UDC 621.391.15:519.7 **DOI:10.30837/rt.2020.1.200.18**

O.V. TSYGANKOVA

ANALYZING OF POSSIBILITY OF USING ELGAMAL ALGORITHM WITH DETERMINISTIC EMBEDDING FOR KEY ENCAPSULATION

Introduction

Suppose some parties, A and B, use some symmetrical encryption algorithm (for example, AES) to encrypt their messages from A to B and from B to A. They get their secret keys from some Trusted Authority (TA). TA generates keys and then delivers them to correspondent users. The simplest and, may be, the optimal way to deliver the secret key to user A is to encrypt it (using some asymmetrical encryption algorithm) with A's public key and then to send it to A via public channel. Such procedure is called "key encapsulation".

Key encapsulation algorithms are widely used in the modern cryptography and represented in national and ISO/IEC standards of key encapsulations [1, 2].Building the key encapsulation algorithm [3], which may be used as a national standard, is an actual problem nowadays. Ukrainian cryptographers are also working on such standard [4]. Modified Elliptic Curve Integrated Encryption Scheme (ECIES), included in the ANSI X9.63, ISO/IEC 18033-2, IEEE 1363a and SECG SEC1 standards, was used in the project of national standard for key encryption.

In this article we discuss some alternative encryption algorithm on elliptic curve which also may be used for this purpose.

Generally speaking, we can use arbitrary asymmetric encryption algorithm for key encapsulation. One of the simplest and preferable algorithms is ElGamal encryption algorithm [5]. To use this algorithm on elliptic curve, we need algorithms for embedding key into point on elliptic curve and for retrieving it back. Several lines of work in both the number theory and cryptography literature have considered the problem of deterministically mapping field element to point on elliptic curve. However, only probabilistic algorithms of such embedding existed until 2016, when deterministic algorithm for hash embedding was proposed in [6]. But key embedding is much more complicated procedure than hash embedding.

In what follows we describe how this algorithm for key embedding can be built and then discuss the problems that appear if we want to use it in key encapsulation.

To formalize our problem, we need the next designations.

Let $E(F_p)$ be an elliptic curve over F_p given with the equation

$$E: y^2 = g(x), \text{ where } g(x) = x^3 + ax + b, \quad a, b \in F_p, ab \neq 0.$$

We assume that key length is n and for some large prime p we have $p > 2^n$. In this case we can consider vector k as binary representation of some element $k \in F_p$.

Our purpose is to build mapping $F_p \to E(F_p)$, which maps each element $k \in F_p$ into correspondent point $P_k \in E(F_p)$. Moreover, such mapping should be invertible for key retrieving from the point.

1. Classical ElGamal cryptosystem and its elliptic analogue

Classical ElGamal public-key encryption scheme was built in multiplicative group F_p^* for large prime p. Its security is based on the intractability of the discrete logarithm problem [7,8]. To understand the problem of key embedding into the elliptic curve point let us describe the basic ElGamal and Elliptic Curve ElGamal encryption schemes.

Let p be large prime, g be a generator of F_p^*.

Also suppose that party A has his private key a, $2 \le a \le p-2$, and correspondent public key $h = g^a \bmod p$.

Assume TA generates key k (for example, for AES encryption) and should deliver it to parties A and B using only public channels. Suppose TA uses classical ElGamal algorithm (in F_p^*) to encrypt the key k. In this case it does the next steps.

Algorithm 1.

Classical ElGamal algorithm (key encryption)

1. Generates random r, $2 \le r \le p-2$.
2. Evaluates $C_1 = g^r \bmod p$.
3. Evaluates $R = h^r \bmod p$.
4. Evaluates $C_2 = k \cdot R \bmod p$.
5. Forms ciphertext $C = (C_1, C_2)$ and sends it to A.

Then TA does the same procedure for user B, using B's private key.

When A obtains ciphertext $C = (C_1, C_2)$ he decrypts it and finds key k, using secret key a, as $k = C_1^{-a} \cdot C_2 \bmod p$. User B do the same with correspondent cipher text and his secret key.

These scheme can be easily transformed to correspondent elliptic analogue, but there exist some nuances. For example, we may mention Koblitz paper [9] where this analogue was firstly proposed. But the bottleneck of correspondent elliptic algorithm is the step 4 in Algorithm 1: we need k to be some point on elliptic curve. The remaining steps of the algorithm are directly transferred to elliptic case.

Let for large prime p the base point P of elliptic curve (EC) $E(F_p)$ has order n. Party A has his private key a, $2 \le a \le n-2$, and correspondent public key $G = aP$. TA generates key k and should deliver it to parties A and B using only public channels. Suppose TA uses ElGamal algorithm in over $E(F_p)$ to encrypt the key k embedded into point on elliptic curve K. In this case it does the next steps.

Algorithm 2.

EC ElGamal algorithm (key encryption)

1. Generates random r, $2 \le r \le n-2$.
2. Evaluates $C_1 = rP$.
3. Evaluates $C_2 = K + rG$.
4. Forms ciphertext $C = (C_1, C_2)$ and send it to A.

Then TA does the same procedure for user B, using B's private key.

A obtains cipher text $C = (C_1, C_2)$ and decrypts it using secret key a, as $K = C_2 - aC_1$. User B do the same with correspondent cipher text and his secret key.

To solve the problem of embedding key into point on elliptic curve and retrieving it, Koblitz in [9] proposed some probabilistic algorithm of embedding k into elliptic curve point. But usage of such algorithm makes system very complicated and inconvenient. That is why elliptic analogue of ElGamal algorithm is not used in practice till nowadays.

In the next paragraph we will build deterministic algorithm for key embedding into elliptic curve point. We also describe the inverse algorithm, i.e. the algorithm of retrieve key from elliptic curve point and proof the correctness of these algorithms.

2. Embedding algorithm justification

To build key embedding algorithm, we use the algorithm of hash imbedding into elliptic curve point, which was recently proposed by Boneh and others in [6]. We firstly describe the Boneh's algorithm and then explain how we can modify it for our purposes. Note that in our case we should be able not only to embed the key into point on the curve, but also to reveal it univocally after decryption.

Let for some $\xi \in F_p$ we have $\xi \notin Q_p$ (i.e. ξ is quadratic non-residue in F_p, Q_p is the set of all quadratic residues in F_p). Then for key $\xi \in F_p$ set $u_k = k^2 \xi$. Note that u_k is also quadratic non-residue in F_p, or another words $u_k \notin Q_p$. Note that we should exclude case $u_k = -1$, which may happened with negligible probability (only if $-1 \notin Q_p$ and $k^2 \bmod p = -\xi^{-1} \bmod p$).

Now find the value x_k such that the next equation holds:

$$g(u_k x_k) = u_k{}^3 g(x_k). \tag{1}$$

The equation (1) is equivalent to the equation

$$(u_k x_k)^3 + a u_k x_k + b = u_k{}^3 \left(x_k{}^3 + a x_k + b \right),$$

from where we obtain

$$x_k = b\left(u_k^3 - 1\right)\left(a u_k \left(1 - u_k^2\right)\right)^{-1}. \tag{2}$$

The equality (2) can be simplified as

$$x_k = b \cdot \frac{u_k^3 - 1}{a u_k \left(1 - u_k\right)^2} = \frac{b\left(u_k^2 + u_k + 1\right)}{-a u_k \left(u_k + 1\right)}$$

$$x_k = -\frac{b}{a} \cdot \left(\frac{u_k^2 + u_k + 1}{u_k^2 + u_k} \right)$$

$$x_k = -\frac{b}{a} \cdot \left(1 + \frac{1}{u_k{}^2 + u_k} \right) \tag{3}$$

For such x_k we have

$$g(x_k) g(u_k x_k) = g(x_k) u_k^3 g(x_k) =$$
$$= g(x_k)^2 u_k^3 = g(x_k)^2 \left(k^2 \xi\right)^3 = \left(g(x_k) k^3 \xi\right)^2 \xi$$

i.e. $g(x_k) g(u_k x_k)$ is a quadratic non-residue in F_p. It means that exactly one of the next two elements, $g(x_k)$ or $g(u_k x_k)$, is a quadratic residue in F_p, and the other is non-residue.

If $g(x_k) \notin Q_p$ then redefine $x_k \leftarrow u_k x_k$. In this case to retrieve u_k we use the equation

$$\frac{x_k}{u_k} = -\frac{b}{a} \cdot \left(1 + \frac{1}{u_k{}^2 + u_k} \right) \tag{4}$$

instead of (3).

Hence we have $g(x_k) \in Q_p$, so there exist two square roots from $g(x_k)$ in F_p, $y_{1,2} = \pm\sqrt{g(x_k)}$. Note that one of $y_{1,2}$ has least significant bit equal to 0 (i.e. $lsb = 0$), and other has $lsb = 1$. We can chose any of these two roots according to some predefined rule, for example, with $lsb = 0$ and define $y_k = y_1$, if $lsb\left(y_1\right) = 0$, and $y_k = y_2$, else. Therefore the point $P_k(x_k, y_k) \in E\left(F_p\right)$, correspondent to field element $k \in F_p{}^*$, was constructed and actually the mapping $F_p \to E\left(F_p\right)$ wasconstructed which implies that for any key $k \in F_p{}^*$ there exists correspondent point $P_k(x_k, y_k) \in E\left(F_p\right)$.

The common form for this mapping for key $k \in F_p$, some fixed $\xi \notin Q_p$, and $u_k = k^2\xi$ can be described as

$$P_k = \begin{cases} \left(x_k, \sqrt{g(x_k)}\right), if g(x_k) \in Q_p; \\ \left(u_k x_k, \sqrt{u_k^3 g(x_k)}\right), if g(x_k) \notin Q_p. \end{cases}$$

So we obtain the next algorithm for key embedding into elliptic curve point.

Algorithm 3.

Key embedding into elliptic curve point.

Input: $a, b, k, \xi \in F_p$, $\xi \notin Q_p$

1. Evaluate $u_k = k^2\xi$.
2. Evaluate $t_1 = lsb(u_k)$.
3. Evaluate $x_k = -\frac{b}{a}\left(1 + \frac{1}{u_k^2 + u_k}\right)$.
4. Evaluate $g_k = x_k^3 + ax_k + b$; $t_2 \leftarrow 0$.
5. If $g_k \notin Q_p$ then $x_k \leftarrow u_k x_k$, $t_2 \leftarrow 1$, and $g_k \leftarrow u_k^3 g_k$.
6. Evaluate $y_{1,2} = \sqrt{g(x_k)}$.
7. If $lsb(y_k) = 1$ then $y_k \leftarrow p - y_k$.
8. Evaluate $t_3 = lsb(k)$.

Output: $P_k(x_k, y_k)$, t_1, t_2, t_3.

The main problem which appears now is: how to reveal the key k from the point $P_k(x_k, y_k)$? If we solve it, we will have simple algorithm for key encapsulation based on ElGamal encryption algorithm.

Note that values t_1, t_2, t_3 in Algorithm 3 serve just for the solution of the problem. It is described in the next paragraph.

3. Algorithm of retrieving k from point $P_k(x_k, y_k) \in E(F_p)$

In what follows, we are going to build the algorithm for retrieving k. We use the equalities (3) and (4) for this purpose. From these equalities, we can get u_k as the solution of correspondent quadratic equation. In case when $g_k \in Q_p$ we use the equality (3) and obtain

$$(u_k^2 + u_k)\left(ab^{-1}x_k + 1\right) + 1 = 0;$$

$$u_k = \frac{-\left(ab^{-1}x_k + 1\right) \pm \sqrt{\left(ab^{-1}x_k + 1\right)^2 - 4\left(ab^{-1}x_k + 1\right)}}{2\left(ab^{-1}x_k + 1\right)};$$

$$u_k = \frac{-1 \pm \sqrt{\left(ab^{-1}x_k + 1\right) - 4}}{2};$$

$$u_k = \frac{p-1}{2}\left(-1 \pm \sqrt{\left(ab^{-1}x_k - 3\right)}\right).$$

In case when $g_k \notin Q_p$ the transformation $x_k \leftarrow u_k x_k$ was done in step 5 of Algorithm 3, so we get equality (4). Solving correspondent quadratic equation we get the value u_k as:

$$\frac{ab^{-1}x_k}{u_k}+1+\frac{1}{u_k\left(u_k+1\right)}=0;$$

$$u_k{}^2+u_k\left(ab^{-1}x_k+1\right)+\left(ab^{-1}x_k+1\right)=0;$$

$$u_k=\frac{-\left(ab^{-1}x_k+1\right)\pm\sqrt{\left(ab^{-1}x_k+1\right)^2-4\left(ab^{-1}x_k+1\right)}}{2};$$

$$u_k=\frac{p-1}{2}\left(ab^{-1}x_k+1\right)\left(1\pm\sqrt{1-4\left(ab^{-1}x_k+1\right)^{-1}}\right).$$

Now we can recover the value k from u_k.

Note that to get the value u_k uniquely we need the bit values t_1, t_2, t_3. from Algorithm 3. So we get the next Algorithm for key retrieving.

Algorithm 4.

Key retrieving from elliptic curve point.

Input: $x_k \in F_p$, $t_1, t_2, t_3 \in \{0,1\}$.

1. If $t_2=0$ then evaluate $u_k=\frac{p-1}{2}\left(-1\pm\sqrt{\left(ab^{-1}x_k-3\right)}\right)$ and choose u_k such that *lsb* (u_k) = t_1

else evaluate $u_k=\frac{p-1}{2}\left(ab^{-1}x_k+1\right)\left(1\pm\sqrt{1-4\left(ab^{-1}x_k+1\right)^{-1}}\right)$ and choose u_k such that *lsb* (u_k) = t_1.

2. Evaluate $k=\sqrt{-u_k}$

3. If *lsb(k)* $\neq t_3$. than evaluate $k=p-k$.

Output: k.

Conclusion

We described deterministic algorithms for key embedding into elliptic curve point and for key retrieving. These two algorithms give us an opportunity to use elliptic ElGamal algorithm for key encapsulation. Note that such algorithm is much more efficient (in speed) than one used in Belorussian national standard for Key Transport and is at least not less efficient than proposed Ukrainian project of standard for key encapsulation. But to make more definite conclusion about its merits and demerits, this algorithm should be analyzed in more detail. This analysis will be the topic of our further researches.

References:

1. СТБ 34.101.45-2013 Информационные технологии и безопасность. Алгоритм электронной цифровой подписи и транспорта ключа на основе эллиптических кривых. Available via https://apmi.bsu.by/resources/std.html/.
2. ISO/IEC 18033-2:2006 Information technology – Security techniques – Encryption algorithms / Part 2: Asymmetric ciphers.
3. Проект національного стандарту Інформаційні технології. Криптографічний захист інформації. Алгоритм шифрування коротких повідомлень, що ґрунтується на скручених еліптичних кривих Едвардса. Available via http://crypton.ua/images/Проект_стандарту.pdf
4. V.Shoup. A Proposal for an ISO Standard for Public Key Encryption. Preprint, December 2001. Available via https://www.shoup.net/papers/iso-2.pdf
5. Wenbo Mao. Modern Cryptography: Theory and Practice. PrenticeHall, 2003. 707 p.
6. Wahby Riad S. and Dan Boneh. Fast and simple constant-time hashing to the BLS12-381 elliptic curve // IACR Cryptologye Print Archive. 2019. 403 p.
7. P. van Oorschot S. Vanstone, A. Menezes. Handbook of Applied Cryptography. CRC Press, 1996.
8. W. Diffie, M. Hellman. New directions in cryptography // IEEE Trans. Inform. Theory. 1976. Vol. IT-22. P. 472-492,
9. Neal Koblitz. Elliptic Curve Cryptosystems. January 1987. Vol. 48. Number 177. P. 203-209.

National Technical University of Ukraine
"Igor Sikorsky KPI",
Institute of Physics and Technology

Received 09.02.2020