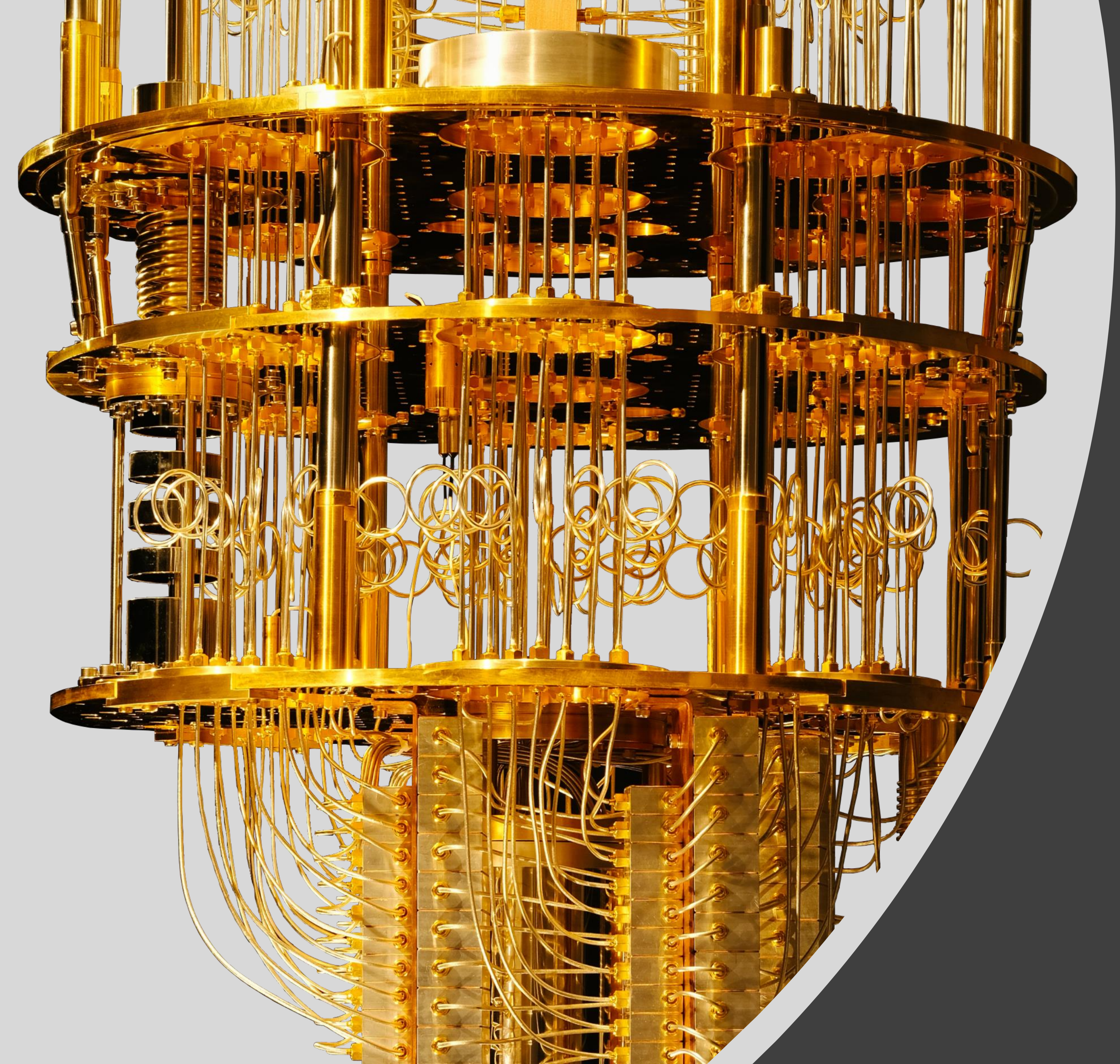


Quantum Cryptography

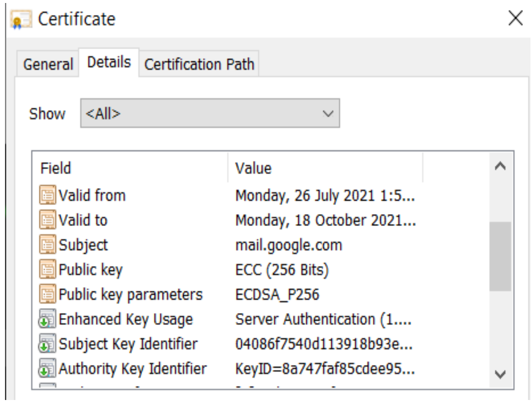
LWE- Learning With Errors

COMP3601 21T3

Cryptography vs. Quantum Computers

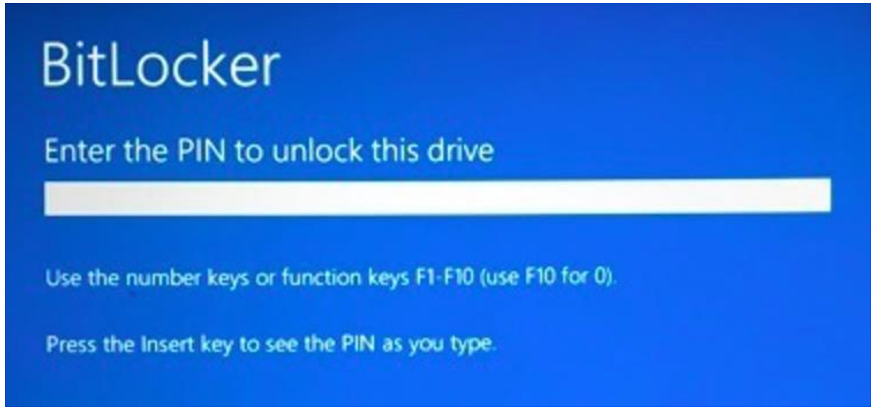
- Cryptography uses complex/ hard to reverse calculations to encrypt/ make data secure

HTTPS- gmail.com



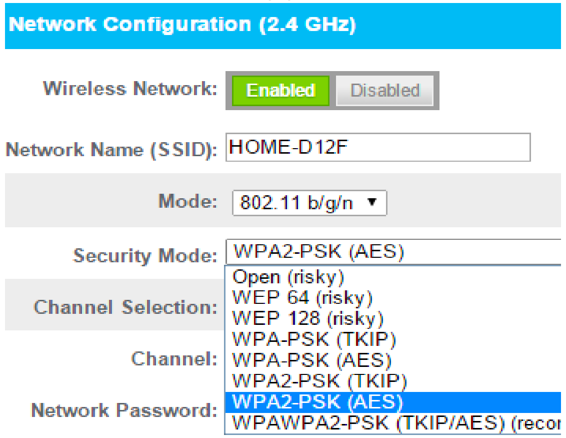
ECC- Elliptic Curve Cryptography

Disk Encryption – Microsoft BitLocker



AES- Advanced Encryption Standard

Data Encryption – WiFi



AES- Advanced Encryption Standard

- Quantum computers can process calculations simultaneously or differently than how current computers calculate
- So where do they meet

Symmetric Key Vs. Asymmetric Key Cryptography

Alice

ENCRYPTION

K_s

M = “Hello”

M_1 = !$eW&iY

Transmit over an unprotected channel

Symmetric Key cryptography

DECRYPTION

K_s

M = “Hello”

Bob

Alice

ENCRYPTION

K_{public}

M = “Hello”

M_2 = *&mzT3#

Transmit over an unprotected channel

Asymmetric (public) Key cryptography

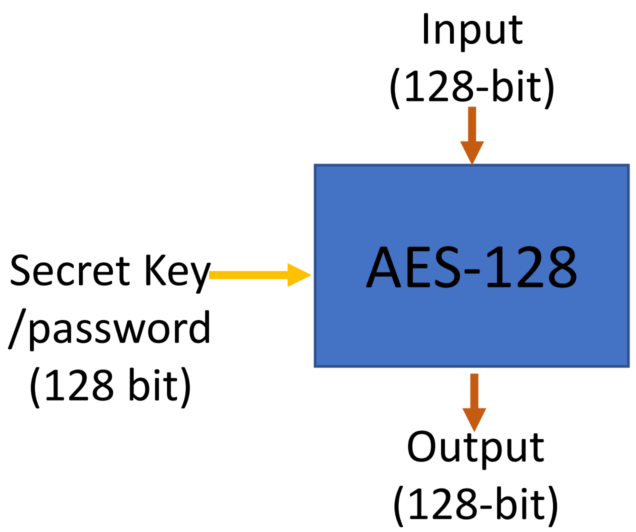
DECRYPTION

$K_{private}$

M = “Hello”

Bob

One example: break AES

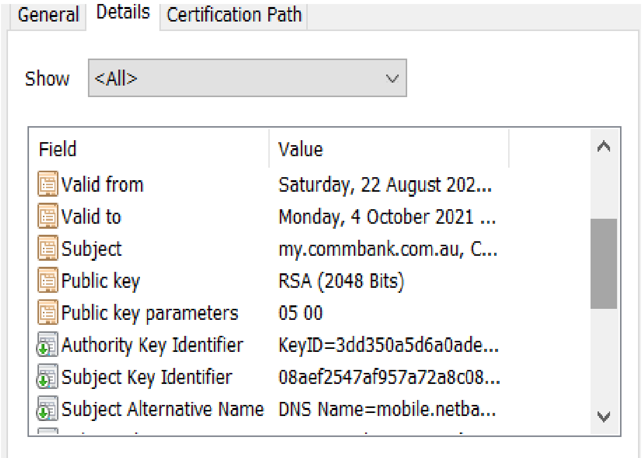


- Brute force 128-bit key = 2^{128} = 3.4028 x 10^{38}

- If you assume there are 8 billion people on Earth
- Each person has 10 devices (computers, mobile phones, gaming consoles, smart watches, …)
- Each device can test 1 billion keys per second
- After trying 50% of possibilities, we will find the correct key
- Whole world can crack a single 128-bit in 67.439 billion years
- Age of the Earth = 4.543 billion years
- Age of the universe = 13.77 billion years

Quantum computers may crack AES-128 in 6 months

Break RSA - 1024

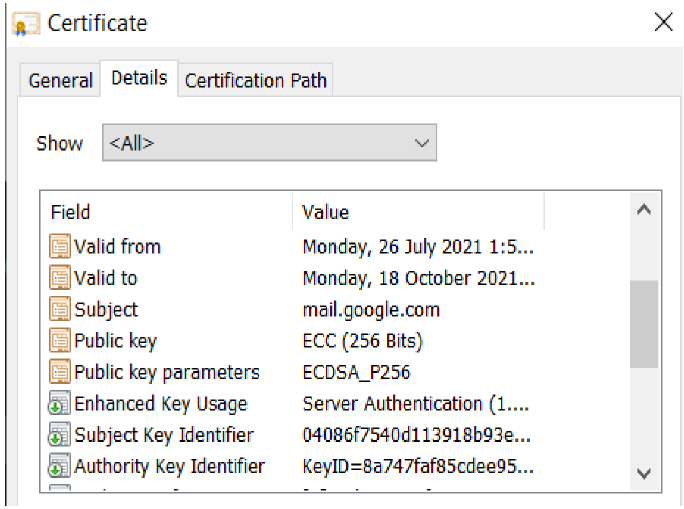


- RSA depends on prime numbers [link]
- Brute force RSA 1024 needs 1.88×10^{302} calculations

copied from [link]

- AES-128 brute force = 3.4028×10^{38}

HTTPS- gmail.com



Quantum computers can crack RSA 1024 in 3.58 hours [1]

RSA 2048 in 28.6 hours [1]

Quantum computers can crack ECC 256 in 10.5 hours [1]

[1] https://www.nap.edu/read/25196/chapter/6#98

Current Cryptographic Algorithms vs. Quantum Computers

Symmetric Key Cryptography

AES-256

HMAC SHA256

Considered as Quantum Secure – large key sizes

Grover's algorithm showed the brute force attack time to its square root. AES-128 the attack time becomes reduced to 2^{64} (not highly secure)

Asymmetric Key Cryptography

RSA

Depends on factoring numbers

ECC

Depends on EC discrete logarithms

Can be broken quickly using a Quantum Computer

Quantum Secure Algorithms – Postquantum cryptography

National Institute of
Standards and Technology
U.S. Department of Commerce

- National Institute of Standard and Testing (NIST) started finding quantum secure algorithms
- https://csrc.nist.gov/projects/post-quantum-cryptography
- Started in 2016
- Currently in round 3
- Expected to publish draft standards in 2023–2025

Image: NIST

LWE- Learning with Errors

- Proposed in 2005 by Regev [link]
- Considered as quantum secure
- New implementations/ proposals
- We often project LWE as a Lattice Problem [link]
- We will use a simple implementation for your project

Example – given blue matrices, can you deduce red matrix (column vector)?

$$\underbrace{\begin{bmatrix} 3 & 5 & 8 & 17 & 1 \\ 2 & 8 & 6 & 9 & 13 \\ 16 & 15 & 11 & 8 & 5 \\ 14 & 6 & 18 & 13 & 10 \end{bmatrix}}_{Z_{19}^{4X5}} \times \underbrace{\begin{bmatrix} \; \\ \; \\ \; \\ \; \\ \; \end{bmatrix}}_{Z_{19}^{5X1}} = \underbrace{\begin{bmatrix} 9 \\ 1 \\ 13 \\ 16 \end{bmatrix}}_{Z_{19}^{4X1}}$$

Example -

$$\mathbb{Z}_{19}^{4X5} \qquad \mathbb{Z}_{19}^{5X1} \qquad \mathbb{Z}_{19}^{4X1}$$

$$\begin{bmatrix} 3 & 5 & 8 & 17 & 1 \\ 2 & 8 & 6 & 9 & 13 \\ 16 & 15 & 11 & 8 & 5 \\ 14 & 6 & 18 & 13 & 10 \end{bmatrix} \mathbf{X} \begin{bmatrix} 16 \\ 9 \\ 1 \\ 5 \\ 13 \end{bmatrix} = \begin{bmatrix} 9 \\ 1 \\ 13 \\ 16 \end{bmatrix}$$

Gaussian Elimination

https://web.maths.unsw.edu.au/~potapov/1141_2013/chapter-3.pdf

Example – add errors

This brings us Search LWE problem:
Given blue matrices find red matrix

Noise/ Error

A= $\mathbb{Z}_{19}^{4X5}$ × S= $\mathbb{Z}_{19}^{5X1}$ + e= $\mathbb{Z}_{19}^{4X1}$ = B= $\mathbb{Z}_{19}^{4X1}$

$$\begin{bmatrix} 3 & 5 & 8 & 17 & 1 \\ 2 & 8 & 6 & 9 & 13 \\ 16 & 15 & 11 & 8 & 5 \\ 14 & 6 & 18 & 13 & 10 \end{bmatrix} \times \begin{bmatrix} 16 \\ 9 \\ 1 \\ 5 \\ 13 \end{bmatrix} = \begin{bmatrix} 3x16 + 5x9 + 8x1 + 17x5 + 1x13 = 199 \bmod 19 = 9 \\ 2x16 + 8x9 + 6x1 + 9x5 + 13x13 = 324 \bmod 19 = 1 \\ 16x16 + 15x9 + 11x1 + 8x5 + 5x13 = 507 \bmod 19 = 13 \\ 14x16 + 6x9 + 18x1 + 13x5 + 10x13 = 491 \bmod 19 = 16 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 10 \\ 1 \\ 12 \\ 16 \end{bmatrix}$$

Considered as Quantum safe/ hard

Known algorithms require $2^{O(n)}$ time

LWE Search Problem

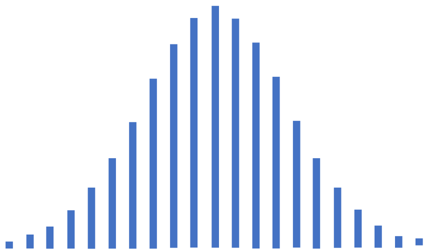
- Pick a vector A $\in \mathbb{Z}_q^n$ from the uniform distribution over A $\in \mathbb{Z}_q^n$,

  q is poly(n) – we choose a prime number

- Pick e from the distribution ϕ

  We assume distribution ϕ is Gaussian, mean = 0

  $\sqrt{n} \leq$ Std. deviation $\ll$ q, ‘rate’- $\alpha \in \mathbb{R}$ (only cover alpha fraction of $\mathbb{Z}_q^n$)



- Evaluate B = $\langle A, S \rangle$/q + e,
- Output pairs (A_i, B_i) ; i=1,…,n

LWE Search problem: Find S $\in \mathbb{Z}_q^n$, given many (A_i, B_i)

$$B \approx \langle A, S \rangle$$

LWE Decision Problem

- Given a set of (A, B) parameters/ pairs, can you guess uniform random pairs over $\mathbb{Z}_q$ vs. pairs generated by B = $\langle A,S \rangle$/q + e?

16	15	11	8	5	12	Random?
5	1	9	3	13	2	Random?
3	10	6	7	17	8	Random?
14	6	18	13	10	16	Random?

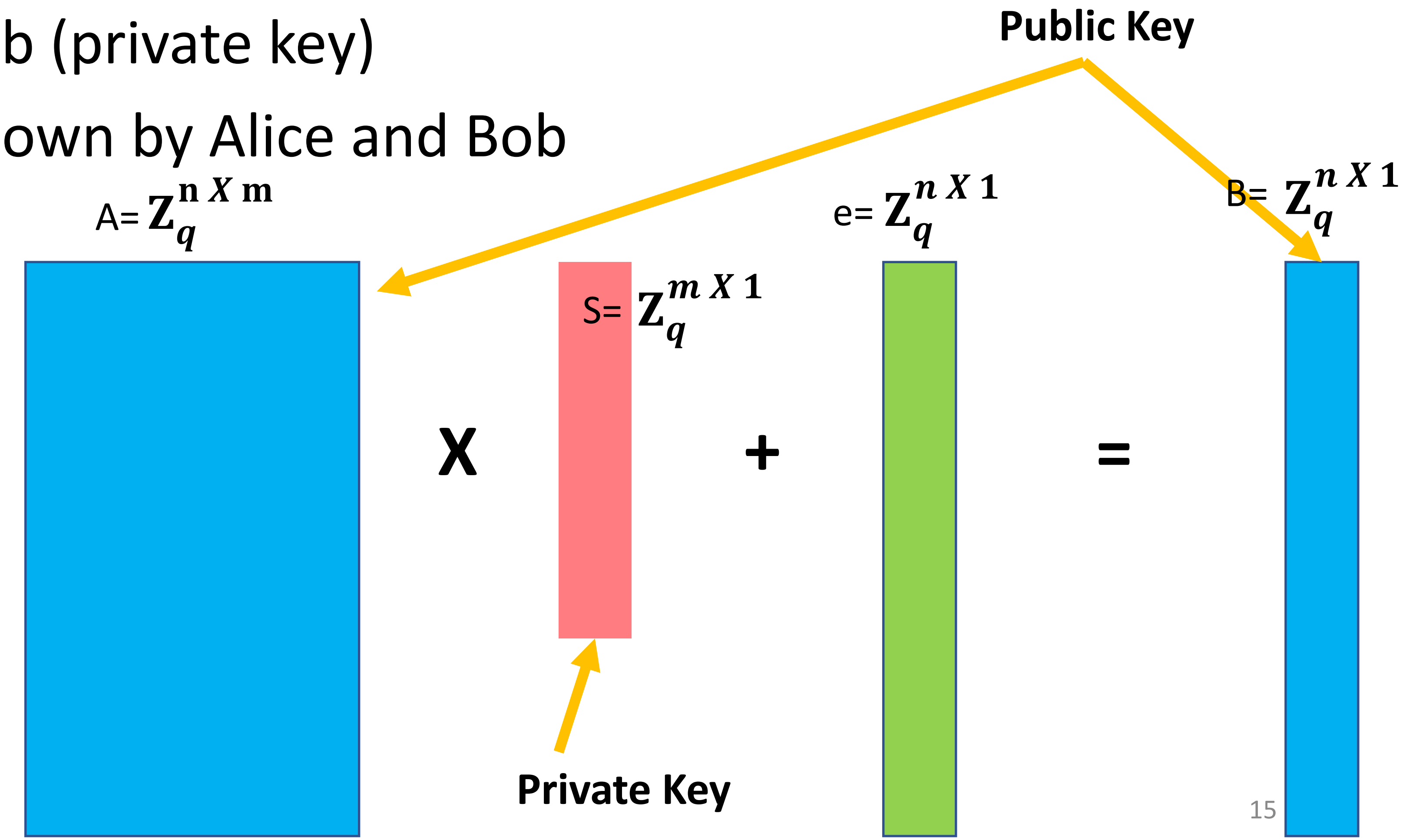
$$\begin{bmatrix} 3 & 5 & 8 & 17 & 1 \\ 2 & 8 & 6 & 9 & 13 \\ 16 & 15 & 11 & 8 & 5 \\ 14 & 6 & 18 & 13 & 10 \end{bmatrix} \times \begin{bmatrix} 16 \\ 9 \\ 1 \\ 5 \\ 13 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 10 \\ 1 \\ 12 \\ 16 \end{bmatrix}$$

LWE Public Cryptography

- How to use LWE problem to implement a simple public key crypto algorithm
- 3 Components
  - Key generation
  - Encryption
  - Decryption
- Used to encrypt 1-bit(1,0), for L-bits repeat encryption L times
- There may be errors in decryption[one of the tasks you evaluate]

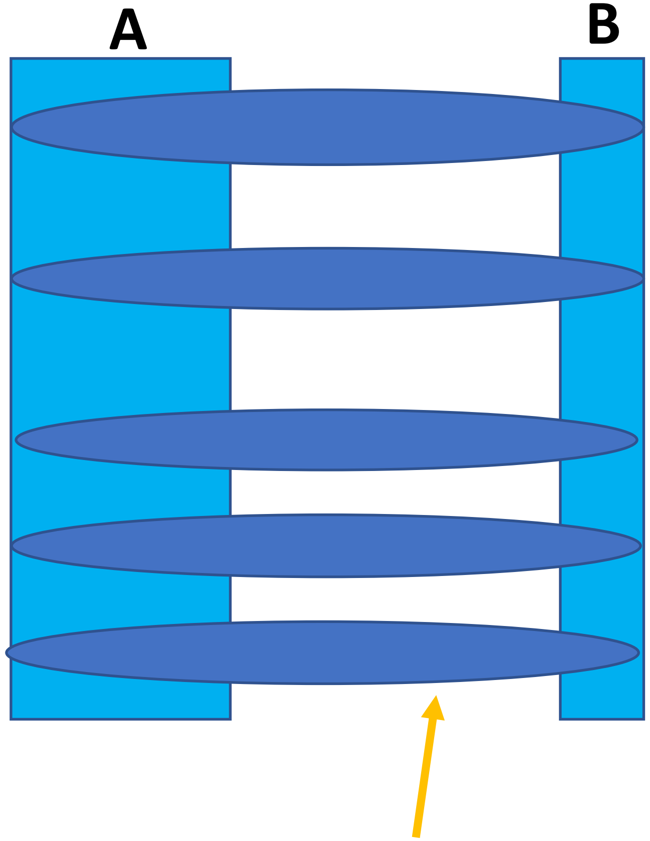
Key Generation

- Bob generates public keys (A, B) and sends to Alice
- S is only known by Bob (private key)
- q is assumed to be known by Alice and Bob



Encryption/ Encode

- Alice generates the message (let's say one bit) M={0,1}
- Generates u and v



Randomly sample; typical value $\approx$ n/4

$$\mathbf{u} = \left(\sum \mathbf{A}_{\text{sample}}\right) \bmod \mathbf{q}$$

$$\mathbf{v} = \left(\sum \mathbf{B}_{\text{sample}} - \mathbf{M}.\frac{q}{2}\right) \bmod \mathbf{q}$$

If S is a scalar
u is a scalar
If S is a column vector
u is a row vector

Encrypted/ encoded values are (u, v)

Decryption/ Decode

- Bob receives u and v
- Bob calculates **D**, **D = (v – u.s) mod q**

If **D** is less than **q/2**, the **message (M)** is **0**. If **D** is greater than **q/2**, the **message(M)** is **1**.

If **D** is between -**q/4 and q/4**, the **message (M)** is **0**.
else the **message(M)** is **1**.

Example

- Lets assume n=12, m=4, q=23, S={4 7 5 5}

A					S						e		B
18	4	16	20		4		280		4		-1		3
8	11	12	12		7		229		22		2		1
1	11	17	9		5		211		4		0		4
13	22	22	12		5		376		8		0		8
20	19	22	16				403		12		-2		10
4	22	12	0	X		=	230	mod q	0	+	2	=	2
9	15	22	18				341		19		0		19
17	9	2	3				156		18		2		20
0	21	1	11				207		0		1		1
21	11	7	5				221		14		-1		13
17	5	13	8				208		1		2		3
12	9	12	15				246		16		1		17

Example

- Lets assume n=12, m=4, q=23
- M = 1
- A, B and q known

A

18	4	16	20
8	11	12	12
1	11	17	9
13	22	22	12
20	19	22	16
4	22	12	0
9	15	22	18
17	9	2	3
0	21	1	11
21	11	7	5
17	5	13	8
12	9	3	15

B

3
1
4
8
10
2
19
20
1
13
3
17

u =($\sum \mathbf{A_{sample}}$ mod q)

= [8 11 12 12] + [20 19 22 16] + [9 15 22 18] + [17 5 13 8]

= [54 50 69 54] mod 23

= [8 4 0 8]

v = ($\sum \mathbf{B_{sample}} - \mathbf{M}.\frac{q}{2}$) mod q

= (1 + 10 + 19 + 3) – 1x11.5

= (33) – 1x11.0 $\leftarrow$ floor

= 22 mod 23

= 22

Example

- Lets assume n=12, q=23, s={4 7 5 5}
- M=?

Bob receives (u,v)

u = [8 4 0 8]

v = 22

D = (v – u.s) mod q

$$= \left(22 - \mathbf{[8\ 4\ 0\ 8]} * \begin{matrix} 4 \\ 7 \\ 5 \\ 5 \end{matrix}\right) \bmod 23$$

(-n) mod k = k – (n mod k)

= (22 – 100) mod 23

= -78 mod 23 = 23 – (78 mod 23)

= 14

D is greater than q/2, the message (M) is 1

$$\mathbf{v} = \sum \mathbf{B_{sample}} - \mathbf{M}.\frac{q}{2}$$

= (1 + 10 + 19 + 3) – 0x11.5

= (33) – 0x11.0

= 33 mod 23

= 10

Assume M=0

u = [8 4 0 8]

v = 10

D = (v – u.s) mod q

$$= \left(10 - \mathbf{[8\ 4\ 0\ 8]} * \begin{matrix} 4 \\ 7 \\ 5 \\ 5 \end{matrix}\right) \bmod 23$$

= (10 -100) mod 23

= 2

D is less than q/2, the message (M) is 0

When there is no error (e=0)

18	4	16	20		4		280			4
8	11	12	12		7		229			22
1	11	17	9		5		211			4
13	22	22	12		5		376			8
20	19	22	16				403			12
4	22	12	0	X		=	230		mod q	0
9	15	22	18				341			19
17	9	2	3				156			18
0	21	1	11				207			0
21	11	7	5				221			14
17	5	13	8				208			1
12	9	12	15				246			16

Example

- Lets assume n=12, m=4, q=23
- M = 1

A

18	4	16	20
8	11	12	12
1	11	17	9
13	22	22	12
20	19	22	16
4	22	12	0
9	15	22	18
17	9	2	3
0	21	1	11
21	11	7	5
17	5	13	8
12	9	3	15

B

4
22
4
8
12
0
19
18
0
14
1
16

u = ($\sum A_{\text{sample}}$) mod q

= [8 11 12 12] + [20 19 22 16] + [9 15 22 18] + [17 5 13 8]

= [54 50 69 54] mod 23

= [8 4 0 8]

u = [8 4 0 8]

If there is no error, D is either q/2 or 0;

B
4
22
4
8
12
0
19
18
0
14
1
16

u = [8 4 0 8]

v = $(\sum \mathbf{B}_{\mathbf{sample}} - \boldsymbol{M}.\frac{q}{2})$ **mod q**

= (22 + 12 + 19 + 1) – 1x11.5

= (54) – 1x11.0

= 43 mod 23

v = 20

D = (v – u.s) mod q

= (20 – **[8 4 0 8]** * $\begin{matrix} 4 \\ 7 \\ 5 \\ 5 \end{matrix}$) mod 23

= (20 – 100) mod 23

= 12

D is q/2

(difference due to rounding)

v = $(\sum \mathbf{B}_{\mathbf{sample}} - \boldsymbol{M}.\frac{q}{2})$ **mod q**

= (22 + 12 + 19 + 1) – 0x11.5

= (54) – 0x11.0

= 54 mod 23

v = 8

D = (v – u.s) mod q

= (8 – **[8 4 0 8]** * $\begin{matrix} 4 \\ 7 \\ 5 \\ 5 \end{matrix}$) mod 23

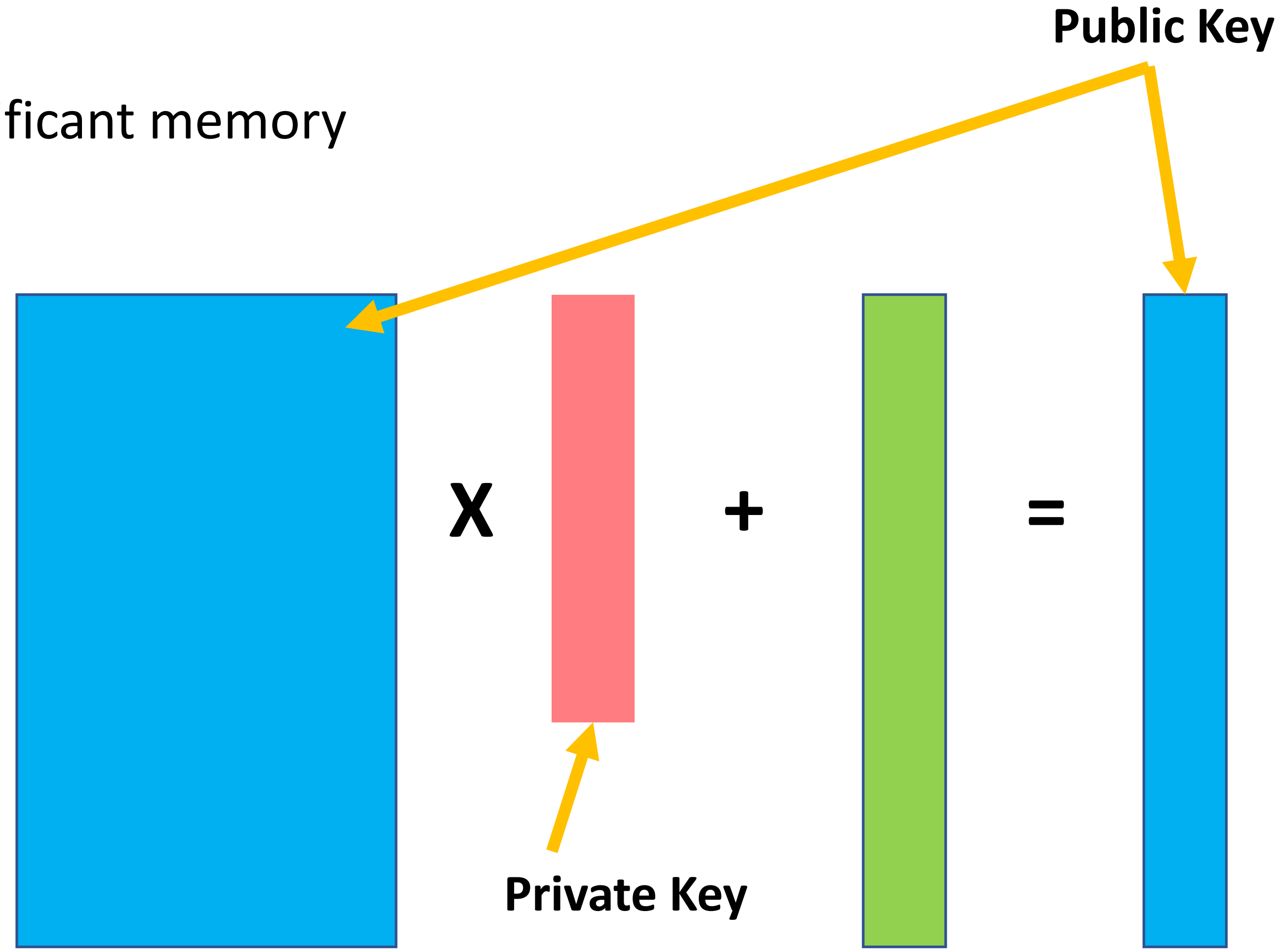
= (8 – 100) mod 23

= 0

D is 0

Extra – Ring LWE & Homomorphic Computing

- Crypto systems often use Ring LWE
  - Why? inefficient!!!
  - You need to store A which consumes significant memory
  and matrix multiplications
  $m=2048$, $n=64$, $q = 65099$
  $Size_A = 2048 \times 64 \times 16bit = 256KB$
- What if we use a Polynomial X
- Multiplication in a polynomial ring,
  $Z_q[X]/(X^n+1)$
- We only need to send X,
  can use FFT to speed up
- LWE 200-400KB public key
  Ring LWE 1-2KB public key



Extra – Ring LWE & Homomorphic Computing

- Can we perform operations on encrypted data?
- Typically, $ENC(X_1) + ENC(X_2) \neq ENC(X_1+X_2)$
- What is the advantage?
- Homographic computing involves performing operations

(+,-,x,/,…) on encrypted data

- Not easy as you may think, ENC makes data nonlinear, applies avalanche effects.
- We can expand LWE problem into homomorphic computing
- Fun to explore LWE problems → Final year thesis