Chapter

Implementation of Elliptic Curve25519 in Cryptography

Intan Muchtadi-Alamsyah and Yanuar Bhakti Wira Tama

Abstract

Bernstein's design implementation of elliptic Curve25519 in key exchange is claimed to be highly secure and efficient. This curve is, for example, used in the key exchange scheme of TextSecure for Instant Messaging. In this paper, we present an implementation of elliptic Curve25519 in the simplified Elliptic Curve Integrated Encryption Scheme, thus showing that elliptic Curve25519 can also serve other purposes than key exchange. The curve is in Montgomery form, which makes it possible to use Montgomery ladder. Point compression, point decompression, encryption, and decryption algorithms are presented for the simplified Elliptic Curve Integrated Encryption Scheme.

Keywords: elliptic curve, cryptography, Montgomery ladder, integrated encryption scheme

1. Introduction

Curve25519 is an elliptic curve in Montgomery form with base field F_p and $p = 2^{255}–19$. In [1], Bernstein explains its design implementation, which is claimed to be highly secure and efficient. It is, for example, used in the key exchange scheme of TextSecure for Instant Messaging [2]. The advantage of using this curve is that for some point operations, we can use only the x-coordinate, which simplifies the computations and also saves storage.

In previous papers we have presented implementations of elliptic curves in Weierstrass form in a binary field: the implementation of a binary field arithmetic operation algorithm [3, 4] and the implementation of the simplified Elliptic Curve Integrated Encryption Scheme (S-ECIES) in a binary field [5]. In the current paper, we present the implementation of Curve25519 in S-ECIES, thus showing that Curve25519 can also serve other purposes than key exchange.

2. Elliptic curve Montgomery form

Before defining Curve25519, we will give some basic theory on elliptic curves. This paper is only concerned with elliptic curves in Montgomery form, not Weierstrass form. An elliptic curve over F_p in Montgomery form is defined by the equation.

$$By^2 = x^3 + Ax^2 + x, \tag{1}$$

where $A(B^2 - 4) \neq 0$.

On the points of the elliptic curve, we may define point addition, negation, and doubling. We define point negation as follows: let E be an elliptic curve over F_p and point $P(x,y)$ be a point on E. We define point negation of P as $-P(x, -y)$. Let $P(x_1,y_1)$ and $Q(x_2,y_2)$ be two distinct points on E. Then the point addition is $P+Q$ (x_3,y_3), where

$x_3 = (\lambda^2 - A - x_1 - x_2), y_3 = \lambda(x_1 - x_3) - y_1$ and $\lambda = (y_2 - y_1)/(x_2 - x_1)$. If $P = Q$, then the doubling point $P + P$ is $2P(x_4,y_4)$, where

$$x_4 = (\lambda^2 - A - 2x_1), y_4 = \lambda(x_1 - x_4) - y_1 \tag{2}$$

and $\lambda = (3{x_1}^2 + 2Ax_1 + 1)/(2By_1)$.

The points on the elliptic curve along with point at infinity O form a commutative group with point addition as its operation.

We define scalar point multiplication as follows: given a positive integer m, scalar point mP is defined by $mP = P+P+...+P$ (m times addition of P).

The advantage of using Montgomery form rather than Weierstrass form is that in Montgomery form, it is possible to operate without y-coordinates.

Elliptic curve operation in Montgomery form without y-coordinates can be done as follows [6]: let $(X:Y:Z)$ be the projective representation of point $P(x,y)$ in E, define $nP = (X_n:Y_n:Z_n)$, and write (x,y) as $(X/Z,Y/Z)$. It is clear that $(m+n)P = mP + nP$. If $P_m(x_1,y_1) = mP$ and $P_n(x_2,y_2) = nP$, $x_1 = X_m/Z_m$ and $x_2 = X_n/Z_n$, then point addition is P_m+P_n $(x_3,y_3) = (m+n)P$, where $x_3 = X_{m+n}/Z_{m+n}$ and

$$X_{m+n} = [(X_m - Z_m)(X_n + Z_n) + (X_m + Z_m)(X_n - Z_n)]^2 \tag{3}$$

$$Z_{m+n} = [(X_m - Z_m)(X_n + Z_n) - (X_m + Z_m)(X_n - Z_n)]^2 \tag{4}$$

Point doubling is $2P_n(x_4,y_4) = 2nP = P_{2n}$, where $x_4 = X_{2n}/Z_{2n}$ and

$$X_{2n} = (X_n + Z_n)^2(X_n - Z_n)^2 \tag{5}$$

$$Z_{2n} = (4X_nZ_n)\left[(X_n + Z_n)^2 + (A-2)/4 * (4X_nZ_n)\right], 4X_nZ_n = (X_n + Z_n)^2 - (X_n - Z_n)^2 \tag{6}$$

Based on the work by Okeya and Sakurai reported in [7], we can recover the y-coordinate in projective coordinates. Let $P(x,y)$, $P_1(x_1,y_1)$, $P_2(x_2,y_2)$ be points on a Montgomery-form elliptic curve. Express $P_1 = (X_1/Z_1,Y_1/Z_1)$, $P_2 = (X_2/Z_2, Y_2/Z_2)$, and define ${X_1}^{rec}$, ${X_2}^{rec}$, ${X_3}^{rec}$ as follows:

$$X_1^{rec} = 2ByZ_1Z_2X_1 \tag{7}$$

$$Y_1^{rec} = Z_2\left[(X_1 + xZ_1 + 2AZ_1)(X_1x + Z_1) - 2A{Z_1}^2\right] - (X_1 - xZ_1)^2X_2 \tag{8}$$

$$Z_1^{rec} = 2ByZ_1Z_2Z_1 \tag{9}$$

Assuming $P_2 = P_1+P$, then in projective coordinates the relation $({X_1}^{rec} : {Y_1}^{rec} : {Z_1}^{rec}) = (X_1 : Y_1 : Z_1)$ holds.

3. Curve25519 and simplified ECIES

Curve25519 is the elliptic curve of Montgomery form

$$y^2 = x^2 + 486662x^2 + x \tag{10}$$

on F_{p2}, where p is the prime number 2^{255}-19. Based on Bernstein's paper [1], there are two subgroups of Curve25519 with large-size order, i.e., $\{O\} \cup \{E$

$(F_{p2}) \cap (F_p \times F_p)\}$ with size order $8 \times (2^{252}$ + 27742317777372353535851937790 883648493) and $\{O\} \cup \{E(F_{p2}) \cap (F_p \times \sqrt{2F_p})\}$ with size order $4 \times (2^{253}$–55484 63555474470 7071703875581767296995).

S-ECIES is based on the elliptic curve discrete logarithm problem described as follows [8]: let p be a prime number larger than 3. Let E be an elliptic curve over F_p such that E contains a cyclic subgroup H, generated by P, of prime order m. The plaintext space is $F_p{}^*$ and the ciphertext space is $(F_p \times F_2) \times F_p{}^*$. The key space is L = {(*E, P, Q, n, m*): $Q = nP$}. Curve E and points P, Q, and m become public keys, and n becomes the private key.

For every $a \in F_p{}^*$ and a secret number $k \in [1, n-1]$, the encryption function is

$$e(a,k) = (\text{Point} - \text{Compress}(\text{kP}), a.a_0 \bmod p) \in \left(F_p \times F_2\right) \times F_p{}^*, \tag{11}$$

where $a_0 \neq 0$ is the absis of kQ.

For every $(V, c) \in (F_p \times F_2) \times F_p{}^*$, the decryption function is

$$d(V,c) = c(x_0)^{-1}, \tag{12}$$

where (x_0, y_0) is the coordinate of Point-Decompress(V).

We know that the groups $\{O\} \cup \{E(F_{p2}) \cap (F_p \times F_p)\}$ and $\{O\} \cup \{E(F_{p2}) \cap (F_p \times \sqrt{2\text{F}_\text{p}})\}$ are finite with group size at $8 \times p_1$ and $4 \times p_2$, respectively, for some primes p_1 and p_2. Hence, E contains a subgroup with prime order; therefore, Curve25519 can be implemented in ECIES.

4. Implementation

In this section, we will give several algorithms in Curve25519 for implementation in S-ECIES, i.e., Montgomery ladder, point compression, point decompression, and others.

An advantage of using an elliptic curve in Montgomery form is that Montgomery ladder can be used for scalar point multiplication.

Algorithm 1 Montgomery Ladder.

INPUT: scalar n, point P

OUTPUT: nP

1. $R_0 \leftarrow O$
2. $R_1 \leftarrow P$
3. for $i \leftarrow m$ down to 0
4. if $d_i = 0$
5. $R_1 \leftarrow R_0 + R_1$ (*Point Addition*)
6. $R_0 \leftarrow 2R_0$ (*Point Doubling*)
7. else
8. $R_0 \leftarrow R_0 + R_1$ (*Point Addition*)
9. $R_1 \leftarrow 2R_1$ (*Point Doubling*)

10. end if

11. end for

12. return(R_0)

Now, we can talk about point compression and point decompression in Curve25519. The algorithm for point compression is straightforward from the existence of two points with the same x-coordinate on an elliptic curve, but with a different y-coordinate, i.e., point (x,y) and point $(x,-y)$, which is equal to point $(x,p-y)$. Because p is odd prime, if y is an odd number, then $p-y$ is an even number and vice versa. Hence, we can compress point (x,y) by $(x, y \bmod 2)$, of which the possible result is $(x,0)$ or $(x,1)$.

Remember that in Curve25519 the y-coordinate is defined when y is not a quadratic residue or $(x,y\sqrt{2})$. By the same argument, if $(x,y\sqrt{2})$ is on E, then $(x-(p-y)\sqrt{2})$ is also on E. However, before we can compress a point with form $(x,y\sqrt{2})$, we have to divide the y-coordinate with $\sqrt{2}$ to avoid problems in real computation. Then, the possible result when we compress the point with form $(x,y\sqrt{2})$ is also $(x,0)$ or $(x, 1)$.

Algorithm 2. Point Compression
INPUT: Point(x,y).
OUTPUT: Point(x,i)

1. if y quadratic residue modulo p then

2. $i \leftarrow y \bmod 2$

3. return (x,i)

4. else

5. $y \leftarrow y/\sqrt{2}$.

6. $i \leftarrow y \bmod 2$

7. return (x,i)

8. end if

The inverse algorithm for point compression is point decompression, i.e., recalling the "real" y-coordinate from point compression.

Algorithm 3. Point Decompression.
INPUT Point (x,i).
OUTPUT Point (x,y)

1. $z \leftarrow x^3+486662x^2+x$

2. if z quadratic residue modulo p then

3. $y \leftarrow \sqrt{z} \bmod p$

4. if $y = i \bmod 2$ then

5. return (x,y)

6. else

7. return$(x,p-y)$

8. end

9. else

10. $z \leftarrow z/2 \bmod p$

11. $y \leftarrow \sqrt{z} \bmod p$

12. if $y = i \bmod 2$ then

13. return $(x, y\sqrt{2})$

14. else

15. return$(x, (p-y)\sqrt{2})$

16. end

17. end

The next algorithms are used to recover the y-coordinate in elliptic curve Montgomery form, because we need it in ECIES.

Now we can give the algorithms for encryption and decryption. For a point generator P in Curve25519 that has a prime order n, if Alice sends message x to Bob with private key m so Q = mP, then Alice encrypts the message with the following algorithm:

Algorithm 4. Encryption in Simplified ECIES
INPUT: Plaintext a
OUPUT: Ciphertext $(V(x_1,y_1),c)$

1. $k \leftarrow$random$([1,n-1])$

2. $R(x_1,z_1) \leftarrow (k-1)P$

3. $Q(x_2,z_2) \leftarrow R(x_1,y_1)+P$

4. $R(y_1) \leftarrow$*Recovery-Y*$(P,R(x_1,z_1),Q(x_2,z_2))$

5. $U(x_3,y_3) \leftarrow R+P$

6. $V(x_3,y_3) \leftarrow$Point-Compression$(U(x_3,y_3))$

7. $V(x_4,y_4) \leftarrow kQ$

8. $y \leftarrow x_0.a$

9. return$(V(x_3,y_3),y)$

Note that in the above algorithm in line 4, there is the command "Recovery-Y." This command is based on Okeya and Sakurai [7].

If Bob wants to read the actual message from Alice, then Bob decrypts Alice's message using the following algorithm:

Algorithm 5. Decryption in Simplified ECIES.
INPUT: Ciphertext($y1$,$y2$)
OUTPUT: Plaintext a

1. $(x_0,y_0) \leftarrow mPoint\text{-}Decompress(y1)$

2. $a \leftarrow x_0^{-1}$

3. $b \leftarrow y_2 a$

4. *return* b

Since this elliptic curve contains a cyclic subgroup of prime order, it is possible to apply S-ECIES. For example, fix base point $P(X:Y:Z)$ with $X = 9$, $Z = 1$ (because in Curve25519, z_1 always has a value of 1), and the y-coordinate can be chosen randomly between odd and even integers that satisfy $y^2 = x^3 + 486662x^2 + x$. The chosen base point P has prime point order, with point order $m = 2^{252} + 27742317777372353535851937790883648493$. Hence, the curve can be implemented in S-ECIES.

Then, we choose a random integer, k, between 1 and m-1. Then, scalar multiplication of k with point $x = 9$ by using the Montgomery ladder algorithm produces $kP(X_k::Z_k)$, and by using a y-coordinate recovery algorithm we can get $kP(X_k:Y_k:Z_k)$. After that, we convert the projective coordinates to affine coordinates to get $kP(X_k/Z_k,Y_k/Z_k)$, and we use $Point\text{-}Compress(kP)$. Then the y-coordinate of ciphertext is the multiplication of plaintext x with x_3, where we get x_3 from $kQ = (x_3,y_3)$. Since we only use the x-coordinate of kQ, we can use Montgomery ladder with scalar k and point $Q = nP$.

For decryption, we first decompress $V(x_1,y_1)$ and then use private key n to get scalar multiplication nV, using only the Montgomery ladder algorithm. The last step is multiplying the y-coordinate of ciphertext with the inverse of the x-coordinate of nV to get the plaintext x. This inverse exists, because we are working in a prime field and the x-coordinate of V is not zero.

Now, we discuss arithmetic in F_p with $p = 2^{255}–19$. There are two operations in F_p, addition and multiplication. However, in F_p with $p = 2^{255}–19$, it is not that easy. Bernstein [1] used radix $2^{25.5}$, which is a polynomial with form $\sum \alpha_i x_i$ with i is a number between 0 and 9 and α_i is a multiple of $2^{[25.5i]}$ (where $[x]$ is the smallest integer that is larger than x) and $\alpha_i/2^{[25.5i]}$ is an integer between -2^{25} and 2^{25}. With the restriction that if i is an odd number then $\alpha_i/2^{[25.5i]}$ is between -2^{24} and 2^{24}, while if i is an even number then $\alpha_i/2^{[25.5i]}$ is between -2^{25} and 2^{25}, therefore, every element in F_p with $p = 2^{255}–19$ can be converted in radix polynomial form. The following algorithm converts integers to radix as follows:

Algorithm 6. Integers to radix $2^{25.5}$
INPUT: n
OUTPUT: $R(x)$

1. $d \leftarrow$ BINARY(n)

2. $p \leftarrow$ LENGTH(d)

3. $a \leftarrow 0$

4. while $p > 26$ do

5. if $a = 0 \bmod 2$ then
6. $p \leftarrow p\text{-}26$
7. $k_a \leftarrow 26$
8. else
9. $p \leftarrow p\text{-}25$
10. $k_a \leftarrow 25$
11. end
12. $a \leftarrow a+1$
13. end
14. $sum \leftarrow ZEROS(1,a)$
15. $k_a \leftarrow p\text{-}1$
16. for $i \leftarrow 1$ to p do
17. if $d(i)=1$ then
18. $sum(a) \leftarrow sum(a)+2^{ka}$
19. end
20. end
21. for $i \leftarrow a\text{-}1$ downto 0 do
22. $l \leftarrow k_i\text{-}1$
23. for $j \leftarrow p+1$ to $p+k_i$ do
24. if $d(j)=1$ then
25. $sum(i) \leftarrow sum(i)+2^{l}$
26. end
27. $l \leftarrow l\text{-}1$
28. end
29. $p \leftarrow p+ki$
30. end
31. $g(x) \leftarrow (sum(0)+\ldots+sum(a)x^{a})$
32. Return $g(x)$

From the above algorithm, first convert the integer to binary representation, and then from the right partition every 26,25,26,25,...,*k*, with $0 \leq k \leq 25$, as an example of an integer with length of binary representation is 231, then partition from the right 26,25,26,25,26,25,26,25,26,1. Every partition states the value sum of $d(i)2^{i-1}$, with d(i) is the value of the order of the binary representation that is either 0 or 1. Also, the *j*-th partition is the coefficient of x^{j-1}.

Example: Suppose we have a 15-digit number, 325606250916557, which has binary representation "1001010000010001100 01110 01110 11000 01010 10110 01101." For integers, 325606250916557 has two partitions, i.e., 001110110000101010110011101 and 100101000000100011000111. Therefore, the coefficient of x0 is $0.2^{25} + 0.2^{24} + 1.2^{23} + \ldots + 0.2^{1} + 1.2^{0}$, which if we calculated would be the value 15477453. In the same way, coefficient x1 would be the value 4851911. Thus, the number 325606250916557 represented by radix $2^{25.5}$ would be 4851911x + 15477453. Also, we can use.

addition and multiplication in radix $2^{25.5}$.

After we have converted any integer, there is an additional problem when the coefficient of radix $2^{25.5}$ exceeds our definition. For this problem, Bernstein [1] has already provided a solution.

5. Applications

Communication systems in the future are expected to interact between diverse types of devices. This allows the user to construct a personal distributed environment using a combination of different communication technologies. The security of transmitted data between these devices is a very important aspect.

Nowadays instant messaging is popular for personal and business communications instead of short messages (SMS) on mobile devices. However, most mobile messaging applications do not protect confidentiality or message integrity. Supervision over private communications conducted by the NSA motivates many people to use alternative messaging solutions for security and privacy of communication on the Internet. A messaging app that claims to be secure instant messaging and has attracted a lot of attention is TextSecure.

Elliptic curve cryptosystem (ECC) is a public-key cryptography suitable for use in environments with limited resources such as mobile devices and smart cards. In cryptography, Curve25519 is an elliptic curve that offers 128 security bits and is designed for use in the Elliptic Curve Diffie-Hellman (ECDH) key agreement key design scheme. This curve is one of the fastest ECC curves and more resistant to the weak number random generator.

In the TextSecure application, Curve25519 is used for key exchanges and authentication. However, in this paper we show that Curve25519 can also be implemented in simplified Elliptic Curve Integrated Encryption Scheme (S-ECIES). Therefore Curve25519 serves for key exchange, authentication, encryption, and decryption. As Curve25519 is built in such a way as to avoid potential attacks on implementation and avoid side channel attacks and random number generator issues, one may expect more secure communication systems.

6. Conclusion

The curve being used in this paper is $y^2 = x^3 + 48666x^2 + x$, a Montgomery curve, over the prime field 2^{255}–19. This protocol uses elliptic point compression (only the *X*-abscissa), allowing for efficient use of Montgomery ladder for ECDH, which uses only *XZ* coordinates.

In this research we develop efficient algorithms for elliptic curve cryptography using Curve25519 which is implemented in security of instant messaging.

Several algorithms have been established for the implementation of Curve25519 in simplified ECIES: Montgomery ladder for scalar point multiplication, point compression and point decompression, encryption and decryption in simplified ECIES, and the algorithm integer to radix for the arithmetic in F_p with $p = 2^{255}-19$.

In a future research, implementation of Curve25519 in Elliptic Curve Digital Signature Algorithm may be attempted.

Acknowledgements

This research is funded by Hibah Riset KK ITB 2017.

Author details

Intan Muchtadi-Alamsyah* and Yanuar Bhakti Wira Tama
Algebra Research Group, Faculty of Mathematics and Natural Sciences, Institut Teknologi Bandung, Indonesia

*Address all correspondence to: ntan@math.itb.ac.id

References

[1] Bernstein DJ. Curve25519: New Diffie-Hellman speed records in Public Key Cryptography—PKC, Lecture Notes in Computer Science. Vol. 3958. New York, USA: Springer; 2006. pp. 207-228

[2] Frosch T, Mainka C, Bader C, Bergsma F, Schwenk J, Holz T. How Secure is TextSecure? Cryptology ePrint Archive Report 2014/904. 2014. Available from: https://eprint.iacr.org/2014/904

[3] Maulana M, Senjaya WF, Rahardjo B, Muchtadi-Alamsyah I, Paryasto MW. Implementation of finite field arithmetic operations for polynomial and normal basis representations. In: Proceeding of 3rd International Conference on Computation for Science and Technology. 2015. pp. 129-134

[4] Paryasto MW, Rahardjo B, Yuliawan F, Muchtadi-Alamsyah I, Kuspriyanto. Composite field multiplier based on look-up table for elliptic curve cryptography implementation. ITB Journal of Information and Communication Technology. 2012;**6**(1): 63-81

[5] Susantio DR, Muchtadi-Alamsyah I. Implementation of elliptic curve cryptography in binary field. Journal of Physics Conference Series. 2016;**710**: 012022

[6] Montgomery PL. Speeding the Pollard and elliptic curve methods of factorization. Mathematics of Computation. 1987;**48**:243-264

[7] Okeya JK, Sakurai K. Efficient elliptic curve cryptosystems from a scalar multiplication algorithm with recovery of the y-coordinate on a Montgomery form elliptic curve. Lecture Notes in Computer Science. 2001;**2162**:126-141

[8] Stinson D. Cryptography: Theory and Practice. 3rd ed. Boca Raton: Chapman & Hall/CRC; 2006