Lattice-Based Cryptography

Huijing Gong

10/21/2019

- Slides courtesy of Dana Dachman-Soled, Valeria Nikolaenko, Chris Peikert, and Oded Regev

Traditional Crypto Assumptions

- Recall…

Traditional Crypto Assumptions

- Discrete Log: Given $g^x \, mod \, p$, find x.
  - (Decisional) Diffie-Hellman Assumptions $(g^x, g^y, g^{xy}), \; (g^x, g^y, g^z)$
- More: Factoring

Are They Secure?

- Algorithmic Advances:
  - Factoring: Best algorithm time $2^{\tilde{O}(n^{\frac{1}{3}})}$ to factor n-bit number.
  - Discrete log: Best algorithm $2^{\tilde{O}(n^{\frac{1}{3}})}$ for groups Z_p^*, where p is n-bit.

- Quantum Computers:
  - Shor's algorithm solves both factoring and discrete log in quantum polynomial time ($\tilde{O}(n^2)$).

Are They Secure?

“For those partners and vendors that have not yet made the transition to Suite B algorithms (ECC), we recommend not making a significant expenditure to do so at this point but instead to **prepare for the upcoming quantum resistant algorithm transition**.... Unfortunately, the growth of elliptic curve use has bumped up against the fact of continued progress in the research on quantum computing, necessitating a re-evaluation of our cryptographic strategy. ”—NSA Statement, August 2015


NIST Kicks Off Effort to Defend Encrypted Data from Quantum Computer Threat

April 28, 2016

Google Dabbles in Post-Quantum Cryptography

By Richard Adhikari
Jul 12, 2016 2:06 PM PT

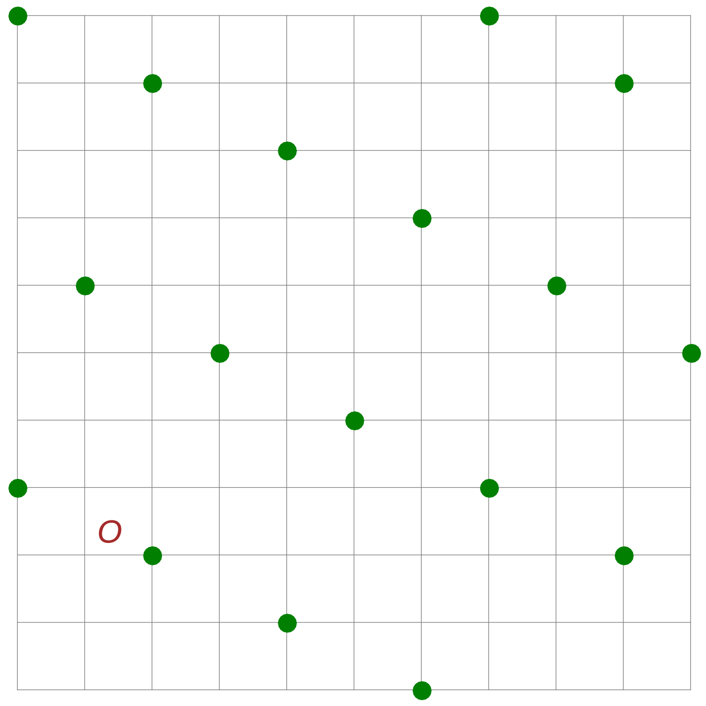
Print
Email

Post-Quantum Approach

- Believed to be hard for quantum computers.
- Versatile: Can essentially construct all cryptosystems out of these assumptions.
- Candidates: Lattice-based Crypto, Hash-based Crypto, code-based, etc.

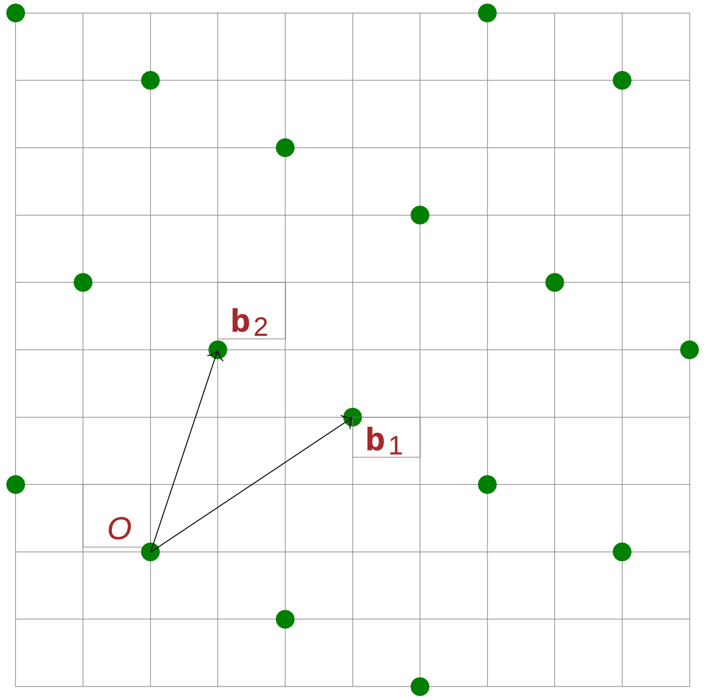
What's a Lattice?

- A periodic 'grid' in $\mathbb{Z}^m$ (Formally: full-rank additive subgroup.)



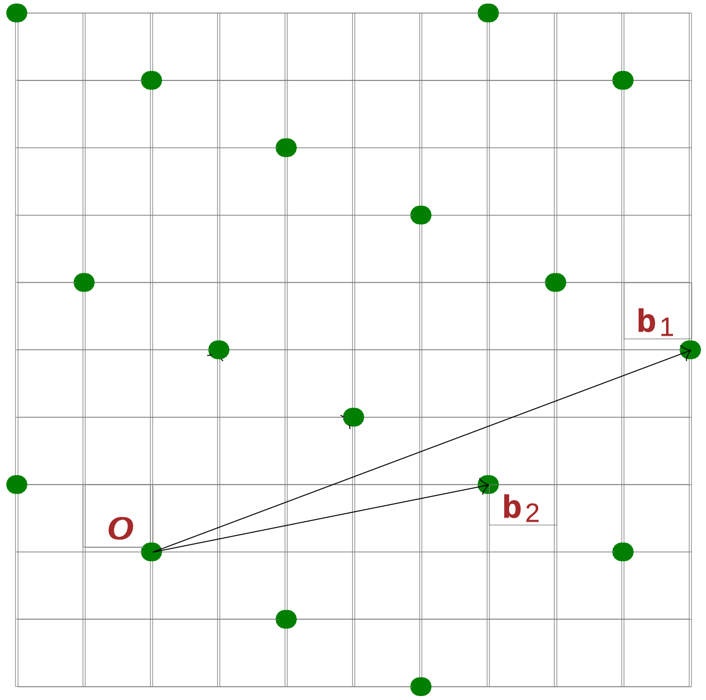
What's a Lattice?

- A periodic 'grid' in $\mathbb{Z}^m$ (Formally: full-rank additive subgroup.)
- Basis $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_\mathrm{m}\}$
- Lattice $= \sum_{j=1}^{m} \mathbb{Z} \cdot \mathbf{b}_j$



What's a Lattice?

- A periodic 'grid' in $\mathbb{Z}^m$ (Formally: full-rank additive subgroup.)
- Basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_\mathrm{m}\}$
- Lattice $= \sum_{j=1}^{m} \mathbb{Z} \cdot \mathbf{b}_\mathrm{j}$



(Other representations too . . .)

Hard Lattice Problem: Learning With Errors (LWE)

[Regev’05]

There is a secret vector s in Z_p^n (we'll use Z_{17}^4 as a running example)

An oracle (who knows s) generates a random vector a in Z_p^n and “small” noise element e in Z

The oracle outputs (a,b=<a,s>+e mod 17)

8
3
12
5

This procedure is repeated with the same s and fresh a and e

Our task is to find s

Learning With Errors (LWE) Problem

There is a secret vector s in Z_p^n (we'll use Z_{17}^4 as a running example)

An oracle (who knows s) generates a random vector a in Z_p^n and "small" noise element e in Z.

The oracle outputs (a,b=<a,s>+e mod 17)

2	13	7	3

1

This procedure is repeated with the same s and fresh a and e

Our task is to find s

Learning With Errors (LWE) Problem

There is a secret vector s in Z_p^n (we'll use Z_{17}^4 as a running example)

An oracle (who knows s) generates a random vector a in Z_p^n and "<u>small</u>" noise element e in Z.

The oracle outputs $(a, b = \langle a, s \rangle + e \bmod p)$

$$\begin{pmatrix} 2 & 13 & 7 & 3 \end{pmatrix} * \begin{pmatrix} 8 \\ 3 \\ 12 \\ 5 \end{pmatrix} + 1 = 13$$

This procedure is repeated with the same s and fresh a and e

Our task is to find s

Learning With Errors (LWE) Problem

There is a secret vector s in $\mathbb{Z}_p^n$ (we'll use Z_{17}^4 as a running example)

An oracle (who knows s) generates a random vector a in $\mathbb{Z}_p^n$ and

"<u>small</u>" noise element e in $\mathbb{Z}$.

The oracle outputs $(a, b = \langle a, s \rangle + e \ mod \ p)$

$$\begin{bmatrix} 2 & 13 & 7 & 3 \\ 4 & 7 & 9 & 1 \end{bmatrix} * \begin{bmatrix} 8 \\ 3 \\ 12 \\ 5 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 13 \\ 12 \end{bmatrix}$$

This procedure is repeated with the same s and fresh a and e

Our task is to find s

Learning With Errors (LWE) Problem

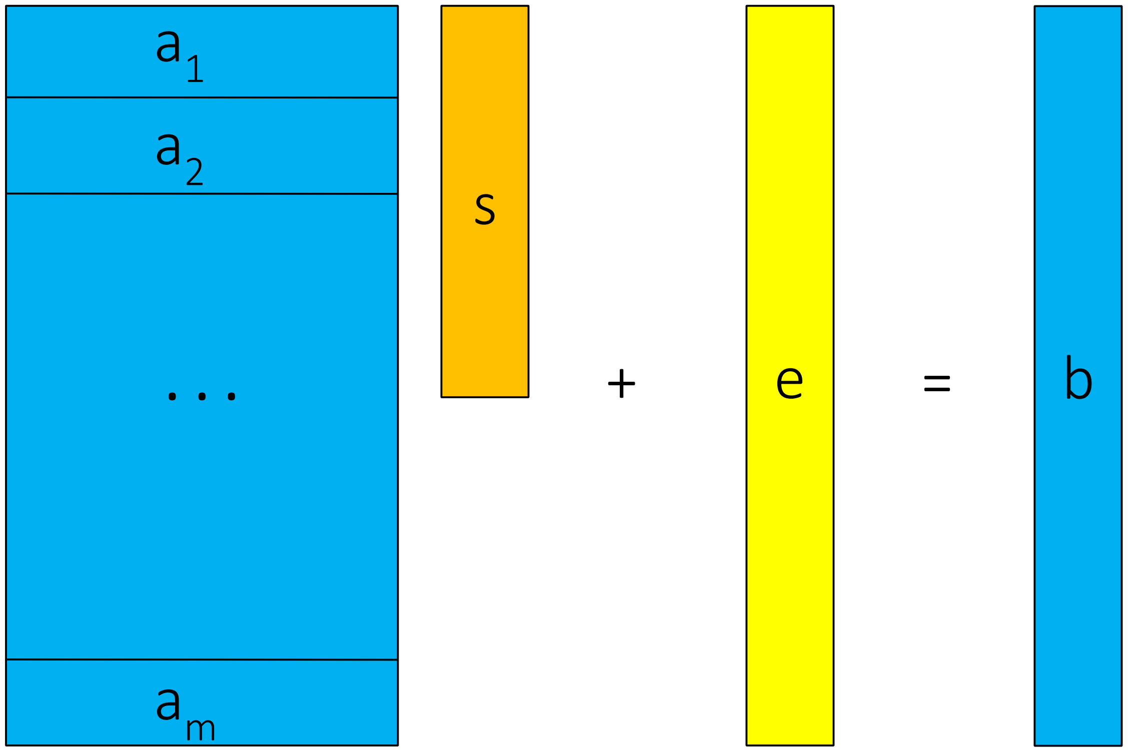
There is a secret vector s in Z_p^n (we'll use Z_{17}^4 as a running example)
An oracle (who knows s) generates a random vector a in Z_p^n and
"<u>small</u>" noise element e in Z.
The oracle outputs $(a, b = \langle a, s \rangle + e \ mod\ p)$

$$\begin{bmatrix} 2 & 13 & 7 & 3 \\ 4 & 7 & 9 & 1 \\ 6 & 14 & 5 & 11 \end{bmatrix} * \begin{bmatrix} 8 \\ 3 \\ 12 \\ 5 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 13 \\ 12 \\ 3 \end{bmatrix}$$

This procedure is repeated with the same s and fresh a and e

Our task is to find s

Learning With Errors (LWE) Problem



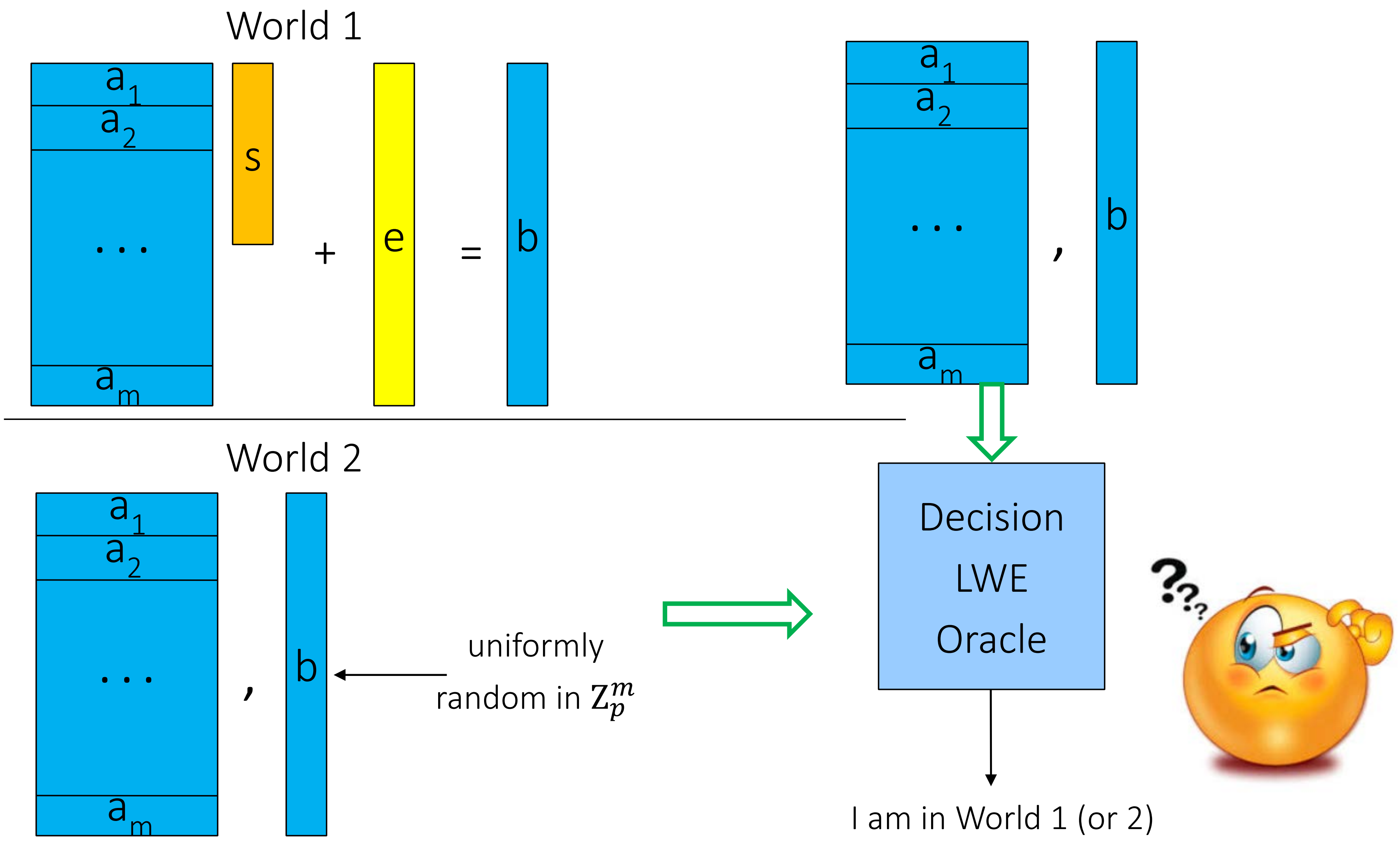
LWE Instance

$$A, \vec{b} = A\vec{s} + \vec{e} \ mod \ p$$

Once there are enough a_i , the s is uniquely determined

Theorem [Regev '05] : There is a polynomial-time quantum reduction from solving certain lattice problems in the worst-case to solving LWE.

Decisional LWE Problem



LWE is Versatile

- What kinds of crypto can we do with LWE?
  - Key Exchange, Public Key Encryption
  - Oblivious Transfer
  - Actively Secure Encryption (w/o random oracles)
  - Block Ciphers, Pseudorandom Functions
  - Identity-Based Encryption (w/ RO)
  - Hierarchical ID-Based Encryption (w/o RO)
  - Fully Homomorphic Encryption
  - Attribute-Based Encryption for arbitrary policies
  - and much, much more. .

Recall…

Diffie-Hellman key exchange

Parameters g, p

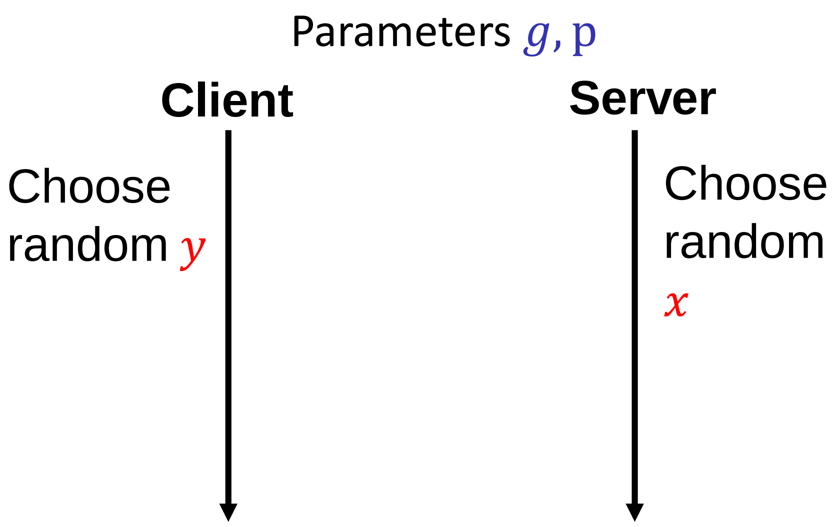
Client **Server**

Nonsecret value in blue and secret value in red.

1. Client and Server agree on the algorithm parameters g and p
2. Client and Server generate their own private keys, named y and x, respectively
3. Server computes g^x and sends it to Client.
4. Client computes g^y and sends it to Server.
5. Client computers $(g^x)^y$ and uses it as its secret.
6. Server computers $(g^y)^x$ and uses it as its secret.

Recall…

Diffie-Hellman key exchange

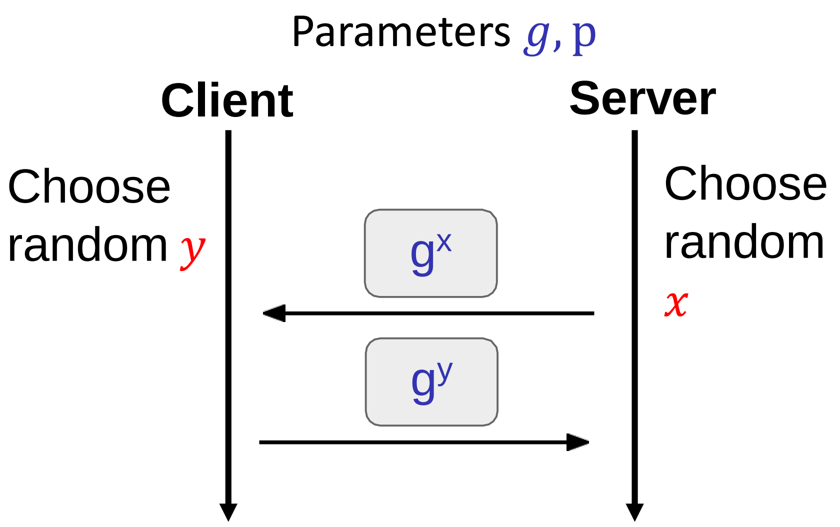


Nonsecret value in blue and secret value in red.

1. Client and Server agree on the algorithm parameters g and p
2. Client and Server generate their own private keys, named y and x, respectively
3. Server computes g^x and sends it to Client.
4. Client computes g^y and sends it to Server.
5. Client computers $(g^x)^y$ and uses it as its secret.
6. Server computers $(g^y)^x$and uses it as its secret.

Recall…

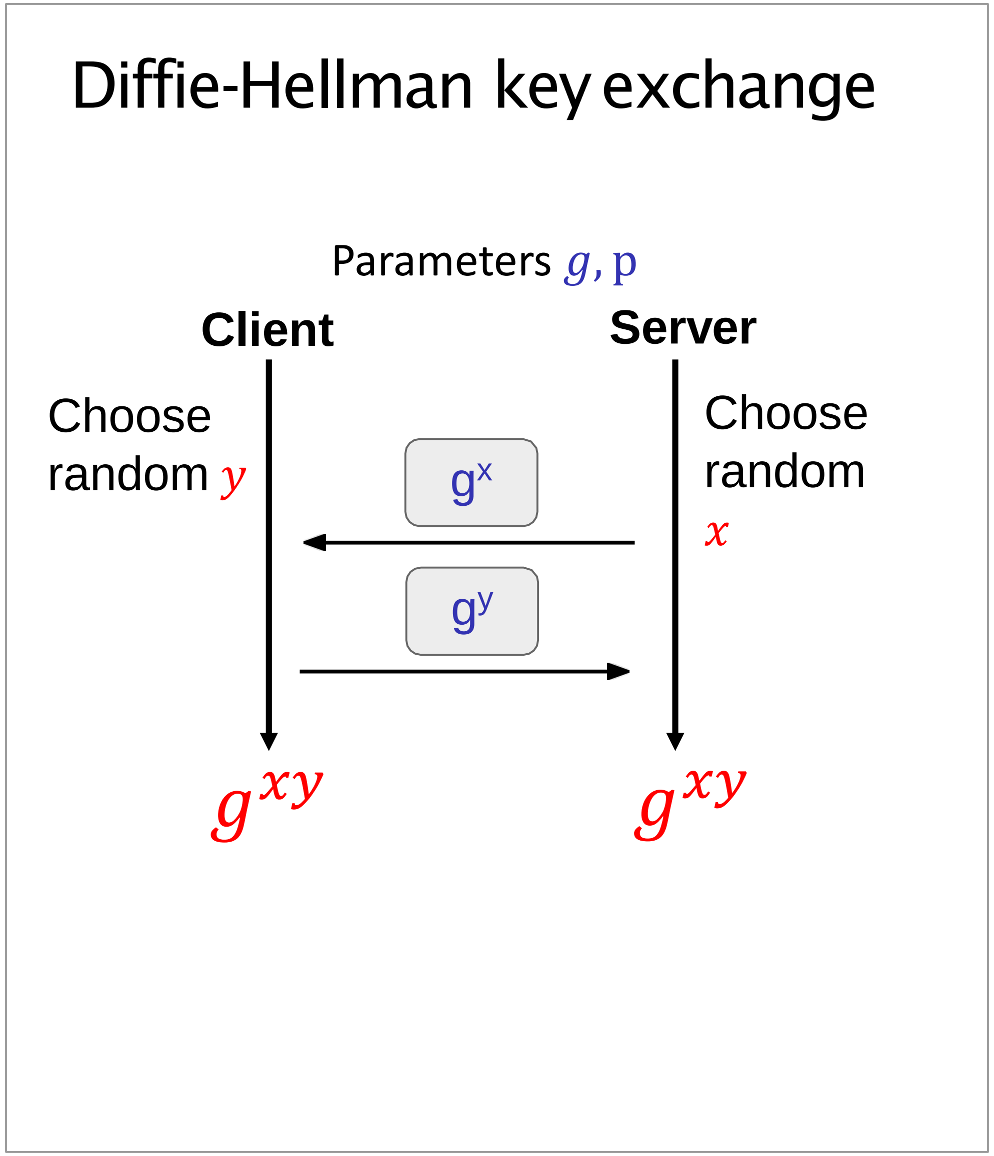
Diffie-Hellman key exchange



Nonsecret value in blue and secret value in red.

1. Client and Server agree on the algorithm parameters g and p
2. Client and Server generate their own private keys, named y and x, respectively
3. Server computes g^x and sends it to Client.
4. Client computes g^y and sends it to Server.
5. Client computers $(g^x)^y$ and uses it as its secret.
6. Server computers $(g^y)^x$ and uses it as its secret.

Recall…

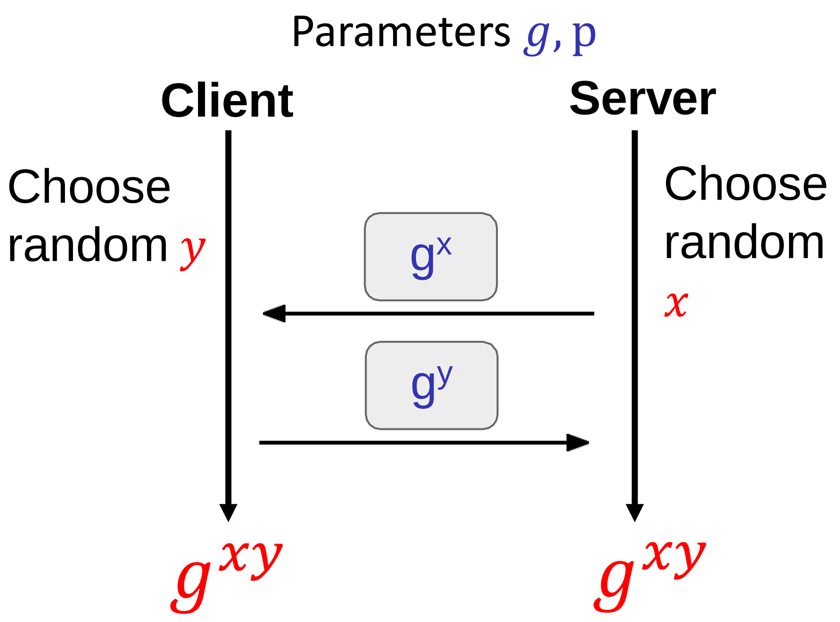


Nonsecret value in blue and secret value in red.

1. Client and Server agree on the algorithm parameters g and p
2. Client and Server generate their own private keys, named y and x, respectively
3. Server computes g^x and sends it to Client.
4. Client computes g^y and sends it to Server.
5. Client computers $(g^x)^y$ and uses it as its secret.
6. Server computers $(g^y)^x$ and uses it as its secret.

Recall…

Diffie-Hellman key exchange



Attacker sees g, g^x, g^y, but cannot sees x, y, g^{xy}.
In the view of attacker g^{xy} looks like random

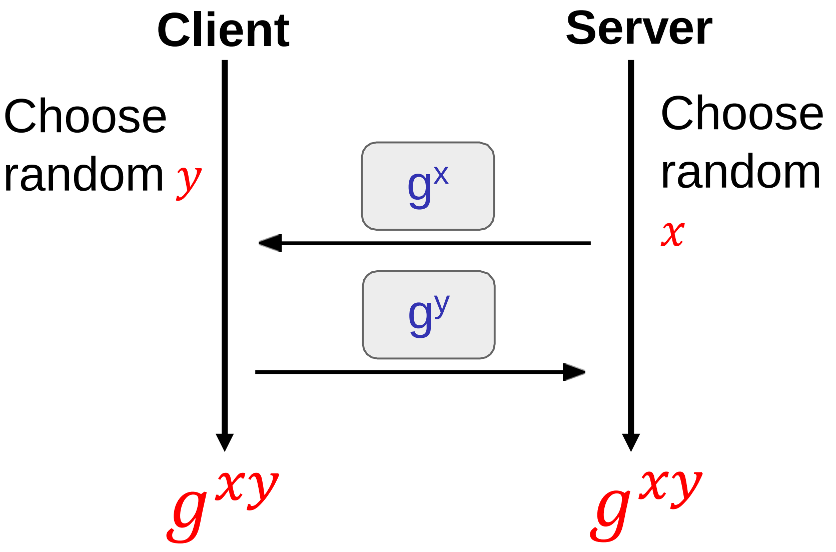
Nonsecret value in blue and secret value in red.

1. Client and Server agree on the algorithm parameters g and p
2. Client and Server generate their own private keys, named y and x, respectively
3. Server computes g^x and sends it to Client.
4. Client computes g^y and sends it to Server.
5. Client computers $(g^x)^y$ and uses it as its secret.
6. Server computers $(g^y)^x$and uses it as its secret.

DH Key Exchange Translates to LWE

Diffie-Hellman key exchange

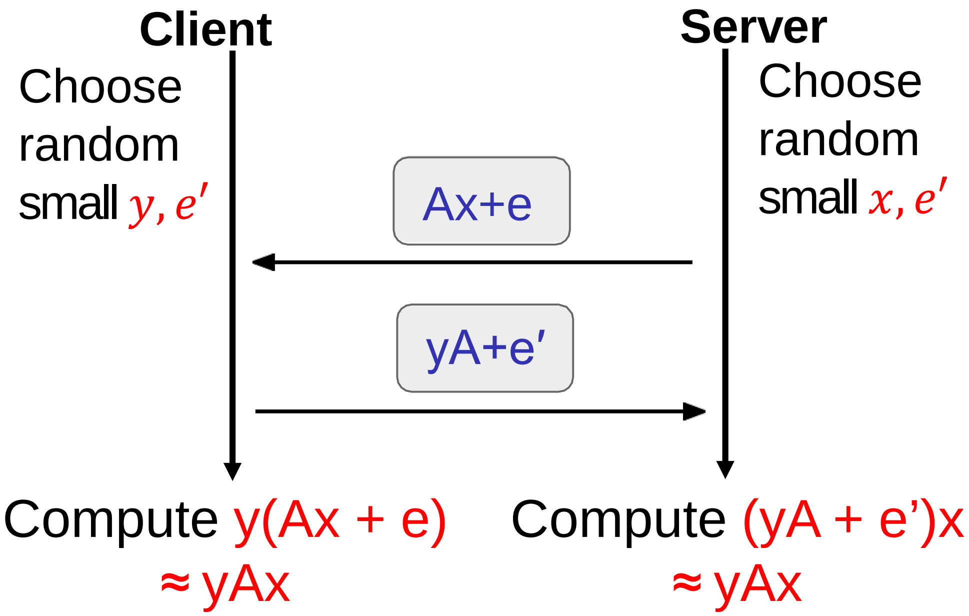
Parameters g, p



Attacker sees g, g^x, g^y,
but cannot sees x, y, g^{xy}.
In the view of attacker g^{xy} looks like random

LWE key exchange [DXL12]

Parameter A



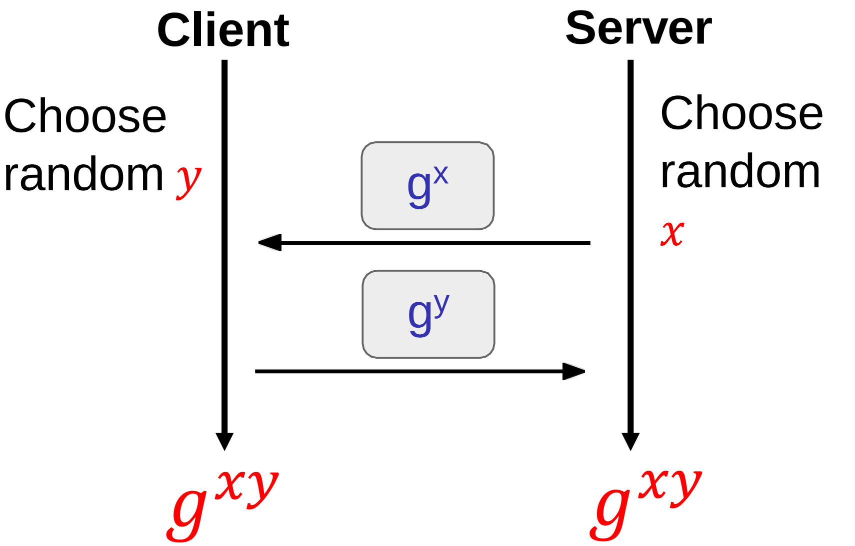
Attacker sees A, Ax+e, yA+e',
but cannot sees y, e', x, e'..
In the view of attacker KeyCon(yAx)
looks like random

[Pei14] C. Peikert. Lattice cryptography for the Internet. In Post-Quantum Cryptography. Springer, 2014
[DXL12] Ding, J., Xie, X., Lin, X. A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem. https://eprint.iacr.org/2012/688

DH Key Exchange Translates to LWE

Diffie-Hellman key exchange

Parameters g, p



Attacker sees g, g^x, g^y,
but cannot sees x, y, g^{xy}.
In the view of attacker g^{xy} looks like random

LWE key exchange [DXL12]

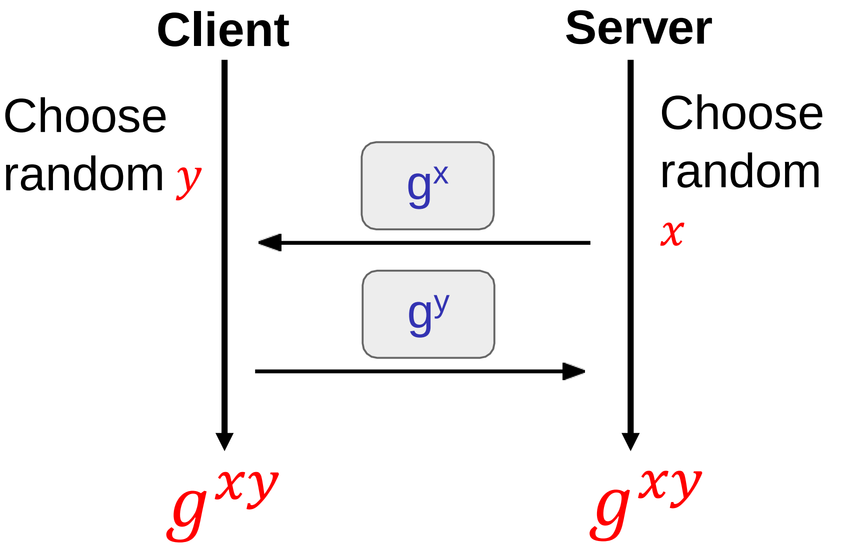
Parameter A

Client **Server**

DH Key Exchange Translates to LWE

Diffie-Hellman key exchange

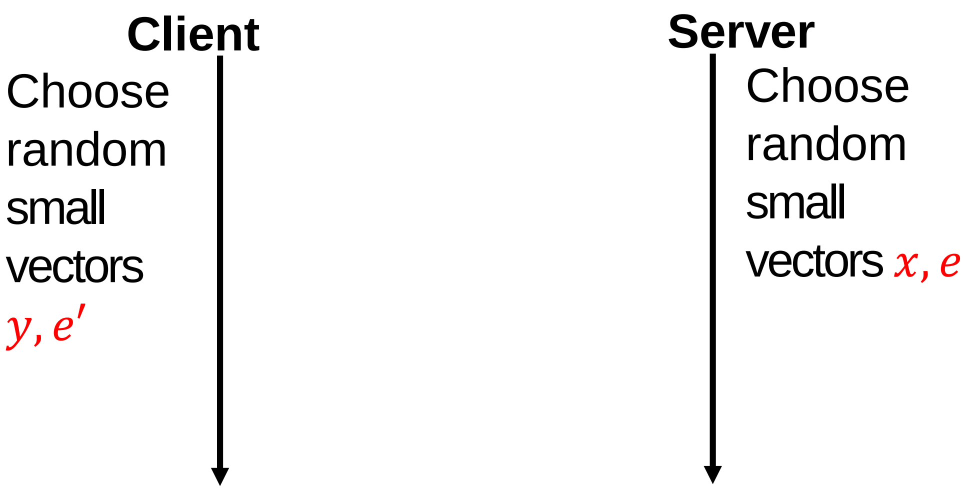
Parameters g, p



Attacker sees g, g^x, g^y,
but cannot sees x, y, g^{xy}.
In the view of attacker g^{xy} looks like random

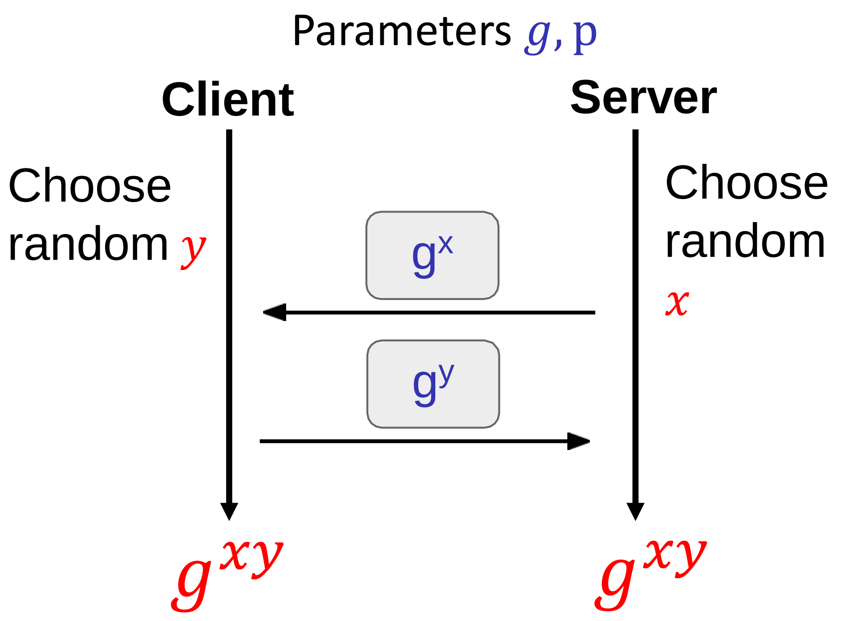
LWE key exchange [DXL12]

Parameter A



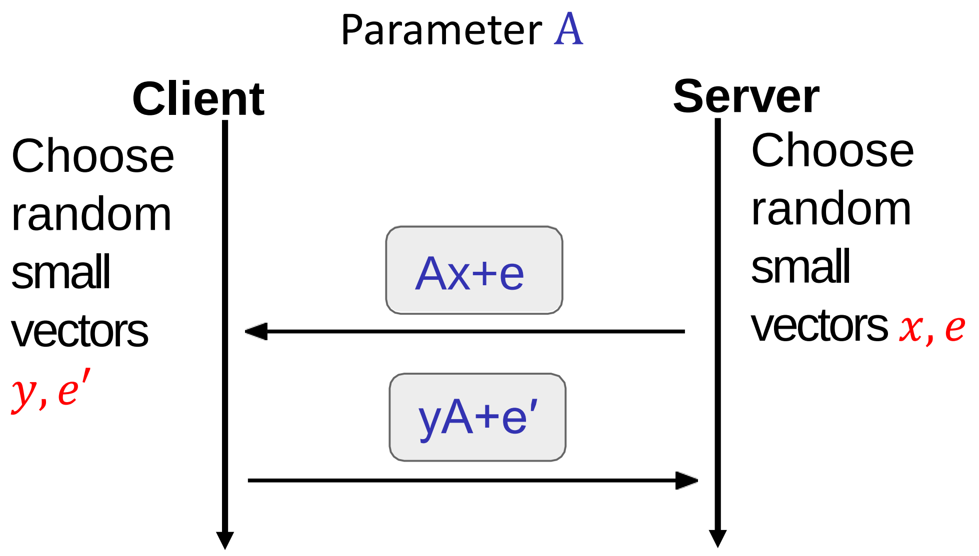
DH Key Exchange Translates to LWE

Diffie-Hellman key exchange



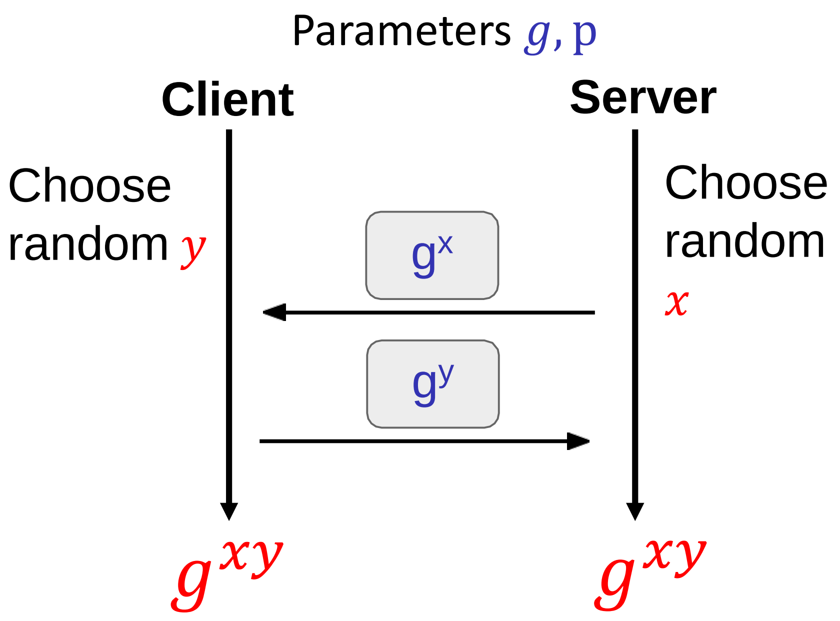
Attacker sees g, g^x, g^y,
but cannot sees x, y, g^{xy}.
In the view of attacker g^{xy} looks like random

LWE key exchange [DXL12]



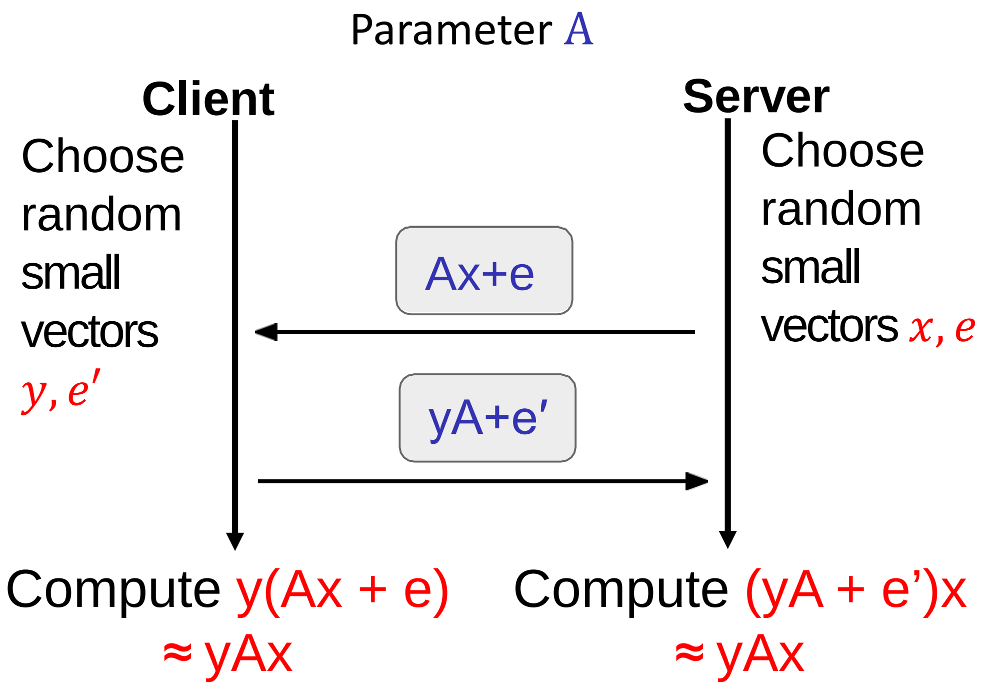
DH Key Exchange Translates to LWE

Diffie-Hellman key exchange



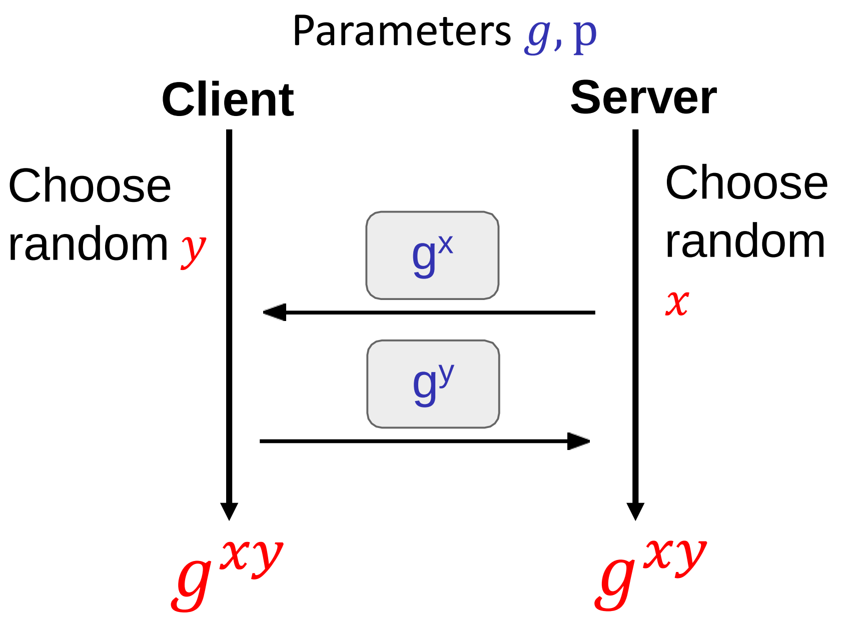
Attacker sees g, g^x, g^y,
but cannot sees x, y, g^{xy}.
In the view of attacker g^{xy} looks like random

LWE key exchange [DXL12]



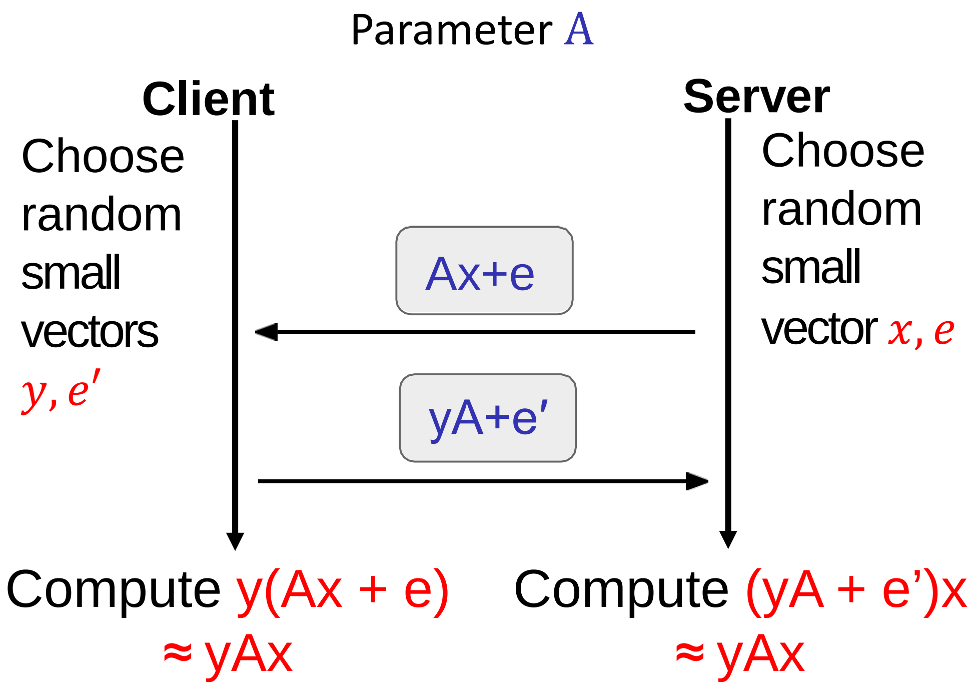
DH Key Exchange Translates to LWE

Diffie-Hellman key exchange



Attacker sees g, g^x, g^y,
but cannot sees x, y, g^{xy}.
In the view of attacker g^{xy} looks like random

LWE key exchange [DXL12]



Attacker sees A, Ax+e, yA+e′,
but cannot sees y, e', x, e'..
In the view of attacker KeyCon(yAx) looks like random

Key Exchange Implementation

- NewHope [ADPS’15]: Ring-LWE key exchange *a la* [LPR’10,P’14],
  - with many optimizations and conjectured ≥ 200-bit quantum security.
  - Comparable to or even faster than state-of-the-art ECDH w/ 128-bit (non-quantum) security.
  - Google has experimentally deployed NewHope+ECDH in Chrome canary and its own web servers.
- Frodo [BCDMNNRS’16]: Plain-LWE key exchange,
  - with many tricks and optimizations. Conjectured ≥ 128-bit quantum security.
  - About 10x slower than NewHope, but only ≈2x slower than ECDH

Next Mission – Now you are a cryptanalyst

- Our goal:

  Given a pair $(A, \vec{b})$, want to know whether it is generated as

  $A, \vec{b} = A\vec{s} + \vec{e} \ mod\ p$ OR $A, \vec{b} = random$

- Dual Attack:
  - If we can find a short vector $\vec{w}$ such that $\vec{w}\ A\ mod\ p = 0$,
  - Then compute inner product $\langle \vec{w}, \vec{b} \rangle$

Next Mission – Now you are a cryptanalyst

- Our goal:

  Given a pair $(A, \vec{b})$, want to know whether it is generated as

$A, \vec{b} = A\vec{s} + \vec{e} \; mod \; p$ OR $A, \vec{b} = random$

$\vec{w} \cdot \vec{b}$

$\vec{w} \cdot \vec{b}$

Next Mission – Now you are a cryptanalyst

- Our goal:

  Given a pair $(A, \vec{b})$, want to know whether it is generated as

$A, \vec{b} = A\vec{s} + \vec{e} \mod p$ OR $A, \vec{b} = random$

$$\vec{w} \cdot \vec{b}$$
$$= \vec{w}(A\vec{s} + \vec{e})$$

$$\vec{w} \cdot \vec{b}$$

Next Mission – Now you are a cryptanalyst

- Our goal:

  Given a pair $(A, \vec{b})$, want to know whether it is generated as

$$A, \vec{b} = A\vec{s} + \vec{e} \; mod \; p$$

OR

$$A, \vec{b} = random$$

$$\vec{w} \cdot \vec{b} = \vec{w}(A\vec{s} + \vec{e}) = (\vec{w}A)\vec{s} + \vec{w} \cdot \vec{e}$$

$$\vec{w} \cdot \vec{b}$$

Next Mission – Now you are a cryptanalyst

- Our goal:

Given a pair $(A, \vec{b})$, want to know whether it is generated as

$A, \vec{b} = A\vec{s} + \vec{e} \; mod \; p$ OR $A, \vec{b} = random$

$$\vec{w} \cdot \vec{b}$$
$$= \vec{w}(A\vec{s} + \vec{e})$$
$$= (\vec{w}A)\vec{s} + \vec{w} \cdot \vec{e}$$
$$= \vec{w} \cdot \vec{e}$$
= small

small

$$\vec{w} \cdot \vec{b}$$

Next Mission – Now you are a cryptanalyst

- Our goal:

Given a pair $(A, \vec{b})$, want to know whether it is generated as

$A, \vec{b} = A\vec{s} + \vec{e} \mod p$ OR $A, \vec{b} = random$

$$\vec{w} \cdot \vec{b}$$
$$= \vec{w}(A\vec{s} + \vec{e})$$
$$= (\vec{w}A)\vec{s} + \vec{w} \cdot \vec{e}$$
$$= \vec{w} \cdot \vec{e}$$
= small

small

$$\vec{w} \cdot \vec{b}$$
$$= random$$

Next Mission – Now you are a cryptanalyst

- Our goal:

  Given a pair $(A, \vec{b})$, want to know whether it is generated as

$$A, \vec{b} = A\vec{s} + \vec{e} \ mod\ p$$

OR

$$A, \vec{b} = random$$

$$\vec{w} \cdot \vec{b} = \vec{w}(A\vec{s} + \vec{e}) = (\vec{w}A)\vec{s} + \vec{w} \cdot \vec{e} = \vec{w} \cdot \vec{e}$$

= small

small

$$\vec{w} \cdot \vec{b} = random$$

Next Mission – Now you are a cryptanalyst

- Our goal:

  Given a pair $(A, \vec{b})$, want to know whether it is generated as

$$A, \vec{b} = A\vec{s} + \vec{e} \bmod p \qquad \text{OR} \qquad A, \vec{b} = random$$

- If we can find a short vector $\vec{w}$ such that $w\ A \bmod p = 0$,
- Then compute inner product $\langle \vec{w}, \vec{b} \rangle$
- Wait…That sounds very easy to attack, why LWE is hard?

Next Mission – Now you are a cryptanalyst

- Our goal:

  Given a pair $(A, \vec{b})$, want to know whether it is generated as

  $A, \vec{b} = A\vec{s} + \vec{e} \; mod \; p$ OR $A, \vec{b} = random$

HARD

- If we can find a <u>short</u> vector $\vec{w}$ such that $\vec{w} \, A \; mod \; p = 0$,
- Then compute inner product $\langle \vec{w}, \vec{b} \rangle$
- Wait…That sounds very easy to attack, why LWE is hard?

- For more…check

Short Integer Solution Problem!

Thank You

Questions?