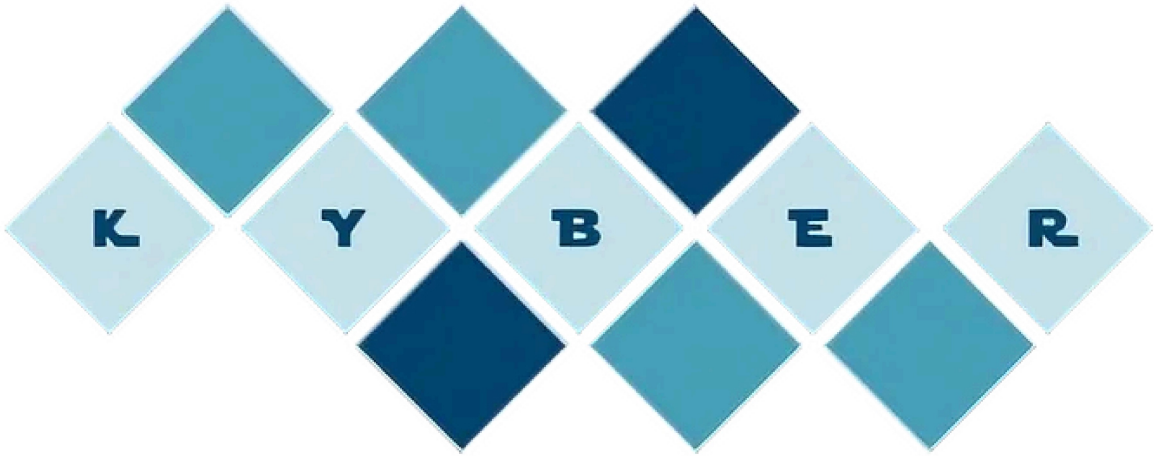
CRYSTALS Kyber : The Key to Post-Quantum Encryption

Udara Pathum · Follow

5 min read · Jan 5, 2024



Kyber Logo — PQ Crystals

In the digital realm, quantum computing threatens traditional encryption. NIST's Post-Quantum Cryptography Standardization initiative aims to set new encryption standards. They've selected advanced cryptographic algorithms to withstand quantum computing's power.

Quantum Computing: The Demise of Traditional Cryptography

In today's digital world, our secrets and data stay safe thanks to powerful codes that take supercomputers millions of...

medium.com

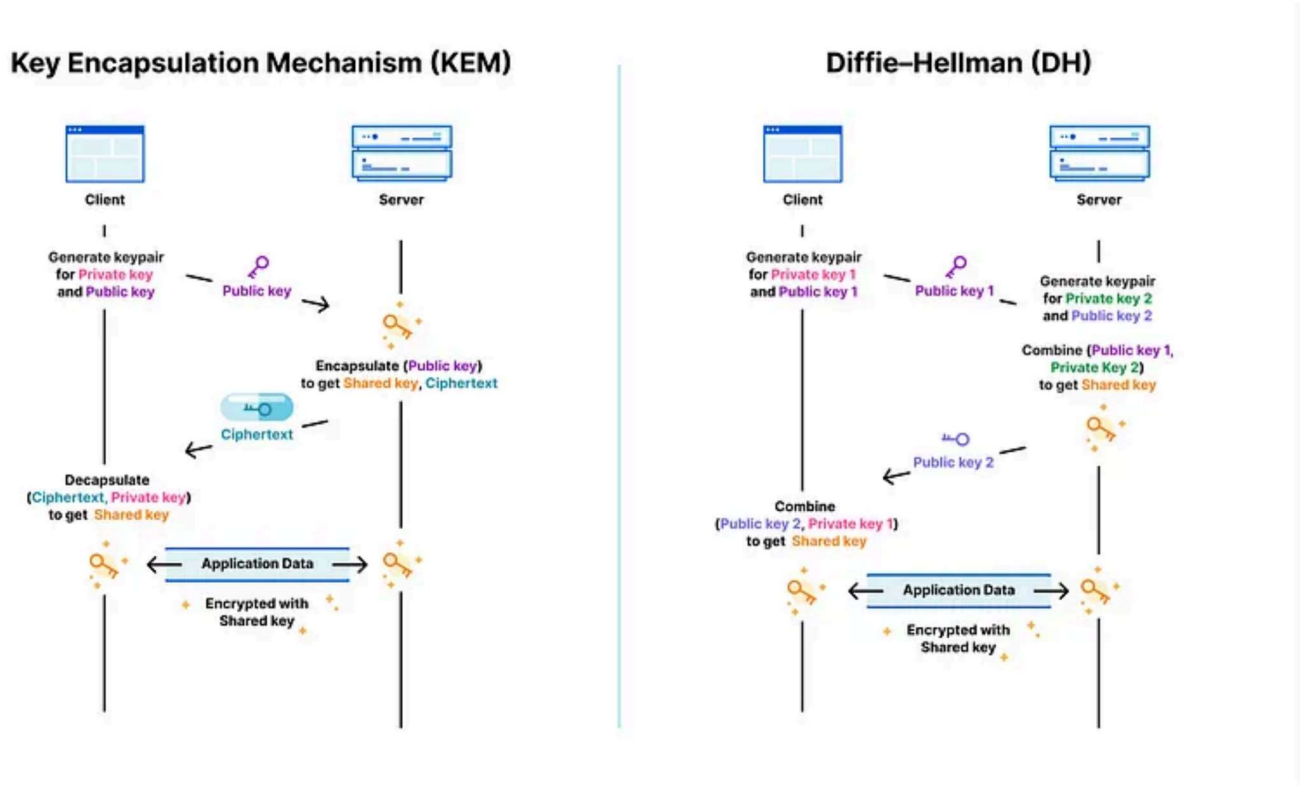
Kyber, one of these selected algorithms, is purpose-built to resist quantum attacks. Its design, based on lattice structures, strengthens data security against potential quantum threats, offering a promising path in quantum-resistant encryption.

Kyber, a secure Key-Encapsulation Mechanism (KEM), relies on the challenge of solving the learning-with-errors (LWE) problem over module lattices for its security. Let's dive into the core of Kyber's strength, examining how it navigates the intricacies of LWE to ensure robust encryption in the face of potential quantum threats.

Key-Encapsulation Mechanism (KEM)

A Key-Encapsulation Mechanism (KEM) is used to send a symmetric key between two parties using asymmetric algorithms. Unlike **Diffie-Hellman key exchange method** where the shared secret is directly generated through mutual computations, a KEM employs asymmetric algorithms. In this method, the sender encapsulates the symmetric key within a cipher-text using the recipient's public key. Upon receiving the cipher-text, the recipient then decapsulates and retrieves the symmetric key using their private key,

ensuring a secure and authenticated exchange without directly sharing the symmetric key during transmission.



KEM vs Diffie-Hellman — Cloudfare

The Learning With Errors (LWE) Problem

Imagine the following system of linear equations where **A** and **b** are the public key and **s** is the private key vector which is a solution to the equation **A×s = b**. This can be solved easily by using Gaussian elimination. The answer in this case is **s = (0, 13, 9, 11)**.

$$\begin{bmatrix} 14 & 15 & 5 & 2 \\ 13 & 14 & 14 & 6 \\ 6 & 10 & 13 & 1 \\ 10 & 4 & 12 & 16 \end{bmatrix} \times \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} 262 \\ 374 \\ 258 \\ 336 \end{bmatrix}$$

We can slightly modify the equations by increasing the number of equations, and introducing an error vector **e** with small whole numbers and make the equation **A×s+e=b** making it much more complex to compute **s.** Additionally, modular arithmetic is added to increase the complexity of the equations.

$$A = \begin{bmatrix} 14 & 15 & 5 & 2 \\ 13 & 14 & 14 & 6 \\ 6 & 10 & 13 & 1 \\ 10 & 4 & 12 & 16 \\ 9 & 5 & 9 & 6 \\ 3 & 6 & 4 & 5 \\ 6 & 7 & 16 & 2 \end{bmatrix} e = \begin{bmatrix} 1 \\ -1 \\ 0 \\ -1 \\ 1 \\ 0 \\ 1 \end{bmatrix} s = \begin{bmatrix} 0 \\ 13 \\ 9 \\ 11 \end{bmatrix} q = 17$$

$$b = A \times s + e \pmod{q} = \begin{bmatrix} 8 \\ 16 \\ 3 \\ 12 \\ 9 \\ 16 \\ 3 \end{bmatrix}$$

The Ring Learning With Errors (Ring-LWE) Problem

This is a mathematical challenge in lattice-based cryptography, where the goal is to conceal a secret polynomial within noisy data sampled from a structured ring. I'll attempt to provide a simplified definition.

- $\mathbf{a}(x)$ is a polynomial from polynomial ring $Z_p[X]/(X^n+1)$ where all coefficients are from Z_p
- $\mathbf{e}(x)$ and $\mathbf{s}(x)$ are also polynomials from $Z_p[X]/(X^n+1)$ with small coefficients
- Then, $\mathbf{b}(x) = \mathbf{a}(x) \,.\, \mathbf{s}(x) + \mathbf{e}(x)$ where $b(x)$ is also a polynomial from $Z_p[X]/(X^n+1)$

Note that here all **a, b, s** and **e** are polynomials where in LWE **A** is a matrix. Now let's learn how to do calculations in a polynomial ring.

Addition over a polynomial ring

This is simillar to the way to traditionally adding polynomials except the coefficients should be from Z_p. For instance, if the ring is defined modulo x^3+1, the result of adding the above polynomials might be $(2x^2+3x+1) + (x^2-4x+5) \equiv 3x^2-x+6 \pmod{x^3+1}$.

Multiplication over a polynomial ring

Let's use the following example to explain how the multiplication works between two polynomials.

$$a, b \in \mathbb{Z}_{13}[X]/(X^4 + 1)$$

$$a = \left(4x^3 + 1x^2 + 11x + 10\right)$$
$$b = \left(6x^3 + 9x^2 + 11x + 11\right)$$

To do the multiplication **a × b,** we are going to convert polynomial **a** into a *circulant matrix.* As you can see in the image, each column of matrix **A** is a cyclic shift from the column before where the last element is negated when shifting to the front.

$$A = \begin{bmatrix} 4 & -10 & -11 & -1 \\ 1 & 4 & -10 & -11 \\ 11 & 1 & 4 & -10 \\ 10 & 11 & 1 & 4 \end{bmatrix} B = \begin{bmatrix} 6 \\ 9 \\ 11 \\ 11 \end{bmatrix}$$

$$A \times B \mod q = \begin{bmatrix} -198 \\ -189 \\ 9 \\ 214 \end{bmatrix} \mod 13 = \begin{bmatrix} 10 \\ 6 \\ 9 \\ 6 \end{bmatrix}$$

How Kyber works

Algorithm	n	k	q
Kyber512	256	2	3329
Kyber768	256	3	3329
Kyber1024	256	4	3329

Kyber Specification Parameters

The provided table presents the values of n, k, and q as per the Kyber specification. Yet, for the purpose of explaining the functioning of Kyber, we will employ more straightforward parameters: *k=2*, *q=17,* and *n=4*.

Key Generation

The private key of Kyber uses *k* number of polynomials which have a degree of *n* (called **s**). This is generated using random small coefficients.

$$s = (-x^3 - x^2 + x, -x^3 - x)$$

A Kyber public key consists of two elements. A matrix of random polynomials **A** ($k \times k$) and a vector of polynomials **t**. Matrix **A** is generated using coefficients ($< q$). To calculate vector **t**, an additional error vector **e** is required. This is also generated using random small coefficients. Then we can calculate $\mathbf{t}=\mathbf{A}\times\mathbf{s}+\mathbf{e}$. Note that all operations are under the *polynomial ring* $Z_{17}[X]/(X^4+1)$.

$$A = \begin{pmatrix} 6x^3 + 16x^2 + 16x + 11 & 9x^3 + 4x^2 + 6x + 3 \\ 5x^3 + 3x^2 + 10x + 1 & 6x^3 + x^2 + 9x + 15 \end{pmatrix}$$

$$e = \left(x^2, x^2 - x\right) t = \left(16x^3 + 15x^2 + 7, 10x^3 + 12x^2 + 11x + 6\right)$$

Encryption

To encrypt the message **m**, we need to convert it into a binary polynomial. Then we need to multiply it by $\lfloor q/2 \rfloor$ (integer closest to $q/2$). Lets take 11 as the message for the example.

$$m_b = 1x^3 + 0x^2 + 1x^1 + 1x^0 = x^3 + x + 1$$
$$m = \left\lfloor \frac{q}{2} \right\rceil \times m_b = 9x^3 + 9x + 9$$

We need 3 random small polynomials **r**, $\mathbf{e_1}$, $\mathbf{e_2}$

$$r = \left(-x^3 + x^2, x^3 + x^2 - 1\right)$$
$$e_1 = \left(x^2 + x, x^2\right)$$
$$e_2 = -x^2 - x$$

Then we encrypt the value **m** using the public key (**A**,**t**). The encryption procedure calculates two values **u** and **v.**

$$u = A^T r + e_1$$
$$v = t^T r + e_2 + m$$
$$u = \left(11x^3 + 11x^2 + 10x + 3, 4x^3 + 4x^2 + 13x + 11\right)$$
$$v = \left(7x^3 + 6x^2 + 8x + 15\right)$$

Decryption

We can use the secret polynomial **s** retrieve the secret **m** from ciphertext. Note that m is still noisy.

$$m_n = v - s^T u$$
$$m_n = e^T r + e_2 + m + s^T e_1$$
$$m_n = 7x^3 + 14x^2 + 7x + 5$$

The noise can be removed by comparing the received value to the closest valid message. In this case, by checking if closer to $\lfloor 2/q \rceil = 9$ or 0 (or q). Then we get the rounded polynomial $9x^3+0x^2+9x+5$, and diving by 9 will give **m**.

Now both parities has shared secret **m** which they can use for asymmetric encryption.

Further Reading

CRYSTALS Kyber KEM for hybrid encryption with Java

Kyber is a Post-Quantum Key-Encapsulation Mechanism (KEM) which can be used for share secrets between two parties. It...

medium.com

Reference

1. https://pq-crystals.org/kyber/
2. https://doi.org/10.6028/NIST.FIPS.203.ipd
3. https://blog.cloudflare.com/post-quantum-key-encapsulation/
4. https://blog.cloudflare.com/post-quantum-for-all/
5. https://cims.nyu.edu/~regev/papers/lwesurvey.pdf
6. https://www.youtube.com/watch?v=gp7KFOs7y3g
7. https://cryptopedia.dev/posts/kyber/

Kyber | Post Quantum Cryptography | Encryption | Asymmetric Encryption

Written by Udara Pathum

41 Followers

Senior Software Engineer @ WSO2

Follow

More from Udara Pathum

Udara Pathum in Identity Beyond Borders

JWT vs Opaque Tokens: All You Need to Know

In modern web applications, authentication and authorization are essential components...

Feb 22, 2023 70

Rashmini Naranpanawa in Identity Beyond Borders

OAuth 1.0 Vs OAuth 2.0

OAuth (Open Authorization) is a protocol used for access delegation, where resource...

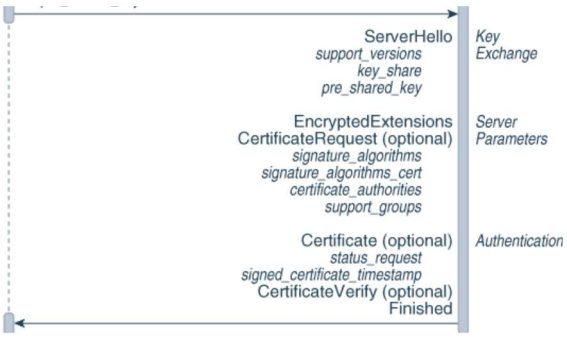
Aug 29, 2021 224 2

Janak Amarasena in Identity Beyond Borders

What the heck is PKCE?

PKCE is short for Proof Key for Code Exchange. It is a mechanism that came into...

Sep 13, 2019 847 6



Udara Pathum

X25519Kyber768 Post-Quantum Key Exchange for HTTPS...

The X25519Kyber768 algorithm combines the elliptic curve-based X25519 with the...

Feb 3 21 2

See all from Udara Pathum

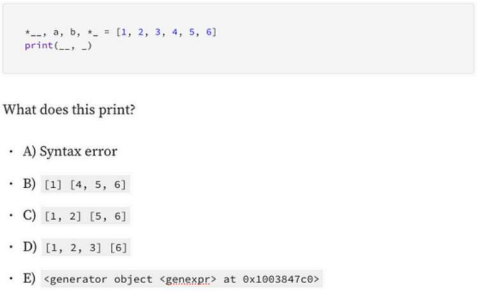
Recommended from Medium

Afan Khan in JavaScript in Plain English

Microsoft is ditching React

Here's why Microsoft considers React a mistake for Edge.

Jun 6 3.2K 70



Liu Zuo Lin

You're Decent At Python If You Can Answer These 7 Questions...

No cheating pls!!

Mar 6 6.7K 30

Lists

Productivity

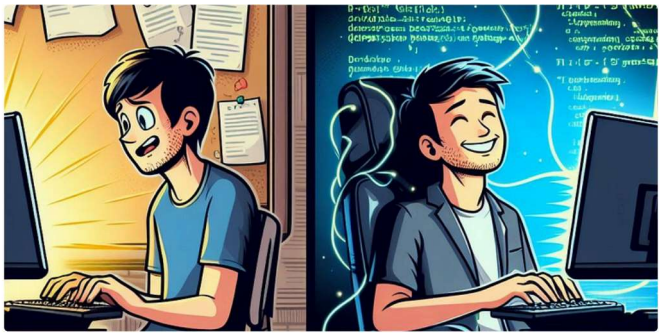
241 stories · 520 saves

Kevin Beaumont in DoublePulsar

Inside the failed attempt to backdoor SSH globally—that got...

Why the threat actor rushed deployment.

Mar 31 964 11

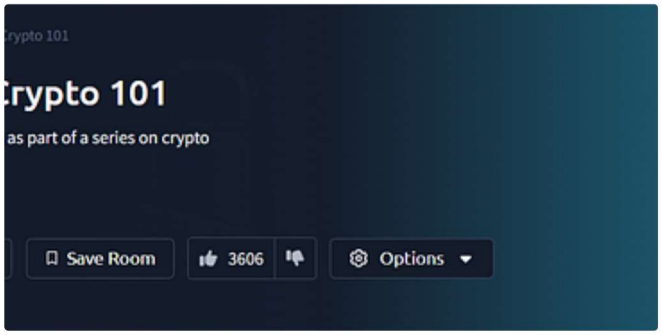


Abhay Parashar in The Pythoneers

17 Mindblowing Python Automation Scripts I Use Everyday

Scripts That Increased My Productivity and Performance

Jul 29 6.2K 47



Gideon Okechukwu

Encryption—Crypto

Task 1: What will this room cover?

3d ago

Derek Johnson

I'm Unemployed for Over Two Years (as a software engineer)

In 2022, I worked on a contract as a software engineer at Apple. Apple dissolved our entir...

Jun 1 6.1K 149

See more recommendations