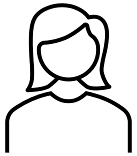
Cryptography Engineering

- Lecture 13 (Feb 04^{th}, 2026)
- Today's notes:
  - Some attacks on Cryptosystems (and how to prevent them)
  - Toward Post-Quantum Cryptography

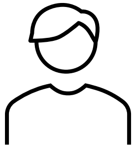
Attacks using Invalid Inputs

- The adversary sends data that violates the protocol or data format.
- **Example: DHKE**



$(\mathbb{G}, g, q)$:

A q-order group $\mathbb{G}$ with a generator g



$x \leftarrow_\$ \mathbb{Z}_q$ $\qquad$ $y \leftarrow_\$ \mathbb{Z}_q$

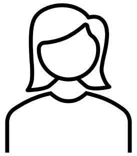
$$\xrightarrow{X = g^x}$$

$$\xleftarrow{Y = g^y}$$

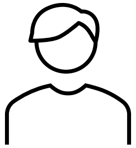
$K_{\text{Alice}} = Y^x$ $\qquad$ $K_{\text{Bob}} = X^y$

Attacks using Invalid Inputs

- The adversary sends data that violates the protocol or data format.
- **Example: DHKE**



$(\mathbb{G}, g, q)$:
A q-order group $\mathbb{G}$ with a generator g



Alice		Bob
$x \leftarrow_\$ \mathbb{Z}_q$	$X = g^x$ $\longrightarrow$	$y \leftarrow_\$ \mathbb{Z}_q$
	$Y = g^y$ $\longleftarrow$	
$K_{\text{Alice}} = Y^x$		$K_{\text{Bob}} = X^y$

- The security holds **if the protocol runs on specific groups**
- **What if we use an element outside the group $\mathbb{G}$?**

Attacks using Invalid Inputs

- The adversary sends data that violates the protocol or data format.
- **Example: DHKE**

$(\mathbb{G}, g, q)$:

A $\boldsymbol{q}$-order group $\mathbb{G}$ with a generator $\boldsymbol{g}$

- $\mathbb{G}$ can be a subgroup of another group $\mathbb{G}'$
- Co-factor: $|\mathbb{G}'|/|\mathbb{G}|$ (the h value on the RHS figure)

Curve1174

251-bit prime field Weierstrass curve.

Curve from https://eprint.iacr.org/2013/325.pdf

$$y^2 \equiv x^3 + ax + b$$

Parameters

Name	Value
p	0x7fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff7
a	0x486BE25B34C8080922B969257EEB54C404F914A29067A5560BB9AEE0BC67A6D
b	0xE347A25BF875DD2F1F12D8A10334D417CC15E77893A99F4BF278CA563072E6
G	(0x3BE821D63D2CD5AFE0504F452E5CF47A60A10446928CEAECFD5294F89B45051, 0x66FE4E7B8B6FE152F743393029A61BFB839747C8FB00F7B27A6841C07532A0)
n	0x1FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF77965C4DFD307348944D45FD166C971
h	0x04

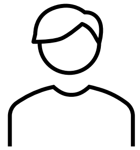
Source: https://neuromancer.sk/std/other/Curve1174

Attacks using Invalid Inputs

- The adversary sends data that violates the protocol or data format.
- **Example: DHKE**

$(\mathbb{G}, g, q)$:
A q-order group $\mathbb{G}$ with a generator g

- $\mathbb{G}$ can be a subgroup of another group $\mathbb{G}'$
- Co-factor: $|\mathbb{G}'|/|\mathbb{G}|$ (the h value on the RHS figure)
- **Use the co-factor to check group membership**



$$\xrightarrow{X = g^x}$$

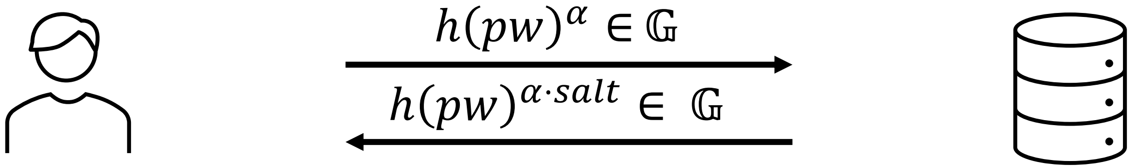
Check $X^h = 1$?
// 1 is the identity group element
If so, reject
else:

$$y \leftarrow_\$ \mathbb{Z}_q$$

Attacks using Invalid Inputs

- **Toy Example of attacking OPAQUE:**

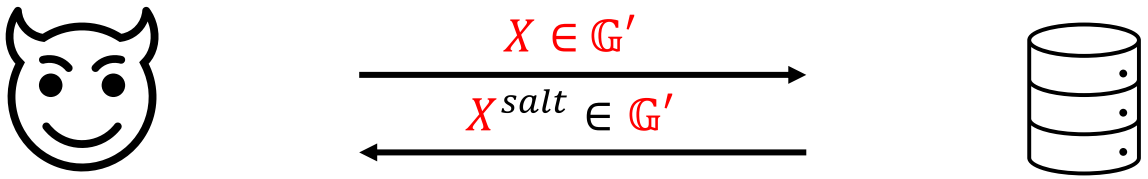
$(\mathbb{G} \subset \mathbb{G}', g, q, h = 2)$:
A q-order group $\mathbb{G}$ with a generator g, and $|\mathbb{G}'|/|\mathbb{G}| = h$



Attacks using Invalid Inputs

- **Toy Example of attacking OPAQUE:**

$(\mathbb{G} \subset \mathbb{G}', g, q, h = 2)$:
A $\boldsymbol{q}$-order group $\mathbb{G}$ with a generator $\boldsymbol{g}$,
and $|\mathbb{G}'|/|\mathbb{G}| = h$

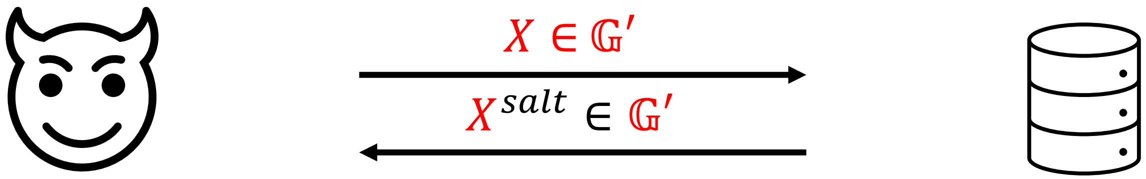


Find an element X s.t.
X's order is 2

Attacks using Invalid Inputs

- **Toy Example of attacking OPAQUE:**

$(\mathbb{G} \subset \mathbb{G}', g, q, h = 2)$:
A $\boldsymbol{q}$-order group $\mathbb{G}$ with a generator $\boldsymbol{g}$, and $|\mathbb{G}'|/|\mathbb{G}| = h$



Find an element X s.t.
X's order is 2

Little Algebra:
If X's order is 2, then $X^r = X^{(r \bmod 2)}$ **=> We can determine the parity of the salt:** $salt$ is an odd/even number if $X^{salt} = X$

Attacks using Invalid Inputs

- Other Example:
  - Invalid Curve Attacks (e.g. ECDSA): Using insecure curves.
  - Invalid public keys
  - ...

- Lessons: Follow the standards(/specifications/...), and keep updating with them...

Downgrade Attacks

- Exploit vulnerabilities in compatibility or protocol negotiation to downgrade cryptographic protocols to weaker or obsolete versions.

- Example: TLS ciphercuite negotiation
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (secure)
  - TLS_RSA_WITH_RC4_128_SHA (no forward secrecy)

- Lessons: Use the latest protocol version (such as TLS 1.3), disable insecure or outdated protocols/suites on both sides.

More Examples about Reuse

- Previous Example: Randomness Reuse in the DSA signature => Recovery of secret key

- Why shouldn't we reuse randomness?

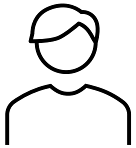
➢ An informal principle: Security of cryptosystem comes from *the secret key* and *the randomness*
  - ➢ Secret key: **High entropic, the "source" of security, ...**
  - ➢ Randomness/nonce/salt: **Independency when using the same key, Freshness, ...**

More Examples about Reuse

- Example: Reuse randomness in the Hashed ElGamal Encryption

ElGamalEnc(public_key $= g^x$, plaintext $= m$)
// $(\mathbb{G}, g, q)$: A q-order group $\mathbb{G}$ with a generator g

1. $r \leftarrow_{\$} \mathbb{Z}_q$
2. $c_0 = g^r$
3. $c_1 = H(g^{xr}) \oplus m$
4. Return (c_0, c_1)



Encrypt m and m' using the same randomness

$\xrightarrow{g^r, H(g^{xr}) \oplus m}$

$\xrightarrow{g^r, H(g^{xr}) \oplus m'}$

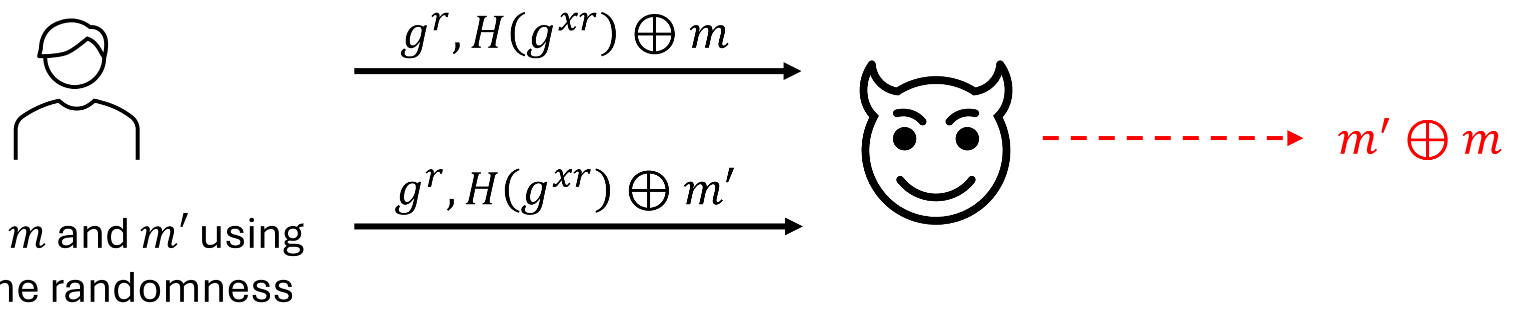
More Examples about Reuse

- Example: Reuse randomness in the Hashed ElGamal Encryption

ElGamalEnc(public_key $= g^x$, plaintext $= m$)

// $(\mathbb{G}, g, q)$: A q-order group $\mathbb{G}$ with a generator g

1. $r \leftarrow_\$ \mathbb{Z}_q$
2. $c_0 = g^r$
3. $c_1 = H(g^{xr}) \oplus m$
4. Return (c_0, c_1)



More Examples about Reuse

- Examples: Reuse salt in OPAQUE
- Suppose that Alice's password is pw_A, Bob's password is pw_B, and the password files stored in the server are:

Username:	Bob	Username:	Alice
salt:	r	salt:	r
enc_AKE_keys:	$\mathrm{AEAD}_{rw_B}(\ldots)$	enc_AKE_keys:	$\mathrm{AEAD}_{rw_A}(\ldots)$

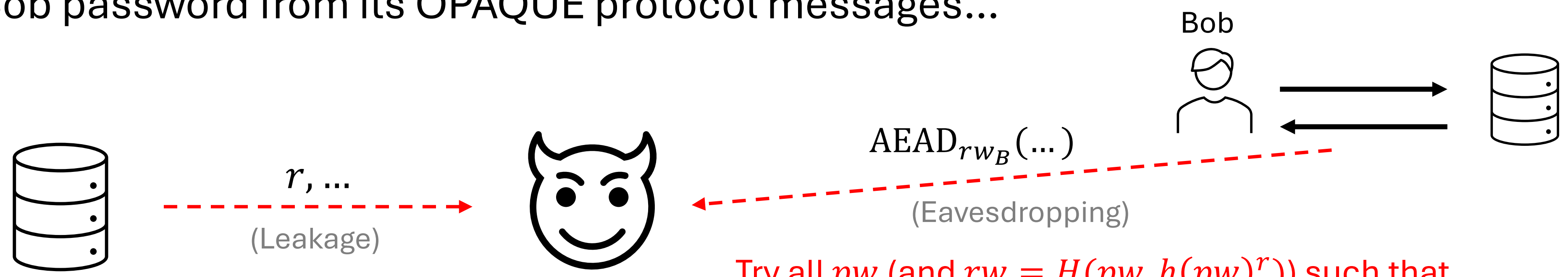
- Is it secure? Why?

More Examples about Reuse

- Examples: Reuse salt in OPAQUE
- Suppose that Alice's password is pw_A, Bob's password is pw_B, and the password files stored in the server are:

Username:	Bob	Username:	Alice
salt:	r	salt:	r
enc_AKE_keys:	$\mathrm{AEAD}_{rw_B}(\dots)$	enc_AKE_keys:	$\mathrm{AEAD}_{rw_A}(\dots)$

- **Potential risks:** If Alice's password file is leaked, then the adversary can launch offline attacks to recover Bob password from its OPAQUE protocol messages…



Try all pw (and $rw = H(pw, h(pw)^r)$) such that AEAD.Dec does not output rejection…

More Examples about Reuse

- Other examples:
  - Reuse randomness in Schnorr/Schnorr-like signature schemes…
  - Reuse of IV in the AES-GCM mode, or short IV…
  - …

Side-Channel Attacks

- Side-channel information: By-product information when the system runs cryptographic algorithms.
  - E.g., time, power consumption, cache access patterns, ...

- Example:
  - Timing Attacks
  - Cache Attacks
  - ...

- An Example of Timing Attack: A website checks a user's password character by character, returning an error as soon as it finds the first mismatch.

- Lessons: Use **constant-time algorithms**, masking sensitive operations, ...

Towards Post-Quantum Cryptography

- All previous attack examples are about **wrong implementations** of cryptographic algorithms, but not about the algorithms themselves…
  - ➢ Example: Breaking the ElGamal encryption => Solving DH problems…

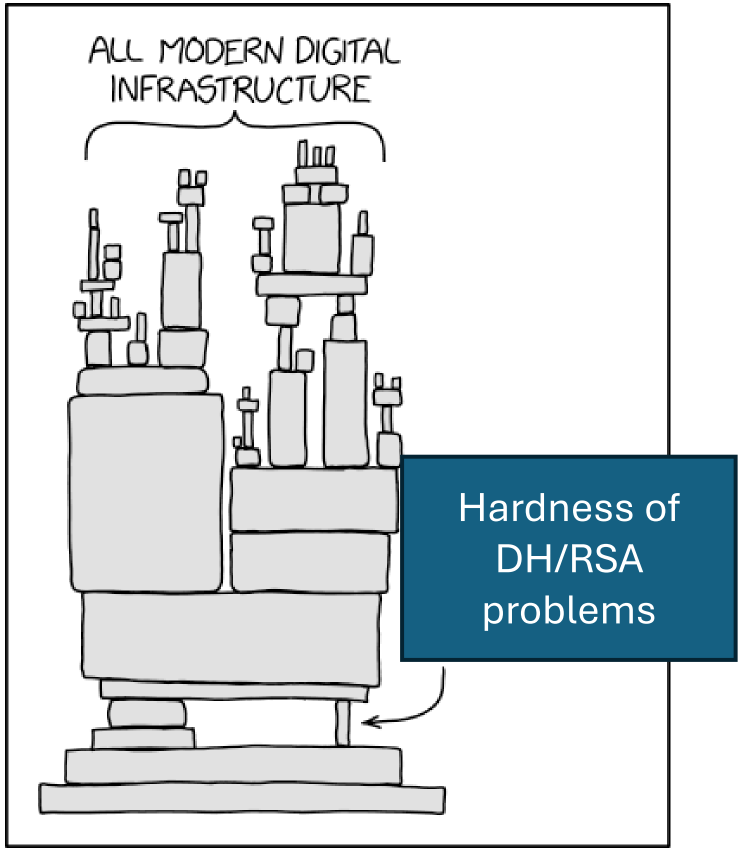
Towards Post-Quantum Cryptography

- All previous attack examples are about wrong implementations of cryptographic algorithms, but not about the algorithms themselves...
  - ➢ Example: Breaking the ElGamal encryption => Solving DH problems...

- Modern cryptography builds on **hardness assumptions**:
  - ElGamal encryption, DHKE, DSA, TLS 1.3, and others all rely on the hardness of Diffie-Hellman or RSA problems...
  - We assume these problems are hard to solve (i.e., there is no polynomial-time algorithm).

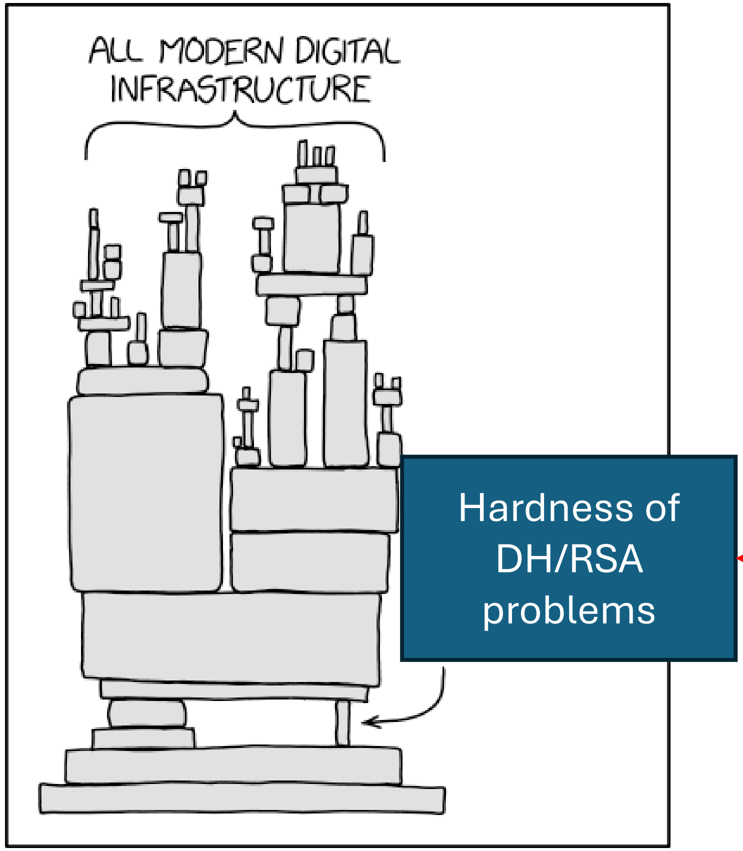
- What if these assumptions are broken?

Towards Post-Quantum Cryptography

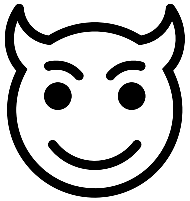


Source: xkcd/2347 and Nadia Heninger's talk in PKC2024

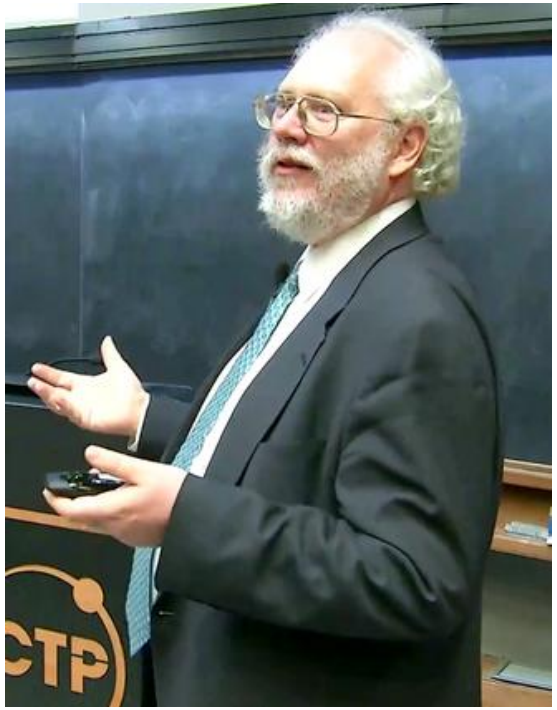
Towards Post-Quantum Cryptography



Source: xkcd/2347 and Nadia
Heninger's talk in PKC2024

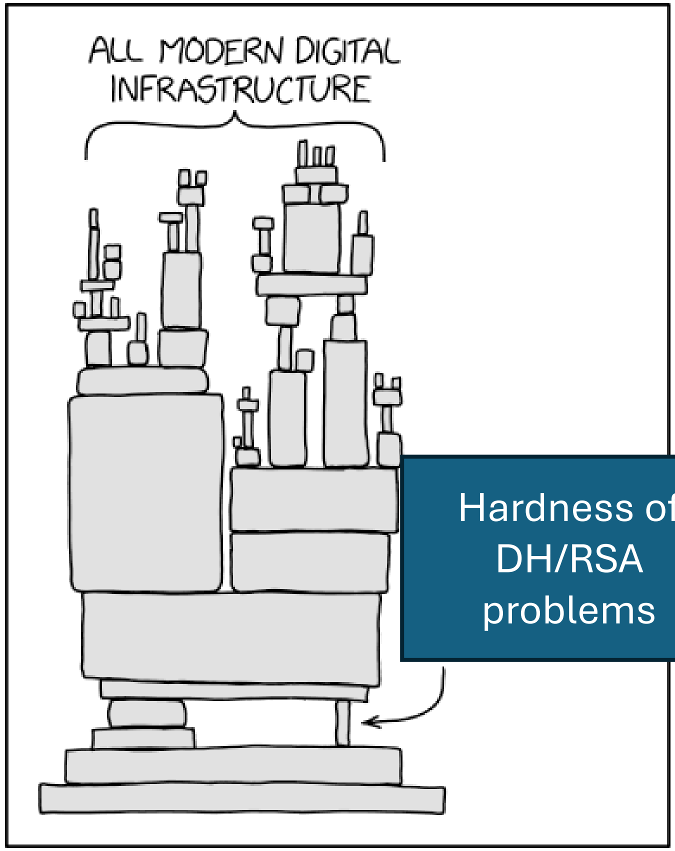


Shor's algorithm
(quantum)



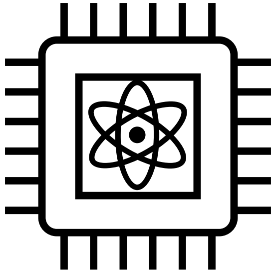
Peter Williston Shor
(image from Wikipedia)

Towards Post-Quantum Cryptography

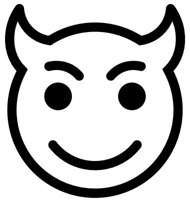


Hardness of DH/RSA problems

Source: xkcd/2347 and Nadia Heninger's talk in PKC2024



Recent progress in Quantum Computers/Mechanisms…

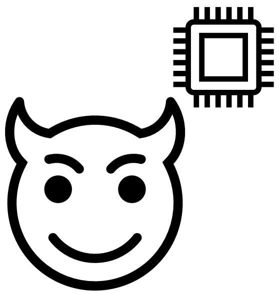


Shor's algorithm

Impact on Cryptography

- In the **pre**-quantum world…

- Symmetric-key cryptography
  - Hash functions: SHA2, SHA3,…
  - Symmetric-key (authenticated) encryption: AES, AES-GCM…
  - KDF, MAC, PRNG,…

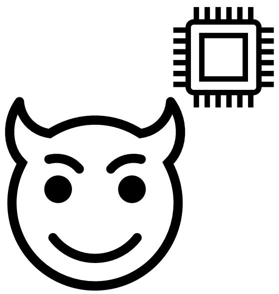


Impact on Cryptography

- In the **pre**-quantum world...

- Symmetric-key cryptography
  - Hash functions: SHA2, SHA3,...
  - Symmetric-key (authenticated) encryption: AES, AES-GCM...
  - KDF, MAC, PRNG,...

- Basis of confidence: **Extensively studied, publicly reviewed, ...**
  - (Or we could say that they themselves are assumptions...)

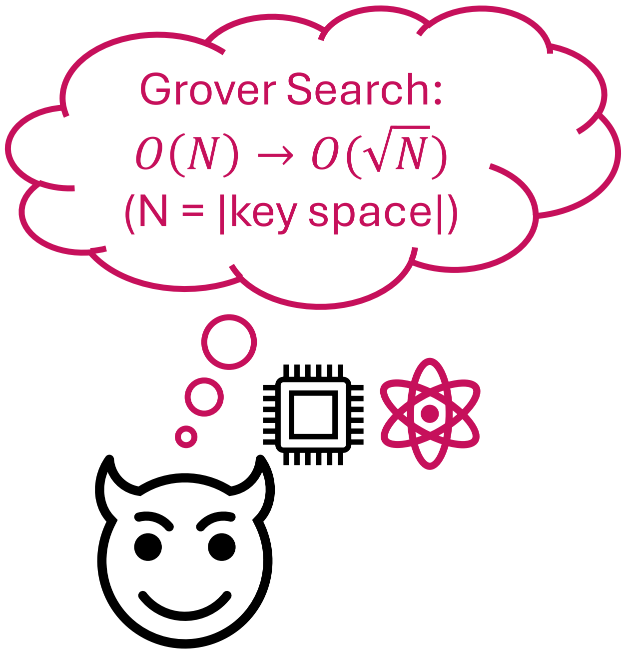


Impact on Cryptography

- In the **post**-quantum world...

- Symmetric-key cryptography
  - Hash functions: SHA2, SHA3,...
  - Symmetric-key (authenticated) encryption: AES, AES-GCM...
  - KDF, MAC, PRNG,...

- **Basis of confidence: Extensively studied, publicly reviewed, ...**

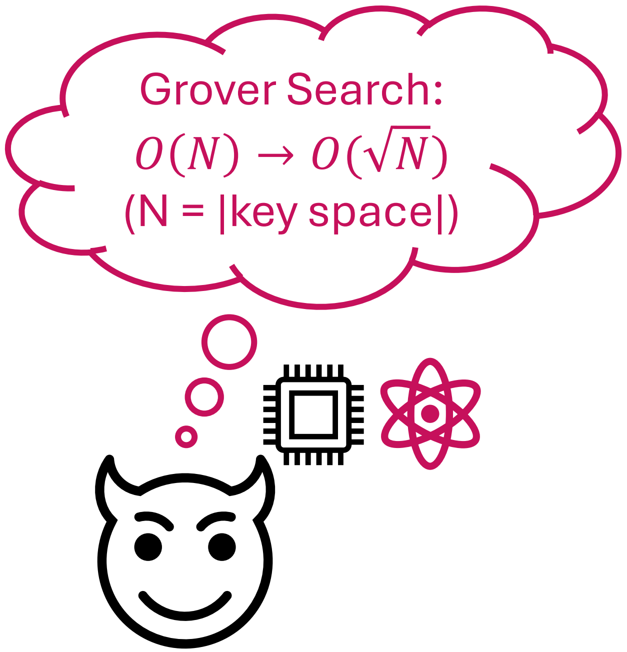


Impact on Cryptography

- In the **post**-quantum world…

- Symmetric-key cryptography
  - Hash functions: SHA2, SHA3,…
  - Symmetric-key (authenticated) encryption: AES, AES-GCM…
  - KDF, MAC, PRNG,…

- **Basis of confidence: Extensively studied, publicly reviewed, …**
- **Solution: Double the key size…** (not always true)

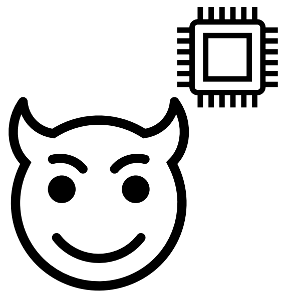


Impact on Cryptography

- In the **pre**-quantum world...

- Public-key cryptography
  - Key exchange: (EC)DHKE, TLS, ...
  - Public-key encryption: ElGamal encryption, DHIES, ...
  - Signature: DSA, RSA, ...
  - ...

- **Basis of confidence:**
  - Provable security (e.g., rigorous security proofs, ...)
  - Well-studied and publicly reviewed hardness assumptions
  - **Classical assumptions: DH (from discrete-log), RSA (from factoring), ...**

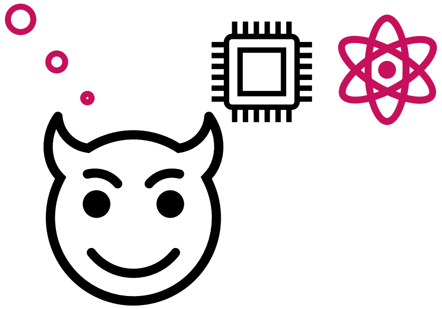


Impact on Cryptography

- In the **post**-quantum world...

- Public-key cryptography
  - Key exchange: (EC)DHKE, TLS, ...
  - Public-key encryption: ElGamal encryption, DHIES, ...
  - Signature: DSA, RSA, ...
  - ...

Quantum Fourier transform (QFT):
solve DLOG and Factoring.
$N^{O(1)} \rightarrow \boldsymbol{O(log(N))}$,
where N = group/ modulus size



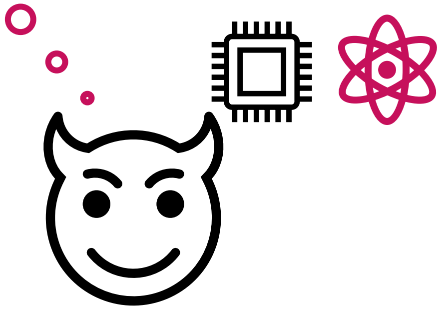
- **Basis of confidence:**
  - Provable security (e.g., rigorous security proofs, ...)
  - Well-studied and publicly reviewed hardness assumptions
  - **Classical assumptions: DH (from discrete-log), RSA (from factoring), ...**

Impact on Cryptography

- In the **post**-quantum world…

- Public-key cryptography
  - Key exchange: (EC)DHKE, TLS, …
  - Public-key encryption: ElGamal encryption, DHIES, …
  - Signature: DSA, RSA, …
  - …

Quantum Fourier transform (QFT):
solve DLOG and Factoring.
$N^{O(1)} \rightarrow \boldsymbol{O(log(N))},$
where N = group/ modulus size



- **Basis of confidence:**
  - Provable security (e.g., rigorous security proofs, …)
  - Well-studied and publicly reviewed hardness assumptions
  - ~~Classical assumptions: DH (from discrete-log), RSA (from factoring), …~~
  - **New assumptions are needed.**

Post-quantum Assumptions

- Assumptions that are believed to be **quantum-secure:**
  - Lattice-based
  - Isogeny-based
  - Code-based
  - ...

Post-quantum Assumptions

- New Direction: Post-Quantum Cryptography
  - Cryptographic algorithms run on classical computers, but **remain secure against future quantum computers**…
- Still follow the methodology of modern cryptography: Assumptions => Schemes.

- **Hardness Assumptions even against quantum adversaries:**
  - **Lattices (Crystal-Kyber/ML-KEM, Crystal-Dilithium/ML-DSA)**
  - Isogeny (of Elliptic Curves)
  - Code-based
  - …

Post-quantum Assumptions

- We have implemented some post-quantum cryptosystems (Homework 2)…
  - PQ-TLS
  - KEM-TLS
  - Both are based on ML-KEM (Kyber) and ML-DSA (Dilithium)

Transition from Pre-Quantum to Post-Quantum

- Should we immediately change everything to be post-quantum?

Transition from Pre-Quantum to Post-Quantum

- Should we immediately change everything to be post-quantum?

- Efficiency of classical algorithms v.s. post-quantum algorithms: (e.g., ECDSA v.s. CRYSTALS-Dilithium)

	ECDSA	Dilithium
sk size	~32B	~1.3KB
pk size	~32B	~2.5KB
signature size	~64B	~2.5KB
Running time	t	10~100*t

Transition from Pre-Quantum to Post-Quantum

- Should we immediately change everything to be post-quantum?

- Efficiency of classical algorithms v.s. post-quantum algorithms: (e.g., ECDSA v.s. CRYSTALS-Dilithium)

	ECDSA	Dilithium
sk size	~32B	~1.3KB
pk size	~32B	~2.5KB
signature size	~64B	~2.5KB
Running time	t	10~100*t

- Studies on classical cryptography: since 1970s
- Large-scale studies on post-quantum cryptography: since 2010s
  - SIDH, a primitive that was believed to be post-quantum secure, was broken…
  - Who is the next one?

Transition from Pre-Quantum to Post-Quantum

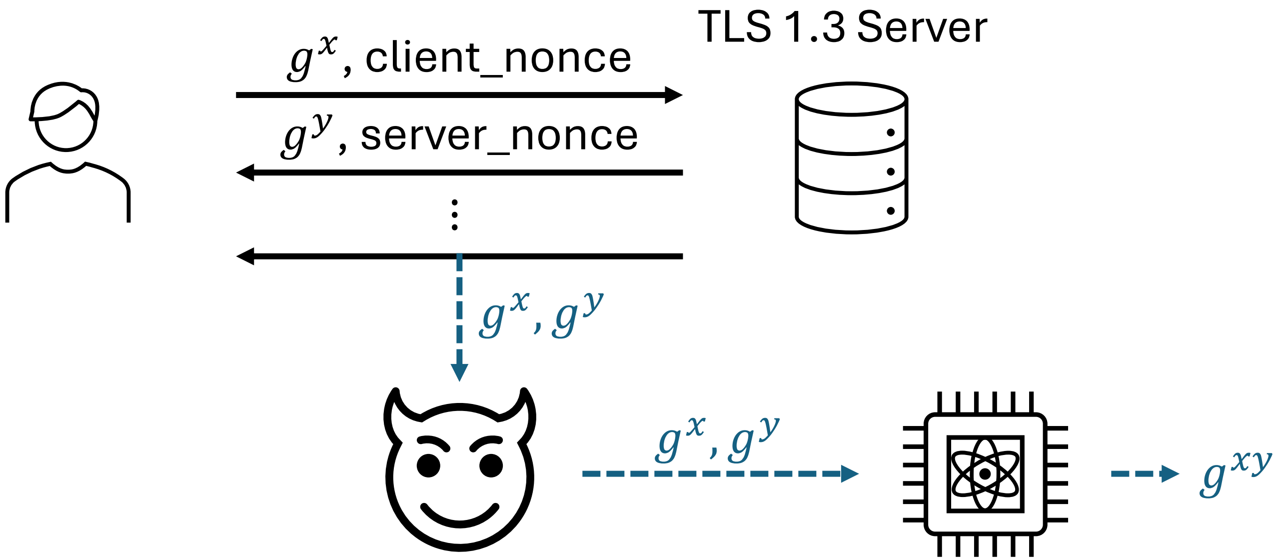
- Should we wait until the first large-scale quantum computer appears?

- “Harvest Now, Decrypt Later”: The adversary stores today’s encrypted data (harvest now). In the future, quantum computers decrypt this data (decrypt later)

Transition from Pre-Quantum to Post-Quantum

- Should we wait until the first large-scale quantum computer appears?

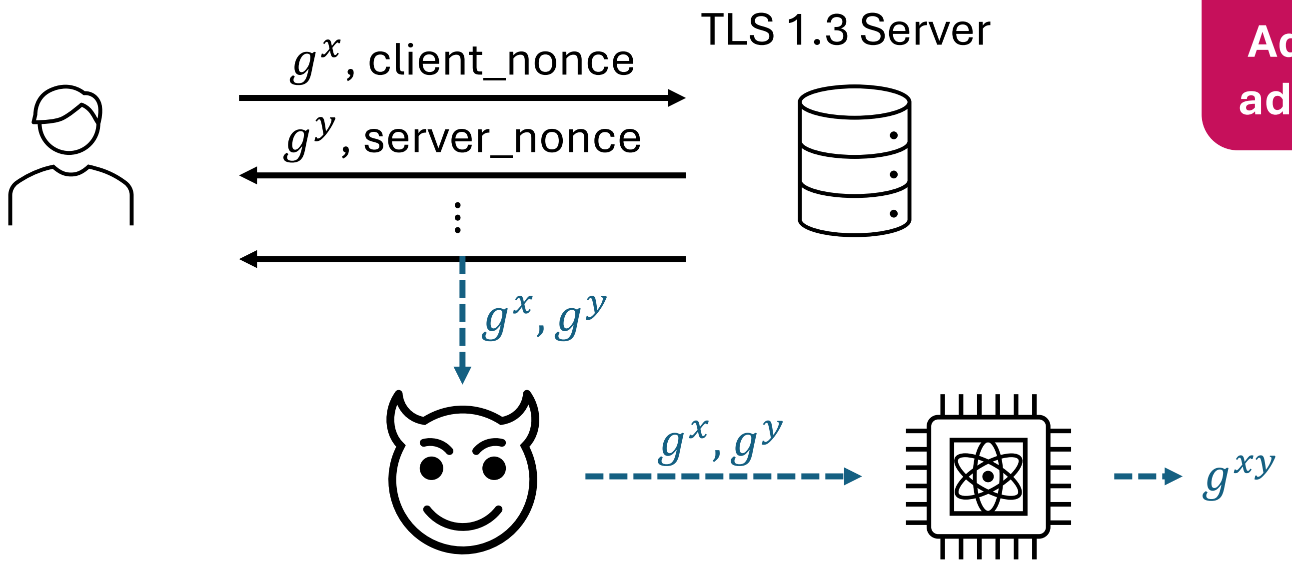
- “Harvest Now, Decrypt Later”: The adversary stores today’s encrypted data (harvest now). In the future, quantum computers decrypt this data (decrypt later)



Transition from Pre-Quantum to Post-Quantum

- Should we wait until the first large-scale quantum computer appears?

- “Harvest Now, Decrypt Later”: The adversary stores today’s encrypted data (harvest now). In the future, quantum computers decrypt this data (decrypt later)



Solution:
Add PQ-secure component so that the adversary cannot decrypt the TLS key…

Transition from Pre-Quantum to Post-Quantum

- Hybrid Cryptography
  - Classical algorithms + post-quantum algorithms
  - Example: ECDH + ECDSA in TLS 1.3 -> (ECDH + Kyber) + ECDSA

Transition from Pre-Quantum to Post-Quantum

- Hybrid Cryptography
  - Classical algorithms + post-quantum algorithms
  - Example: ECDH + ECDSA in TLS 1.3 -> (ECDH + Kyber) + ECDSA

The ECDH in TLS 1.3

$\xrightarrow{g^x\text{, client\_nonce}}$

$\xleftarrow{g^y\text{, server\_nonce, ...}}$

$\vdots$

A simple KE based on Kyber KEM

$(epk, esk) \leftarrow \text{KeyGen}$ $\xrightarrow{\quad epk \quad}$

$\xleftarrow{\quad c \quad}$ $(c, K) \leftarrow \text{Encaps}(epk)$

$K \leftarrow \text{Decaps}(esk, c)$

- Advantages: Classical security provided by ECDH + Quantum security provided by Kyber

Transition from Pre-Quantum to Post-Quantum

- Hybrid Cryptography
  - Classical algorithms + post-quantum algorithms
  - Example: ECDH + ECDSA in TLS 1.3 -> (ECDH + Kyber) + ECDSA

Modify the KE part in TLS 1.3:
ECDH+ Kyber KEM

$(epk, esk) \leftarrow \text{KeyGen}$ $\quad g^x, epk,$ client_nonce $\longrightarrow$

$\longleftarrow$ $g^y, c,$ server_nonce, ... $\quad (c, K) \leftarrow \text{Encaps}(epk)$

$K \leftarrow \text{Decaps}(esk, c)$

$\vdots$

Keys = KeySchedule(...|| g^{xy} || K ||...)

Transition from Pre-Quantum to Post-Quantum

- Hybrid Cryptography
  - Classical algorithms + post-quantum algorithms
  - Example: ECDH + ECDSA in TLS 1.3 -> (ECDH + Kyber) + ECDSA

Modify the KE part in TLS 1.3:
ECDH+ Kyber KEM

$(epk, esk) \leftarrow$ KeyGen $\quad g^x, epk,$ client_nonce $\longrightarrow$

$\longleftarrow$ $g^y, c,$ server_nonce, … $\quad (c, K) \leftarrow$ Encaps(epk)

$K \leftarrow$ Decaps(esk, c)

⋮

Keys = KeySchedule(…|| g^{xy} || K ||…)

K insecure => Keys remain secure!

Transition from Pre-Quantum to Post-Quantum

- Hybrid Cryptography
  - Classical algorithms + post-quantum algorithms
  - Example: ECDH + ECDSA in TLS 1.3 -> (ECDH + Kyber) + ECDSA

Modify the KE part in TLS 1.3:
ECDH+ Kyber KEM

$(epk, esk) \leftarrow$ KeyGen g^x, epk, client_nonce →

← g^y, c, server_nonce, … $(c, K) \leftarrow$ Encaps(epk)

$K \leftarrow$ Decaps(esk, c)

⋮

Keys = KeySchedule(…|| g^{xy} || K ||…)

g^{xy} insecure in the future => Keys remain secure!

Transition from Pre-Quantum to Post-Quantum

- Some other PQ replacements (or need to be replaced):
  - X3DH -> PQXDH -> (fully PQ-secure X3DH-style protocols...)
  - PQ-secure Password-based authentication protocols
  - PQ-secure OPRF
  - ...

Transition from Pre-Quantum to Post-Quantum

- Some other PQ replacements (or need to be replaced):
  - X3DH -> PQXDH -> (fully PQ-secure X3DH-style protocols...)
  - PQ-secure Password-based authentication protocols
  - PQ-secure OPRF
  - ...

Many open problems!