# Comparative study of Post-Quantum Key-Exchange Mechanisms and its implementations

Aditya Bhardwaj

# Contents

## Key exchange mechanism

Sharing a secret between two parties without revealing the secret to any third parties

- Two most widespread algorithms used
  **RSA** and **Diffie-Hellman**

- Elliptic Curve Cryptography -

  ECDH → Elliptic Curve Diffie-Hellman
  ECDHE → (Elliptic Curve Diffie-Hellman Ephemeral)

## The Quantum Threat

Difficult problems

- Integer Factorization, Discrete Logarithm
- Shor algorithm → 1994

  Factorization in polynomial time
- Grover's search algorithm → 1996

  Search for an element in $\sqrt{N}$ steps for total N elements

## Post-Quantum Cryptography

Quantum-safe mathematical techniques

- Lattices
- Error correcting codes
- Multivariate equations
- Supersingular elliptic curve isogenies

NIST initiated a process to evaluate and standardize one or more quantum-resistant public-key cryptographic algorithms

## Post-Quantum Cryptography Standardization

Quantum-safe mathematical techniques

- PQCrypto 2016, Deadline - end of 2017
- 23 signature schemes and 59 PKE/KEM schemes

Finalists

- **Lattice-based**: CRYSTALS-Kyber, NTRU Prime, FrodoKEM

- **Code-based**: BIKE, Classic McEliece, HQC

# PQC Standardization - NIST Security Levels

- Each algorithm has different parameters
- Different security levels

| Levels | Definition, as least as hard to break as ... |
|---|---|
| 1 | To recover the key of AES-128 by exhaustive search |
| 2 | To find a collision in SHA-256 by exhaustive search |
| 3 | To recover the key of AES-192 by exhaustive search |
| 4 | To find a collision in SHA-384 by exhaustive search |
| 5 | To recover the key of AES-256 by exhaustive search |

# Post-Quantum Cryptography Standardization

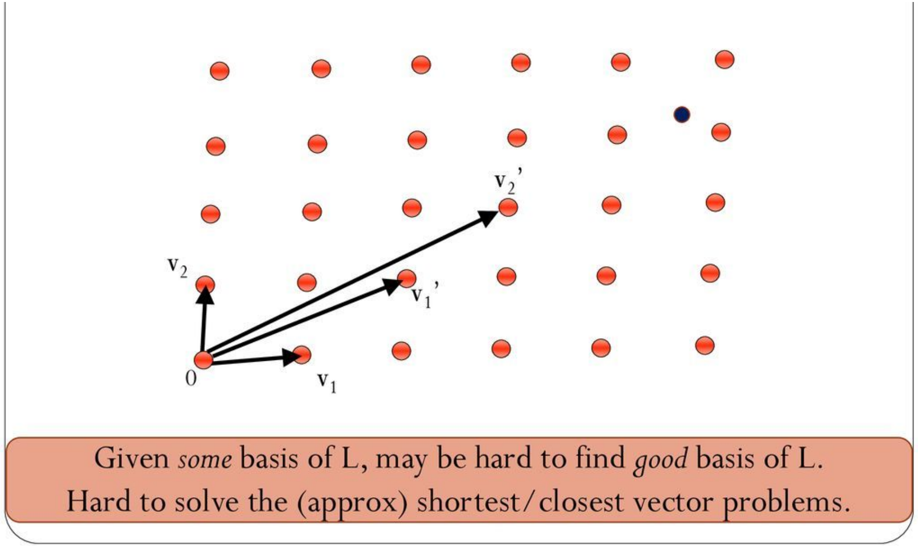
Quantum-safe mathematical techniques

- PQCrypto 2016, Deadline - end of 2017
- 23 signature schemes and 59 PKE/KEM schemes

Finalists

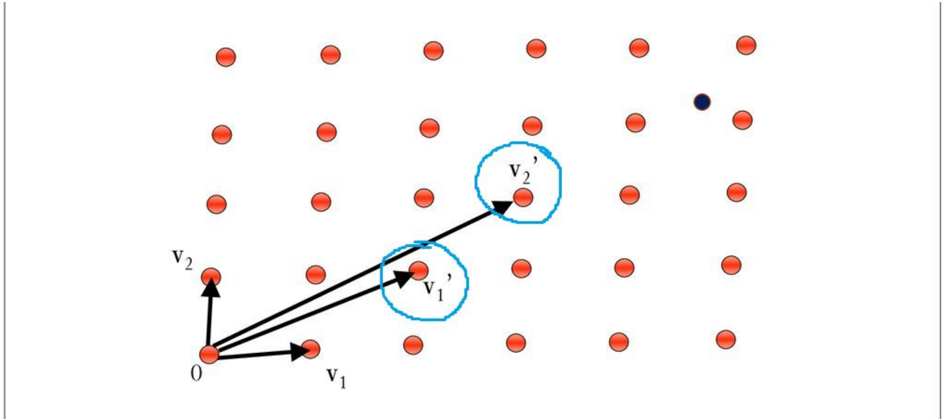
- **Lattice-based**: CRYSTALS-Kyber, NTRU Prime, FrodoKEM
- Code-based: BIKE, Classic McEliece, HQC

# Lattice-based Cryptography - Lattices




- **Infinite set of points** generated by addition,subtraction of vectors
- Same lattice can be generated by different **'basis'**

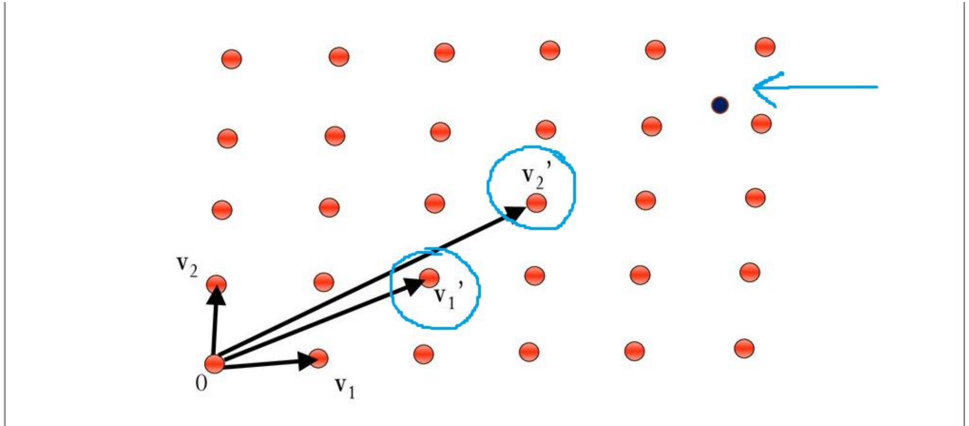
# Lattice-based Cryptography - Lattices




Given *some* basis of L, may be hard to find *good* basis of L.
Hard to solve the (approx) shortest/closest vector problems.

- Some problems are hard... given a bad basis

# Lattice-based Cryptography - Lattices




Given *some* basis of L, may be hard to find *good* basis of L.
Hard to solve the (approx) shortest/closest vector problems.

- Some problems are hard… given a bad basis
- But easy with good basis

# Lattice-based Cryptography - Lattices

$$3x - 2y + 1z = 5$$

$$4x + 3y - 3z = 1$$

$$-6x + 4y - 7z = 3$$

$$\begin{array}{c} \begin{array}{cccc} x & y & z & c \\ \downarrow & \downarrow & \downarrow & \downarrow \end{array} \\ \left[\begin{array}{ccc|c} 3 & -2 & 1 & 5 \\ 4 & 3 & -3 & 1 \\ -6 & 4 & -7 & 3 \end{array}\right] \end{array}$$

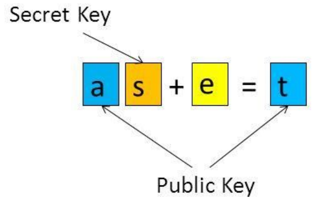
$A * s = t \Rightarrow$ Gaussian elimination

$A * s + \underline{e} = t \Rightarrow$ Reduction to lattice hard problem

# Learning with errors problem

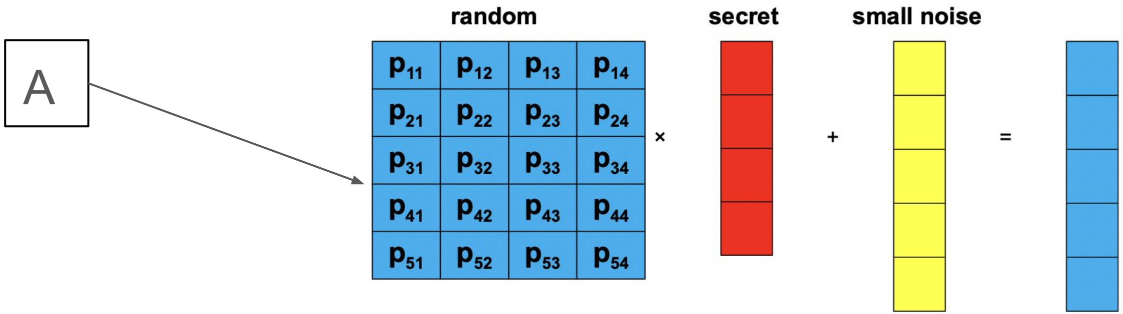
$$A = \overset{\substack{\textbf{random} \\ \mathbb{Z}_{13}^{7\times 4}}}{\begin{bmatrix} 4 & 1 & 11 & 10 \\ 5 & 5 & 9 & 5 \\ 3 & 9 & 0 & 10 \\ 1 & 3 & 3 & 2 \\ 12 & 7 & 3 & 4 \\ 6 & 5 & 11 & 4 \\ 3 & 3 & 5 & 0 \end{bmatrix}} \times \overset{\substack{\textbf{secret} \\ \mathbb{Z}_{13}^{4\times 1}}}{\begin{bmatrix} \ \\ \ \\ \ \\ \ \end{bmatrix}} + \overset{\substack{\textbf{small noise} \\ \mathbb{Z}_{13}^{7\times 1}}}{\begin{bmatrix} \ \\ \ \\ \ \\ \ \\ \ \\ \ \\ \ \end{bmatrix}} = \overset{\mathbb{Z}_{13}^{7\times 1}}{\begin{bmatrix} 4 \\ 7 \\ 2 \\ 11 \\ 5 \\ 12 \\ 8 \end{bmatrix}}$$

- $A * s = t \ \Rightarrow$ Gaussian elimination
- $A * s + \underline{e} = t \ \Rightarrow$ Reduction to lattice hard problem




Large structure! Can use polynomials for fast multiplication

# Module learning with errors problem

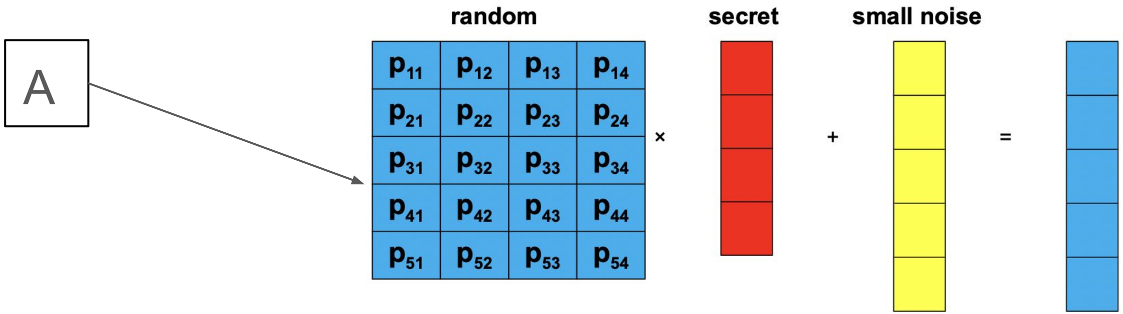



every matrix entry is a polynomial in $\mathbb{Z}_q[x]/(x^n + 1)$

- $A * s = t$ ⇒ Gaussian elimination
- $A * s + \underline{e} = t$ ⇒ Reduction to lattice hard problem

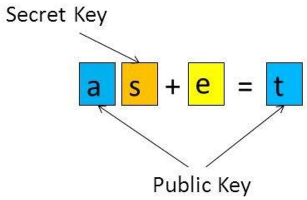
Polynomials for fast multiplication → Fast Fourier Transform

# Module learning with errors problem




every matrix entry is a polynomial in $\mathbb{Z}_q[x]/(x^n + 1)$

- $A * s = t \Rightarrow$ Gaussian elimination
- $A * s + \underline{e} = t \Rightarrow$ Reduction to lattice hard problem




Polynomials for fast multiplication $\rightarrow$ Fast Fourier Transform

# Ring-LWE

| | | | |
|---|---|---|---|
| 4 | 1 | 11 | 10 |
| 3 | 4 | 1 | 11 |
| 2 | 3 | 4 | 1 |
| 12 | 2 | 3 | 4 |
| 9 | 12 | 2 | 3 |
| 10 | 9 | 12 | 2 |
| 11 | 10 | 9 | 12 |

**q = 13**

degree is at most 3

**Ring elements** $r \in R_q = \mathbb{Z}_q[X]/(X^n + 1)$:

- Coefficients integers modulo $q$
- Degree at most $n - 1$ i.e. $r = r_0 + r_1 \cdot X + \cdots + r_{n-1} \cdot X^{n-1} \in \mathbb{Z}_q[X]/(X^n + 1)$
- Coefficient Embedding $r = (r_0, \ldots, r_{n-1}) \in \mathbb{Z}_q^n$

$4 + 1x + 11x^2 + 10x^3$

$(4 + 1x + 11x^2 + 10x^3)\ x$

$4x + 1x^2 + 11x^3 + 10x^4 \quad \text{mod } x^4 + 1$

$-10 \text{ mod } 13 = 3$

## Crystal KYBER and NTRU

- **LWE** works with vectors of integers
- **RING LWE** works with polynomials
- **Module LWE** works with vectors of polynomials

**KYBER** works with **Module LWE** – $Z_q(X)/ (X^n + 1)$

**NTRU Prime** - $Z_q(X)/ (X^n - x - 1)$

*Nth degree TRUncated polynomial ring*

## Methodology

A KEM offers three functions:

(pk, sk) = **KEM_KeyGen()**

- outputs public and private key

(ct, ss) = **KEM_Encrypt(pk)**

- generates a random value ss and encrypts it to ct using the public key pk

(ss) = **KEM_Decrypt(sk, ct)**

- decrypts ciphertext ct to plaintext ss using private key sk

# Methodology

(pk, sk) = **KEM_KeyGen()**

- outputs public and private key

(ct, ss) = **KEM_Encrypt(pk)**

- generates a random ss and encrypts it to ct using the public key pk

(ss) = **KEM_Decrypt(sk, ct)**

- decrypts ciphertext ct to plaintext ss using private key sk

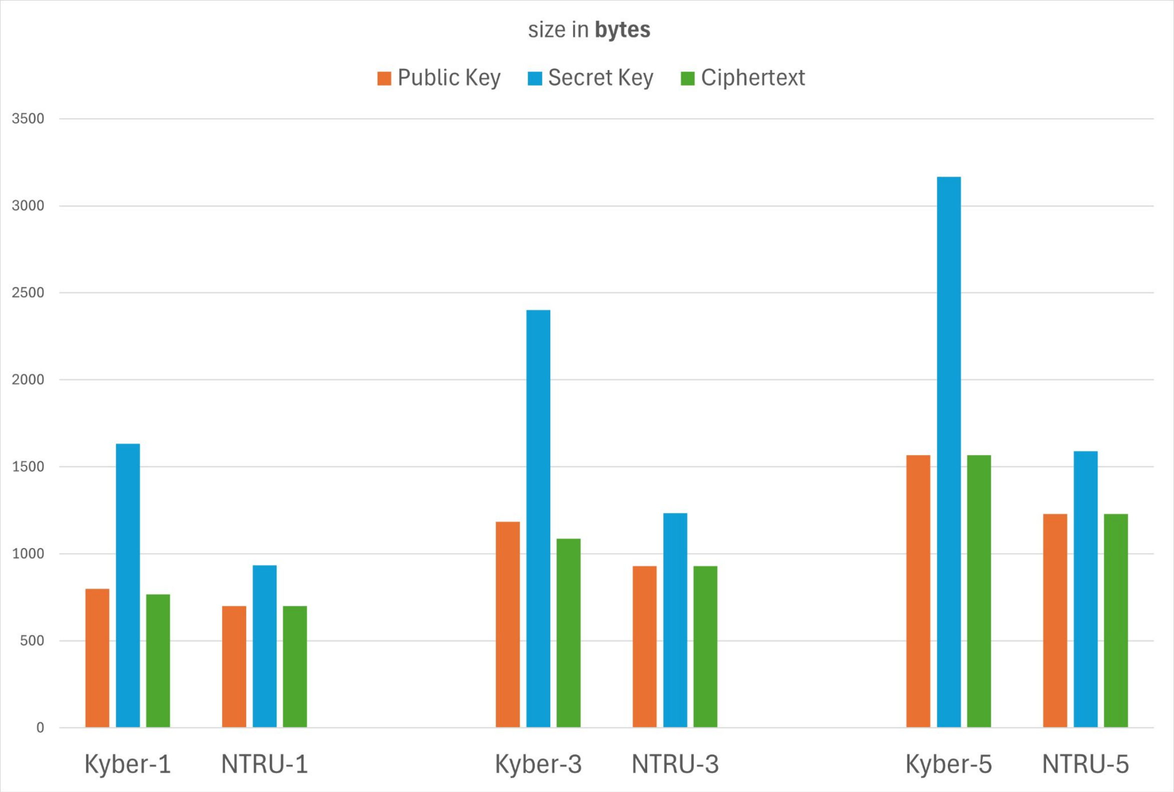
→ Space

→ Performance

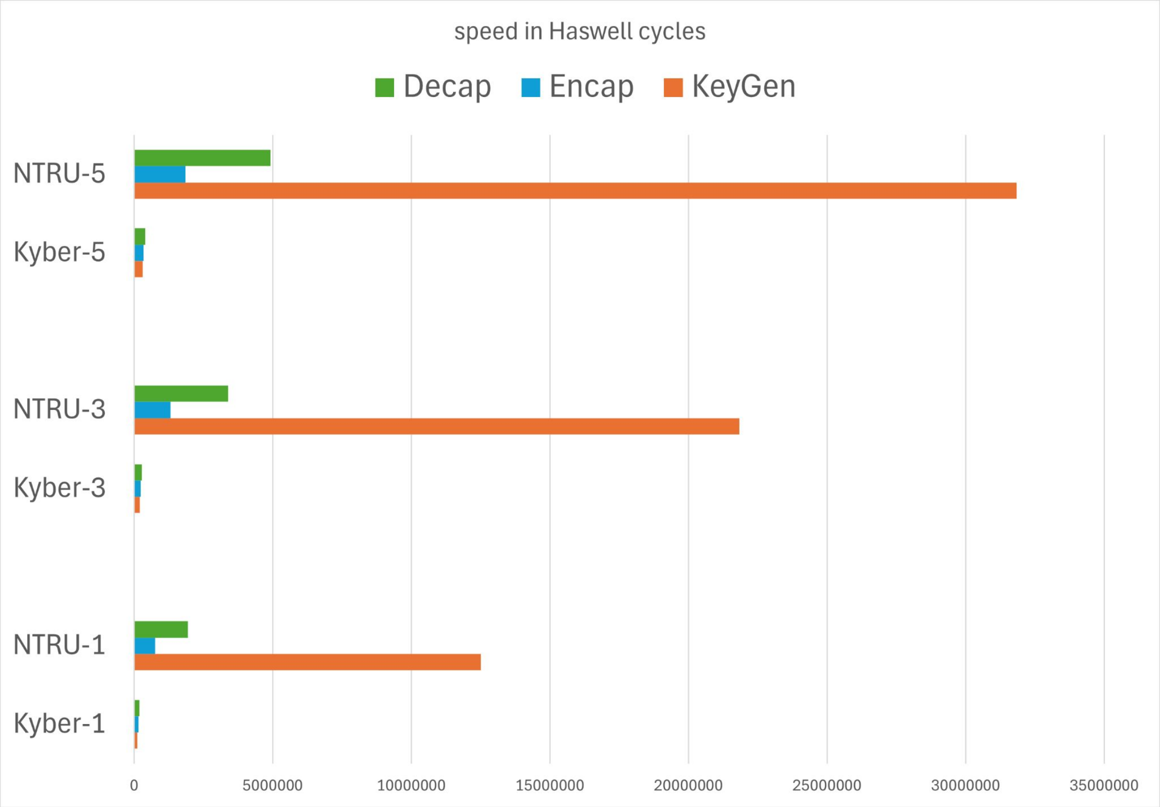
# Running the algorithms locally

```
KEM Details:
  Name: Kyber512
  Version: https://github.com/pq-crystals/kyber/commit/74cad307858b61e434490c75f812cb9b9ef7279b
  Claimed NIST level: 1
  Is IND-CCA: true
  Length public key (bytes): 800
  Length secret key (bytes): 1632
  Length ciphertext (bytes): 768
  Length shared secret (bytes): 32

Client public key:
3E 23 92 76 30 3A C5 92 ... D9 A5 70 84 42 77 12 39

It took 1716026 nanosecs to generate the key pair.
It took 447834 nanosecs to encapsulate the secret.
It took 49322 nanosecs to decapsulate the secret.

Client shared secret:
BB 4B ED 55 89 CD 27 39 ... C7 96 A6 65 B5 CA A0 AD

Server shared secret:
BB 4B ED 55 89 CD 27 39 ... C7 96 A6 65 B5 CA A0 AD

Shared secrets coincide? true
```

# Space requirements



# Performance



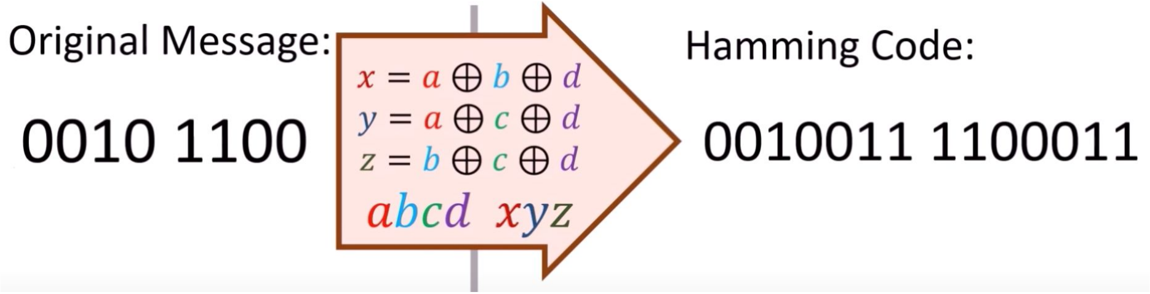
# Code-based cryptography

Hardness of **decoding randomly generated linear codes**

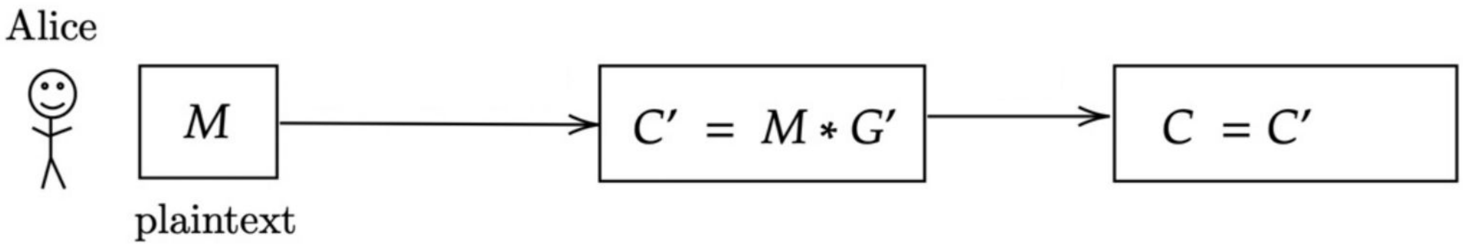
# Code-based cryptography

Hardness of decoding randomly generated linear codes

Error correcting codes → Hamming Code



## Code-based cryptography

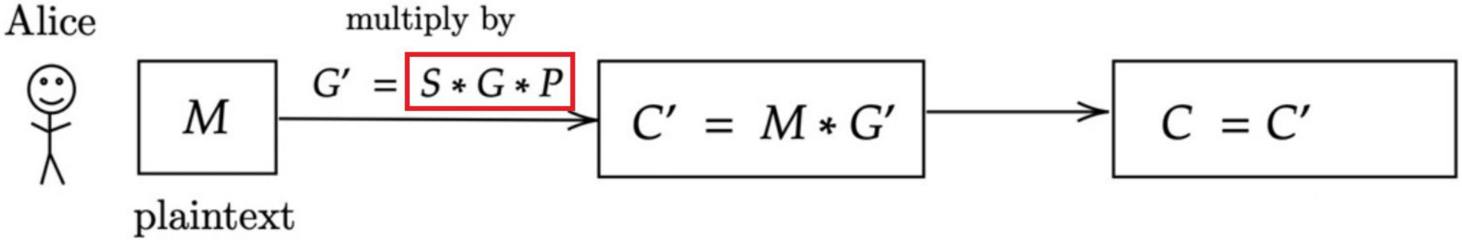



**G’ is a generator matrix** that helps create codeword

→ Has a decoding algorithm

# Code-based cryptography

G’ is a generator matrix

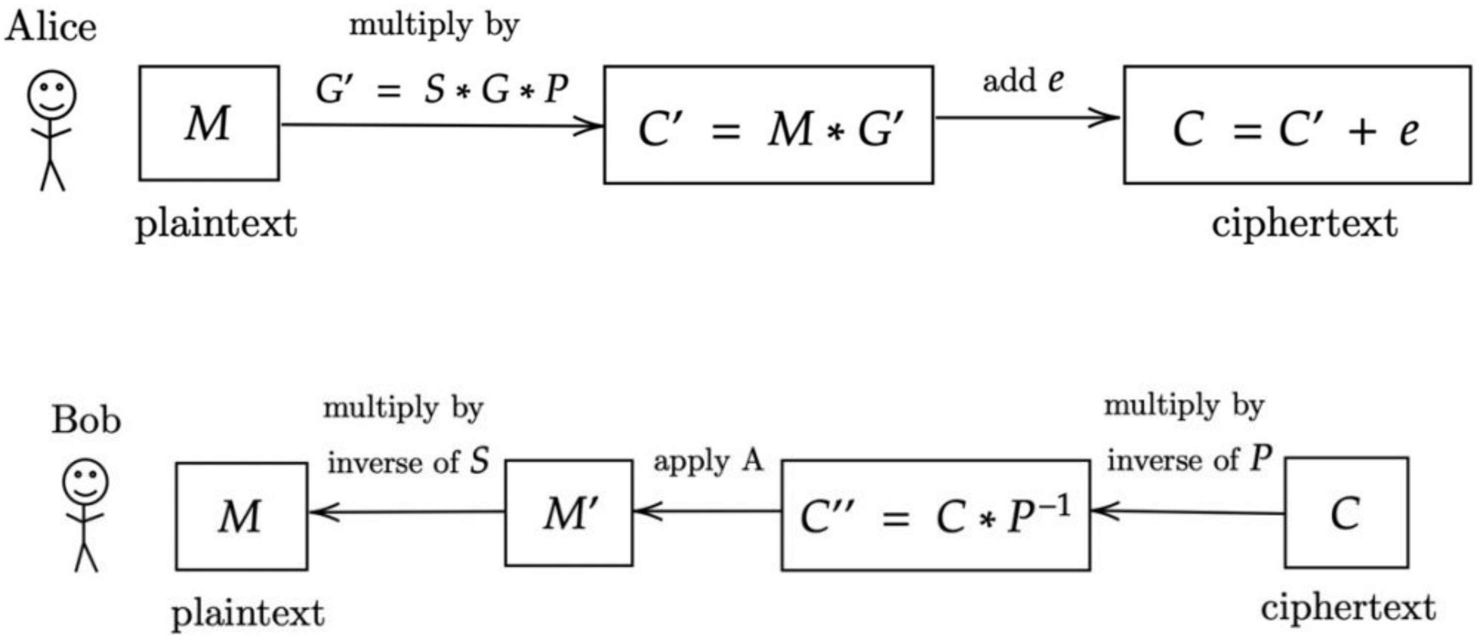



S → invertible matrix

P → permutation matrix

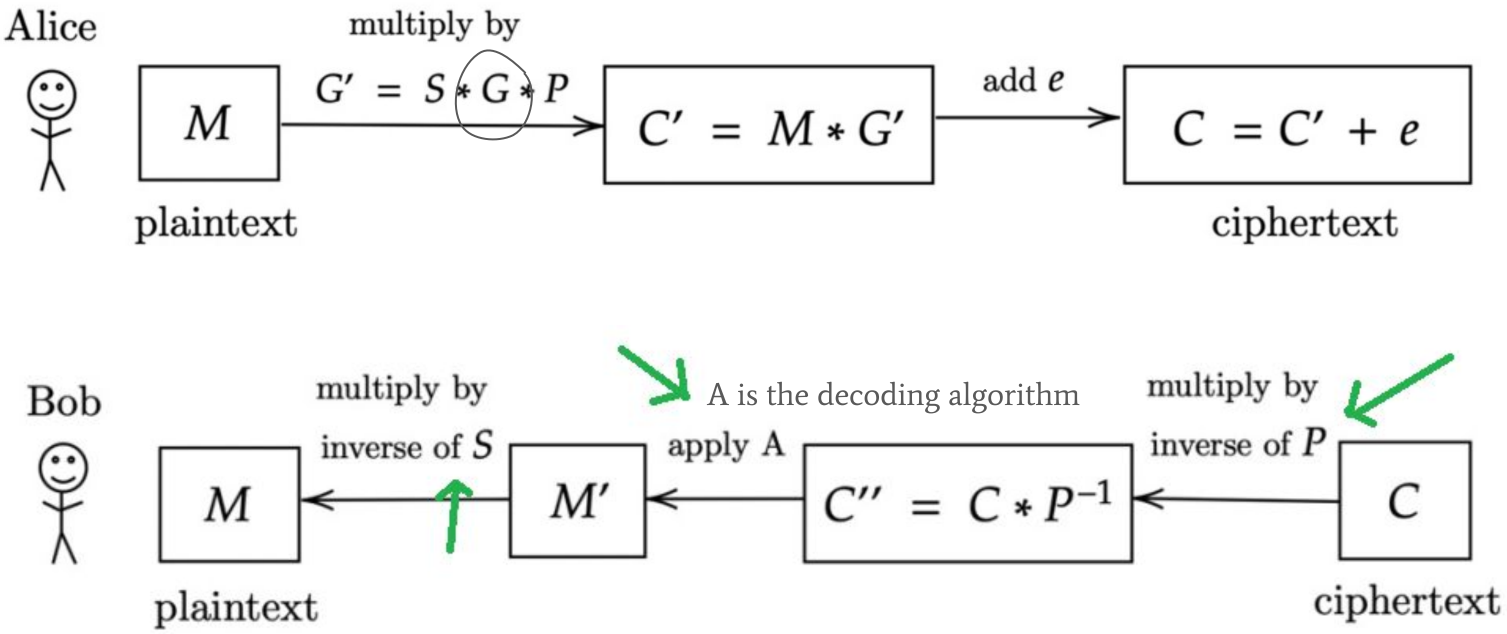
## Code-based cryptography

G’ is a generator matrix



## Code-based cryptography

G' is a generator matrix



## Code-based cryptography

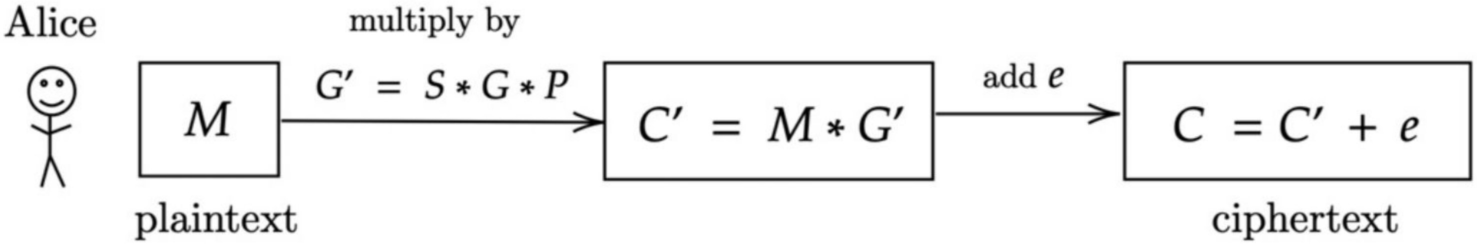
Main idea:
Error correcting code with fast, efficient decoding algorithm

Hamming code
4-bit messages → 7-bit codewords

- **Classic McEliece** ⇒ Binary Gappa Code
- **BIKE** ⇒ QC-MDPC (Quasi-cyclic moderate-density parity-check)

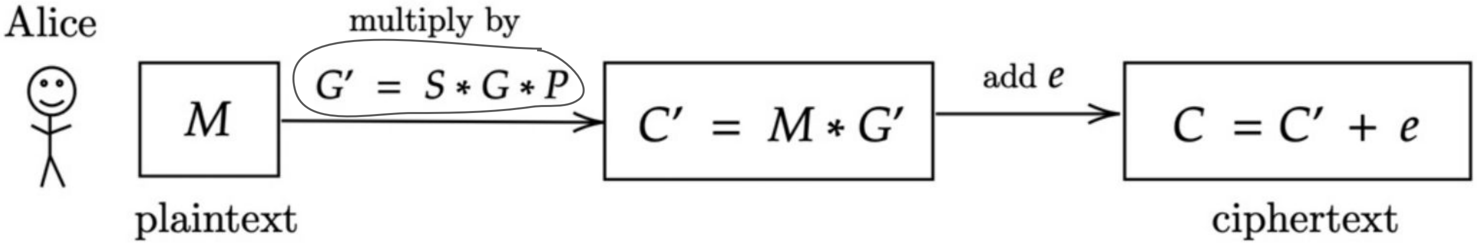
## Code-based cryptography

- Classic McEliece ⇒ Binary Gappa Code
- BIKE ⇒ QC-MDPC (Quasi-cyclic moderate-density parity-check)



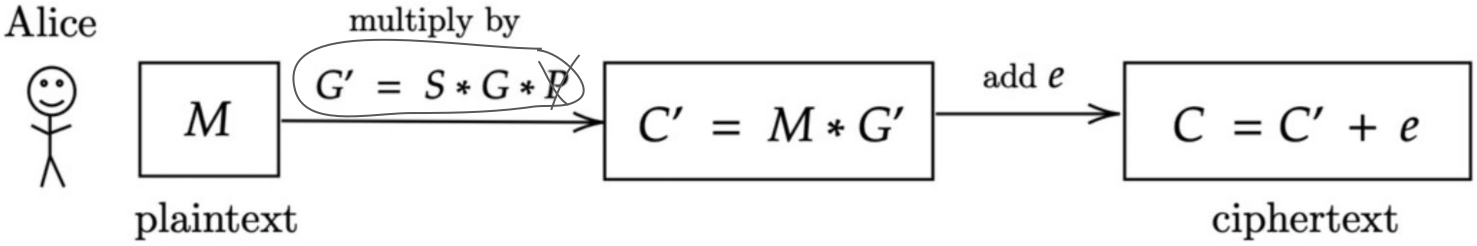
## Code-based cryptography

- **Classic McEliece** ⇒ Binary Gappa Code
- BIKE ⇒ QC-MDPC (Quasi-cyclic moderate-density parity-check)



## Code-based cryptography

- Classic McEliece ⇒ Binary Gappa Code
- **BIKE** ⇒ QC-MDPC (Quasi-cyclic moderate-density parity-check)



## Code-based cryptography

### Classic McEliece

- Fast encapsulation and decapsulation
- Smallest ciphertext among all NIST submissions

But...

$$G' = S * G * P$$

VERY large public-key sizes

Introduced in **1978**

Classic McEliece is used in the PQ-WireGuard

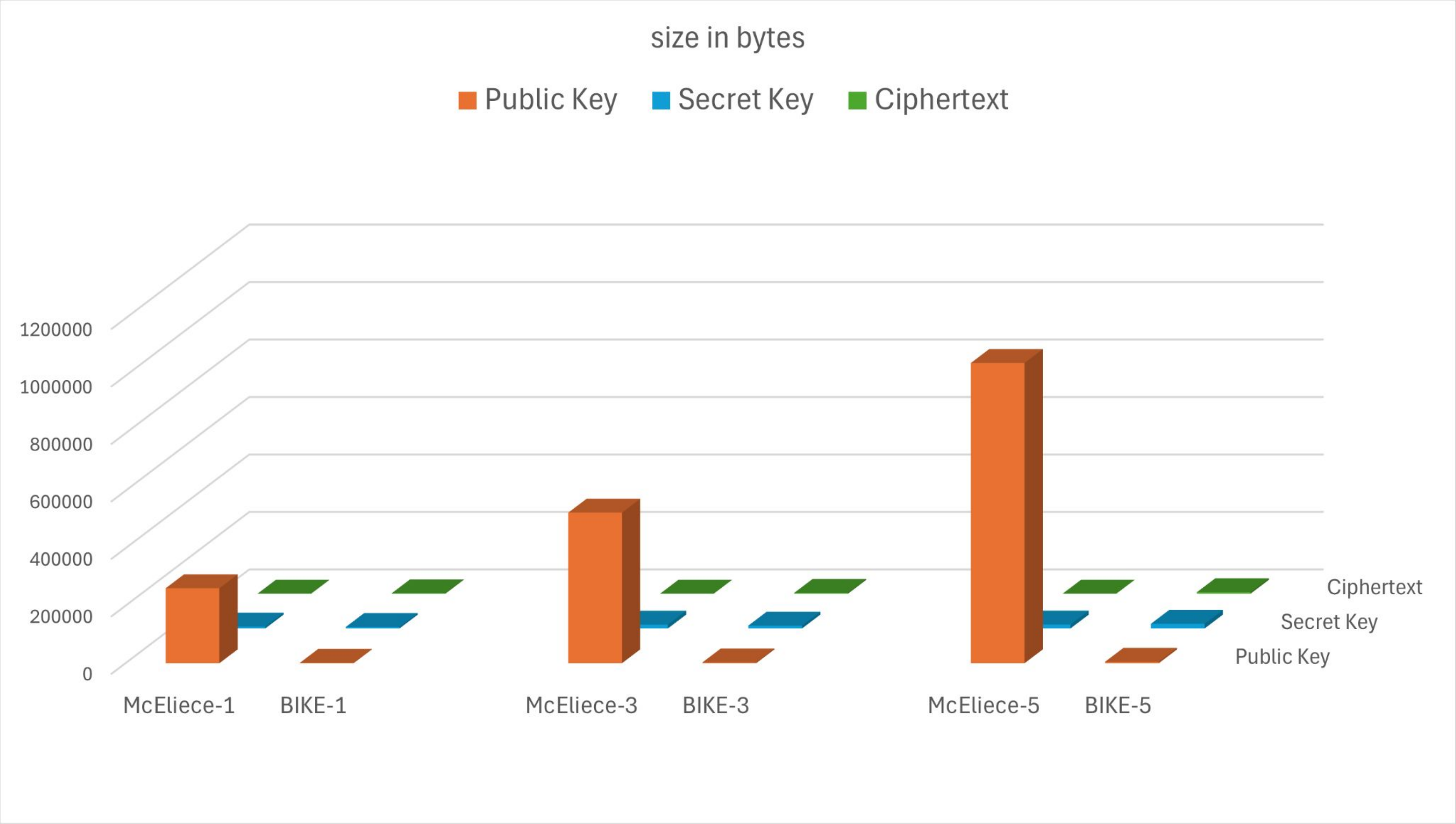
## Code-based cryptography

**BIKE (**Bit Flipping Key Encapsulation**)**

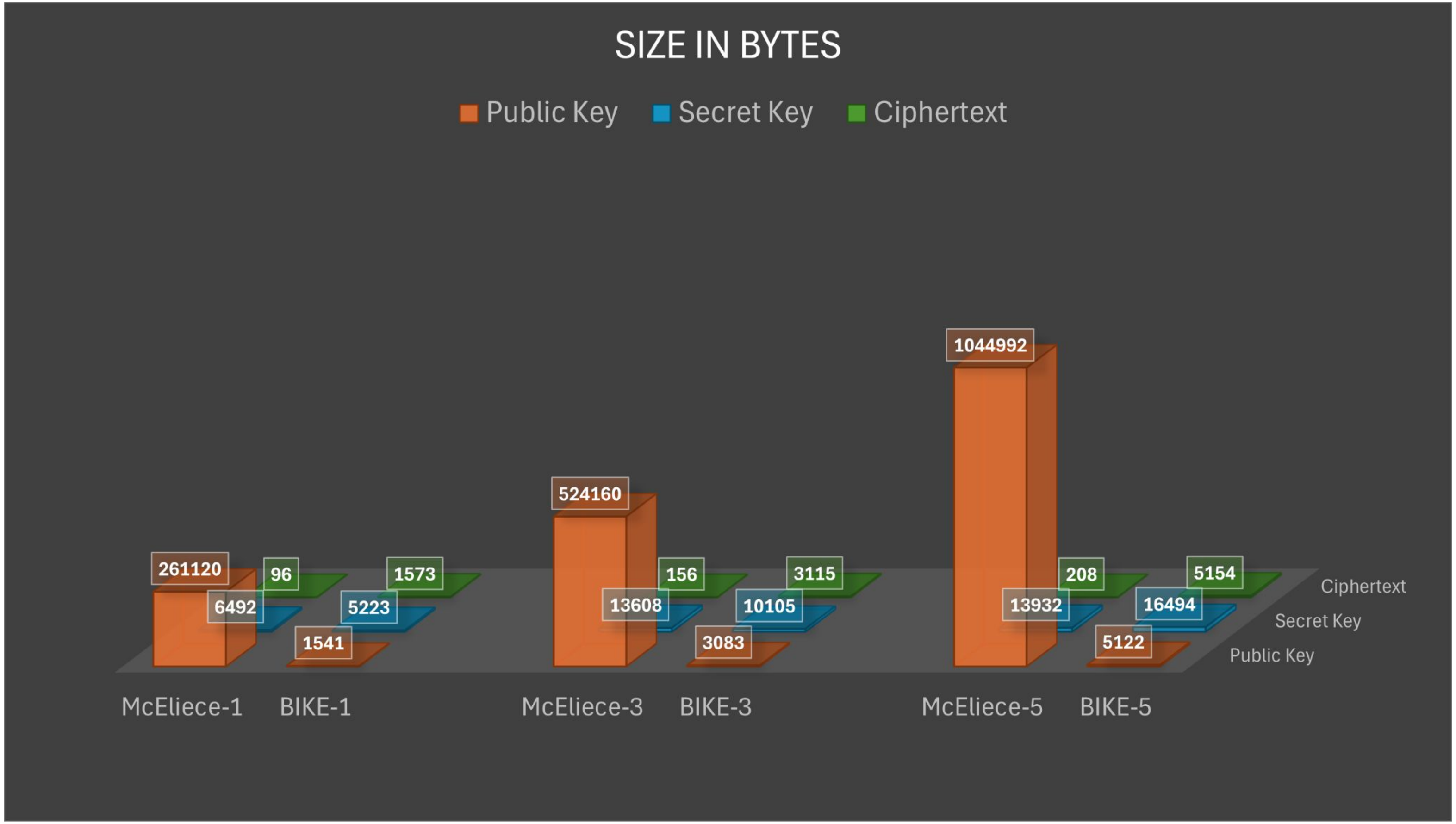
- Smallest ciphertext
- Smaller public-key size
- Not CCA-secure

5122 bytes

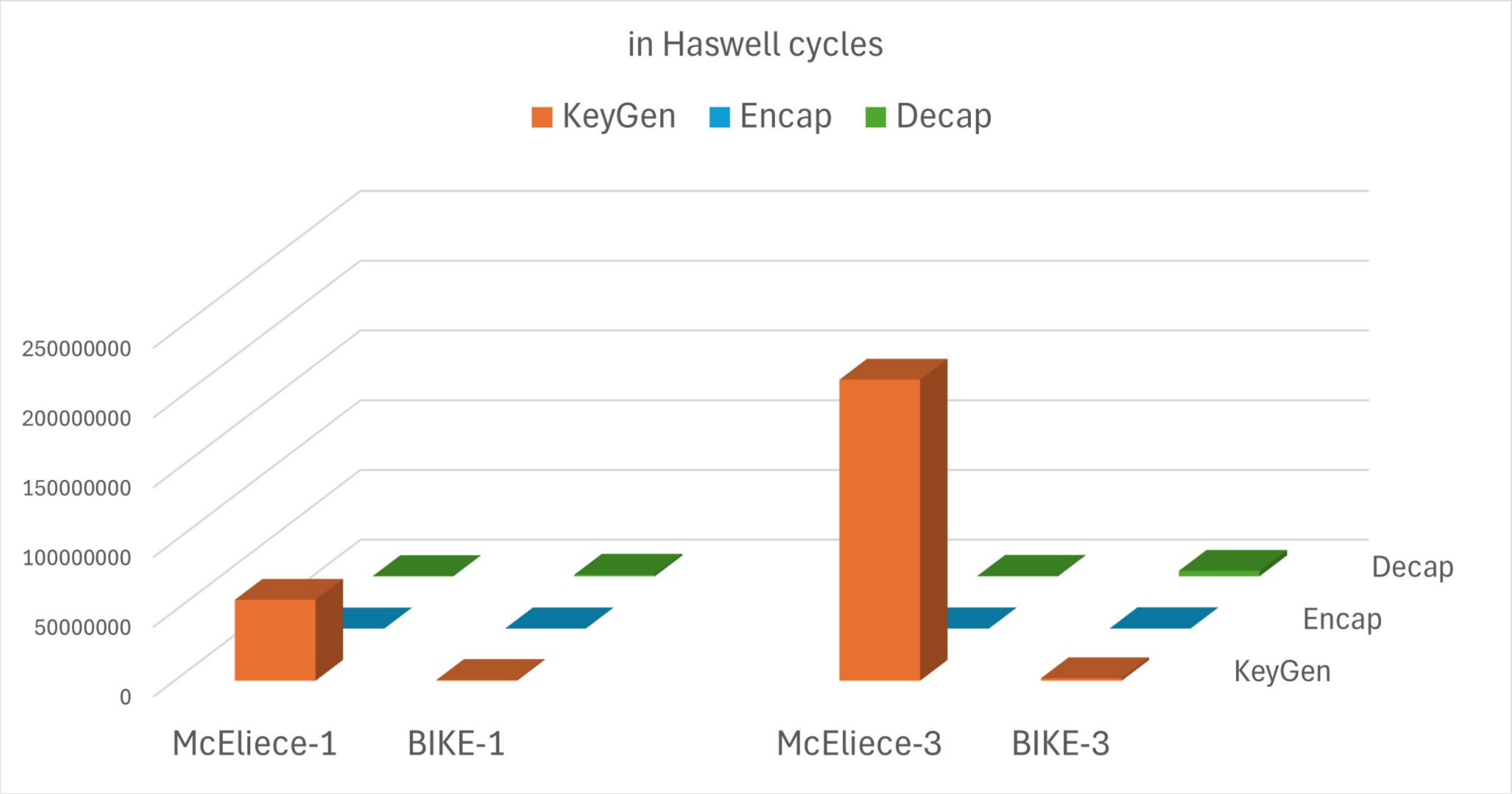
1357824 bytes in Classic McEliece

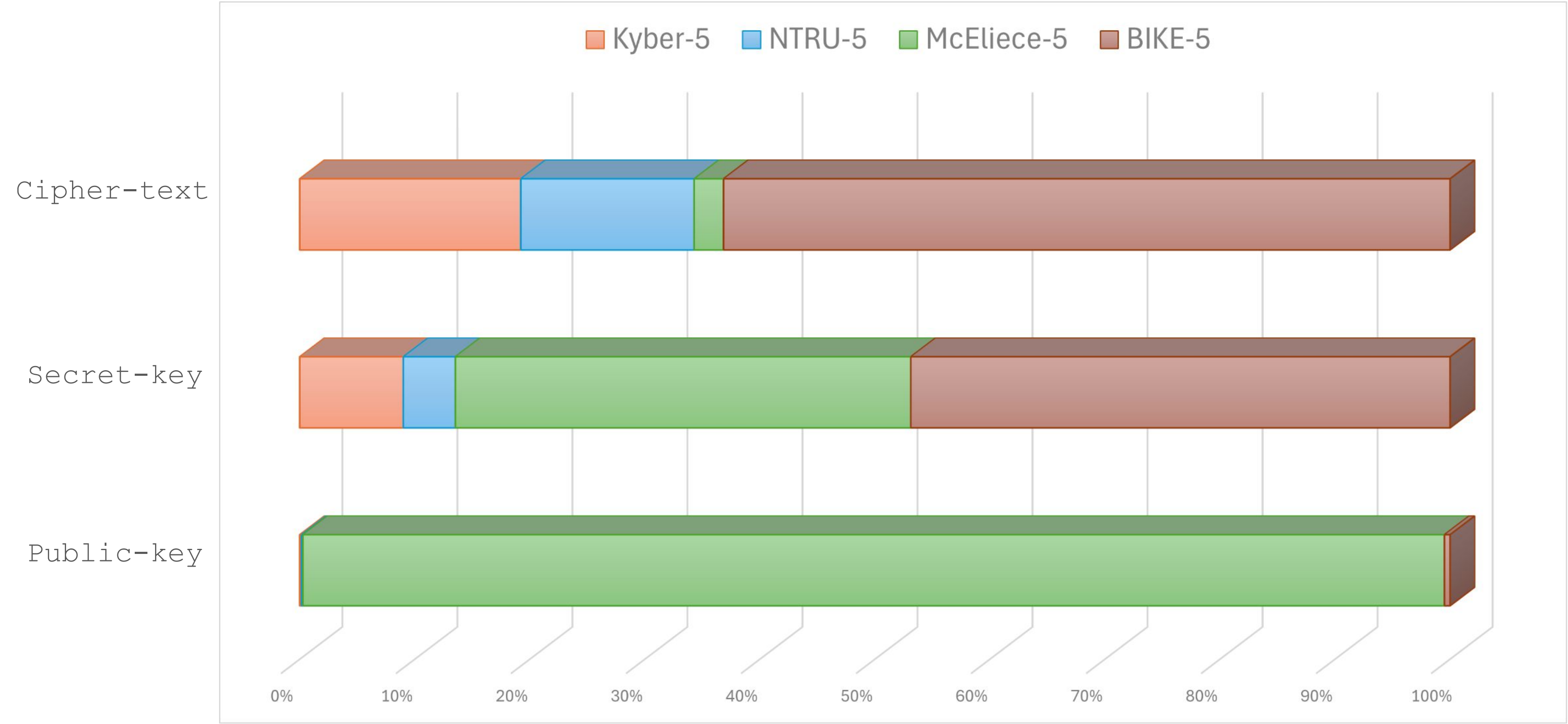


size in bytes
Public Key
Secret Key
Ciphertext
1200000
1000000
800000
600000
400000
200000
0
McEliece-1
BIKE-1
McEliece-3
BIKE-3
McEliece-5
BIKE-5
Ciphertext
Secret Key
Public Key


SIZE IN BYTES
Public Key
Secret Key
Ciphertext
261120
96
6492
1573
5223
1541
524160
156
13608
3115
10105
3083
1044992
208
13932
5154
16494
5122
Ciphertext
Secret Key
Public Key
McEliece-1
BIKE-1
McEliece-3
BIKE-3
McEliece-5
BIKE-5

# Performance





Kyber-5
NTRU-5
McEliece-5
BIKE-5
Cipher-text
Secret-key
Public-key
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%

## Wrapping it up

- NIST has chosen KYBER for standardisation

- Others are kept as alternatives

Being implemented

- WireGuard VPN
- TLS
- IPSec
- OpenSSH 9.0 has NTRU implemented