Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 89 (2016) 187 – 192

Procedia Computer Science

Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016)

Mutual Entity Authentication Protocol Based on ECDSA for WSN

Ayaz Hassan Moon[a,*], Ummer Iqbal[a] and G. Mohiuddin Bhat[b]

[a] *National Institute of Electronics and Information Technology, J & K*
[b] *University of Kashmir J & K*

Abstract

Digital Signatures are primarily employed to achieve data integrity, data origin authentication and non-repudiation. In comparison to RSA-based digital signatures, the ECC (Elliptical Curve Cryptography) based Signature schemes like ECDSA (Elliptical Curve Digital Signature Algorithm) are more efficient and suited for a resource-constrained network like WSN. In a WSN, proper entity authentication is to be achieved before the sensor data is exchanged. The use of ECDSA is not suitable for achieving mutual authentication between the entities like the base station, cluster heads and nodes. Based on the study of ECDSA, the paper designs a mutual authentication protocol with the help of a computationally low signature scheme. To the best of our knowledge, this is the first digital signature based scheme which achieves mutual entity authentication. The paper also brings out performance parameters of the developed protocol.

Keywords: Authentication; Mutual Entity Authentication; Wireless Sensors Network; ECDSA; TinyOS.

1. Introduction

Authentication is an assurance about the identities of communicating nodes or principals in any network. It involves a process to ascertain that the data has come from the alleged source and has not been modified en route. An adversary may spoof the source address and then inject malicious packets into the network so as to camouflage its originality[1]. As WSN primarily involve sensor data, based on which certain alarms and actuators are activated, authentication becomes imperative. Mutual entity authentication plays an important role in securing WSN against many types of attacks like impersonation, falsification of data, replay attacks etc. leading to DoS. Not many computationally efficient schemes have been developed to achieve this security primitive suitable for WSN.

DSA is a widely acknowledged mechanism for accomplishing authentication. It was initially proposed by the National Institute of Standards and Technology (NIST) in August 1991 and was notified in Federal Information Processing Standard of U.S Government called the Digital Signature Standard (DSS). Elliptic Curve Digital Signature Algorithm (ECDSA) which is based on DSA, provides a proficient message authentication scheme for WSN[2]. In this paper based on the study and research of the ECDSA, a mutual authentication scheme comprising of 4 phases has been proposed.

*Corresponding author. Tel.: 0194-2300502; Fax: 0194-2300949.
E-mail address: moonah@rediffmail.com

doi:10.1016/j.procs.2016.06.038

Table 1. Signature Generation.

Input: parameter group $D = (q, FR, S, a, b, P, n, h)$, private key d, message m.	
Step	Operation
1	Selecting k
2	Calculation: Calculate $kP = (x_1, y_1)$
3	Calculation: Calculate $r = \bar{x}_1 \bmod n$. If $r = 0$, then to 1).
4	Calculation: Calculate $e = H(m)$.
5	Calculation: Calculate $s = k^{-1}(e + dr) \bmod n$. If $s = 0$, then to 1).
6	Return: Return (r, s).
Output: signature (r, s).	

Table 2. Signature Verification.

Input: parameter group $D = (q, FR, S, a, b, P, n, h)$, private key Q, message m, signature (r, s).	
Step	Operation
1	Check out weather r and s are between 1 and $n - 1$. If anyone is not, the verify result is false.
2	Calculate $e = H(m)$.
3	Calculate $w = s^{-1} \bmod n$
4	Calculate $u_1 = ew \bmod n$
5	Calculate $X = u_1 p + u_2 Q$
6	If $X = \infty$, then refuse this signature else Take the x-axis coordinates x_1 of X to integer X_1, calculate $v = x_1 \bmod n$
Output: Return: If $v = r$, return acceptance of this signature, else return not accept	

1.1 Elliptical Curve Discrete Log Problem

ECC based cryptosystem was proposed by Miller[3] and Koblitz[4]. An elliptic curve comprises of a set of numbers (x, y), regarded as points on that curve. The curve has to satisfy the following equation:

$$y^2 = x^3 + ax + b \tag{1}$$

Elliptic curve cryptography offers a higher level of security in comparison to more traditional public crypto scheme like RSA. The hardness of the curve is assumed because of its elliptic curve discrete logarithmic problem ECDLP[5] nature, for which the best known algorithm is exponential[6]. This enables ECC to work with smaller key size to achieve the same level of security as RSA does with comparatively larger key size. It has been reported than 160 Bit ECC can provide comparable security to 1024 Bit RSA[7]. Use of smaller key size results in achieving faster computation efficiency, saving of bandwidth, memory and energy. These features in ECC make it a front-end choice for WSN security.

1.2 Elliptical curve digital signature algorithm

The elliptic curve digital signature algorithm is the elliptic curve analogue of DSA and serves the same purposes of key generation, signature generation, and signature verification[5,6]. As given in Wang[8], ECDSA comprises of 2 parts

- Signature Generation
- Signature Verification

The Elliptic Curve parameters can be described as: $T = (FR, E, P, n, h)$, where FR is the field representation, E is an elliptic curve, P is the generator point, n is the order of P a large prime number and h is cofactor. The system uses a hash function H. The user chooses a private key d, and calculates its corresponding public key as $P_d = P_* d$. The signature generation and verification is shown in Table 1 and Table 2.

Table 3. Notations used.

Symbol	Notation
$h()$	One way-hash function
p	A prime number
F_p	A finite field
E	Elliptic curve
G	Generator point of order n
ID_i	Node identity of i^{th} node
ID_j	Node identity of j^{th} node
BS	Base station
S_b	Random Number Chosen by BS, to act as private key
P_b	Public Key of BS, $P_k = S_b.G$
r_i	Private key Chosen by i^{th} Node
P_i	Public Key of i^{th} Node
r_j	Private key Chosen by j^{th} Node
P_j	Public Key of j^{th} Node
s_i	Signature of i^{th} node
s_j	Signature of j^{th} node
R_1	Random number chosen by i^{th} node
R_2	Random number chosen by j^{th} node

2. Proposed Node Authentication Protocol

The protocol design comprises of following phases: Initialization phase, Signature generation phase, Signature verification phase, Mutual authentication phase. The notations used in this paper are given in Table 3.

2.1 Initialization phase

Base station functions like a key generation and distribution center and sets up the system parameters as following:

The parameter group of system is $D = (F_p, E, G, n, h, H, S_b, P_s)$.

2.2 Signature generation phase

It is a pre-deployment phase in which base Station selects a random number s_b between 1 to $n - 1$ as its private key and computes its public key $P_b = G.s_b$. For each node i, the base station generates a random number r such that $1 < r < n - 1$. It computes public key for each node $P_i = G.r_i$. Base station computes a signature for each node using signature function $s_i(r_i, Id, s_b)$.

Signature of each node

$$s_i = r_i^{-1}[r_i.h(Id) + s_b] \tag{2}$$

Each node is preloaded with its specific signature $s_{i.}$, public certificate of base station s_b and public certificate of each node P_i is made public to the network.

2.3 Signature verification phase

This is a selfcheck by the node to ensure that the signature s_i pertains to it and has been correctly generated and thereafter loaded into it. The node uses its node identification, public certificate, and the public certificate of the base

Table 4. Mutual authentication phase.

Step	Node i	Node j
1.	Select a random number R_1 Compute GR_1	Select a random number R_2 Compute and send $G.R_2$ to Node i
2.	Compute: $s_i(GR_1 + r_iG), s_iG, GR_2$ Send it to node j	
3.		Verify: $s_i(GR_1 + r_iG) = (s_i.G).R_1 + s_i(r_i.G)$ If LHS=RHS then node j authenticates node i
4.		Compute: $s_j(GR_2 + r_jG), s_jG$ Send it to node i
5.	Verify: $s_j(GR_2 + r_jG) = (s_j.G).R_2 + s_j(r_j.G)$ If LHS=RHS then node i authenticates node j	

station to verify the authenticity of its signature.

$$s_i = r_i^{-1}[r_i.h(Id) + s_b] \tag{3}$$

$$r_i = s_i^{-1}[r_i.h(Id) + s_b] \text{ : Multiplying each side by } G$$

$$G.r_i = G.s_i^{-1}[r_i.h(Id) + s_b] = s_i^{-1}.Gr_i.h(Id) + s_i^{-1}.G.s_b \tag{4}$$

$$G.r_i = s_i^{-1}.h(Id).\text{public certificate of node } i + s_i^{-1}.\text{public certificate of base station}$$

Node i verifies its signature value si as given at (1) by computing the RHS value of equation (2) and comparing it with LHS. If the equation (2) is satisfied then the signature si of node i is verified. Signature verification is undertaken by the other nodes in a similar manner.

2.4 Mutual authentication

Let node i and node j have to authenticate each other. Their mutual authentication can be achieved by using their respective signatures, their public certificates and the public certificate of the base station. Let their signatures be S_i and S_j as following:

$$s_i = r_i^{-1}[r_ih(Id) + s_b] \tag{5}$$

$$s_j = r_j^{-1}[r_jh(Id) + s_b] \tag{6}$$

The mutual authentication process flow between Node i and Node j are shown in Table 4.

3. Time and Energy Analysis

Initialization and signature generation phases are done at pre-deployment, thus, their contribution to energy consumption at post-deployment is not considered. For capturing computational time of different key operations like Point Addition, Scalar Multiplication, Inverse a fundamental setup was created using MicaZ[9] mote and a MIB520[9] Gateway. A nesC program has been written on TinyOS[10, 11] platform to compute on the time of various operation on MicaZ mote.The program has been enabled with TinyECC[12] library. The computational time recorded for different operations is shown in Table 5. Computational time has been recorded over 3 standard SECG curves: Secp 192r1, Secp 160r1, Secp 128r1 to analyze the effect of key size on energy consumption.

Table 5. Computational Time Recorded.

	Time Taken (Seconds)		
Operation	Secp192r1	Secp160r1	Secp128r1
Scalar Multiplication	6.07	3.90	2.23
Point Addition	0.21	0.14	0.098
Inverse Operation	0.16	0.12	0.069

Table 6. Critical Operation in 4 phases of Mutual Entity Authentication Protocol.

S. No.	Phase	Scalar Multiplication	Point Addition	Inverse Multiplication
1	Initialization	–	–	–
2	Signature Generation	–	–	–
3	Signature Verification	2	1	1
4	Mutual Authentication	4	2	–

Table 7. Energy and Time Analysis of Mutual Entity Authentication Protocol.

		Secp192r1		Secp160r1		Secp128r1	
S. No.	Phase	Time Taken (Secs)	Energy Consumed (mJ)	Time Taken (Secs)	Energy Consumed (mJ)	Time Taken (Secs)	Energy Consumed (mJ)
1	Initialization	–	–	–	–	–	–
2	Signature Generation	–	–	–	–	–	–
3	Signature Verification	12.51	739.34	8.06	476.34	4.62	273.04
4	Mutual Authentication	24.78	1464.49	15.88	938.50	10.88	643.00
Total		37.29	2203.83	23.94	1414.84	15.5	916.04

The number of critical operations in various phases of the authentication framework is tabulated in Table 6. Energy is calculated by using $E = V * I * T$ (joules) is used, where V and I stand for voltage and current drawn respectively and T is the execution time for each operation. The calculations have been done targeting MicaZ node powered by 02 AA batteries with 3 Volts and 19.7 mA load current. The energy and time calculations are shown in Table 7.

4. Conclusions

Based on the concept of ECC and ECDSA, a mutual authentication protocol comprising of 3 phases has been designed and proposed. An experimental setup was created to compute the time taken by various significant operations. Time and energy analysis of the proposed protocol was carried on Micaz motes. As key Size plays an important role in the level of security, time and energy analysis was carried out on 192, 160 and 128-bit curves. The protocol finds its feasibility in smart city applications where low power sensor nodes are used.

References

[1] A. H. Moon, N. A. Shah, Iqbal Ummer and Ayub Adil, Simulating and Analyzing Security Attacks in WSN Using Qualnet, *IEEE Conference on ICMIRA*, pp. 68–76, (2013).

[2] A. Khalique, K. Singh and S. Sood, Implementation of Elliptic Curve Digital Signature Algorithm, *Int. J. Comput. Appl.*, vol. 2, no. 2, pp. 21–27, (2010).

[3] V. S. Miller, Use of Elliptic Curves in Cryptography, *Advances in Cryptology CRYPTO85 Proceedings*, Springer, pp. 417–426, (1986).

[4] N. Koblitz, Elliptic Curve Cryptosystems, *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, (1987).

[5] B. Menzes, Network Security and Cryptography, Cengage Learning.

[6] D. Hankerson, Guide to Elliptic Curve Cryptography, Springer.

[7] A. H. Moon and Khan Ummer, Authentication Protocols for WSN using ECC and Hidden Generator, *International Journal of Computer Applications*, (0975–8887), vol. 133, no. 13, (2016).

[8] W. H. Wang, Y. L. Cui and T. M. Chen, Design and Implementation of an ECDSA-Based Identity Authentication Protocol on WSN, *3rd IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications*, (2009).
[9] Mote Works, Getting Started Guide, March (2013). PN: 7430-0102-02.
[10] TinyOS. http://www.tinyos.net.
[11] P. Levis and D. Gay, TinyOS Programming, Cambridge University Press (2009).
[12] Liu, P. Ning, *et al.*, Tiny ECC: A Configurable Library for Elliptical Curve Cryptography in Wireless Sensor Networks, IPSN, (2008).