots & K_{r,\ell} \end{pmatrix}$$

- Each submatrix $K_{i,j}$ small enough to fit (including header) into network packet.
- Client feeds the $K_{i,j}$ to server & handles storage for the server.
- Server computes $K_{i,j}e_j$, puts result into cookie.
- Cookies are encrypted by server to itself using some temporary symmetric key (same key for all server connections).
  No per-client memory allocation.
- Cookies also encrypted & authenticated to client.
- Client sends several $K_{i,j}e_j$ cookies, receives their combination.
- More stuff to avoid replay & similar attacks.

McTiny

Partition key

$$K' = \begin{pmatrix} K_{1,1} & K_{1,2} & K_{1,3} & \dots & K_{1,\ell} \\ K_{2,1} & K_{2,2} & K_{2,3} & \dots & K_{2,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ K_{r,1} & K_{r,2} & K_{r,3} & \dots & K_{r,\ell} \end{pmatrix}$$

- Each submatrix $K_{i,j}$ small enough to fit (including header) into network packet.
- Client feeds the $K_{i,j}$ to server & handles storage for the server.
- Server computes $K_{i,j}e_j$, puts result into cookie.
- Cookies are encrypted by server to itself using some temporary symmetric key (same key for all server connections).
  No per-client memory allocation.
- Cookies also encrypted & authenticated to client.
- Client sends several $K_{i,j}e_j$ cookies, receives their combination.
- More stuff to avoid replay & similar attacks.
- Several round trips, but no per-client state on the server.

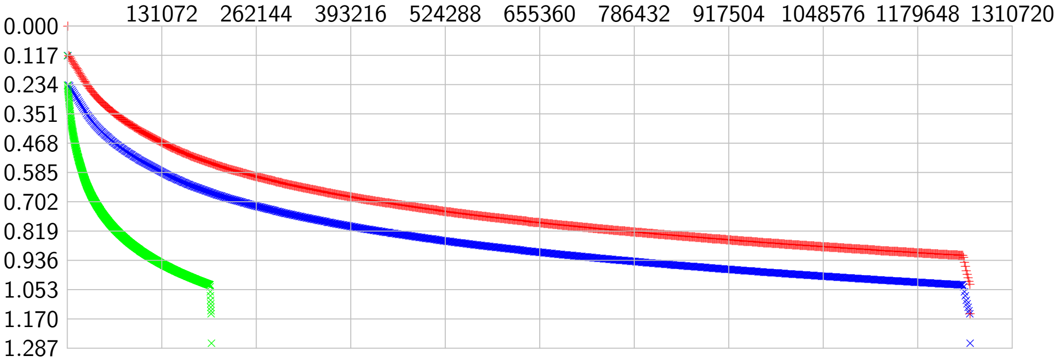
Packet sizes in each phase of mceliece6960119

phase		bytes/packet	packets	bytes
0	query	810	1	810
	reply	121	1	121
1	query	1226	952	1 167 152
	reply	140	952	133 280
2	query	1185	17	20 145
	reply	133	17	2 261
3	query	315	1	315
	reply	315	1	315
	queries		971	1 188 422
	replies		971	135 977

Entries count only application-layer data and not counting UDP/IP/Ethernet overhead.

A public key is 1 047 319 bytes.

Measurements of our software (`https://mctiny.org`)



Client time vs. bytes sent, bytes acknowledged, bytes in acknowledgments.
Curve shows packet pacing from our new user-level congestion-control library.

WireGuard

- WireGuard is a VPN protocol.
- VPN stands for Virtual Private Network.

WireGuard

- WireGuard is a VPN protocol.
- VPN stands for Virtual Private Network. (Not that that explains much)
- Relevant distinction from TLS scenario: Client connects to known, fixed server.

WireGuard

- WireGuard is a VPN protocol.
- VPN stands for Virtual Private Network. (Not that that explains much)
- Relevant distinction from TLS scenario: Client connects to known, fixed server.
- In WireGuard the server is known by a long-term DH key.
- This public key is exchanged out of band.

WireGuard

Client		**Server**
knows LTpk_S		has $\mathrm{LTsk}_S, \mathrm{LTpk}_S$

.... Actual start ....

$(\mathrm{sk}_C, \mathrm{pk}_C) \leftarrow_{\$} \mathrm{KGen}$

$k_1 \leftarrow \mathrm{DH}(\mathrm{sk}_C, \mathrm{LTpk}_S)$ $\quad \xrightarrow[\text{something with } k_1]{\mathrm{pk}_C} \quad$ $k_1 \leftarrow \mathrm{DH}(\mathrm{LTsk}_S, \mathrm{pk}_C), \text{check } k_1$

$(\mathrm{sk}_S, \mathrm{pk}_S) \leftarrow_{\$} \mathrm{KGen}$

$k_2 \leftarrow \mathrm{H}(k_1, \mathrm{DH}(\mathrm{sk}_S, \mathrm{pk}_C))$

$\xleftarrow[\text{something with } k_2]{\mathrm{pk}_S}$

$k_2 \leftarrow \mathrm{H}(k_1, \mathrm{DH}(\mathrm{sk}_C, \mathrm{pk}_S))$

$\xrightarrow[\text{or keys derived from } k_2]{\text{content encrypted with } k_2}$

'WireGuard' with KEMs

Client		**Server**
knows KEM LTpk_S		has KEM $\mathrm{LTsk}_S, \mathrm{LTpk}_S$

. . . . Actual start . . . .

$(\mathrm{sk}_C, \mathrm{pk}_C) \xleftarrow{\$} \mathrm{KGen}$		
$(k_1, c_1) \leftarrow \mathrm{KEM{-}Enc}(\mathrm{LTpk}_S)$	$\xrightarrow[\mathrm{Enc}(k_1, \mathrm{pk}_C)]{c_1}$	$k_1 \leftarrow \mathrm{KEM{-}Dec}(\mathrm{LTsk}_S, c_1)$
		$\mathrm{pk}_C \leftarrow \mathrm{Dec}(k_1, \mathrm{Enc}(k_1, \mathrm{pk}_C))$
		$(k_2', c_2) \leftarrow \mathrm{KEM{-}Enc}(\mathrm{pk}_C)$
	$\xleftarrow[\text{stuff to verify}]{c_2}$	$k_2 \leftarrow \mathrm{H}(k_1, k_2')$
$k_2' \leftarrow \mathrm{KEM{-}Dec}(\mathrm{sk}_C, c_2)$		
$k_2 \leftarrow \mathrm{H}(k_1, k_2')$	$\xrightarrow[\text{or keys derived from } k_2]{\text{content encrypted with } k_2}$	

Post-quantum WireGuard https://eprint.iacr.org/2020/379

- In Post-Quantum WireGuard the server is known by a long-term KEM key LTpk_S.
- This public key is exchanged out of band.

Post-quantum WireGuard https://eprint.iacr.org/2020/379

- In Post-Quantum WireGuard the server is known by a long-term KEM key LTpk_S.
- This public key is exchanged out of band.
- This key can be large, we do not pay for it in bandwidth!

Post-quantum WireGuard https://eprint.iacr.org/2020/379

- In Post-Quantum WireGuard the server is known by a long-term KEM key LTpk_S.
- This public key is exchanged out of band.
- This key can be large, we do not pay for it in bandwidth!
- c_1 is a KEM ciphertext, this should be small.
- Short-term KEM public key pk_C is sent and should be small.

Post-quantum WireGuard https://eprint.iacr.org/2020/379

- In Post-Quantum WireGuard the server is known by a long-term KEM key LTpk_S.
- This public key is exchanged out of band.
- This key can be large, we do not pay for it in bandwidth!
- c_1 is a KEM ciphertext, this should be small.
- Short-term KEM public key pk_C is sent and should be small.
- Post-quantum WireGuard uses Classic McEliece for the long-term KEM and lattice-based Saber for the short-term KEM.
- This showcases the small ciphertexts of Classic McEliece and does not notice the public-key size.

Different deployment strategy

PQConnect: An Automated Boring Protocol for Quantum-Secure Tunnels

- Do not patch PQC onto existing network protocols, but add a new layer with superior security.

Different deployment strategy

PQConnect: An Automated Boring Protocol for Quantum-Secure Tunnels

- Do not patch PQC onto existing network protocols, but add a new layer with superior security.
- Can be gradually deployed.
- Add support for VPN-like tunnels to clients and servers

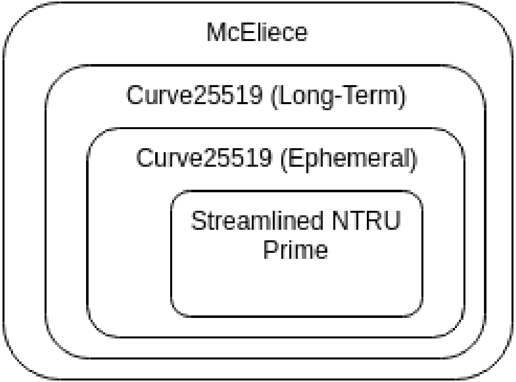
Different deployment strategy

PQConnect: An Automated Boring Protocol for Quantum-Secure Tunnels

- Do not patch PQC onto existing network protocols, but add a new layer with superior security.
- Can be gradually deployed.
- Add support for VPN-like tunnels to clients and servers but do this to the endpoints, not some intermediate VPN server.
- PQConnect is designed for security, handshake and ratcheting proven using Tamarin prover (formal verification tool).
- Use Curve25519 (pre-quantum) and Classic McEliece (conservative PQC) for long-term identity keys.
- Use Curve25519 (pre-quantum) and lattice-based Streamlined NTRU Prime (PQC) for ephemeral keys.

PQConnect handshake: Nesting schemes

Most conservative system on the outside.



Attacker can see long-term Curve25519 identity key,
can break it with a quantum computer,
but cannot obtain DH value as client's share is wrapped.

PQConnect handshake: Handling McElice keys

- McEliece is used for the long-term key, i.e., this key does not change.
- Store key for frequently visited sites (Google, Gmail, Facebook, Twitter,...)
- Link key download to obtaining IP address via DNS lookup.
  This is how the client know where to connect to. PQConnect piggy-backs on this with a hash of the key and info on where to download the key.

PQConnect handshake: Handling McElice keys

- McEliece is used for the long-term key, i.e., this key does not change.
- Store key for frequently visited sites (Google, Gmail, Facebook, Twitter,...)
- Link key download to obtaining IP address via DNS lookup.
  This is how the client know where to connect to. PQConnect piggy-backs on this with a hash of the key and info on where to download the key.
- Split key as in McTiny, download in small chunks and verify with hash;
  PQConnect also includes the Curve25519 key (256 bits, just a small corner).
- PQConnect benefits from small McEliece ciphertexts.
- Combine with lattice-based crypto for balance in ciphertext and public key size; security concerns alleviated by nesting.
- More information on protocol:
  `https://research.tue.nl/en/studentTheses/pqconnect`
  Paper and software still forthcoming.

Key ratchet advances by message and time

Complete protocol follows picture on previous slide.
All systems linked together to generate initial key c_0.
Keys are updated (ratcheted) to protect against later decryption by theft of computer equipment.
Immediately advance ratchet in 3 ways:

- New epoch master key: c_1.
- New branch keys: $c_{0,1}, c_{0,2}$.
- New message key: $c'_{0,1}$.

Delete key as soon as no longer needed.
Message keys can deal with delayed transmissions.

$$
\begin{tikzcd}
c_0 \arrow[dd] \arrow[dr] & & & & \\
& c_{0,1} \arrow[d] \arrow[dr] & & & \\
c_1 \arrow[dd] \arrow[dr] & c'_{0,1} & c_{0,2} \arrow[d] \arrow[dr] & & \\
& c_{1,1} \arrow[d] \arrow[dr] & c'_{0,2} & c_{0,3} \arrow[d] \arrow[dr] & \\
c_2 \arrow[dd] \arrow[dr] & c'_{1,1} & c_{1,2} \arrow[d] \arrow[dr] & c'_{0,3} & \ddots \\
& c_{2,1} \arrow[d] \arrow[dr] & c'_{1,2} & c_{1,3} \arrow[d] \arrow[dr] & \\
\vdots & c'_{2,1} & c_{2,2} \arrow[d] \arrow[dr] & c'_{1,3} & \ddots \\
& & c'_{2,2} & c_{2,3} \arrow[d] \arrow[dr] & \\
& & & c'_{2,3} & \ddots
\end{tikzcd}
$$

Further information

- https://pqcrypto.org our overview page.
- PQCrypto 2016, PQCrypto 2017, PQCrypto 2018, PQCrypto 2019, PQCrypto 2020, PQCrypto 2021 with many slides and videos online.
- https://pqcrypto.eu.org: PQCRYPTO EU Project.
  - PQCRYPTO recommendations.
  - Free software libraries (libpqcrypto, pqm4, pqhw).
  - Many reports, scientific articles, (overview) talks.
- YouTube channel Tanja Lange: Post-quantum cryptography.
- https://2017.pqcrypto.org/school: PQCRYPTO summer school with 21 lectures on video, slides, and exercises.
- https://2017.pqcrypto.org/exec and https://pqschool.org/index.html: Executive school (less math, more perspective).
- Quantum Threat Timeline from Global Risk Institute, 2019; 2021 update.
- Status of quantum computer development (by German BSI).
- NIST PQC competition.