ECC Based Threshold Cryptography for Secure Data Forwarding and Secure Key Exchange in MANET (I)

Levent Ertaul, Weimin Lu

Department of Math and Computer Science
California State University, Hayward
25800 Carlos Bee Blvd.
Hayward, CA 94542-3092 USA
lertaul@csuhayward.edu
wlu@horizon.csuhayward.edu

Abstract. This paper proposes a new approach to provide reliable data transmission in MANET with strong adversaries. We combine Elliptic Curve Cryptography and Threshold Cryptosystem to securely deliver messages in n shares. As long as the destination receives at least k shares, it can recover the original message. We explore seven ECC mechanisms, El-Gamal, Massey-Omura, Diffie-Hellman, Menezes-Vanstone, Koyama-Maurer-Okamoto-Vanstone, Ertaul, and Demytko. For secure data forwarding, we consider both splitting plaintext before encryption, and splitting ciphertext after encryption. Also we suggest to exchange keys between a pair of mobile nodes using Elliptic Curve Cryptography Diffie-Hellman. We did performance comparison of ECC and RSA to show ECC is more efficient than RSA.

1 Introduction

Mobile ad hoc networks are different from mobile wireless IP networks in that there are no base stations, wireless switches, and infrastructure services like naming, routing, certificate authorities, etc. Because mobile nodes join and leave the network dynamically, sometimes even without a notice, and move dynamically, network topology and administrative domain membership can change rapidly. Thus it is important to provide security services such as availability, confidentiality, authentication [1, 2], access control, integrity, and non-repudiation.

As in other networks, cryptography is the foundation for all network security services [3] in MANET, and key management is the major factor to guarantee a secure ad hoc network [4]. However, key management in ad hoc network has to be distributed service [5] as there is no fixed infrastructure to provide centralized service. Another major challenge is to deliver reliable data transmission when some nodes may be compromised [6]. Attackers can disrupt data transmission and incur significant data loss by tampering with, fraudulently redirecting, or even dropping data traffic.

First we suggest seven highly reliable data dispersing and data reconstruction mechanisms using ECC algorithms and TC. Then a key exchange mechanism using ECC Diffie-Hellman and TC is proposed as an alternative to RSADH.

2 Secure Data Transmission

We use Shamir's secret sharing scheme [7] to provide reliable data transferring. There are two basic mechanisms to combine ECC and TC. The first method is to split the messages into n pieces before we use ECC to encrypt them individually and send them to the receiver. At the receiving end, each share of secret is decrypted using ECC respectively. Then k piece shares in plaintext are combined to recover the secret. The second method is to encrypt the plaintext using ECC encryption algorithm before we split the ciphertext into n shares. The receiver with at least k shares of ciphertext is able to recover the ciphertext. Finally the destination can use ECC decryption algorithm to decrypt the ciphertext to get the original plaintext.

We can use any source routing protocols [8], like SDR, to find out the number of routes of disjoint nodes between the sender and the receiver to choose the number of shares n. Then, depending on n and the estimated number of compromised nodes in the network, we can come up with the number of share threshold k.

2.1 Transformation between a Plaintext and a Point on Elliptic Curve

Koblitz [9] gives a method to convert a message to an elliptic curve point, and vice versa. We have a Galois Field GF(P), where P is a prime number and P > 3. An integer number k can used to determine how likely we are able to convert any plaintext into a corresponding point on elliptic curve. First we represent the message in ASCII code. Then we add a constant to each character in ASCII to get M such that kM < P. Next we search for an x such that there is a y and (x, y) is a point on the elliptic curve, where $kM < x \leq k(M + 1)$. To recover M, we compute $\lfloor (x-1)/k \rfloor$. We notice that when k is larger, it is more likely to find a point (x, y) for the message M. But there always is a chance that there is not a corresponding point (x, y) for the message M such that $kM < x \leq k(M + 1)$.

2.2 ECC El-Gamal Cryptosystem

Elgamal works out an ingenious public key cryptosystem [11]. Suppose that the ECC has a point G on an elliptic curve $E_p(a, b)$, and the order of G is q. P is a large prime. Bob's private key and public key are n_B, $0 < n_B < q$, and $K_B = n_B G$.

2.2.1 Share Split before Encryption

- First we choose a prime number p > max(M, n), and define $a_0 = M$, the message.
- Then we select k - 1 random, independent coefficients $a_1, a_2, \ldots, a_{k-1}$, $0 \leq a_j \leq p-1$, defining the random polynomial $f(x) = \sum_{j=0}^{k-1} a_j x^j$ over Z_p, a Galois prime field.
- We compute n shares, $M_t = f(x_t) \bmod p$, $1 \leq t \leq n$, where x_t can be just the public index t for simplicity, and convert them to points P_t on elliptic curve $E_p(a, b)$.

- Alice picks a random number r, and sends rG and $P_t + rK_B$ to Bob with index t.
- Bob recovers each elliptic curve point by calculating $P_t + rK_B - n_B rG = P_t$.
- Bob converts P_t to M_t, and deduces M by using Lagrange interpolation formula

$$M = f(0) = \sum_{i=1}^{k} M_{t_i} \prod_{j=1}^{k} \frac{0 - x_{t_j}}{x_{t_i} - x_{t_j}} = \sum_{i=1}^{k} M_{t_i} \prod_{j=1}^{k} \frac{-t_j}{t_i - t_j} \quad (1)$$

2.2.2 Share Split after Encryption

- Alice converts the secret M to a point P_M on the elliptic curve.
- Alice uses El-Gamal encryption to get $P_1 = rG$ and $P_2 = P_M + rK_B$.
- Let $P_2 = (x_2, y_2)$. We choose two random polynomials f_1, f_2 of degree k-1 in GF(p) such that $f_1(0) = x_2$, $f_2(0) = y_2$, and split x_2, y_2 into n shares of secret respectively. Alice sends P_1 and n shares of $P_2(x_2, y_2)$ with their corresponding indices to Bob.
- Bob recovers x_2, and y_2 , and calculates the point $P_M = P_2 - n_B P_1$.
- Eventually Bob will convert the point P_M to the secret M.

Instead of sending n pieces of x_2, y_2 to Bob, Alice can choose a random k – 1 degree polynomial f with $a_0 = x_2$ and $a_1 = y_2$. Thus Alice and Bob can use Vandermonde matrix [10] instead of Lagrange interpolation to share more than one secretes.

2.3 The Massey-Omura Protocol

Since Massey-Omura encryption [12] requires four transmissions between Alice and Bob, it is not an efficient solution for threshold crypto-system. Let N be the order of $E_p(a, b)$. Alice and Bob choose their secret key n_A and n_B respectively, such that $\gcd(n_A, N) = 1$ and $\gcd(n_B, N) = 1$. $NR = \infty$ for any point R on the curve according to Lagrange's theorem. $n_B^{-1} P_{M_3} = n_B^{-1} n_B n_A^{-1} n_A P_M = n_B^{-1} n_B (P_M + qNP_M) = n_B^{-1} n_B P_M = P_M$

Encryption algorithm:

- Alice calculates plaintext M's corresponding point P_M on the elliptic curve, and sends $P_{M_1} = n_A P_M$ to Bob.
- Bob sends back $P_{M_2} = n_B P_{M_1}$ to Alice.
- Alice sends $P_{M_3} = (n_A^{-1} \bmod N) P_{M_2}$ to Bob.

Decryption algorithm:

Bob calculates $(n_B^{-1} \bmod N)\, P_{M_3} = P_M$, and recovers plaintext M by P_M.

2.3.1 Share Split before Encryption

- Alice splits the secret M into n shares of secret M_t, $1 \le t \le n$.
- Alice converts a share M_t into a point P_t on the curve, and sends $P_{t_1} = n_A P_t$ to Bob.
- Bob sends $P_{t_2} = n_B P_{t_1}$ to Alice.

- Alice computes $P_{t\,3} = n_A^{-1} P_{t\,2}$ and sends it to Bob, $n_A^{-1} \in Z_N$.
- Bob computes $n_B^{-1} P_{t\,3}$ and it is P_t, $n_B^{-1} \in Z_N$.
- With at least k share of P_M, Bob recovers P_M, and converts the P_M to the secret M.

2.3.2 Share Split after Encryption

- Alice converts the secret M to a point $P_M = (x, y)$ on the curve.
- Alice computes $P_{M\,1} = n_A P_M = (x_1, y_1)$, splits x_1 and y_1 into n shares respectively, and sends n pieces of $x_{1\,t}$ and $y_{1\,t}$ to Bob.
- Bob combines k pieces of $x_{1\,t}$ and $y_{1\,t}$ separately to get (x_1, y_1), i.e., $P_{M\,1}$, computes $P_{M\,2} = n_A P_{M\,1} = (x_2, y_2)$, splits x_2 and y_2 into n shares respectively, and sends n pieces of $x_{2\,t}$ and $y_{2\,t}$ to Alice.
- Alice combines k pieces of $x_{2\,t}$ and $y_{2\,t}$ separately to get (x_2, y_2), i.e., $P_{M\,2}$, computes $P_{M\,3} = n_A^{-1} P_{M\,2} = (x_3, y_3)$, splits x_3 and y_3 into n shares respectively, and sends n pieces of $x_{3\,t}$ and $y_{3\,t}$ to Bob.
- Bob combines k pieces of $x_{3\,t}$ and $y_{3\,t}$ separately to get (x_3, y_3), i.e., $P_{M\,3}$, computes $P_M = n_B^{-1} P_{M\,3}$, and converts the point P_M to the secret M.

2.4 ECC Diffie-Hellman Protocol

A generalization of the original Diffie-Hellman key exchange in Z_p^* found a new depth when Koblitz [13] suggested that such a protocol could be used with the group over an elliptic curve. The order of a point G on an elliptic curve $E_p(a, b)$ is q. P is a large prime. The secret key $K = n_A n_B G$ is generated using DH algorithm.
Encryption algorithm:

- Alice finds the point P_M corresponding to M, and sends $P_M + n_A n_B G$ to Bob.

Decryption algorithm:

- Bob subtracts $n_A n_B G$ from $P_M + n_A n_B G$, and converts P_M to the plaintext M.

2.4.1 Share Split before Encryption

- Alice splits the secret M into n shares of secret M_t, $1 \le t \le n$.
- Alice converts one share M_t to a point P_t on the curve.
- Alice computes $P_t + n_A n_B G$ and sends it to Bob.
- Bob recovers P_t by subtracting $n_A n_B G$ from $P_t + n_A n_B G$.
- With at least k share of P_M, Bob recovers P_M, and converts the P_M to the secret M.

2.4.2 Share Split after Encryption

- Alice converts the secret M to a point $P_M = (x, y)$ on the curve.
- Alice computes $P_C = n_A n_B G + P_M = (x_C, y_C)$.
- Alice splits x_C and y_C into n shares of x_{C_t} and y_{C_t} respectively, $1 \le t \le n$.
- Alice sends n pieces of x_{C_t} and y_{C_t} to Bob.
- Bob combines k pieces of x_{C_t} and y_{C_t} separately to get (x_C, y_C), i.e., P_C.
- Bob computes $P_M = P_C - n_A n_B G$, and converts the point P_M to the secret M.

2.5 The Menezes-Vanstone Cryptosystem

Menezes-Vanstone Elliptic Curve Cryptosystem [14] is a solution to the problem of encoding a message in a point. It uses a point on an elliptic curve to mask a point in the plane. It is fast and simple. Let H be a cyclic subgroup of $E_p(a, b)$ with the generator G. Bob has a private key n_B, and a public key $n_B G$. The message *M* is converted into a point $P_M = (x, y)$ in *GF(p)*.

Encryption algorithm:

- Alice select a random number $r < |H|$, and calculates $rn_B G = (x_k, y_k)$.
- Alice sends $(rG, x_k x \bmod p, y_k y \bmod p)$ to Bob.

Decryption algorithm:

- Bob calculates $n_B rG = rn_B G = (x_k, y_k)$.
- Bob recovers *x* and *y* by $x_k^{-1} x_k x \bmod p$ and $y_k^{-1} y_k y \bmod p$.
- Bob converts the point(x, y) to get the original plaintext M.

2.5.1 Share Split before Encryption

- Alice splits the message *M* into *n* shares of secret M_t, $1 \le t \le n$.
- Alice converts each share M_t into a point P_t.
- Alice select a random number $r < |H|$, and calculates $rn_B G = (x_k, y_k)$.
- Alice sends $(rG, x_k x_t \bmod p, y_k y_t \bmod p)$ to Bob.
- Bob calculates $n_B rG = rn_B G = (x_k, y_k)$.
- Bob recovers x_t and y_t by $x_k^{-1} x_k x_t \bmod p$ and $y_k^{-1} y_k y_t \bmod p$.
- With at least k share of P_M, Bob recovers P_M, and converts the P_M to the secret M.

2.5.2 Share Split after Encryption

- Alice converts the message *M* into a point P_M.
- Alice select a random number $r < |H|$.
- Alice calculates $rn_B G = (x_k, y_k)$, and calculates $z = x_k x \bmod p$, and $w = y_k x \bmod p$.
- Alice splits *z, w* into *n* shares of z_t, and w_t respectively, $1 \le t \le n$.
- Alice sends *rG* and *n* pieces of z_t, and w_t to Bob.
- Bob combines *k* pieces of z_t and w_t separately to get *(z, w)*.
- Bob calculates $n_B rG = rn_B G = (x_k, y_k)$.
- Bob recovers P_M by $x_k^{-1} z = x_k^{-1} x_k x \bmod p$ and $y_k^{-1} w = y_k^{-1} y_k y \bmod p$.
- Eventually Bob converts P_M to the secret *M*.

2.6 The Koyama-Maurer-Okamoto-Vanstone Cryptosystem

KMOV [15] conjugates the polynomial-time extraction of roots of polynomials over a finite field with the intractability of factoring large numbers. Bob chooses two large prime numbers, p and q, such that $p \equiv q \equiv 2 \bmod 3$. Let n = pq, 0 < b < p and b < q, and N = lcm(p+1, q+1). Bob picks up his public key e with gcd(e, N) = 1. His private key d is $e^{-1} \bmod N$.
Encryption algorithm:

- Alice represents M as a point P_M on elliptic curve $E_n(0, b)$, and sends eP_M to Bob.

Decryption algorithm:

- Bob calculates $deP_M = (rN + 1) P_M = P_M$, where r is an integer.
- Bob recovers the original plaintext M by P_M.

2.6.1 Share Split before Encryption

- Alice splits the secret M into n shares of secret M_t, $1 \le t \le n$.
- Alice converts a piece of share M_t into a point P_t on the curve.
- Alice computes eP_t and sends it to Bob.
- Bob recovers P_t by $deP_t = P_t$.
- With at least k share of P_M, Bob recovers P_M, and converts the P_M to the secret M.

2.6.2 Share Split after Encryption

- Alice converts the secret M to a point $P_M = (x, y)$ on the curve.
- Alice computes $P_C = eP_M = (x_C, y_C)$.
- Alice splits x_C and y_C into n shares of x_{C_t} and y_{C_t} respectively, $1 \le t \le n$.
- Alice sends n pieces of x_{C_t} and y_{C_t} to Bob.
- Bob combines k pieces of x_{C_t} and y_{C_t} separately to get (x_C, y_C), i.e., P_C.
- Bob computes $P_M = dP_C$, and converts the point P_M to the secret M.

2.7 The Ertaul Crypto-system

P is the generator point while x is the private key, and Y = x*P is the public key. $H((xi, yi)) = Hash(xi \oplus yi)$ is a HASH function such as MD5, SHA-1.
Encryption algorithm:

1. Alice selects a random value r from Z_q.
2. Alice computes U = r*P and $V = H(r*Y) \oplus M$, and sends C = (U, V) to Bob.

Decryption algorithm:

1. Given a ciphertext C = (U, V), Bob computes x*U= x*r*P = r*x*P.
2. Bob computes $V \oplus H(r*x*P) = H(r*Y) \oplus M \oplus H(r*x*P) = M$

2.7.1 Share Split before Encryption

- Alice splits the secret M into n shares of secret M_t, $1 \le t \le n$.
- Alice selects a random value r from Z_q, and computes U = r*P.

- For each share M_t, Alice computes $V_t = H(r*Y) \oplus M_t$.
- Alice sends ciphertext $C_t = (U, Vt)$ to Bob.
- Given a ciphertext C_t, Bob computes $x*U= x*r*P$.
- Bob computes $H(r*x*P)$ and $Vt \oplus H(r*x*P) = H(r*Y) \oplus M_t \oplus H(r*x*P) = M_t$
- With at least k share of M_t, Bob is able to recover M.

2.7.2 Share Split after Encryption

1. Alice selects a random value r from Z_q, computes $U = r*P$.
2. Alice computes $V = H(r*Y) \oplus M$, splits V into n shares of secret V_t, $1 \le t \le n$.
3. Alice sends ciphertext $C_t = (U, V_t)$ to Bob.
4. Bob recovers V, and computes $x*U= x*r*P$.
5. Bob computes $H(x*r*P)$ and $V \oplus H(x*r*P) = H(r*Y) \oplus M \oplus H(x*r*P) = M$.

2.8 The Demytko cryptosystem

Demytko [16] uses a fixed randomly chosen elliptic curve $E_n(a, b)$ over the ring $\mathbb{Z}_n$, where n = pq is an RSA modulus. It relies on the fact that if a number x is not the x-coordinate of a point on an elliptic curve $E_p(a, b)$, then it will be the x-coordinate of a point of the twisted curve $\overline{E_p(a,b)}$ defined as, in addition to the point at infinity, the set of points (x, y) satisfying $\overline{E_p(a,b)}$: $y^2 = x^3 + ax + b$, where $y = u\sqrt{v}$, $u \in F_p$, and v is a fixed quadratic non-residue modulo p. Let $|E_p(a, b)| = 1 + p + \alpha$, $|\overline{Ep(a, b)}| = 1 + p - \alpha$, $|E_q(a, b)| = 1 + q + \beta$, $|\overline{Eq(a, b)}| = 1 + q - \beta$.

$N_1 = \text{lcm}(p + 1 + \alpha, q + 1 + \beta)$ $\quad$ $N_2 = \text{lcm}(p + 1 + \alpha, q + 1 - \beta)$

$N_3 = \text{lcm}(p + 1 - \alpha, q + 1 + \beta)$ $\quad$ $N_4 = \text{lcm}(p + 1 - \alpha, q + 1 - \beta)$

e is chosen such that $\gcd(e, N_i) = 1$, and private key d_i is calculated by $d_i = e^{-1} \bmod N_i$, i = 1 to 4. Let x represent the plaintext and s the ciphertext (where $0 \le x, s \le n - 1$).

Encryption algorithm:

Alice sends (s, t) = e(x, y) where $y = \sqrt{x^3 + ax + b}$ and $t = \sqrt{s^3 + as + b}$.

Decryption algorithm:

Bob determines which d_i of the four inverses of e should be used based on the Jacobi symbols $(c^3 + ac + b/p)$ and $(c^3 + ac + b/q)$. Bob computes$(x, y) = d_i(s, t)$.

2.8.1 Share Split before Encryption

- Alice splits the secret M into n shares of secret M_t, $1 \le t \le n$.
- For each share M_t, Alice computes $C_t = e(x_t, y_t)$, and sends ciphertext C_t to Bob.
- Given a ciphertext C_t, Bob chooses which d of the four inverses of e to be used.
- Bob computes $(x_t, y_t) = dC_t$, and recovers M.

2.8.2 Share Split after Encryption

- For a message M, Alice computes C = e(M, y).
- Alice splits C into n shares of secret C_t, $1 \leq t \leq n$, and sends ciphertext C_t to Bob.
- With at least k share of C_t, Bob is able to recover C.
- Bob chooses which d of the four inverses of e to use, and computes (M, y) = dC.

3 Key Exchange Method

Suppose that the order of a point G on an elliptic curve $E_p(a, b)$ is q. P is a large prime. Reliable key exchange using Threshold Crypto-systems works like below.

- First, Alice chooses a secret number n_A with $0 < n_A < q$.
- Bob chooses a secret number n_B with $0 < n_B < q$.
- Alice computes her public key $K_A = (x_A, y_A) = n_A G$.
- Alice splits x_A and y_A into n separate shares, x_{A_t} and y_{A_t}, and sends them to Bob.
- Bob computes his public key $K_B = (x_B, y_B) = n_B G$.
- B splits x_B and y_B into n separate shares, x_{B_t} and y_{B_t}, and sends them to Alice.
- Alice uses k shares of x_{B_t} and y_{B_t} separately to recover x_B, and y_B, i.e., $n_B G$.
- Alice calculates secret key $K = n_A n_B G$
- Bob uses k shares of x_{A_t} and y_{A_t} separately to recover x_A, and y_A, i.e., $n_A G$.
- Bob computes the same key by $n_B n_A G = n_A n_B G$.

4 Performance and Complexity Comparison

4.1 Computation Complexity of Share Split before Encryption and Share Split after Encryption

There are mainly three types of operations in our methods, point addition, point exponentiation and Lagrange interpolation. Doubling of points takes one inverse operation, 10 additions, and six multiplications while addition of two different points takes one inverse operation, 10 additions, and five multiplications. Addition points and doubling of points of basic ECC arithmetic are of comparable computation complexity.

Next we will compare the complexity of point addition and point exponentiation, i.e., P + Q and rG. Let w = p's length in bits and r is a number in GF(p). We can represent r in binary as $r_w r_{w-1} \ldots r0$ or $r = \sum_{i=1}^{w} 2^{i-1} r_i$. $rG = (\sum_{i=1}^{w} 2^{i-1} r_i)G$

Since integer multiplications are much more expensive than integer additions, from now on, we will take only multiplications into consideration. w - 1 additions of two different points need 5(w – 1) multiplications and w – 1 inverse operations. We can see that addition of two points is far cheaper than exponentiation of a point. A (k, n) Shamir's secret sharing algorithm has a complexity of $O(k^2)$, k^2 multiplications and k^2

+ k additions. As k << w ($160 \leq w \leq 256$), it is straightforward to see that ECC encryption/decryption is much more expensive than Lagrange interpolation in the same prime field. Table 1 lists the number of three types of calculations required for each ECC-TC algorithm plus number of packets and packet sizes between senders and receivers.

w is the length in bits needed to represent the largest number that can be used, i.e., p, and w = $\lceil \log p \rceil$. Each packet contains all necessary information for each round of communication between the sender and the receiver, including each individual share x_i and its index i. From this table, we can see, in general, share split before encryption is slower than share split after encryption. Among seven ECC-TC algorithms, MV and Ertaul are the most best from the perspective of computing power requirements because they have the least number of elliptic curve exponentiation calculations over prime fields. If network bandwidth is the critical factor, KMOV and Demytko are better choices.

Table 1. Complexity comparison of seven ECC secret sharing algorithms

ECC	Share split before encryption			Share split after encryption			Pkt size	Pkt #
	rG	P+Q	Lagrange	rG	P+Q	Lagrange		
EG	3n	2n	1	3	2	2	5w	n
MO	4n	0	1	4	0	6	3w	3n
DH	0	2n	1	0	2	2	3w	n
MV	3	0	1	3	0	2	5w	n
KMOV	2n	0	1	2	0	2	3w	n
Ertaul	3	0	1	3	0	1	4w	n
Demytko	2	0	1	2	0	1	3w	n

4.2 Performance advantages comparison of ECC with RSA

Currently, for the same level of resistance against the best known attacks, the system parameters for an elliptic-curve-based system can be chosen to be much smaller than

the parameters for RSA or mod p systems [17]. Table 2 and 3 are taken from [17, 18], and are directly comparable to RSA numbers for the same platform. Table 3 uses ECCDH for ECC encryption and decryption. It calculates the time taken to compute the secret key $n_A n_B G$. Encryption and decryption time are symmetric in ECCDH. Table 4 is from [19]. In table 5, we calculate the timings of seven ECC secret sharing algorithms by only considering point exponentiation since that is bulk of the calculation. Clearly ECCDH is the fastest algorithm. Same as before splitting secret after encryption has better performance than splitting secret before encryption.

Table 2. Key sizes in bits for equivalent levels

Symmetric	ECC	DH/DSA/RSA
80	163	1024
128	283	3072
192	409	7680
256	571	15360

Table 3. Sample ECC and RSA timings in milliseconds over prime fields

Processor	MHz	163-ECC	192-ECC	1024-RSAe	1024-RSAe	2048-RSAe	2048-RSAd
UltraSPARC II	450	6.1	8.7	1.7	32.1	6.1	205.5
StrongARM	200	22.9	37.7	10.8	188.7	39.1	1273.8

Table 4. ECC of Koblitz curve over F_{2^m} and RSA ($e = 2^{16} + 1$) timings in milliseconds on Pentium II 400MHZ

163-ECAESe	163-ECAESd	233-ECAESe	233-ECAESd	283-ECAESe	283-ECAESd
4.37	2.85	7.83	4.85	11.02	6.78
1024-RSAe	1024-RSAd	2048-RSAe	2048-RSAd		
3.86	66.56	13.45	440.69		

Table 1 and Table 5 clearly indicate share split before encryption is not as efficient as share split after encryption. The reason is that share split before encryption splits a secret into n shares and does n encryptions instead of just one in share split after encryption. If the environment has limited computing power, like mobile network or embedded system, MV, KMOV and Demytko take 2 elliptic curve exponentiations while other algorithms requires at least 3. On the other hand, if the application is running over a network with very limited capacity, packet size is more important than complexity. In that case, Massey-Omura, KMOV, and Demytko are the best choices as the packet size is 3w, 3w, and 2w respectively. We noticed that Ertaul ECC-TC has the following advantages: Converting messages into points eliminated, ECC become a block cipher, proven secure cryptosystems (ECC and MD5/SHA-1). Furthermore it is a stable algorithm, i.e., it has the same complexity in both share split before encryption and share split after encryption. It is close to the best candidates in all cases.

Table 5. ECC secret sharing timings in milliseconds over prime fields.

ECC	share split before encryption				share split after encryption			
	163-bit Sun	192-bit Sun	163-bit ARM	192-bit ARM	163-bit Sun	192-bit Sun	163-bit ARM	192-bit ARM
EG	18.3n	26.1n	68.7n	113.1n	18.3	26.1	68.7	113.1
MO	24.4n	34.8n	91.6n	150.8n	24.4	34.8	91.6	150.8
DH	6.1	8.7	22.9	37.7	6.1	8.7	22.9	37.7
MV	12.2	17.4	45.8	75.4	12.2	17.4	45.8	75.4
KMOV	12.2n	17.4n	45.8n	75.4n	12.2	17.4	45.8	75.4
Ertaul	18.3	26.1	68.7	113.1	18.3	26.1	68.7	113.1
De-mytko	18.3n	26.1n	68.7n	113.1n	12.2	17.4	45.8	75.4

5. Related Works

A different approach has been proposed to alleviate the detrimental effects of packet dropping by detecting misbehaving nodes and reporting such events, and maintaining a set of metrics reflecting the past behavior of other nodes [20]. It consists of two entities, the watchdog and the pathrater. When a node forwards a packet, the node's watchdog verifies that the next node in the path also forwards the packet. If the next node does not forward the packet, then it is malicious. The pathrater uses this feedback to choose the best route that is most likely to deliver packets.

Secure Message Transmission Protocol [21] is a comprehensive protocol that tolerates rather than detects and isolates malicious nodes. SMT requires a security association between the two end communicating nodes, i.e., the source and the destination. It uses Active Path Set, a set of diverse, node disjoint paths to transfer dispersed pieces of each outgoing message using Information Dispersal Algorithm. SMT can operate with any underlying secure routing protocol. The message and redundancy data are divided into a number of pieces so that if M out of N transmitted pieces are received successfully, the original message can be correctly reconstructed. The sender updates the rating of each path in its APS based on the feedback provided by the destination. The destination validates the incoming pieces and acknowledges the successfully received ones through a feedback across multiple routes back to the source.

We need to implement our solution and run it by a simulator to compare it with other related works to see the performance improvement.

6 Conclusions

The security-sensitive applications of ad hoc networks require high degree of security, but ad hoc networks are inherently vulnerable to security attacks. Threshold cryptography is a valid approach to build a highly available and highly secure key management service by distributing trust among a group of servers. Elliptic curve cryptogra-

phy provides an efficient alternative to RSA public key encryption. We successfully use ECC and TC to provide the best of both worlds in MANET environment.

References

[1] T.P. Pedersen, "A Threshold Cryptosystem without a Trusted Party", In *Proc. Of Eurocrypt'91*, Lecture Notes in Computer Science, LNCS 547, Springer Verlag, pp.522-526, 1991

[2] Kazuo Takaragi, Kunihiko Miyazaki, Masashi Takahashi, *A Threshold Digital Signature Issuing Scheme without Secret Communication*

[3] Amitabh Mishra and Ketan Nadkarni, Security in Wireless Ad Hoc Networks, *The Handbook of Ad Hoc Wireless Networks,* December 2002, pp. 30.1-30.51

[4] L. Zhou and Z.J. Haas, Securing Ad Hoc Networks, *IEEE Network Magazine*, Nov./Dev. 1999

[5] Dan Zhou, Security Issues in Ad Hoc Networks, The *Handbook of Ad Hoc Wireless Networks,* December 2002, pp. 32.1-30.14

[6] Panagiotis Papadimitratos and Zygmunt Haas, Securing Mobile Ad Hoc Networks, *The Handbook of Ad Hoc Wireless Networks,* December 2002, pp. 31.1-31.17

[7] A. Shamir, How to Share a Secret, in *Communications of the ACM, vol.22, no.11*, pp.612-613, 1979

[8] P. Papadimitratos, Z.J. Haas, Secure Routing for Mobile Ad Hoc Networks, in: *Proceddings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, San Antonio, TX, January 27-31, 2002

[9] N. Koblitz, *A Course in Number Theory and Cryptography (Graduate Texts in Mathematics, No 114)*, Springer-Verlag, 1994

[10] W. Trappe, L. C. Washington, *Introduction to Cryptography: with Coding Theory*, Prentice Hall, 2002

[11] T. Elgamal, A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *IEEE Transactions on Information Theory,* IT-31(4):469-472, July 1985

[12] L. C. Washington, *Elliptic Curves: Number Theory and Cryptography*, Chapman & Hall/CRC, 2003

[13] N. Koblitz, Elliptic Curve Cryptosystems, *Math. Comp.*, 48,203-209, 1987

[14] A. Menezes, S. A. Vanstone, Elliptic Curve Cryptosystems and Their Implementation, *Journal of Cryptology*, 6 (1993), 209-224

[15] K. Koyama, U. Maurer, T. Okamoto, S. A. Vanstone, New Public-Key Schemes Based on Elliptic Curves over the Ring Z_n, *Proceedings of Crypto'91*, LNCS 576, Springer-Verlag, pp. 252-266, 1992

[16] N. Demytko, A New Elliptic Curve Based Analogue of RSA, *EUROCRYPT'93*, LNCS 765 40-49 (1993)

[17] Kristin Lauter, The Advantages of Elliptic Curve Cryptography for Wireless Security, IEEE Wireless Communications, February 2004

[18] V. Gupta, S. Gupta, S. Chang, Performance Analysis of Elliptic Curve Cryptography for SSL, ACM Workshop Wireless Security, Mobicom 2002, Atlanta, GA, September 2002

[19] Michael Brown, Donny Cheung, Darrel Hankerson, Julio Lopez Hernandez, Michael Kirkup, and Alfred Menezes, PGP in Constrained Wireless Devices, in: *Proceedings of the 9th USENIX Security Symposium*, Denver, CO, August 2000

[20] S. Marti, T.J. Giuli, K. Lai, M. Baker, Mitigaing Routing Misbehavior in Mobile Ad Hoc Networks, in: *Proceedings of the 6th MobiCom*, Boston, MA, August 2000

[21] P. Papadimitratos, Z.J. Haas, Secure Message Transmission in *Mobile Ad Hoc Networks*