Amirkabir University of Technology
(Tehran Polytechnic)
Vol. 46 - No. 1 - Spring 2014, pp. 31- 44

Amirkabir International Journal of Science & Research
(Modeling, Identification, Simulation & Control)
(AIJ-MISC)

An ECC-Based Mutual Authentication Scheme with One Time Signature (OTS) in Advanced Metering Infrastructure

F. Naji Mohades[1] and M. H. Yaghmaee Moghadam[2*]

1- M.Sc. Student, Department of Computer Engineering, Imam Reza International University, Mashhad, Iran
2- Professor, Department of Computer Engineering and Center of Excellence on Soft Computing and Intelligent Information Processing, Ferdowsi University of Mashhad, Mashhad, Iran

ABSTRACT

Advanced metering infrastructure (AMI) is a key part of the smart grid; thus, one of the most important concerns is to offer a secure mutual authentication. This study focuses on communication between a smart meter and a server on the utility side. Hence, a mutual authentication mechanism in AMI is presented based on the elliptic curve cryptography (ECC) and one time signature (OTS) consists of two phases: a key and signature generation phase as well as a signature verification phase. The next challenge, is securing communication messages. Accordingly, a message authentication mechanism based on ECC and OTS is proposed in this paper. Such protocols are designed based on resource constraint problem on the consumer side and security requirement satisfaction in AMI. Security of the protocol with BAN logic is proved and possibility of signature forgery via the mathematical principle of birthday paradox formula is represented. In the end, security of the protocol is scrutinized with informal methods and is simulated on Java. Simulation and analytical results show that proposed protocols are more secure and efficient than similar methods against most of the security attacks.

KEYWORDS

Advanced Metering Infrastructure, Elliptic curve, Mutual authentication, One time signature, Smart grid, Key management.

*Corresponding Author, Email: corresponding.auther@aaa.com

1. INTRODUCTION

Smart grid (SG) [1] is a modernized electrical grid created to solve the problems of traditional electrical power industry [2]. Smart grid uses a bidirectional flow of electricity and information among suppliers and consumers; therefore, consumers may monitor and control their consumption, save excessive energy and return it to Power Company [3]. Additionally, utility side utilizes two-way flow for reliable, efficient and transparent metering, collection, monitoring and control of electricity consumption [4],[5]. As the smart grid grows, the more security concerns related to its communication increase. The advanced metering infrastructure (AMI) [6] is related to the systems that measure, collect, save, and analyze electricity consumption data taken from smart meters in grid. This structure includes software, hardware, a communication network, customer-associated systems, a meter data management system, and an AMI control server [4]. AMI is a key part of the smart grid and the security of its communication must be provided to cope with concerned cyber security challenges [3],[10]. This study focuses on the required communication between a smart meter entity and a server on the utility side, named control center, as shown in Figure 1. Power companies must generate a large number of smart meters, as well as economic reasons, necessitating that smart meters are almost low cost which causes the smart meter to be a device with low computational capabilities.

An important issue in the exchange between a smart meter and control center server is message authentication [7]. Using message authentication, a receiver can be verified if the received message comes from the actual sender and has no forgery during the transmission. To ensure request sending from a real sender, messages must be authenticated. Without authenticated messages, an attacker can modify the message, forge a new one, or replay an old message with malicious purposes.

AMI has constraints in communication bandwidth, computation time, and computational resources of smart meters [8]. Traditional public-key infrastructure based digital signature schemes cannot prepare security in the AMI because of: 1) increased communication load (large key sizes which increase communication bandwidth), 2) increased time for authentication, signing, decryption/ verification messages (which increase latency) and 3) the limited computational resources of smart meters.

For AMI communications, the most recent authentication protocol named Tunable Signing and Verification (TSV) proposed by Qinghua Li et al [9] reduces the signature size at the price of increased computations that made this inappropriate for the operations of the smart meters side. Therefore, the need for better OTS schemes that will require fewer resources, with low signature and computations size is important in the AMI. To solve these constraints limit, in this study, we use one-time signature scheme (OTS) as well as ECC for message and mutual authentication. Hence, with taking advantage of elliptic curve discrete logarithm problem (ECDLP), our mechanism achieved a significant reduction in signing overhead, pre-computation cost, and storage overhead.

In this paper, the computational capabilities of smart meters and security requirements of AMI are focused together in both our ECC-based mutual and message authentication algorithms, also we explore here, for the first time, the use of one-time signatures as well as elliptic curve cryptography for mutual authentication in the smart grid. Due to this innovation a significant reduction in encryption computation has occurred.

To address a new security algorithm for smart grid communications, the unique characteristics of AMI security can be summarized as follows:

1) *Use the low computational resource and memory storage:* results in Table 3 show low resources usage of our algorithms.

2) *Recommend a fast verification algorithm:* A smart grid system that connects a large number of smart meters to servers, the number of meter reading messages to be verified might slow down the process.

3) *Data confidentiality and message integrity:* Because messages contain sensitive information about charges or customer and utility company privacy, it is necessary to establish a secure communication channel for preserving confidentiality and integrity; the utilization of ECDLP, random numbers and hash functions provide confidentiality and integrity.

4) *Prevention of potential cyber-attacks*: If a smart meter gets malicious, adversary should not influence the compromised meter to access information on other meters or pass into the AMI. The utilization of pre-shared secret pw, between the server and the meter prevents this problem to occur.

5) *Prevention of using secure channel and smart card*: In many of mutual authentication mechanisms, they use secure channel or smart card for communication; but, it is unreasonable in smart grid implementation.

6) *Appropriate keys and signature size*: Unlike other OTS methods, using ECC in our mechanism provides short keys and signature size.

A. Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) was proposed in 1985 by Victor Miller (IBM) [11], and Neil Koblitz [11] (University of Washington) as an alternative mechanism for implementing public-key cryptography. ECC provides the same security level as other public key cryptography algorithms like RSA, ElGamal or Diffie-Hellman via shorter keys. This makes ECC a suitable option for implementing security services in constrained devices like the smart meters.

Commonly, ECC is exhibited as an Elliptic curves points (x, y) on Z_P by means of the following equation.

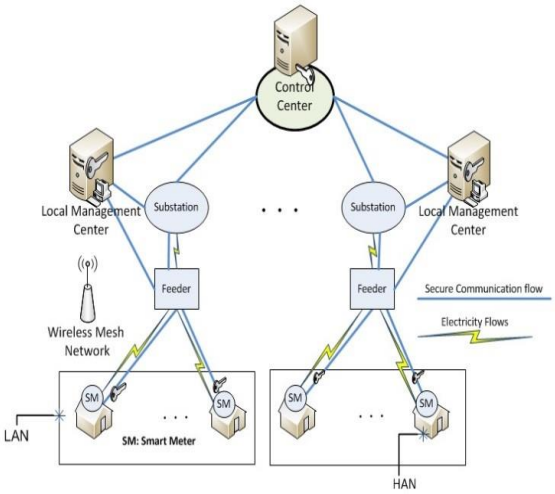

Fig. 1. AMI communication architecture

$$\begin{aligned} &Y^2 \equiv X^3 + AX + B \bmod P \\ &(X, Y) \in Z_P, P > 3 \\ &A, B \in Z_P \\ &A^3 + 27B^2 \not\equiv 0 \bmod P \end{aligned} \tag{1}$$

B. One Time Signature

OTS was suggested separately by Lamport [13]and by Rabin [14], and then was improved through a number of projects [15], [16], [17] and [18]. In OTS, there are two phases, including signing and verification.

i. Key Generation And Signing

The signer first generates t strings as SK or a private key set and then applies a one way function F on SK to generate PK or a public key set. After that he/she picks message m and execute $h = H(m)$. At last, he/she somehow (related to OTS method usage) obtains signature $\{S_1, S_2 \dots S_i\}$ from SK utilizing h. Finally, sends signature, message, and PK for verification.

ii. Verification

The verifier applies hash function H on m: $h = H(m)$ to obtain a signature $\{S_1, S_2 \dots S_i\}$ using h. Now he/she should verify the signature, so he/she applies one way function F on signature and checks it with the PK set to verify the message m.

OTS is conceptually similar to PKC-based signatures since the signer employs a private key to sign a message and the verifier takes the signer's public key for signature verification [9]. Still OTS is more efficient of PKI-based signature computation and verification, because of the use of a one-way function without trapdoor for signing and signature verification; but it suffers from Man In The Middle (MITM) attack and large signature size.

C. One Way Fuction

A function f from a set X to a set Y is called a *one-way function* if $f(x)$ is "easy" to compute for all $x \in X$ but for "essentially all" elements $y \in Im(f)$ it is "computationally infeasible" to find any $x \in X$ such that $f(x) = y$ [19].

D. Birthday Paradox

The birthday problem asks how many people must be in a room to have at least 50% chance in which two persons are born in the same day of the year (assuming birthdays are distributed evenly) or how many balls must be thrown into bins in order to expect at least one birthday collision happens. Suppose x_{ij} as an indicator of a random variable to count the number of collisions in the birthday for the person j is on the day i or the ball j going into the bin i, and m as people or balls, n as days or bins, then we have:

$$\begin{aligned} &\mathbf{E(X) = \sum_{i \neq j} Pr[X_{ij} = 1] = \frac{1}{n}\binom{m}{2}} \\ &E(X) = \frac{1}{365}\binom{m}{2} = 1 \text{ we get } m \geq 23 \end{aligned} \tag{2}$$

Contributions: In this paper, we present two protocols. The first one is an elliptic curve cryptography (ECC) based mutual authentication protocol with one-time signature (OTS) proposed for secure communication in the AMI control system. The second one is an ECC-based message authentication protocol with OTS developed for secure message exchange in AMI communication. In this paper, we extend and improve our previous work in [20]. The rest of the paper is organized as follows. In Section 2,

we introduce the related work. In Section 3 and 4, we present our schemas. In section 5.A, we analyze the security of our proposed signature with mathematic formula. In section 5.B, we show the BAN logic analysis for our protocol. In section 5.C, we informally analyze protocols. In section 5.D, we compare our scheme with others. At last, in 5.D.1 we implement algorithms.

2. Related Work

A. Biba

The Bins and Balls (BiBa) proposed by Perring [21] is a signature scheme with a low verification overhead and a small signature size. BiBa is one of the fastest signature schemas; however, its biggest disadvantage is the public key size. Kgwadi et al. in [22] looked into an authentication algorithm for the SG, and show that BiBa has a 75% smaller cost than elliptic curve digital signature algorithm (ECDSA). BiBa uses one-way function without trapdoors-based birthday paradox, and utilizes the findings of collisions in the hash value of SElf-Authenticating vaLues ($SEALs$). The property for $SEALs$ is that the verifier can efficiently authenticate the $SEAL$ based on the public key, and that it is computationally infeasible for an adversary to find a valid $SEAL$, given a public key. The security of this method is defined in a manner that an adversary has less $SEALs$ than verifier; therefore he/she has a low probability to forge a signature. We explain the BiBa schemas in a simplest way.

i. Key Generation And Signing

Signer generates t random string named $SEALs$, forms $h = H(m)$, pick G_h from Hash family function G, apply G_h on all $SEALs$ to find one collision where $G_h(S_i) = G_h(S_j)$, then S_i and S_j forms the signature. After that, he/she sends message m and signature $\{S_i, S_j\}$ to the verifier.

ii. Verification

The verifier first should verify the $SEALs$, hence checks $S_i \neq S_j$ and authenticates them efficiently like a Merkle tree; then, computes $h = H(m)$ and checks $G_h(S_i) = G_h(S_j)$.

B. Tsv

Qinghua Li et al. in [9] proposed a Tuable Signing and Verificaion (TSV) scheme. First, they generate t random l bit strings that are foundations of a private key. They use F a one way function without trapdoors, as $F^x(S)$ is the result of applying x times F over S. $F^0(S) = S$, $F^1(S) = F(S)$. $SPLIT(h)$ denotes a function that receives string h as input and splits h into r substring, every substring has $\log_2 t$ long, the any h_j is a mark of element i, for example, h_1 is a mark of i_1 and h_r is mark of i_r. Now we describe TSV in three steps.

i. Key Generation

After generating $\{S_1, S_2, \dots, S_i\}$, TSV for each S_i structs a chain of one way function f of length $w+1$, i.e., $S_i \rightarrow F(S_i) \rightarrow \cdots \rightarrow Fw(S_i)$. These t chains form private key SK. And public key is $PK = (V_1, V_2, \dots, V_t)$, where $V_i = F^{w+1}(S_i)$.

ii. Signing

To sign message m, firstly, signer performs $h = H(m \mid C)$, where C is counter with initial value 0 and H is a secure hash function. Next, he/she sends h to function $SPLIT(h)$ to obtain $i_j$set. All i_j should be different and the i_j in the same group should be sorted in a decreasing form. If else increment c and repeat the process. The signature of message m is: $(c, (F^{w-wq1}(S_{i1}), \dots, F^{w-wqr}(S_{ir})))$.

iii. Verification

As soon as the verifier receives signature $(c, (F^{w-wq1}(S_{i1}), \dots, F^{w-wqr}(S_{ir})))$ and messagem'; firstly, he/she performs $h = H(m' \mid C')$, then calls $SPLIT(h)$. Next, it checks the obtained i_j from $SPLIT(h)$ to be different and the i_j in the same group ordered in the decreasing form. Finally, it confirms correctness of $F^{w_{qj}+1}(S'_j) = v_{i_j}$ for each j.

iv. Brief Analysis Of TSV Protocol

The main and very important problem with this authentication method is vulnerability to man in the middle (MITM) attack. In this dangerous attack, the adversary is unable to generate its own t random strings, pick one way function F, and subsequently generate SK and PK set. After that he/she generates a fake message with "home customer power outage" content via server ID, and will sign this malicious message by the TSV signing method. Then, it sends the message, the sign, and own public key to a smart meter.

Smart meter receives the packet. Firstly, it confirms correctness of the signature via public key which is located inside the received packet, which obviously will be passed. Then, it should check the correctness of message m. Therefore, it fist computes h, maps h to i_j elements, and forms the signature. Finally, it compares the signature and received signature; clearly, this comparison will be passed as well. The receiver has no way to discover the received public key which is not from the real server. Sending such fake messages from the control center server will have disastrous consequences for the grid.

3. EMAOTS: ECC BASED MUTUAL AUTHENTICATION WITH OTS PROTOCOL

In this section, the EMAOTS protocol is presented. Let us consider mutual authentication between the smart meter and the control center utilizing a pre-shared password which we define PW. Furthermore, we assume that both parties have knowledge of the EC parameters set $\{a, b, P, h, n, G\}$ and hash function. Table 1 presents the list of parameters and their definitions used in our design.

A. Description Of EMAOTS Protocol

Shown in Figure 3, the EMAOTS protocol has the following phases:

a) SM: Smart meter is the initiator. SM generates a random string set $\{S_1, S_2, \dots, S_i\}$ named $SEALs$ from

1: **Input**: The mask of timestamp as t_{mask}, Q^x as the x coordinate of point Q. Family of hash functions $G = \{g_0, \dots, g_{n-1}\}$.
2: **Output**: A valid signature, $sig = \{S_i, S_j, S_k, C\}$
3: **Initialization**: $C := \emptyset$
4: do
5: Compute $h=H(Q^x \oplus t_{mask} \oplus C)$
6: Pick G_h from Hash Function Family G
7: Apply G_h on $SEALs = \{S_1, S_2 \dots S_t\}$
8: Seek for $G_h(S_i) = G_h(S_j) = G_h(S_k)$
9: if failed increment C and Go to 5
10: if succeed Return ***sig***

Fig. 2. The mutual authentication protocol

SELf Authenticating Values, then picks time stamp T_1 where T_1 denotes the current time, and applies the function H_1 on timestamp $t_{mask1} = H_1(T_1)$ (as H_1 maps current time to an integer). After that concatenates t_{mask1} and pre-shared password PW, then multiplies the result to the group generator P to obtain point Q as $Q = (PW \oplus t_{mask1}) \times P$. Next, it sends Q^x, and t_{mask1} to algorithm1, where Q^x is the x coordinate of point Q.

In line 5 of algorithm1, C is a counter; SM obtains h, then it uses h as indicator of family hash function G and picks G_h. Now he/she applies secure hash function G_h on $SEALs\{S_1, S_2, \dots, S_i\}$. Now we have t hashed $SEALs\{G_h(S_1), G_h(S_2), \dots, G_h(S_i)\}$.

The problem is finding a 3-way collision in this set. However, now G_h is a secure hash function in the random oracle model and it is collision resistant; hence, like BiBa method we should limit the result of hash function G_h to n element as shown in Table 1. Taking advantage of this technique, we still have the same security as before; additionally, we could find a collision without losing hash function security. Next, the algorithm in line 8 seeks for a three-collision, if found collision we add them to the

TABLE 1. NOTATION LIST

Notation	Description
|	*The string concatenation operation*
×	*An elliptic curve scalar multiplication*
$H()$	*Secure hash functions in the random oracle model*
G_P	*Cyclic group of prime order n of P*
P	*Large prime generator of group*
n	*Order of elliptic curve*
$H_1()$	*Secure one-way hash function H1: {0,1 → Z_P^*}*
T	*The time stamp*
PW	*Pre shared key between SM and server*
G	*Hash function family in the random oracle model*
G_h	*{0,1}^{m2}→[0,n-1] is an instance in the Hash function family G selected via indicator h*
Z_p	***Finite field of order p with integer***

Figure 2 illustrates generation of the signature in an example. Finally, after we find a valid signature Sig, SM sends the first packet in mutual authentication protocol to the *server*.

signature set and process is done; however, if else, increment counter C is chosen and we go to line 5.

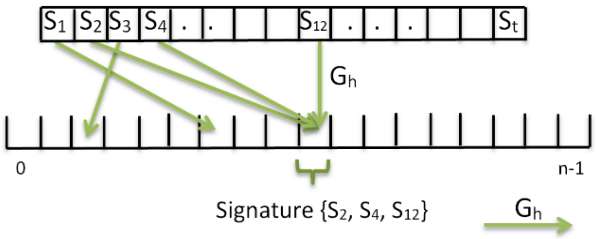

Fig. 3. Finding valid signature

b) Control Center Server: Upon receiving first packet from smart meter, the $server$ uses ID_{SM} to evoke pre-shared PW between itself and SM from the database, and checks the freshness of T_1. The freshness of T_1 is confirmed by performing T_1'-$T_1 \leq \Delta T$, where T_1' is the current time and ΔT is a valid time interval. If T_1 is not fresh, server aborts the current session. Then it uses T_1, PW and P to compute $Q = (PW \oplus t_{mask1}) \times P$, and $h =$

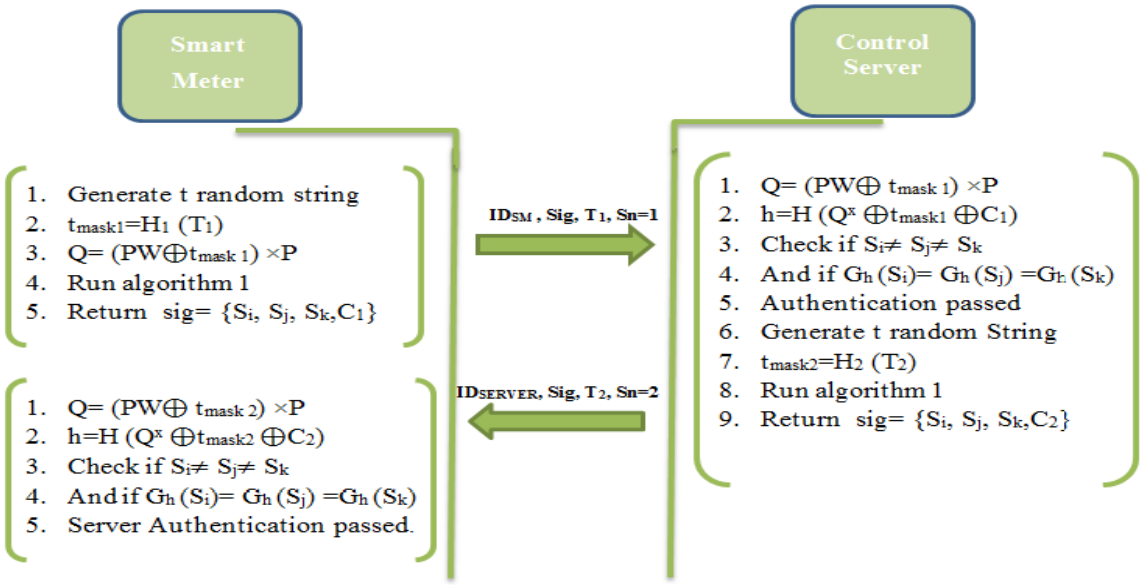

Fig. 4. Mutual Authentication Protocol

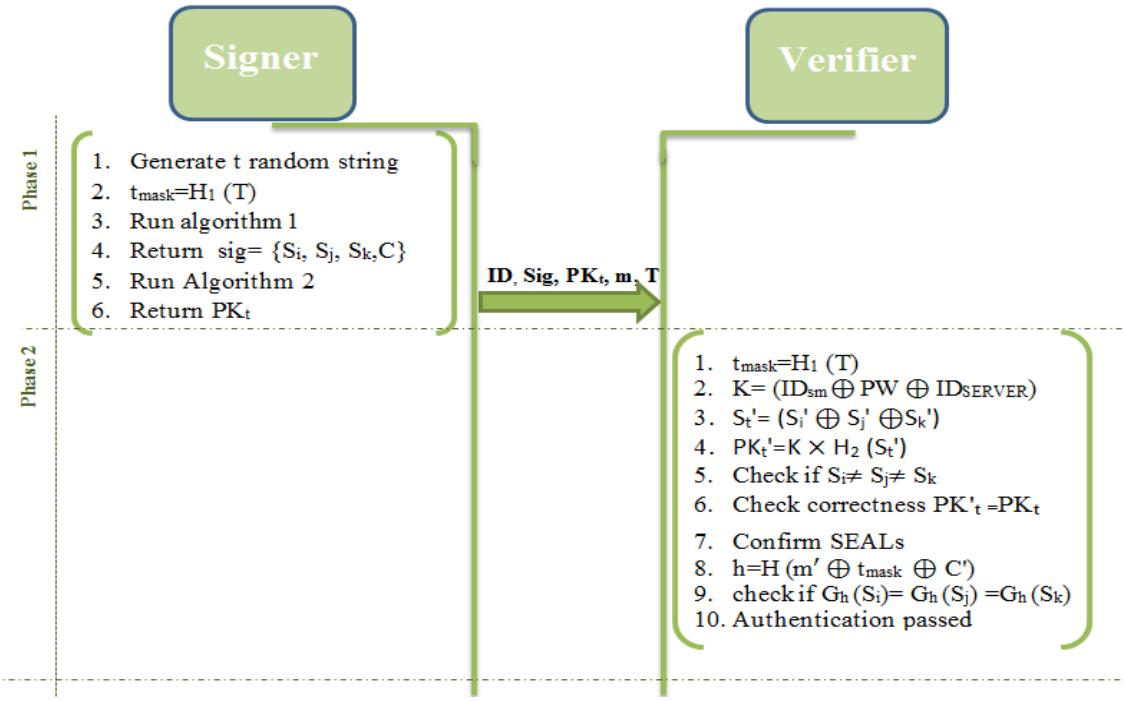

Fig. 5. ECC-Based Message Authentication Protocol.

$H(Q^X \oplus t_{mask1} \oplus C_1)$, the *server* picks G_h and applies that on the signature set. Then, the *server* checks if $S_i \neq S_j \neq S_k$, and next it confirms whether $G_h(S_i)$, $G_h(S_j)$, $G_h(S_k)$ are equal together or not. If they are equal the signature is valid and smart meter's identity is verified in the *server*. If not, the authentication will fail.

Phase 2: Server → Smart meter: {ID_{SERVER}, Sig, T_2, Sn=2}

*a) **Control Center Server:** In this phase because of thesd mutual authentication process, to avoid server*

masquerading attack, the $server$ should prove his/her identity to the smart meter. Thereupon, the same as phase 1, it generates t random string named $SEALs$, then picks T_2 where it indicates the current time for computing $t_{mask2} = H_1(T_2)$.Then, it calculates $Q = (PW \oplus t_{mask2}) \times P$, where P is group generator. At last, it sends $Q^x$and t_{mask2} to the algorithm1, where Q^X is the x coordinate of point Q. Finally, after finding a valid signature Sig, the control center server sends a second packet in mutual authentication protocol to SM.

After the two-phase process between the SM and the $server$, they are authenticated to each other and can have secure communication, but still we need an appropriate message authentication method for message integrity assurance.

4. ECC-Based Message Authentication With Protocol

The message authentication protocol is driven by any smart meter that is to send a message to control the server or a control server that is to send a message to a smart meter.

A. Description Of Message Authentication Protocol

Shown in Figure 4, the EMAOTS protocol has the following phases:

Phase1: Signer→Verifier: {ID, m, T, sig, PK_t}

Firstly, the signer generates t random l-bit string named SEALs: {S_1,S_2,…,S_i}. Next, it picks time stamp T where T denotes the current time, and applies the hash function H_1 on the timestamp $t_{mask} = H_1(T)$ (as H_1maps current time to an integer). Then, the signer sends t_{mask} and message m to algorithm 2 to find a valid signature.

1. **Input:** The mask of timestamp as t_{mask}, message m, and Family of hash function G=*{g_0. …, g_{n-1}}*.
2. **Output:** A valid signature, *sig={S_i. S_j. S_k. C}*
3. Initialization: $C := \emptyset$
4. do
5. Compute $h=H(m \oplus t_{mask} \oplus C)$
6. Pick G_h from Hash Function Family G
7. Apply G_h on $SEALs$= {S_1, S_2… S_t}
8. Seek for $G_h(S_i) = G_h(S_j) = G_h(S_k)$
9. if failed increment C and Go to 5
10. if succeed Return ***sig***

Fig. 6. Finding valid signature for message m

In Line 5 of algorithm 2, singer concatenates message m, mask of timestamp, and the counter C, then it uses h as pointer to family of hash function G and picks G_h. It performs hash function G_h on all SEALs set. Now, we have t hashed $SEALs$ {$G_h(S_1)$, $G_h(S_2)$, …, $G_h(S_i)$}. Afterwards, the signer is sought for three-way collision within t hashed $SELAs$. If he/she finds the collisions, adds them to signature set and process is done. Otherwise, it increments counter C and go to line 5. After finding the signature, the signer utilizes algorithm 3 to obtain a temporary public key.

Algorithm 3 takes t_{mask}, $SEALs$ from the signature set and the concatenation of pre-shared secret value PW, and identify both parties(SM and $Server$). In line 3, algorithm concatenates three $SEALs$ within signature to obtain S_{Total} parameter. For the sake of avoiding password guessing attack, masquerade attack and mainly MITM attack, we exploit hash function H_2 so that it maps the string to a point on Elliptic curve(EC) as $H_2(S_{Total})$ where the result is an EC point. Next, in line 3, the signer calculates the temporary public key, where " × " denotes an EC multiplication. As a result of using the EC multiplication, we take benefit of Elliptic Curve Discrete Logarithm Problem (ECDLP) so indicating K is computationally infeasible. Finally the signer packets the message, the signature, the timestamp, the temporary public key, and the signer's ID. Then he/she sends it to the verifier.

Phase2: Verification

Upon receiving first packet from signer, firstly, the verifier checks the freshness of T from the packet, if ΔT is a logical time interval, then he/she composes t_{mask} as $t_{mask} = H_1(T)$. Next, it calculates $K = (ID_{SM} \oplus PW \oplus ID_{SERVER})$. Then, it takes $SEALs$ from signature and performs $S'_t = (S'_i \oplus S'_j \oplus S'_k)$. Furthermore, it uses K and S_t' to compute $PK'_t = K \times H_2(S'_t)$ and then checks if $S_i \neq S_j \neq S_k$; next, it checks correctness of $PK'_t = PK_t$. If they were equal, verifier could confirm $SEALs$ from the signature. In addition, it calculates $h = H(m' \oplus t_{mask} \oplus C')$, and checks if $G_h(S_i) = G_h(S_j) = G_h(S_k)$, then the verifier can verifies the validity of massage m. Otherwise, the verifier aborts the verification process.

1. Input: Take mask of timestamp as t_{mask}, SEALs And K= ($ID_{sm} \oplus PW \oplus ID_{Server}$)
2. Output: Temporary Public Key PK_t
3. $S_{Total}=(S_i \oplus S_j \oplus S_k)$
4. $PK_t=K \times H_2(S_{Total})$
5. Return PK_t

Fig. 7. Finding Temporary public key

5. ANALYSIS

In this section, we analyze our protocols in several ways. Firstly, we examine the probability of finding and forgery of the signature. Then, we analyze, security of mutual authentication protocol formally. After that, we informally inspect the influence of different attacks on the proposed protocol. Finally, to study our protocol performance, we compress them with several similar methods; furthermore, we simulate our mechanism on Java, and present results of simulation in section D.

A. Security Of Signature

In this section, we describe the Occupancy Problems for Bins and Balls algorithm and illustrate our protocol security per formula. Occupancy problems deal with pairings of objects. The basic occupancy problem is about placing t balls into n bins. Suppose X_i is the random variable which counts the number of balls in the bin i (so X_i is not an indicator). Clearly:

$$\sum_{i=1}^{n} X_i = t \tag{3}$$

where X_i has the Binomial distribution [16]. To find this out, suppose X_{ij} is the indicator of random variable for ball j going into the bin i, so that $X_i = \sum X_{ij}$ and X_{ij} $\begin{cases} 1 \text{ if ball j goes into bin i} \\ 0 \quad \text{otherwise} \end{cases}$

Then each X_{ij} represents a Bernoulli trial with probability $p = 1/n$, which is the probability of ball j going into bin i. Since X_i is a sum of Bernoulli trials, it has the binomial distribution.

Specifically, for the probability of a particular bin having exactly k balls, it has a distribution of the form:

$$Pr[X_i = k] = \binom{t}{k} P^k (1-P)^{t-k}$$
$$\binom{t}{k} (\frac{1}{n})^k \left(1 - \frac{1}{n}\right)^{t-k} \tag{4}$$

However here we need the probability of a particular bin having at least k balls, if we look at any subset of balls of size k; then, the probability that the subset of balls falls into the bin i is:

$$\left(\frac{1}{n}\right)^k \tag{5}$$

Note that we no longer have the $\left(1 - \frac{1}{n}\right)^{t-k}$ factor because we do not care about where the rest of the balls fall. We then take a union bound of these probabilities over all $\binom{t}{k}$ subsets of size k. The events we are summing over, though; they are not disjoint.

Therefore, we can only show that the probability of a bin having at least k balls is at most

$$\binom{t}{k} (\frac{1}{n})^k \tag{6}$$

Now, we show this formula with an example. Assume $n = 1000$ as bins or the result of hash family function G_h range [1, n-1], $t = 1024$ as balls or $SEALs$ and $k = 2$ as a two-way collision. Therefore the probability of finding at least one two-way collision is equal to: $Pr[k]=\binom{1024}{2}(\frac{1}{1000})^2 = 0.523776$ or 52%.

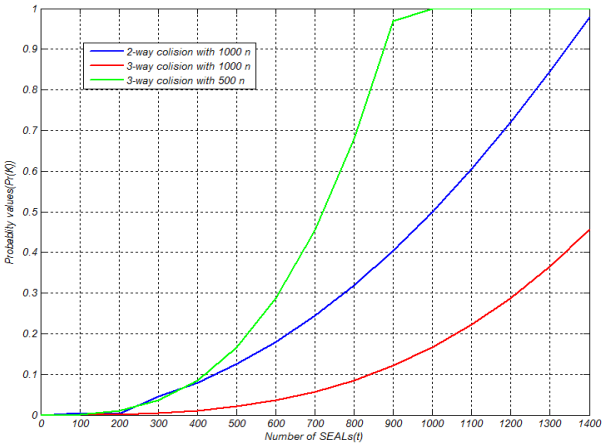

Fig. 8. Probability of finding signature

It is 52% chance to find a signature on first examination. We then assume 10 $SEALs$ have been revealed. So $Pr[forge]=\binom{10}{2}(\frac{1}{1000})^2 = 0.000045$ or 0%, it means an attacker has 0% chance to forge a signature. Figure 5 represent probability of finding signature with different $SEALs$. In blue line when we have 1000 $SEALs$ and 1000 n, the probability of finding a signature with a two-way collision at first try is 50% in average. In red line when we have 1400 $SEALs$ and 1000 n, the probability of finding a signature with a three-way collision in first try is 45% in average; therefore, we should decrease the number of n and $SEALs$ because with three-way collision we have more security and we can use less n and $SEALs$. So in green line, we have a three-way collision, 500 n and 700 $SEALs$ with probability 50% in first try in average. As a consequence, in performance analysis, we assume $t =$ 700.

B. Format Security Analysis

BAN logic [23] is a popular method for the analysis of mutual authentication protocol [26][24], which offers a formal method for reasoning about beliefs of security protocol participants. In this section, we prove the security of mutual authentication protocol via this logic.

i. BAN Logic Notation

BAN logical notation used in this paper as follows:

1)**P|~ *X***: P sends a message consists of X, and believes it at the sending time.

2)**#(X):** X is fresh, and X has not been sent any time before the current run of the protocol. Usually, X is a nonce.

3)**P$\overset{K}{\leftrightarrow}$ Q:** (Read 'k is a good key for P and Q'.) k will never be discovered by any principal but P, Q, or a principal trusted by P or Q. (The last case is necessary, since the server often sees, indeed generates k.)

4)**P$\overset{X}{\leftrightarrow}$ Q**: X is a shared secret known only to P and Q.

5)$\{X\}_K$**:** Short for "$\{X\}k$ from P" (Read 'X encrypted with k (from P)'.) This is the notation for encryption. Principals can recognize their own messages. Encrypted messages are uniquely readable and verifiable as such by holders of the right keys.

6)$\langle X\rangle_Y$**:** X combined with Y so that Y's presence proves the identity of whoever uttered X.

7)$\{X\}_K$**:** Short for "$\{X\}k$ from P" (Read 'X encrypted with k (from P)'.) This is the notation for encryption. Principals can recognize their own messages. Encrypted messages are uniquely readable and verifiable as such by holders of the right keys.

8)$\langle X\rangle_Y$**:** X combined with Y so that Y's presence proves the identity of whoever uttered X.

9)**P |⇒ X :** P has jurisdiction over X. The principal P is an authority on X and should be trusted on this matter.

ii. Original Protocol

From Section 3 and Figure 3, the message sequence in Mutual Authentication phases are as follows:

Message 1: Smart Meter→Server: ID_{SM}, Sig1, SN (we know signature containing $\{S_i, S_j, S_k, C_1, T_1\}$), we extracted message 1 as:

SM→ SERVER: ID_{SM} ، $[S_i, S_j, S_k]_{PW}$ ، T_1 ،C ، SN

Message 2: Server→Smart Meter: ID_{SERVER} ، $Sig2$ ، SN. Similar to previous Messages:

SERVER→ SM: ID_{SERVER} ، $[Si, Sj, Sk]PW$ ، $T2$، C ، SN

In messages 1, SM sends ID_{SM}, $Sig1, T1, C,$ and SN to $server$ indicative required parameter for SM authentication. The signature contains three sequence random strings, where these are nonce. In addition, the $server$ sends its own required parameter for authentication to SM in the second message according to their shared secrets, i.e. PW.

iii. Idealized Protocol

To exhibit our protocol in standard form, we use A and B, instead of SM, and $SERVER$. T_1 used with SM on packet representation via N_a. N_a is nonce, and in BAN logic notation, $Nonces$ commonly include a timestamp or a number that is used only once such as a sequence number. $Sig1$ in message 1 is a special string that SM uses to prove its identity and encrypted via PW between $SERVER$ and SM. $SEALs$ within signature produce in each session; therefore, $SEALs$ are temporary values and we express these with N'_a. Unlike timestamp, $SEALs$ have not been produced to show messages' freshness, but these are the basis of signature. According to notation 8 in section B.1, because of standardization of messages,

suppose $N'_a$is encrypted via PW and algebraic properties of our cryptosystem are not considered. We use K_{ab} to present PW. The clear text ID, the counter C, and SN are omitted as they do not contribute to the logical properties of our protocol. The idealized protocol is defined as:

M1: A→B: $N_a, \langle N'_a\rangle_{K_{ab}}$;

M2: B →A: $N_b, \langle N'_b\rangle_{K_{ab}}$;

iv. Security Goals

The protocol is purposed to provide a means for a smart meter A and a server B to prove their identity to each other using produced signatures. That is:

G_1: B |≡ A |~ Sig1;

G_2: A |≡ B |≡ Sig1;

G_3: A |≡ B |~ Sig2;

G_4: B |≡ A |≡ Sig2;

G_1 means the B belief that A sent signature $Sig1$ and believes in it; thus, A believes that B believes $Sig1$ (G_2). Consequently, identity of A proves to B. Similarity, G_3 means A believes that B sent signature $Sig2$ and believes in it; therefore, B believes that A believes $Sig2$ (G_4). Consequently, identity of B will prove to A.

v. Intitiative Assumptions

To analyze the mutual authentication protocol, we give the following assumptions:

A_1: A |≡ A$\overset{K_{ab}}{\longleftrightarrow}$B A_2: B |≡ A$\overset{K_{ab}}{\longleftrightarrow}$B

A_3: A $|\equiv$ B $|\Rightarrow N'_b$ A_4: B $|\equiv$ A $|\Rightarrow N'_a$

A_5: A $|\equiv \#(N_a)$ A_6: B $|\equiv \#(N_b)$

A_7: A $|\equiv \#(N'_a)$ A_8: B $|\equiv \#(N'_b)$

N'_a is fresh and B believes that A sent a message containing N'_a and believes in it, then B believes N'_a is fresh. Now with this new result, G_3, and applying the nonce verification rule, we have: $\frac{B \mid\equiv \#(N'_a), B \mid\equiv A \mid\sim N'_a}{B \mid\equiv A \mid\equiv N'_a}$.

TABLE 2. THE COMPUTATIONAL COST COMPARISON

Methods	Key Generation Cost	Signing Cost	Verification Cost	PK Size	Signature Size
BiBa	t + t H	2t	2r + 1	tL	rL
HORSE	t + t H	1H + 1	r + 1 H	tL	rL
HSLV	t + t H	1 H + r!μ	(r + 1)H + r	tL	rL+log (r!μ)
LSHV	t+(t)(r-1) H + 2t H	(r+1) H	$\frac{r(r+1)}{2}+1$	tL	rL+log μ
TSV	t+(t)(w+1)H + (t+1) H	(t+1) H	tL	rL	rL + log($\mu \coprod_{r=1}^{g} n_r$!)
ECDSA	1 PM	1 PM + 2 mod + 1 Minv + 1H	2PM + 1PA + 2mod + 2Minv + 1H	2	2L
EMAOTS	t+1H + 1 PM	(t+1) H + t + log t	(K+2)H + 1 PM	1	3L
Proposed Message Authentication	t+2H + 1PM	(t+1) H + t + tlogt	(K+3) H + 1PM	1 point on EC	3L

vi. Protocol Analysis

Using previous assumptions and required security goals analyze the protocol:

1)Message Meaning rule

A_2 gives B $|\equiv A \overset{K_{ab}}{\longleftrightarrow} B$ and M_1 means that $B \lhd \langle N'_a \rangle_{K_{ab}}$. We apply them to the message-meaning rule for shared secrets [14-15] that $\frac{P \mid\equiv P \overset{Y}{\leftrightarrow} Q \ , \ P \lhd \langle X \rangle_Y}{P \mid\equiv Q \mid\sim X}$, and replacing A_2 and M_1 on the rule we have: $\frac{B \mid\equiv B \overset{K_{ab}}{\longleftrightarrow} A \ , \ B \lhd \langle N'_a \rangle_{K_{ab}}}{B \mid\equiv A \mid\sim N'_a}$ which gives us B $\mid\equiv$ A $\mid\sim$ N'_a predicate, then with replacing $Sig1$ on N'_a that is G_1. This predicate says: if entity B believes that K_{ab} is a good key for communication between A and B, and B receives a message encrypted with K_{ab}, then B believes that A sent a message containing $N'_a$and believes in it. A_1 gives that B $|\equiv A \overset{K_{ab}}{\longleftrightarrow} B$ and M_2 means that $A \lhd \langle N'_b \rangle_{K_{ab}}$. According to message meaning rule and previous predicate we have: A $\mid\equiv$ B $\mid\sim N'_b$, Then, with replacing $Sig2$ on N'_b that is **G_3**.

2)Nonce Verification rule

With A_7, proved G_3, and nonce verification rule, we have: $\frac{A \mid\equiv \#(N'_a), B \mid\equiv A \mid\sim N'_a}{B \mid\equiv \#(N'_a)}$ that says, if A believes that

3)Jurisdiction rule

Jurisdiction rule says $\frac{P \mid\equiv Q \mid\Rightarrow X \ , \ P \mid\equiv Q \mid\equiv X}{P \mid\equiv X}$, using A_4, and last the result in the previous section we have:

$\frac{B \mid\equiv A \mid\Rightarrow N'_a \ , B \mid\equiv A \mid\equiv N'_a}{B \mid\equiv N'_a}$, and since $\frac{A \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B \ , \ A \mid\equiv \{N'_a\}_{K_{ab}} \ , \ B \lhd \langle N'_a \rangle_{K_{ab}} \ , \ B \mid\equiv N'_a}{A \mid\equiv B \mid\equiv N'_a}$, in other words $A \mid\equiv B \mid\equiv Sig1$ that we reach to **G_2**.

Similarity via A_8 and goal G_1, reach the truth of $A \mid\equiv \#\langle N'_b \rangle$. From this new result and G_1 we obtain $A \mid\equiv B \mid\equiv N'_b$. Then, we apply jurisdiction rule on A_3 and last predicate to achieve correctness of $A \mid\equiv N'_b$. At last, to prove G_4, from $\frac{B \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B \ , \ B \mid\equiv \{N'_b\}_{K_{ab}} \ , \ A \lhd \langle N'_b \rangle_{K_{ab}} \ , \ A \mid\equiv N'_b}{B \mid\equiv A \mid\equiv N'_b}$ we obtain $B \mid\equiv A \mid\equiv Sig2$ or **G_4**.

C. Informal Security Analysis

1) Provable security: security is related to several standard assumptions like Bidirectional Diffie-Hellman

(BDF) or Computational Diffie-Hellman (CDH). Since the proposed protocol is based on mathematical basis and ECC, it utilizes ECDLP security and birthday paradox proofs. In this section, we study attacks and analyse our model security against them.

TABLE 3. COMPARISONS AMONG THE PROPOSED PROTOCOL AND PREVIOUSLY PROPOSED SCHEMES ECDSA: 283 BIT DOMAIN PARAMETERS,HSLV,TSV,OTS: L = 40, T = 100, AND K,R = 32, OUR PROTOCOL: T = 100.

Performance Properties	Scheme	Problem	Operations - Client Side	Operations - Server Side	Estimated Cost Client Side (ms)	Estimated Cost Server Side (ms)	MITM attack
Li et al. [15]	HSLV	OTS+ HORSE	tRNG + (t+1)H + 1 XOR + (K)MD + 2 H + 1XOR + (K)MD	tRNG + (t+1)H + 1 XOR + (K)MD + 2 H + 1XOR + (K)MD	8.3723	121.375	Yes
Li et al. [15]	TSV	OTS+ HORSE	(t)RNG + (3tw+k+2)H + 2kMD + 2 XOR	(t)RNG + (3tw+k+2)H + 2kMD + 2 XOR	8.4756	122.404	Yes
ANSI,IEEE,NIST estandard	ECDSA	ECDLP	2RNG + 2H + 4PM + 1MR + 1MA + 3 MD + 1PA	2RNG + 2H + 4PM + 1MR + MA+ 3 MD + 1PA	3.97209	57.7882	No
Perring[18]	BIBA	OTS	(3t+r)RNG + t H	(3t+r)RNG + t H	0.13944	1.1292	Yes
Proposed Protocol(Mutual Authentication)	EMAOTS	ECDLP+OTS	(t+2)RNG + (t+8) H + 2PM + 6XOR	(t+2)RNG + (t+8) H + 2PM + 6XOR	1.7127	24.5902	No
Proposed Protocol(Message Authentication)	EMA	ECDLP+OTS	(t+1)RNG + (t+2)H + $1H_1$ + 1PM + 6XOR	5H + $1H_1$ + 1PM + 6XOR	0.92122	13.038	No

2) Replay attack: since protocol uses timestamp for each packet, a hostile cannot perform a replay attack on messages. Additionally, each party use random strings and secure perfect hash functions; they protect the protocol against replay attacks.

3) *Key Privacy & Insider Attack Resilience:* In this approach there is not a pre-built private and public key, the signature and the temporary public key is created from a message that is supposed to send in every time and there is not a shared information between Server and all smart meters. Owing to this reason an insider attack cannot happen, and other smart meters cannot exploit smart meter information to arrange a local attack.

4) Off-line Guessing Attack Resilience: Let assume an attacker eavesdrops channel and obtains signature; then, he/she runs a dictionary attack on Q^x and finds this parameter, but based on ECDLP, the attacker is not capable to find PW, also the attacker could not use Q^x in future packet because it updates with timestamp mask every time

5) Denning-Sacco Attack Resilience: Our hash functions are resilient against second pre-image attack and also we do not use session key; therefore, a Denning Sacco attack could not perform on our protocol.

6) MITM Attack: In mutual authentication protocol and in the first step, we send the first packet with ID_{SM}, signature and a sequence number. Suppose a MITM captures a packet and change the signature with his/her information, but in destination, $server$ first authenticates the signature and if it fails, the server aborts the process and SM should initialize the process again; this operation exactly happens for the second packet in the SM side.

7) Denial of Service (DOS) attack: If a valid SM gets malicious it can arrange a DOS attack against server with an initial mutual authentication request repeatedly. To prevent this attack, the server can restrict the number of mutual authentication request for a certain SM in a period of time.

8) Key escrow: Our protocol does not use the Key Generator Center (KGC), and there is a direct communication between smart meters and the server; thus, protocols do not have the key escrow problem; Furthermore, in this scheme we do not use session key, and the key session escrow problem is not observed as well.

TABLE 4. CRYPTOGRAPHC OPERATION TIME (IN MILLISECONDS)

Symbol	Operation	Server Side	Client Side
RNG	Random number	0.00042	0.0031
H	String to number hash	<0.0001	<0.001
H1	String to point hash	0.08	0.947
PM	Point multiplication	0.83	12.08
PA	Point addition	<0.001	<0.01
Minv	Modular inversion	0.13	1.89
MA	Modular addition	0.13	1.89
MR	Modular reduction	0.13	1.89
MD	Modular division	0.13	1.89
EM	Elementary multiplication	<0.0001	<0.001

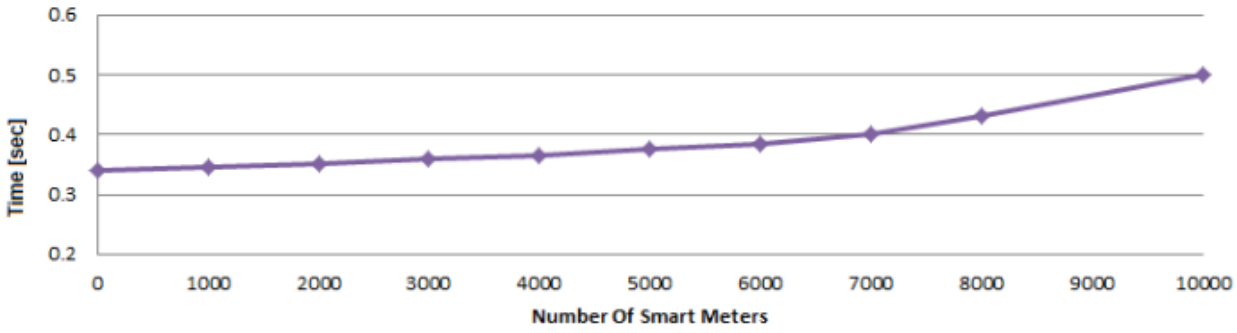

Fig. 9. : End to End Delay verification for Mutual Authentication Protocol.

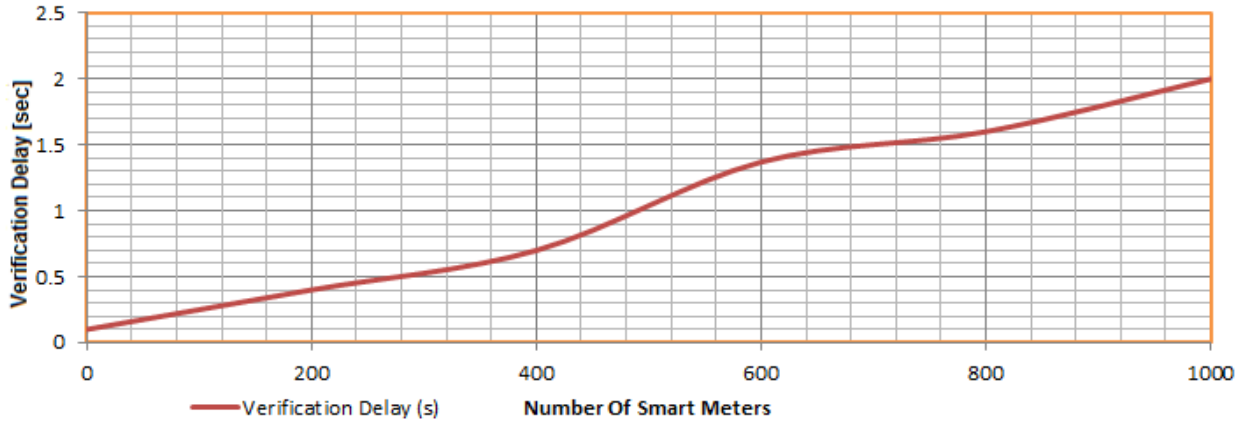

Fig. 10. : Verification Delay for Message Authentication Protocol.

D. Performance Analysis

In this section we intend to compare both proposed protocols, EMAOTS and ECC-based message authentication with OTS protocol, with five other similar methods. BiBa [18] was introduced in part A of section 2; HORS protocol was proposed by Reyzin and Reyzin [21] is another symmetric method for message authentication with OTS, three proposed protocols HSLV, LSHV, and TSV by Li, Q. et al. [15] were brought in part B of related work, and the ECDSA basic ECC-based schema in ANSI, IEEE, and NIST standards.

In Table 2, for the convenience of evaluating the computational cost, some notations are defined as follows:

H: symbol of using hash function.

Log: symbol of using the logarithm in base 10

S. Mul: symbol of using scalar multiplication on EC.

t: number of generated l-bit strings.

K: K is notation of $K-way$ collision and could be two or three.

r: r is notation for choosing r element of t strings and usually is as large as t. Choosing an appropriate t and r in this table is related to the required security and availability of computational space.

$\boldsymbol{\mu}$ **:** This notation is equal to $\frac{t^K}{t\,(t-1)\ldots(t-K+1)}$

PM: symbol of using point multiplication

PA: symbol of using point addition

Minv: symbol of using modular inversion

Mod [m,n]: gives the remainder on division of m by n

In Table 2, EMAOTS is our mutual authentication protocol, and the proposed message authentication is our message authentication protocol. Six methods, comparatively, only established message authentication and they do not consider mutual authentication in these methods. In the key generation cost part, PM cost, which we refer to as scalar multiplication on EC used for solving MITM attack in the previous method. As shown in Table 2, via considering more security in the proposed methods, they are more efficient in computational cost.

In Table 3, to prove the improvement, we compared the proposed algorithms under the above recommendation with ECDSA, OTS, HSLV and TSV as shown in Table 3. In the case of ECDSA, we measured the execution time for signing and verifying with 283 bit domain parameters according to [25]. In the case of HSLV and TSV, the parameters are set: l = 40, t = 100, w=50, and k = 32 according to [9]. The client (the smart meter) is a low-power computing device while the server is regarded as a powerful device. All the operations are built with MIRACLE [26], a standard cryptographic library. The hardware platform is a PIV 3-GHZ processor with 512-MB memory and a Windows XP operating system according to [26]. The running times of different operations are listed in Table 4.

We ignore the cost of exclusive XOR, and multiplication (M) operations because they are insignificant compared to other operations. Here, the mathematics and cryptographic operations performed on both the client side and the server side are given in column four and five, respectively. The estimated timings for these operations can be found in column six and seven, respectively. These timings are calculated using the primitive arithmetic and cryptographic operation timings given in Table 4.

As expected, performance of hash based schemes is significantly better than that the other schemes, Like BIBA protocol, because the computational cost of calculating a hash value is very low. However, hash based schemes have some security weaknesses. As BiBa has a low security level in comparison with other schemas. In BiBa the signature size and verification time are small, but the public-key size is large with reasonable overhead for signature generation. As can be seen, the proposed protocols have significant reduction in processing time. Because the proposed protocols are a combination of the two ECC-based and OTS-based methods, our mechanism combined advantages of them and the significant improvement achieved.

i. Simulation

In this section, we simulate our protocols via Java programming in a platform with this description; server: A HP server, Xeon E3 series, Core i3, Pentium, Celeron processor with 16-GB memory and a RHEL 6.3 operating system. Smart meter: an Intel atom 1800.0 MHz processor with 512-MB memory and a Knoppix operating system.

Scenario: in this scenario 1000 smart meters, send request to control center server at the same time. In the first protocol smart meters send requests for mutual authentication session, and in the second protocol first party sends signed messages. In order to achieve more security and reliability in simulation protocol should be executed in 50 parallel runs, and the average of this 50 parallel runs.

Figure 6 shows end to end delay (per second) with increasing number of smart meters in mutual authentication protocol. Figure 7 illustrates the verification delay (per second) with increasing number of smart meters in the message authentication protocol.

6. CONCLUSIONS

In this paper, we have proposed an ECC-based mutual authentication protocol and a message authentication protocol using OTS in AMI. According to the compressions in performance analysis section, the proposed protocols are more efficient and practical than similar works. Compared to existing schemes, our schemes have an impressive reduction in public key, signature size, and verification cost. We utilized ECC to solve the MITM problem. The proposed mechanism takes advantage of OTS for fast verification. In future, we will improve signing step computation.

REFERENCES

[1] Kim, Y.S., Heo, J., “Device Authentication Protocol for Smart Grid Systems Using Homomorphic Hash,” IEEE Network and Communication. Vol.14, pp. 606-613, 2012.

[2] Ray, P. D., Harnoor, R., Hentea, M., “Smart Power Grid Security: A Unified Risk Management Approach,” IEEE Int’l. Carnahan Conf. Security Tech., pp. 5–8, 2010.

[3] Li, X., Liang, X., Lu, R., Shen, X., Lin, X., Zhu, H., “Securing smart grid: cyber attacks, countermeasures, and challenges,” IEEE Communications Magazine, vol. 50, no. 8, pp. 38–45, 2012.

[4] U. S. Department of Energy, [online] Available: www.oe.energy.gov.

[5] Yan, Y., Hu, R.Q., Das, S.K., Sharif, H. Qian, Y., “An Efficient Security Protocol for Advanced Metering Infrastructure in Smart Grid,” IEEE Trans. Network. vol. 27, pp. 64-71, 2013.

[6] Wan, z., Wang, G., Yang, Y. and Shi, S., "SKM: Scalable Key Management for advanced metering infrastructure in smart grids," IEEE Trans. Ind. Electron., vol. 61, no. 12, pp.7055 -7066, 2014.

[7] Li, H., Lu, R., Zhou, L., Yang, B., Shen, x., "An Efficient Merkle-Tree-Based Authentication Scheme for Smart Grid," Journal of System IEEE trans., pp. 655 - 663, 2014.

[8] Saputro, N. and Akkaya, K., "On preserving user privacy in smart grid advanced metering infrastructure applications," Security Commun. Netw., vol. 7, no. 1, pp. 206–220, 2014.

[9] Li, Q., Cao, G, "Multicast Authentication in Smart Grid with One-Time Signature. IEEE Trans. smart grid," pp. 686-696, 2011.

[10] Metke, A. R., Ekl, R. L., "Security technology for smart grid networks," IEEE Trans. Smart Grid, vol. 1, iss. 1, pp. 99–107, 2010.

[11] Miller, V.S., "Use of Elliptic Curves in Cryptography," Advances in Cryptology — CRYPTO '85 Proceedings, pp. 417-426, 1986.

[12] Koblitz, N., "Elliptic curve cryptosystems," Mathematics of Computation, pp. 203–209. JSTOR 2007884, 1987.

[13] Lamport, L., "Constructing digital signatures from a one way function," Technical Report. CSL-98, SRI International, 1979.

[14] Rabin, M.O., "Digitalized signatures," In Richard A. Demillo, David P. Dobkin, Anita K. Jones, and Richard J. Lipton, editors, Foundations of Secure Computation, 1978.

[15] Merkle, R.C., "Secrecy, authentication, and public key systems," UMI Research Press, 1982.

[16] Gao, W., Li, Q., Zhao, B., Cao, G., "Multicasting in delay tolerant networks: a social network perspective," ACM, New York, pp. 299-308, 2009.

[17] Reyzin, L., Reyzin, N., "Better than BiBa: Short one-time signatures with fast signing and verifying," Proc. of the Australian Conference on Information Security and Privacy, pp. 144-153, 2002.

[18] Bos, J.N.E., and Chaum, D., "Provably unforgeable signatures," Advances in Cryptography-CRYPTO, pp.1-14, 1993.

[19] Menezes, A.J., Oorschot, P.C.V., Vanstone, S.A. "Handbook of Applied Cryptography," CRC Press LLC, USA, pp. 52-53, 1997.

[20] Yaghmaee, M.H., Naji.M, F, "A Lightweight Mechanism for Mutual Authentication in Smart Grid," Challenges of Implementing Active Distribution System Managment.0223. Cired Workshop 2014.

[21] Perring, A, "The BiBa one-time signature and broadcast authentication protocol," Proceedings of the 8th ACM conference on Computer and Communication security, pp. 28-37, 2001.

[22] Kgwadi, M., Kunz, T., "Securing RDS broadcast messages for smart grid applications," In Proceeding of the 6th International Wireless Communications and Mobile Computing, ACM Conference, pp. 1177-1181, 2010.

[23] Burrow, M., Abad, M., Needham, R., "A logic of authentication," ACM Trans. Comput. Syst. 8, pp. 18–36, 1990.

[24] Syverson, P., Cervesato, I., "The logic of authentication protocols," in: Foundations of Security Analysis and Design, Tutorial Lectures, in: Lecture Notes in Comput. Sci., vol. 2171, Springer, pp. 63–136, 2001.

[25] Certicom Research, SEC2: Recommended elliptic curve domain parameters v1.0, 2000.

[26] Shamus Software Ltd., Miracl Library, <http://www.shamus.ie/index.php?page=home>.

[27] Debiao, H., Jianhua, Ch., and Jin, H., "An ID-based client authentication with key agreement protocol for mobile client–server environment on ECC with provable security," elsevier, journal of Information Fusion, pp. 223-230, 2012.