American Journal of Applied Sciences 9 (10): 1660-1667, 2012
ISSN 1546-9239

Enhanced RSACRT for Energy Efficient Authentication to Wireless Sensor Networks Security

[1]Kalyani, P. and [2]C. Chellappan
[1]Department of Computer Science and Engineering, I.R.T.T., Erode, India
[2]Department of Computer Science and Engineering, Anna University Chennai, India

Abstract: Problem statement: The Security in WSN has become a challenge because of its inherent limitations of resources imposed on the sensor nodes. Authentication and key management scheme allows WSNs to be used with confidence and maintains integrity of data in sensor networks. **Approach:** In the WSN limited resource scenario the generation of public key done and are randomly pre distributed while deploying the sensor nodes. Since the sensor nodes are limited by energy it needs to concentrate on fast decryption and verification of message. In order to enable the fast decryption we have proposed a novel method which uses three algorithms hybrid to achieve the increase in decryption speed which in turn reduces the energy used for computation and enhances its performance compared with the existing authentication using classical RSA algorithm. **Results:** The proposed Enhanced Variant of RSA with CRT using Garner's algorithm to achieve fast decryption speed and provides better performance when compared to the existing RSA. The private key is generated and passed so that the receiver node need not generate it which consumes more computational cost, power and memory at the decryption stage. Also we have done the signing and verification which avoids the message spoofing attack and enable message confidentiality. Further the ERSACRT is designed to counter measure to few attacks possible on RSA. We implemented the ERSACRT in java and tested for system parameters like memory, time, speed and efficiency and compared with that of RSA. **Conclusion/Recommendation:** The proposed algorithm ERSACRT is efficient and secured along with improved counter measures for secured communication in WSN with reduced energy and computational time.

Key words: Wireless Sensor Networks (WSN), authentication in WSN, key management scheme, classical RSA, RSACRT, ERSACRT

INTRODUCTION

Wireless sensor networks are used for various applications in a wide variety of areas since they provide better solutions to real world problems. The security in WSN has become a critical issue because of the severe resource constraints like limited computational capacity, limited battery power consumption and memory imposed on the sensor nodes. But it appears that they are more prone to attacks than wired networks. Sensor networks consist of numerous low cost, little devices and are in nature self organizing ad hoc systems. The job of the sensor network is to monitor the physical environment, gather and transmit the information to other sink nodes. Generally, radio transmission ranges for the sensor networks are in the orders of the magnitude that is lesser that of the geographical scope of the unbroken network (Vass and Vidacs, 2007). Hence, the transmission of data is done from hop-by-hop to the sink in a multi-hop manner. Reducing the amount of data to be relayed thereby reduces the consumption of energy in the network.

Yang *et al*. (2004) surveyed about the challenges and issues of security in wireless sensor networks. They are susceptible to a variety of attacks, including node capture, physical tampering and denial of service, prompting a range of fundamental research challenges, an attacker can easily eavesdrop on, inject or alter the data transmitted between sensor nodes. Security allows WSNs to be used with confidence and maintains integrity of data. Providing security in wireless sensor networks is pivotal due to the fact that sensor nodes are inherently limited by resources such as power, bandwidth, computation and storage. Law *et al*. (2005) suggested several methods to overcome the issues and all the operations are sensitive to possible attacks and they have not concentrated on the key management schemes. Authentication and key management scheme

Corresponding Author: Kalyani, P., Department of Computer Science and Engineering, I.R.T.T., Erode, India

only makes sure the communicating nodes possess the necessary keys, at the same time protecting the confidentiality, integrity and authenticity of the communicated data in the wireless sensor networks.

Related work: Bhoopathy and Parvathi (2012) suggested an Energy Constrained Secure Hierarchical Data Aggregation in Wireless Sensor Networks using key management scheme. Based on distance to sensor nodes and its energy level the aggregator detects the node. Separate keys were distributed to the two levels i.e., sensor node to the aggregator and aggregator to the sink. Whenever a data had to be sent from a sensor node to another node; initially the sensor node encrypts the data using a key and sends it to the aggregator. Node compromise is the major problem in sensor networks that lead to internal attacks. In contrast to disables nodes, compromised nodes actively seek to disrupt or paralyze the network. From Song *et al*. (2007) the method of obtaining the compromised nodes and counters are given. Attackers capture the sensor node and reprogram them. WSNs can be locate the compromised nodes by monitor node activity, location. Attacker can deploy nodes with larger computing capability such as laptops to attack sensor nodes.

Xiong *et al*. (2010) proposed a fast and Lightweight Pairing-based Cryptographic Library for Wireless Sensor Networks. They present the first fully functional pairing-based cryptographic library for WSNs. The library is fast and lightweight and has an additional of one identity-based encryption scheme and two short signature schemes included. They also proposed several new algorithms and techniques and show that they significantly improve the speed and reduce the memory usage of the library. The performance results of implementing the three pairing-based cryptographic schemes show that pairing based cryptosystems are feasible and applicable in WSNs.

By applying forwarding algorithm technique based on CRT, it is possible to reduce the energy consumed for each node and consequently to increase the network lifetime of WSN (Campobello *et al*., 2010). Hence methods to enhance security and efficiency of routing protocols for WSN have gained importance. Campagna and Sethi (2004) analyzed a key recovery method for RSA signature generation or decryption implementations using the Chinese Remainder Theorem (CRT) speed up. The CRT-based RSA implementation is common in both low computing power devices and high speed cryptographic acceleration cards. This recovery method is designed to work in conjunction with a side-channel attack where the CRT exponents are discovered from a message decryption or signature generation operation, the public exponent is assumed small and the public modulus is unknown.

Calculations made by Amin *et al*. (2008) show that RSA is not well suited for WSNs. Comparing ECC-160 (Elliptic Curve Cryptosystem) and RSA-1024 indicates that the effort needed for RSA cryptography is rather too much. While the application of the even stronger ECC-224 still seems to be feasible, the time and power consumption for the equivalent RSA-2048 is far beyond the acceptable level. In addition to key management and secure communication, public-key cryptography can be the enabling technology for numerous other WSN applications, including securely connecting pervasive devices to the Internet and distributing signed software patches.

The RSA uses less computational power and hence increases the network lifetime when compared with ELGAMAL. From the observation made by Kayalvizhi *et al*. (2010), RSA algorithm provides better security for wireless sensor networks and it consumes 14.5% less energy than the ELGAMAL algorithm. Therefore, RSA is adoptable for the wireless sensor networks as it consumes less energy. The requirement for energy efficiency suggests that in most cases computation is favored over communication, as communication is three orders of magnitude more expensive than computation as suggested by Wander *et al*. (2005). The requirement also suggests that security should never be overdone. More computationally intensive algorithms cannot be used to incorporate security due to energy considerations.

Problem identification: There are many key management schemes are available for the authentication, we proposed a key management scheme called Enhanced Variant of RSA with CRT using Garner's algorithm to achieve fast decryption speed and provides better performance when compared to the existing RSA. We take the advantage of using CRT and shifting some of the decryption function to the key generation phase and calculation of time consuming operations are done at the key generation phase itself. Also the private key is generated and passed so that the receiver need not generate it which consumes more computational cost, power and memory at the decryption stage. Also we have done the signing and verification during the encryption and decryption phase itself it avoids the message spoofing by any intruder. Thus making it to secure against the message spoofing attack and enable message confidentiality. Further the ERSACRT is designed to counter measure to few attacks possible on RSA like Mathematical attack

(numeric or algebraic), Timing attack a part of side channel attack, Small Private Key attack (SPK), Short Secret Exponent attack (SSE). We implemented the ERSACRT in java and tested for system parameters like memory, time, speed and efficiency and compared with that of RSA.

Classical RSA algorithm: The RSA cryptosystem, is the most widely used public key Cryptosystem. This is a block cipher in which the plaintext and cipher text are integers between 0 and N for some n bits. A typical size for n is 1024 bits is optimal. The RSA consists of three phases namely 1. Key generation, 2. Encryption 3. Decryption:

- Key Generation Public key is {N, e} and exponent is {e} and Private key is {p, q, d} and private exponent is {d}
- Encryption: Input : M, N, e; Algorithm : C ← M^e Mod N; Output : C
- Decryption: Input : C, d, N; Algorithm : M ←C^d mod N; Output : M
- Memory cost: System parameters: C,M,N are n bits in size needs 3 registers, d, e are of n/2 bits in size needs each one register. So the total memory = 3n+n/2+n/2 = 4n where n is the bit length of modulus N

Attacks on RSA: The following types of attacks are possible if RSA algorithm is used. They are 1.Mathematical attack(numeric or algebraic) 2. Timing attack a part of side channel attack 3. Small Private Key attack (SPK) 4. Short Secret Exponent attack (SSE)

Mathematical attack: If the values of φ (n) and the public key (the modulus N and exponent e) are known, then it is possible to determine the private key d with the use of extended Euclidean algorithm. Given a, b, we can find 2 integers x, y such as: ax + by = gcd (a, b)

Timing attack: RSA factors can be recovered by measuring the amount of time taken to perform the operation on keys. This type of attack is known as timing attack. By analyzing the computation time for C^d mod n with several values of C, the attacker can recover the private key d, bit by bit.

Small Private Key attack (SPK): To improve the RSA decryption performance in the matter of running-time, Alice might tend to use a small value of d, rather than a large random number. A small private key indeed will improve performance dramatically, but unfortunately, an attack posed by Wiener (1990) shows that a small d leads to a total collapse of RSA cryptosystem. This break of RSA is based on Wiener's Theorem, which in general provides a lower constraint for d. Wiener has proved that the value of d is efficient when d <1/3 ∗N^(1/4).

Short secret exponent attack: The time to carry out modular exponentiation increases with the number of bits set to one in the exponent *e*. For encryption, an appropriate choice of *e* can reduce the computational effort required to carry out the computation of c = m^e mod n. Popular choices like 3, 17 and 65537 are all primes with only two bits set: 3 = 0011'B, 17 = 0×11, 65537 = 0×10001. When e = 3 used in basic RSA it is susceptible to short secret exponent attack even though small e increases the speed. So in general the e value used for standard RSA is 65537.

MATERIALS AND METHODS

Fast decryption RSA-CRT algorithm: The public key is given by the pair (e, N) and the private key is given by the tuple (dp, dq, p, q). Note that e will be roughly the same order of magnitude as ϕ (N) with high probability. Thus, in order to reduce the decryption time even further in RSA-CRT, the encryption time is essentially maximized. As with RSA in order to be secure the values of the CRT-exponents, dp and dq, cannot be chosen to be arbitrarily small. Since the computations of p^{-1} mod q and d^{-1} are calculated one time during the decryption phase in our previously proposed algorithm Palanisamy and Chellappan (2011). Further the speed of the algorithm is improved. Also in the decryption phase according to the Euler's theorem if the value of N is done with the totient: ϕ (N) = (p-1) (q-1), it will increase the speed since the p, q are half bit size (n_d) when compared to n bit. Assuming schoolbook multiplication with operands of size m/2 = $[\log_2(n)]/2$, modular multiplications can be computed in roughly 1/4 of the time as m-bit modular multiplications. Thus the CRT reduces computation time through Montgomery multiplication by nearly 3/4 resulting in up to a 4x speedup.

RSACRT algorithm:

- Key Generation: the generated keys are Public key is {N, e}and Private key is {p, q, dp, dq}
- Encryption: Input: M, N, e; Algorithm: C ←M^e Mod N; Output: C
- Decryption: Input: C, p, q, Dp, Dq; and Output: M
- Algorithm: Cp←C^{dp}mod p; Cq←C^{dq} mod q; Mo ← ((Cq-Cp).p^{-1} mod q)mod q; M ←Cp+Mo.p; return M

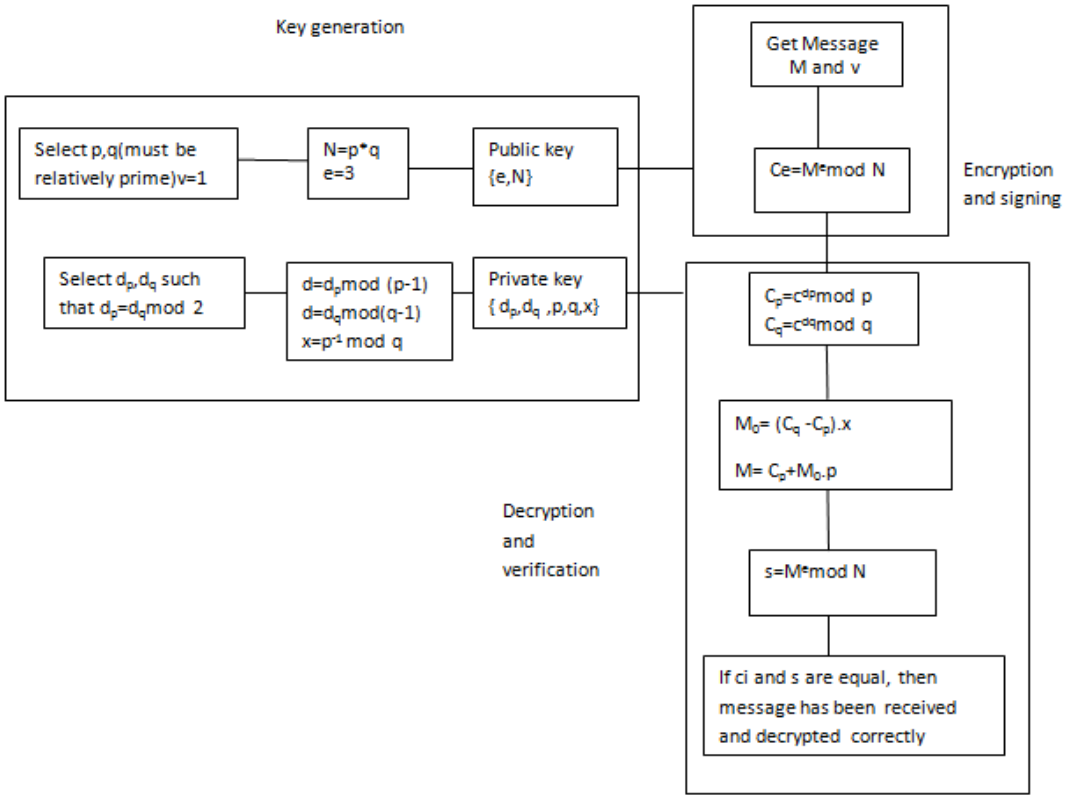

Fig. 1: Functional Blocks of ERSA-CRT with Signing and Verification

Memory cost: System parameters: n is the bit length of N , since N = pq ,the bit length of p and q is n/2 and so the bit length of the CRT exponents dp, dq is also n/2. M_0 is the intermediate variable of n/2 bit size. C, M, N is n bit size. The memory Cost = 3n+4n/2 = 5n. Further C_p, C_q, M_0 are intermediate cipher text and message variables with n/2 bit size. So the accumulator memory is 3n/2. The total memory for this RSA with CRT is 5n +3n/2 = 13n/2. System memory + Accumulator memory)

Security analysis: The speed of the algorithm is based on the selection of exponent value. It is not possible to select d as small value for RSA as it would not be secure. If we select small d then the algorithm will be attacked, called small private key attack Wiener (1990) proved in his work that a small d leads to a total collapse of RSA cryptosystem. In RSACRT algorithm the decrypting exponents of CRT are dp, dq instead of d. we do not compute modulus d, instead we compute modulus of factors (dp and dq) .So this method is secure against SPK attack i.e., small d attack (i.e.,) the attack by Wiener's theorem does not apply here because the value of d mod ϕ (N) can be large.

Enhanced variant of RSACRT using Garner's algorithm: We propose an energy constrained secure key management scheme for authentication in wireless sensor networks to maintain data confidentiality and integrity. In order to further speed up the function of decryption, we used the concept of Garner's algorithm and applying the Euler's totient function ϕ(n) in the decryption phase of CRT in RSACRT. The algorithm which shifts some of the work done by decryption to the key generation phase of RSA CRT. The calculation of p^{-1} (mod q) takes more time and can be computed in advance in the key generation phase and passed as private key of n_d bit along with other key values to the decryption phase. This version of RSACRT is called as Enhanced Variant of RSACRT with improved performance and secure against the attacks like Mathematical attack, Timing attack, Small Private key attack, Fault compute attack. The functional block diagram of the ERSACRT is given below in Fig. 1. The sign and verification used counters the message spoofing attack and provides message authenticity and integrity.

ERSACRT algorithm:

Key generation:

Step 1: Select two random prime integers p and q of nearly the same size
Step 2: Compute N = p*q
Step 3: Select dp and dq such that gcd(dp, p-1)= 1, gcd(dq, p-1) = 1 and dp = = dq mod 2.
Step 4: Compute d such that d = = dp mod (p-1) and d = = dq mod (q-1)
Step 5: Compute $e = d^{-1}$ (mod ϕ (N)), where ϕ (N) = (p-1)(q-1)
Step 6: Compute $x = p^{-1}$ mod q
Hence the generated keys are Public key {N,e}and Private key {p,q,dp,dq,x}
Encryption: Input: M, N, e; Output: C; Algorithm:
C $\leftarrow M^e$ Mod N; Return C;
Decryption: Input: C, P, Q, Dp, Dq,X; Output: M;
Algorithm: Cp$\leftarrow C^{dp}$mod P; Cq$\leftarrow C^{dq}$ Mod Q; Mo $\leftarrow$ ((Cq-Cp).X) Mod Q; M $\leftarrow$Cp+Mo.P; Return M;2.4. Memory Cost

System parameters: n is the bit length of N , since N = pq ,the bit length of p and q is n/2 and so the bit length of the CRT exponents dp, dq is also n/2. M_0 is the intermediate variable of n/2 bit size. Further x is given by p^{-1} mod q of n/2 bit size is the intermediate variable passed as private key from the key generation phase.

Cost = 3n+4n/2+n/2 = 11n/2. Further $Cp_{, Cq,} M_0$ are intermediate cipher text and message variables with n/2 bit size. So the accumulator memory is 3n/2. So Total cost = (System memory + Accumulator) = 11n/2+3n/2 = 7n.

Security analysis: The time to carry out modular exponentiation increases with the number of bits set to one in the exponent *e*. For encryption, an appropriate choice of *e* can reduce the computational effort required to carry out the computation of $c = m^e$ mod n. Popular choices like 3, 17 and 65537 are all Fermat's primes with only two bits set: 3 = 0011'B, 17 = 0×11, 65537 = 0×10001. When e = 3 used in basic RSA it is susceptible to short secret exponent attack even though small e increases the speed. So in general the e value used for standard RSA is 65537. In proposed ERSACRT we used the value for e is 3. In this algorithm it is not possible for short secret exponent attack since the value of e is calculated as $e = d^{-1}$ (mod ϕ (N)), where ϕ(N) = (p-1)(q-1) and the value of d = dp mod (p-1) , d = dq mod (q-1).

RESULTS

Based on the result of the implementation of the algorithms in java we have given Comparisons of Performance of RSA, RSACRT and ERSACRT with respect to the system parameters, memory utilized for computation and time taken decryption. The results are shown in the Table 1-3 and in the Fig. 2 and 3.

Table 1: Memory comparison

	Total memory	System memory	Accumulators
RSA without CRT	4n bits	4n bits	0 bit
n=1024 bits	4096 bits	4096 bits	0 bit
RSA with CRT	13n/2 bits	5n bits	3n/2 bits
n=1024 bits	6656 bits	5120 bits	1536 bits
ERSACRT with garner	7n bits	11n/2 bits	3n/2 bits
n=1024 bits	7168 bits	5632 bits	1536 bits

Table 2: Storage space requirements (In Mb)

Key size	160 bits	256 bits	512 bits	1024 bits	2048 bits
RSA	0.164856	0.166184	0.166376	0.166728	0.167152
RSA CRT	0.134144	0.134384	0.134864	0.135824	0.138110
ERSACRT	0.13472 0	0.135008	0.135584	0.136608	0.139040

Table 3: Comparison of decryption time

Key size (bits)	RSA (ns)	RSA CRT (ns)	E RSA-CRT (ns)
160	1662599	1739996(-1.04 Times)	1621268(1.02 times)
256	2135892	2342728(-1.09 Times)	2086860(1.02 times)
512	7511887	4044862(1.85 times)	3434607(2.1 times)
1024	44231715	9908789(4.4 times)	8650542 (5.9 times)
2048	318372233	79440652(4 times)	47163045(6.7 times)

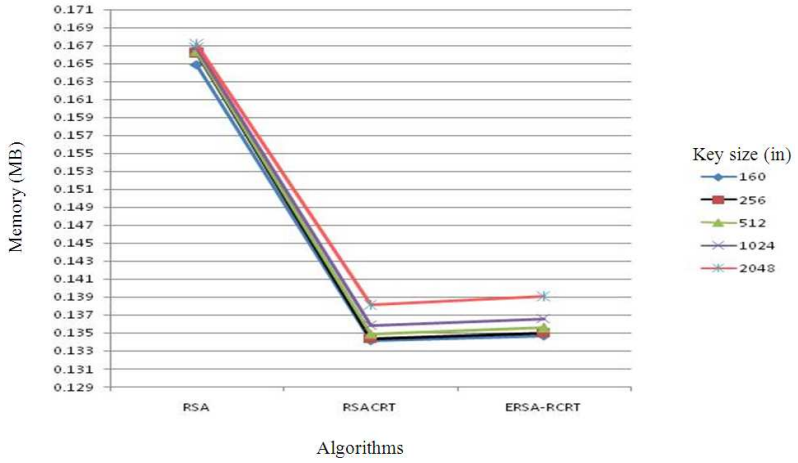

Fig. 2: Memory utilized by RSA, RSACRT, ERSACRT for various key sizes

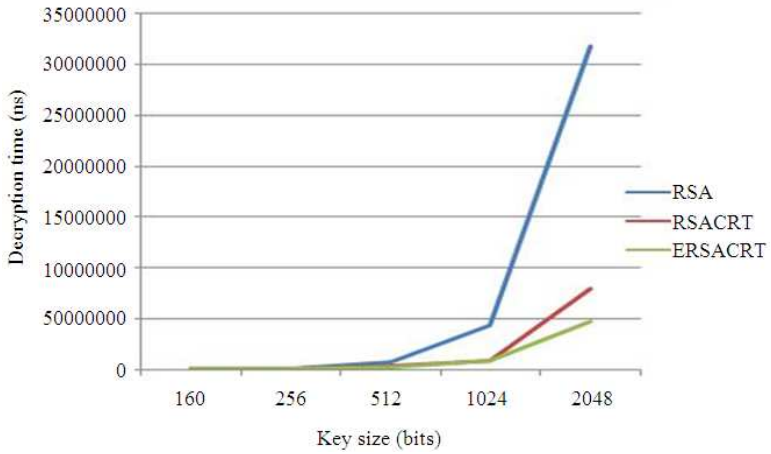

Fig. 3: Decryption Time for RSA, RSACRT and ERSACRT

DISCUSSION

Time cost: For RSA, choosing e small ($2^{16+1} = 65537$) almost effectively yields an encryption running time dependent only on the bit length n of the modulus N, i.e. $O(n^2)$. The structure of the decryption exponent d cannot be tailored to fit the repeated square-and-multiply algorithm in the same way, but will often be long and consist of a random number of 1's in its binary representation. The worst case scenario is that $|d| \approx |N|$ yielding a running time of $O(n^3)$. Hence the time complexity of RSA decryption is $O(n^3)$.In case of RSA CRT, the N value is divided into p and q and then computations are carried out for modulus of half the size, for both Cp and Cq. Hence the time complexity of RSA CRT decryption is $2*[O (n/2)^3]$. This is four times faster than the conventional RSA.

In case of ERSA CRT, the $p^{-1} \bmod q$ is computed at key generation phase, which is equivalent to 20 multiplications. So the time complexity of ERSA CRT decryption is $2*[O (n/2)^3]- 40 n^2/2 +n$. This means that ERSA CRT is five times faster than RSA. Also, we have used the Euler totient φ(N) instead of N for calculating d. That is, $d = e^{-1} \bmod \varphi(N)$ reduces the costs of multiplication since it uses (p-1)(q-1) instead of p*q. In addition to the above we have taken small e (e = 3) which will also increases the speed further. Thus the complexity becomes ((n/2)-1)*((n/2)-1), instead of (n/2)*(n/2) along with that of e = 3. Thus the computation of dp and dq, in turn requires lesser computations and hence we arrive nearly 6.7 times faster decryption in ERSA CRT, compared to the original RSA which is shown in the result of our implementation.

Memory comparison: RSA CRT decryption is about 4 times faster with CRT and ERSACRT with Garner's is 6 times faster. However the speed up in decryption time of RSACRT and ERSACRT costs more memory i.e., it consumes 1.63 and 1.75 times more memory than that of RSA shown in Table 1 and this memory cost is acceptable when considering the increased speed of decryption.

Security measures against attacks:

Brute force: It comprises of methodically examining all the possible keys with n bits (1024) in anticipation of the exact key to be found.. Since the key size chosen here is very large and the private keys are generated as CRT exponent with n/2 bits (512) and passed it is highly time consuming and is not possible because the decryption time of the chosen cipher text is very small in the order of nanoseconds in the proposed scheme.

Mathematical attack: The first attack is the attempt to factor the modulus n. Because knowing the factorization of n, we can obtain φ (n), from which d can be determined by d = 1/e mod φ (n).The factoring algorithm takes exponential time to determine the factors. In our case, this equation become de + k. φ (n) = 1 because gcd (e, φ(n)) =1 along with another condition which is gcd (d, φ(n)) =1. Thus for knowing the private key d to read the message is more difficult and takes exponential time.

Small Private Key attack (SPK): To improve the RSA decryption performance in the matter of running-time, one should use a small value of d, rather than a large random number. A small private key indeed will improve performance dramatically, but unfortunately, an attack posed by Wiener theorem shows that a small d leads to a total collapse of RSA cryptosystem and provides a lower constraint for d. Wiener has proved that the value of d is efficient when $d < 1/3 * N^{(1/4)}$. In our method Small Private Key attack is not possible in RSACRT and ERSA CRT because we do not compute modulus d, instead we compute modulus of factors (dp and dq) thus making them stronger and secure against the attack by using CRT exponents in decryption phase with small private exponent.

Timing attack or chosen cipher text attack: By precisely measuring the time taken to perform an RSA decryption, the attacker can quickly discover the private decryption exponent *d*. By analyzing the computation time for C^d mod n with several values of C, the attacker can recover the private key d in bit by bit. Both of these SPK and Timing attacks are not possible in RSACRT and ERSA CRT because we do not compute modulus d, instead we compute modulus of factors (dp and dq) thus making RSA CRT stronger and secure against timing attack by using CRT in decryption phase. Further the private key for ERSACRT is {p, q, dp, dq, x} where the x value is calculated in key generation phase as p inverse modulus q.

Here the decryption time is further reduced by applying Garner's algorithm in CRT and is very faster than the basic RSA algorithm.

Memory requirement: The memory utilized by normal RSA and proposed methods RSACRT and ERSACRT are compared and given in the following Table 2 also graph drawn for the memory table and compared in Fig. 2. From the graph it is obvious that the memory utilized by our proposed algorithms is less than the normal RSA algorithm for the same key size. This is possible because the CRT exponent dp, dq are used for decrypting the message which are only half the size of n bit key N. Further in the ERSACRT we shifted the load of CRT, i.e., the calculation of p^{-1} mod q from the decryption phase to the key generation phase and the value is passed as private key. Also it is noted that the memory usage for decryption is almost constant for any key size. Hence it is better to consider the time and speed of decryption which alone will be the prime factor for energy consumption.

Decryption time and speed: From the implementation of the algorithms in java code, the time taken for decryption process of RSA, RSACRT and ERSACRT are given in the Table 3. The timing table shows the reduction of decryption time for various key sizes. From the result it is clear that the decryption time is much reduced in our proposed RSACRT and ERSACRT algorithm when compared to standard RSA algorithm. For example n=2048, the speed of ERSACRT is almost **seven** times faster and for RSACRT it is **four** times faster. The Fig. 3 shows the comparison of decryption time of RSA; RSACRT and proposed ERSACRT algorithms.

From the above graph (Fig. 3) it is noted that the time taken for decryption for the key size 160, 256, 512bits are almost same. For the n = 1024 only slight increase in the time and for greater key size the time taken is also high. Hence we have to consider the optimal value for key size which is 1024 because if the key size increased the memory also linearly increases. Above graph shows that the performance of ERSA-CRT is better when compared to RSA and RSA CRT. The improved performance can be clearly observed as the key size gets increased. At lower key size, the

performance of RSA and ERSA CRT are more or less equal. This is because the key size is small and hence, decryption computations in RSA become quite simpler but in ERSA CRT, the extra cost is implemented in CRT calculations. Hence from the tested result given above our proposed methods gives better performance with regards to memory, speed and time cost which in turn the energy consumed along with strong and secure against attacks also.

CONCLUSION

The vulnerability of RSA is studied and it is concluded that the system using RSA is not always protected in spite of its strength. The timing attack showed that the attacker can break the system without knowing the encryption and decryption algorithm. Thus the proposed algorithm is efficient and secured along with improved counter measures for secured communication in WSN with reduced decryption time which intern reduces the energy consumption and communication overheads. From the implementation results it is clearly shown that our proposed authentication schemes RSACRT, ERSACRT are well suited for authentication to Wireless Sensor Network security.

REFERENCES

Amin, F., A.H. Jahangir and H. Rasifard, 2008. Analysis of public-key cryptography for wireless sensor networks security. World Acad. Sci. Eng. Technol., 41: 529-534.

Bhoopathy, V. and R.M.S. Parvathi, 2012. Energy constrained secure hierarchical data aggregation in wireless sensor networks. Am. J. Applied Sci., 9: 858-864. DOI: 10.3844/ajassp.2012.858.864

Campagna, M. and A Sethi, 2004. Key recovery method for CRT implementation of RSA. Pitney Bowes, Inc.

Campobello, G., A. Leonardi and S. Palazzo, 2010. Energy saving and reliability in wireless sensor networks using a CRT based packet splitting algorithm. University of Messina, Italy.

Kayalvizhi, R., M. Vijayalakshmi and V. Vaidehi, 2010. Energy analysis of RSA and ELGAMAL algorithms for wireless sensor networks. Commun. Comput. Inform. Sci., 89: 172-180. DOI: 10.1007/978-3-642-14478-3_18

Law, Y.W., P. Hartel, J.D. Hartog and P. Havinga, 2005. Link-layer jamming attacks on S-MAC. Proceedings of the 2nd European Workshop on Wireless Sensor Networks, Jan. 31-Feb. 31, IEEE Xplore Press, pp: 217-225. DOI: 10.1109/EWSN.2005.1462013

Palanisamy, K. and C. Chellappan, 2011. Authenticated broadcast in heterogeneous wireless sensor networks using Chinese remainder theorem algorithm. J. Comput. Sci., 7: 849-853. DOI: 10.3844/jcssp.2011.849.853

Song, H., L. Xie, S. Zhu and G. Cao, 2007. Sensor node compromise detection: The location perspective. Proceedings of the 2007 International Conference on Wireless Communications and Mobile Computing, (IWCMC' 07), ACM Press, New York, USA., pp: 242-247. DOI: 10.1145/1280940.1280993

Vass, D. and A. Vidacs, 2007. Distributed data agregation with geographical routing in wireless sensor networks. Proceedings of the IEEE International Conference on Pervasive Services, Jul. 15-20, IEEE Xplore Press, Istanbul, pp: 68-71. DOI: 10.1109/PERSER.2007.4283891

Wander, A.S., N. Gura, H. Eberle, V. Gupta and S.C. Shantz, 2005. Energy analysis of public-key cryptography for wireless sensor networks. Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications, Mar. 8-12, IEEE Xplore Press, pp: 324-328. DOI: 10.1109/PERCOM.2005.18

Wiener, M., 1990. Cryptanalysis of short RSA secret exponents. IEEE Trans. Inform. Theory, 36: 553-558. DOI: 10.1109/18.54902

Xiong, X., D.S. Wong and X. Deng, 2010. TinyPairing: A fast and lightweight pairing-based cryptographic library for wireless sensor networks. Proceedings of the 2010 IEEE Wireless Communications and Networking Conference, Apr. 18-21, IEEE Xplore Press, Sydney, pp: 1-6. DOI: 10.1109/WCNC.2010.5506580

Yang, H., H. Luo, F. Ye, S. Lu and L. Zhang, 2004. Security in mobile ad hoc networks: Challenges and solutions. IEEE Wireless Commun., 11: 38-47. DOI: 10.1109/MWC.2004.1269716