EW_{256357}: A New Secure NIST P-256 Compatible Elliptic Curve for VoIP Applications' Security

Nilanjan Sen[1(✉)], Ram Dantu[2], and Kirill Morozov[2]

[1] Western Illinois University, Macomb, IL 61455, USA
N-Sen@wiu.edu
[2] University of North Texas, Denton, TX 76207, USA
{ram.dantu,kirill.morozov}@unt.edu

Abstract. Selection of a proper elliptic curve is the most important aspect of Elliptic Curve Cryptography (ECC). Security of ECC is based on the Elliptic Curve Discrete Logarithm Problem which is believed to be unsolvable. Some of the well-known elliptic curve standards are NIST FIPS 186-2, Brainpool, and ANSI X9.62. Among these, NIST-recommended curves are a popular choice for industrial applications, in particular, for Internet security as a part of TLS/SSL, and even in real-time media encryption which uses Voice over IP (VoIP) technology. Specifically, NIST P-256 curve is widely used in these applications. Some NIST curves have disadvantages related to security issues, and therefore it is important to search for secure alternatives. In our work, we propose a new secure short Weierstrass curve EW_{256357} at the 128-bit security level and compare it with the NIST P-256 curve. Our proposed curve is compatible with NIST P-256 curve but features better security. Based on the performance analysis of related curves in our previous and present works in terms of delay and jitter, we say that our proposed curve is suitable for the real-time media encryption.

Keywords: Elliptic curve · Cryptography · ECC · Security

1 Introduction

Elliptic Curve Cryptography (ECC) is a very popular asymmetric encryption scheme proposed separately by Victor Miller [28] and Neal Koblitz [24] in 1985. Since Elliptic curves (EC) have smaller key size and better computational efficiency compare with other asymmetric key encryption techniques such as RSA, the ECC based key exchange protocols, digital signatures, and message encryption have become very popular in the computer and network security field.

Different types of ECs of different security levels are available for commercial or research purposes. Some of the popular ECs are developed by organizations such as National Institute of Standards and Technology (NIST), Teletrust and

N. Park et al. (Eds.): SecureComm 2020, LNICST 336, pp. 291–310, 2020.
https://doi.org/10.1007/978-3-030-63095-9_19

so on. Usually the ECs are used for key exchange and digital signatures. Elliptic Curve Diffie-Hellman Key Exchange (ECDHE) and Elliptic Curve Digital Signature Algorithm (ECDSA) are some well-known applications based on Elliptic curves. Some elliptic curves (other than NIST curves) are Brainpool curves [2], "Curve25519" curve developed by Bernstein [10], "Ed448-Goldilocks" curve developed by Hamburg [21] and so on. NIST's elliptic curves are considered to be the most popular elliptic curves. Some of them are prime curves and some of them are binary curves [1]. However, there are some drawbacks to NIST-recommended curves including debates about the possible presence of backdoor in the NIST-recommended curves [23,25]. These issues drive the researchers to develop more secure elliptic curves. In this work, we have developed a new elliptic curve EW_{256357} at the 128-bit security level, where E stands for Elliptic curve, W stands for Weierstrass curve, and 256357 represents the primary number used in this curve, i.e., $2^{256} - 357$. Our proposed curve is more secure compare with NIST P-256 curve with respect to the security parameters of Elliptic Curve Discrete Logarithm Problem (ECDLP) and some "ECC security" features (relevant to short Weierstrass curves) [12] which are discussed in Sect. 7.1. Based on our present work, we can say that our proposed curve provides better security to VoIP media, which means real-time audio and video.

2 Related Work

The most popular forms of EC are Weierstrass form, Edward form and Montgomery form. Multiple EC standards are available such as SEC2, NIST FIPS 186-2, Brainpool etc. which use one of these EC forms. NIST standards are more popular for industrial applications, but due to some drawbacks in NIST-recommended curves, researchers are developing more secure and efficient curves in terms of EC operations and security. Brainpool curves, developed by Teletrust, are generated over the prime field, use short Weierstrass form and use pseudo-random prime numbers [2]. Bernstein has proposed "Curve25519", a Montgomery curve which uses an efficient Montgomery ladder for ECDHE operation [10]. An Edward curve developed by Hamburg is "Ed448-Goldilocks". It uses Solinas trinomial prime for fast arithmetic operations in both 32 and 64-bit machines [21]. J. W. Bos et al. of Microsoft Research have proposed a set of prime order Weierstrass and (twisted) Edward elliptic curves and analyze them from a performance and security perspective. Their Weierstrass curves are backward compatible with current prime order NIST curves [18]. Their proposed curves provide high security by supporting constant time, exception-free scalar multiplication. The proposed curves' arithmetic operations are faster than the corresponding NIST curves.

3 Motivation

In this work, we propose a short Weierstrass prime curve at the 128-bit security level. A similar type of very popular elliptic curve viz. NIST P-256 curve is

widely used but it has some serious drawbacks (discussed in Subsect. 3.1) which motivate us to propose a more secure elliptic curve. In addition, we are interested in real-time media encryption which uses Voice over IP (VoIP) technology. Use of VoIP applications is proliferating in every sector of our daily life such as real-time video streaming, live audio, video-on-demand, webinars and so on. Recent COVID-19 pandemic has increased the use of real-time applications immensely throughout the world. The security of these applications has become a major concern, as well. ECC can be a better choice in this context.

3.1 Some Drawbacks of NIST Curves

- The NIST curve generation process is not fully transparent. The generation of the second curve coefficient b is not justified. NIST uses a random 160-bit seed value and a hash function to generate b. Some researchers fear that if NSA knows any weakness of the curves then the security of the curves may be compromised [13].
  Our curve generation process is fully transparent.
- The so-called "twist security" is an important security feature of ECC [12], the lack of which may lead to twist-security attack. A curve is twist secure if its order and its corresponding twist curve's order are prime [22]. The NIST P-256 curve does not qualify this, so it is weakly twist secure. Bernstein and Lange have also mentioned NIST P-256 curve as a weakly twist secure curve in [13].
  Our proposed curve is twist-secure.
- Bernstein and Lange [13] mentioned that, though OpenSSL is the most popular open-source library (which is used for NIST curves implementation), there are several reports regarding the cryptographic failures in OpenSSL and many of them are still not fixed due to implementation difficulty [13]. OpenSSL team has improved their code quality a few years ago [20] but it is not reported by the coders or researchers yet whether all those issues are resolved or not.
  In our experiment, we have used Java cryptography library "BouncyCastle" to implement our proposed curve.
- Bernstein and Lange [11] also claimed that "NIST's ECC standards create unnecessary complexity in ECC implementations".
  Our proposed curve implementation process is simple and straight forward in nature.
- There have been questions about probable presence of a backdoor in the Dual Elliptic Curve Deterministic Random Bit Generator used in the NIST elliptic curve generation process. This serious matter had been revealed due to the leakage of some NSA documents [23,25]. Steve Lipner, one of the committee members of the Committee of Visitors which was formed to review NIST cryptographic processes, recommended in his report that "NIST should ensure that there are no secret or undocumented components or constants in its cryptographic standards whose origin and effectiveness cannot be explained. (Transparency of product) This recommendation would preclude the issuance

of the EES with its reliance on the then' secret Skipjack algorithm as well as the Dual_EC_DRBG with its reliance on an elliptic curves whose origin was undocumented and whose security could not be verified." [5]. Though the Dual_EC_DRBG is now excluded from the standards, still there are doubts over the other pseudo-random generators used to generate NIST curves, especially after Edward Snowden's revelation [8].
Our curve generation process is transparent and fully described.

In our previous work [26] we showed that the use of ECC based encryption schemes would ensure better real-time media protection. We implemented the ECC encryption scheme in the real-time VoIP systems, and we showed that the ECC implementation worked well in VoIP applications without degrading the quality of the VoIP data. The delay and jitter values were within the range as per the ITU-T G.114 recommendation [4]. Furthermore, we analyzed the performance of the curves in terms of the video data rate as well. The video data rate was the ratio of the received video data rate to that video frame's play time. It was measured in bytes per second. Usually, the high video data rate yields better video quality. In our experiment [26], we noticed that during the video transmission, the video data rates were nearly similar in both AES-encrypted and NIST P-256 curve encrypted streams. Figure 1 [26] depicts the data rate comparison of AES-encrypted video streams and ECC-encrypted video streams using NIST curves. We can see that NIST 256-bit elliptic curve encrypted video streams' data rates are nearly same as AES-encrypted video streams' data rates, but if the EC key size increases, the data rate decreases for the first few frames (e.g. NIST 571-bit curve), that deteriorates the video quality. Hence, 256-bit curve will be more suitable for VoIP media encryption purpose.

This performance has motivated us to develop a new curve which is compatible with the NIST P-256 curve and features better security. *The word "compatible" is used in the sense that our proposed curve will be compatible with the applications which support NIST P-256 curve.* In this work, we have tested and confirmed that our proposed 256-bit curve is suitable for real-time media encryption. We have shown that the proposed curve's performance is faster compare with the NIST P-256 curve in terms of basic elliptic curve arithmetic operations which are essential for elliptic curve key generation and encryption/decryption tasks. We have also compared the performance of the proposed curve with the NIST P-256 curve in terms of real-time audio and video encryption. Furthermore, We have proved that the proposed curve is more secure than NIST P-256 curve.

4 Curve Generation

NIST recommends two types of curves for ECC, the pseudo-random prime curves over GF(p) where p is the prime number, and binary curves over GF(2^m) where m is the degree of the field extension. The coefficients of these types of curves are generated using a cryptographic hash function. NIST recommends the equation

$$y^2 = x^3 - 3x + b \ (mod\ p) \tag{1}$$

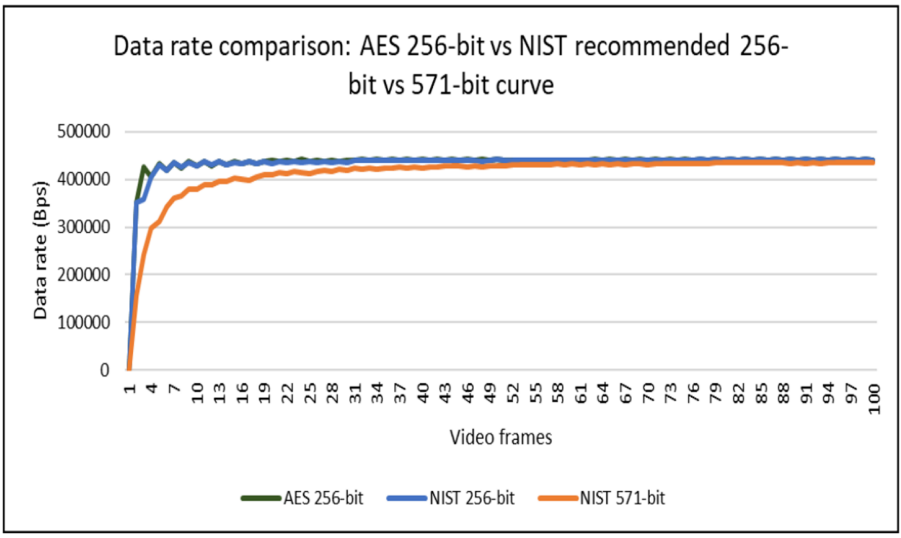

Fig. 1. Data rate comparison between AES and two NIST-recommended curves. Video data rate of ECC NIST 256-bit encrypted video stream is more than that of NIST 571-bit key for first 50–55 frames.

for the pseudo-random prime curve over GF(p). The coefficients of these curves are selected in a way such that the elliptic curve operations' efficiency can be optimized [1].

In this work, we have considered prime curves only. Researchers prefer prime curves over binary curves as the former is more reliable because the discrete logarithm problem can be solved on the binary curves [16]. In general, six parameters are used to generate the prime curves over GF(p) which is represented in a six-tuple notation $< p, r, a, b, G, h >$ where:

1. p is a prime number;
2. r is the prime order of the curve;
3. The first curve coefficient a;
4. The second curve coefficient b;
5. G is the base point where x, y are the coordinates of the base point, denoted as G_x and G_y;
6. h is the cofactor of the curve and h is usually 1.

There are three types of elliptic curves that are commonly used: short Weierstrass curves, twisted Edward curves and Montgomery curves. The short Weierstrass model supports all curves which are defined over large prime fields. At the same time, the twisted Edwards and Montgomery models support only a subset of elliptic curves [18]. In our work, we have used the short Weierstrass model to generate the new curve.

Security strength or security level is the most important aspect of cryptography. By stating that the security strength is "n-bit", we imply that 2^n operations

are required to break the system. The security strength of a symmetric key cryptosystem is generally measured as its key size (assuming that no better attacks than brute-forcing the keys are known). For instance, the security strength of AES encryption with the 128-bit key is considered to be 128 bits. At the same time, the security strength of asymmetric key cryptosystems is normally smaller than their key size, due to the existence of attacks that are more efficient than brute-force attacks. For instance, ECC generally requires a 256-bit key to support the 128-bit security level. And in particular, this is the case for our proposal, which has a 256-bit key and a 128-bit security level.

5 Methodology

We have proposed a short Weierstrass curve of the form $y^2 = x^3 - 3x + b$ because this form is used in all NIST-recommended prime curves. We have chosen short Weierstrass prime field curve because most of the elliptic curves are defined over prime fields [19] without sacrificing any security bit which is very important for ECDLP security [18]. This work is solely based on how we can generate an elliptic curve suitable for VoIP media encryption which will be more secure than NIST P-256 curve. Since the NIST P-256 is very popular and widely used curve which uses short Weierstrass form, we have developed a similar type of curve for the compatibility, that means our curve can be implemented in those applications easily which support NIST P-256 curve. No Montgomery or Edward curve is considered in this work for this reason.

Elliptic curve generation procedure includes choosing different curve parameter values. The basic idea of choosing elliptic curve parameter values comes from the work of J. W. Bos et al. [18]. A very important task for elliptic prime curve generation is to choose a prime number. Brainpool curves use pseudo-random prime numbers to generate the prime curves but their performance is not as good as NIST curves. NIST curves use pseudo-Mersenne prime which supports efficient modular arithmetic like fast modular reduction [1]. J. W. Bos et al. [18] used two different forms of primes, pseudo-Mersenne primes, and Montgomery friendly primes. The pseudo-Mersenne primes have the property to make the elliptic curves indistinguishable compared to Montgomery-friendly prime curves, because in Montgomery-friendly prime p, $\frac{p+1}{2}$ is not close to 2^n for any value of n [14]. We have used pseudo-Mersenne prime for our proposed curve and have chosen the other parameters of the curve deterministically. The detailed methodology of other curve parameters' selection is described in the next section.

6 New Curve Parameter Selection

6.1 Prime Number Selection

Prime number generation is an important operation in ECC. That prime number is used for the curve's arithmetic modulus operations. Different EC standards use

different prime number generation methods. NIST recommends a special type of prime number which is known as generalized Mersenne number. The advantage of these prime numbers over general or random prime numbers is that the fast reduction can be carried out during modular multiplication [1].

There are different forms of primes used for Elliptic curves. One form is $2^{\alpha} - \gamma$. This type of prime is known as a pseudo-Mersenne prime. Another form of prime is $2^{\alpha}(2^{\beta} - \gamma) - 1$, known as a Montgomery friendly prime. The later reduces some computations. The logic behind the Montgomery friendly prime is based on the concept of Montgomery multiplication which states that during modular reduction operations, the division operations can be replaced by logical shift operations which are comparatively less expensive. In both forms α, β, and γ are positive integers [18].

Though Montgomery friendly primes reduce some computations, the pseudo-Mersenne prime curves are better in terms of indistinguishability as discussed in Methodology section [14]. So, in our work, we have used the pseudo-Mersenne prime of the form $2^{\alpha} - \gamma$. The value of α is 2 times the security level of the curve in terms of bits. For example, for 128-bit security, the value of α is 256. Thus, we can choose a very large prime in this way. Based on their work in [18], J. W. Bos et al. has proposed numsp256d1 curve [17] which uses the largest 256-bit pseudo-Mersenne prime number $2^{256} - 189$, and the property of this curve is already tested with the largest 256-bit pseudo-Mersenne prime. During the curve generation phase we chose the 256-bit second largest pseudo-Mersenne prime number $2^{256} - 357$ and compared its performance with $2^{256} - 189$. *We checked that this prime number was as good as* $2^{256} - 189$, *and hence we chose* $2^{256} - 357$ *for our proposed curve. The performance analysis of our proposed curve, NIST P-256 curve and numsp256d1 curves on basic elliptic curve operations are given in* Table 1 *and* Table 2. The chosen prime for our proposed curve is equivalent to 3 mod 4 for efficient modular operations [18]. To check the primality of this number, we have used Miller-Rabin primality test algorithm [6].

6.2 Order of the Curve

Every cryptographically strong prime curve contains a certain number of points, that number must be a prime. This number is known as the order of that curve. Determining the order of the curve for a very large prime is a time-consuming process. One very efficient algorithm, known as Schoof-Elkies-Atkin point counting algorithm based on Schoof's algorithm [27], is used to get the total number of points on that elliptic curve and hence the order of the curve over a finite field. The inputs of this algorithm are values of a, b and the prime p.

The order r is generally calculated as $r = p + 1 - t$, where t is known as trace, where $1 < |t| \leq 2\sqrt{p}$ [18]. In this work, we have used Schoof-Elkies-Atkin (SEA) point counting algorithm to find a prime order curve (i.e., the number of points present in the curve is prime). We have executed the algorithm with $a = -3$ and $b = 1$, and the value of b is incremented by 1 until the prime order is found.

Another feature of the proposed curve is that, it is twist secure, that means for a specific value of b, both $y^2 = x^3 - 3x + b$ and $y^2 = x^3 - 3x - b$ curves have prime order. The second curve is known as the twist of the first curve.

6.3 Coefficients of the Curve

We have chosen the value of the coefficient a as -3 because the elliptic curves of the form $y^2 = x^3 - 3x + b$ provide the fastest elliptic curve arithmetic [3]. NIST and Brainpool prime curves also have the same value for the coefficient a. The parameter b is chosen deterministically, and it belongs to the field F_p. As mentioned in the previous sub-section, we have found b while determining the order of the curve using SEA algorithm. The algorithm started with $b = 1$ and incremented b by 1 until it found the prime order of the curve. Most importantly, the coefficient b should support twist-security feature (discussed later). In our proposed curve, the value of b is 5029.

6.4 Choosing the Base Point

For different applications of the elliptic curve such as key generation or encryption, the base point plays a very important role. The two coordinates of the base point x and y should lie between 0 and p, where p is the prime order of the field F_p. To find the base point of the proposed curve, We started with $x = 1$. We have used Algorithm 1 to find the base point $G(x, y)$.

To check the validity of x, we computed the Legendre symbol $(c|p)$ where $c|p = c^{\frac{p-1}{2}} \ (mod\ p)$. The Legendre symbol finds whether a number is a quadratic residue modulo an odd prime. As per the rule, if the $(c|p)$ value is either 0 or -1, then c is not a quadratic residue, so we need to repeat the process with the next value of x after incrementing it by 1. To find a valid coordinate x, we used Cipolla's algorithm to find the coordinate y by calculating $y^2 \equiv c \ (mod\ p)$. Cipolla's algorithm is used to solve the congruence of the form $x^2 \equiv n \ (mod\ p)$. Since the equation was quadratic, y had two roots. We chose the smallest one. After obtaining y, we checked whether $0 < y < p$, and then selected $G(x, y)$ as the base point.

6.5 Cofactor

The cofactor of the elliptic curve is the ratio of the order of the curve and the order of the base point. We know that the order of the curve is the total number of points that lie on the curve. On the other hand, the order of the base point G of that curve is the smallest value n for which $n.G = \infty$, where ∞ is the point at infinity. For any elliptic curve E over a finite field F_p, if the order of the curve is $\#E(F)$ and the order of the base point is m, then cofactor h is calculated as $\frac{\#E(F)}{m}$. The value of the cofactor should be very small and an integer, that means $\#E(F)$ should be divisible by m. As per the NIST specification, the cofactor value should be 1, 2, or 4, and the cofactor of the prime curves must

Algorithm 1: Choosing the base point $G(x, y)$

Input: The prime number p, two curve coefficients a and b, and x coordinate of the base point
Output: y coordinate of the base point
$x \leftarrow 1$;
while *true* **do**
  $c \leftarrow x^3 + ax + b \ (mod\ p)$;
  $l \leftarrow c^{\frac{p-1}{2}} \ (mod\ p)$;
  if $l = 0$ *or* $l = -1$ **then**
    $x \leftarrow x + 1$;
  else
    break the loop;
Solve $y^2 = c \ (mod\ p)$ using Cipolla's algorithm;
y has two roots. Select the smallest one;

be 1 [1]. Our proposed curve's cofactor is 1 also. That means the order of the proposed curve and the order of the base point are same.

The curves with cofactor 1 are secure against the well-known small-subgroup attack on Diffie-Hellman. In the small-subgroup attack, the attacker sends a point of small order as his public key to a legitimate user. The user then computes the shared secret key using his private key and the attacker's public key. Since the attacker's public key has small order, the secret key does not have many possibilities, so the attacker can get the information of the user's private key from the secret key. This type of attack can be prevented by the elliptic curves with cofactor 1 [12]. Since the proposed curve's co-factor is 1, it is secure against the small-subgroup attack.

6.6 Proposed New Curve

Based on the requirements and methodology defined above, we have generated a 256-bit twist secure new curve at 128-bit security level whose parameters are given below:

The curve (EW_{256357}): $y^2 = x^3 - 3x + 5029$ over F_p

1. Prime number $p = 115792089237316195423570985008687907853269984665640564039457584007913129639579$
2. Order of the curve $r = 115792089237316195423570985008687907852793585971461506558239498229566154872651$
3. $a = -3$
4. $b = 5029$
5. $G_x = 1$
6. $G_y = 10139468005879303417234759080642520107517648300263810609498332653928648296 36$
7. Cofactor $h = 1$

7 Performance Analysis of the New Curve

7.1 Performance Comparison Between EW_{256357} and NIST P-256 Curves Regarding Different Elliptic Curve Operations

We have successfully tested the new curve to check its performance in different elliptic curve operations viz. point addition, point doubling, scalar multiplication, fixed-window based scalar multiplication, Elliptic Curve Diffie-Hellman Key Exchange (ECDHE) and Elliptic Curve Digital Signature Algorithm (ECDSA) for constant execution time, and have compared the performance of the proposed curve with NIST P-256 curve. The corresponding execution times are also shown for numsp256d1 curve proposed by J. W. Bos et al. [17]. The system configuration used for this experiment was Intel 3.40 GHz Core i7-3770 processor, 16 GB main memory, and 64-bit Ubuntu OS. The code was written in Java to implement the curves and the other elliptic curve arithmetic operations. Each operation was executed 1000 times and average execution time was calculated for each of them. The comparison results are given in Table 1 and Table 2, the execution time unit is millisecond.

Table 1. Execution time comparison of different elliptic curve operations (in milliseconds)

Curve	Point addition	Point doubling	Scalar multiplication	Fixed window- based scalar multiplication
EW_{256357}	0.0169	0.0196	6.9298	5.7132
NIST P-256	0.0198	0.0254	6.9564	5.7505
numsp256d1	0.0200	0.0269	7.1362	5.9063

Table 2. Execution time comparison of ECDHE and ECDSA operations (in milliseconds)

Curve	ECDHE	ECDSA
EW_{256357}	12.2958	18.5187
NIST P-256	12.3144	18.5280
numsp256d1	12.3964	18.6591

Scalar multiplication is a very important operation required for elliptic curve key generation and encryption/ decryption operations. Its execution time varies depending on the size of the secret scalar value. An adversary can guess the secret scalar by exploiting that information using side-channel attacks [18]. Fixed

window-based scalar multiplication is one of those scalar multiplication algorithms used for constant execution time irrespective of the size of the secret scalar. In this experiment, the window size of the fixed window-based scalar multiplication is 4.

The proposed curve has performed better in basic elliptic curve arithmetic operations, but the execution time differences are not much in two curves. The execution times are so close because the proposed curve and the NIST P-256 curve use short Weierstrass form where the value of the coefficient a is -3. The values of the coefficient b are different for all three curves, but that coefficient is not used in any of the elliptic curve operations. For instance, the two basic elliptic curve arithmetic operations are point addition and point doubling. If we consider two points $P(x_p, y_p)$ and $Q(x_q, y_q)$ on the elliptic curve E where x_p, y_p and x_q, y_q are the x and y coordinates of P and Q respectively, then the addition of P and Q will yield another point $R(x_r, y_r)$ on E. The point R can be calculated as follows [9]:

$$x_r = \left(\frac{y_q - y_p}{x_q - x_p}\right)^2 - x_p - x_q \tag{2}$$

$$y_r = \left(\frac{y_q - y_p}{x_q - x_p}\right).(x_p - x_r) - y_p \tag{3}$$

On the other hand, the point doubling operation, where $P + P = 2P$, $P(x_p, y_p)$ and $2P(x_d, y_d)$ are the points on E and $P \neq -P$, can be calculated as follows [9]:

$$x_d = \left(\frac{x_p^2 + a}{2y_p}\right)^2 - 2x_p \tag{4}$$

$$y_d = \left(\frac{x_p^2 + a}{2y_p}\right).(x_p - x_d) - y_p \tag{5}$$

where a is the coefficient of the short Weierstrass form.

From the last four equations we can see that the coefficient b of short Weierstrass form is not required in the two basic elliptic curve arithmetic operations, the point addition and point doubling, and these two arithmetic operations are used in elliptic curve scalar multiplication, ECDHE and ECDSA. This is the reason why the execution times of different EC arithmetic operations are similar in the two curves. *We have stated earlier that the proposed curve is compatible with the NIST P-256 curve. The primary goal of this work is to propose an elliptic curve which is suitable for VoIP media encryption and more secure than NIST P-256 curve.*

7.2 Comparison Between EW_{256357} and NIST P-256 Curve Regarding ECDLP and ECC Security

D. J. Bernstein and T. Lange have jointly proposed criteria for Elliptic Curves which are safe for cryptographic use [12]. *Since the proposed curve is a prime curve and we have used the short Weierstrass equation, we have considered only those criteria suitable for this curve.* we have considered the Elliptic Curve Discrete Logarithm Problem (ECDLP) security and Elliptic Curve Cryptography (ECC) security criteria to test the proposed curve and compare it with NIST P-256 curve. The comparison result is shown in Table 3.

Though ECDLP is an important property which ensures the security of the ECC, in reality the attackers can break ECC without solving ECDLP [12]. For example, some standard curves may be vulnerable because they may not work properly for some rare curve points, they may reveal secret data for some points which do not lie on those standard curves, or attackers may get secret information through branch or cache timing attacks. These may happen because ECDLP security is more theoretical than practical [12]. Since NIST curves are considered to be the standard curves, they also have the aforementioned problems. Thus, apart from the ECDLP security, we have to ensure ECC security as well. The proposed curve satisfies all required criteria of both ECDLP and ECC security, while the NIST P-256 curve does not satisfy all of the criteria. The security parameters that we have considered are discussed below. The first four parameters are ECDLP security parameters and the last one is the ECC security parameter.

- **Rho Complexity:** The Pollard Rho method is one of the methods used to break ECDLP. Rho complexity is calculated as $\log_2(\sqrt{\pi/4}.\sqrt{r})$ or $0.886\sqrt{r}$ where r is the order of the curve [18]. ECDLP can be broken easily if the value of r is small, because $0.886\sqrt{r}$ number of additions are required to break it. Hence, the value of r should be very large. Since the proposed curve is a 256-bit curve, the Rho complexity of the curve is $2^{127.8}$ as in the NIST P-256 curve, which is large enough for ECDLP security.
- **Transfer:** It is another method where ECDLP can be converted to a linear algebraic group discrete logarithm problem. One of the transfer methods is

Table 3. Comparison of two curves regarding ECDLP and ECC security

Criteria	EW_{256357}	NIST P-256
ECDLP security		
Rho complexity	$2^{127.8}$	$2^{127.8}$
Transfers	Yes	Yes
CM field discriminants	Yes	Yes
Rigidity	Fully rigid	Manipulative
ECC security		
Twist security	Strongly secured	Weakly secured

the Additive Transfer method, where the prime number used to generate the curve over the prime field and the order of that curve are same. Attackers can exploit this to break the ECDLP which is also known as Smart-ASS attack [12]. The proposed curve's prime number and the order of the curve are different, so it is not vulnerable to this attack.
Second type of transfer is the Multiplicative Transfer method where $(p^n - 1)$ is divisible by r, where p is the prime number, r is the order of the curve and n is an integer, $n \geq 1$. The minimum value of n for which $(p^n - 1)$ is divisible by r is known as the embedding degree of the group. If the embedding degree is small, then the curve becomes vulnerable to MOV (Menezes-Okamoto-Vanstone) attack [12]. This attack uses Weil pairing to convert the discrete logarithm problem on the points of an elliptic curve to a discrete logarithm problem on finite fields which can be solved easily [7]. Therefore, the embedding degree value should be very large to prevent MOV attacks. Different standard curves support different embedding degrees. The proposed curve supports embedding degrees large enough to prevent MOV attacks.

– **Complex-Multiplication (CM) field discriminants:** This parameter is applicable to the elliptic curves which have very large endomorphism ring. This is used to find the elliptic curve with proper order. The order of the elliptic curve should be prime for a cryptographically secure elliptic curve. If p is the prime number of the prime group F_p, then the order of the group $r = p + 1 - t$ where t is known as the trace of the curve. According to the Hasse's theorem, $-2\sqrt{p} \leq t \leq 2\sqrt{p}$ [12].
Algorithm 2 is used to find the CM field discriminant D [12]:
The value of D is negative. So, the absolute value of D should be large enough to ensure the ECDLP security. As per Bernstein and Lange's approach, the absolute value of D should be more than 2^{100} [12]. The proposed curve's D value is more than 2^{100}.
– **Rigidity:** This feature prevents the generation of multiple curves from a specific curve generation process. If the curve generation process is not rigid, the attackers can generate many curves using that process and can choose a weak curve which is vulnerable to the secret attack. The curve generation process is fully rigid if it explains the generation of all parameters in detail, for instance curve equation, the coefficient of the curve, the base point selection criteria, etc. If the manipulatable curve generation process is not transparent, attackers can potentially generate multiple curves from it.
Our curve generation process is fully rigid, because we have explained the parameter generation procedures of our proposed curve. At the same time, the NIST P-256 curve generation process is manipulative, because this curve uses a hash function and a 160-bit random seed to generate the coefficient b. The random seed is unexplained, so the hash function may be very strong, but the attackers may try to find a specific seed value from a large set of seed values to find vulnerability for the secret attack [12].
– **Twist security:** Twist security is very important feature with respect to ECC security. It is required to prevent some well-known attacks such as Invalid-curve attacks [12]. Invalid-curve attacks were proposed by Biehl,

Algorithm 2: Calculating CM field discriminant

Input: The prime number p, and trace t
Output: The CM field discriminant D
find the largest integer m where $t^2 - 4p$ is divisible by m^2;
if $\frac{t^2-4p}{m^2} \ (mod\ 4) = 1$ **then**
  $D \leftarrow \frac{t^2-4p}{m^2}$;
else
  $D \leftarrow 4.\frac{t^2-4p}{m^2}$;

Meyer and Müller [15]. In this attack, the attacker sends a point Q of small order of another curve to the legitimate user. If the user computes the shared secret key by multiplying his private key n and the point Q, this may reveal the information about the user's private key. For instance, the attacker can send many such invalid points of small order like 2, 3, 4, etc. After getting the corresponding secret keys, he will get the values of $n\ mod\ 2$, $n\ mod\ 3$ and so on. By using Chinese Remainder Theorem, the attacker can compute the private key of the user. A twist secure curve can prevent such attacks.
A curve E is said to be twist secure if both E and its twist $E^{/}$ are cryptographically strong, and the minimum criteria is that their orders must be prime [22]. According to Hasse-Weil theorem, if $y^2 = x^3 + ax + b$ is an elliptic curve E over a prime field F_p, p is a prime number and $p > 3$, then the order of the curve $E = p + 1 + t$, where t is known as trace of Frobenius, and $|t| \leq 2\sqrt{p}$. The corresponding twisted curve of E defined over F_p, represented as $E^{/}$, has its order calculated as $p + 1 - t$.
Our proposed curve is twist secure. The equation of the curve is

$$E : y^2 = x^3 - 3x + 5029 \tag{6}$$

The value of the trace of Frobenius of the proposed curve is
t = 476398694179057481218085778346974766929.
The corresponding twist of the curve is

$$E^{/} : y^2 = x^3 - 3x - 5029 \tag{7}$$

The orders of both E and $E^{/}$ curves over F_p are prime. But the NIST P-256 curve is not a strongly twist secure curve. The order of the NIST curve $y^2 = x^3 - 3x + b$ is prime (the value of b is given in [1]), but the order of its corresponding twist curve $y^2 = x^3 - 3x - b$ is not prime. So, the NIST P-256 curve has weak twist security. Bernstein and Lange have also mentioned in [13] that NIST P-256 has weaker twist.

7.3 Performance Comparison Between EW_{256357} and NIST P-256 Curves Regarding Real-Time Audio and Video Encryption Using VoIP

We have tested the performance of our proposed curve on the real-time audio and video encryption and compared the performance with the NIST P-256 curve. The experiments are done using client-server architecture. We have used Ubuntu 16.04 (64 bits) operating system on both the server and the client machines. Java and Bouncy Castle (an open-source lightweight cryptography API for Java) are used for curve implementation, key generation and to implement the encryption and decryption processes. We have used Elliptic Curve Integrated Encryption Scheme (ECIES), which was initially proposed by Bellare and Rogaway and later modified by Shoup [9,29], to encrypt the VoIP packets. Usually, asymmetric key encryption schemes are not directly used for message encryption. But for this experiment, we have encrypted RTP payload using the NIST P-256 curve and our proposed curve to analyze their performance on VoIP applications. In the experimental setup, the server is connected to the network through the Ethernet connection and the client is connected through the institutional Wi-Fi. The audio and video encryption systems follow the same methodology where the server first receives the client's ephemeral elliptic curve public key. The server then encrypts chunks of data using that key and sends the encrypted chunks one by one to the client. The client decrypts the encrypted payload using its ephemeral elliptic curve private key. The encrypted stream is sent to the client as a Real-time Transport Protocol (RTP) payload. UDP is used as the transport layer protocol. MJPEG and WAV files are used for video and audio encryption respectively. The machine configurations are given below:

Server Configuration:

1. Intel 3.40 GHz Core i7-3770 processor
2. 16 GB Main memory

Client Configuration:

1. Intel 2.50 GHz Core i5-2450M processor
2. 6 GB Main memory

We have measured the performance of the curves based on the end-to-end delay and the jitter in milliseconds. We have conducted 5 experiments for each curve (for audio and video encryption) and then calculated the mean of the corresponding results. The end-to-end delay is calculated by adding network latency from the server to the client, the encryption time at the server-end and the decryption time at the client-end. The corresponding results are depicted in Fig. 2, 3, 4 and Fig. 5.

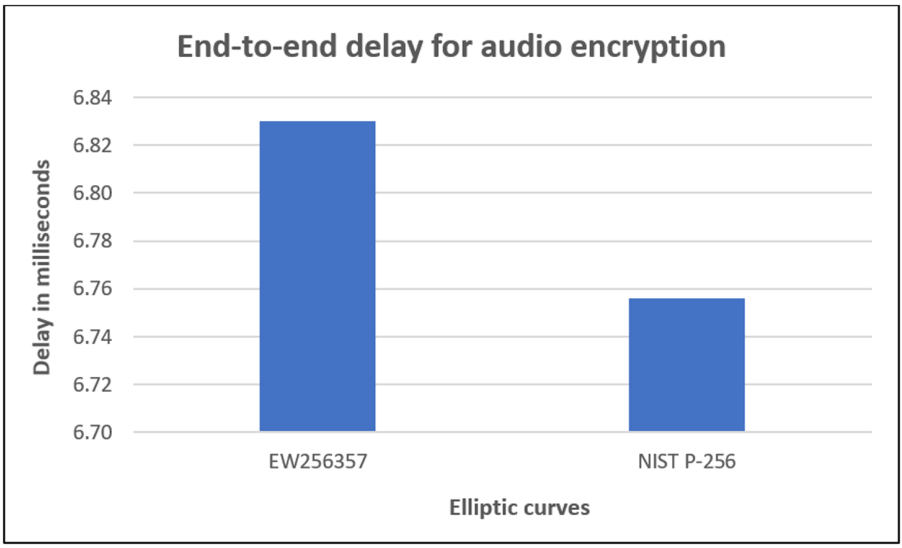

Fig. 2. End-to-end delay graph for audio encryption using EW_{256357} and NIST P-256 curves. The delay of NIST P-256 curve is less than the EW_{256357} curve, but the delay difference between EW_{256357} and NIST P-256 curve is only 0.07 ms.

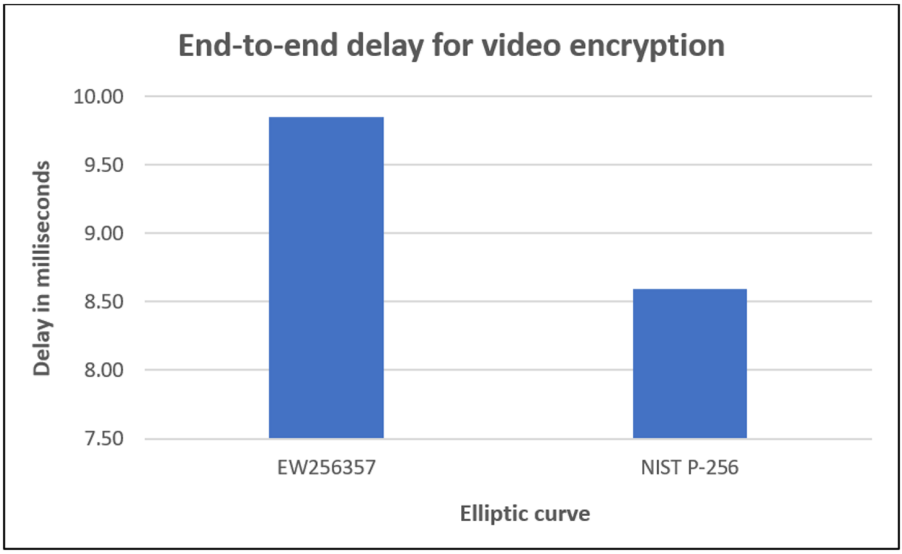

Fig. 3. End-to-end delay graph for video encryption using EW_{256357} and NIST P-256 curves. The delay of NIST P-256 curve is less than the EW_{256357} curve, but the delay difference between EW_{256357} and NIST P-256 curve is only 1.25 ms.

From the two end-to-end delay graphs for audio and video encryption in Fig. 2 and Fig. 3, we can say that the NIST P-256 curve's end-to-end delay is less than the proposed curve. However, the delay differences between the proposed curve and the NIST P-256 curve is only 0.07 ms in audio encryption and 1.25 ms in video encryption, which are negligible. Figure 4 and Fig. 5 depict the jitter plots of two curves. The jitter value of our proposed curve is less than the NIST P-256 curve. The NIST P-256 curve has 0.03 ms and 0.04 ms more jitter than our proposed curve in audio and video encryption, respectively.

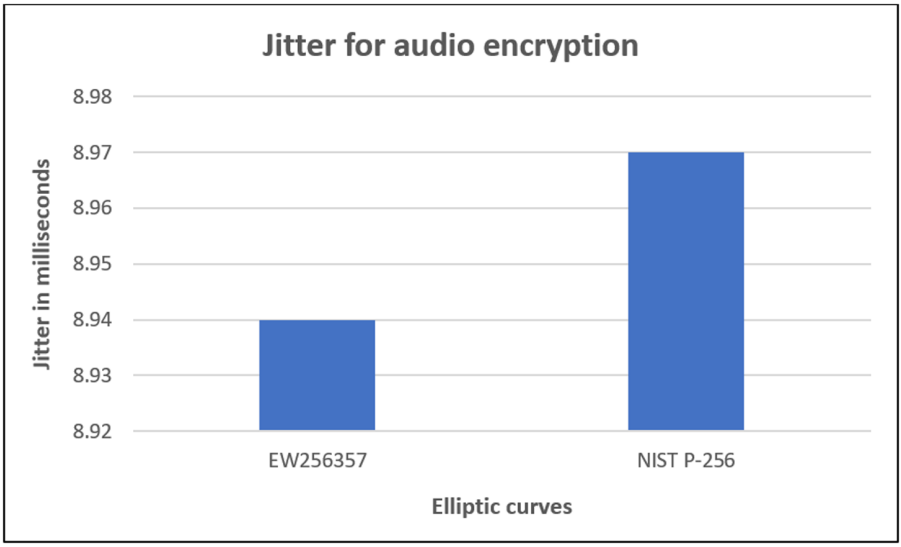

Fig. 4. Jitter graph for audio encryption using EW_{256357} and NIST P-256 curves. The jitter value of NIST P-256 curve is more than EW_{256357} and the difference is 0.03 ms.

The experiments show that our proposed curve's execution time is faster than the NIST P-256 curve in basic EC arithmetic operations. The network jitter is also low in our proposed curve. Our proposed curve's end-to-end delay time is little more than the NIST P-256 curve, but that issue is not as important as far as the curve's security is concerned. Since this work's primary focus is to develop a more secure curve than the NIST P-256 curve, we can see that our proposed curve provides more security than the NIST P-256 curve, and its performance is on a par with the NIST P-256 curve. This experiment proves that the EW_{256357} curve is a better choice for VoIP applications in security and overall performance.

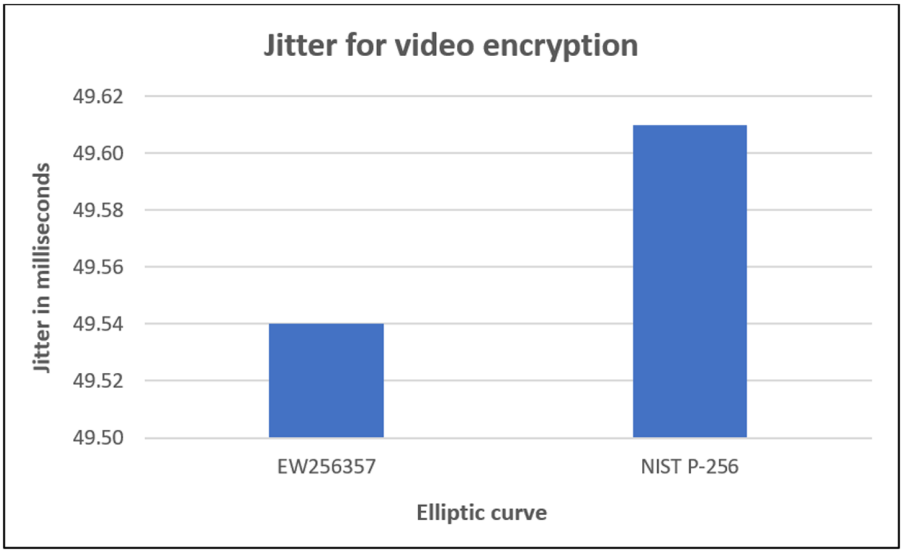

Fig. 5. Jitter graph for video encryption using EW_{256357} and NIST P-256 curves. The jitter value of NIST P-256 curve is more than EW_{256357} and the difference is 0.07 ms.

8 Conclusion

In this work, we have discussed the generation of a new 256-bit elliptic curve EW_{256357} at 128-bit security level, which is more secure than NIST P-256 curve. Our work is based solely on generating a better short Weierstrass curve than the NIST 256-bit curve in terms of security. We have not considered other types of curves like Edward or Montgomery curves. We have tested the performance of our proposed curve and compared it with NIST P-256 curves. We have also shown that the proposed curve's parameters are well explained, and the entire curve generation process is transparent, whereas the NIST P-256 curve generation process is manipulative. The proposed curve is a strongly twist-secure curve but the NIST P-256 curve is not. Based on the performance analysis of related curves, we can conclude that our proposed curve is suitable and a better choice for VoIP media encryption.

The elliptic curves of short Weierstrass form do not support Edward additions. This operation is supported by the Edward curves. Some ECC security features such as completeness or indistinguishability are supported by the Montgomery and Edward curve [12]. Montgomery curves support a scalar multiplication method known as Montgomery ladder which is much faster than the standard multiplication methods used by short Weierstrass curves because unlike short Weierstrass curves, Montgomery ladder method requires only x-coordinate of the elliptic curve point for the scalar multiplication [12]. In the future, we will extend this work to develop Montgomery and Edward curves which will support more ECC security features and will perform better than the existing curves.

Acknowledgement. This research is based upon work supported by the National Science Foundation under awards 1241768 and 1637291.

References

1. Digital Signature Standard (DSS). Federal Information Processing Standards Publication 186–4. https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.186-4.pdf
2. ECC Brainpool. ECC Brainpool Standard Curves and Curve Generation. https://www.teletrust.de/fileadmin/files/oid/oid_ECC-Brainpool-Standard-curves-V1.pdf
3. IEEE 1363–2000: Standard specifications for public key cryptography. https://standards.ieee.org/standard/1363-2000.html
4. ITU-T, Series G: Transmission Systems and Media, Digital Systems and Networks. https://www.itu.int/rec/T-REC-G.114-200305-I
5. Report and Recommendations of the Visiting Committee on Advanced Technology of the National Institute of Standards and Technology. https://www.nist.gov/sites/default/files/documents/2017/05/09/VCAT-Report-on-NIST-Cryptographic-Standards-and-Guidelines-Process.pdf
6. Rabin, M.O.: Probabilistic algorithm for testing primality. J. Number Theory **12**, 128–138 (1980)
7. Lynn, B.: Elliptic Curves - The MOV attack. https://crypto.stanford.edu/pbc/notes/elliptic/movattack.html
8. Hales, C.: The NSA Back Door to NIST. Not. AMS **61**(2), 190–192
9. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer, Heidelberg (2004). https://doi.org/10.1007/b97644
10. Bernstein, D.J.: Curve25519: new Diffie-Hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006). https://doi.org/10.1007/11745853_14
11. Bernstein, D.J., Lange, T.: Failures in NIST's ECC standards. https://cr.yp.to/newelliptic/nistecc-20160106.pdf
12. Bernstein, D.J., Lange, T.: SafeCurves: choosing safe curves for elliptic-curve cryptography. https://safecurves.cr.yp.to. Accessed 20 June 2020
13. Bernstein, D.J., Lange, T.: Security dangers of the NIST curves. https://cr.yp.to/talks/2013.05.31/slides-dan+tanja-20130531-4x3.pdf
14. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings. In: ACM Conference on Computer and Communications Security
15. Biehl, I., Meyer, B., Muller, V.: Differential fault attacks on elliptic curve cryptosystems. In: Annual International Cryptology Conference, pp. 131–146 (2000)
16. Faugère, J.-C., Perret, L., Petit, C., Renault, G.: Improving the complexity of index calculus algorithms in elliptic curves over binary fields. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 27–44. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_4
17. Bos, J.W., Costello, C., Longa, P., Naehrig, M.: Specification of curve selection and supported curve parameters in MSR ECCLib. https://www.researchgate.net/publication/281897794_Specification_of_Curve_Selection_and_Supported_Curve_Parameters_in_MSR_ECCLib
18. Bos, J.W., Costello, C., Longa, P., Naehrig, M.: Selecting elliptic curves for cryptography: an efficiency and security analysis. J. Cryptographic Eng. **6**(4), 259–286 (2016)

19. Bos, J.W., Halderman, J.A., Heninger, N., Moore, J., Naehrig, M., Wustrow, E.: Elliptic curve cryptography in practice. https://eprint.iacr.org/2013/734.pdf
20. Caswell, M.: OpenSSL Wins the Levchin Prize. https://www.openssl.org/blog/blog/2018/01/10/levchin/
21. Hamburg, M.: Ed448-Goldilocks, a new elliptic curve. Cryptology ePrint Archive, Report 2015/625 (2015)
22. Lochter, M., Wiemers, A.: Twist Insecurity, International Association for Cryptologic Research. https://pdfs.semanticscholar.org/3428/3663d6d5bfa60c6dfeafadbf50d69e9b9b40.pdf
23. Scott, M.: Backdoors in NIST elliptic curves. https://www.miracl.com/press/backdoors-in-nist-elliptic-curves
24. Koblitz, N.: Elliptic curve cryptosystems. Math. Comput. **48**(177), 203–209 (1987)
25. Perlroth, N.: The New York Times, Government announces steps to restore confidence on encryption standards. http://bits.blogs.nytimes.com/2013/09/10/government-announces-steps-to-restore-confidence-on-encryption-standards
26. Sen, N., Dantu, R., Jagannath, V., Thompson, M.: Performance Analysis of Elliptic Curves for Real-time Video Encryption, pp. 64–71. National Cyber Summit, USA (2018)
27. Schoof, R.: Counting points on elliptic curves over finite fields. J. Theory Numbers Bordeaux **7**, 219–254 (1995)
28. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986). https://doi.org/10.1007/3-540-39799-X_31
29. Shoup, V.: A Proposal for an ISO Standard for Public Key Encryption. https://www.shoup.net/papers/iso-2_1.pdf. Accessed 15 July 2019