Security of Ad Hoc Networks and Threshold Cryptography

Levent Ertaul, Nitu Chavan
California State University, Hayward
lertaul@csuhayward.edu, nchavan@horizon.csuhayward.edu

Abstract

A Mobile Ad hoc Network (MANET) is a system of wireless mobile nodes that dynamically self-organize in arbitrary and temporary network topologies allowing people and devices to inter-network without any preexisting communication infrastructure. Taking into account its nature and challenges and security issues, we briefly discuss current security solutions, particularly Threshold Cryptography (TC). Already TC is being sought in computer networks to provide security in terms of availability, confidentiality, and secure key or data distribution. So we investigate to find out what makes it difficult to implement TC in ad hoc networks. To find answers to these questions, we discuss our RSA-based Threshold Cryptography (RSA-TC) implementation. Through our implementation, we have put forth various drawbacks of key sharing in RSA-TC and have suggested an alternative method of splitting message. Our work further proves why RSA-TC is unsuitable in MANET. Finally, we explore Elliptic Curve Cryptology (ECC) and suggest reasons why ECC based Threshold Cryptography (ECC-TC) could be a very effective alternative to RSA-TC schemes in MANET.

1. Introduction

Mobile Ad hoc Network (MANET) is emerging as an important area for new developments in the field of wireless communication. The premise of forming a MANET is to provide wireless communication between heterogeneous devices, anytime and anywhere, with least or no infrastructure [1], [2], [3], [4]. These devices, for instance cell phones, laptops, palmtops remote systems, etc. carry out communication with other nodes that come in their radio range of connectivity. Each participating node provide services such as message forwarding, providing routing information, authentication, etc. to form a network with other nodes spread over an area.

Security of MANETs is questioned due to its unique characteristics such as wireless communication, infrastructure-less network, dynamic membership, and heterogeneous devices. External vulnerabilities like eavesdropping and dynamic network and internal constraints like limited computational and storage capabilities pose challenges in implementing a secure ad hoc network. Hence, basic security requirements of MANET are availability, authentication, integrity, confidentiality, authorization, and trust management [1], [2], [3], [4], [5].

This paper covers in brief current security solutions in MANET. In addition, this paper sheds light on TC and its application in computer and ad hoc networks. Our RSA-TC implementation is discussed. Through this implementation, we put forth the practical issues faced while implementing this scheme in a MANET. Also, considering its limitations, we suggest under what circumstances, RSA-TC could be implemented in MANET. Further, we demonstrate why RSA-TC is unsuitable for MANET. Lastly, we discuss why ECC based TC could be considered as an alternative to realize benefits of TC in MANET.

2. Current security solutions in MANET

2.1. Secure routing

Routing of packets form a basis of the MANET where intermediate nodes route the data from the source to the destination. Assumption is that encryption keys have already been established between the communicating nodes [2]. The efficient packet routing is one of the crucial functionalities required in an ad hoc network [2]. It includes monitoring network traffic, prioritizing the sending of the data packets, authenticating the packets from legitimate nodes, and keeping track of updated routes [3]. Thus, as the message is broadcasted, each node carries out above mentioned functions to thwart various attacks based on the routing protocol.

0-7803-9305-8/05/$20.00 © 2005 IEEE.

2.2. Secure data forwarding

Secure routing is the pre-requisite for implementing secure data forwarding [2]. The motivation is to securely forward data in MANETs in the presence of malicious nodes after the route between the source and target is discovered. There are various schemes proposed for secure data forwarding such as data forwarding based on neighbor's rating, implementing currency system in network for packet exchange, and redundantly dividing and routing message over multiple network routes.

For example, *Secure Message Transmission (SMT)* is a secure data forwarding scheme in which first the active paths are discovered between two nodes using secure routing protocol. Based on *N* active paths, the message is divided into *N* different parts such that any *M* parts can be used to recover this message. These *N* partial messages are then routed on the recognized paths. The destination can recover a message when *M* or more partial messages are received. Thus, this scheme ensures that the message reaches the destination even if a few packets are dropped in transit.

Both the above security solutions are essential to ensure that the ad hoc networks survive even in the presence of malicious nodes. Thus, by implementing the above solutions the nodes can communicate securely without relying on all nodes on only one route.

Extending further the concept of dividing the message using *SMT* protocol, the threshold cryptography can be implemented to redundantly fragment the message into *N* parts such that using any *t* parts the message can be recovered [6].

3. Threshold cryptography

Threshold cryptography (TC) involves sharing of a key by multiple individuals called *shareholders* engaged in encryption or decryption. The objective is to have distributed architecture in a hostile environment. Other than sharing keys or working in distributed manner, TC can be implemented to redundantly split the message into *n* pieces such that with *t* or more pieces the original message can be recovered. This ensures secure message transmission between two nodes over *n* multiple paths.

Threshold schemes generally involve key generation, encryption, share generation, share verification, and share combining algorithms. Share generation, for data confidentiality and integrity, is the basic requirement of any TC scheme. Threshold models can be broadly divided into single secret sharing threshold e.g. Shamir's *t-out-of-n* scheme based on Lagrange's interpolation and threshold sharing functions e.g. geometric based threshold [6]. These schemes are being used to implement threshold variants of *RSA, El Gamal, and Diffie-Hellman* cryptographic algorithms that have characteristic, $E(x + y) = E(x) * E(y)$, called homomorphism [7].

TC finds its application in document authorization/signing or verification in organizations [7], a voting system for allowing access to system resources [8], e-commerce transactions, distributed online certification authority [9], and key distribution [1], [2] in computer networks. TC can be implemented in various applications in a MANET. Applications such as coordinating efforts of military attacks using wireless devices in the battlefield or in disaster-struck area, wireless connectivity of various home appliances, and establishing communication among laptops, PDAs and other wireless devices at conferences, are ideal grounds for adopting TC. When compared with computer networks, it is easy to deduce that to implement TC in MANETs is a challenging task due to its dynamic and distributed nature and constrained resources at each network node.

In next section, we present our efforts to apply threshold scheme based on RSA in ad hoc networks and explore possible difficulties that would have to be addressed.

4. RSA-TC Implementations in MANET

In RSA,

i) $C = M^{d} \bmod N$ and $M' = M = C^{e} \bmod N$

ii) $C = M^{e} \bmod N$ and $M' = M = C^{d} \bmod N$

In RSA-TC authentication/signature scheme,

$C' = \prod_{i=0 \text{ till } i=t} C^{xi * f'(xi)} \bmod N,$

where $C_i = C^{xi} \bmod N$,

$f(x) = (a_0 x^0 + a_1 x^1 + \ldots + a_{(t-1)} x^{(t-1)}) \bmod \phi(N)$

and $a_0 = d$

$f'(x_i) = \prod_{j=0, j \neq i \text{ till } j=t} (x_j / (x_i - x_j)) * f(x_i) \bmod \phi(N)$

Thus,

$C' = M^{\{\Sigma_{i=0, j=0, j \neq i \text{ till } i=t, j=t} (x_j / (x_i - x_j)) * f(xi)\}} \bmod N$

$M' = M = C'^{e} \bmod N = C^{e} \bmod N$

Figure 1. RSA and RSA-TC scheme using Lagrange interpolation

As shown in Fig. 1, RSA-TC has been implemented using *JAVA 1.4* in Unix environment on *SUN Sparc Ultra 5_10* machines. The application is designed to simulate an ad hoc network node with a capability to send messages using both RSA threshold encryption and decryption. In any given scenario, there is a sender *S*, a receiver *R*, and multiple shareholders *SH*.

4.1. RSA-TC assumptions

All the nodes have unique ids. *S* and *R* are already identified in the network. Since there is no trusted dealer/third party to generate shared keys, this function is carried out by *S* and it retains $\phi(N)$. *R* does the job of combining messages. Hence, a separate combiner is not defined. Multiple disjoint routes through each of the *SH* are already traced. Here we are not dealing with routing issues, so we assume that the routes can be identified using any of the available routing protocols.

4.2. Modules required in RSA-TC

4.2.1. Generation of RSA keys. Prime numbers *p*, *q* are generated using available functions in Java. The key size is variable, but we are recording data for 512, 1024, and 2048 bits. p and q are generated such that *(p-1) /2* and *(q-1)/2* are also prime [10]. Thus, the $\phi(N) = (p-1)*(q-1)$ would be divisible only by 2 and 4. From *p* and *q*, the RSA keys are determined such that $N = p*q$, *e* is selected, and then $d = e^{-1} \bmod \phi(N)$, where $\phi(N) = (p-1)*(q-1)$. Public key *(N, e)* is known to all shareholders and the receiver. The receiver uses it for checking integrity of the recovered message.

4.2.2. Determination of threshold: The sender's available neighbors will act as its shareholders. Based on *'n'* available neighbors, the threshold *t* is randomly generated such that $(t \geq (n+1)/2)$ and $t < n$, where $n \geq 2$.

In this implementation, *(n, t)* values are fixed to one of the following: (10, {6, 8, 10}), (15, {8, 11, 15}), or (20, {11, 15, 20}).

4.2.3. Share generation: For calculating key shares and for combining partial messages, Shamir's secret sharing scheme using Lagrange interpolation is implemented. For its polynomial, the coefficients are randomly generated over the modulus $\phi(N)$. The co-efficient zero depends on the type of threshold scheme. For threshold encryption, it would be *e*, while for threshold decryption it would be set to *d*. The x_i-values used for calculating the shares are 1 to *n*, rather than randomly picking these values. The generated shares and x_i-values are distributed among the shareholders during the key sharing process.

4.3. RSA-TC model

As shown in Fig. 2, applying Fermat's theorem [11] in our model, Lagrange interpolation and polynomial generation were carried out over mod $\phi(N)$ to generate the partial keys $f(x_i)$ as explained in Fig. 3. The shareholders only apply $f(x_i)s$ to the message and forward these partial signatures C_is along-with the x_i-values to the receiver. After receiving *t* or more C_is, the receiver selects *t* C_is for recovery of *C*. The receiver encrypts x_i-values using the sender's public key *e*, and sends it to the sender via more than one route. The sender calculates respective x_i'-values using Lagrange interpolation over mod $\phi(N)$ and sends them back to the receiver. The receiver then apply these x_i'-values to the respective partial signatures and combines the results to recover the final *C*. It then computes $C^e \bmod N$ to recover the final message M for verification.

As shown in Fig. 2, applying Fermat's theorem [11] in our model, Lagrange interpolation and polynomial

Given a prime p, let a be a positive integer number not divisible by p, then

$$a^{(p-1)} \equiv 1 \bmod p$$

Applying this theorem to RSA modulus N, we get

$$a^{(p-1)(q-1)} \equiv 1 \bmod (p*q) \text{ i.e. } a^{(N)} \equiv 1 \bmod N$$

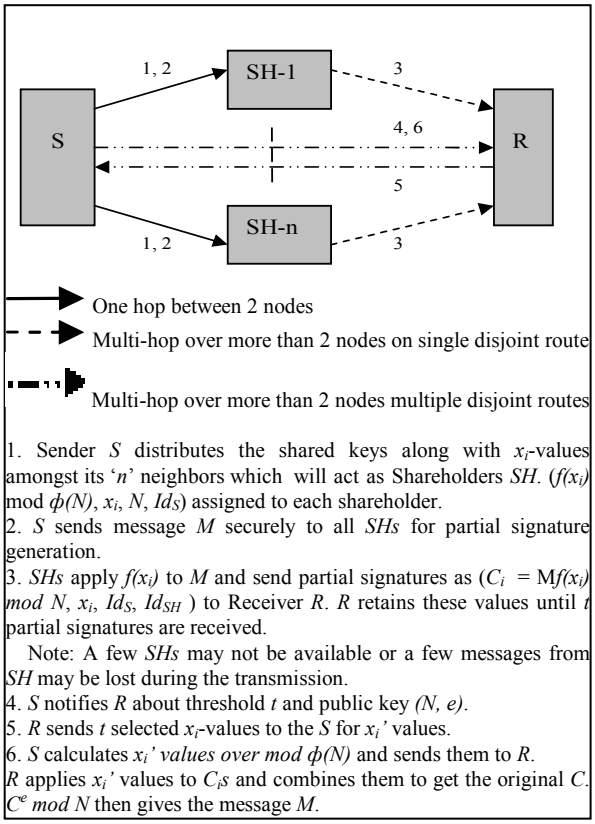
To get partial messages say C_i, it should be computed as:

$$C_i = M^{[\, f(xi) \bmod \phi(N) \, * \, xi' \bmod \phi(N)\,]} \bmod N$$

$f(x_i)$, $x_i' < \phi(N)$ as per Fermat's theorem.

Thus, $C = \prod C_i \bmod N$, where $i = 0...t$.

Figure 2. Fermat's theorem and its application in RSA-TC



1. Sender *S* distributes the shared keys along with x_i-values amongst its '*n*' neighbors which will act as Shareholders *SH*. ($f(x_i)$ mod $\phi(N)$, x_i, *N*, Id_S) assigned to each shareholder.
2. *S* sends message *M* securely to all *SHs* for partial signature generation.
3. *SHs* apply $f(x_i)$ to *M* and send partial signatures as ($C_i = M^{f(x_i)} \bmod N$, x_i, Id_S, Id_{SH}) to Receiver *R*. *R* retains these values until *t* partial signatures are received.

Note: A few *SHs* may not be available or a few messages from *SH* may be lost during the transmission.

4. *S* notifies *R* about threshold *t* and public key *(N, e)*.
5. *R* sends *t* selected x_i-values to the *S* for x_i' values.
6. *S* calculates x_i' *values over mod* $\phi(N)$ and sends them to *R*.

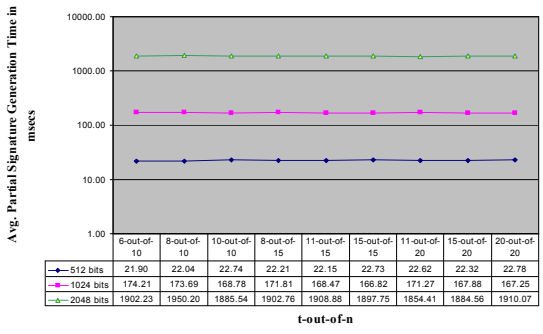
R applies x_i' values to C_is and combines them to get the original *C*. $C^e \bmod N$ then gives the message *M*.

Figure 3. Protocol for RSA-TC authentication/signature scheme in MANET

generation were carried out over mod $\phi(N)$ to generate the partial keys $f(x_i)$ as explained in Fig. 3. The shareholders only apply $f(x_i)s$ to the message and forward these partial signatures C_is along-with the x_i-values to the receiver. After receiving t or more C_is, the receiver selects t C_is for recovery of C. The receiver encrypts x_i-values using the sender's public key e, and sends it to the sender via more than one route. The sender calculates respective x_i'-values using Lagrange interpolation over mod $\phi(N)$ and sends them back to the receiver. The receiver then apply these x_i'-values to the respective partial signatures and combines the results to recover the final C. It then computes C^e *mod* N to recover the final message M for verification.

$\phi(N)$ is not shared with shareholders and no partial messages are stored at the shareholders. The sender carries out computation of the x_i'-values. Thus, the shareholders need not know t or other x_i-values that are obtained by the receiver. Instead of sending the x_i-values to all the shareholders, the receiver sends them to the sender via multiple reverse routes, less than t, thus reducing the message-exchanges carried over the wireless network. In this case, it does not affect the message-exchange even if a few

4.4. Performance data



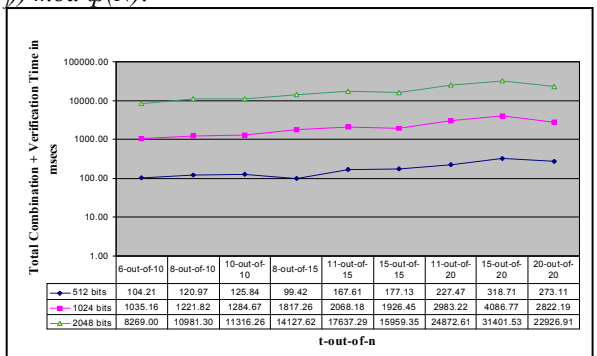
	6-out-of-10	8-out-of-10	10-out-of-10	8-out-of-15	11-out-of-15	15-out-of-15	11-out-of-20	15-out-of-20	20-out-of-20
512 bits	21.90	22.04	22.74	22.21	22.15	22.73	22.62	22.32	22.78
1024 bits	174.21	173.69	168.78	171.81	168.47	166.82	171.27	167.88	167.25
2048 bits	1902.23	1950.20	1885.54	1902.76	1908.88	1897.75	1854.41	1884.56	1910.07

Processor: SUN Sparc Ultra 5_10, Timings for 500 runs

Figure 4. Average Partial Signature Generation Time (E_{Time}) at each Shareholder in RSA-TC

In Fig. 4, for a given key size, the average signature generation time (E_{Time}) is approximately same at each shareholder irrespective of n and t because all the distributed shared keys have bit lengths equivalent to d. (Note that all the results, Fig. 4 and Fig. 5, have been collected for 500 runs.)

Fig. 5 proves that, for fixed n, the combination and verification time (D_{Time}) increases with increase in t which is equivalent to number of partial cipher texts C_is to be retrieved. But when $t = n$ the D_{Time} drops. In later case, calculation of x_i'-values is easier due to all consecutive x_i-values and multiplicative inverse always exists in mod $\phi(N)$ for all $x_i' = \prod_{j=0, j\neq i \text{ till } j=t} (x_j / (x_i - x_j)) \bmod \phi(N)$.



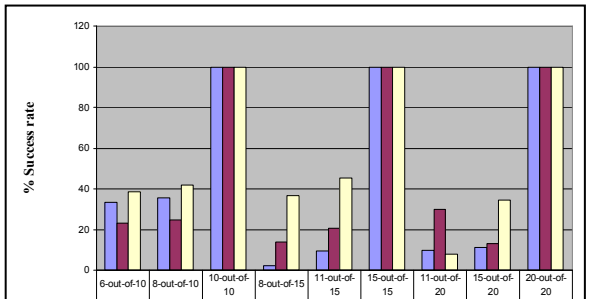
	6-out-of-10	8-out-of-10	10-out-of-10	8-out-of-15	11-out-of-15	15-out-of-15	11-out-of-20	15-out-of-20	20-out-of-20
512 bits	104.21	120.97	125.84	99.42	167.61	177.13	227.47	318.71	273.11
1024 bits	1035.16	1221.82	1284.67	1817.26	2068.18	1926.45	2983.22	4086.77	2822.19
2048 bits	8269.00	10981.30	11316.26	14127.62	17637.29	15959.35	24872.61	31401.53	22926.91

Processor: SUN Sparc Ultra 5_10, Timings for 500 runs

Figure 5. Total Combination and Verification Time in *mSecs* at Receiver in RSA-TC

Further, one would expect $D_{Time} >= (t * E_{Time})$, but the decryption key e is 65537, which is comparatively much smaller than d or any partial encryption keys $f(x_i)$. Also, not all the x_i'-values have bit size equal to $f(x_i)$. Taking into consideration above facts, the timings D_{Time} obtained here are justified i.e. $D_{Time} <= (t * E_{Time})$.

Fig. 4 and Fig. 5 demonstrate that increasing key size by 2, E_{Time} as well as D_{Time} increases exponentially. This is an expected behavior as in regular RSA scheme these timings increase exponentially by doubling key size.



	6-out-of-10	8-out-of-10	10-out-of-10	8-out-of-15	11-out-of-15	15-out-of-15	11-out-of-20	15-out-of-20	20-out-of-20
512 bits	33.4	35.4	100	2.4	9.2	100	9.8	11.2	100
1024 bits	23.2	24.8	100	13.8	20.6	100	29.8	13	100
2048 bits	38.4	41.8	100	36.8	45.2	100	7.67	34.4	100

Processor: SUN Sparc Ultra 5_10, Success rate for 500 runs

Figure 6. Success Rate in RSA-TC implementation for different t-out-of-n cases

% Success rate is measured as (Success/Total messages)*100. Note that failure = (Total messages – success). Fig. 6 compares %success rate for all *t-out-of-n* scenarios for given key sizes. Success rate varies for different key sizes when $t \neq n$. For all key sizes, for fixed n, success rate increases as t increases from $n/2$ to n, but, interestingly, optimal success rate is 100% when $t = n$. Next, though for fixed n and t, it seems that

increasing key size increases success rate, but note that selection of RSA keys and hence, $\phi(N)$ play a major role in this variation. Thus, for same key sizes and $t \neq n$, success rate will vary based solely on $\phi(N)$.

The observed advantage of RSA-TC is that success rate is 100% for $t = n$. Thus, if $\phi(N)$ is available with the sender, then using steps 5 and 6 in Fig. 3, *n-out-of-n* scheme can be implemented in MANET.

However, from above results, RSA-TC exhibits a few drawbacks that make it difficult to implement it in MANET. First, as $\phi(N)=(p-1)(q-1)$ is even, inverse of all numbers do not exist in mod $\phi(N)$ [6], [12]. Since Lagrange interpolation is carried out over mod $\phi(N)$, the question of determining n values of x, where all subsets of t x-values can re-compute x_i'-values, was raised. For maximum success rate at any n, t can be varied, and t that gives maximum success rate could be selected. It is observed in Fig. 6, that as t was gradually increased from $n/2$ to n, combinations of x_i-values i.e. $n!/((n-t)! * t!)$, decreased but the success rate of retrieving x_i'-values increased. Further, when $t = n$, success rate was 100%. Hence, for different t-values and given $\phi(N)$ at sender, pre-determination of set of x_i–values is required for a reasonable success rate.

Second, considering multiple computations and delays due to message exchanges with multiple nodes, receiver has to store partial messages until M is recovered. This may render the receiver incapable of storing more messages. In addition to this, given a key-size of z bits, each node in the network stores at least $3z$ bits, i.e. $(f(x_i) \bmod \phi(N), x_i, N)$, and a unique identity (Id) for each sender for which it acts as a shareholder. Note that the bit length of associated Id will be much less than z. For processing message signature generation and verification, additional memory is required to temporarily store intermediate results. Further, exponential calculations for $C_i = M^{f(xi)} \bmod \phi(N) \bmod N$ are very costly as bit length of $f(x_i)$ is equivalent to that of $\phi(N)$. Thus, RSA-TC imposes a significant load of storing and processing keys and messages at each node.

We would like to suggest alternative RSA-TC scheme. If $\phi(N)$ is secret but still RSA-TC is to be implemented then, instead of keys, message could be split before or after encryption. Lagrange's interpolation, in *mod N* field, could be used to divide message at the sender. In this scheme, shareholders are not required on disjoint routes. Since (e, N) is known so the receiver can calculate x_i'-values, thus eliminating the steps 5 and 6 in Fig. 3. In this case, the success rate would be 100% for any *t-out-of-n* case since N is multiple of two prime numbers. Also, x_i'-values would always be available in *mod N* field. But note that if message is split into n pieces before encryption this would increase RSA computations by n times. Hence, splitting message after encryption and then forwarding partial pieces on disjoint path would work and require encryption timings equivalent to a RSA scheme.

Table 1. Key sizes in bits for equivalent levels

Symmetric	ECC	DH/DSA/ RSA
80	163	1024
128	283	3072
192	409	7680
256	571	15,360

Table 2. Sample ECC exponentiation over GF(p)and RSA encrypt/Decrypt timings in mSecs

Processor	MHz	163-ECC	192-ECC	1024-RSAe	1024-RSAd	2048 RSAe	2048 RSAd
Ultra SPARCII 400MHz	450	6.1	8.7	1.7	32.1	6.1	205.5
StrongARM 200MHz	200	22.9	37.7	10.8	188.7	39.1	1273.8

ECC: rG operation, RSAe: RSA Public key operation, RSAd: RSA Private key operation

Table 3: 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
Demytko	18.3n	26.1n	68.7n	113.1n	12.2	17.4	45.8	75.4

Sun: Ultra Sparc II 450 MHz ARM: Strong ARM 200 MHz

Due to exponential computations, RSA scheme require lots of computational capacity, bandwidth, power, and storage. ECC-TC could be a better option in MANET. From Table 1 and 2 [4], ECC provides equivalent security as RSA, but with reduced key sizes and at faster speed. With smaller keys, ECC requires less memory and bandwidth and gives better efficiency than RSA [13]. Research has been done to prove that ECC scheme is suitable for applications on mobile devices [14]. Apart from above reasons, ECC works in prime field p, so we assume that compared to RSA-TC, ECC-TC would be easy to implement using Shamir's *t-out-of-n* scheme. Further, success rate could be 100%.

Many variants of ECC based algorithms exist such as *ECC El Gamal* [15], *EC Diffie-Hellman* [16] (*EC-*

DH), *Massey-Omura (MO)*, *Menezes-Vanstone (MV)*, *Koyama-Maurer-Okamoto-Vanstone (KMOV)*, *Ertaul*, *and Demytko* [17]. These variants can be modified to implement ECC-TC in MANET. From table 3 [17], *DH*, *MV* and *Ertaul* have been identified as best possible ECC-TC algorithms suitable for MANETs. These algorithms are efficient in both share split before and after encryption.

Moving forward, our goal is to implement ECC based *DH*, *MV*, *Ertaul*, and *El Gamal* for share as well as message splitting before and after encryption in simulated MANET environment and to compare its performance with RSA-TC.

5. Conclusions

In the RSA-TC implementation, we have proved that knowledge of $\phi(N)$ is must for sharing keys. It is clearly demonstrated here, that irrespective of key size and for known $\phi(N)$ at the sender, the success rate increases as t is increased from $n/2$ to n. Further, 100% success rate can be achieved with *n-out-of-n* RSA-TC scheme. As in regular RSA, RSA-TC implementation confirmed that the signature generation and signature verification time increases exponentially when key sizes are doubled. In this paper, it is established that the combining and verifying time is less than t times partial signature generation time. Rather than sharing keys, we have suggested an alternative of splitting the message at the sender to achieve 100% success rate without knowledge of $\phi(N)$. Thus, our work proves that RSA-TC using key sharing is unsuitable in resource-constrained MANETs due to high storage, computation, and bandwidth requirements. Finally, considering the growth of ad hoc networks in coming years, it is crucial to seriously consider the security of these networks. At this point, though RSA-TC is unsuitable for MANETs but ECC-TC appears (*DH, MV, Ertaul*, and *El Gamal*) to be an option to apply threshold cryptography in these networks. Further exploration of ECC-TC algorithms is required to prove that TC could be implemented to take a step closer in achieving enhanced ad hoc network security.

6. References

[1] A. Mishra and K. M. Nadkarni, "Security in wireless ad hoc networks – A Survey", in *The Handbook of Ad Hoc Wireless Networks,* M. Ilyas, Ed. Boca Raton: CRC Press, 2002, pp. 30.1-30.51.

[2] P. Papadimitratos and Z. Hass, "Securing Mobile Ad Hoc Networks", in *The Handbook of Ad Hoc Wireless Networks,* M. Ilyas, Ed. Boca Raton: CRC Press, 2002, pp. 31.1-31.17.

[3] H. Yang, H. Luo, F. Ye, S. Lu, and U. Zhang, "Security in Mobile Ad Hoc Networks: Challenges and Solutions", *IEEE Wireless Communications,* vol. 11, no. 1, Feb. 2004, pp. 38-47.

[4] K. Lauter, "The advantages of Elliptic Curve Cryptography For Wireless Security", *IEEE Wireless Communications,* vol. 11, no. 1, Feb. 2004, pp. 62-67.

[5] W. A. Arbaugh, "Wireless Security is Different", *IEEE Computer,* vol. 36, no. 8, Aug. 2003, pp. 99-101.

[6] Y. Desmedt and Y. Frankel, "Threshold cryptosystems", in *Advances in Cryptology - Crypto '89, Proceedings, Lecture Notes in Computer Science 435,* G. Brassard, Ed., Santa Barbara: Springer-Verlag,1990, pp. 307-315.

[7] Y. Desmedt, "Some Recent Research Aspects of Threshold Cryptography", in *Information Security, Proceedings (Lecture Notes in Computer Science 1396),* E. Okamoto, G. Davida, and M. Mambo, Eds., Tatsunokuchi: Springer-Verlag, 1997, pp. 158-173.

[8] Y. Desmedt and S. Jajodia, (1997, July). *"Redistributing secret shares to new access structures and its applications"*. Available: www.isse.gmu.edu/techrep/1997/97_01_jajodia.pdf

[9] L. Zhou, "Towards Fault-tolerant and Secure On-line Services". Ph.D. dissertation, Dept. of Computer Science, Cornell Univ., Ithaca, NY, 2001. Available: http://citeseer.ist.psu.edu/zhou01towards.html

[10] A. Lysyanskaya, (1999), "Efficient Threshold and Proactive Cryptography Secure against the Adaptive Adversary". Available: http://citeseer.ist.psu.edu/lysyanskaya99efficient.html

[11] W. Stallings, *Cryptography and Network Security: Principles and Practice.* Delhi: Pearson Education (Singapore), 2002, ch.7.

[12] M. Narasimha, G. Tsudik, and J. Yi, "On the Utility of Distributed Cryptography in P2P and MANETs: the Case of Membership Control." [Online]. Available: http://citeseer.ist.psu.edu/688081.html

[13] G. V. S. Raju, "Wireless Network Security." Available: http://cias.utsa.edu/Presentations/TIPS04-Raju.ppt

[14] W. Chou, "Elliptic Curve Cryptography and Its Applications to Mobile Devices." Available: http://www.cs.umd.edu/Honors/reports/ECCpaper.pdf

[15] T. Elgamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Transactions on Information Theory, vol. 31(4), July 1985, pp. 469-472.

[16] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computation,* vol. 48(177), pp. 203-209, 1987.

[17] L. Ertaul and W. Lu, "ECC Based Threshold Cryptography for Secure Data Forwarding and Secure Key Exchange in MANET (I)," *Networking 2005, LCNS 3462,* University of Waterloo, Canada, May 2005, pp. 102-113.