sensors

MDPI

Article

iTrust—A Trustworthy and Efficient Mapping Scheme in Elliptic Curve Cryptography

Hisham Almajed, Ahmad Almogren * and Mohammed Alabdulkareem

Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11633, Saudi Arabia; 438105079@student.ksu.edu.sa (H.A.); kareem@KSU.EDU.SA (M.A.)

* Correspondence: ahalmogren@ksu.edu.sa

Received: 21 October 2020; Accepted: 27 November 2020; Published: 30 November 2020

Abstract: Recently, many platforms have outsourced tasks to numerous smartphone devices known as Mobile Crowd-sourcing System (MCS). The data is collected and transferred to the platform for further analysis and processing. These data needs to maintain confidentiality while moving from smartphones to the platform. Moreover, the limitations of computation resources in smartphones need to be addressed to balance the confidentiality of the data and the capabilities of the devices. For this reason, elliptic curve cryptography (ECC) is accepted, widespread, and suitable for use in limited resources environments such as smartphone devices. ECC reduces energy consumption and maximizes devices' efficiency by using small crypto keys with the same strength of the required cryptography of other cryptosystems. Thus, ECC is the preferred approach for many environments, including the MCS, Internet of Things (IoT) and wireless sensor networks (WSNs). Many implementations of ECC increase the process of encryption and/or increase the space overhead by, for instance, incorrectly mapping points to EC with extra padding bits. Moreover, the wrong mapping method used in ECC results in increasing the computation efforts. This study provides comprehensive details about the mapping techniques used in the ECC mapping phase, and presents performance results about widely used elliptic curves. In addition, it suggests an optimal enhanced mapping method and size of padding bit to secure communications that guarantee the successful mapping of points to EC and reduce the size of padding bits.

Keywords: concatenation method; elliptic curve cryptography; encoding phase; mapping phase; mobile crowed-sourcing systems; probability method

1. Introduction

The rapid increase of smartphones that are equipped with many useful sensors [1,2], such as, Global Positioning System (GPS), gyroscopes, meter sensors, etc., and their ability to connect to the Internet using 3G/4G/5G connectivity, have increased the growth and the use of MCSs [3]. MCS is an approach to outsource tasks to several smartphones to collect specific data into a centralized platform to analyze and process these data in order to share it with specific users. There are many applications behind MCSs, such as monitoring the environment, monitoring the road traffic, etc., as depicted in Figure 1. For example, Alhogail, Areej, et al [4] developed an Umrah Electronic Guide system that facilitated GPS Positioning and counting techniques to self-guide Pilgrims during Umrah activities. However, MCSs transmitted these data over an insecure network (the Internet). Therefore, the confidentiality of these data and the identity of the smartphone users need to be maintained. In addition, the MCS uses low capability devices (i.e., smartphones) to collect and transmit these data; thus, the cryptography system needs to address these limitations.

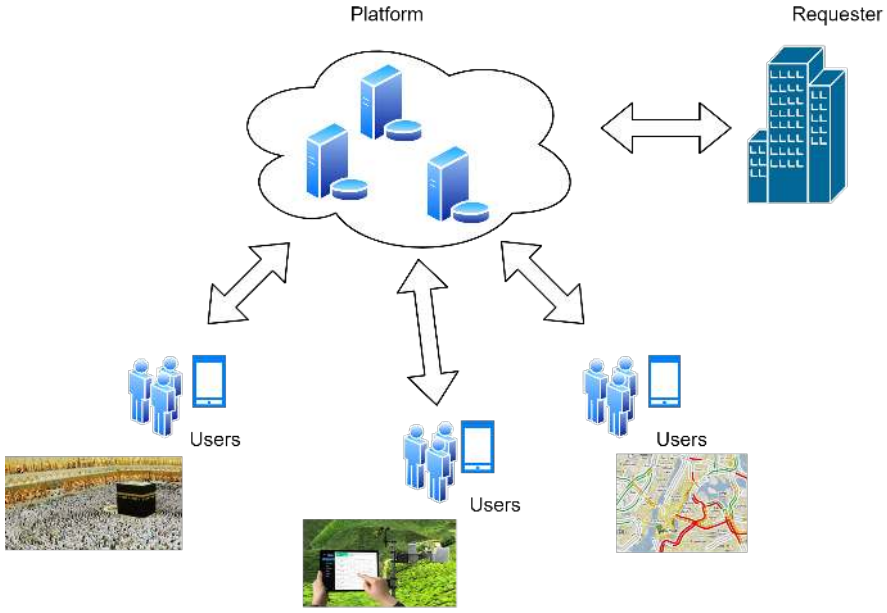

Figure 1. Applications of a Mobile Crowd-sourcing System.

Lightweight encryption schemes such as elliptic curve cryptography (ECC) are becoming increasingly desirable due to the growing interest surrounding the use of low computing power devices, particularly those associated with the Internet of Things (IoT) and wireless sensor networks (WSNs) [5–12]. Encryption schemes of this kind satisfy the need to maintain the confidentiality and integrity of transmitted data without compromising performance. Schemes such as ECC consist of many phases [13,14]. These phases are: initialising, encoding, mapping, encryption, signing, verifying, decrypting, decoding, and —finally—converting to the message [15]. Given the multi-phase nature of ECC and several other encryption schemes, it is possible that security flaws, vulnerabilities, and performance overheads may increase [16]. For this reason, deriving value from the use of ECC depends on the effective implementation of each phase, as well as robust and reliable performance evaluation. Therefore, this study helps to address MCS and the expansion of cities and urbanization for disseminating the services through efficient ECC and an enhanced mapping phase.

The mapping phase in ECC consists of a mathematical equation that represents the elliptic curve, which is given as follows:

$$y^2 \equiv x^3 + a\ x + b\ mod\ p \tag{1}$$

where $a, b \in \mathbb{Z}_p$ and $4a^3 + 27b^2 \neq 0\ mod\ p$. A given x is said to be mapped to the elliptic curve if and only if there exists a corresponding y that satisfies Equation (1). If no such y exists, then x is not mapped to the elliptic curve. The crucial advantage associated with mapping points to an elliptic curve stems from an exploitation of the elliptic curve discrete logarithm problem (ECDLP) [17], which constitutes the base of ECC. However, if a message M, encrypted using ECC, did not map to an elliptic curve (i.e., the x value of M has no corresponding y), then it is necessary to increment x and recalculate until y is found [18–20]. However, the increment to x changes M, thereby resulting in the wrong decoding phase for the retrieval of M. Thus, to secure messages in the encoding and decoding phases, certain bits should be concatenated to mapping points to avoid changing the original message. In the decoding phase, these padding bits are removed from the mapped points in a safe and secure manner.

Preliminary: Elliptic Curve Cryptography (ECC)

ECC is used widely in constrained environments, particularly those relying on low computing power devices (e.g., IoT and WSNs) [21–23]. It provides the same level of cryptographic hardness as do other asymmetric cryptography protocols, but it is marked by small key sizes and higher

performance [24,25]. For instance, cryptography schemes relying on a 1024-bit key with the Rivest-Shamir-Adleman (RSA) algorithm achieve the same level of cryptographic hardness that is associated with ECC with a 160-bit key. The result of the difference in key sizes leads to the low capabilities devices performing more effective computing [26,27]. The base of hardness in ECC is the discrete logarithm structure of elliptic curves over finite fields [28,29], where the ECC is used to exchange keys, and to encrypt transmitted message between two parties [30–32]. Additionally, ECC is used to ensure the integrity of transmitted messages and non-repudiation using elliptic curve digital signature algorithms (ECDSA) [33,34]. Many schemes use ECC to secure communications, and these schemes vary depending on the type of ECC that used [35–37]. For instance, some schemes use ECC to exchange a shared key between two parties, other schemes are applied to secure the confidentiality and integrity of messages.

Two operations are defined on the elliptic curve. The first is addition '+'. Let P and Q be two points on the elliptic curve, where $P = (x_1, y_1)$ and $Q = (x_2, y_2)$. Then the operation $P + Q$ on the elliptic curve is defined as $= (x_1, y_1) + (x_2, y_2) = (x_3, y_3)$. Notably, (x_3, y_3) is the third point on the elliptic curve that intersects with the line between P and Q. If $P = Q$, then $P + P = (x_1, y_1) + (x_1, y_1) = 2P$ is defined as point doubling.

The major operation in ECC is group multiplication [38,39]. It is the number of operations of group point doubling. It consists of two variables: firstly, d, which is an integer known only to the participants, and which serves as the private key; and secondly, $G = (x_i, y_i)$, which is the base point on the elliptic curve. The public key is the product of the operation dG, which is the d doubling times for the base point G. This operation results in point (x_j, y_j). ECC's security stems from the computational hardness associated with finding d when the adversary has the base point G and the public key [17]. The abovementioned ECDLP stipulates that there is no efficient algorithm that yields d in polynomial time.

To secure communications, maintain data integrity, and exchange keys, ECC consists of several phases [13,14]. Certain ECC applications use these phases to provide authenticated encryption (AE), while others use the phases to offer integrity or confidentiality. The following are the phases involved in ECC:

- Initialising and generating system parameters, which includes defining the elliptic curve and base points, and calculating the private key P_r and the public key P_u.
- Encoding the plaintext message M to numerical values for use in the next phase.
- Mapping the numerical values to the elliptic curve to exploit the ECDLP.
- Encrypting the mapped values.
- Hashing the encrypted message (i.e., for signing).
- Verifying the received ciphertext.
- Decrypting the ciphertext.
- Decoding the decrypted ciphertext to convert it into numerical values.
- Converting numerical values into the plaintext message M.

In ECC, the initializing and generating phase requires the greatest effort in terms of computation. This is because it is during this stage that the computation of the keys is completed. As previously noted, the public key is obtained by calculating the $d \times G$ where the d is the private key known by the sender only and G is the EC base point [40,41]. Several strategies are available for optimising this phase, for instance, improving scalar multiplication on the elliptic curve. The encoding phase involves techniques that convert the message characters into numerical values. This is necessary because the ECC cryptosystem deals with numbers [42]. Similarly, the mapping phase facilitates the mapping of the numerical values outputted from the previous phase to the elliptic curve, where its equation is used to identify the elliptic curve's pair points [43]. Next, the encryption phase involves it being represented by the summation of the mapped points and the public key. Finally, the transmitted ciphertext is signed to authenticate the sender and, in this way, secure it against tampering. The remaining phases invert

the preceding phases, in which the recipient verifies the received ciphertext, decrypts the ciphertext, decodes the numerical values, and converts it into plaintext.

This paper's objective is to present an effective mapping phase performance in ECC. To achieve this objective, several secondary goals must be addressed. In particular, it is necessary to gain insight into the current techniques used to pad bits in the mapping phase. Additionally, for each technique, this paper presents a performance analysis and evaluation. In turn, this paper provides a comprehensive investigation of the mapping phase for several known and widely used elliptic curves. For instance, secp192k1, NIST-224, and secp256k1 are examined [44–46]. Finally, this paper proposes effective padding bit values and a method that guarantees successful mapping, as well as efficiency using the least number of bits. A performance evaluation of the proposed method is given, comparing its results against those of the schemes presented in the related work section. The rest of this paper is organised as follows: the next section discusses related work on the mapping phase; the following section presents the study of padding bits and the effective bit values for padding encoded messages; the subsequent section describes the evaluation performance of the proposed method; and the last section offers concluding remarks, as well as avenues for future work.

2. Related Works

Existing schemes use ECC to reduce the encryption processing overhead. This is valuable due to the limitations of low computing power devices [47–49]. However, many of these schemes provide scant details about how a message should be padded in order to map it successfully to the elliptic curve [50–52]. Noteworthily, existing schemes have introduced significant enhancements in many areas of the elliptic curve, including scalar multiplication on the elliptic curve, encoding phase processing, and the mapping phase. To illustrate, MCS schemes in [53,54] and similar proposed schemes in [55,56] use ECC without elaborating on the processing associated with each phase. For this reason, these schemes provide reduction on the processing computation and power consumption. However, many proposed schemes have enhanced the phases involved in ECC. For instance, refs. [57,58] introduced efficient algorithms to increase scalar multiplication performance on the elliptic curve. Equally important, other proposed schemes have described how the ECC phases can be performed, but they neglect to use (or to elaborate on) the approach to padding bits used to secure the mapping phase [59,60]. Moreover, many studies have provided details on the mapping phase, specifically the padding bits step, where the mapped point requires additional bits to secure the ciphertext mapped to the elliptic curve. The remainder of this review of related work focuses on these studies and, in particular, addresses the process of how the padding bits step can improve the transmission performance and reduce the size of the ciphertext.

Four decades ago, ECC was proposed by Koblitz and Miller [61], and it began to be widely used in the beginning of this millennium [62–68]. The first curve used in ECC was introduced by Koblitz in 1987 [69]. Koblitz specified the steps needed to map plaintext to the suggested curve. More specifically, the author proposed several methods to ensure the successful mapping of x_1 to the elliptic curve. Some of these methods necessitate a large computational overhead, which means they are unsuitable for environments that rely on low computing power devices. However, Koblitz proposed one method that was suitable for such environments, where x_1 is assumed to be an integer value. Thus, in this method, it is concatenated with 3 digits. In turn, the new x_1 value is safely incremented until y_1 is obtained. Once (x_1, y_1) is mapped to the elliptic curve, the x_1 is straightforwardly decoded by removing the concatenated three digits. Resultantly, concatenating three digits is equivalent to padding 24 bits to x_1, which corresponds to 2^{24} rounds to find the corresponding y_1.

In 2018, Tiwari & Kim [70] introduced a novel approach using DNA-based ECC. In this approach, genome sequences are used to assign different values to each character set in the message. In turn, every m is mapped to the elliptic curve by multiplying it with a random integer r such that $mr < p$, where p is a large prime. Furthermore, if the mapping of mr fails, then the value is incremented by 1 and the mapping method is repeated. Moreover, when the mapping process is completed, the authors

described the reverse approach to represent the original value of m by taking the ceiling value after dividing the mapped point by r. Using this approach, m can be secured when mapping it to the elliptic curve for r rounds. However, a limitation of this approach is that the authors failed to specify requirements regarding the acceptable size of r. Similarly, the authors did not describe the impact of selecting an improper value of r.

Message mapping and reverse mapping in ECC was introduced by Sengupta & Ray in 2016 [71]. In their paper, the authors gathered the characters taken from a message into a group to map it to the elliptic curve. Subsequently, the authors suggested concatenating N bits to this group of characters. The authors stated that the value of N enables the counting of the number of rounds needed to map the group, where the number of rounds amounts to 2^N. In addition, the authors stated that no known algorithm existed for finding the optimal value of N, and the only approach involved determining the coordinates on the curve (i.e., solving the ECDLP). Thus, the authors used an 8-bit value of N as they suggested that the maximum value of N is always less than a certain value. They observed that 8 bits were adequate for use as the maximum of that value. Using this value, mapping to the EC gives 2^8 rounds for guaranteeing successful mapping to the elliptic curve.

In 2009, King [72] described an approach for mapping a message to an elliptic curve using a probabilistic strategy. The author used the binary representation of a message M in the probabilistic equation as the value of x_i. Thus, to find the corresponding y_i, the approach involves computing $x_i^3 + ax_i + b$. If y_i has a square root, then the x_i has mapped to EC and it has a corresponding y_i. However, If y_i mapped in the first round, then the mapping method continues to compute y_i by incrementing x_i until y_i is found. The number of rounds is defined by a random integer k, which is used in $x_i = M \times k$. To derive the original value of x_i, the floor value of $\frac{x_i}{k}$ is needed. In certain cases, for a given k, it is not possible to map y_i to the elliptic curve because k is too small. Similarly, an overly large value of k increases of size overhead of x_i, resulting in an increase in the data transmission overhead. It is important to note that the author defined the probability of y_i being mapped successfully based on $\frac{1}{2^k}$.

3. The Proposed Scheme-Mapping and Padding Method to ECC

The proposed scheme consists of nine phases started by generating parameters; then encoding messages to numerical values, followed by mapping these values to EC, the encrypting phase, signing the encrypted message. The rest of the phases are the reverse operations of the previous phases, beginning with verifying the signature of the received cipher text, then the decryption phase, followed by decoding phase, and converting the received points into plaintext. The main goal of this research is to study the padding methods in mapping phase where many proposed schemes did not provide a comprehensive details about the used padding method. In addition, it is noteworthy that the padding bits size is an important factor on mapping phase, where many of current studies neglect to provide it more focus and as a result it may lead to increase the size of padding bits which increase the size of mapped points or decrease the size of padding bits which result to increase the probability mapping phase failure.

Padding phase is an important phase in the ECC to make sure that the mapping points to EC are successfully restored in the decoded phase. To map a character to the EC, first it needs to convert to numerical value to map it to EC. Afterward, the converted value needs the padding phase to make sure that, if the first mapping operation to the EC failed, it is safe to increase it by one to try the mapping operation again. Finally, the mapped point can restored easily to the original value in the decoded phase. Padding phase can affect the performance of ECC schemes implemented on devices in constrained environments in two ways. The first way results from the size of padding bits needed to guarantee that the encoded points map successfully to the elliptic curve. The size of the padding bits represents the number of rounds that the encoded points can increment without affecting the original value. The second way relates to the fact that the padding bits can affect the ECC scheme in terms of the process by which the encoded points are padded. Two methods are commonly employed to pad

encoded points, namely the probability method and concatenating method. The first method pads bits by multiplying the encoded points by a random integer k, where the results can increment k times to map the encoded points to the elliptic curve safely. The original value is retrieved by taking the floor value after dividing the mapped points by k. The second method involves directly concatenating n bits to the encoded points. In this approach, 2^n rounds can be safely incremented. After mapping the encoded points, the original value is retrieved by removing the n concatenated bits.

3.1. Generating System Parameters

The main task of this phase is to generate the parameters and constructs the encryption key between the two parties. Table 1 describes the symbols used in the proposed scheme.

Table 1. List of symbols used to generate scheme parameters.

Symbol	Description
d_s	Sender private key
d_r	Recipient private key
G	Base point on elliptic curve
PU_s	Sender public key = $d_s \times G$
PU_r	Recipient public key = $d_r \times G$
p	Large prime number (192-bit)
a, b	EC coefficients, s.t. $4a^3 + 27b^2 \ mod \ p \neq 0$
H	Hash function to sign the message C_M
k_{sh}	Shared session key
M	Total number of characters in the message
B	Number of blocks for each message
N	Number of characters on each block
IV	Initial vector randomly selected (192-bit)
k	Randomly securely selected from $[1, p-1]$
C_M	The encrypted message

The key k_{sh} that shared between two parties used to encrypt mapped points on the elliptic curve. In order to create this key, the sender uses his/her private key d_s, and multiply it to recipient's public key PU_r, thus, the sender generates k_{sh}. Benefiting from ECDLP, the recipient generates the k_{sh} in the same way by multiplying his/her private key d_r to sender's public key PU_s. This process illustrated in Figure 2.

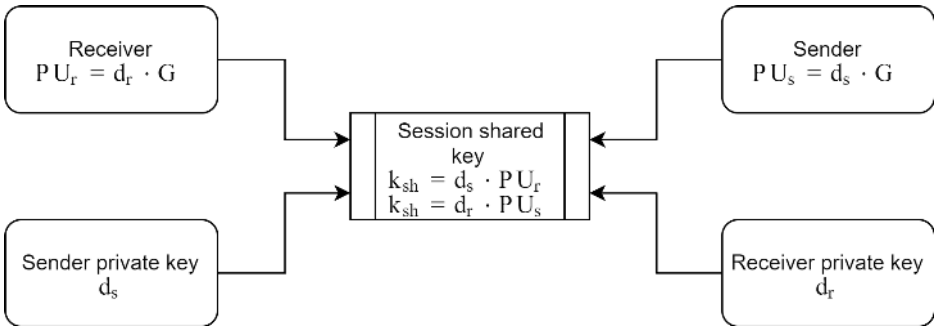

Figure 2. Generate the shared key between two parties.

3.2. Encoding and Mapping the Message to EC

The EC encoding and mapping approaches steps are enhanced to increase the performance and decrease the computation efforts. Each plaintext is divided into set of blocks notated by B, and each block B has N characters. The calculation of N is the floor of the size of prime number generated from Table 1 subtracted by 8 divided by 8. The following equation describes the process:

$$N \leq \left\lfloor \frac{p-8}{8} \right\rfloor \quad (2)$$

Similarly, the count of blocks B needed is the division of the total number of characters in the message M by the size of characters for each block N. The following equation describes the process:

$$B = \left\lceil \frac{M}{N} \right\rceil \quad (3)$$

The encoding step for each block B is completed by converting its ASCII code to binary to perform cipher block chaining (CBC) to secure the cipher texts against several encryption attacks. Equally importantly, the mapping phase needs to append set of bits to each encoded blocks. This step is an important process to secure and guarantee the successful mapping to EC. The details of padding bits is described in subsection D.

3.3. Authenticated Encryption of the Mapped Points

Many proposed schemes consider the mapping phase is appropriate and enough to secure the transmitted message. However, mapping points to an EC is the first step to secure the cipher text and it is needed for the encryption step which is the addition of encryption key with the mapped points as follows: $C_M = k_{sh} + Mapped\ points$. Following that, it is necessary to maintain the integrity of the cipher text C_M using the ECDSA. The first step is to obtain $e = HASH(C_M)$ and take the left most p bits of e. Then, the second step is randomly select k and calculate $(x, y) = kG$, then calculate r, where $r = x \ mod \ p$ and $r \neq 0$. Finally, the signed cipher text is the pair of (r, s) where $s = (z + d_s \times r)\, k^{-1}$.

3.4. Mapping Phase and Padding Encoded Points

The hardiness of ECDLP resides in the mapping phase and the correctness of the steps of mapping points to EC. Failing of mapping point to EC means that the encrypted data using ECC is weak. Several security issues are raised by failing to map points to EC, for instance, ignore the padding bits that result in failing to map points to EC by 50%. Equally importantly, pad encoded points with a small size of bits raise the percentage of the fail of the mapping phase. Similarly, the increase of the size of padding bits leads to an increase in the computation and transition overhead, particularly for low computation devices (such IoT) that need to deal with a huge amount of data (such as big data processing).

In the review of related work, one of two padding methods was used in all of the proposed schemes: the probability method, which relies on integer multiplication, and the concatenation method, which combines the encoded points with a specific number of padding bits. Hence, it is worth evaluating the performance of both methods to identify viable ways in which minimise the computational overhead, particularly for devices operating in constrained environments. This performance evaluation constitutes the focus of the next two subsections.

3.4.1. Probability Method

The main computational overhead associated with the probability method arises from the need to multiply the encoded points by a random integer k. The value of k represents the number of rounds needed to map the encoded points. For instance, if the number of rounds needed to map the encoded points is 25, then $k = 25$. However, the value of 25 increases the number of padding bits that must be added to the encoded points by $\lceil \log 25 \rceil = 5$. Moreover, padding with 5 bits provides $2^5 = 32$ rounds. Resultantly, $32 - 25 = 7$ rounds remain unused when multiplying the encoded points by 25. Figure 3 illustrates the maximum number of unused rounds for padding bit sizes ranging from 0 to 6.

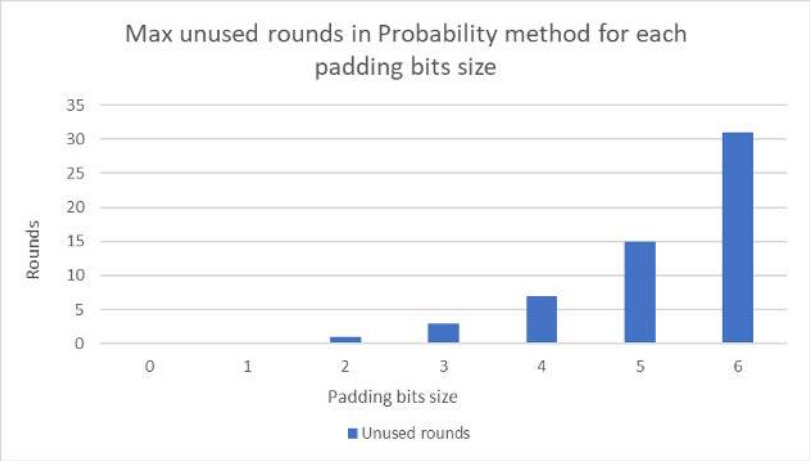

Figure 3. Maximum number of unused rounds for various padding bit sizes.

3.4.2. Concatenation Method

The main computational overhead associated with the concatenation method arises from the need to append padding bits to the encoded points. In this method, the number of appended bits is selected based on the number of rounds needed to secure the mapping phase of the encoded points. The appended bits of size 5 is 00000, thereby providing up to $2^5 = 32$ rounds to secure the padding phase. In the concatenation phase, no rounds are unused because every appended bit is used. However, the concatenation method is an expensive approach, especially for low computing power devices. The complexity of string concatenation is computed as $O(n^2)$ [73], where n is the number of padding bits. Contrastingly, the complexity of integer multiplication is computed as $O(n \log n)$ [74], where n is the size of the multiplication value. Figure 4 compares the complexity of the two methods.

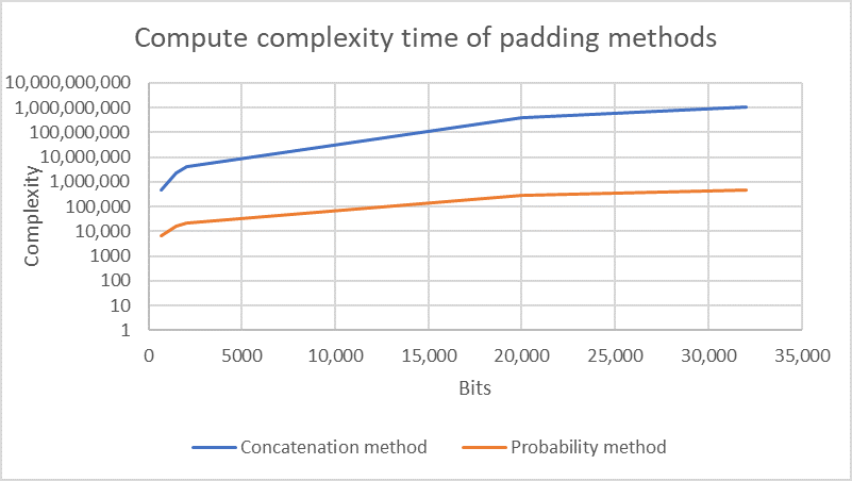

Figure 4. Complexity of probability and concatenation padding methods.

3.5. Our proposed Padding Method

Many ECC schemes have proposed several sizes for the padding bits that are used to secure the mapping phase to the elliptic curve. However, there is no comprehensive study of the suitable size of such bits. Certain schemes have proposed that 8 bits should be padded to each character mapped to the elliptic curve, while other schemes have proposed adding 8 padding bits to a set of aggregate characters. Similarly, schemes that use the probability method typically fail to specify the value of k, thereby leading either to an increase in the size of the transmitted data overhead or to an increase in the probability of a failed mapping. Based on King's [72] research, the successful mapping occurs in the 8th round, as illustrated in Figure 5.

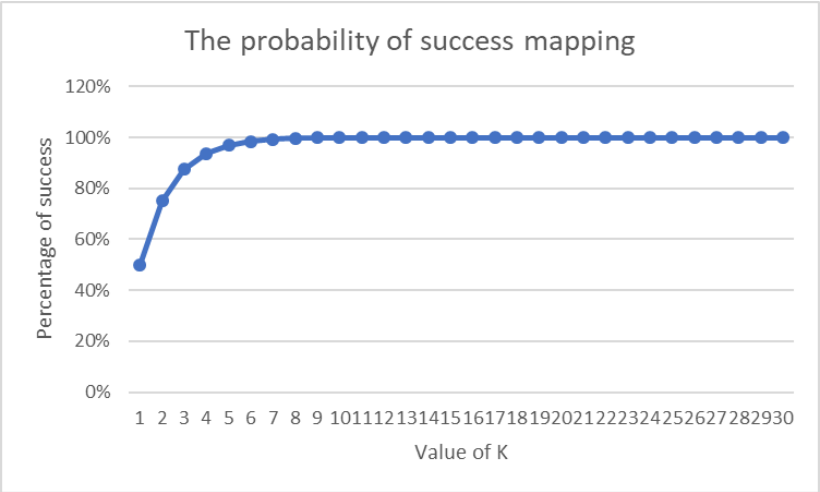

Figure 5. King's [72] probability value of successful mapping.

In this research, several experiments were conducted to evaluate the size of the padding bits required to increment x_i safely and to find the corresponding y_i value. Similarly, this research also aims to maximized the performance of the padding bits method and overcome the weakness of both methods mentioned in the previous sections. The experiment addressed the following cases:

- Case 1: Random 192-bit integer based on the requirements of secp192k1.
- Case 2: Random 224-bit integer based on the requirements of nistP224.
- Case 3: Random 256-bit integer based on the requirements of secp256k1.
- Case 4: Random 384-bit integer based on the requirements of secp384r1.
- Case 5: Random 512-bit integer based on the requirements of secp512r1.

For each case, the experiment was repeated 10 million times for each curve to evaluate the maximum number of rounds needed to successfully map the random number to the curve. The results show that, for certain random numbers, the maximum number of rounds was less than 25.

Figures 6–10 present the results of the experiments for each of the cases. The figures show the percentages and numbers of successfully mapped points for each round (the orange curves and blue columns, respectively).

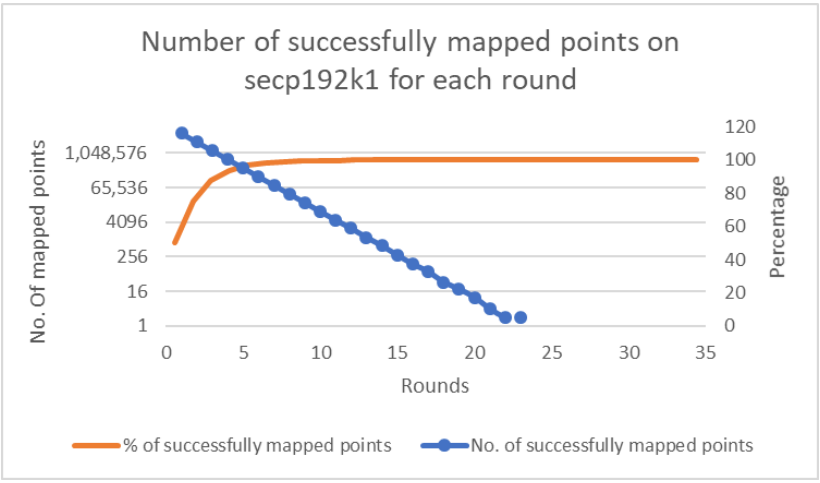

Figure 6. Case 1—Successfully mapped points count to secp192k1.

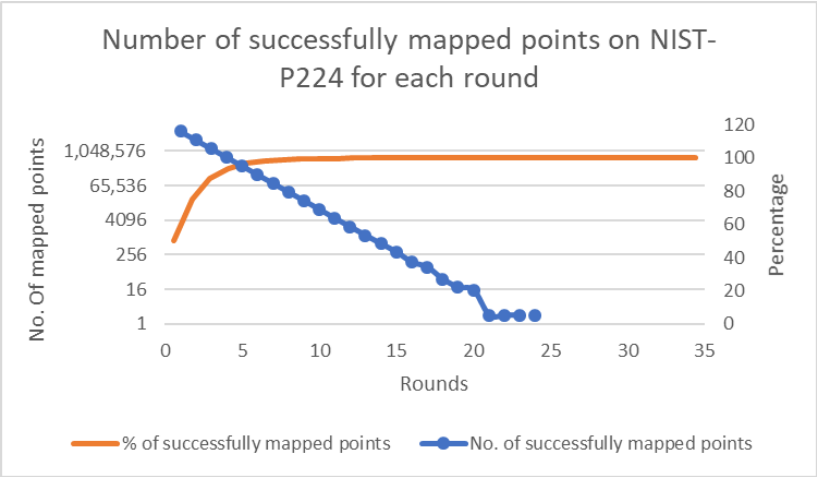

Figure 7. Case 2—Successfully mapped points count to nistP224.

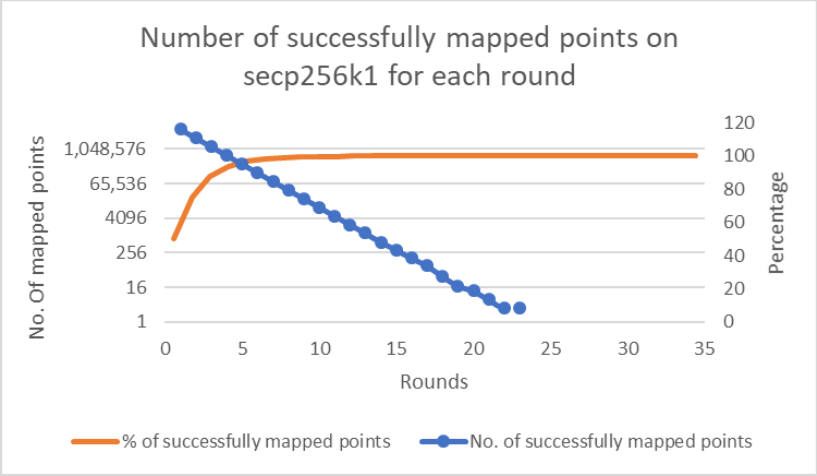

Figure 8. Case 3—Successfully mapped points count to secp256k1.

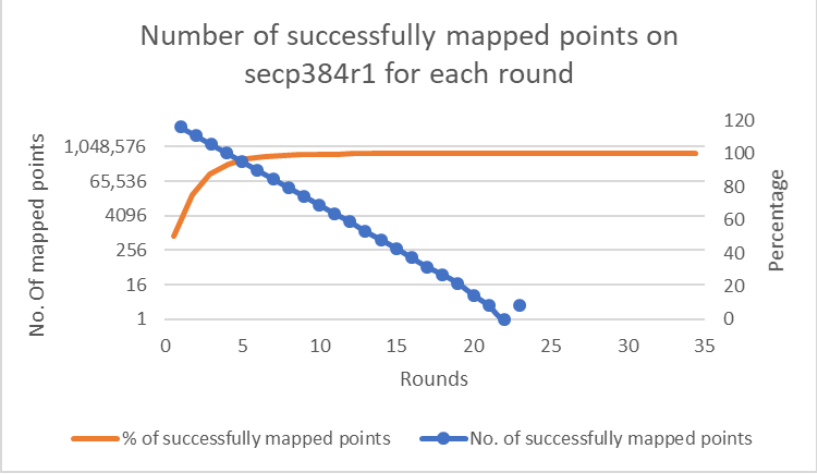

Figure 9. Case 4—Successfully mapped points count to secp384r1.

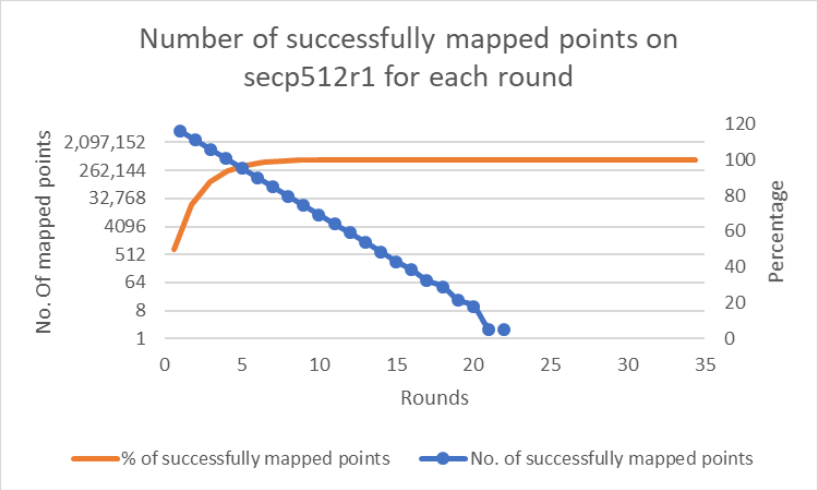

Figure 10. Case 5—Successfully mapped points count to secp512r1.

The experiments presented showed that, for every case, the number of rounds did not exceed 25. Thus, the size of padding bits that are needed to secure the mapping phase should not exceed $\lceil \log_2 25 \rceil = 5$. Thus, it is possible to overcome the weakness in the probability method where the unused rounds may increase based on the value of k. In addition, there is a weakness in the concatenation method, which increases the computation overhead. This research provides the enhancement of the probability method by adding one step to choose $k = 2^{\lceil \log k \rceil}$. The Algorithm 1 describes this process:

Algorithm 1: The proposed enhanced probability method algorithm.

Input: *Mapping points M*
Output: *Padded mapping points x*
Randomly select value k;
let $k = 2^{\lceil \log k \rceil}$;
for $j = 0$ *to* $k - 1$ *do*;
  let $x = M \times K + j \bmod p$;
  if x mapped to EC then break;
if $j < k$ *return x else return failed*;

The enhanced probability method gains the strength of the concatenation and probability methods. In addition, it overcomes the weakness of both methods to increase the performance of the proposed system. What we mean by the strength of concatenation method is the number of rounds for the size of appended bits as depicted in Figure 3. Similarly, the strength of probability method means the redaction of the complexity that offered by concatenation method as depicted in Figure 4. In the following section, we provide the performance evaluation for the enhanced probability method. It is worth mentioning that, for computation evaluation, both the enhanced probability and probability methods provide the same result. However, the enhanced probability method overcomes the weakness of the probability method in terms of unused rounds in comparison with the size of appended bits.

4. Performance Evaluation

The probability and concatenation methods each have strengths and weaknesses in terms of their performance. Thus, in order to compare these methods, we simulated both approaches by writing Java code to compute the evaluation of both methods' performance on several elliptic curves, namely secp192k1, nistP224, secp256k1, secp384r1, and secp512r1. Figures 11–15 illustrate the results of this performance evaluation, indicating that, for all evaluated elliptic curves, the concatenation method (depicted in blue) required greater computational effort when compared to the enhanced

probability method. Specifically, the computational requirement for the concatenation method was 35 times greater compared to the enhanced probability method for secp192k1, and it was 10–20 times greater for the other elliptic curves. The probability methods provide the same results as the enhanced probability method in terms of the computation evaluation test.

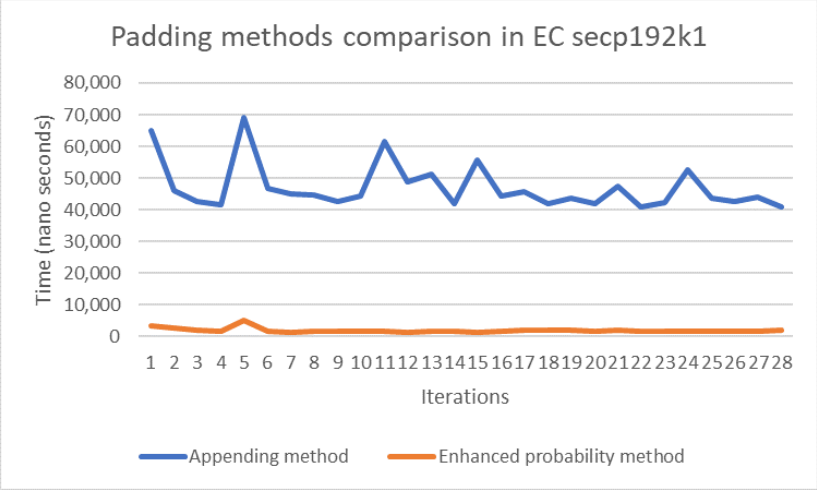

Figure 11. Performance evaluation for padding methods on secp192k1.

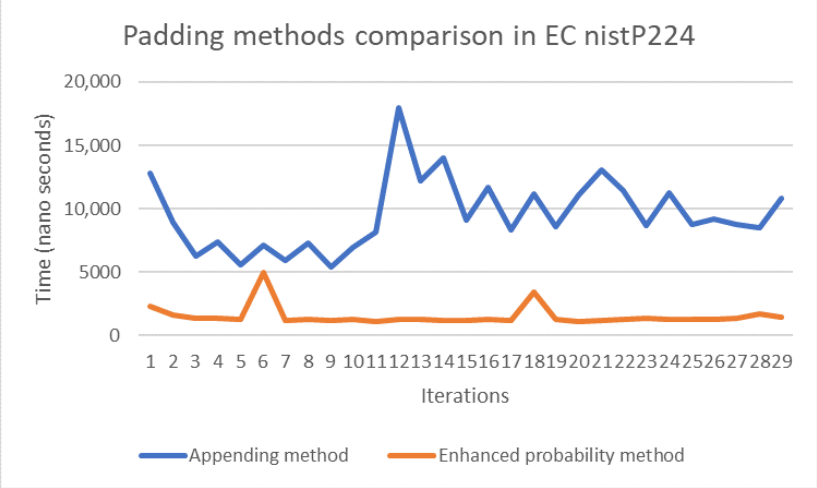

Figure 12. Performance evaluation for padding methods on nistP224.

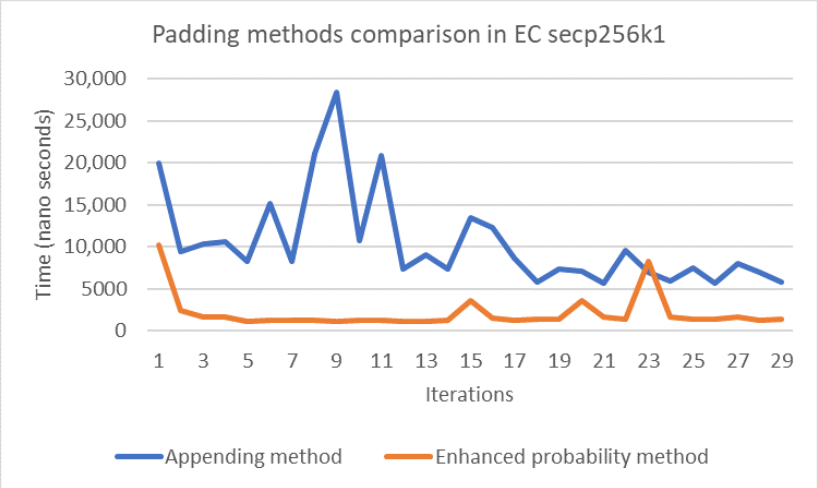

Figure 13. Performance evaluation for padding methods on secp256k1.

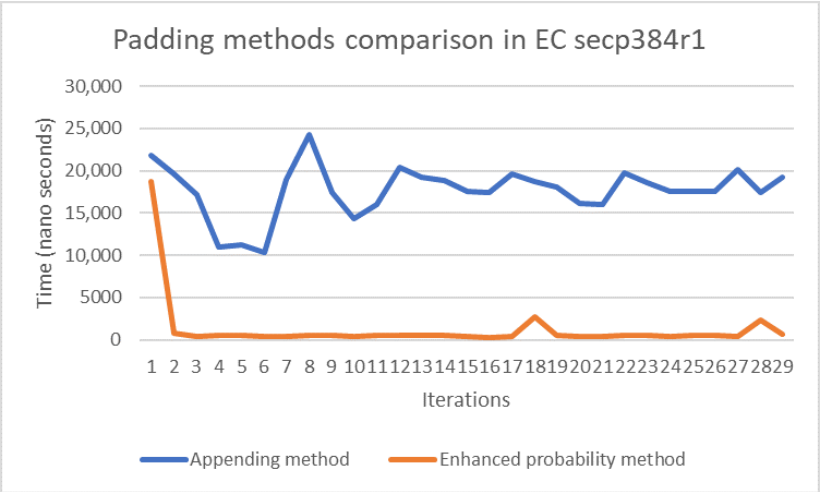

Figure 14. Performance evaluation for padding methods on secp3841.

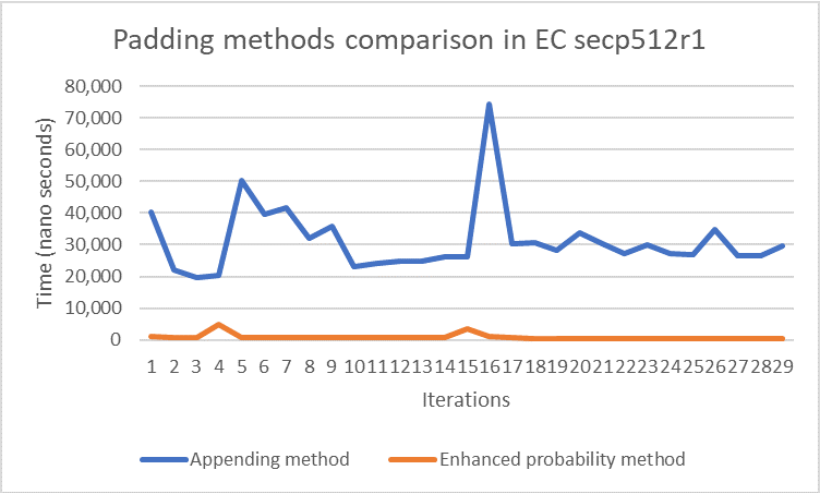

Figure 15. Performance evaluation for padding methods on secp512r1.

The previous findings motivated a direct comparison of the computational overhead associated with each of the elliptic curves for every padding method. For instance, we were interested in quantifying the increase in computational requirements based on the elliptic curve used. Figure 16 shows the variation between the computation on all elliptic curves for the enhanced probability method, indicating differences of less than 3 times between the highest and lowest loads. Similarly, Figure 17 shows the variation between the computation on all elliptic curves for the concatenation method. As the figures indicate, the differences increased in comparison with the enhanced probability method, and the variation between the highest and lowest loads reached 8 times.

Similarly, a comparison between Enhanced probability method and Concatenation method on term of Memory usage. As in the previous evaluation test, we measure the memory space utilized for using both methods on the same set of EC. As a result, the enhanced probability method uses less memory space than concatenation method. This result is depicted in Figure 18.

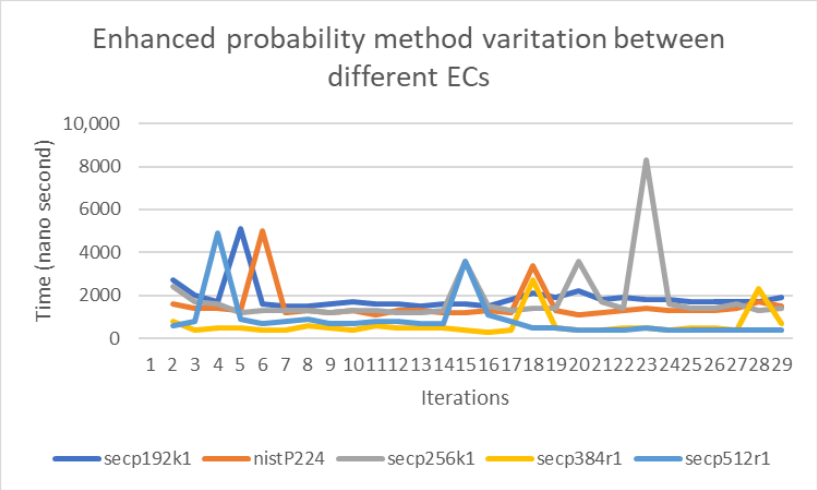

Figure 16. The variation between the set of Elliptic Curves (ECs) in the enhanced probability method.

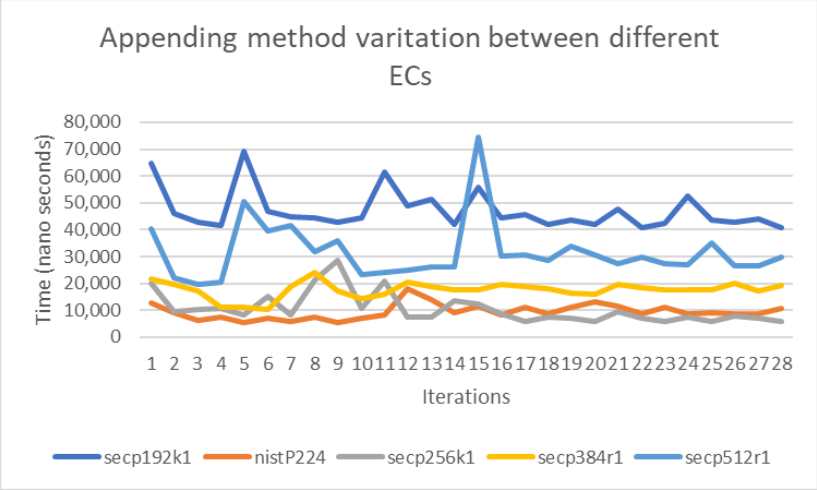

Figure 17. The variation between the set of ECs in the concatenation method.

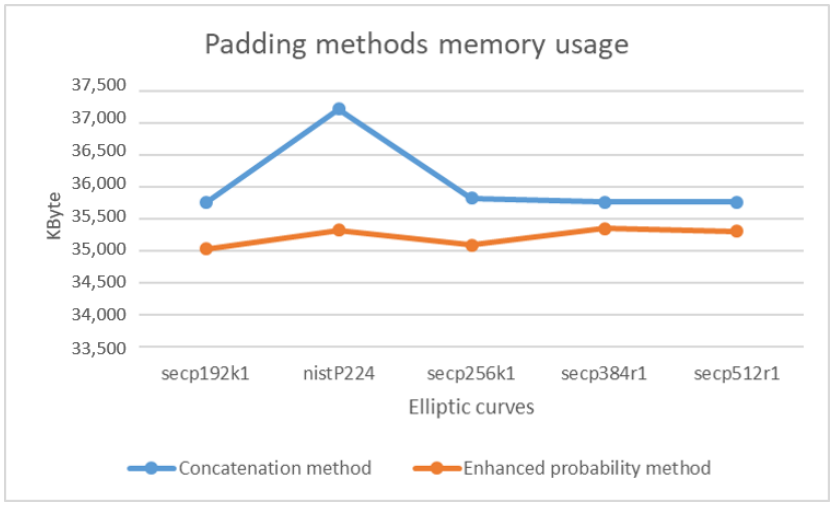

Figure 18. The Memory usage on padding methods using the set of ECs.

In the other hand, the evaluation of the enhanced probability method in terms of minimum rounds to the size of appended bits is depicted in Figure 19. In this comparison, the enhanced probability method provides more rounds than the probability for each padding bit. For instance, for 5 bits

appended to encoded points, the enhanced probability offers minimum 32 rounds. However, for the same same size, the probability method offers minimum 17 rounds. Thus, the enhanced probability method provides a better chance to map encoding points to EC than the the probability method.

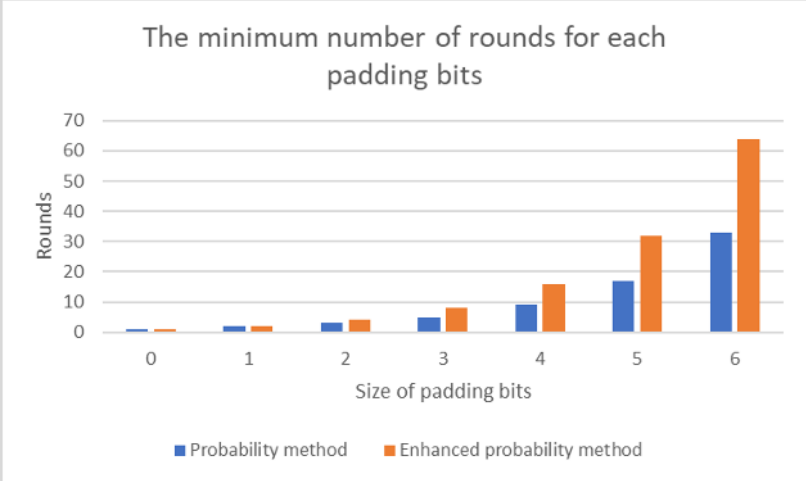

Figure 19. Number of minimum rounds for each size of appended bits.

To illustrate the overall comparison of the mapping methods, we summarize our findings in Table 2. In this table we show the detailed performance evaluation between our proposed method, (the enhanced probability method), the probability method and the concatenation method. It is clear that our proposed method is better than the other methods by gaining the strength features of those methods.

Table 2. The comparison between mapping methods.

Criteria	**1**	**2**	**3**
Number of rounds to value k	k	$2^{\lceil \log k \rceil}$	$2^{\lceil \log k \rceil}$
Provide max number of rounds	N	Y	Y
RAM usage per operation	Less	High	Less
Variation between different ECCs	Less	High	Less
CPU utilization per operation	Less	High	Less
Complexity of the method	$O(n \log n)$	$O(n^2)$	$O(n \log n)$

1: Probability method, 2: Concatenation method and 3: our proposed method (Enhanced probability method)

It is worth mentioning that, in our experiments, we used the Bluej Java Development Environment to code the mapping and padding phase. The environment used to test the code is based on Windows 10, and, in terms of hardware, Intel Core i7-4510U, 128GB SSD, and 8GB RAM.

5. Conclusions

This paper examined considerations relating to the optimal use of padding bits in the ECC mapping phase. It emphasised the importance of padding bits, particularly in terms of their performance implications. Additionally, the consequences arising from the improper addition of padding bits to encoded points were illustrated. Many proposed schemes choose a padding method without a clear explanation. Furthermore, most schemes tend not to provide details about the chosen sizes of padding bits. Therefore, this research sought to determine the optimal size of padding bits for the secure mapping of encoded points to an elliptic curve. It identified and enhanced a padding method that increases device performance and reduces the computational overhead without undermining security. Moreover, an evaluation of simulation performance was provided to illuminate and support the research findings.

In future work, the implications of implementing the suggested padding bits and enhanced mapping method in a real-world environment will be studied. The authors will also compare the computation results obtained in a real-world environment against those reported in this study's simulation.

Author Contributions: Conceptualization, H.A. and A.A.; methodology, A.A.; software, H.A.; validation, A.A.; formal analysis, H.A.; investigation, H.A.; resources, A.A.; data curation, A.A. and M.A.; writing—original draft preparation, H.A.; writing—review and editing, A.A. and M.A.; visualization, H.A.; supervision, A.A.; project administration, A.A.; funding acquisition, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding or This research was funded by the Deputyship for Research & Innovation, "Ministry of Education" in Saudi Arabia grant number IFKSURG-1437-035 And The APC was funded by IFKSURG-1437-035.

Acknowledgments: The authors extend their appreciation to the Deputyship for Research & Innovation, "Ministry of Education" in Saudi Arabia for funding this research work through the project number IFKSURG-1437-035.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
IoT	Internet of Things
IV	Initial Vector
MCS	Mobile Crowed-sourcing System
AE	Authenticated Encryption
ECIES	Elliptic Curve Integrated Encryption Scheme
ECDLP	Elliptic Curve Discrete Logarithm Problem

References

1. Alkhalifa, I.S.; Almogren, A.S. NSSC: Novel segment based safety message broadcasting in cluster-based vehicular sensor network. *IEEE Access* **2020**, *8*, 34299–34312. [CrossRef]
2. Ahmad Awan, K.; UdDin, I.; Almogren, A.; Almajed, H. AgriTrust—A Trust Management Approach for Smart Agriculture in Cloud-based Internet of Agriculture Things. *Sensors* **2020**, *20*, 6174. [CrossRef] [PubMed]
3. Sarker, S.; Razzaque, M.A.; Hassan, M.M.; Almogren, A.; Fortino, G.; Zhou, M. Optimal selection of crowdsourcing workers balancing their utilities and platform profit. *IEEE Internet Things J.* **2019**, *6*, 8602–8614. [CrossRef]
4. Alhogail, A.; Alshabanat, L.; Almusharraf, N.; Alkharis, A.; Almusharraf, B. Umrah Electronic Guide (Umrah E-Guide). In Proceedings of the 2019 International Conference on Information and Communications Technology (ICOIACT), Haikou, China, 5–7 July 2019; pp. 403–407.
5. Tayyaba, S.K.; Khattak, H.A.; Almogren, A.; Shah, M.A.; Din, I.U.; Alkhalifa, I.; Guizani, M. 5G Vehicular Network Resource Management for Improving Radio Access Through Machine Learning. *IEEE Access* **2020**, *8*, 6792–6800.
6. Hassan, M.M.; Gumaei, A.; Huda, S.; Almogren, A. Increasing the Trustworthiness in the Industrial IoT Networks Through a Reliable Cyberattack Detection Model. *IEEE Trans. Ind. Inform.* **2020**, *16*, 6792–6800. [CrossRef]
7. Haseeb, K.; Islam, N.; Almogren, A.; Din, I.U. Intrusion Prevention Framework for Secure Routing in WSN-Based Mobile Internet of Things. *IEEE Access* **2019**, *7*, 185496–185505. [CrossRef]
8. Almogren, A.S. Intrusion detection in Edge-of-Things computing. *J. Parallel Distrib. Comput.* **2020**, *137*, 259–265. [CrossRef]
9. AlMajed, H.N.; AlMogren, A.S. Simple and Effective Secure Group Communications in Dynamic Wireless Sensor Networks. *Sensors* **2019**, *19*, 1909. [CrossRef]

10. Awan, K.A.; Din, I.U.; Almogren, A.; Guizani, M.; Altameem, A.; Jadoon, S.U. Robusttrust–a pro-privacy robust distributed trust management mechanism for internet of things. *IEEE Access* **2019**, *7*, 62095–62106. [CrossRef]
11. Din, I.U.; Hassan, S.; Almogren, A.; Ayub, F.; Guizani, M. PUC: Packet Update Caching for energy efficient IoT-based Information-Centric Networking. *Future Gener. Comput. Syst.* **2020**, *111*, 634–643. [CrossRef]
12. Al-Qarni, B.H.; Almogren, A.; Hassan, M.M. An efficient networking protocol for internet of things to handle multimedia big data. *Multimed. Tools Appl.* **2019**, *78*, 30039–30056. [CrossRef]
13. Ganesh, M.G.G. Secure Method for Text Encryption using Elliptic Curve Cryptography. *Int. J. Adv. Sci. Res. Eng. Trends* **2018**, *3*, 11–15.
14. Kumar, R. Cryptanalysis of Protocol for Enhanced Threshold Proxy Signature Scheme Based on Elliptic Curve Cryptography for Known Signers. In *Knowledge Computing and Its Applications*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 191–211.
15. Haseeb, K.; Islam, N.; Almogren, A.; Din, I.U.; Almajed, H.N.; Guizani, N. Secret sharing-based energy-aware and multi-hop routing protocol for IoT based WSNs. *IEEE Access* **2019**, *7*, 79980–79988. [CrossRef]
16. Almajed, H.N.; Almogren, A.S.; Altameem, A. A Resilient Smart Body Sensor Network Through Pyramid Interconnection. *IEEE Access* **2019**, *7*, 51039–51046. [CrossRef]
17. Mrabet, A.; El-Mrabet, N.; Lashermes, R.; Rigaud, J.B.; Bouallegue, B.; Mesnager, S.; Machhout, M. High-performance Elliptic Curve Cryptography by Using the CIOS Method for Modular Multiplication. In Proceedings of the International Conference on Risks and Security of Internet and Systems, Roscoff, France, 5–7 September 2016; pp. 185–198.
18. Shah, D.P.; Shah, N.P. Implementation of Digital Signature Algorithm by using Elliptical Curve p-192. *Aust. J. Wirel. Technol. Mobil. Secur.* **2019**, *1*, 1–4. [CrossRef]
19. Abdullah, K.E.; Ali, N.H.M. Security Improvement in Elliptic Curve Cryptography. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 122–131. [CrossRef]
20. Hu, X.; Zheng, X.; Zhang, S.; Li, W.; Cai, S.; Xiong, X. A High-Performance Elliptic Curve Cryptographic Processor of SM2 over GF (p). *Electronics* **2019**, *8*, 431. [CrossRef]
21. Li, X.; Niu, J.; Bhuiyan, M.Z.A.; Wu, F.; Karuppiah, M.; Kumari, S. A robust ECC-based provable secure authentication protocol with privacy preserving for industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2017**, *14*, 3599–3609. [CrossRef]
22. Kumari, S.; Renuka, K. A provably secure biometrics and ECC-based authentication and key agreement scheme for WSNs. *Int. J. Commun. Syst.* **2019**, 33, e4194. [CrossRef]
23. Patel, C.; Doshi, N. Cryptanalysis of ecc-based key agreement scheme for generic IoT network model. In Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 6 July 2019; pp. 1–7.
24. Chandel, S.; Cao, W.; Sun, Z.; Yang, J.; Zhang, B.; Ni, T.Y. A Multi-dimensional Adversary Analysis of RSA and ECC in Blockchain Encryption. In Proceedings of the Future of Information and Communication Conference, San Francisco, CA, USA, 14 March 2019; pp. 988–1003.
25. Mallouli, F.; Hellal, A.; Saeed, N.S.; Alzahrani, F.A. A Survey on Cryptography: Comparative Study between RSA vs ECC Algorithms, and RSA vs El-Gamal Algorithms. In Proceedings of the 2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), Paris, France, 21–23 June 2019; pp. 173–176.
26. Almajed, H.N.; Almogren, A.S. SE-Enc: A Secure and Efficient Encoding Scheme Using Elliptic Curve Cryptography. *IEEE Access* **2019**, *7*, 175865–175878. [CrossRef]
27. Iqbal, Z.; Javaid, N.; Iqbal, S.; Aslam, S.; Khan, Z.A.; Abdul, W.; Almogren, A.; Alamri, A. A domestic microgrid with optimized home energy management system. *Energies* **2018**, *11*, 1002. [CrossRef]
28. Li, B. Group Structure of Special Parabola and Its Application in Cryptography. *Appl. Comput. Math.* **2019**, *8*, 88–94. [CrossRef]
29. Nelson, K.; Solymosi, J.; Tom, F.; Wong, C. The number of rational points of hyperelliptic curves over subsets of finite fields. *Involv. J. Math.* **2019**, *12*, 755–765. [CrossRef]
30. Phimphinith, A.; Anping, X.; Zhu, Q.; Jiang, Y.; Shen, Y. An Enhanced Mutual Authentication Scheme Based on ECDH for IoT Devices Using ESP8266. In Proceedings of the 2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN), Chongqing, China, 12–15 June 2019; pp. 490–496.

31. Advani, N.; Rathod, C.; Gonsai, A.M. Comparative Study of Various Cryptographic Algorithms Used for Text, Image, and Video. In *Emerging Trends in Expert Applications and Security*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 393–399.
32. Hosam, O.; Ahmad, M.H. Hybrid design for cloud data security using combination of AES, ECC and LSB steganography. *Int. J. Comput. Sci. Eng.* **2019**, *19*, 153–161. [CrossRef]
33. Kittur, A.S.; Pais, A.R. A new batch verification scheme for ECDSA signatures. *Sādhanā* **2019**, *44*, 157. [CrossRef]
34. Li, Y.; Zhang, P. Security Analysis and Improvement of Elliptic Curve Digital Signature Scheme. In Proceedings of the International Conference on Artificial Intelligence and Security, New York, NY, USA, 26–28 July 2019; pp. 609–617.
35. Chatterjee, S.; Samaddar, S.G. A Robust Lightweight ECC-Based Three-Way Authentication Scheme for IoT in Cloud. In *Smart Computing Paradigms: New Progresses and Challenges*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 101–111.
36. Al Hamid, H.A.; Rahman, S.M.M.; Hossain, M.S.; Almogren, A.; Alamri, A. A security model for preserving the privacy of medical big data in a healthcare cloud using a fog computing facility with pairing-based cryptography. *IEEE Access* **2017**, *5*, 22313–22328. [CrossRef]
37. Manzoor, A.; Javaid, N.; Ullah, I.; Abdul, W.; Almogren, A.; Alamri, A. An intelligent hybrid heuristic scheme for smart metering based demand side management in smart homes. *Energies* **2017**, *10*, 1258. [CrossRef]
38. Wu, T.; Wang, R. Fast unified elliptic curve point multiplication for NIST prime curves on FPGAs. *J. Cryptogr. Eng.* **2019**, 1–10. [CrossRef]
39. Shahroodi, T.; Bayat-Sarmadi, S.; Mosanaei-Boorani, H. Low-Latency Double Point Multiplication Architecture Using Differential Addition Chain Over GF (2^m). *IEEE Trans. Circ. Syst. Regul. Pap.* **2019**, *66*, 1465–1473. [CrossRef]
40. Fournaris, A.P.; Dimopoulos, C.; Moschos, A.; Koufopavlou, O. Design and leakage assessment of side channel attack resistant binary edwards Elliptic Curve digital signature algorithm architectures. *Microprocess. Microsyst.* **2019**, *64*, 73–87. [CrossRef]
41. Hussain, B.; Hasan, Q.U.; Javaid, N.; Guizani, M.; Almogren, A.; Alamri, A. An Innovative Heuristic Algorithm for IoT-Enabled Smart Homes for Developing Countries. *IEEE Access* **2018**, *6*, 15550–15575. [CrossRef]
42. AlSaad, S.N.; Naji, A.K. Elliptic Curve Video Encryption in Mobile Phone Based on Multi-Keys and Chaotic Map. *Al-Mustansiriyah J. Sci.* **2018**, *29*, 106–116.
43. Reyad, O. Text message encoding based on elliptic curve cryptography and a mapping methodology. *Inf. Sci. Lett.* **2018**, *7*, 7–11. [CrossRef]
44. Dasgupta, D.; Shrein, J.M.; Gupta, K.D. A survey of blockchain from security perspective. *J. Bank. Financ. Technol.* **2019**, *3*, 1–17. [CrossRef]
45. Chen, B.; Hu, C.; Zhao, C.A. Note on scalar multiplication using division polynomials. *IET Inf. Secur.* **2016**, *11*, 195–198. [CrossRef]
46. Tan, S.; Yeow, K.; Hwang, S.O. Enhancement of a Lightweight Attribute-Based Encryption Scheme for the Internet of Things. *IEEE Internet Things J.* **2019**, *6*, 6384–6395. [CrossRef]
47. Ahmad, A.; Khan, A.; Javaid, N.; Hussain, H.M.; Abdul, W.; Almogren, A.; Alamri, A.; Azim Niaz, I. An optimized home energy management system with integrated renewable energy and storage resources. *Energies* **2017**, *10*, 549. [CrossRef]
48. Haseeb, K.; Almogren, A.; Islam, N.; Ud Din, I.; Jan, Z. An energy-efficient and secure routing protocol for intrusion avoidance in IoT-based WSN. *Energies* **2019**, *12*, 4174. [CrossRef]
49. Huda, S.; Yearwood, J.; Hassan, M.M.; Almogren, A. Securing the operations in SCADA-IoT platform based industrial control system using ensemble of deep belief networks. *Appl. Soft Comput.* **2018**, *71*, 66–77. [CrossRef]
50. Al-Qurishi, M.; Rahman, S.M.M.; Hossain, M.S.; Almogren, A.; Alrubaian, M.; Alamri, A.; Al-Rakhami, M.; Gupta, B.B. An efficient key agreement protocol for Sybil-precaution in online social networks. *Future Gener. Comput. Syst.* **2018**, *84*, 139–148. [CrossRef]
51. Awan, K.A.; Din, I.U.; Almogren, A.; Almajed, H.; Mohiuddin, I.; Guizani, M. NeuroTrust-Artificial Neural Network-based Intelligent Trust Management Mechanism for Large-Scale Internet of Medical Things. *IEEE Internet Things J.* **2020**. [CrossRef]

52. Sultana, T.; Almogren, A.; Akbar, M.; Zuair, M.; Ullah, I.; Javaid, N. Data sharing system integrating access control mechanism using blockchain-based smart contracts for IoT devices. *Appl. Sci.* **2020**, *10*, 488. [CrossRef]
53. Kumar, V.; Li, H.; Park, J.M.J.; Bian, K. Enforcement in spectrum sharing: Crowd-sourced blind authentication of co-channel transmitters. In Proceedings of the 2018 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), Seoul, Korea, 22–25 October 2018; pp. 1–10.
54. Kumar, V.; Li, H.; Park, J.M.J.; Bian, K. Crowd-Sourced Authentication for Enforcement in Dynamic Spectrum Sharing. *IEEE Trans. Cogn. Commun. Netw.* **2019**, *5*, 625–636. [CrossRef]
55. Ferretti, L.; Marchetti, M.; Colajanni, M. Fog-based Secure Communications for Low-power IoT Devices. *ACM Trans. Internet Technol.* **2019**, *19*, 27. [CrossRef]
56. Khan, S.; Khan, R. Elgamal Elliptic Curve Based Secure Communication Architecture for Microgrids. *Energies* **2018**, *11*, 759. [CrossRef]
57. Ay, A.U.; Mancillas-López, C.; Öztürk, E.; Rodrıguez-Henrıquez, F.; Savaş, E. Constant-time hardware computation of elliptic curve scalar multiplication around the 128 bit security level. *Microprocess. Microsyst.* **2018**, *62*, 79–90. [CrossRef]
58. Liu, S.; Yao, H.; Wang, X.A. Fast elliptic curve scalar multiplication for resisting against SPA. *Int. J. Comput. Sci. Eng.* **2018**, *17*, 343–352. [CrossRef]
59. Singh, L.D.; Singh, K.M. Image encryption using elliptic curve cryptography. *Procedia Comput. Sci.* **2015**, *54*, 472–481. [CrossRef]
60. Das, P.; Giri, C. An Efficient Method for text Encryption using Elliptic Curve Cryptography. In Proceedings of the 2018 IEEE 8th International Advance Computing Conference (IACC), Greater Noida, India, 14–15 December 2019; pp. 96–101.
61. Sau, S.; Baidya, P.; Paul, R.; Mandal, S. Binary Field Point Multiplication Implementation in FPGA Hardware. In *Intelligent and Cloud Computing*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 387–394.
62. Hankerson, D.; Hernandez, J.L.; Menezes, A. Software implementation of elliptic curve cryptography over binary fields. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Worcester, MA, USA, 17–18 August 2000; pp. 1–24.
63. Bailey, D.V.; Paar, C. Efficient arithmetic in finite field extensions with application in elliptic curve cryptography. *J. Cryptol.* **2001**, *14*, 153–176. [CrossRef]
64. Leung, K.; Ma, K.; Wong, W.K.; Leong, P.H.W. FPGA implementation of a microcoded elliptic curve cryptographic processor. In Proceedings of the 2000 IEEE Symposium on Field-Programmable Custom Computing Machines (Cat. No. PR00871), Napa Valley, CA, USA,17–19 April 2000; pp. 68–76.
65. Smart, N.P. A comparison of different finite fields for elliptic curve cryptosystems. *Comput. Math. Appl.* **2001**, *42*, 91–100. [CrossRef]
66. Zhong, T.W.N.X.C. Elliptic Curve Cryptography-based Combined Public Key Technique. *Comput. Eng. Appl.* **2003**, *21*, 1–3.
67. Dyka, Z.; Langendoerfer, P. Area efficient hardware implementation of elliptic curve cryptography by iteratively applying Karatsuba's method. In Proceedings of the Design, Automation and Test in Europe, Munich, Germany, 7–11 March 2005; pp. 70–75.
68. Icart, T. How to hash into elliptic curves. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 2009; pp. 303–316.
69. Koblitz, N. Elliptic curve cryptosystems. *Math. Comput.* **1987**, *48*, 203–209. [CrossRef]
70. Tiwari, H.D.; Kim, J.H. Novel Method for DNA-Based Elliptic Curve Cryptography for IoT Devices. *ETRI J.* **2018**, *40*, 396–409. [CrossRef]
71. Sengupta, A.; Ray, U.K. Message mapping and reverse mapping in elliptic curve cryptosystem. *Secur. Commun. Netw.* **2016**, *9*, 5363–5375. [CrossRef]
72. King, B. Mapping an Arbitrary Message to an Elliptic Curve When Defined over GF (2^n). *IJ Netw. Secur.* **2009**, *8*, 169–176.
73. Rahman, A.N.M.B. We Don't Need StringBuilder for Simple Concatenation-DZone Java. 2019. Available online: Https://dzone.com/articles/string-concatenation-performacne-improvement-in-ja (accessed on 19 January 2020).

74. Klarreich, E. Multiplication Hits the Speed Limit. *Commun. ACM* **2019**, *63*, 11–13. [CrossRef]