Alekhnovich's Cryptosystem

Mattia Campana, 2024

1 Introduction

In 2006, Clive Humby coined a slogan that has proven increasingly prescient over time: "Data is the new oil." Nothing could be truer in today's context. Much like any valuable resource, data has become a coveted asset, with individuals and entities vying to acquire and leverage it to their advantage. Consequently, from the outset, the need for secure communications and data protection was paramount.

Over the years, technological advancements have given rise to new algorithms and protocols designed to maximize data security. Tremendous strides have been made in safeguarding sensitive information, and yet, we find ourselves facing an imminent threat: the advent of quantum computers.

This threat to data security is not a hypothetical scenario; it is a reality on the horizon. Quantum computers, leveraging the principles of quantum mechanics, possess the potential to reshape the digital landscape by solving complex problems at unprecedented speeds. As we increasingly rely on digital communication and data storage for our most critical operations, the need for robust cryptographic defenses against quantum threats has never been more urgent.

2 Theoretical Foundamentals

To address the imminent threat posed by quantum computers, the computer science community has long been engaged in the quest for alternatives to current symmetric and asymmetric encryption algorithms. Concerning the latter, various proposals have been presented to the public and the community, yet only a few have been recognized as valid and genuinely effective against quantum computers. As for other algorithms, we have thus far been unable to demonstrate either their efficacy or inefficacy in the face of quantum computers. Among these is the Alekhnovich cryptosystem, the focus of our exploration in this treatise.

To delve deeper into this cryptosystem, it is undoubtedly essential to introduce several concepts from the fields of coding theory and geometry and linear algebra. This introduction will enhance the comprehensibility of its implementation, both in terms of logic and procedures.

2.1 Linear Algebra

Linear algebra, a cornerstone of mathematics, delves into the analysis of vector spaces and their transformations. Its significance lies in its wide-ranging applications, encompassing the solution of systems of linear equations and the exploration of multidimensional objects in space. To undertake these analyses, linear algebra employs a diverse toolkit, including vector spaces, linear equations, and matrices.

Matrices, rectangular arrays of numbers, symbols, or expressions meticulously organized in rows and columns, assume a pivotal position in linear algebra. They offer a concise and structured representation of vectors, linear transformations, and systems of linear equations. Matrices, amenable to various operations such as addition, multiplication, and transposition, open pathways to tackling a broad spectrum of mathematical challenges

Matrices Sum The sum of two matrices A and B of the same dimension is defined as the matrix C whose entries are the sum of the corresponding entries of A and B. In other words, if $A = [a_{ij}]$ and $B = [b_{ij}]$, then $C = [c_{ij}]$ where $c_{ij} = a_{ij} + b_{ij}$.

Matrices Product The product of a matrix A with dimensions $(m \times n)$ and a matrix B with dimensions $(n \times p)$ is defined as a matrix C with dimensions $(m \times p)$. The entry c_{ij} of C is defined as the sum of the products of the entries in row i of A and the corresponding entries in column j of B. In other words, $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} C = A \times B = \begin{bmatrix} 1 \cdot 5 + 2 \cdot 7 & 1 \cdot 6 + 2 \cdot 8 \\ 3 \cdot 5 + 4 \cdot 7 & 3 \cdot 6 + 4 \cdot 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 41 & 50 \end{bmatrix}$$

$\mathbb{Z}_2$ Field and $\mathbb{Z}_2^n$ Vector Space In both cases, we refer to the set $\mathbb{Z}_2$, representing the set of integers $mod 2$, euqals to $\{0, 1\}$. In the case of fields, this set combines two operations that remain closed concerning the field set. Regarding the vector space $\mathbb{Z}_2^n$, its made by vectors of size n, where each element belongs to $\mathbb{Z}_2$, where addition and scalar multiplication are defined operations, also closed within the set. Furthermore, there is certainty about the validity of certain properties such as associativity and distributivity.

In summary, a field constitutes a structure with specified arithmetic properties, while a vector space denotes a collection of objects (vectors) adhering to specific regulations (dictated by vector addition and scalar multiplication), and these vectors can be formed over a field. Fields furnish the scalars necessary in defining vector spaces.

2.2 Coding Theory

The term coding theory refers to the analysis of code properties and its numerous applications. In the realm of communications and information processing, the

concept of a code pertains to a set of rules designed to transform data, such as text, numbers, images, etc., into other forms that are more suitable for storage, transmission, or encryption.

Code The code it can seen as a function that given a word x encode that word over a chosen alphabet as $C : X \to \sum^*$, where $C(x)$ is the code associated whith x. Proprietis of code:

1. injective $\to$ non-singular(the length is arbitrary)
2. non-singular $\to$ uniquely decodable

Hamming Distance The Hamming distance is a measure used to quantify the dissimilarity between two strings of equal length in information theory, coding theory, and computer science. It calculates the minimum number of substitutions required to change one string into another by altering individual elements.

For instance, consider two strings: 101010 and 111011. Their Hamming distance is 2 because, to convert the first string into the second, two substitutions are needed: changing the second and fifth elements from 0 to 1.

Channel Coding The detection and correction of errors in a code ensure the integrity of data information, whether transmitted over noise-exposed channels or for simple storage purposes. The principle is straightforward: redundancy, the addition of seemingly unnecessary information, is essential for anyone wishing to verify data accuracy or recover lost or corrupted information fragments. Various algorithms have been implemented over time, ranging from the most obvious and simple to the most effective and complex, including:

- Repetition Code: a coding scheme that repeats blocks of bits n times, determining the correct value between 0 and 1 depending on which repeats more than $n/2$ times.
- Error-correcting Code: this represents more of an error detection family. A code with a minimum Hamming distance can identify up to $d-1$ errors in a code word (e.g. $d = 2 \to$ Parity bit, $d = 4 \to$ Extended Hamming).

Generator and Check Matrices In the context where the code C represents a linear subspace of $\mathbb{F}_q^n$, it can be expressed as a linear combination of its codewords, which form a basis typically organized into rows within a generating matrix G. This matrix serves as a representation of the code itself, simplifying the transformation of data vectors into code words through straightforward matrix multiplication. Subsequently, a check matrix (or parity check matrix) is constructed, characterized by its kernel [1] being C. The code that H can gener-

[1] Which is typically the pre-image of 0. An important special case is the kernel of a linear map. The kernel of a matrix, also known as the null space, refers to the kernel of the linear map defined by the matrix.

ate is termed the dual code of C. When the G matrix takes the form $[\mathbb{I}_k|P]$, it is considered to be in standard form, and consequently, the H matrix appears as $[-P^T|\mathbb{I}_{n-k}]$.

Cryptographic Coding Cryptographic coding is a field dedicated to securing data integrity, confidentiality, and authentication. Encryption, a key component, converts readable data into secure, unreadable ciphertext, ensuring confidentiality. Decryption, its counterpart, allows authorized users with the correct key to access and transform ciphertext back into its original form for use.

Public-key cryptography, also known as asymmetric cryptography, stands out for its use of key pairs - a public key for encryption and a private key for decryption. This innovation simplifies secure communication by allowing individuals to openly share their public keys while keeping their private keys confidential. This concept plays a pivotal role in protecting digital systems in today's interconnected world.

3 Alekhnovich's Cryptosystem

3.1 Overview

The purpose of this paper is not to explain in detail the operation of Alekhnovich algorithm and why, unlike those currently implemented, it could offer advantages in the face of quantum computers. To better understand the various steps, it is necessary to provide an overview of its operation:

The Algorithm The algorithm begins by generating two random matrices, $A \in \mathbb{Z}_2^{k \times n}$ and $S \in \mathbb{Z}_2^{l \times k}$. After, it generates a third random matrix, $E \in \mathbb{Z}_2^{l \times n}$. The rows of E are chosen from random and independent vectors of weight t (the weight is computed as $\sqrt{n}$).The algorithm then defines the matrix Y as $Y = SA + E$.

The matrices Y and A represent the public key of the algorithm. The public key is shared with anyone who wants to send encrypted messages to the sender. The plain text to encrypt space is $\mathbb{M} = C \subset \{0,1\}^l$ where C is an error-correcting code. The error-correcting code implies an algorithm used to ensure that the message can be recovered even if it is corrupted by noise. Now following steps are required for a secure exchange of information:

- To encrypt the message, the sender performs the following steps:
  1. The sender generates a random t-weight vector e of $\mathbb{F}_2^n$.
  2. The sender multiplies e^T by the public key Y and A.
  3. The sender adds the message m to the result of the multiplication Ye^T.

  The encrypted message is then given by the following vector:

  $$C(m) = (Ae^T, m + Ye^T)$$

  .

- To decrypt the message, the receiver performs the following steps:
  1. The receiver multiplies the first part of the encrypted message, Ae^T, by the secret key S getting $SAe^T = Ye^T - Ee^T$.
  2. The receiver subtracts from the second part of the encrypted message, $m + Ye^T$, the result of the previous multiplication.

  Now the receiver finds itself with a vector of the type:

  $$(m + Ye^T) - (Ye^T - Ee^T) = m + Ee^T$$

The algorithm works by exploiting the fact that the product of the matrix E and the vector e of weight t is likely to be zero. This is due to the random positions of the few 1s in both both the rows of E and the vector e; the probability that even just one of these coincides is very low. In the worst case, the code will distance itself from the original with a maximum Hamming distance of t, which is manageable with the error-correcting algorithm of the chosen code.

3.2 Key Generation

The initial step involves generating the matrices A, S, and E for both public and private key generation. The former two matrices, represented by *uint64_t*, are produced utilizing the xoshiro256 PRNG. To leverage this library, we must initialize its seed vector comprising 4 *uint64_t* elements. This initialization can be achieved using the `randombytes(x, X_BYTES)` API provided by librandombytes:

Algorithm 1: Matrix Generation

Result: uint64_t[ ][ ] M
Let X_BYTES be a constant with a value of 8
`// Initialization of the seed`
for $i \leftarrow$ *0* **to** *3* **do**
    seed $\leftarrow$ char[X_BYTES]
    `randombytes(seed, X_BYTES)`
    for $j \leftarrow$ *0* **to** *X_BYTES-1* **do**
        $s[i] = s[i] \ll 8$
        $s[i] \mid = seed[j]$
`// Initialization of the matrix`
$M \leftarrow$ uint64_t[ROWS][COLUMNS]
for $i \leftarrow$ *0* **to** *ROWS-1* **do**
    for $j \leftarrow$ *0* **to** *COLUMNS-1* **do**
        M[i][j] $\leftarrow$ `xoshiro256.next()`
return *M*

Now that we have the function to generate matrices A and S, we need to generate matrix E to compose our public key Y. As described by Alekhnovich, matrix E is composed of weight-t vectors:

Algorithm 2: Matrix E

Result: uint64_t[ ][ ] E //where every row has T 1s
`// Initialize the seed of xoshiro as before`
Declare a uint64_t `matrix`[ROWS][COLUMNS]
for $i \leftarrow$ *0* **to** *ROW-1* **do**
    uint64_t row[COLUMNS] $\leftarrow$ 0
    for $t \leftarrow 1$ **to** T **do**
        repeat
            Generate random position p between 0 and $K - 1$
        until *row[p / size of(uint64_t)] << (p mod size of uint64_t) = 1*
        row[p / size of(uint64_t)] | = (1 << (p mod size of uint64_t))
    `matrix`[i] $\leftarrow$ row;
return *matrix*

After doing so, it is necessary to compute the key Y, which together with the matrix A, represents my public key:

Algorithm 3: Compute Y

```
Data: uint64_t[ ][ ] A, S, E
Result: uint64_t[ ][ ] Y
                                        // To simplify the row column product
Function transposed(M):
    Data: Matrix M
    Result: M transposed
    T ← uint64_t[M.COLUMNS * 64][M.ROWS / 64] filled with 0s
    for i ← 0 to M.ROWS-1 do
        for j ← 0 to M.COLUMNS-1 do
            for k ← 0 to 63 do
                bit ← M[i][j] ≫ k & 1
                M[j * 64 - (63 - k)][i] | = bit ≪ 63 - k
    return M
T ← transposed(A)
                                                    // Bit-wise AND + XOR
Function bax(a, b):
    Data: Two vectors of uint64_t
    Result: The bit-wise AND followed by a XOR
    result_and ← array of size n
    for i ← 0 to n − 1 do
        result_and[i] ← v1[i]&v2[i]
    result_xor ← 0
    for i ← 0 to n − 1 do
        result_xor ← result_xor ⊕ result_and[i]
    result ← 0
    for i ← 0 to 63 do
        bit ← (result_xor >> i) & 1
        result ← result ⊕ bit
    return result
                                                                // Compute Y
Y ← uint64_t[ROWS][COLUMNS]
for i ← 0 to S.ROWS-1 do
    for j ← 0 to T.ROWS-1 do
        bit ← bax(S[i], T[j])
        Y[i][j / size of(uint64_t)] | = bit ≪ (j (mod size of uint64_t))
return Y
```

3.3 Encryption

3.4 Decryption

3.5 Error Code

4 Future Developments

The core concept of public-key cryptography revolves around the ability to make one key publicly accessible, allowing anyone to encrypt outgoing messages. At the same time, only the entity in possession of the private key has the capability to decrypt and restore the original message. This innovative concept greatly simplifies the process of entities reaching a consensus on a private key. It now facilitates secure communication through each other's public keys, even in the absence of prior contact, streamlining the establishment of secure communication.

Quantum computing brings forth the concept of qubits, which are the fundamental units of quantum information, and utilizes the phenomenon of superposition. In superposition, qubits can exist in multiple states simultaneously. However, the act of observation collapses these states, revealing only one state while discarding the others.

A significant breakthrough in this context is the Quantum Fourier transform. When a superposition is periodic, it displays a distinct and identifiable frequency.

Now, regarding RSA versus Quantum Computing: RSA security relies on the discrete logarithm problem, which entails the computational infeasibility of solving a logarithm within modular arithmetic. This problem involves finding the exponent (logarithm) to which a given number (the base) must be raised modulo a prime number (the modulus) to obtain another specified number. This becomes computationally infeasible for classical computers when the values of these variables are chosen to be sufficiently large and random.

While various methods have been known to expedite the breaking of RSA, such as Baby Step Giant Step, these have been impractical due to the considerable time required. However, the use of quantum computers offers a significant advantage when it comes to determining the exponent for the initial 'bad guess,' as it addresses the challenge of finding the period at which the remainders of $g^x = y \pmod{N}$ repeat. A quantum Fourier transform, by observing a random remainder, reveals a repeating pattern or period within the states, leading to a potential breakthrough in solving this problem.

Despite the need for a substantial number of qubits, which is decreasing over time due to advancements in technology, there is an urgent need to develop post-quantum cryptographic solutions to enhance our digital security in response to this evolving threat landscape.