**Applied Mathematical Sciences, Vol. 16, 2022, no. 4, 215 - 232**
**HIKARI Ltd, www.m-hikari.com**
**https://doi.org/10.12988/ams.2022.916778**

# A Neural Network Contribute to Reverse Cryptographic Processes in Bitcoin Systems: *attention* on SHA256

**Francesco Colasanto**

Universita degli Studi di Firenze
Dipartimento di Matematica e Informatica "Ulisse Dini", Italy

**Luca Grilli**

Universita degli Studi di Foggia
Dipartimento di Economia, Management e Territorio, Italy
Corresponding author

**Domenico Santoro**

Universita degli Studi di Bari Aldo Moro
Dipartimento di Economia e Finanza, Italy

**Giovanni Villani**

Universita degli Studi di Bari Aldo Moro
Dipartimento di Economia e Finanza, Italy



**Abstract**

Bitcoin is a digital currency created in January 2009 following the housing market crash that promises lower transaction fees than traditional online payment mechanisms. Though each bitcoin transaction is recorded in a public log, the names of buyers and sellers are never revealed. While that keeps bitcoin users' transactions private, it also lets them buy or sell anything without easily tracing it back to them. Bitcoin is based on cryptographic evidence, which therefore does not suffer from the weakness present in a model based on trust in guarantee authorities. The use of cryptography is of crucial importance in the

Bitcoin system. In addition to maintaining data secrecy, in the case of Bitcoin, cryptography is used to make it impossible for anyone to spend money from another user's wallet. In our paper, we develop the idea that it is possible to reverse the cryptography process based on hash functions (one-way) through Machine Translation with neural networks. Assuming this hypothesis is true and considering some quantistic algorithms to decrypt certain types of hash functions, we will highlight their effects on the Bitcoin system.

**Mathematics Subject Classification:** 62M45, 91G60, 97R40

**Keywords:** Neural Networks, Hash function, Bitcoin, Quantum computing

# 1 Introduction

Cryptography represents a branch of mathematics that finds application in many sectors and, in a particular way, to the financial intermediation sector that was never considered until a few years ago. This field of mathematics uses techniques to protect information security (see Menezes et al. [22]) and, in particular, privacy, data integrity, authentication, and non-repudiation. The three main types of algorithms used for cryptography are represented by Secret Key Cryptography (SKC), Public Key Cryptography (PKC), and Hash Functions, as shown by Kessler [18]. The first algorithm is based on symmetric encryption, the second one on asymmetric encryption a finally, the last one uses the irreversibly encrypt information. Hash functions are in the spotlight, especially when used for creating digital signatures. This type of function associates a fixed-length digest $h(m)$ to a string of any length $m$, and it verifies three fundamental properties for each hash function: preimage resistance, according to which it is computationally impossible to find a preimage; 2nd-preimage resistance and collision resistance, according to which it is computationally impossible to find two distinct inputs that produce the same digest, as it is witnessed by Vaudenay [39].
This paper aims to show how a hash function can be attacked using neural networks across a particular version of a dictionary attack. In this sense, the hash functions are not to be considered from a mathematical point of view, but the generated output is to be understood as a particular language that, through Machine Translation, can obtain a translation in the reverse direction. A mechanism of this type aims to be a "translator", e.g., SHA256 to English, as is the case for natural language.
The paper structure is the following: in section 2 there are references to the previous main works; in Section 3 we analyze the Neural Machine Translation

(NMT), attention patterns, and the use of transformers; in Section 4 the NMT is applied to a dataset consisting of the hashes of a series of words. Section 5 analyzes how hash encryption is used in the Bitcoin system, what it would entail reversing the encryption process, and some of the main hypotheses of applying quantum computing to decrypt the output of a particular cryptographic function based on elliptic curves. Finally, in section 6, some conclusions are drawn.

## 2 Related Work

Some classic examples of hash functions are MD5 and SHA256. The former stands for "Message-Digest "and it was invented by Rivest [27] which hashes bitstrings onto 128 bits and MerkleDamgård construction to 128 compression functions [23]. This algorithm consists of five phases and 64 operations, as shown by Rachmawati et al. [26], which generate an output unique from other digests. The latter, i.e., SHA256, stands for "Secure Hash Algorithm", and it represents one of the five variants belonging to the SHA family. These hash functions were born in 1993 by the U.S. Department of Commerce and National Institute of Standards and Technology [35] based on the MD5 hash. The first version of the SHA, often identified as SHA-0, exploits the Merkle-Damgård paradigm to 160 compression function and hash onto 160 bits. Also, its compression functions are built upon the Davies-Meyer construction (one-way). The subsequent version always proposed by U.S. Department of Commerce and National Institute of Standards and Technology [36], i.e., SHA-1, still produces a 160-bit digest in output, but, unlike the previous version, it uses a rotation of bits in the preparation of the message (ROTate Left, ROTL). Newer versions [37], i.e., SHA-2, produce a digest whose size is represented in the nomenclature: SHA224, SHA256, SHA384, and SHA512. One of the most popular is SHA256, particularly in digital signature and Bitcoin cryptography. For this purpose, Rachmawati et al. [26] shows the sequence of steps to generate the output of a SHA256. The integrity of these hash functions derives from their ability to be computationally secure to resist attacks of various kinds. The first type of attack used to demonstrate the resistance of the hash function consists in attacking the bitsize [22] which checks if, give two inputs $x_i$ and $x_j$, the hash function produces two equal strings in output, creating collisions. These collisions occur since these functions are used to encrypt strings of any length producing an output of constant size, requiring $2^{n/2}$ operations, where $n$ is the $n$-bit hash function. For instance, the MD5 would take $2^{64}$ operations (complexity) in order to search for collisions exploiting the birthday paradox, as it can be seen by Flajolet et al. [17]. This paradox, developed by von Mises, highlights how the probability of finding in a group two people who have a birthday on the same day approaches one with 50 elements. However, as demonstrated

by Dobbertin [13], it is possible to find collisions after $2^{26}$ computations for MD4 and MD5. So we can state that they are weak hash functions. Similarly, SHA-0 and SHA-1 hash families are vulnerable to collisions. In particular SHA-0 is attackable with complexity $2^{61}$ through a perturbation vector which indicates where local collisions are initiated, as shown by Chabaud and Joux [8]. Moreover, as analyzed by Stéphane and Peyrin [31], a boomerang attack, i.e., a generalization of collision search speed-up techniques, it is required an average time of one hour to find a collision with complexity $2^{33.6}$. On the other hand, about SHA-1, a collision was found across a cloud server system with a complexity equal to $2^{65.1}$ (see Stevens et al. [32]). Finally, no collisions have yet been found for the SHA-2 family. The second type of attack is represented by brute force, as analyzed by Vaudenay [39]. The brute force attack can be applied to any encryption algorithm and is based on exhaustive research, in which all possible keys are tried until the correct one is found; however, this type of attack is computationally impossible as it has in the worst case a complexity equal to $2^n$. A version of brute force attack is the dictionary attack, in which a dictionary is a collection of precomputed words to make the key search quicker. The probability of success in this type of attack is represented by $M/N$, where $M$ is the number of words in the dictionary and $N$ is the number of possible keys. Finally, the third type of attack is represented by the so-called meet-in-the-middle (MITM), as proposed by Merkle and Hellman [24]. This type consists of breaking up the possible key spaces in two blocks to have $N = N' \times N''$, where $N'$ and $N''$ are the numbers of possible keys in each of the newly created blocks. An exhaustive search in these blocks would involve a complexity of $2^{2n}$, and on average, the search has a complexity of $n2^{n/2}$. A new approach to augment cryptographic processes is chaos-based cryptography. For example, Alqarni et al. [1] proposes a technique utilizing the chaotic maps with adaptive symmetry to create chaos-based encryption schemes; or Chenaghlu et al. [9] that develops a new chaotic system to design a secure hash function; or again Ünal Cavusoğlu et al. [34] that develop a chaos-based Random Number Generator (RNG) used as the basis for a new type of RSA encryption (Chaotic RSA - CRSA).

# 3 Neural Machine Translation

Machine Translation (MT) represents a branch of Natural Language Processing (NLP) that studies the translation from one natural language to another. Statistical models (SMT) have been the most used for a long time. Their goal was to search, among all the sentences $e$ of the target language, the one with the highest probability $e^{'} = argmax_e p(e|f)$, where $f$ is a sentence of the source language [42]. However, SMT was highly complex and required particular artifacts to capture certain language phenomena. However, with the help

of neural networks as components of an SMT model, it was possible to have several improvements. For instance, a feedforward neural model to create new translation architectures created by Bengio et al. [5], or consider instead of single words for the target language sets of these by Zamora-Martinez et al. [41], or use neural networks for translation modeling [11], or aggregate relations in sentences [28]. Neural machine translation (NMT), on the other hand, uses neural networks to transform the input sentence into the output one through different architectures (see Bahdanau et al. [3], Stahlberg [30]). The critical feature of NMT is to generate $n$-gram autonomously, i.e. subsequences of a sentence obtained by considering the words that make up the sentence in the $n$-dimensional window (as explained by Manning and Schütze [21]), obtaining better results than previous versions, as shown by Baltescu et al. [4]. The most used architecture in NMT, before the advent of transformers, is represented by an Encoder-Decoder network known as Sequence-to-Sequence (Seq-2-Seq) developed by Sutskever et al. [33], used for the first time in the translation from English to French. This architecture consists of two Recurrent Neural Networks (RNN) which perform the functions of Encoder and Decoder, respectively: the first acquires the sentences as input and encodes them into a fixed-size context vector, while the second uses the context vector to generate the output. Generally, Encoders and Decoders use LSTM (Long-Short Term Memory) cells; in particular, in the Encoder, the input is reproduced inversely so that the last word read by a cell corresponds to the first word of the connected cell constituting the context vector while, in the Decoder, from the context vector the information is passed through the different layers of the network one after the other. An estimator to predict the accuracy of the words produced is the one introduced by Luong et al. [19] through a feature selection strategy.

## 3.1 Attention

A problem highlighted by Cho et al. [10] is that these RNNs, due to fixed-length encoding, cannot understand complex syntactic structures in the case of translation of long sentences. A solution, proposed by Bahdanau et al. [3], is the introduction of the concept of *attention* that allows to obtain a context vector based on the hidden states in such a way as to have no longer a single vector $c(\mathbf{x})$, but a series of vectors $c_i(\mathbf{x})$ at each time interval $i$:

$$c_i(\mathbf{x}) = \sum_{j=1}^{N} \alpha_{i,j} h_j, \tag{1}$$

where $\alpha_{i,j}$ represents the amount of attention corresponding to the output $h_j$. In Bahdanau et al. [3], the attention score $\alpha_{i,j}$ is determined as:

$$\alpha_{i,j} = Softmax(v^\top \tanh(W_1 h_i + W_2 s_j)), \tag{2}$$

where $v, W_1, W_2$ are parameters and $h_i, s_j$ are the hidden states.
A second kind of attention is that introduced by Luong et al. [20], often referred to as dot-product attention, in which the attention score is calculated as:

$$\alpha_{i,j} = Softmax(h_i^\top W s_j). \tag{3}$$

The fundamental difference between these two types of attention is that the dot-product attention is much faster and more space-efficient.
Another type of attention, which has now become fundamental, is that introduced by Vaswani et al. [38], also named multi-head attention. This model consists of a sequence of attention transformer blocks, each of which is made up of two sublayers, i.e., multi-head attention and feedforward network so that the output of the multi-head attention is the concatenation of the outputs of each attention head (see Stahlberg [30]). Using the notation introduced by Vaswani et al. [38], it results:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{4}$$

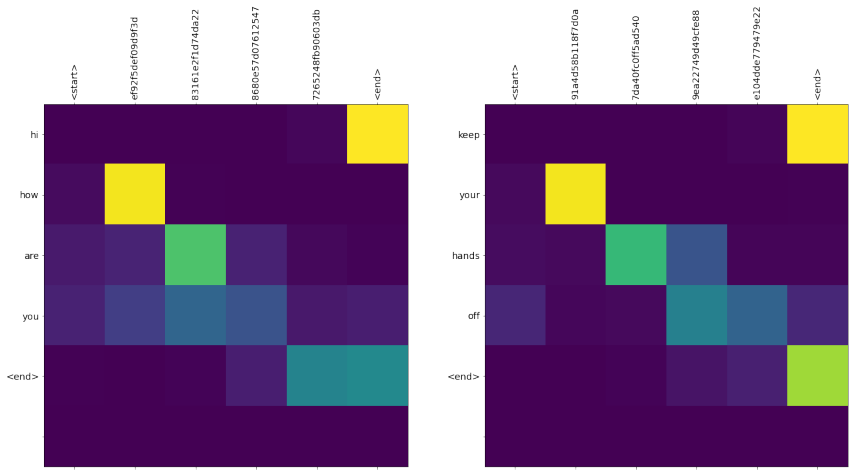
where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$, $W^O$ is a weight matrix, and $Q, K, V$ are sets of queries, keys, and values respectively packed into matrices. This kind of attention has led to the birth of a completely attention-based architecture called **transformers**, where the dependencies between inputs and outputs avoid the use of feedforward networks.

# 4 *Attention* on SHA256

Our paper aims to consider the output of hash functions as strings belonging to a specific language. From a certain point of view, it is possible to consider the characters of these strings as characters of a natural language based on a Latin or ideogrammatic alphabet. In particular, if we consider an alphabet based on ideograms such as Chinese or Japanese, each corresponds to a series of notions, just as a series of ideograms joined together could correspond to a notion that is still different from the personal meaning. Let us consider the hash function SHA256, which is very widespread in the Blockchain and the Bitcoin system applications. An output of this hash is a string consisting of 64 characters, corresponding to a 256-bit digest, belonging to an alphabet $\Gamma = \{A, ..., Z, a, ...z, 0, ...9\}$. The idea that we want to graft is how a specific natural language string can find a corresponding "translation" in an $n$-gram of characters of the hash output. We know that this type of hash allows us to encode strings of any size in the output of 64 characters but, assuming the previous hypothesis as true, we could then set ourselves the goal of determining which $n$-gram best allows us to get a reverse translation from SHA256 to a natural language. From now we will only consider English. To highlight this

core idea, we will use a Seq-2-Seq neural network with Tensorflow framework[1], that uses the attention equations of Bahdanau et al. [3] and whose code was implemented through Google Colab. The dataset used in this case is based on the English-Italian dictionary[2] implemented with a list of movie titles to which several changes have been made. Firstly, since the dataset contained a series of sentences with the corresponding translation into Italian, these have been eliminated, leaving only the English sentences. Secondly, we created a script (Python) that associates the corresponding SHA256 hash to each of the remaining sentences, respecting the number of words that made up the sentence. The key feature is that the dataset thus created contains sentences of different sizes, from a single word up to concatenations of more than 20 of these. In this dataset, cleaned of special characters, an association was created between each word constituting a sentence and an ID. Also, in Seq-2-Seq, which uses GRU (Gated Recurrent Unit) cells, training was limited to 70,000 sentences to speed up the process. The output of this model is represented by the translation of a sentence and a heat map that highlights how the attention mechanism has assigned the weights to the different inputs. Since the goal is to determine which $n$-gram is best for improving attention in translation from SHA256, several datasets were created based on the one described above, where the hash was "split"to highlight the resulting attention and, consequently, Seq-2-Seq has been trained several times with different results.
The hash was split into 4 substrings of 16 characters each in the first case. In several cases, the translation was completely successful while, in others, it was partial. In Figure 1, the heat maps of some sentences are represented. In particular, Figure 1(a) corresponds to a correct translation. In contrast, in Figure 1(b), the translation is incomplete as the original sentence was "keep your hands off me", and the resulting translation is "keep your hands off". Obviously, in this and the following cases, the assignment of the weights in the attention process highlighted in the heat maps is not completely precise. The words used to test the network are the same as those present in the dataset. However, using other sentences never seen, in some cases, Seq-2-Seq could return translations of only one portion of the sentence, which indicates how the network can learn. In the second case, the hash was split into 8 substrings of 8 characters each, whose heat maps are represented in Figure 2. In this case, Figure 2(a) highlights a correct translation of the hash while, in Figure 2(b), the translation is completely wrong since the original sentence was "blog vhs". The network translated into "turbo mary", words still present in the dataset. However, the fundamental problem here is the distribution of attention, as evidenced in the heat maps. In the third case, the hash was split into 16 substrings of 4 characters each, whose heat maps are represented in Figure 3.

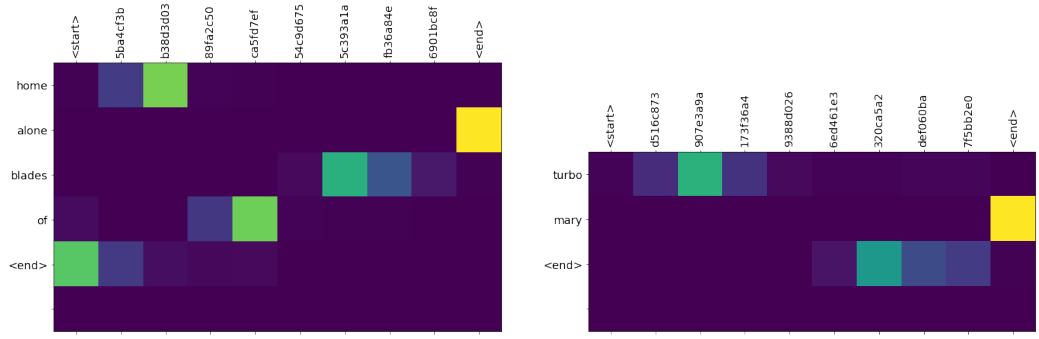
[1] `https://www.tensorflow.org/tutorials/text/nmt_with_attention`
[2] `http://www.manythings.org/anki/`




(a) Corresponding to "hi how are you"

(b) Corresponding to "keep your hands off me"

Figure 1: SHA256 hash split into 4 substrings

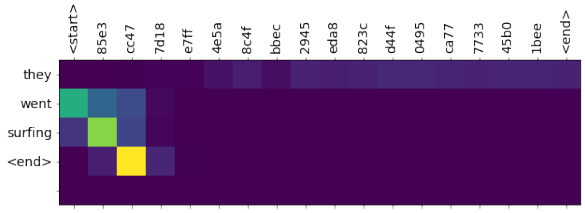



(a) Corresponding to "home alone blades of"

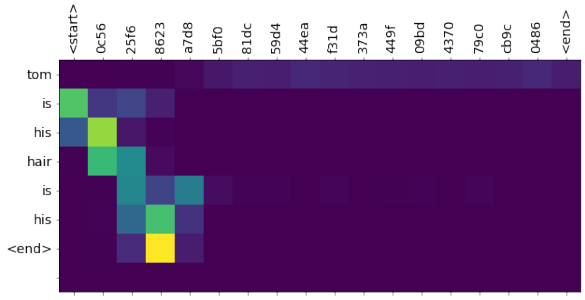
(b) Corresponding to "blog vhs"

Figure 2: SHA256 hash split into 8 substrings

As in the previous case, Figure 3(a) relates to a successful translation while 3(b) is wrong as the original sentence was "headhunter", and the translation was "tom is his hair is his". However, splitting the hash into 16 substrings does not seem convincing since the attention is almost entirely derived only from the first 3, as evidenced by the heat maps.
In the latter case, the hash was split into 32 substrings of 2 characters each, whose heat maps are represented in Figure 4. In this case, the translation in both cases turns out to be wrong. In particular, the Figure 4(a) refers to the sentence "eat it", and produced as a translation "tom is a good father", while, Figure 4(b), refers to "eat up"and produced as a translation "tom is getting angrier". Furthermore, this type of split does not produce good results in terms

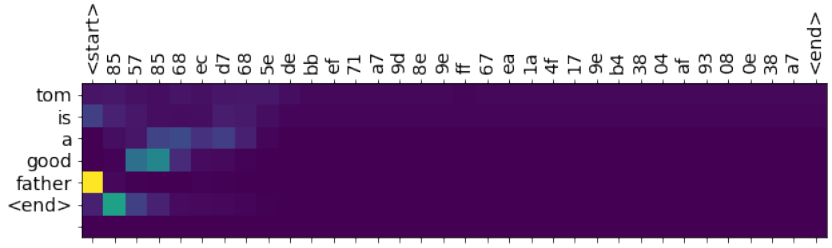



(a) Corresponding to "they went surfing"

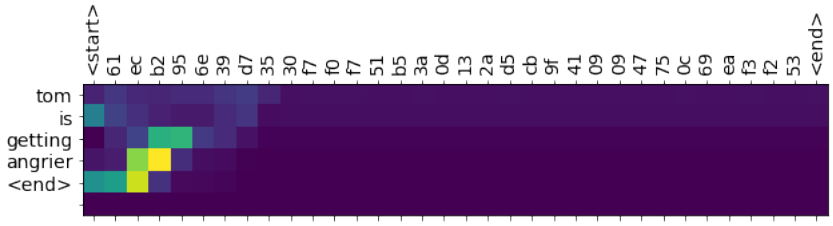



(b) Corresponding to "headhunter"

Figure 3: SHA256 hash split into 16 substrings




(a) Corresponding to "eat it"

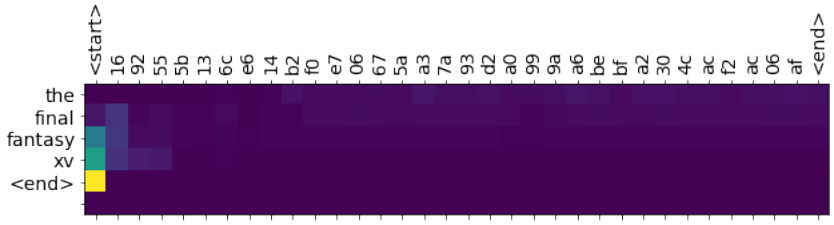



(b) Corresponding to "eat up"

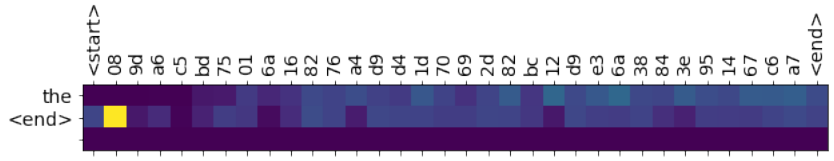
Figure 4: SHA256 hash split into 32 substrings

of attention since, according to the heat maps, only the first 7 substrings, on average, are responsible for the translation. A particular situation occurred in this hypothesis: the Seq-2-Seq, while continuing to have the wrong translation, managed to translate hashes that were not present in the dataset, returning a combination of words not yet present in the training dataset that order.

Figure 5(a), whose hash to translate corresponds to the sentence “walle the”




(a) Corresponding to ”walle the“




(b) Corresponding to ”the final fantasy xv“

Figure 5: Particular translation not present in the training set

gets as a translation “the final fantasy xv.” The hash corresponding to this last sentence (not present in the dataset with the word “the”) obtains the single word as translation “the”. Unfortunately, the network translations are not always accurate, deriving from a training set consisting of only 150,000 lines. Hence, exponentially increasing the dataset size could improve the translations performed, while testing new types of hash splits would improve attention.

# 5 Application to Bitcoin

Bitcoin is a cryptocurrency created by Nakamoto [25] that uses the digital signature mechanism for the transfer of money from one subject to another and uses the Blockchain as a register in which to record all transactions. Cryptography in the entire system plays a fundamental role as it is used to solve the Proof-of-Work problem to verify a transaction and create the bitcoin address, which we will consider in this paper to evaluate the consequences.
A Bitcoin address [2] is a string consisting of 34 hexadecimal characters that allow us to exchange money between different wallets. Therefore, it is a public address, visible to all, necessary to carry out transactions. This address, however, represents the terminal process of a series of cryptographic steps based on public and private keys. The first step is to create a private key used to “sign” the authorization to execute a transaction. This key is based on a random number between 1 and $2^{256}$ (Bitcoin generates a 256-bit number) defined according to the elliptic curve used by the system [2]. In formal terms, the random number is transformed into a 256-bit number via the hash function

SHA256, and if the number is less than $n - 1$, where $n = 1.158 \cdot 10^{77}$, then it is usable as a private key. In this case, the elliptic curve, also used in the next stage of generating the public key, is **secp256k1**. This curve, based on the discrete logarithm problem, produces an elliptic function on the field $\mathbb{Z}_p$, where $p = 2^{256} - 2^{32} - 977$ is a Fermat prime, given by $y^2 = x^3 + 7$ (representation on $\mathbb{R}$). A point $P$ on the elliptic curve is generally represented by a pair of coordinates $(x, y)$ on a more visible field, such as $\mathbb{R}$. A common way to generate a private key is to exploit a single seed, and, in this case, the wallet is called deterministic or "seeded". The characteristic of these wallets is that the seed used to generate the key is represented by a sequence of words (Mnemonic Code Words) that uniquely generates a specific seed. The Mnemonic Words are generated through a process indicated as BIP-39, which starts at a random sequence of 128/256-bit determines the hash SHA256, which will be considered a checksum in which the first bits are added to the random sequence. This sequence is split into several sections, and each section is mapped through a predetermined 2048 word dictionary [2]. At this point, the words obtained are concatenated, possibly with the addition of a passphrase, and transformed using the HMAC-SHA512 algorithm, which is based on SHA512, obtaining a 512-bit output that represents the seed. The left 256-bits of this seed represent the master private key, denoted as $k$. This private key thus determined can be used to construct the public key by multiplying it by a point called *generator point* ($G$) belonging to secp256k1 so that the public key thus generated is still a point of the elliptic curve [2]:

$$K = k \cdot G \tag{5}$$

The new public key determined is a starting point to apply other cryptographic functions to obtain the Bitcoin address. From $K$, the "Double Hash" (or HASH160) is computed, which is based on the computation of the RIPEMD160 of $K$'s SHA256:

$$A = RIPEMD160(SHA256(K)) \tag{6}$$

where $A$ is the 160-bit Public Key Hash. RIPEMD160, created by Dobbertin et al. [14], is considered the European antagonist of the MD5 hash and is an evolution of the RIPEMD128, which was vulnerable to collisions. The RIPEMD160 generates 160-bit output, and although there are later versions, no collisions have yet been found. The hash $A$ obtained undergoes a last cryptographic process before becoming a Bitcoin address. In particular, it is encoded through the Base58check, introduced in this system to make huge numbers more visible through an alphanumeric representation. Base58 is an evolution of Base64 in which several characters that could create ambiguities, such as 0 and $O$, have been eliminated. In the case of the Bitcoin system, a specific prefix is applied to the data to be converted, identified by $0x00$. A checksum, concatenated at the end, deriving from the application of the

"double-SHA" (or $SHA^2$) of which we consider only the first 4 bytes [2]. The result of all this process is the Bitcoin address.
In summarizing, we can state that starting from a series of words making up the wallet, it is possible to generate a specific hash that represents the private key; from this private key, it is possible to generate a hash that represents the public key and finally from the latter it is possible to generate an encoded hash that represents the Bitcoin address. Thanks to the different cryptographic functions used, all these steps make the process one-way due to the computational complexity. Remembering that the only value visible from the outside to anyone in the Bitcoin address, if we suppose that the process described in the previous section is true and therefore it is possible through neural networks to "decrypt" the SHA256 hash and if the algorithm to solve the problem of the discrete logarithm on elliptic curves (secp256k1) could be implemented, then at the address it would be possible to go back to the private key and "sign" the various transactions as if the owner of the wallet carried them out. Furthermore, once the private key has been obtained, it would still be possible to go back to the Mnemonic Words and therefore access the wallet of other subjects.
The opportunity of "wallet theft" could have devastating effects on the global economic system, not only because the ability of cryptographic algorithms to encrypt information would fail but above all because the trust that investors have in cryptocurrencies could fall. Since today cryptocurrencies are starting to represent an essential part of the assets of large investors, a collapse in the value of Bitcoins could mean burning billions of dollars. However, the ability to access other people's wallets could be helpful in some cases: since without Mnemonic Words, it is impossible to access your wallet, if the owner were to die, it would not be for the heirs to be able to access any inheritance. In this situation, reversing the cryptographic process would be helpful.

## 5.1 Quantum Computing

An open question, however, is the problem of the discrete logarithm on elliptic curves due to its computational complexity. Since it's not possible to solve it with current tools, algorithms that exploit Quantum Computing have been hypothesized. In the 1980s, physicist Feynman [15] completely revolutionized the idea of automatic computing, providing some pioneering [16] ideas that opened new horizons towards the new science defined today as Quantum Computing. Feynman's idea was to bring the phenomena of the "new" physics (quantum mechanics) into the world of classical computer science to exponentially increase the computational capacity of classical computers. Many articles followed his path, in which an automatic system based on quantum formalism is rigorously formulated ([6], [12], [7]) such as Quantistic Touring Machine

(QTM), which is the quantum evolution of the classical Turing Machine. As a result, a new computational complexity theory based on the new quantum formalism emerges in these articles, where new complexity classes of problems are introduced. The goal of using quantum computers is to demonstrate that a given problem is in the BPP (Bounded-error Probabilistic Polynomial time) class. Previous models of quantum computers are fundamentally based on two "phenomena" belonging to the quantum formalism: the *superposition principle* and *entanglement*. From the mathematical formalism of quantum mechanics [40] we know that the natural framework of a quantum system is a complex Hilbert space $\mathcal{H}$ and that the temporal evolution of states is marked by an ordinary differential equation on Hilbert spaces known today as the Schrödinger equation. In the specific case of quantum computer science, the fundamental quantistic system is the *Qubit* that is the quantistic translation of the classical concept of *Bit*. Thus, being the qubit a quantistic system, we know that the possible states of the qubit live in a complex Hilbert space. Then, the natural question is: "Which Hilbert space represents the possible physical states of our qubit?". The answer to this question is strictly connected to the possible measurement that we affect on our qubit. In particular, because the qubit has to be a generalization of a classical bit, every time we effect measurement on qubit, we can obtain only two possible values, which are 0 and 1. So in the case of qubit physical system, we have to use a Hilbert complex space of dimension 2, in which we can find two linearly independent states $|0\rangle$ and $|1\rangle$, that respectively represent the qubit state in which the probability of measuring 0 is 1, and the qubit state in which the probability of measuring 1 is 1. Thus if $|\psi\rangle$ is a general state of our qubit, we can say that exists $z_0, z_1 \in \mathbb{C}$ such that

$$|\psi\rangle = z_0|0\rangle + z_1|1\rangle \tag{7}$$

that is the mathematical formulation of the superposition principle. Moreover, we have to say that the physical interpretation of the complex scalars $z_0, z_1$ is strictly connected to the probability of measuring respectively 0 and 1. Having described a physical system of a single qubit, to make the most of the quantum effects on automatic computing, it is necessary to consider physical systems formed by several qubits (more qubits correspond to an increase in computing power). To consider more qubits, we need to exploit the quantum phenomenon of *entanglement*, which consists in putting the states of each qubit in a condition of strong correlation. This situation is mathematically formulated through the tensorial product of Hilbert spaces; in particular, having $n$ qubits whose respective Hilbert spaces are $\mathcal{H}_1, ..., \mathcal{H}_n$ then the Hilbert space that will represent the total physical system (in entangled) will be

$$\mathcal{H} = \mathcal{H}_1 \otimes ... \otimes \mathcal{H}_n. \tag{8}$$

Thus, the Hilbert dimension of $\mathcal{H}$ (by the property of tensorial product) is $2^n$. The exponential growth of the size of $\mathcal{H}$ as the number of qubits varies

clear that thanks to the *superposition principle*, through the use of quantum logic networks (like Hadamard gate) and *oracles* (physical representation of the unitary operator that allows the passage from one state to another) problems that with classical computations required execution times with exponential complexity, with the quantum approach admit high probability solutions with polynomial complexity.
So, returning to the problem of reversing the cryptographic process, combining what was said in the previous sections for decrypting the SHA256 outputs with the Shor [29] algorithm (which demonstrates that the discrete logarithm is in BPP, therefore solvable with a high level of probability in polynomial time) we could reverse the one-way process that generates Bitcoin addresses.

# 6 Conclusions

Our core idea in this paper is to try to use the ability of neural networks to "reverse" the cryptographic process, especially of SHA256. All this can be done by considering the output of the hash functions as a string belonging to a specific language. It is possible to obtain the links between different $n$-grams of the string and consequently obtain a translation into a natural language through attention mechanisms. In particular, in this model, the Bahdanau's attention was used through a Seq-2-Seq architecture, trying to understand what could be the best split to apply to the characters of the output string. Furthermore, we analyzed how cryptography intervenes in the Bitcoin system, what consequences this method's application could have, and the use of quantum algorithms if the cryptographic process is reversed. In this sense, the translation can be compared to a dictionary attack created by the neural network.
In future works, it is possible to consider, instead of Seq-2-Seq, the architecture of the transformers and increase the size of the dataset exponentially to see what improvements it can bring. In addition, the text generation technique could be used to create a dictionary based on the words present in the dataset of this technique, which can be used to expand the translation.

# References

[1] M. Alqarni, E. E. Mahmoud, M. Abdel-Aty, K. M. Abualnaja, P. Trikha, and L. S. Jahanzaib. Fractional chaotic cryptovirology in blockchain - analysis and control. *Chaos, Solitons & Fractals*, 148:110989, 2021. doi: 10.1016/j.chaos.2021.110989.

[2] A. Antonopoulos. *Mastering Bitcoin: Programming the Open Blockchain:*

*Unlocking Digital Cryptocurrencies*. O'Reilly & Associates Inc, US, 2017. ISBN: 978-1491954386.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - San Diego, United States*, 2015.

[4] P. Baltescu, P. Blunsom, and H. Hoang. Oxlm: A neural language modelling framework for machine translation. *Prague Bulletin of Mathematical Linguistics*, 102:81–92, 2014.

[5] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[6] P. Benioff. Quantum mechanical hamiltonian models of turing machines. *Journal of Statistical Physics*, 29:515546, 1982. doi: 10.1007/BF01342185.

[7] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1993. doi: 10.1137/S0097539796300921.

[8] F. Chabaud and A. Joux. Differential collisions in sha-0. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 56–71, DE, 1998. Springer Berlin Heidelberg.

[9] M. A. Chenaghlu, S. Jamali, and N. N. Khasmakhi. A novel keyed parallel hashing scheme based on a new chaotic system. *Chaos, Solitons & Fractals*, 87:216–225, 2016. doi: https://doi.org/10.1016/j.chaos.2016.04.007.

[10] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, US, 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012.

[11] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoderdecoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[12] D. Deutsch. Quantum theory, the churchturing principle and the universal quantum computer. *Proceedings of The Royal Society A: Mathematical, Physical*, 400:97–117, 1985. doi: 10.1098/rspa.1985.0070.

[13] H. Dobbertin. Cryptanalysis of md4. *Journal of Cryptology*, 11:253271, 1998. doi: 10.1007/s001459900047.

[14] H. Dobbertin, A. Bosselaers, and B. Preneel. Ripemd-160: A strengthened version of ripemd. In D. Gollmann, editor, *Fast Software Encryption*, pages 71–82, UK, 1996. Springer Berlin Heidelberg.

[15] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467488, 1982. doi: 10.1007/BF02650179.

[16] R. P. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16:507531, 1986. doi: 10.1007/BF01886518.

[17] P. Flajolet, D. Gardy, and L. Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics*, 39(3):207–229, 1992. doi: 10.1016/0166-218X(92)90177-C.

[18] G. C. Kessler. An overview of cryptography. 2017. Available at `http://www.garykessler.net/library/crypto.html`.

[19] N.-Q. Luong, L.Besacier, and B.Lecouteux. Towards accurate predictors of word quality for machine translation: Lessons learned on frenchenglish and englishspanish systems. *Data & Knowledge Engineering*, 96-97:32–42, 2015. doi: 10.1016/j.datak.2015.04.003.

[20] T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, US, 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166.

[21] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, US, 1999. ISBN: 0-262-13360-1.

[22] A. J. Menezes, P. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, US, 1996. ISBN: 0-8493-8523-7.

[23] R. C. Merkle. One way hash functions and des. In G. Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 428–446, US, 1990. Springer New York.

[24] R. C. Merkle and M. E. Hellman. On the security of multiple encryption. *Association for Computing Machinery*, 24(7), 1981. doi: 10.1145/358699.358718.

[25] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Bitcoin.org*, 2008.

[26] D. Rachmawati, J. T. Tarigan, and A. B. C. Ginting. A comparative study of message digest 5(MD5) and SHA256 algorithm. *Journal of Physics: Conference Series*, 978:012116, 2018. doi: 10.1088/1742-6596/978/1/012116.

[27] R. L. Rivest. Rfc1321: The md5 message-digest algorithm. 1992.

[28] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, US, 2017. Curran Associates, Inc.

[29] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM*, 43(2):303–332, 1996. doi: 10.1137/S0036144598347011.

[30] F. Stahlberg. Neural machine translation: A review and survey. *arXiv:1912.02047v2*, 2020.

[31] M. Stéphane and T. Peyrin. Collisions on sha-0 in one hour. In K. Nyberg, editor, *Fast Software Encryption*, pages 16–35, DE, 2008. Springer Berlin Heidelberg.

[32] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov. The first collision for full sha-1. In J. Katz and H. Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 570–596, US, 2017. Springer International Publishing.

[33] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[34] Ünal Cavusoğlu, A. Akgül, A. Zengin, and I. Pehlivan. The design and implementation of hybrid rsa algorithm using a novel chaos based rng. *Chaos, Solitons & Fractals*, 104:655–667, 2017. doi: https://doi.org/10.1016/j.chaos.2017.09.025.

[35] U.S. Department of Commerce and National Institute of Standards and Technology. Secure hash standard - shs: Federal information processing standards publication 180. 1993.

[36] U.S. Department of Commerce and National Institute of Standards and Technology. Secure hash standard - shs: Federal information processing standards publication 180-1. 1995.

[37] U.S. Department of Commerce and National Institute of Standards and Technology. Secure hash standard - shs: Federal information processing standards publication 180-2. 2005.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[39] S. Vaudenay. *A Classical Introduction to Cryptography: Applications for Communications Security*. Springer, US, 2005. ISBN: 978-0387254647.

[40] J. von Neumann. Mathematische begründung der quantenmechanik. *Mathematisch-Physikalische Klasse*, pages 1–57, 1927.

[41] F. Zamora-Martinez, M. J. Castro-Bleda, and H. Schwenk. N-gram-based machine translation enhanced with neural networks for the french-english btec-iwslt10 task. *International Workshop on Spoken Language Translation (IWSLT) 2010*, pages 45–52, 2010.

[42] J. Zhang and C. Zong. Deep neural networks in machine translation: An overview. *IEEE Intelligent Systems*, 30(5):16–25, 2015. doi: 10.1109/MIS.2015.69.

**Received: May 27, 2022; Published: June 14, 2022**