Google | Privacy, Safety & Security

PQC at Google

PQ Crypto 2023

August 17, 2023

Introduction

Sophie Schmieg

Senior Staff Cryptography Engineer

- PhD in Algebraic Geometry
- Leading Google's ISE Crypto team

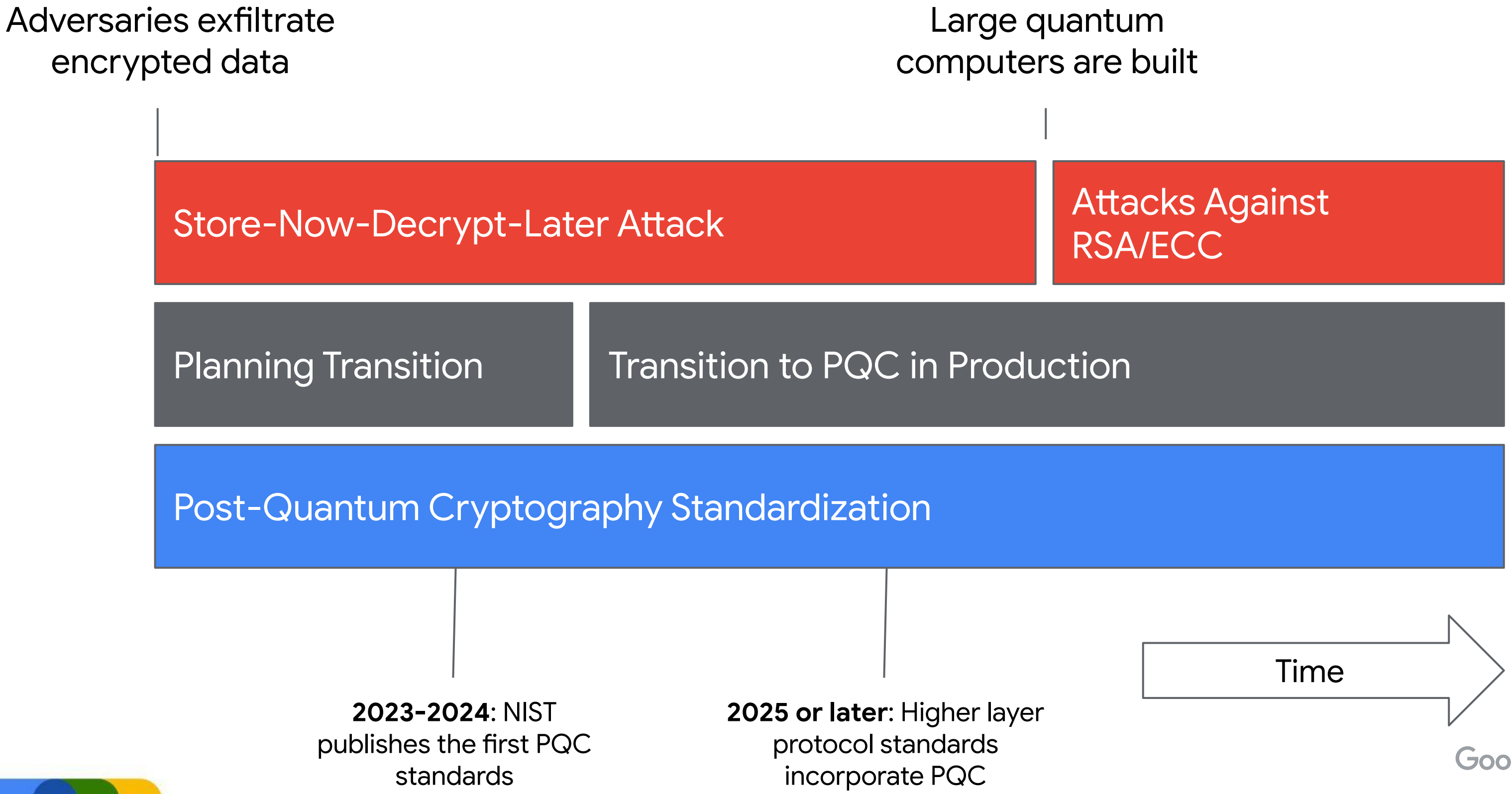
Agenda

1. The Post-Quantum Threat Model
2. Case Study: PQ ALTS
3. Primitives and Standards

The Post-Quantum Threat Model

and how it applies to Google

Why is this important now?



Adversaries exfiltrate encrypted data

Large quantum computers are built

Store-Now-Decrypt-Later Attack

Attacks Against RSA/ECC

Planning Transition

Transition to PQC in Production

Post-Quantum Cryptography Standardization

2023-2024: NIST publishes the first PQC standards

2025 or later: Higher layer protocol standards incorporate PQC

Time

The Post-Quantum Threat Model

Asymmetric Encryption

Used mainly for encryption in transit, allows sending confidential messages to another party, by negotiating a shared key.

Digital Signatures

Used very widely, allows for proof of that the private key owner has endorsed a specific input.

Symmetric Cryptography

Used very widely, especially for encryption at rest and for actually transferring data for encryption in transit, allows to encrypt data with a key.

Fancy Cryptography

Various other uses of cryptography, often to accomplish complicated privacy guarantees.

The Post-Quantum Threat Model

Asymmetric Encryption

Used mainly for encryption in transit, allows sending confidential messages to another party, by negotiating a shared key.

Digital Signatures

Used very widely, allows for proof of that the private key owner has endorsed a specific input.

Symmetric Cryptography

Used very widely, especially for encryption at rest and for actually transferring data for encryption in transit, allows to encrypt data with a key.

Fancy Cryptography

Various other uses of cryptography, often to accomplish complicated privacy guarantees.

The Post-Quantum Threat Model

Asymmetric Encryption

Used mainly for encryption in transit, allows sending confidential messages to another party, by negotiating a shared key.

Digital Signatures

Used very widely, allows for proof of that he private key owner has endorsed a specific input.

Symmetric Cryptography

Used very widely, especially for encryption at rest and for actually transferring data for encryption in transit, allows to encrypt data with a key.

Fancy Cryptography

Various other uses of cryptography, often to accomplish complicated privacy guarantees.

The Post-Quantum Threat Model

Asymmetric Encryption

Used mainly for encryption in transit, allows sending confidential messages to another party, by negotiating a shared key.

Digital Signatures

Used very widely, allows for proof of that the private key owner has endorsed a specific input.

Symmetric Cryptography

Used very widely, especially for tion at rest and for actually rring data for encryption in t, allows to encrypt data with a key.

Fancy Cryptography

Various other uses of cryptography, often to accomplish complicated privacy guarantees.

The Post-Quantum Threat Model

Asymmetric Encryption

Used mainly for encryption in transit, allows sending confidential messages to another party, by negotiating a shared key.

Digital Signatures

Used very widely, allows for proof of that the private key owner has endorsed a specific input.

Symmetric Cryptography

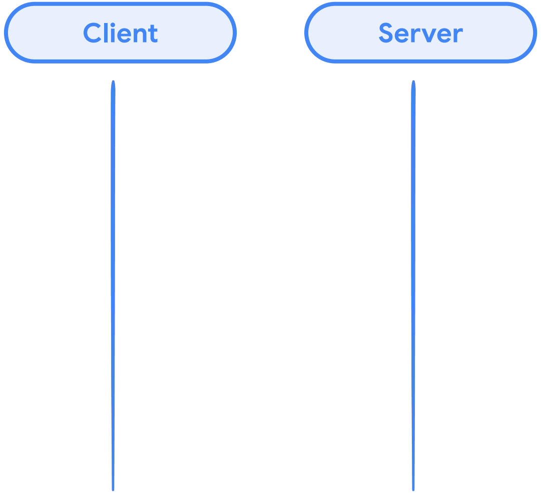
Used very widely, especially for encryption at rest and for actually transferring data for encryption in transit, allows to encrypt data with a key.

Fancy Cryptography

Various other uses of cryptography, often to accomplish complicated privacy guarantees.

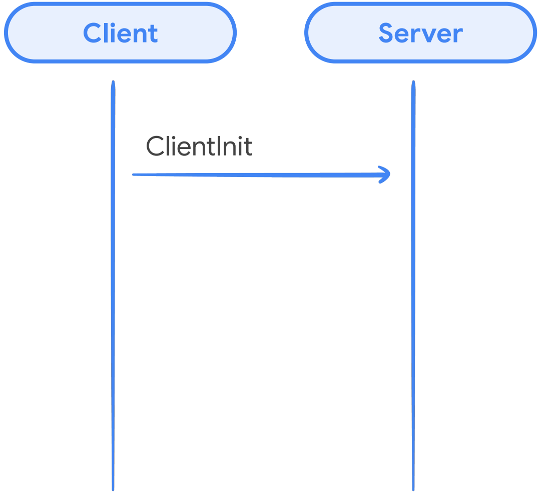
PQ ALTS

ALTS: Overview



Documentation: https://cloud.google.com/docs/security/encryption-in-transit/application-layer-transport-security

ALTS: Overview

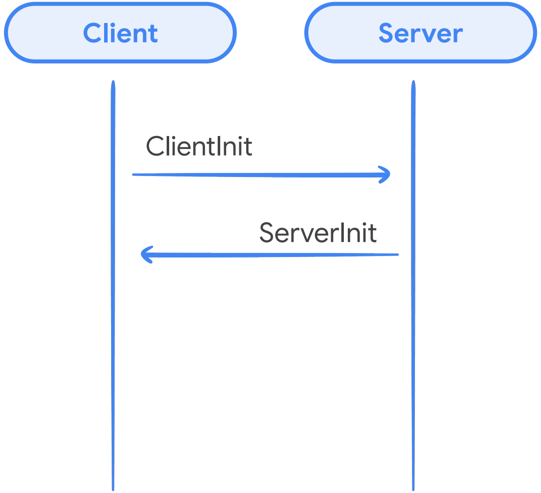


ClientInit

- static ECDH key
- cert for ECDH key

Documentation: https://cloud.google.com/docs/security/encryption-in-transit/application-layer-transport-security

ALTS: Overview



ClientInit

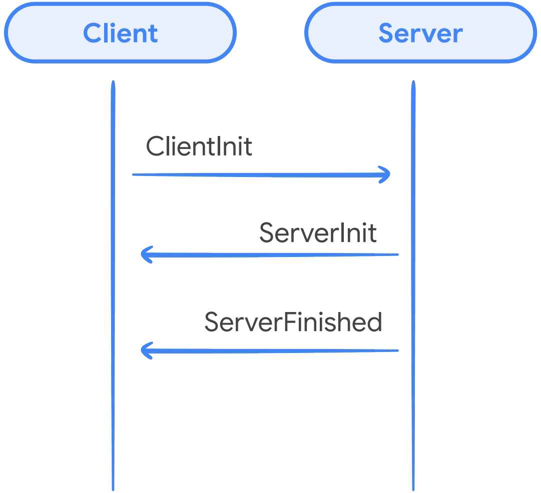
- static ECDH key
- cert for ECDH key

ServerInit

- static ECDH key
- cert for ECDH key

Documentation: https://cloud.google.com/docs/security/encryption-in-transit/application-layer-transport-security

ALTS: Overview



ClientInit

- static ECDH key
- cert for ECDH key

ServerInit

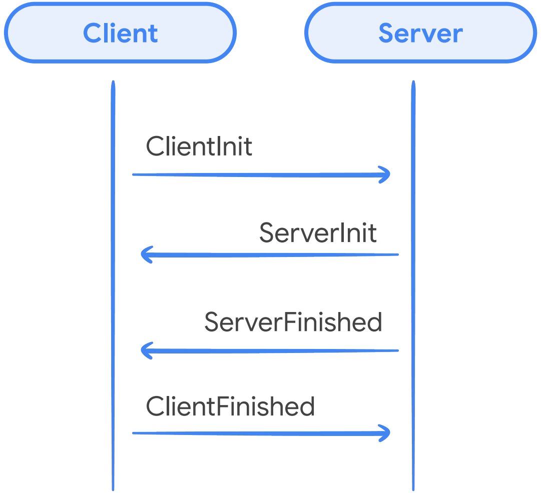
- static ECDH key
- cert for ECDH key

ServerFinished

- HMAC(shared_secret, server_const)

Documentation: https://cloud.google.com/docs/security/encryption-in-transit/application-layer-transport-security

ALTS: Overview



ClientInit

- static ECDH key
- cert for ECDH key

ServerInit

- static ECDH key
- cert for ECDH key

ServerFinished

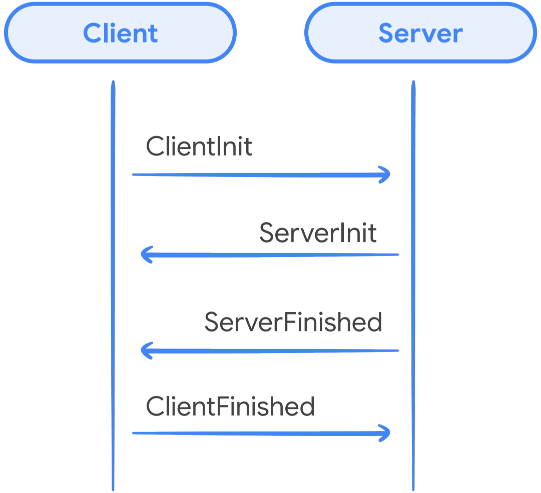
- HMAC(shared_secret, server_const)

ClientFinished

- HMAC(shared_secret, client_const)

Documentation: https://cloud.google.com/docs/security/encryption-in-transit/application-layer-transport-security

ALTS: Overview



ClientInit

- static ECDH key
- cert for ECDH key
- resumption ticket

ServerInit

- resumption confirmation

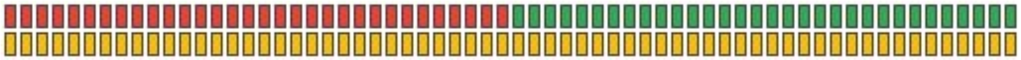
ServerFinished

- HMAC(shared_secret, server_const)

ClientFinished

- HMAC(shared_secret, client_const)

PQC Overview

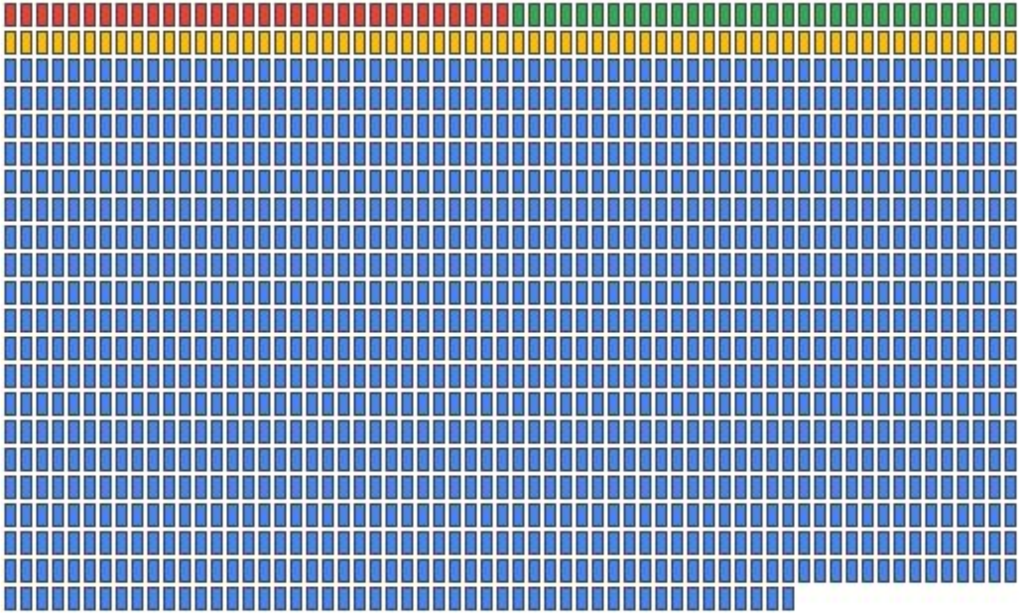


🟥 Protocol Overhead (estimate)

🟩 X25519 Keyshare

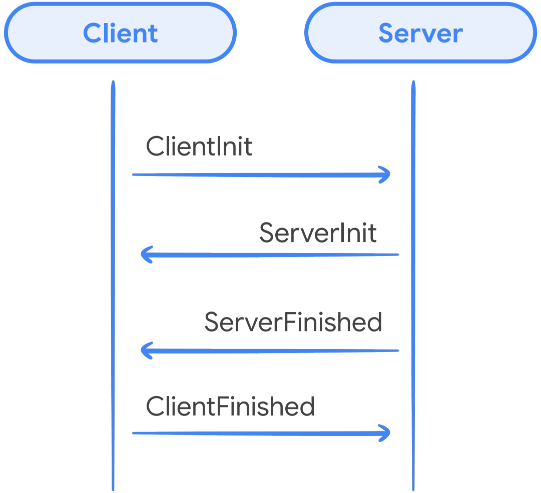
🟨 Certificate

PQC Overview



- 🟥 Protocol Overhead (estimate)
- 🟩 X25519 Keyshare
- 🟨 Certificate
- 🟦 HRSS public key/ciphertext

ALTS PQC



ClientInit

- static ECDH key
- cert for ECDH key
- ephemeral PQC public key

ServerInit

- static ECDH key
- cert for ECDH key
- PQC KEM ciphertext

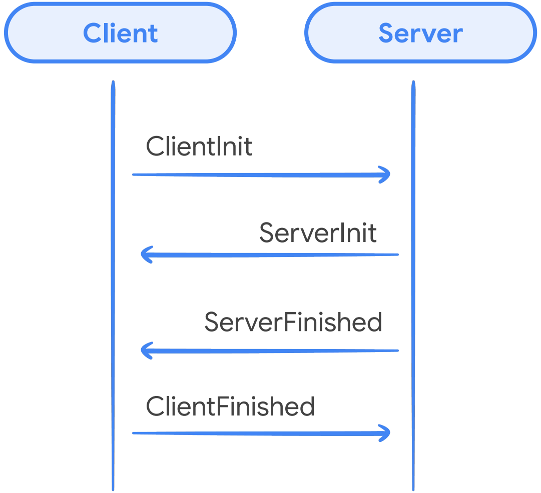
ServerFinished

- HMAC(shared_secret, server_const)

ClientFinished

- HMAC(shared_secret, client_const)

ALTS PQC



ClientInit

- static ECDH key
- cert for ECDH key
- somewhat ephemeral PQC public key

ServerInit

- static ECDH key
- cert for ECDH key
- PQC KEM ciphertext

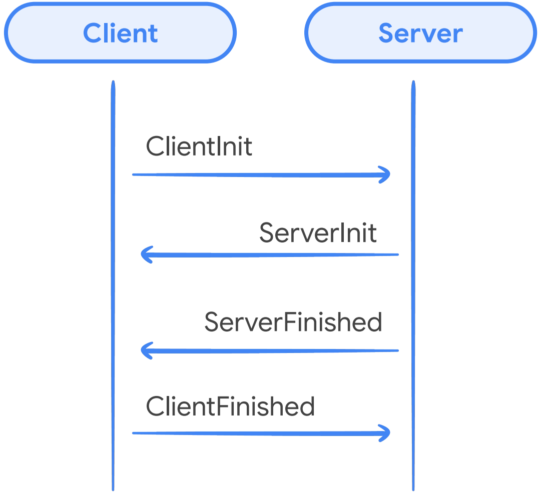
ServerFinished

- HMAC(shared_secret, server_const)

ClientFinished

- HMAC(shared_secret, client_const)

ALTS PQC



ClientInit

- static ECDH key
- cert for ECDH key
- resumption ticket
- somewhat ephemeral PQC public key

ServerInit

- resumption confirmation
- PQC KEM ciphertext

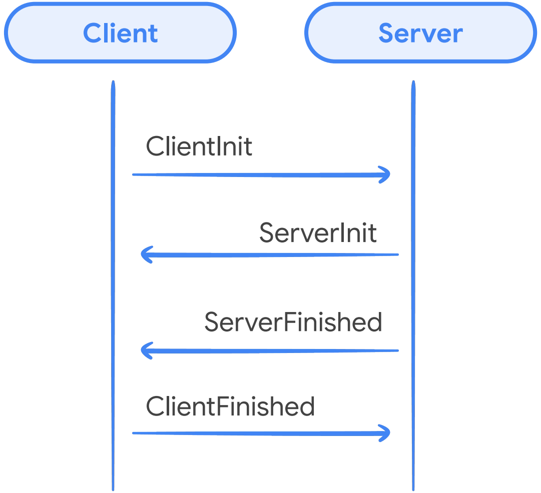
ServerFinished

- HMAC(shared_secret, server_const)

ClientFinished

- HMAC(shared_secret, client_const)

ALTS PQC



ClientInit

- static ECDH key
- cert for ECDH key
- resumption ticket
- somewhat ephemeral PQC public key

ServerInit

- resumption confirmation

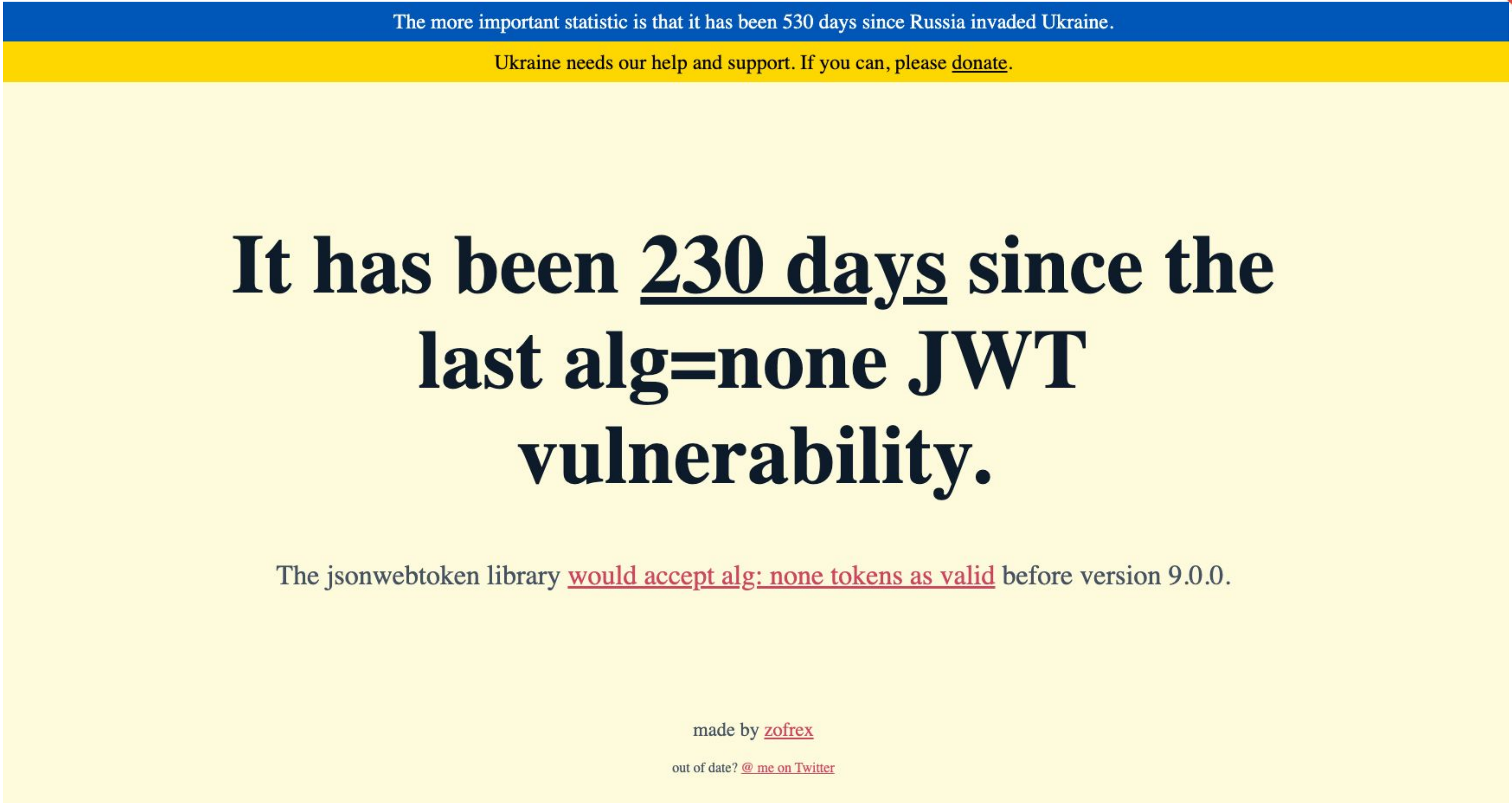
ServerFinished

- HMAC(shared_secret, server_const)

ClientFinished

- HMAC(shared_secret, client_const)

Primitives and Standards



The more important statistic is that it has been 530 days since Russia invaded Ukraine.

Ukraine needs our help and support. If you can, please donate.

It has been 230 days since the last alg=none JWT vulnerability.

The jsonwebtoken library would accept alg: none tokens as valid before version 9.0.0.

made by zofrex

out of date? @ me on Twitter

Guiding Principle

A Cryptographic Key is the full description of a mathematical function, with no information other than the inputs demanded by the primitive required to evaluate it.

Tink Keys

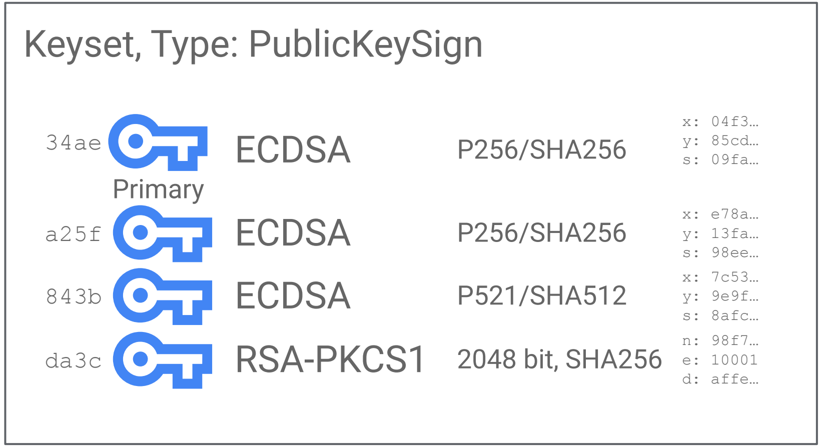
Tink Keys

ECDSA

P256/SHA256

```
x: 04f3…
y: 85cd…
s: 09fa…
```

Tink Keys



Keyset, Type: PublicKeySign

34ae	Primary	ECDSA	P256/SHA256	x: 04f3… y: 85cd… s: 09fa…
a25f		ECDSA	P256/SHA256	x: e78a… y: 13fa… s: 98ee…
843b		ECDSA	P521/SHA512	x: 7c53… y: 9e9f… s: 8afc…
da3c		RSA-PKCS1	2048 bit, SHA256	n: 98f7… e: 10001 d: affe…

Tink Keys

Keyset, Type: PublicKeySign

34ae	🔑 Primary	ECDSA	P256/SHA256	x: 04f3… y: 85cd… s: 09fa…
a25f	🔑	ECDSA	P256/SHA256	x: e78a… y: 13fa… s: 98ee…
843b	🔑	ECDSA	P521/SHA512	x: 7c53… y: 9e9f… s: 8afc…
da3c	🔑	RSA-PKCS1	2048 bit, SHA256	n: 98f7… e: 10001 d: affe…

Sample Signature:

```
01a25f9da0eb…
```

Tink Keys

Keyset, Type: PublicKeySign

34ae Primary	ECDSA	P256/SHA256	x: 04f3… y: 85cd… s: 09fa…
a25f	ECDSA	P256/SHA256	x: e78a… y: 13fa… s: 98ee…
843b	ECDSA	P521/SHA512	x: 7c53… y: 9e9f… s: 8afc…
da3c	RSA-PKCS1	2048 bit, SHA256	n: 98f7… e: 10001 d: affe…

Sample Signature:

```
01a25f9da0eb…
```

Tink Keys

Keyset, Type: PublicKeySign

34ae Primary	ECDSA	P256/SHA256	x: 04f3… y: 85cd… s: 09fa…
a25f	ECDSA	P256/SHA256	x: e78a… y: 13fa… s: 98ee…
843b	ECDSA	P521/SHA512	x: 7c53… y: 9e9f… s: 8afc…
da3c	RSA-PKCS1	2048 bit, SHA256	n: 98f7… e: 10001 d: affe…

Sample Signature:

`01a25f9da0eb…`

Tink Keys

Keyset, Type: PublicKeySign

34ae	Primary	ECDSA	P256/SHA256	x: 04f3… y: 85cd… s: 09fa…
a25f		ECDSA	P256/SHA256	x: e78a… y: 13fa… s: 98ee…
843b		ECDSA	P521/SHA512	x: 7c53… y: 9e9f… s: 8afc…
da3c		RSA-PKCS1	2048 bit, SHA256	n: 98f7… e: 10001 d: affe…

Sample Signature:

01a25f9da0eb…

Tink Keys

Keyset, Type: PublicKeySign

34ae	🔑 Primary	ECDSA	P256/SHA256	x: 04f3… y: 85cd… s: 09fa…
a25f	🔑	ECDSA	P256/SHA256	x: e78a… y: 13fa… s: 98ee…
843b	🔑	ECDSA	P521/SHA512	x: 7c53… y: 9e9f… s: 8afc…
da3c	🔑	RSA-PKCS1	2048 bit, SHA256	n: 98f7… e: 10001 d: affe…

Tink Keys

Keyset, Type: PublicKeySign

34ae	Primary	ECDSA	P256/SHA256	x: 04f3… y: 85cd… s: 09fa…
a25f		ECDSA	P256/SHA256	x: e78a… y: 13fa… s: 98ee…
843b		ECDSA	P521/SHA512	x: 7c53… y: 9e9f… s: 8afc…

Tink Keys

Keyset, Type: PublicKeySign

34ae	Primary	ECDSA	P256/SHA256	x: 04f3… y: 85cd… s: 09fa…
a25f		ECDSA	P256/SHA256	x: e78a… y: 13fa… s: 98ee…
843b		ECDSA	P521/SHA512	x: 7c53… y: 9e9f… s: 8afc…

Tink Keys

Keyset, Type: PublicKeySign

fe71	🔑 Primary	ECDSA + Dilithium	P256/SHA256 Dilithum3	x: 04f3… y: 85cd… s: 09fa… ρ: 0a2b… s_1: 1e4f… …
34ae	🔑	ECDSA	P256/SHA256	x: 04f3… y: 85cd… s: 09fa…
a25f	🔑	ECDSA	P256/SHA256	x: e78a… y: 13fa… s: 98ee…
843b	🔑	ECDSA	P521/SHA512	x: 7c53… y: 9e9f… s: 8afc…

Example: Dilithium3

Dilithium3 consists of three functions:

$$
\begin{aligned}
G &: 0 \xrightarrow{R} \mathcal{P} \times \mathcal{K} \\
S &: \mathcal{K} \times \mathcal{M} \xrightarrow{R} \mathcal{S} \\
V &: \mathcal{P} \times \mathcal{M} \times \mathcal{S} \to \{\top, \bot\}
\end{aligned}
$$

Example: Dilithium3

Dilithium3 consists of three functions:

$$G: 0 \xrightarrow{R} \left(\{0,1\}^{15616} \times \{0,1\}^{88448}\right) \dot{\cup} \{\bot\}$$

$$S: \{0,1\}^{884488} \times \{0,1\}^{*} \xrightarrow{R} \{0,1\}^{26344} \dot{\cup} \{\bot\}$$

$$V: \{0,1\}^{15616} \times \{0,1\}^{*} \times \{0,1\}^{26344} \to \{\top, \bot\}$$

Test vectors that test everything

```
{
        "tcId" : 506,
        "comment" : "special case for x_2 in multiplication by 9",
        "public" : "302a300506032b656e032100dcffc4c1e1fba5fda9d5c98421d99c257afa90921bc212a046d90f6683e8a467",
        "private" :
"302e020100300506032b656e04220420707ee81f113a244c9d87608b12158c50f9ac1f2c8948d170ad16ab0ad866d74b",
        "shared" : "7ecdd54c5e15f7b4061be2c30b5a4884a0256581f87df60d579a3345653eb641",
        "result" : "acceptable",
        "flags" : [
          "Twist"
        ]
      },
```

Hybrid Signatures and Separability

$$G = (G_{1,P}, G_{2,P}, G_{1,K}, G_{2,K})$$

$$S = (S_1, S_2)$$

$$V = V_1 \wedge V_2$$

Less options, please

For us, the PQC standards are

- Kyber768
- Dilithium3
- Sphincs+-SHA256s

(list not final; the standards aren't even out yet)

And maybe, 12 rounds of Keccak is enough

Key Takeaways

Rolling out new crypto at scale takes time

We needed several refinements over multiple years to be able to roll out PQC even in a highly controlled environment.

Standards should be well-defined

Standards need to be defined to prescribe the handling of all inputs, including edge cases.

Gaps in fancy cryptography

While we have a decent selection for asymmetric encryption and digital signatures, we have nowhere near the same flexibility with these new schemes to construct more advanced cryptography (RLWE notwithstanding)

Thank you

Sophie Schmieg

Senior Staff Cryptography Engineer

sschmieg@google.com

"Bonus" Slides

If you see this slide, I have run out of material. All that follows will be an explanation of p-adic lattices, to distract you from that.

I guess I can always reuse the slides for the rump session

$$\begin{tikzcd}
0 \arrow[r] & \bar{\mathbb{G}}_m^t \arrow[r] \arrow[d, hook] & \bar{E} \arrow[r] \arrow[d, hook] & B \arrow[r] \arrow[d, equal] & 0 \\
0 \arrow[r] & \mathbb{G}_m^t \arrow[r] & E \arrow[r] & B \arrow[r] & 0
\end{tikzcd}$$

$$\begin{tikzcd}
0 \arrow[r] & \bar{\mathbb{G}}_m^t \arrow[r] \arrow[d, hook] & \bar{E} \arrow[r] \arrow[d, hook] & B \arrow[r] \arrow[d, equal] & 0 \\
0 \arrow[r] & \mathbb{G}_m^t \arrow[r] & E \arrow[r] & B \arrow[r] & 0
\end{tikzcd}$$

M is a lattice if |.| is injective and -log |M| is a lattice in R^t

$$\begin{tikzcd}
0 \arrow[r] & \bar{\mathbb{G}}_m^t \arrow[r] \arrow[d, hook] & \bar{E} \arrow[r] \arrow[d, hook] & B \arrow[r] \arrow[d, equal] & 0 \\
0 \arrow[r] & \mathbb{G}_m^t \arrow[r] & E \arrow[r] & B \arrow[r] & 0
\end{tikzcd}$$

M is a lattice if |.| is injective and -log |M| is a lattice in R^t

I still don't know how or why someone would construct a cryptosystem out of this. It is useful to describe rigid analytic Jacobians, though.