# Making and breaking implicitly authenticated post-quantum key exchange


**Douglas Stebila**

UNIVERSITY OF WATERLOO

NSERC CRSNG


Joint work with Peter Schwabe and Thom Wiggers
https://eprint.iacr.org/2020/534

Joint work with Nina Bindel and Shannon Veitch
https://eprint.iacr.org/2020/1288

UNIVERSITY OF WATERLOO


IQC Institute for Quantum Computing

CYBER SECURITY AND PRIVACY INSTITUTE
UNIVERSITY OF WATERLOO

# Cryptography @ University of Waterloo

- UW involved in 4 NIST PQC Round 3 submissions:
  - Finalists: CRYSTALS-Kyber, NTRU
  - Alternates: FrodoKEM, SIKE
- UW involved in 4 NIST Lightweight Crypto Round 2 submissions: ACE, SPIX, SpoC, WAGE
- Elliptic curves: David Jao, Alfred Menezes, (Scott Vanstone)
- Information theoretic cryptography: Doug Stinson
- Privacy-enhancing technologies: Ian Goldberg
- Quantum cryptanalysis: Michele Mosca
- Quantum cryptography: Norbert Lütkenhaus, Thomas Jennewein, Debbie Leung
- Gord Agnew, Vijay Ganesh, Guang Gong, Sergey Gorbunov, Anwar Hasan, Florian Kerschbaum

# Motivation

# Authenticated key exchange

- Two parties establish a shared secret over a public communication channel

# Vast literature on AKE protocols

- Many **security definitions** capturing various adversarial powers: BR, CK, eCK, …
- Different types of **authentication credentials**: public key, shared secret key, password, identity-based, …
- **Additional security goals**: weak/strong forward secrecy, key compromise impersonation resistance, post-compromise security, …
- Additional **protocol functionality**: multi-stage, ratcheting, …
- **Group** key exchange
- **Real-world protocols**: TLS, SSH, Signal, IKE, ISO, EMV, …
- …

# Explicit authentication

Alice receives assurance that she really is talking to Bob

# Implicit authentication

Alice is assured that only Bob would be able to compute the shared secret

# Explicitly authenticated key exchange: Signed Diffie–Hellman

| **Alice** | | **Bob** |
|---|---|---|
| $(pk_A, sk_A) \leftarrow \text{SIG.KeyGen}()$ | | $(pk_B, sk_B) \leftarrow \text{SIG.KeyGen}()$ |
| obtain $pk_B$ | | obtain $pk_A$ |
| $x \leftarrow_{\$} \{0, \ldots, q-1\}$ | | |
| $X \leftarrow g^x$ | $\xrightarrow{X}$ | |
| | | $y \leftarrow_{\$} \{0, \ldots, q-1\}$ |
| | | $Y \leftarrow g^y$ |
| | $\xleftarrow{Y, \sigma_B}$ | $\sigma_B \leftarrow \text{SIG.Sign}(sk_B, A \| B \| X \| Y)$ |
| $\sigma_A \leftarrow \text{SIG.Sign}(sk_A, A \| B \| X \| Y)$ | $\xrightarrow{\sigma_A}$ | |
| $k \leftarrow H(sid, Y^x)$ | | $k \leftarrow H(sid, X^y)$ |
| | application data $\Longleftrightarrow$ using authenticated encryption | |

# Implicitly authenticated key exchange: Double-DH

| **Alice** | | **Bob** |
|---|---|---|
| $sk_A \leftarrow_\$ \{0, \ldots, q-1\}$ | | $sk_B \leftarrow_\$ \{0, \ldots, q-1\}$ |
| $pk_A \leftarrow g^{sk_A}$ | | $pk_B \leftarrow g^{sk_B}$ |
| obtain $pk_B$ | | obtain $pk_A$ |
| $x \leftarrow_\$ \{0, \ldots, q-1\}$ | | $y \leftarrow_\$ \{0, \ldots, q-1\}$ |
| $X \leftarrow g^x$ | $\xrightarrow{\quad X \quad}$ | $Y \leftarrow g^y$ |
| | $\xleftarrow{\quad Y \quad}$ | |
| $k \leftarrow H(sid,\ pk_B^{sk_A} \| Y^x)$ | | $k \leftarrow H(sid,\ pk_A^{sk_B} \| X^y)$ |
| | application data $\Longleftrightarrow$ using authenticated encryption | |

Home

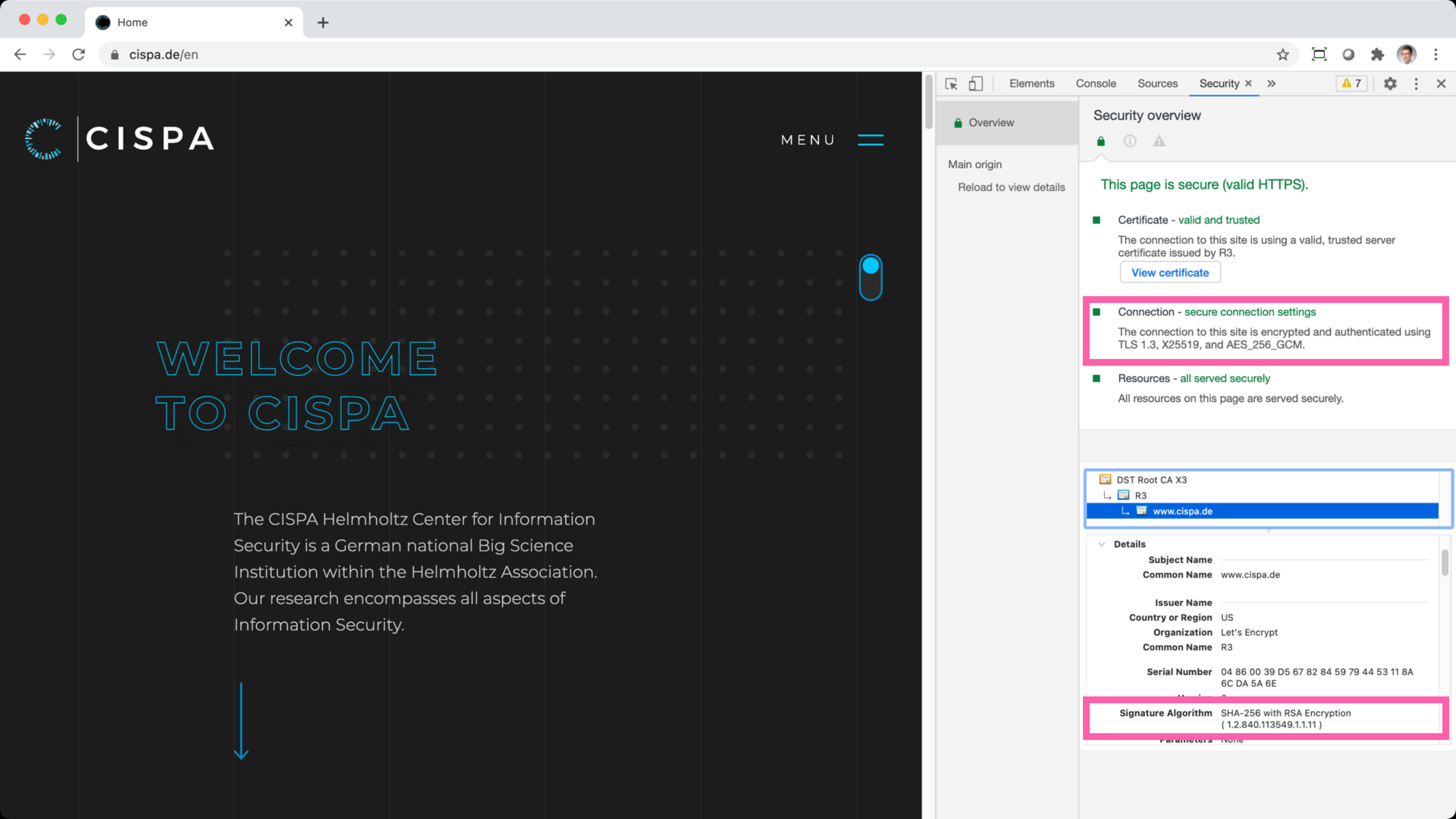
cispa.de/en

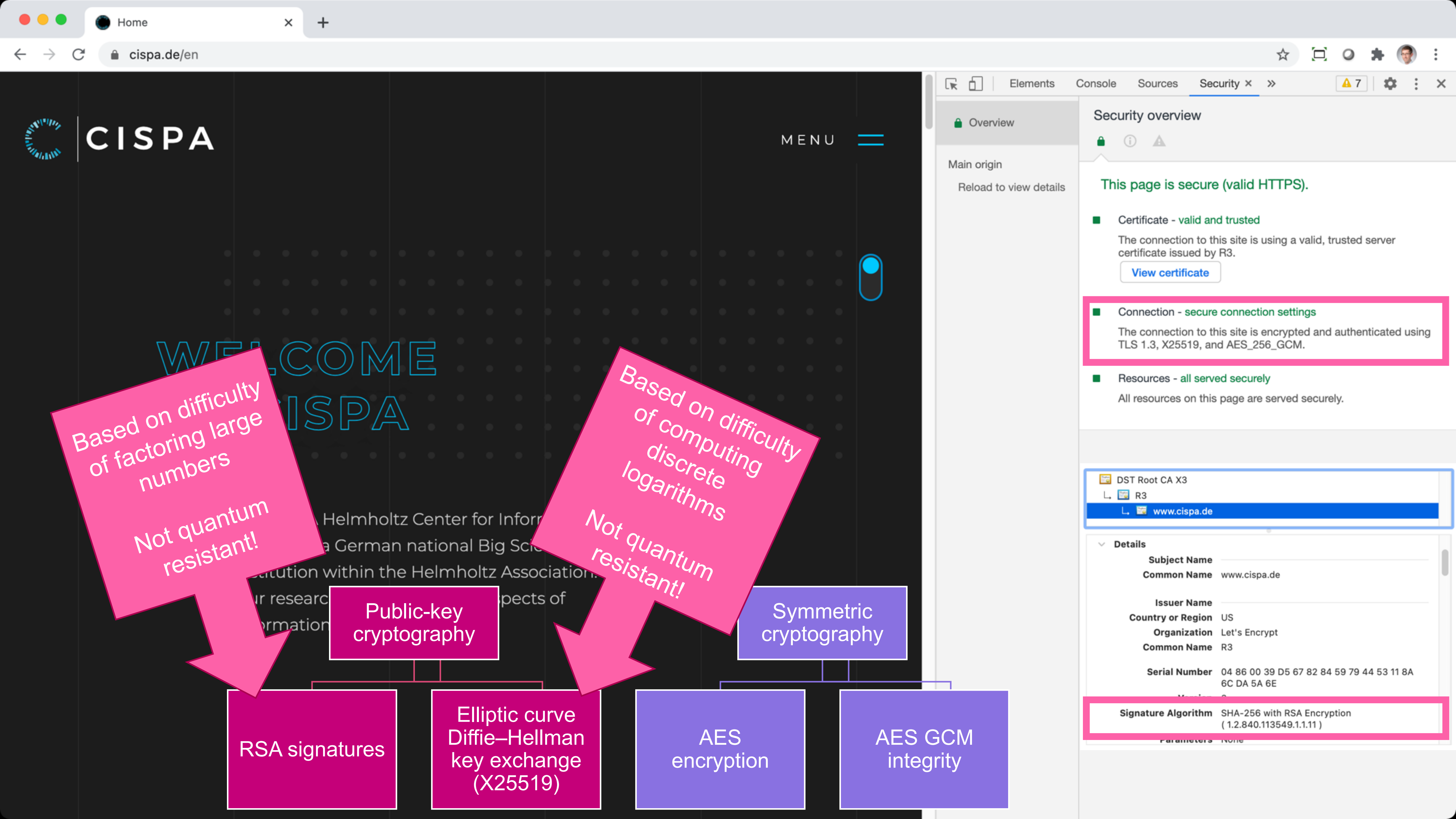
CISPA

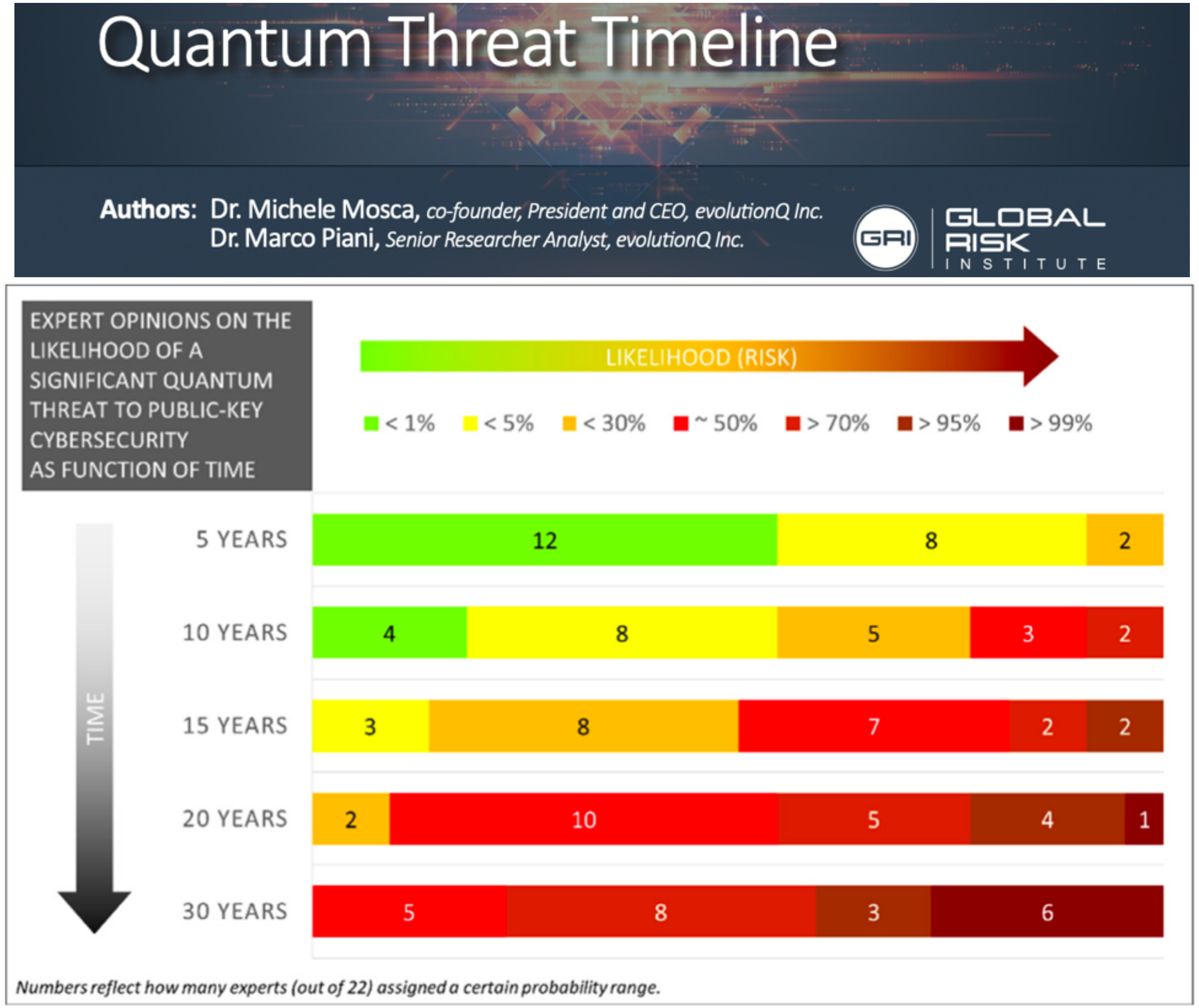
MENU

# WELCOME TO CISPA

The CISPA Helmholtz Center for Information Security is a German national Big Science Institution within the Helmholtz Association. Our research encompasses all aspects of Information Security.


Home
cispa.de/en
CISPA
MENU
WELCOME
TO CISPA
The CISPA Helmholtz Center for Information Security is a German national Big Science Institution within the Helmholtz Association. Our research encompasses all aspects of Information Security.
Elements
Console
Sources
Security
Overview
Main origin
Reload to view details
Security overview
This page is secure (valid HTTPS).
Certificate - valid and trusted
The connection to this site is using a valid, trusted server certificate issued by R3.
View certificate
Connection - secure connection settings
The connection to this site is encrypted and authenticated using TLS 1.3, X25519, and AES_256_GCM.
Resources - all served securely
All resources on this page are served securely.
DST Root CA X3
R3
www.cispa.de
Details
Subject Name
Common Name www.cispa.de
Issuer Name
Country or Region US
Organization Let's Encrypt
Common Name R3
Serial Number 04 86 00 39 D5 67 82 84 59 79 44 53 11 8A 6C DA 5A 6E
Signature Algorithm SHA-256 with RSA Encryption ( 1.2.840.113549.1.1.11 )


Home
cispa.de/en
CISPA
MENU
WELCOME
CISPA
Helmholtz Center for Infor
a German national Big Sci
titution within the Helmholtz Association
r researc
pects of
rmation
Elements
Console
Sources
Security
Overview
Main origin
Reload to view details
Security overview
This page is secure (valid HTTPS).
Certificate - valid and trusted
The connection to this site is using a valid, trusted server certificate issued by R3.
View certificate
Connection - secure connection settings
The connection to this site is encrypted and authenticated using TLS 1.3, X25519, and AES_256_GCM.
Resources - all served securely
All resources on this page are served securely.
DST Root CA X3
R3
www.cispa.de
Details
Subject Name
Common Name www.cispa.de
Issuer Name
Country or Region US
Organization Let's Encrypt
Common Name R3
Serial Number 04 86 00 39 D5 67 82 84 59 79 44 53 11 8A 6C DA 5A 6E
Signature Algorithm SHA-256 with RSA Encryption ( 1.2.840.113549.1.1.11 )
Based on difficulty of factoring large numbers
Not quantum resistant!
Based on difficulty of computing discrete logarithms
Not quantum resistant!
Public-key cryptography
Symmetric cryptography
RSA signatures
Elliptic curve Diffie–Hellman key exchange (X25519)
AES encryption
AES GCM integrity





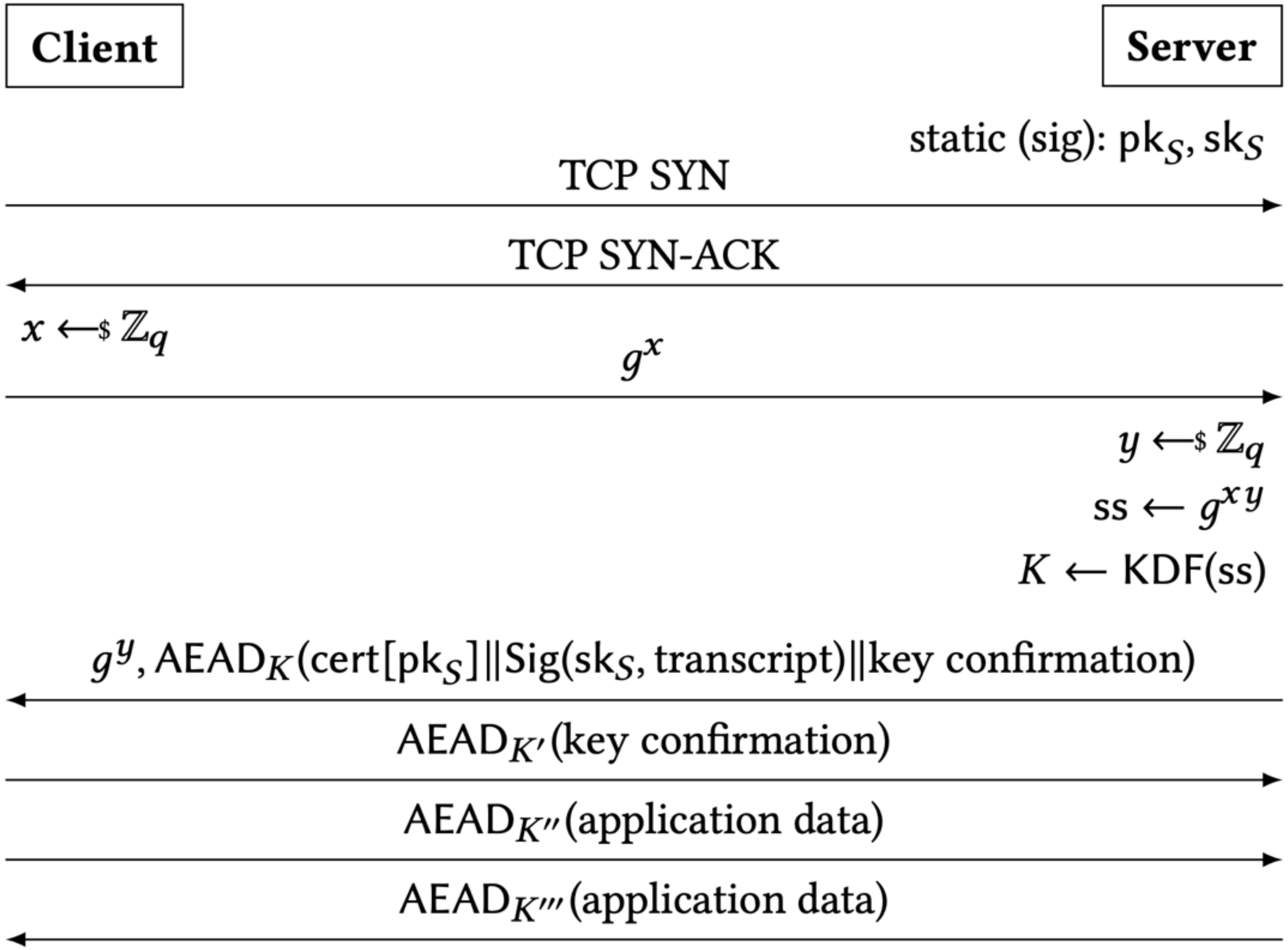
https://globalriskinstitute.org/publications/quantum-threat-timeline/

# Part 1:
# Making implicitly authenticated post-quantum key exchange

Peter Schwabe, Douglas Stebila, Thom Wiggers. Post-quantum TLS without handshake signatures. In Proc. 27th ACM Conference on Computer and Communications Security (CCS) 2020. ACM, November 2020.
https://eprint.iacr.org/2020/534

# TLS 1.3 handshake

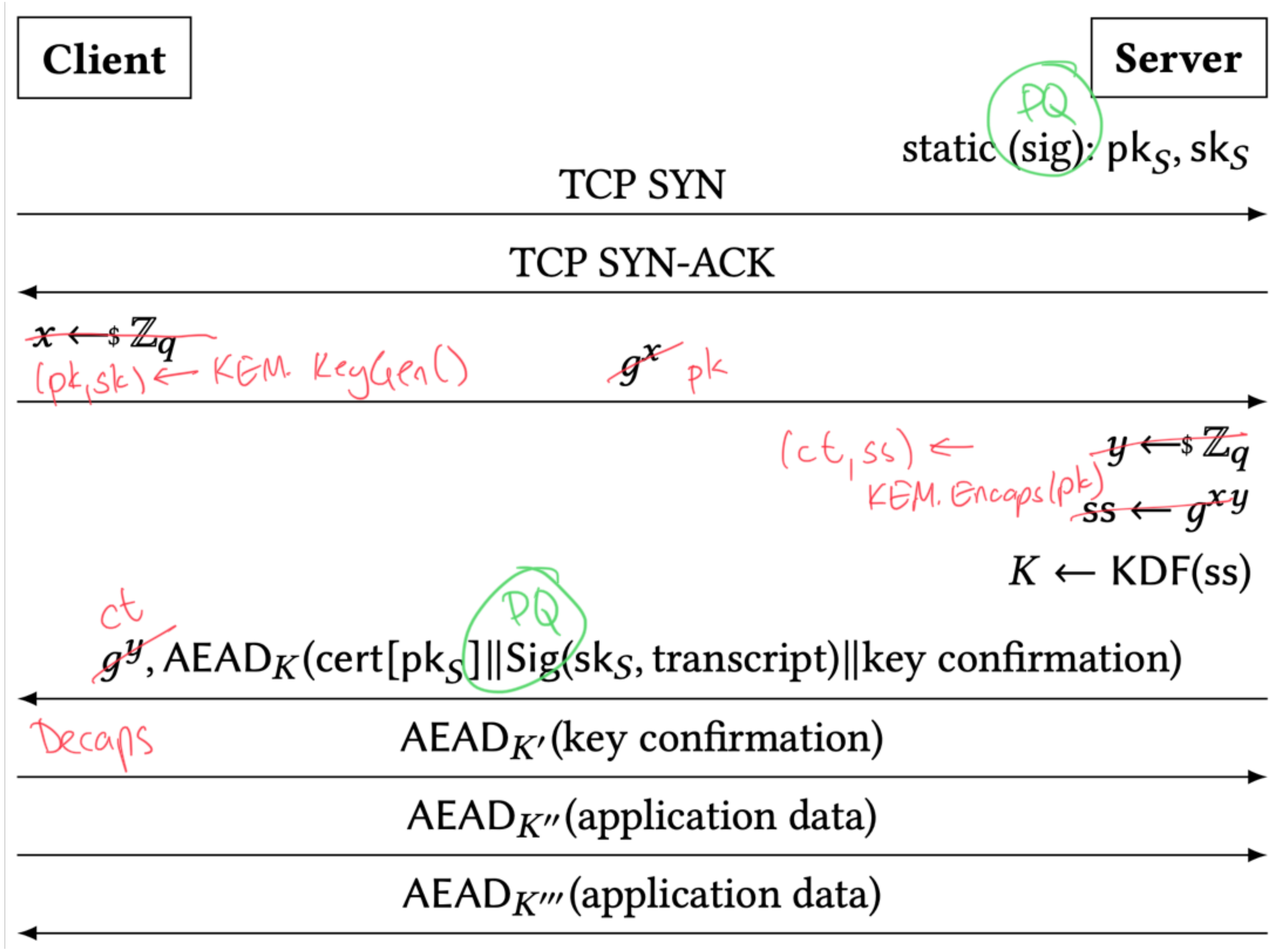
Signed Diffie–Hellman



# TLS 1.3 handshake

~~Signed Diffie–Hellman~~

Post-Quantum!!!



# Problem

post-quantum
signatures
are big

| Signature scheme | | Public key (bytes) | Signature (bytes) |
|---|---|---|---|
| RSA-2048 | Factoring | 272 | 256 |
| Elliptic curves | Elliptic curve discrete logarithm | 32 | 32 |
| **Dilithium** | **Lattice-based (MLWE/MSIS)** | **1,184** | **2,044** |
| **Falcon** | **Lattice-based (NTRU)** | **897** | **690** |
| **XMSS** | **Hash-based** | **32** | **979** |
| **GeMSS** | **Multi-variate** | **352,180** | **32** |

# Solution

use
post-quantum KEMs
for authentication

# Key encapsulation mechanisms (KEMs)

An abstraction of Diffie–Hellman key exchange

$(pk, sk) \leftarrow \mathsf{KEM.KeyGen}()$

$$\xrightarrow{\quad pk \quad}$$

$(ct, k) \leftarrow \mathsf{KEM.Encaps}(pk)$

$$\xleftarrow{\quad ct \quad}$$

$k \leftarrow \mathsf{KEM.Decaps}(sk, ct)$

| Signature scheme | | Public key (bytes) | Signature (bytes) |
|---|---|---|---|
| RSA-2048 | Factoring | 272 | 256 |
| Elliptic curves | Elliptic curve discrete logarithm | 32 | 32 |
| **Dilithium** | **Lattice-based (MLWE/MSIS)** | **1,184** | **2,044** |
| **Falcon** | **Lattice-based (NTRU)** | **897** | **690** |
| **XMSS** | **Hash-based** | **32** | **979** |
| **GeMSS** | **Multi-variate** | **352,180** | **32** |

| KEM | | Public key (bytes) | Ciphertext (bytes) |
|---|---|---|---|
| RSA-2048 | Factoring | 272 | 256 |
| Elliptic curves | Elliptic curve discrete logarithm | 32 | 32 |
| **Kyber** | **Lattice-based (MLWE)** | **800** | **768** |
| **NTRU** | **Lattice-based (NTRU)** | **699** | **699** |
| **Saber** | **Lattice-based (MLWR)** | **672** | **736** |
| **SIKE** | **Isogeny-based** | **330** | **330** |
| **SIKE compressed** | **Isogeny-based** | **197** | **197** |

# Implicitly authenticated KEX is not new

## In theory

- DH-based: SKEME, MQV, HMQV, …
- KEM-based: BCGP09, FSXY12, …

## In practice

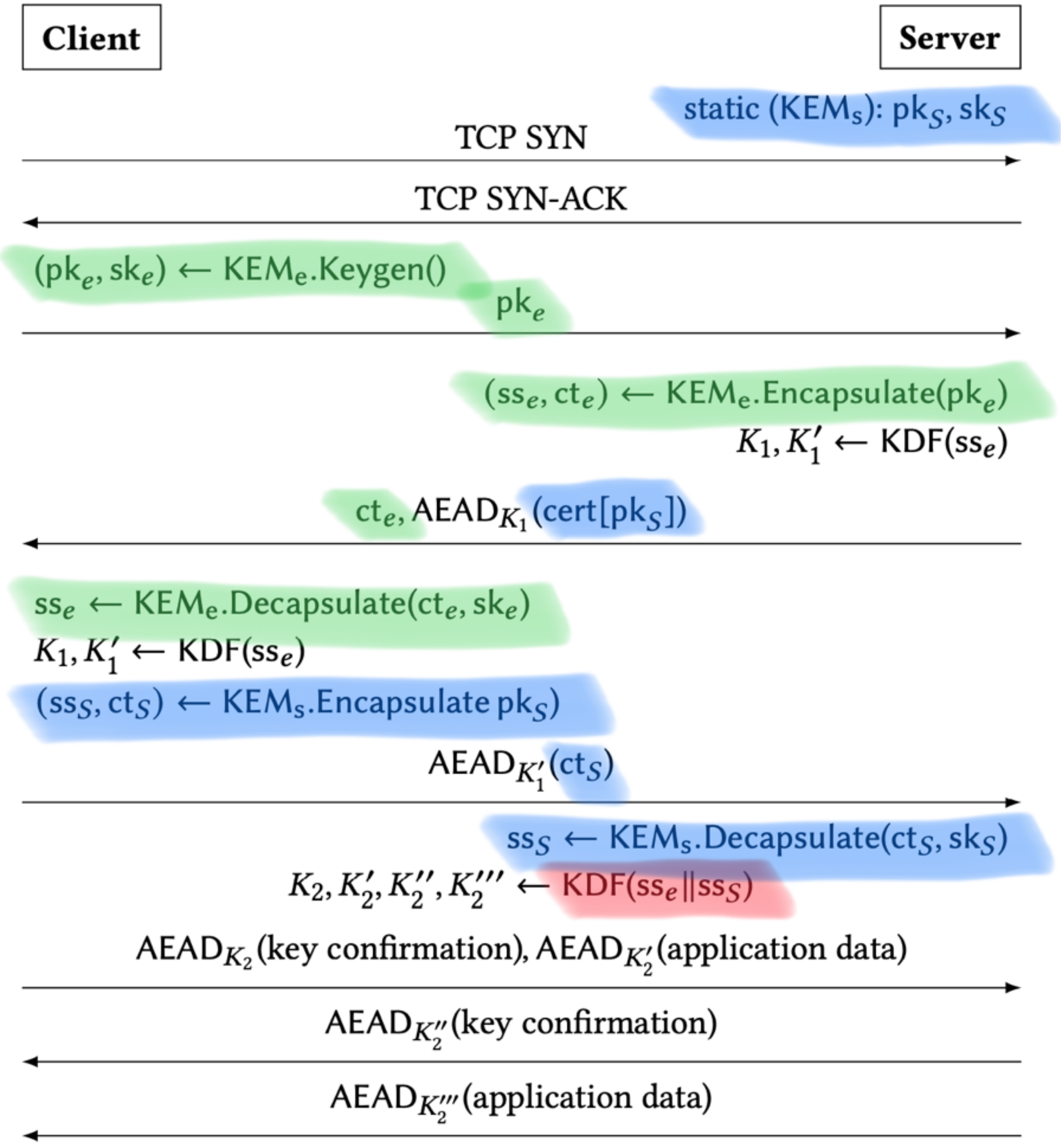
- RSA key transport in TLS ≤ 1.2
  - Lacks forward secrecy
- Signal, Noise, Wireguard
  - DH-based
  - Different protocol flows
- OPTLS
  - DH-based
  - Requires a non-interactive key exchange (NIKE)

# “KEMTLS” handshake

KEM for
ephemeral key exchange

KEM for
server-to-client
authenticated key exchange

Combine shared secrets



# Algorithm choices

**KEM for ephemeral key exchange**

- IND-CCA (or IND-1CCA)
- Want small public key + small ciphertext

**KEM for authenticated key exchange**

- IND-CCA
- Want small public key + small ciphertext

**Signature scheme for intermediate CA**

- Want small public key + small signature

**Signature scheme for root CA**

- Want small signature

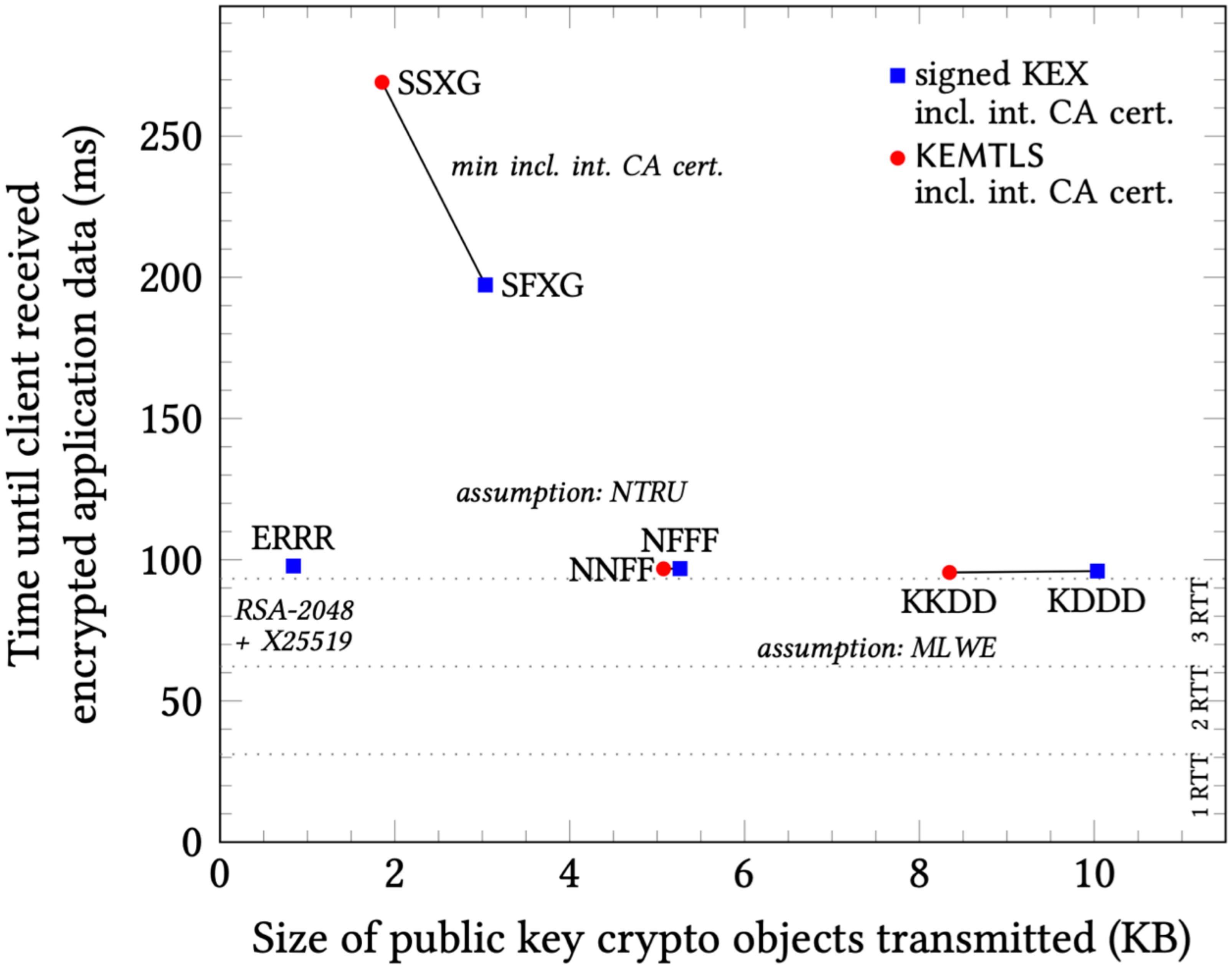
# 4 scenarios

1. Minimize size when intermediate certificate transmitted
2. Minimize size when intermediate certificate not transmitted (cached)
3. Use solely NTRU assumptions
4. Use solely module LWE/SIS assumptions

# Signed KEX versus KEMTLS

Labels ABCD:
A = ephemeral KEM
B = leaf certificate
C = intermediate CA
D = root CA

Algorithms: (all level 1)
Dilithium,
ECDH X25519,
Falcon,
GeMSS,
Kyber,
NTRU,
RSA-2048,
SIKE,
XMSS'

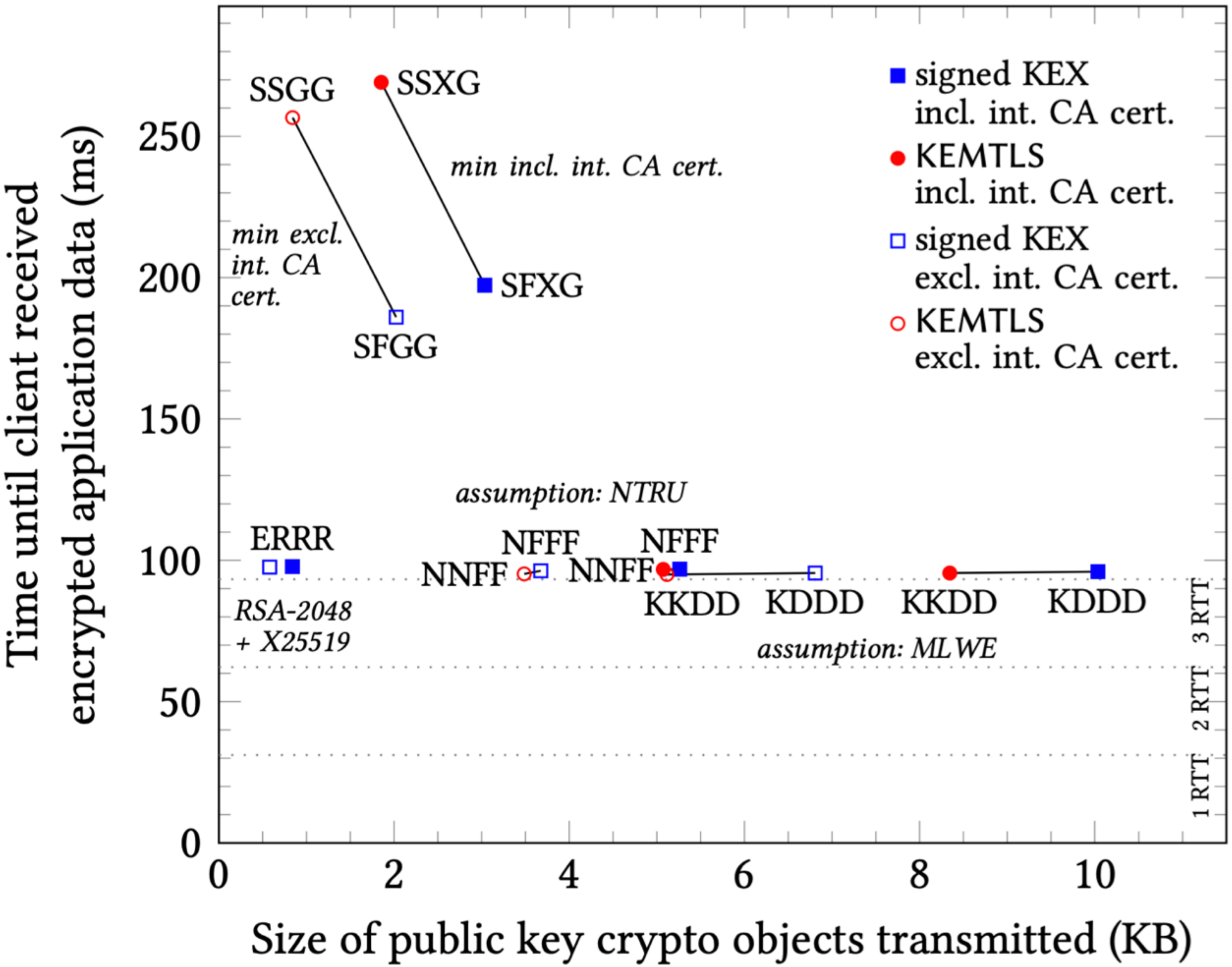



Rustls client/server with AVX2 implementations. Emulated network: latency 31.1 ms, bandwidth 1000 Mbps, 0% packet loss. Average of 100000 iterations.

# Signed KEX versus KEMTLS

Labels ABCD:
A = ephemeral KEM
B = leaf certificate
C = intermediate CA
D = root CA

Algorithms: (all level 1)
Dilithium,
ECDH X25519,
Falcon,
GeMSS,
Kyber,
NTRU,
RSA-2048,
SIKE,
XMSS'



# Observations

- Size-optimized KEMTLS requires < ½ communication of size-optimized PQ signed-KEM
- Speed-optimized KEMTLS uses 90% fewer server CPU cycles and still reduces communication
  - NTRU KEX (27 μs) 10x faster than Falcon signing (254 μs)
- No extra round trips required until client starts sending application data
- Smaller trusted code base (no signature generation on client/server)

# Security

Security model: multi-stage key exchange, extending [DFGS21]

- Key indistinguishability
- Forward secrecy
- Implicit and explicit authentication

Ingredients in security proof:

- **IND-CCA for long-term KEM**
- **IND-1CCA for ephemeral KEM**
- Collision-resistant hash function
- Dual-PRF security of HKDF
- EUF-CMA of HMAC

[DFGS21] Dowling, Fischlin, Günther, Stebila. Journal of Cryptology, 2021. https://eprint.iacr.org/2020/1044

# Security subtleties: authentication

## Implicit authentication

- Client's first application flow can't be read by anyone other than intended server, but client doesn't know server is live at the time of sending
- Also provides a form of deniable authentication since no signatures are used
  - Formally: offline deniability [DGK06]

## Explicit authentication

- Explicit authentication once key confirmation message transmitted
- *Retroactive* explicit authentication of earlier keys

[DGK06] Di Raimondo, Gennaro, Krawczyk. ACM CCS 2006. https://eprint.iacr.org/2006/280

# Security subtleties: downgrade resilience

- Choice of cryptographic algorithms not authenticated at the time the client sends its first application flow
  - MITM can't trick client into using undesirable algorithm
  - But MITM can trick them into temporarily using suboptimal algorithm
- Formally model 3 levels of downgrade-resilience:
  1. Full downgrade resilience
  2. No downgrade resilience to unsupported algorithms
  3. No downgrade resilience

# Security subtleties: forward secrecy

- **Weak forward secrecy 1:** adversary passive in the test stage
- **Weak forward secrecy 2:** adversary passive in the test stage or never corrupted peer's long-term key
- **Forward secrecy:** adversary passive in the test stage or didn't corrupt peer's long-term key before acceptance

- Can make detailed forward secrecy statements, such as:
  - Stage 1 and 2 keys are wfs1 when accepted, retroactive fs once stage 6 accepts

# Certificate lifecycle management for KEM public keys

Starting to be discussed on IETF LAMPS mailing list (Jan. 28, 2021) [1]

**Proof of possession**: How does requester prove possession of corresponding secret keys?

- Not really addressed in practice, since RSA and DL/ECDL keys can be used for both signing and encryption/KEX
- Can’t sign like in a Certificate Signing Request (CSR)
- Could do interactive challenge-response protocol (or just run KEMTLS), but need online verification (RFC 4210 Sect. 5.2.8.3)
- Send cert to requestor encrypted under key in the certificate (RFC 4210 Sect. 5.2.8.2) – but maybe broken by Certificate Transparency?
- Zero-knowledge proof of knowledge?

Thanks to Mike Ounsworth (Entrust Datacard) for raising some of these issues.

[1] https://mailarchive.ietf.org/arch/msg/spasm/FCCZv3Xi3rkbZyZWQnnMQM0EFYY/

# Certificate lifecycle management for KEM public keys

**Revocation**: How can certificate owner authorize a revocation request?

- Put a (hash of a) signature public key in the cert which can be used to revoke the cert?
    - Possibly could simplify to just revealing a hash preimage

# Conclusions on KEMTLS

- Summary of protocol design: implicit authentication via KEMs

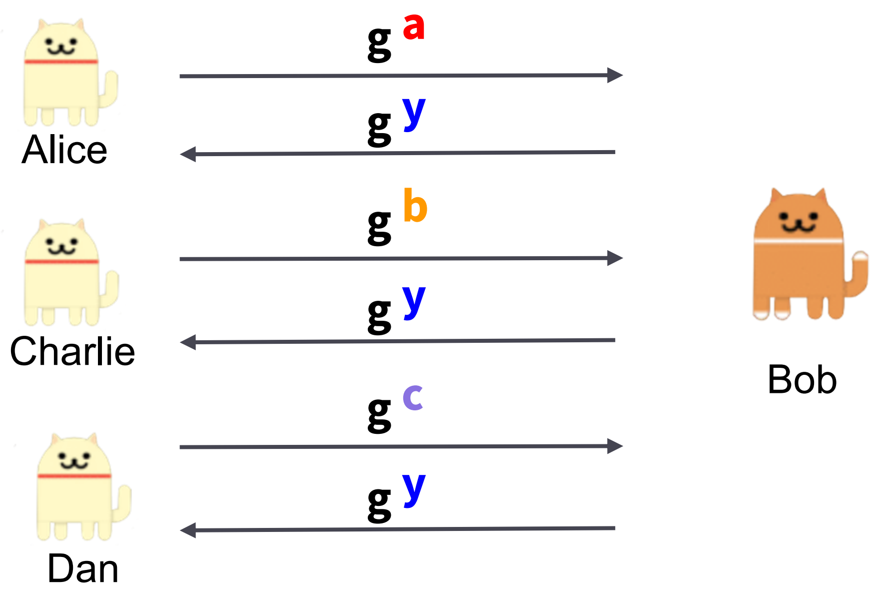
- Saves bytes on the wire and server CPU cycles
- Preserves client request after 1-RTT
- Caching intermediate CA certs brings even greater benefits

- Protocol design is simple to implement, provably secure
- Also have a variant supporting client authentication
- Working with Cloudflare to test within their infrastructure

# Part 2:
# Breaking implicitly authenticated post-quantum key exchange

Nina Bindel, Douglas Stebila, Shannon Veitch. Improved attacks against key reuse in learning with errors key exchange. IACR Cryptology ePrint Archive, October 2020. https://eprint.iacr.org/2020/1288

# Key reuse




Why reuse keys?

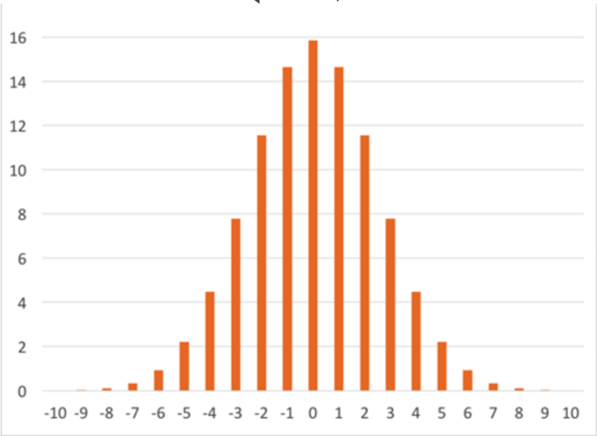
- certification
- storage requirements
- computational workload
- development efforts

# Learning with errors

Given $(A, b)$ with $A \leftarrow_\$ \mathbb{Z}_q^{m \times n}$, $s \leftarrow_\$ \chi_\alpha$, $e \leftarrow_\$ \chi_\alpha$, $b = As + e \mod q$, find $s$.

# Learning with errors

Given $(\color{blue}{A}, \color{blue}{b})$ with $\color{blue}{A} \leftarrow_\$ \mathbb{Z}_q^{m\times n}$, $\color{orange}{s} \leftarrow_\$ \chi_\alpha$, $\color{orange}{e} \leftarrow_\$ \chi_\alpha$, $\color{blue}{b} = \color{blue}{A}\color{orange}{s} + \color{orange}{e} \mod q$, find $\color{orange}{s}$.




Discrete Gaussian distribution

Regev, STOC 2005

# Ring learning with errors

$R_q$

Given $(A, b)$ with $A \leftarrow_\$ \mathbb{Z}_q^{m \times n}$, $s \leftarrow_\$ \chi_\alpha$, $e \leftarrow_\$ \chi_\alpha$, $b = As + e \mod q$, find $s$.

$$R_q = \mathbb{Z}_q[x]/\Phi(x)$$

Polynomial ring over a finite field.

- Commutative

Lyubashevsky, Piekert, Regev, EUROCRYPT 2010

# Basic RLWE-based key exchange

# RLWE-based key exchange

Public: $a \leftarrow_{\$} R_q$

Alice

$s_A \leftarrow_{\$} \chi_\alpha, e_A \leftarrow_{\$} \chi_\alpha$
$p_A = a s_A + 2 e_A$

$s_B \leftarrow_{\$} \chi_\alpha, e_B \leftarrow_{\$} \chi_\alpha$
$p_B = a s_B + 2 e_B$

Bob

Ding, Xie, Lin (eprint 2012/688)

# RLWE-based key exchange

$E := \{-\lfloor \frac{q}{4} \rfloor, \dots, \lfloor \frac{q}{4} \rceil\}$

$$\mathsf{Sig}(v) = \begin{cases} 0 & \text{if } v \in E \\ 1 & \text{otherwise} \end{cases}$$

Public: $a \leftarrow_{\$} R_q$

Alice

$s_A \leftarrow_{\$} \chi_\alpha, e_A \leftarrow_{\$} \chi_\alpha$
$p_A = a s_A + 2 e_A$

Bob

$s_B \leftarrow_{\$} \chi_\alpha, e_B \leftarrow_{\$} \chi_\alpha$
$p_B = a s_B + 2 e_B$

$p_A$ →

$g_B \leftarrow_\sigma R_q$
$k_B = p_A s_B + 2 g_B$
$w_B = \mathsf{Sig}(k_B)$

Ding, Xie, Lin (eprint 2012/688)

# RLWE-based key exchange

$E := \{-\lfloor \frac{q}{4} \rfloor, \ldots, \lfloor \frac{q}{4} \rceil\}$

$$\mathsf{Sig}(v) = \begin{cases} 0 & \text{if } v \in E \\ 1 & \text{otherwise} \end{cases}$$

Public: $a \leftarrow_\$ R_q$

Alice

$s_A \leftarrow_\$ \chi_\alpha, e_A \leftarrow_\$ \chi_\alpha$
$p_A = a s_A + 2 e_A$

Bob

$s_B \leftarrow_\$ \chi_\alpha, e_B \leftarrow_\$ \chi_\alpha$
$p_B = a s_B + 2 e_B$

$p_A$ →

$g_B \leftarrow_\sigma R_q$
$k_B = p_A s_B + 2 g_B$
$w_B = \mathsf{Sig}(k_B)$

← $p_B \quad w_B$

$g_A \leftarrow_\sigma R_q$
$k_A = p_B s_A + 2 g_A$

Ding, Xie, Lin (eprint 2012/688)

$E := \{-\lfloor \frac{q}{4} \rfloor, \ldots, \lfloor \frac{q}{4} \rceil\}$

$\mathsf{Sig}(v) = \begin{cases} 0 & \text{if } v \in E \\ 1 & \text{otherwise} \end{cases}$

# RLWE-based key exchange

Public: $a \leftarrow_{\$} R_q$

Alice

$s_A \leftarrow_{\$} \chi_\alpha, e_A \leftarrow_{\$} \chi_\alpha$
$p_A = a s_A + 2 e_A$

Bob

$s_B \leftarrow_{\$} \chi_\alpha, e_B \leftarrow_{\$} \chi_\alpha$
$p_B = a s_B + 2 e_B$

Alice → Bob: $p_A$

$g_B \leftarrow_\sigma R_q$
$k_B = p_A s_B + 2 g_B$
$w_B = \mathsf{Sig}(k_B)$

Bob → Alice: $p_B \quad w_B$

$g_A \leftarrow_\sigma R_q$
$k_A = p_B s_A + 2 g_A$

$sk_A = \mathsf{Mod}_2(k_A, w_B)$

$sk_B = \mathsf{Mod}_2(k_B, w_B)$

$\mathsf{Mod}_2 : \mathbb{Z}_q \times \{0,1\} \to \{0,1\}$

$\mathsf{Mod}_2(v, w) = (v + w \cdot \frac{q-1}{2}) \mod q \mod 2$

Ding, Xie, Lin (eprint 2012/688)

# RLWE-based key exchange

Public: e.g. $q = 17, a[0] = 9$

Alice

$p_A[0] = 9 \cdot (-2) + 2 \cdot (-2) = -5$

$p_B[0] = 9 \cdot (1) + 2 \cdot (-1) = 7$

Bob

$p_A$ →

Ding, Xie, Lin (eprint 2012/688)

# RLWE-based key exchange

Public: e.g. $q = 17, a[0] = 9$

Alice

$p_A[0] = 9 \cdot (-2) + 2 \cdot (-2) = -5$

Bob

$p_B[0] = 9 \cdot (1) + 2 \cdot (-1) = 7$

$p_A$ →

$$\begin{aligned} \mathsf{Sig}(k_B[0]) &= \mathsf{Sig}(-5 \cdot 1 + 2 \cdot (-1)) \\ &= \mathsf{Sig}(-7) = 1 \end{aligned}$$

← $p_B \quad w_B$

# RLWE-based key exchange

Public: e.g. $q = 17, a[0] = 9$

Alice

$p_A[0] = 9 \cdot (-2) + 2 \cdot (-2) = -5$

$p_B[0] = 9 \cdot (1) + 2 \cdot (-1) = 7$

Bob

$p_A$ →

$$\mathsf{Sig}(k_B[0]) = \mathsf{Sig}(-5 \cdot 1 + 2 \cdot (-1)) = \mathsf{Sig}(-7) = 1$$

← $p_B \quad w_B$

$$\mathsf{Mod}_2(k_A[0], w_B[0]) = \mathsf{Mod}_2(-12, 1) = -12 + 1 \cdot 8 \mod q \mod 2 = 1$$

$$\mathsf{Mod}_2(k_B[0], w_B[0]) = \mathsf{Mod}_2(-7, 1) = -7 + 1 \cdot 8 \mod q \mod 2 = 1$$

Ding, Xie, Lin (eprint 2012/688)

# Attacking basic RLWE key exchange

"Alice"
(Eve)

Bob

$p_A = ??$

$p_A$ →

Goal: Find $s_B$.

$w_B = \text{Sig}(p_A s_B + 2g_B)$

← $p_B \quad w_B$

Fluhrer (eprint 2016/085), Ding, Aslayigh, RV, Fluhrer, Lin (eprint 2016/1176)

# Attacking basic RLWE key exchange

“Alice”
(Eve)

Bob

$p_A = ??$

$p_A$ →

Goal: Find $s_B$.

$w_B = \mathsf{Sig}(p_A s_B + 2g_B)$

← $p_B \;\; w_B$

$$E := \{-\lfloor \tfrac{q}{4} \rfloor, \ldots, \lfloor \tfrac{q}{4} \rceil\}$$

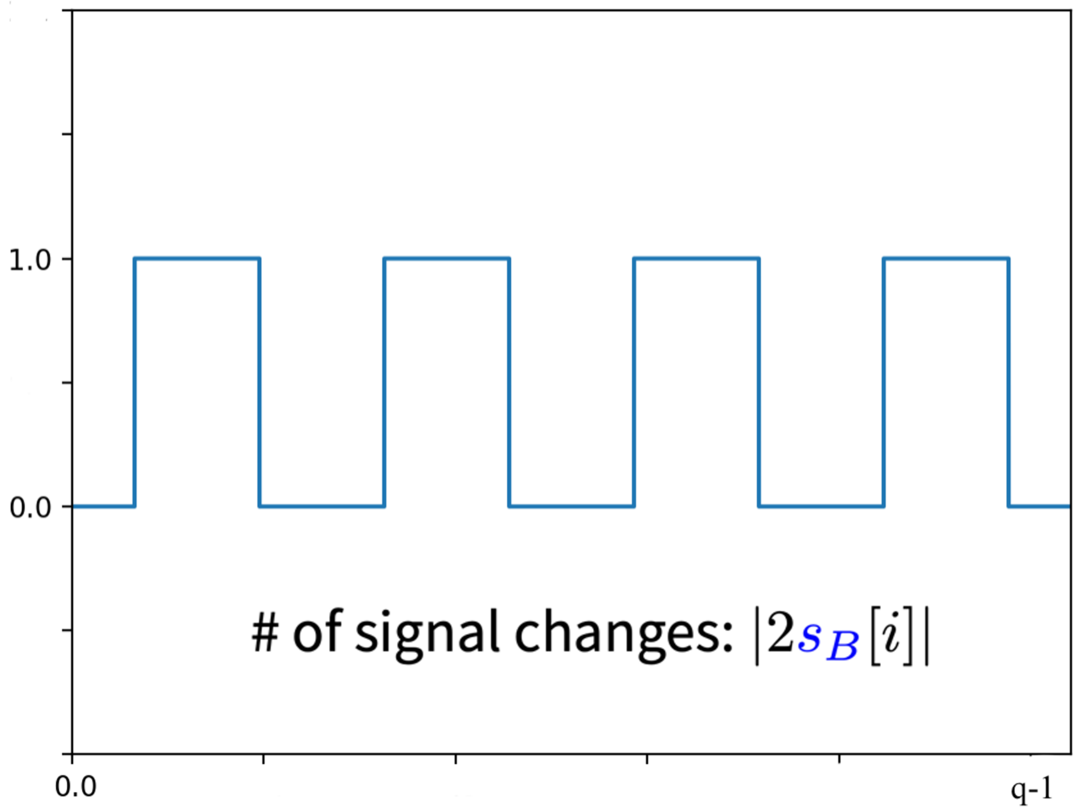
$$\mathsf{Sig}(v) = \begin{cases} 0 & \text{if } v \in E \\ 1 & \text{otherwise} \end{cases}$$

Fluhrer (eprint 2016/085), Ding, Aslayigh, RV, Fluhrer, Lin (eprint 2016/1176)

# Attacking basic RLWE key exchange

$w_B = \mathrm{Sig}(p_A s_B + 2g_B)$ e.g. $s_B[i] = -4, g_B = 0, q = 17$

$$
\begin{aligned}
p_A &= 0, p_A s_B[i] = 0, \quad w_B[i] = 0 \\
p_A &= 1, p_A s_B[i] = -4, w_B[i] = 0 \\
p_A &= 2, p_A s_B[i] = -8, w_B[i] = 1 \\
p_A &= 3, p_A s_B[i] = 5, \quad w_B[i] = 1 \\
p_A &= 4, p_A s_B[i] = 1, \quad w_B[i] = 0 \\
p_A &= 5, p_A s_B[i] = -3, w_B[i] = 0 \\
p_A &= 6, p_A s_B[i] = -7, w_B[i] = 1 \\
p_A &= 7, p_A s_B[i] = 6, \quad w_B[i] = 1 \\
p_A &= 8, p_A s_B[i] = 2, \quad w_B[i] = 0 \\
p_A &= 9, p_A s_B[i] = -2, w_B[i] = 0 \\
p_A &= 10, p_A s_B[i] = -6, w_B[i] = 1 \\
p_A &= 11, p_A s_B[i] = 7, \quad w_B[i] = 1 \\
p_A &= 12, p_A s_B[i] = 3, \quad w_B[i] = 0 \\
p_A &= 13, p_A s_B[i] = -1, w_B[i] = 0 \\
p_A &= 14, p_A s_B[i] = -5, w_B[i] = 1 \\
p_A &= 15, p_A s_B[i] = 8, \quad w_B[i] = 1 \\
p_A &= 16, p_A s_B[i] = 4, \quad w_B[i] = 0
\end{aligned}
$$




Fluhrer (eprint 2016/085), Ding, Aslayigh, RV, Fluhrer, Lin (eprint 2016/1176)

# Attacking basic RLWE key exchange

"Alice"
(Eve)

$p_A = 0$ →

← $p_B$ $\quad w_B = \mathsf{Sig}(p_A s_B + 2g_B)$

Bob

Goal: Find $s_B$.

$p_A = 1$ →

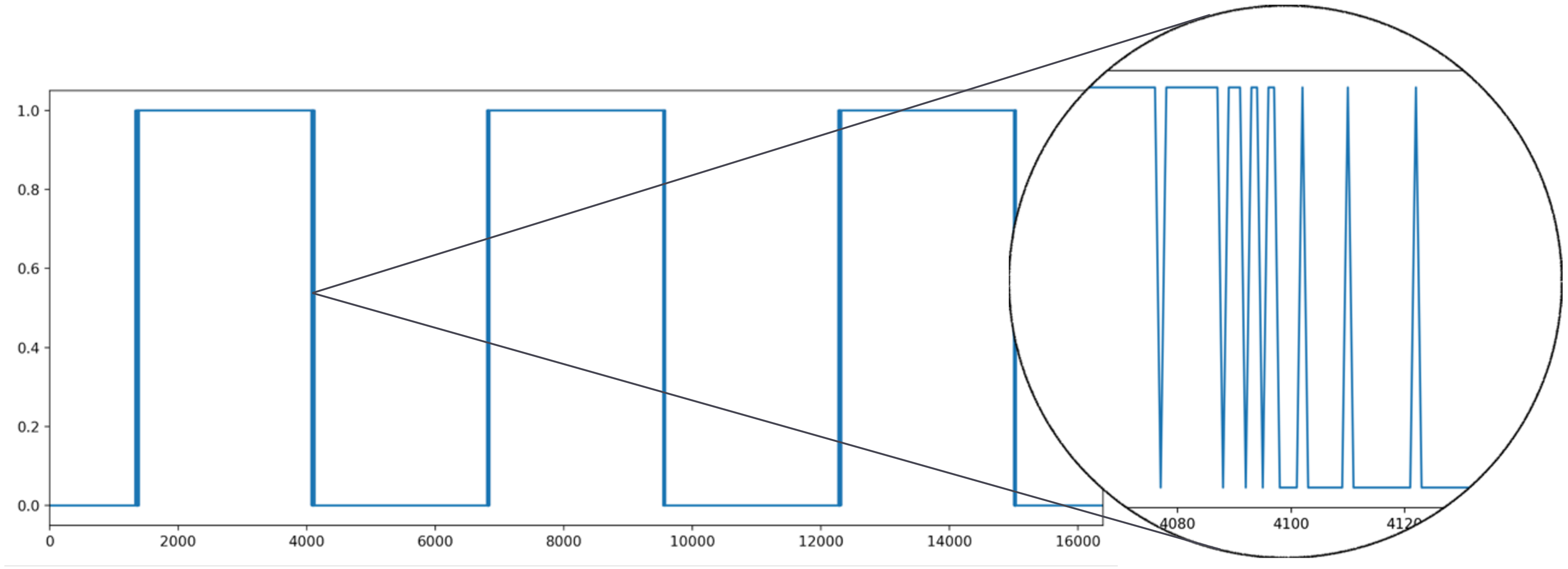
← $p_B$ $\quad w'_B = \mathsf{Sig}(p_A s_B + 2g'_B)$

⋮

Note: $s_B$ stays the same when Bob *reuses keys*.

Fluhrer (eprint 2016/085), Ding, Aslayigh, RV, Fluhrer, Lin (eprint 2016/1176)

# Attacking basic RLWE key exchange




e.g. signals received $s_B[i] = 3, q = 16385$

# Attacking basic RLWE key exchange

“Alice”
(Eve)

$$p_A = k, k \in \{0, q-1\}$$

$$p_A = (1+x)k, k \in \{0, q-1\}$$

Bob

$$k_B = (1+x)ks_B + 2g_B$$

Goal: Find $s_B$.

We have: $|s_B|$

What about signs?

$$k_B[0] = s_B[0] - s_B[n-1] + 2g_B[0]$$
$$k_B[1] = s_B[0] + s_B[1] + 2g_B[1]$$
$$k_B[2] = s_B[1] + s_B[2] + 2g_B[2]$$
$$k_B[3] = s_B[2] + s_B[3] + 2g_B[3]$$
$$\vdots$$

Relative signs ←

We now have: $s_B$ or $-s_B$

# Attacking basic RLWE key exchange

- Absolute value recovery:
  - $q$ queries
- Relative sign recovery:
  - $zq$ queries
- E.g. 26 million samples

- $q$: modulus
- $z$: number of consecutive zeroes

- [DARFL17]: $(1+z)q$
- [DFR18]: $32000n^2\alpha$
- [DRF18]: $(1+z)q/2 + O(1)$

- n: polynomial degree
- $\alpha$: standard deviation of noise

Fluhrer (eprint 2016/085), Ding, Aslayigh, RV, Fluhrer, Lin (eprint 2016/1176)

# RLWE key exchange designed for key reuse

# RLWE key exchange designed for key reuse

- Ding, Branco, Schmitt (eprint 2019/665)
- Uses a technique called “pasteurization” to disrupt Bob’s computations

# RLWE key exchange designed for key reuse

Public: $a \leftarrow_\$ R_q$

Alice

$s_A \leftarrow_\$ \chi_\alpha, e_A \leftarrow_\$ \chi_\alpha$

$p_A = as_A + 2e_A$

Bob

$s_B \leftarrow_\$ \chi_\alpha, e_B \leftarrow_\$ \chi_\alpha$

$p_B = as_B + 2e_B$

Alice → Bob: $p_A$

$f_B \leftarrow_\$ \chi_\alpha$

$g_B \leftarrow_\$ \chi_\alpha$

$\overline{p_A} = p_A + ac + 2f_B$

$k_B = \overline{p_A}(s_B + d) + 2g_B$

$w_B = \mathsf{Sig}(k_B)$

MQV?

Bob → Alice: $p_B \quad w_B$

( $k_A$ computed similarly)

$sk_A = \mathsf{Mod}_2(k_A, w_B)$

$sk_B = \mathsf{Mod}_2(k_B, w_B)$

$c = H_1(\text{“Alice”}, \text{“Bob”}, p_A)$

$d = H_1(\text{“Alice”}, \text{“Bob”}, p_A, p_B)$

Ding, Branco, Schmitt (eprint 2019/665)

# RLWE key exchange designed for key reuse

What's new with $w_B$? $w_B = \mathsf{Sig}(k_B)$

$$\begin{aligned} k_B &= \overline{p_A}(s_B + d) + 2g_B \\ &= (p_A + ac + 2f_B)(s_B + d) + 2g_B \\ &= p_A s_B + p_A d + ac s_B + acd + 2 f_B s_B + 2 f_B d + 2 g_B \\ &= p_A s_B + \underbrace{p_A d + p_B c + acd}_{\text{known value}} + \underbrace{2 f_B s_B + 2 f_B d + 2 g_B - 2 c e_B}_{\text{error term}} \end{aligned}$$

Known value:

$$\begin{aligned} p_A d + p_B c + acd = {} & p_A H_1(\text{“Alice”}, \text{“Bob”}, p_A, p_B) + p_B H_1(\text{“Alice”}, \text{“Bob”}, p_A) \\ & + a H_1(\text{“Alice”}, \text{“Bob”}, p_A) H_1(\text{“Alice”}, \text{“Bob”}, p_A, p_B) \end{aligned}$$

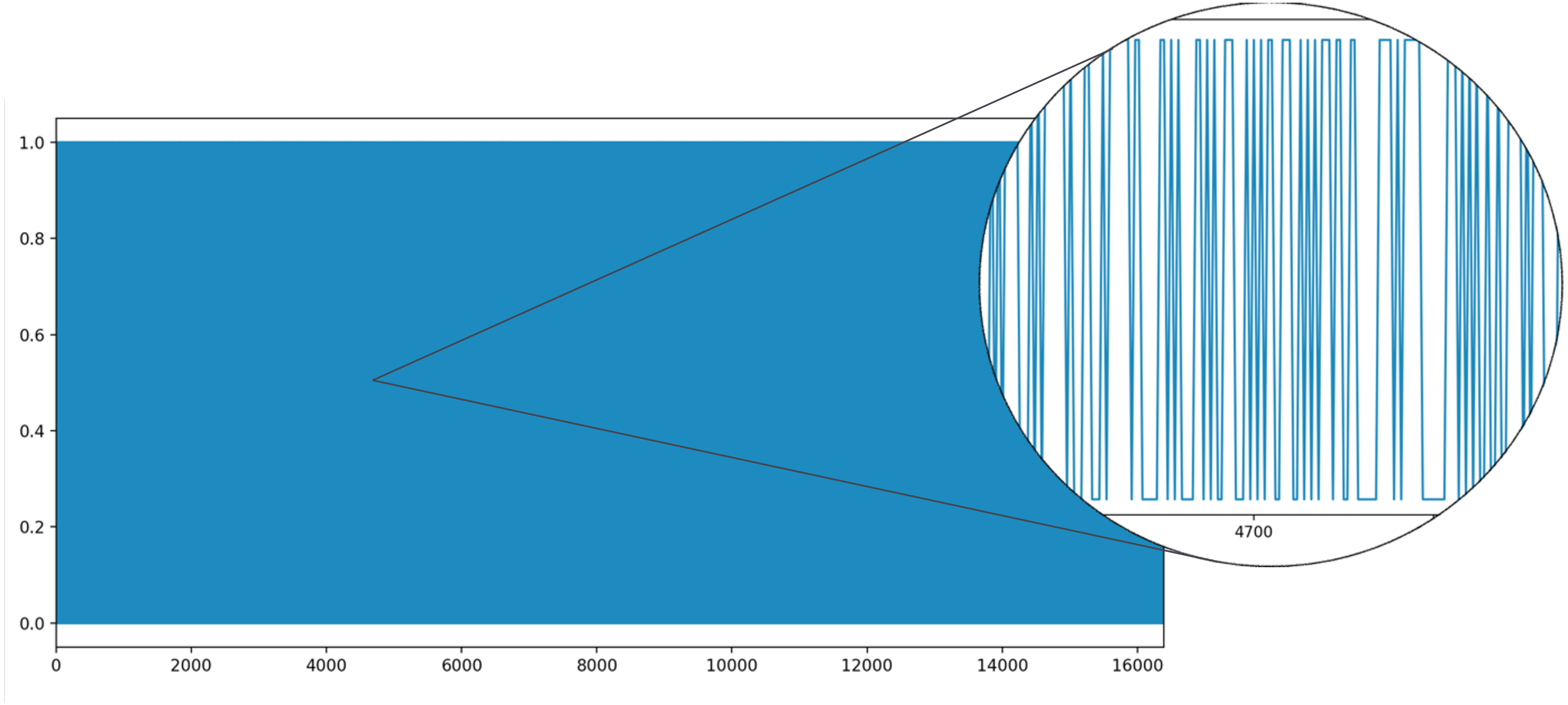
Fixed when $p_A$ is fixed

Varies when Eve's "identity" changes:

$$H_1(\text{“Alice”}, \text{“Bob”}, p_A) \neq H_1(\text{“Charlie”}, \text{“Bob”}, p_A) \neq H_1(\text{“Dan”}, \text{“Bob”}, p_A)$$

**Claim**: The signal function does **not** leak any information about the key $s_B$ ,
even when the same keys are reused.

# RLWE key exchange designed for key reuse




e.g. signals received $s_B[i] = 3, q = 16385$

# Attacking RLWE KEX with key reuse

What's new with $w_B$? $w_B = \text{Sig}(k_B)$

$$
\begin{aligned}
k_B &= \overline{p_A}(s_B + d) + 2g_B \\
&= (p_A + ac + 2f_B)(s_B + d) + 2g_B \\
&= p_A s_B + p_A d + ac s_B + acd + 2f_B s_B + 2f_B d + 2g_B \\
&= p_A s_B + \underbrace{p_A d + p_B c + acd}_{\text{known value}} + \underbrace{2f_B s_B + 2f_B d + 2g_B - 2ce_B}_{\text{error term}}
\end{aligned}
$$

Known value:

$$
\begin{aligned}
p_A d + p_B c + acd = {} & p_A H_1(\text{“Alice”}, \text{“Bob”}, p_A, p_B) + p_B H_1(\text{“Alice”}, \text{“Bob”}, p_A) \\
& + a H_1(\text{“Alice”}, \text{“Bob”}, p_A) H_1(\text{“Alice”}, \text{“Bob”}, p_A, p_B)
\end{aligned}
$$

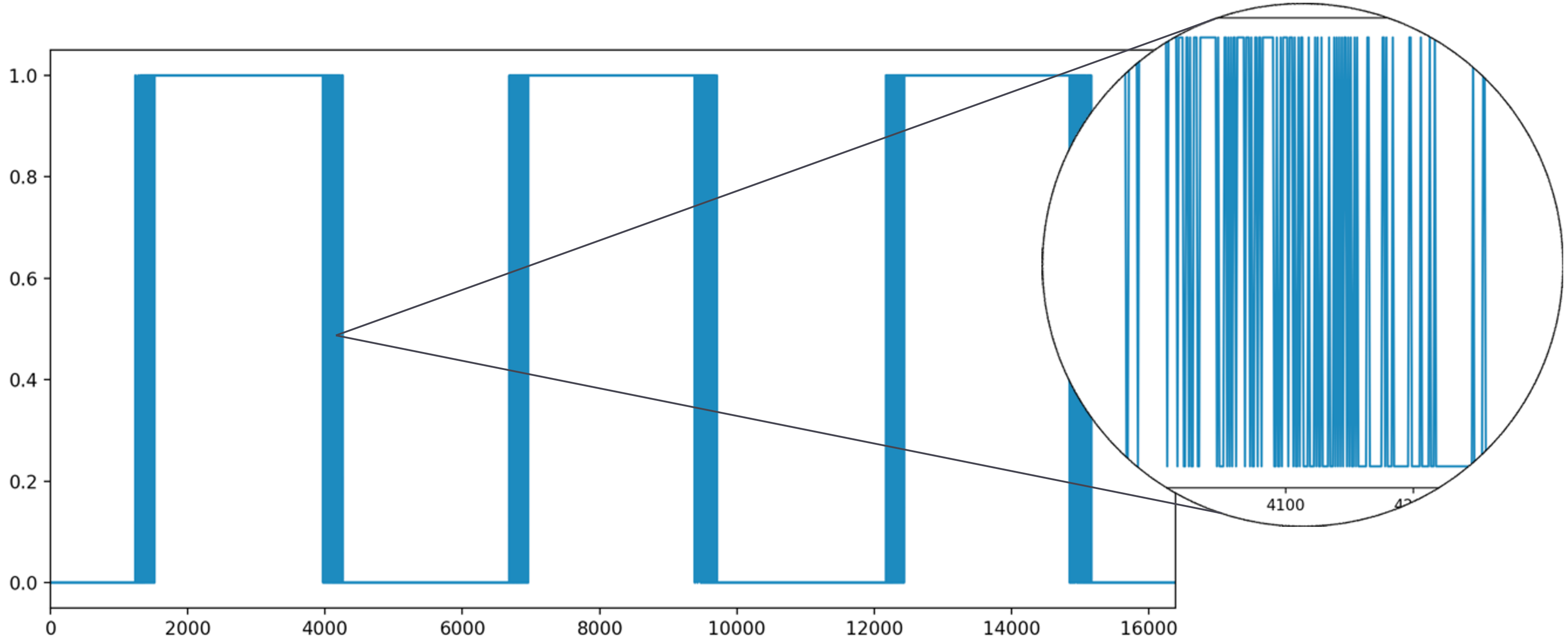
Fixed when $p_A$ is fixed

Varies when Eve's "identity" changes:

$$H_1(\text{“Alice”}, \text{“Bob”}, p_A) \neq H_1(\text{“Charlie”}, \text{“Bob”}, p_A) \neq H_1(\text{“Dan”}, \text{“Bob”}, p_A)$$

**Our observation:**
1/q of the time, the known value will be 0, and we've reduced to the previous protocol (and attack)

# Attacking RLWE KEX with key reuse




e.g. signals received $s_B[i] = 3, q = 16385,$ |known value| $\leq 500$

# Attacking RLWE KEX with key reuse

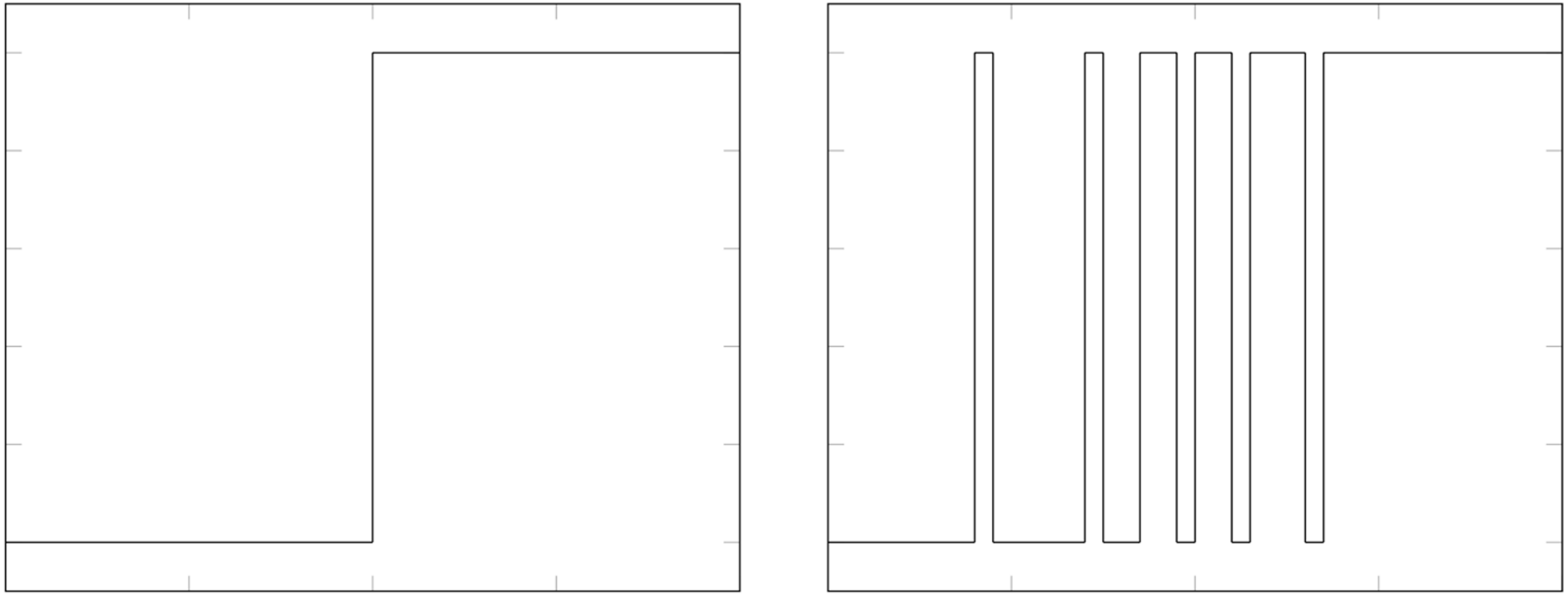
Eve

1. Send $p_A = k, k \in \{0, q-1\}$
2. Collect signals when $|p_A d + p_B c + acd| \leq h$ for some bound $h$.
   Otherwise, try step 1 again with new "identity".
3. Repeat with $p_A = (1+x)k, k \in \{0, q-1\}$ to collect relative signs.

Bob

**Claim**: The signal function does **not** leak any information about the key $s_B$ , even when the same keys are reused. **FALSE.**

# Improving attacks: sparse signal collection



Goal: count 1 signal change

# Sparse signal collection

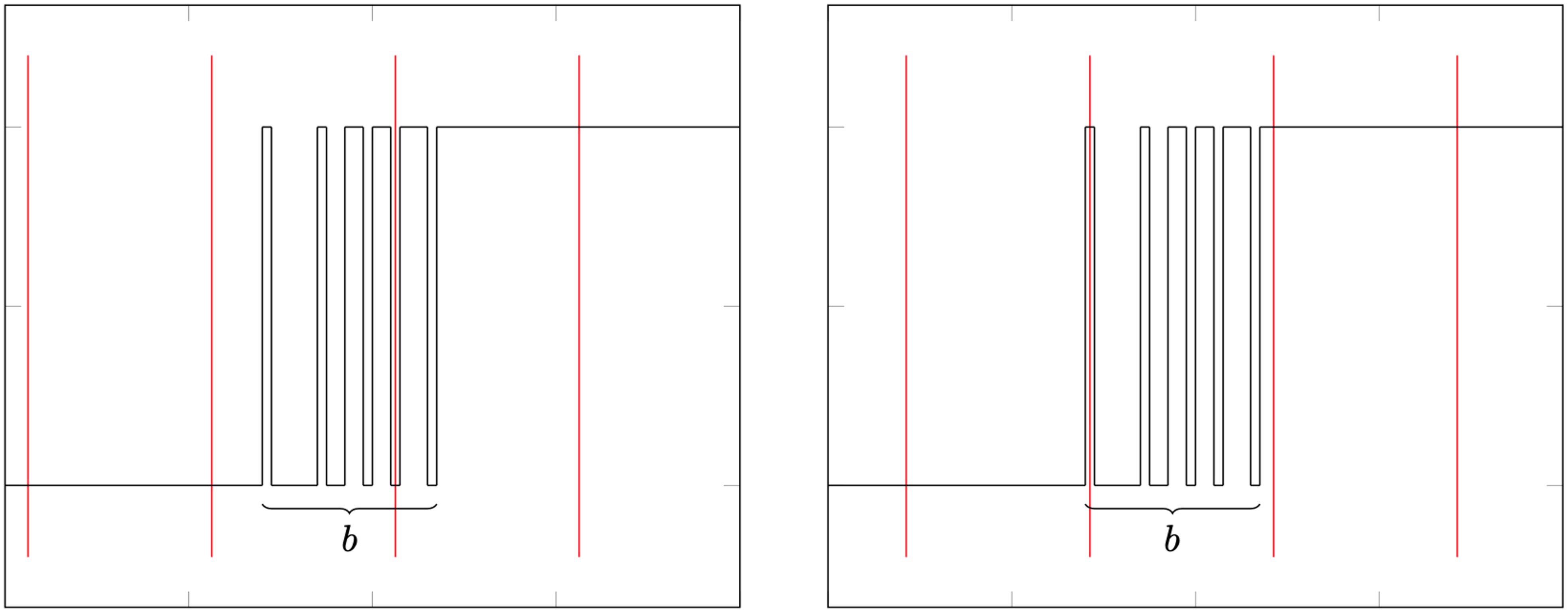
$$k_B = p_A s_B + \underbrace{p_A d + p_B c + acd}_{\text{known value}} + \underbrace{2 f_B s_B + 2 f_B d + 2 g_B - 2 c e_B}_{\text{error term}}$$

known value → bounded by some $h$

error term → normally distributed with standard deviation $\sqrt{12n\sigma^4 + 4\sigma^2}$.

We can determine $b$ = maximum width of the “noisy period”.

# Sparse signal collection




Collect every $b+1$ signal value.

# Sparse signal collection in action

## Attacking plain RLWE key exchange (DXL12)

| | [DARFL16] | Our work |
|---|---|---|
| **n = 1024, q = 16385** | 3.8 hours | 1 minute |

## Attacking RLWE KEX with key exchange (DBS19)

| | Our work |
|---|---|
| **n = 512, q = 26 038 273** | 17 minutes, 14 seconds |
| **n = 1024, q = 28 434 433** | 49 minutes |

# Key exchange protocol designs

| Protocol | Shared secret | Error correction | Security model |
|---|---|---|---|
| *DH-based key exchange* | | | |
| DH [9] | $g^{r_A r_B}$ | — | passive |
| HMQV [20] | $g^{(r_A+cs_A)(r_B+ds_B)}$ | — | CK with wFS |
| CMQV [31] | $g^{(\tilde{r}_A+cs_A)(\tilde{r}_B+ds_B)}$ | — | eCK |
| *LWE-based public key encryption and key exchange* | | | |
| Regev [29], LPR [23] | $\approx ar_A r_B$ | rounding | IND-CPA |
| DXL [16] | $\approx ar_A r_B$ | signal fn. | passive |
| Peikert [26], BCNS [5] | $\approx ar_A r_B$ | reconciliation | passive |
| ZZDSD [32] | $\approx a(r_A + cs_A)(r_B + ds_B)$ | signal fn. | BR with wFS |
| DBS reusable [12] | $\approx a(s_A + c)(s_B + d)$ | signal fn. | key reuse robustness |
| DBS AKE [12] | $\approx a(r_A + s_A + c)(r_B + s_B + d)$ | signal fn. | BR with wFS |

We couldn't figure out how to attack ZZDSD or DBS AKE in BR model

But can apply our technique to attack DBS AKE in eCK model

# Wrapping up

# Open questions

## Making post-quantum AKE

- Non-interactive key exchange (NIKE)
- Static-static key exchange
- eCK-secure constructions directly from LWE
  - True MQV analogue?
- Certificate lifecycle management for KEM keys
- Noise, Signal, …

## Breaking PQ AKE

- Key reuse attacks against ZZDSD and DBS AKE in BR-PFS?

# Making and breaking implicitly authenticated post-quantum key exchange

Douglas Stebila
University of Waterloo

## KEMTLS

Implicitly authenticated TLS without handshake signatures using KEMs

https://eprint.iacr.org/2020/534
https://github.com/thomwiggers/kemtls-experiment/
https://openquantumsafe.org

## Attacks on RLWE key reuse

Faster sparse signal collection and insecurity of DBS key reuse protocol

https://eprint.iacr.org/2020/1288
https://git.uwaterloo.ca/ssveitch/improved-key-reuse

https://www.douglas.stebila.ca/research/presentations/