WaveBit
Nonbinary Computation: I Symmetric Cryptography

J. M. Ramírez and Antonio Di Teodoro
Departamento de Matematicas, Colegio de Ciencias e Ingeniería,
Universidad San Francisco de Quito USFQ, Quito 170901, Ecuador

J. F. Landaeta
Max Planck Institute for Chemical Physics of Solids, 01187 Dresden, Germany

E. Contreras
Departamento de Física, Colegio de Ciencias e Ingeniería,
Universidad San Francisco de Quito USFQ, Quito 170901, Ecuador
(Dated: December 11, 2023)

In this paper, we demonstrate the ability to perform symmetric cryptography with regular continuous waves. With no knowledge of the time interval of each component, there no is no way to carry crypt-analysis by simple language statistics or known plain-text attacks. The proof of concept has been implemented in the shape of an application programming interface (API) and can be easily used by several popular programming languages.

I. INTRODUCTION

Cryptography, the science of secure communication, has undergone significant transformations over time. In this journey, two main types of cryptosystems have played pivotal roles: symmetric and asymmetric [1, 2]. This intro presents a chronological overview of cryptography, from classical concepts to contemporary challenges and potential solutions. Symmetric-key algorithms, the pioneers of cryptographic systems, employ a shared secret key for both encryption and decryption [3, 4]. While efficient for bulk encryption, these systems require all parties to possess the secret key. Examples of classical symmetric-key algorithms include DES, 3DES [5], and RC4 [6]. Despite their strength, challenges such as known-plaintext attacks and chosen-plaintext attacks exist. Careful construction of functions in each round mitigates some of these vulnerabilities [7, 8].

The introduction of public-key cryptography by [2] revolutionized the field. Public-key systems enhance security by enabling secure digital signatures and encryption. Quantum cryptography, as explored in "Quantum Cryptography" by [9], promises unbreakable security through the principles of quantum mechanics. In addressing contemporary challenges, the paper explores methods for achieving physical layer security in dual-hop free-space optical/radio frequency (FSO/RF) systems [10].

Modern cryptographic attacks and countermeasures, as discussed in "A New Approach of the Cryptographic Attacks" [4], provide insights into the evolving landscape of cryptographic vulnerabilities. As the digital age presents new challenges, advanced image encryption schemes safeguard visual data from unauthorized access [11]. In brief, cryptography has evolved from its classical roots to address contemporary challenges. The foundational works of Shannon and the paradigm shift brought by public-key cryptography have shaped the field. Emerging innovations in quantum cryptography and physical layer security offer promising solutions. By understanding this historical context and harnessing cutting-edge techniques, the field of cryptography continues to adapt to meet the ever-evolving demands of secure communication and data protection [12–16].

Let's imagine we want to represent the sequence of numbers $0, 1, 2, \dots 7$ (8 numbers). How can we represent them with 0 s and/or 1 s?

With positions, we can. If we have three ($n = 3$) available positions, [$2^{(n=3)} = 8$]:

000	001	010	011	100	101	110	111
0	1	2	3	4	5	6	7

In the realm of computer science and electronics, binary representation is the cornerstone of all digital information processing. This ingenious system relies on the usage of two fundamental symbols: 0 and 1. The magic of binary representation lies in its remarkable versatility, enabling us to express and manipulate a wide range of information, from numbers and characters to intricate data structures. However, have you ever wondered about the origins of this binary code? How did 0 s and 1 s come to symbolize the fundamental building blocks of digital computing?

The story of binary representation takes us back in time to the electronic innovations that emerged during and after World War II (WWII). In those tumultuous years, the foundations of modern computing were being laid, and the use of binary code became pivotal to this technological evolution.

In the early days of electronic computing, the representation of binary numbers was closely tied to the voltages detected within electronic circuits. These circuits operated using different voltage levels to signify distinct states, and this voltage-based representation played a crucial role in shaping the binary system as we know it today.

Consider a simple example: if we have three available positions ($n = 3$), the binary system can represent up to 2^n different values, which in this case amounts to 8 possibilities. These possibilities are represented as 000, 001, 010, 011, 100, 101, 110, and 111. Each of these sequences of 0 s and 1 s corresponds to a unique numerical value, from 0 to 7. This is the classical binary representation of numbers, where each digit—0 or 1—carries a distinct significance. The origins of the 0s and 1s in binary representation are deeply rooted in the electronic developments of the time, particularly during and after WWII. In electronic circuits, the presence or absence of voltage levels plays a decisive role in representing binary digits. A voltage level below a certain threshold, typically 5 volts, was interpreted as a 0, while a voltage level above that threshold was regarded as a 1.

This fundamental concept of using voltage levels to represent binary digits was not just a matter of convenience but an essential breakthrough. It paved the way for the development of digital computers and provided a universal language for communication between electronic components. This binary representation, based on the presence or absence of voltage, became the bedrock of modern computing. In the historical context of this binary revolution, the works of pioneers such as Claude E. Shannon, as exemplified in "A mathematical theory of communication" [1], were instrumental in shaping the theoretical foundations of digital communication and computation. Furthermore, during WWII and its aftermath, the innovations in electronic engineering, epitomized by works such as "New Directions in Cryptography" [16], laid the groundwork for the practical application of binary representations in the emerging field of computer science.

The ability to symbolize complex information with just two simple symbols—0 and 1—was not merely a matter of convenience; it was a pivotal moment in human history. This binary code, rooted in the voltage levels of electronic circuits, became the language of computers, allowing them to process vast amounts of information and change the world as we know it. As we journey through the history of binary representation, we gain a deeper appreciation for the innovations that have shaped our digital landscape, and the story of 0 s and 1 s emerges as a testament to the power of human ingenuity and technological progress.

In this work, we develop a novel method based on waves that can serve as an alternative to the classical binary one. To be more precise, we identify a wave with a specific position of the binary representation. We demonstrate that with just 7 frequencies, we can generate 128 positions for storing each of our characters, equivalent to a 7-bit number. Therefore, in theory, common sounds sampled at 44.1k Hz could store $1e18$ ($2^{44.1\text{k}}$) positions to encrypt any of the characters that make up a string. We explicitly show how to encode/decode the HELLO WORLD string and then generalize the method to encode/decode almost anything: text, images, videos, etc (in a series of papers). It is worth mentioning that the wave encoding we are proposing here, can potentially be used for training Large Language Models (LLMs [17]; i.e, $\sim 1e18$ parameters and simultaneously multimodal) at a cheaper cost in terms of RAM, disk usage and speed of processing (CPU time) using the *WaveBit-Computer* (WBC) we also propose to build in a future research. This work is organized as follows. In section II we introduce the mathematical background needed to formalize our proposal. In section III, we explain how to go from Bits to *WaveBits*. In section IV we develop our example of Hello World and explain the coded/decoded process. In section V we generalized the process to prepare the terrain for more complex objects such as image, videos and music. We present our API, and its simple usage with PYTHON code shared in a GOOGLE COLAB NOTEBOOK. We summarize our work in section VI.

II. MATHEMATICAL ASPECTS

Let $\mathbf{B} = \{0, 1\} \subset \mathbb{N}$ be the binary set and $\Omega \subset \mathbb{N}^k$ be a closed and bounded domain such that, any point $\overline{\lambda}$ in Ω has the form $\overline{\lambda} = (P_1, \ldots, P_k)$, where $k = 1, \ldots, 7$. Let us take

$$P_i = P_i^{sk} = \delta_{sk},$$

where $\delta_{sk} : \mathbf{B} \times \mathbf{B} \to \mathbf{B}$ is the mapping on the Cartesian. We will refer to this domain Ω as the frequency domain.

Example 1. *For $k = 1, \ldots 7$, $\overline{\lambda} \in \Omega$ can be writen as a 1×7 array* $\overline{\lambda} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}_{1\times 7}$.

Let us define $\overline{\lambda}_i$ as the corresponding position i in $\overline{\lambda}$. For example $\overline{\lambda}_3 = 1$

Remark 1. *$P_i = P_i[p]$ each P_i can be dependent on a certain probability p. We will discuss later how to obtain these random values.*

Example 2. *In the case of $p = \frac{1}{2}$, then $P_i[p = \frac{1}{2}] = 1$*

We can construct the following vector using a mapping random generator $Rgen_i : \mathbb{R} \to \mathbb{R}$, $i = 1, \ldots, k$ represents the i-th component of the vector, then it uses a pseudorandom generator of numbers usually similar to the one in popular interpreted language such as MATHEMATICA (or PYTHON) [18]:

$$\overline{Rgen} = \Big(Rgen_1, \ldots, Rgen_k\Big)$$

Using the $\overline{Rgen}$ vector, we construct the following matrix

$$Mcos(2\pi \cdot \overline{Rgen} \cdot t) = \begin{pmatrix} \cos(2\pi \cdot Rgen_1 \cdot t) \\ \cos(2\pi \cdot Rgen_2 \cdot t) \\ \vdots \\ cos(2\pi \cdot Rgen_k \cdot t) \end{pmatrix}_{k \times 1}$$

and the sine matrix function is

$$Msin(2\pi \cdot \overline{Rgen} \cdot t) = \begin{pmatrix} \sin(2\pi \cdot Rgen_1 \cdot t) \\ \sin(2\pi \cdot Rgen_2 \cdot t) \\ \vdots \\ \sin(2\pi \cdot Rgen_k \cdot t) \end{pmatrix}_{k \times 1}$$

$t \in \mathbb{R}$ is the time frequency. With these matrices, we can define the following functions

Definition 1. *$F : \mathbb{R} \to \mathbb{R}$, then for any $x \in \mathbb{R}$ the function F is defined as*

$$\begin{aligned} F(x) = \\ \overline{\lambda} \cdot Mcos(2\pi \cdot \overline{Rgen} \cdot t) + \overline{\beta} \cdot Msin(2\pi \cdot \overline{Rgen} \cdot t) = \\ \sum_{j=1}^{k} \sum_{i=1}^{k} \lambda_i \cos(2\pi \cdot Rgen_i \cdot t) + \beta_j \sin(2\pi \cdot Rgen_j \cdot t) \end{aligned}$$

where $\overline{\lambda}, \overline{\beta} \in \Omega$.
Special case: If $\beta_j = 0$ for all $j = 1, \ldots, k$ then

$$F(x) = \sum_{i=1}^{k} \lambda_i \cos(2\pi \cdot Rgen_i \cdot t) = \tag{1}$$

$$\langle \overline{\lambda}, Mcos(2\pi \cdot \overline{Rgen} \cdot t) \rangle, \tag{2}$$

where $\langle , \rangle$ represent the scalar product in the L_2 norm or the Euclidean norm.

Remark 2. *In a future work, much more complex norms can be considered.*

Example 3. *For $k = 7$, considering $\overline{\lambda}_{1\times 7} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$ and $\overline{Rgen}_{7\times 1} = \begin{pmatrix} 100 & 101 & 102 & 103 & 104 & 105 & 106 \end{pmatrix}$*

$$\begin{aligned} F(x) = \langle \overline{\lambda}, Mcos(2\pi \cdot \overline{Rgen} \cdot t) \rangle = \\ \cos(2\pi \cdot 100 \cdot t) + \cos(2\pi \cdot 102 \cdot t) + \cos(2\pi \cdot 105 \cdot t) + \cos(2\pi \cdot 106 \cdot t) \end{aligned}$$

An important tool we need is the Fourier Transform

Definition 2. *Let $f(x)$ be a complex-valued function defined on the real line $\mathbb{R}$. The Fourier Transform of f, denoted as $\hat{f}(\xi)$, is defined as follows:*

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i\xi x}dx \tag{3}$$

where ξ is a real number.

Definition 3. *Let $\hat{f}(\xi)$ be a complex-valued function defined on the real line $\mathbb{R}$. The Inverse Fourier Transform of $\hat{f}$, denoted as $f(x)$, is defined as follows:*

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi)e^{2\pi i\xi x}d\xi \tag{4}$$

where x is a real number.

These definitions formalize the Fourier Transform and its inverse, providing a clear mathematical basis for their usage in the paper.

Example 4. *Consider the function:*

$$f(x) = \cos(2\pi ax) + \sin(2\pi bx)$$

where a and b are positive real constants. Using the definition of the Fourier Transform, we have:

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} [\cos(2\pi ax) + \sin(2\pi bx)]\, e^{-2\pi i\xi x}\, dx$$

Now, let us calculate this integral by splitting it into two parts, one for the cosine term and one for the sine term. For the cosine term:

$$\int_{-\infty}^{\infty} \cos(2\pi ax)e^{-2\pi i\xi x}\, dx$$

,

and for the sine term:

$$\int_{-\infty}^{\infty} \sin(2\pi bx)e^{-2\pi i\xi x}\, dx$$

After performing these calculations, we obtain expressions for $\hat{f}(\xi)$ in terms of delta functions and imaginary delta functions centered at a and $-b$, respectively.

In contrast, to find the inverse Fourier Transform, we need to apply the inverse Fourier Transform formula to the expressions obtained in step 1.

For the delta functions:

$$f(x) = \int_{-\infty}^{\infty} \left[\frac{1}{2}\left[\delta(\xi - a) + \delta(\xi + a)\right]\right] e^{2\pi i\xi x}\, d\xi$$

Similarly, for the imaginary delta functions:

$$f(x) = \int_{-\infty}^{\infty} \left[\frac{-i}{2}\left[\delta(\xi - b) - \delta(\xi + b)\right]\right] e^{2\pi i\xi x}\, d\xi$$

After performing the calculations, we find that $f(x)$ is a sum of cosine and sine terms with frequencies a and b, respectively.

In the next section we will implement the tools developed here in the context of the *WaveBits* contrunction.

III. FROM BITS TO WAVEBITS

In this section, we develop our *WaveBit* proposal based on the mathematical background introduced previously. Let us consider the wave

$$W_b(t) = \langle \overline{\lambda}, Mcos(2\pi \cdot \overline{Rgen} \cdot t) \rangle = \lambda_1 \cos(2\pi\nu_1 t) + \lambda_2 \cos(2\pi\nu_2 t) + \lambda_3 \cos(2\pi\nu_3 t), \quad (5)$$

where $\overline{Rgen}_{1\times 7} = \begin{pmatrix} \nu_1 & \nu_2 & \nu_3 & \nu_4 & \nu_5 & \nu_6 & \nu_7 \end{pmatrix}$ and $k = 3$. A WaveBit can be constructed with the following algorithm

Algorithm 1: Secret key (or *waveMap* in this document) Algorithm to transform the string into wave functions. The function *randomChoice()* uses similar methods as in [19]

```
Result: Create the secret key (map) and wave function components
for i=1,2,...,$N_{\text{chain}}$ do
    for j=1,2,...,k do
        $mask_{ij} \leftarrow$ randomChoice($\overline{\lambda}_{1\times k}[j]$)[i]
        $\nu$ mask$_{ij} \leftarrow mask_{ij} \cdot \nu_j$
        if $\nu$mask$_{ij}$! = 0 then
            wave(t)$_{ij} \leftarrow \cos(2\pi \cdot \nu\text{mask}_{ij} \cdot t)$
        else
            wave(t)$_{ij} \leftarrow 0$
        end
    end
end
```

Now, based on that algorithm, we can represent **011** in the space of frequencies with no ambiguity. In Fig. 1 panel a), we show the wave function generated for an interval of 0.1 secs. The details and features are clearly visible due to the scale of the time interval. The larger k is, the richer the features (below to be called spectral features) will be the encoding. However, at larger scales such as 1 second, all these features are hidden to the human eye (see Fig. 1, panel b)). The resulting wave is shown in Fig. 1 panel c). Following the same strategy, we can represent **001**, **010**, **011**, and **100** as shown in Fig. 2

IV. 2^7 (128) WAVEBIT: HELLO WORLD!

Now, our purpose is to show how $k = 7$ is able to cover a wide range of information in the form of a WaveBit. First, we will demonstrate the ability of waves to represent characters or letter with the following steps: (1) We define the group of frequencies $\overline{Rgen}_{1\times 7}$. (2) Then we build waves based on the 2^k combinations we are able to construct with the existence or nonexistence of each of them. That gives us 128 possibilities to represent any *and every* character forming any string (if the length of the string is N_{chain} it would take $(N_{\text{chain}})^{2^{128}}$ trials to decipher the string by brute force). With these two steps we start encoding as in what follows.

A. Encoding

Let us choose a canonical string to show the encoding phase of the method. **String = "Hello World!"** with length N_{chain} will do the job. First split **String** to obtain an array of the composed characters

$$\text{CharA} = \{\text{H}, \text{e}, \text{l}, \text{l}, \text{o}, -, \text{W}, \text{o}, \text{r}, \text{l}, \text{d}, !\}.$$

where the "−" is used to represent the space, which also will be converted into a wave. At the core of the algorithm, the set of frequencies is randomly generated (see Algo. 1):

$$mask_{ij} = \text{randomChoice}(128\text{bits}_{[j]})[i]; \quad \nu\text{mask}_{ij} = mask_{ij} \cdot \nu_j \quad (6)$$

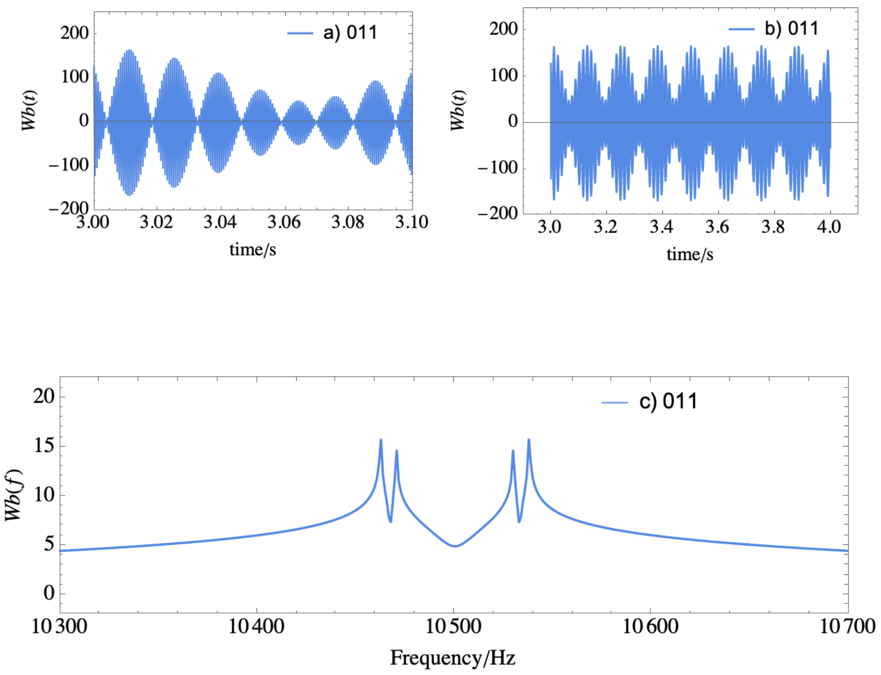

FIG. 1: a) A wave conformed by two frequencies representing one of the 8 possible representations of a number. The 011 wave is close up at only 0.1 secs. b) One full second of the same wave to see how hidden are the details with respect to a). c) The Fourier transform of the wave, both real and imaginary parts.

where the index j is running over each of the characters and the index i is running over the presence or not of the k frequencies.

In Fig. 3, we show how each element in CharA (represented by a wave) as a function of time, presented chronologically, i.e., the “H” first, later the “e”, later the “l” progressively until the last character “!” .

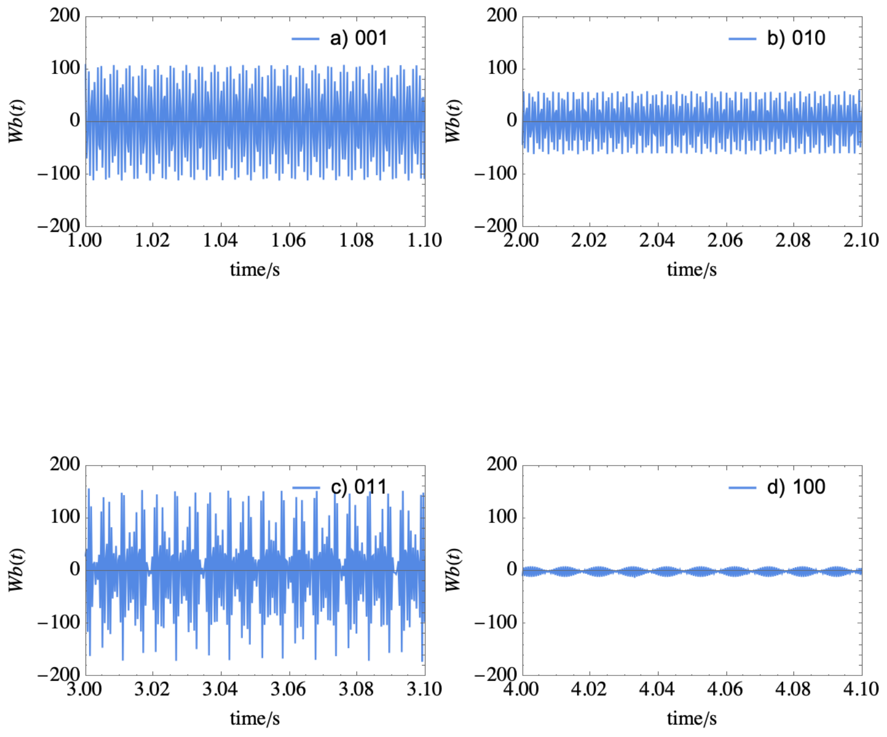

FIG. 2: The diversity in shapes each wave (elapse for 0.1 secs to see details) might have. In our example: a) 001, b) 010, c) 011, d) 100.

We perform the Fourier Transform, namely

$$\hat{W}_b(\nu) = \int_{-\infty}^{\infty} W_b(t) e^{-\mathbf{i}(2\pi\nu)t} dt, \tag{7}$$

of the wave created with Algo.1 by using Algo.2 showed below.

Finally, the encrypted string corresponds to the union operation on the Fourier transform for each of the WaveBits presented in the sequence CharA, namely

$$\bigcup_{i=1}^{N_{\text{chain}}} \text{CharA}_i, \tag{8}$$

as shown in Fig. 4

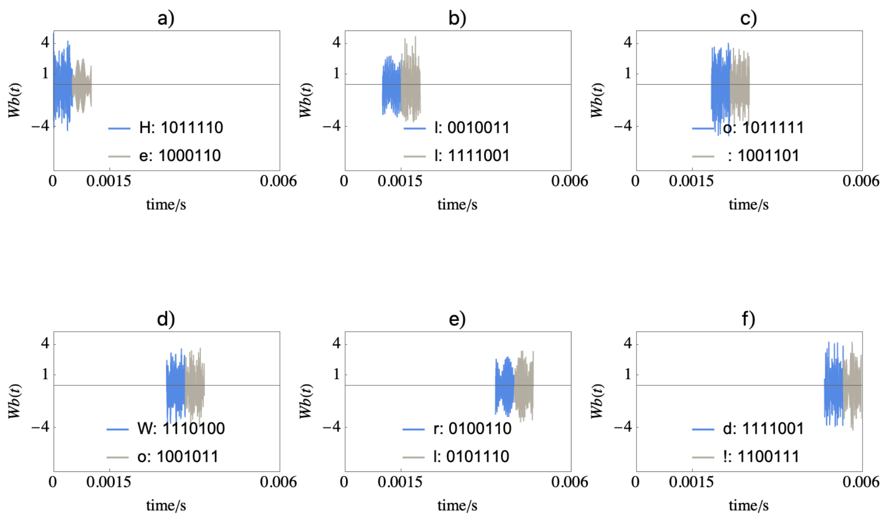

FIG. 3: Here we coded each character in CharA as a *WaveBit* composed of frequencies in the group $\overline{Rgen}_{7\times 1}$, and one of the masks present in the 2^k combination of their presence or not in the wave. In this example, all WaveBits have a duration of 1 sec, but different time intervals for each one are possible in principle.

Algorithm 2: Encrypting message/image/object: Here we replace the letter **i** by **j** for the imaginary part of the Fourier transform to avoid a confusion mixing with the algorithm running index.

```
Result: Encrypting into WaveBit's components

for i=1,2,...,$N_{\text{chain}}$ do
    $CharWave(t)[i] \leftarrow \sum_j^k wave(t)_{ij}$
    $CharNwave(t)[i] \leftarrow CharWave(t)[i]|_{t_0=(i-1)\cdot\Delta t}^{t=i\cdot\Delta t}$
    $CharFourier(\nu)[i] \leftarrow \int e^{-2\pi\nu \mathbf{j} t} CharNwave(t)[i]dt$
end
```

B. Decoding

Algorithm 3: De-encrypting / Decompressing: Here we change the letter **i** by **j** for the imaginary part of the Fourier transform to avoid a confusion mixing with the algorithm running index. The function FindPeaks() uses similar methods as in [20]

```
Result: De-encrypting into Wavebit's components

for i=1,2,...$N_{\text{chain}}$ do
    $CharFourier(\nu)[i] \leftarrow \mathrm{SPLIT}(\hat{W}_b[\nu])|_{t=t_i}^{t=ti+\delta t_i}$
    PEAKmask[i] $\leftarrow \mathrm{FINDpeaks}(CharFourier(\nu)[i])$
    mask[i] $\leftarrow \mathrm{MATCHpeaks}(\mathrm{PEAKmask}[i])$
    $Char_{[i]} \leftarrow \mathrm{Demapping}(\mathrm{mask}[i])$
end
```

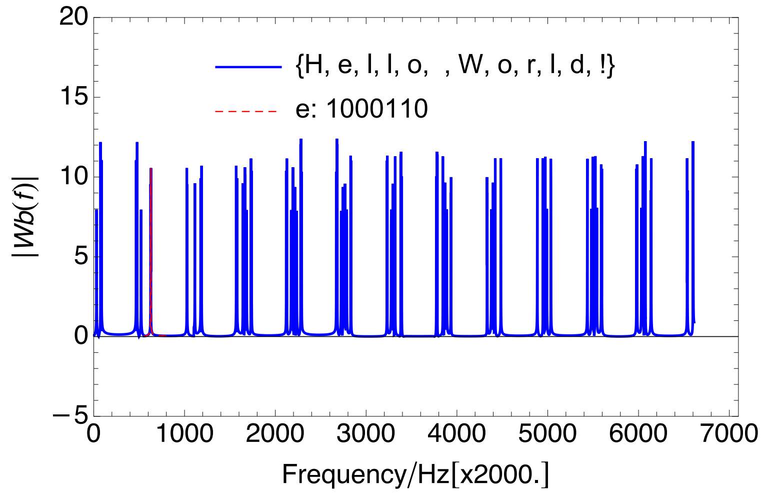

FIG. 4: Unions graphical representation of each of the Fourier transform of all waveBits representing characters in CharA. As a whole they represent the full encrypted string String. The details of the cryptographer lying on the peaks of the plot. The whole sequence of characters is shown in blue. In dashed red is the Fourier transform of the encoded character "e". We plot both the real and imaginary parts of each character in the union operation.

To decode each *WaveBit* into the character back we make use of the previously defined *waveMap* (see **Algo. 1**). The algorithm 3 gives the precise steps we should follow.

It is presumed some components of the encoding process are known. Bellow we show these three elements for the letter 'H'. The two symbols used by $\mathcal{B}$ as express in section 1 are:

$$\mathbf{B}[i] \longleftarrow \{0,1\}[i] \tag{9}$$

with $\{0,1\}$ acting as a mask where 0 (zero) means: that the frequency does not exist and 1 (one) that: it does exist in the WaveBit representing the character or the letter. Let $\odot$ denote pairwise multiplication of vectors, for instance 'H' in CharA is represented as:

$$H_{\text{mask}} = \overline{\lambda}_{1\times 7}[H] \odot \overline{Rgen}_{7\times 1} \tag{10}$$

and $\overline{\lambda}_{1\times 7}[H] = \{0,0,1,1,0,1,0\}$, chosen randomly from the 128 possibilities (there exists no limitation for the characterization, only the produced map is important). Finally we convert to waves using trigonometric functions:

$$Wb_{[H]}(t) = \sum_{i}^{k} \lambda_i f_{\text{trig}}(2\pi \cdot H_{\text{mask}}[i] \cdot t), \tag{11}$$

where $k = 7$ ($\beta_i = 0$ as special case as specified in section 1) in our example, and $f_{\text{trig}}()$ can be any trigonometric function, i.e., $\cos()$, $\sin()$, or a mix of the two. Formally if $H_{\text{mask}}[i] \leftarrow 0$ then $\lambda_i \leftarrow 0$ (see **Algo. 1**).

To go back from the Fourier transform to the character it represents, two major steps are needed:

- To find the peaks and the associated frequencies in the *frequency space*.

- Look for matches with the original frequencies in the *waveMap*.

Whenever we find a match with some tolerance, we assume the existence or nonexistence of the frequency with which we are able to rebuild the mask of the character; i.e, in this example $\overline{\lambda}_{1\times 7}[H] = \{0, 0, 1, 1, 0, 1, 0\}$. In Fig. 5 we show the procedure of finding peaks in the spectra (as a function of frequency) of the string "Hello World!"

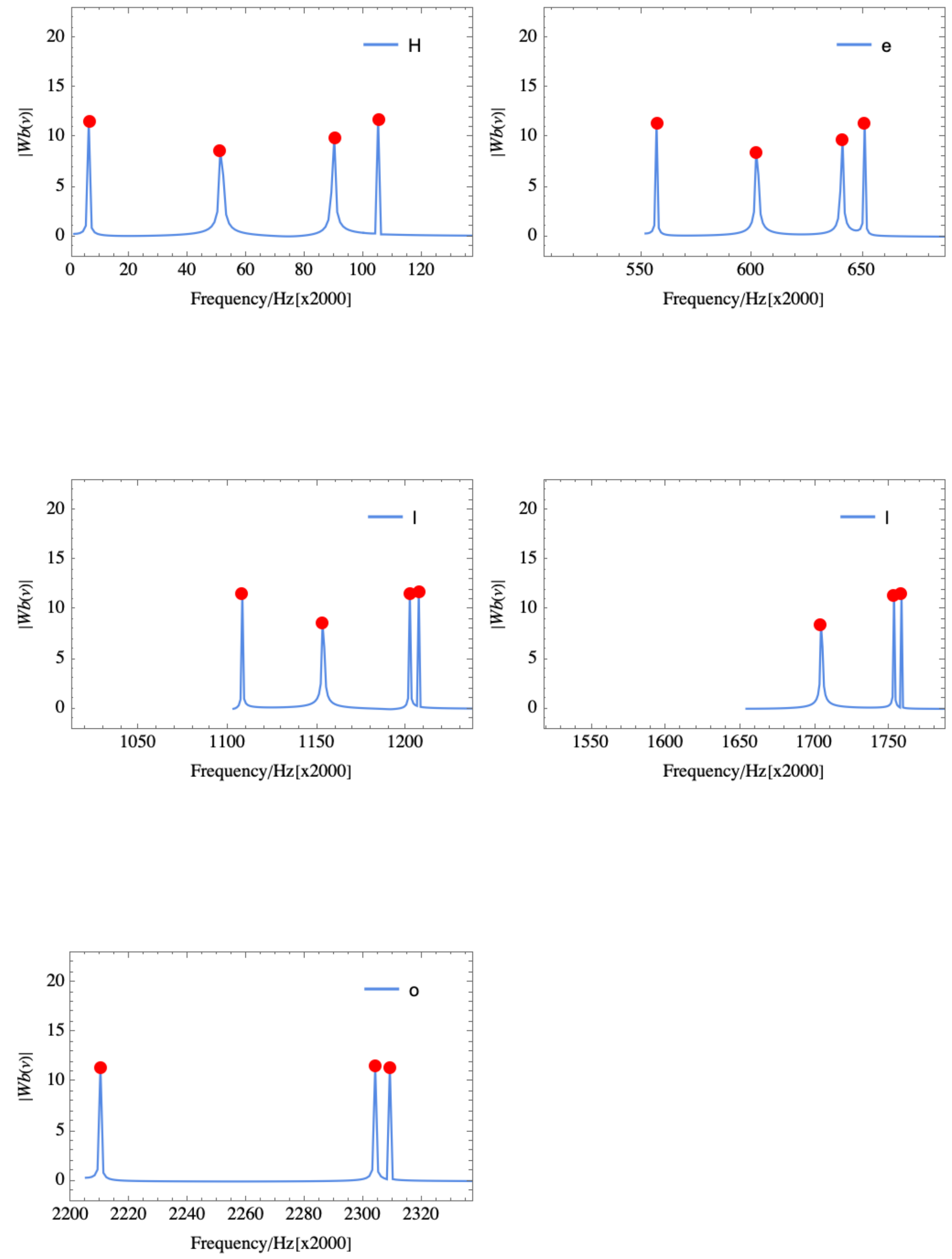

FIG. 5: Finding peaks and demapping. Only shown the word "Hello" so we can visualize the details of how the future detection of the peaks (in red points) in the Fourier transform will look like. The x-axis is the frequencies involved in the wave construction as illustrated in section I.

C. Building back

We find it plausible to convert any string into a wave (Fig. 6). In this section we discuss how to build back each character composing the string.

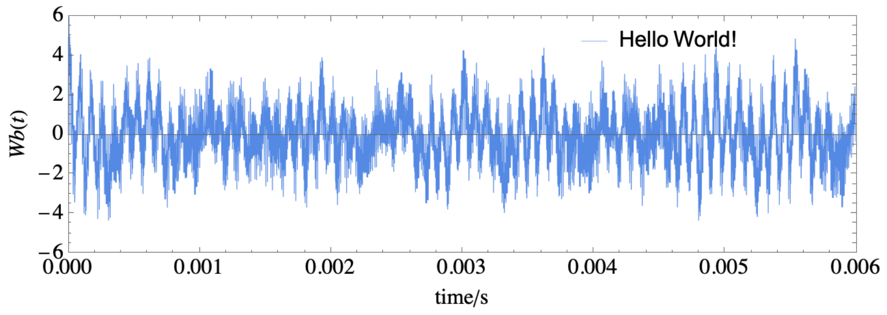

FIG. 6: The complete string “Hello World!” represented as a wave. We are able to produce enough features to “compactify” it within a sound of 6 ms.

Because we have the map, we understand how exactly to decompose the large wave object into the smaller parts. **Algo. 3** shows the algorithm we are using to de-encrypt any wave arriving to our machine:

- We know the beginning and end of each elemental component. We extract that from the time domain wave.
- The second step is to Fourier-transform the chunk of the wave.
- The peaks will mark the frequencies, we detect them and create in reverse the Char location of peaks (ChLoPs).
- Looping the ChLoPs with the secret frequencies, reversing engineering the production of each Char mask.
- Concatenating individual elements to produce the final de-encrypted object.

V. GENERALIZATION

We have demonstrated that with only 7 frequencies we are able to produce 128 positions where each of our characters can be stored, equivalent to a 7-bits number. Therefore, in principle, very common sounds sampled at 44.1k Hz might in principle store 1e18 ($2^{[44.1k]}$) positions to encrypt any of the characters conforming to a string. In the time domain, the wave will represent a sound of 1 second.

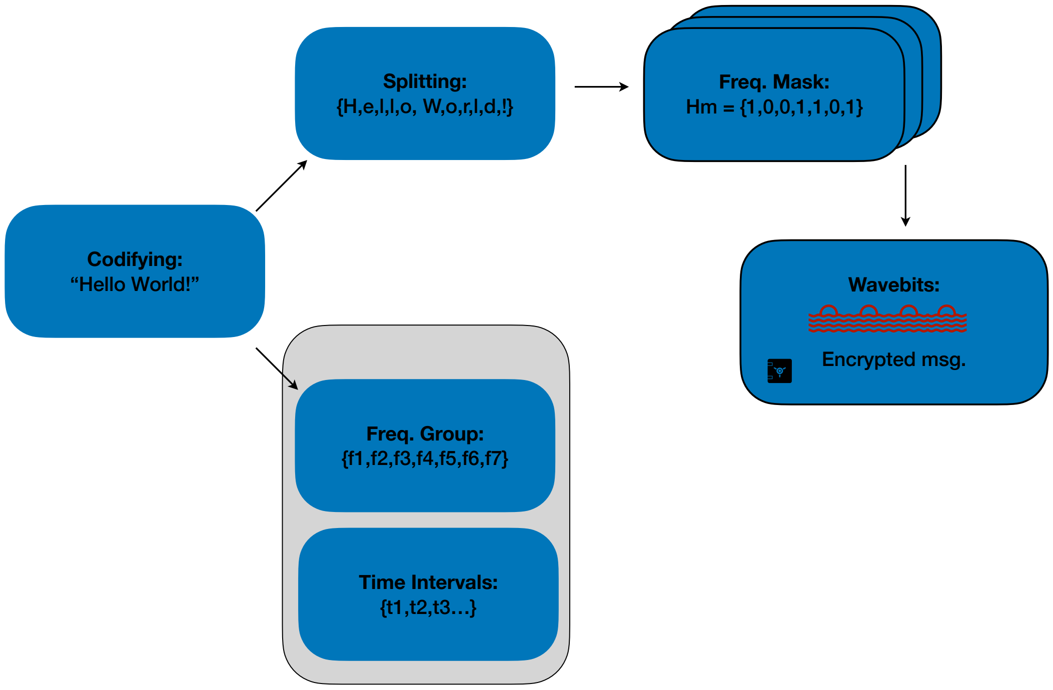

FIG. 7: Encryption phase of objects into pieces of waves. The object is first split into its smaller more fundamental elements. This is followed by a mask procedure from where the creation of a secret key (map) takes place in parallel. Finally the full final object is composed of these smaller pieces and is ready to be transmitted.

The general process to encrypt almost any object: text, images, video, etc, into waves is depicted in Figure 7 with 4 main components:

- Splitting: The map is designed to be de-encrypted into smaller objects. For instance, a whole string can be split into individual characters or we can split into words to form a vocabulary and use it as input for training in machine learning models.
- The application of frequency masks to these objects makes them ready to transform into waves with the auxiliary component of the secret key or map.

- The production of the individual waves after application of the masks and construction of the larger composition of the full object.
- The map serves as a connection between the encryption and de-encryption phases.

VI. USE OF THE API

Here, we share the Google Colab notebook to make use of the REST API: `WaveBit API notebook`
You have to introduce a valid email to receive a valid TOKEN. Changing the `myString` variable you codifies the string. Whoever with the waves generated by the code or with `theSeed` and `theNwavebit` numbers will decode the string and no-one else.

VII. CONCLUSIONS

In this work, we have developed a novel methodology to construct continuous waves that can encrypt almost any object using generalized mathematics which represents a groundbreaking departure from the traditional binary-based data encryption. This novel approach harnesses waveforms and frequencies to not only efficiently represent data but also demonstrate promising advantages across various domains. Firstly, the ability to represent data using a minimal set of frequencies is a significant achievement given that with just 7 frequencies, we can encode 128 *WaveBits* effectively, highlighting the efficiency of this method in terms of data representation. Moreover, the versatility of the methodology shown here is a key feature because it goes beyond text and has the potential to encode a wide array of data, from images to other multimedia content. Undoubtedly, this versatility entails a lot of possibilities for its application in diverse fields. Resource optimization is another remarkable advantage of our proposal as it exhibits the potential to be more efficient than conventional binary-based encryption, manifesting in reduced RAM usage, decreased disk space requirements, and potentially faster processing times. Clearly, these attributes make it a compelling choice for resource-constrained environments. While this article primarily delves into the technical aspects of *WaveBits* encryption, the security implications warrant further exploration because ensuring that encryption can withstand common security threats and attacks is vital for its real-world adoption. In encrypting/decoding, we find a natural $\sim 1e18$ available positions within common 44k Hz sounds that have the potential to serve as Spectral Neural Networks (SNN) as possible candidates to train Large Language Models (LLMS [17]). We will deliver proof of concepts of such ideas in a future work. Further research is required to enhance its security features, validate its robustness, and explore potential optimizations and practical implementations.

VIII. AVAILABILITY OF DATA AND MATERIALS

All data generated or analysed during this study are included in this published article (and its supplementary information files). To be more specific, we delivered the PDF of the code in MATHEMATICA so reviewers can intermediately see the implemented code of the pseudo-code presented without having the actual software. And we also provide the MATHEMATICA notebook code (.nb) so anyone with the software can run the code and see the production of the waves and the decoding methodology. In the section "Use of the API", we add an extra layer of interactivity for the reproduction of the waves, encrypting/decoding but in the shape of an API which connects to the clouds and do the calculations for you with a simple PYTHON interface.

IX. REFERENCES

[1] C. E. Shannon, A mathematical theory of communication, MOCO 10.1145/584091.584093 (2001).

[2] R. L. Rivest, A. Shamir, and L. M. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of The ACM 10.1145/357980.358017 (1978).

[3] C. Shannon, Communication theory of secrecy systems, Bell Syst. Tech. J. 10.1002/J.1538-7305.1949.TB00928.X (1949).

[4] O. Cangea and G. Moise, A new approach of the cryptographic attacks, Communications in computer and information science 10.1007/978-3-642-21984-$9_4$4 (2011).

[5] Ratnadewi, R. P. Adhie, Y. Hutama, A. S. Ahmar, and M. Setiawan, Implementation cryptography data encryption standard (des) and triple data encryption standard (3des) method in communication system based near field communication (nfc) 10.1088/1742-6596/954/1/012009 (2018).

[6] A. Doni, O. A. Maria, and S. Hanif, Implementation of rc4 cryptography algorithm for data file security, Journal of Physics: Conference Series 10.1088/1742-6596/1569/2/022080 (2020).

[7] S. K. Leung-Yan-Cheong and M. Hellman, The gaussian wire-tap channel, IEEE Trans. Inf. Theory 10.1109/TIT.1978.1055917 (1978).

[8] A. Wyner, The wire-tap channel, The Bell System Technical Journal 10.1002/J.1538-7305.1975.TB02040.X (1975).

[9] C. Elliott, Quantum cryptography, IEEE Secur. Priv. 10.1109/MSP.2004.54 (2004).

[10] A. Kumar and P. Garg, Physical layer security for dual-hop fso/rf system using generalized / fading channels, Int. J. Commun. Syst. 10.1002/DAC.3468 (2018).

[11] J. Ma and R. Ye, An image encryption scheme based on hybrid orbit of hyper-chaotic systems 10.5815/IJCNIS.2015.05.04 (2015).

[12] V. Mannalath and A. Pathak, Bounds on semi-device-independent quantum random-number expansion capabilities 10.1103/PHYSREVA.105.022435 (2022).

[13] I. Csiszár and J. Körner, Broadcast channels with confidential messages, IEEE Transactions on Information Theory 10.1109/TIT.1978.1055892 (1978).

[14] S. Pasini, Secure communication using authenticated channels 10.5075/EPFL-THESIS-4452 (2009).

[15] M. Bloch, J. Barros, M. Rodrigues, and S. McLaughlin, Wireless information-theoretic security, IEEE Transactions on Information Theory 10.1109/TIT.2008.921908 (2008).

[16] W. Diffie and M. Hellman, New directions in cryptography, IEEE Transactions on Information Theory 10.1145/3549993.3550007 (1976).

[17] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, B. Yin, and X. Hu, Harnessing the power of llms in practice: A survey on chatgpt and beyond, arXiv.org 10.48550/ARXIV.2304.13712 (2023).

[18] W. R. (2007), Randomreal, https://reference.wolfram.com/language/ref/randomreal.html, Wolfram Language function (2007).

[19] W. R. (2007), Randomchoice, https://reference.wolfram.com/language/ref/randomchoice.html, Wolfram Language function (2007).

[20] W. R. (2014), Findpeaks, https://reference.wolfram.com/language/ref/findpeaks.html, Wolfram Language function (2014).