A tailored authentication and key management for smart grid

Hasen Nicanfar, *Student Member, IEEE,*

***Abstract*—Smart Grid (SG) is a vulnerable system and can be attacked even from aboard, attacks that may cause different level of costly issues and harms on the society as well as on the system devices. Furthermore, in SG we have a variety of sub-systems and applications as well as networks which are working together as a System of System (SoS) model. Therefore one of the most challenging topic in the SG development is security and privacy. Designing a mutual authentication scheme and then a key management protocol are the first tailored steps of designing and implementing the security aspects of any system like SG.**

In this paper we improve and implement the Secure Remote Password protocol to reach a mutual authentication scheme between a Home Area Network (HAN) Smart Meter (SM) and an authentication server in SG using an initial password. We propose using the Public Key Infrastructure for the SG communications, for instance between SM and aggregator (outside of HAN). Also, in order to have an efficient key management protocol, we follow an enhanced model of the Identity-Based cryptography so-called EIBC.

Our proposed mechanisms are capable of preventing various attacks, and at the same time, improves the network overhead caused by the key management controlling packets. In fact, mostly by generating and broadcasting only one function periodically by the key generator entity, our protocol simply refreshes entire nodes public/private key pairs as well as multicast required keys, if any.

***Index Terms*—Mutual Authentication; Key Management; SRP; Security; Smart Meter; Smart Grid; EIBC.**

I. Introduction

PROVIDING a reasonable level of the security and privacy is one of the most important and challenging topics in the smart grid (SG) context that has gained research community attention. SG is a vulnerable system and can be attacked even from aboard, attacks that may cause different level of issues and harms on the devices and society [2]. Providing an authentication scheme and then a key management protocol are the preliminary tailored steps of designing and implementing security for any system such as SG [3].

By definition, authentication means binding identity to a subject or principal. It can be shown by what the subject is capable of doing (like performing a digital signature), or knows (like a password or PIN), or has (like a smart card), or is (biometrics, like fingerprint) . It can be applied to any computer system or to any network based application. For instance, when a user wants to use a system, the combination of the user name and knowing appropriate and aligned password provides the user authentication to the server. Moreover, when two parties want to communicate to each other, they may have a shared password, which in this case they require to be mutually authenticated to each other. In general and mostly in a networking environment, nodes or any two parties should follow mutual authentication to prevent some of the attacks (more detail can be found in literature).

After two parties (a client and a server or any two network nodes) get authenticated to each other, they need to set-up a secure communication to prevent unauthorized parties intrusion. To address this demand, they need to protect their communicating data packets, which normally they use a security key. Therefore, as a second security requirement, they need to have a key management protocol to form the required key/s, and normally, refreshes the key/s. The key can be (i) symmetric also known as private key infrastructure, or (ii) asymmetric also known as Public Key Infrastructure (PKI). In the symmetric key, packets are encrypted and decrypted with the same key, so sender and receiver should share the key. However and in the asymmetric key, sender and receiver use different keys.

Public Key Infrastructure: In the public key encryption, two keys are introduced and being provided per each entity, so-called public key and private key. The private key is supposed to be kept by each entity (Alice) in private and secret; however, the party's public key is defined to be accessible publicly, for instance by Bob. A third party (Trent) also exists that acts as a Private Key Generator (PKG)/Certificate Authority (CA). For instance, Trent (or PKG/CA) issues for each entity an individual certificate consists of the entity's private key. Furthermore, system is inquired to have a solution to distribute the parties' public keys. For example, when Bob wants to send a private message to Alice, he encrypts the message using the Alice's public key. On the other end, Alice would be able to decrypt the received encrypted-message utilizing her own private key. Furthermore, to make Alice assure of message origin (sent by Bob), Bob should sign the message. Normally, Bob uses his own private key to provide a signature, and Alice refers to the Bob's public key for verifying the Bob's signature on the message.

In above mentioned approach, first of all, Alice needs to have access to the Bob's public key as well as her own private key, which both are generated and managed by Trent. Also, these keys are required to be refreshed periodically as per system application, which is also being handled by

This paper is based in part on a paper appeared in Proc. of the first IEEE PES Innovative Smart Grid Technologies (ISGT) Asia conference, as well as a paper in Proc. of the 2012 IEEE International Systems Conference (SysCon).

Hasen Nicanfar is with the WiNMoS Lab, department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, V6T 1Z4 Canada.

E-mail: hasennic@ece.ubc.ca.

Trent. Thus, Trent regenerates keys and informs the parties frequently. The public key should be distributed to the entire parties, and the private key needs to be sent to each one individually and via a secure communication. More detail about the managing public/private key pair as well as key and certificate refreshment can be found in the literatures e.g. our references. To overcome the public key distribution overhead cost, one solution is Identity-Based Cryptography (IBC).

Identity-Based Cryptography: In IBC, system can provide a unique function F (one-way like a hash function) to the entire parties, which can be applied to each party's ID to obtain the party's public key (1). For instance, a party/entity ID can be the party's email address, phone number or IP address, or combination of them. Then, PKG select a random number s, and applies it to each node's public key and product will be the node's private key (2).

$$\begin{cases} PubK(ID) = F(ID) & (1) \\ PrvK(ID) = s * F(ID) = s * PubK(ID) & (2) \end{cases}$$

Let us consider Alice and Bob to be our two parties in the following three main sections of the IBC:

1) ID-Based Encryption (IBE):

a) Setup (Initialization): PKG randomly selects the system non-shared secret value s and also generates PKG's public key and makes it publicly accessible (e.g. by Alice and Bob).

b) Private key extraction: PKG generates the Alice's private key and sends it to Alice via the secure channel.

c) Encryption: Bob applies function F to the Alice ID to obtain the Alice's public key, and encrypts his message to Alice using the Alice's public key.

d) Decryption: Alice utilizes her own private key to decrypt the encrypted message received from Bob.

2) ID-Based Signature (IBS):

a) Setup (Initialization): Similar to the IBE mechanism.

b) Private key extraction: Similar to the IBE mechanism.

c) Signature: Bob signs the message utilizing his own private key, and then sends the signature along with the message to Alice.

d) Verification: Alice applies F to the Bob ID and obtains the Bob's public key, and verifies the received signature from Bob by using the Bob's public key.

3) Key Refreshment: In order to refresh the keys to maintain system security, periodically PKG reselects s and recalculates all entities' private keys and informs them one by one via the secure channel. Since this algorithm takes time, normally PKG supplies the parties with a valid-time that presents the starting time of using the new keys.

Password utilization for authentication: Referring to our discussion about the authentication and prior to any key management, parties should be able to start communicating in order to authenticate each other. For instance, they form a session key to receive their private key from the server (CA, PKG or Trent). There are different solutions assuming to have a preliminary password, which we review some of them under the Password Authenticated Key Exchange (PAKE) protocol umbrella.

One of the solution to form a session (symmetric) key is W. Diffie and M.E. Hellman (D-H) algorithm [4], also know as D-H algorithm. To protect D-H algorithm process from different attacks like Man-In-The-Middle attack, a solution was proposed by S. Bellovin et al. in 1992 [5]. They utilized a password to assure required key establishment messages secrecy. Later on, D.H. Seo et al. developed a two steps PAKE protocol called SAKA. First, both parties obtain a number as product of their shared password, and its reverse. Then, each party picks a random number and multiplies it to the shared number in the first step to be used in D-H algorithm. Their model comprises two more steps for key verification [6]. More PAKE protocols have been developed during last couple of decades, for instance in [7], [8], [9], [10] and [11]. In [12], utilizing a verifier is proposed which assumed having a trusted server as a third party for supporting key establishment. Each party has an individual password and server holds the appropriate verifier. Entities establish temporary session keys used to establish the final symmetric key in this four phases protocol.

In this paper we improve and implement the Secure Remote Password (SRP) protocol [13] to reach a mutual authentication between a Home Area Network (HAN) Smart Meter (SM) and a security or authentication server (SAS) in the SG system, by utilizing an initial password. We propose using PKI for the SG communications, for instance between SM and aggregator (outside HAN communications). Also, in order to have an efficient key management protocol, we follow an enhanced model of IBC, which has been designed recently in our lab so-called EIBC [14]. We will review more detail of SRP and EIBC in Section II.

Our proposed mechanisms are capable of preventing various well-known attacks, for instance Brute-force, Replay, Man-In-The-Middle and Denial-of-Service attacks. Furthermore, we improve the network overhead caused by the key management controlling packet. In fact, in most of the key refreshment cases, only by generating and broadcasting one function periodically by the SAS server that is in charge of the key generation, our protocol simply refreshes the entire nodes' public/private key pairs as well as multicast required security keys, if any.

Following to the introduction and in Section II, we review the literature respecting to our work. In Section III and Section IV, we present our mutual authentication scheme and key management protocol respectively, which are analyzed in Section V. Section VI is our conclusion and also our future works followed by list of our references.

II. Literature Review

In this section, we review some of the related researches to our paper, such as EIBC and SRP. Also, we study some of the recent proposals about the SG network and structure.

A. EIBC: Enhanced Identity-Based Cryptography

Recently we have designed and proposed EIBC in [14]. Following to the above mentioned definitions, we review the areas that we have proposed some modifications/developments:

a) One-way/Hash function $F(.)$: EIBC makes this function to be a dynamic $F_i(.)$. Periodically, PKG broadcasts a function $f_i(.)$ that applies to $F_i(.)$ to obtain $F_{i+1}(.)$, the system new one-way function. In this case, all of the public keys and private keys are being updated. Nodes need to apply $f_i(.)$ to any nodes' public key to obtain the updated node's public key. Also, each node uses $f_i(.)$ as part of the private key refreshment algorithm (we will explain this part shortly). Note that "i" presents system iteration and current/$live$ system state.

$$\begin{cases} F_{i+1}(.) = f_i(F_i(.)) & (3a) \\ PubK_i(ID) = F_i(ID) & (3b) \end{cases}$$

b) System secret value "s": In IBC, system secret value s is the product of a True Random Number Generator (TRNG) managed (and being kept secret) by PKG. EIBC makes s combination of two values, such as a non-shared TRNG value s_i kept by PKG (4a), and a Pseudo Random Number Generator (PRNG) value $\widetilde{s}_i$ shared by the entities (4b), with similar duties.

$$\begin{cases} s_{i+1} = f_{i+1}(s_i) & (4a) \\ \widetilde{s}_{i+1} = a * \widetilde{s}_i + b & (4b) \end{cases}$$

c) Seed value vs. End value: We specified some of the parameters to have a seed value. For instance, PKG has "public key seed value" ($\widetilde{PubK}^{i}_{PKG}$) and "public key end value" ($PubK^{i}_{PKG}$). Moreover, each entity, like Alice, has a private key seed value ($\widetilde{PrvK}^{i}_{A}$) and a private key end value ($PrvK^{i}_{A}$). PKG produces the seed values (5a) and (5b), and entities perform (6a) and (6b) to obtain the live end values in each system iteration:

$$Seed\ values: \begin{cases} \widetilde{PubK}^{i}_{PKG} = s_i.\breve{P}^{i}_{PKG} & (5a) \\ \widetilde{PrvK}^{i}_{A} = s_i.F_i(ID_A) & (5b) \end{cases}$$

$$End\ values: \begin{cases} PubK^{i}_{PKG} = f_i(\widetilde{s}_i).\widetilde{PubK}^{i}_{PKG} & (6a) \\ PrvK^{i}_{A} = f_i(\widetilde{s}_i).\widetilde{PrvK}^{i}_{A} & (6b) \end{cases}$$

d) Key refreshment periods: As it can be seen from the last two points, our system in EIBC requires to update different values. To be more precise, we need to produce a new function $f_i(.)$, manage a new value for $\widetilde{s}_i$, have the set up values a & b refreshed, and finally, having/refreshing a new value for s_i same as the original algorithm in IBC. Hence, EIBC comes with three timers such as Short, Medium and Long term refreshment (STR, MTR LTR) timers to cover the appropriate processes.

- $STR\ process$: Every STR, PKG generates a new function $f_i(.)$ and makes it publicly accessible, along with a Valid-Time (VT) which is the start time of moving to the new system state ($i \rightarrow i+1$). At the time of VT, each party refreshes $\widetilde{s}_i$ based on (4b), refreshes $F_i(.)$ based on (3a) in order to have others' refreshed public keys. Also, the party refreshes the PKG's public key based on (7a) and (7b), as well as its own private key based on (7c) and (7d), utilizing the updated values of $\widetilde{s}_{i+1}$ and $F_{i+1}(.)$:

$$\begin{cases} \widetilde{PubK}^{i+1}_{PKG} = f_{i+1}(\widetilde{PubK}^{i}_{PKG}) & (7a) \\ PubK^{i+1}_{PKG} = f_{i+1}(\widetilde{s}_{i+1}).\widetilde{PubK}^{i+1}_{PKG} & (7b) \\ \widetilde{PrvK}^{i+1}_{A} = f_{i+1}(\widetilde{PrK}^{i}_{A}) & (7c) \\ PrvK^{i+1}_{A} = f_{i+1}(\widetilde{s}_{i+1}).\widetilde{PrvK}^{i+1}_{A} & (7d) \end{cases}$$

- $MTR\ process$: Every MTR, PKG reselects system PRNG parameters a & b along with the requires VT, and shares them with the entire parties to be used starting by VT.
- $LTR\ process$: Every LTR, PKG reselects the system non-shared secret values along with system shared secret values and updates one-way function F_i, in order to refresh the whole keys (entire parties' public and private keys). PKG also refreshes each party's private key, and inform the party along with a VT via a secure channel.

Note that LTR process is similar to the IBC key refreshment algorithm. As it has been analyzed in the [14], EIBC simultaneously improves key management process overhead cost and system security level.

e) Multicast group key support: There are two proposals to support the required multicast key in EIBC. We only review the solution that we are using in this paper as part of Section IV, which is *Multicast group source/receiver key pair*. Each multicast group is identified by a Multicast Group ID (MID) that is being used similar to an entity's ID to obtain group's Source Multicast Key (SMK) referring (1). At the same time, referring (5b) and (6b), each group would have a Receiver Multicast Key (RMK) managed by SAS. Each Multicast Group Source (MGS) entity receives that group's SMK and RMK, and grants membership to a Multicast Group Receiver (MGR) entity by sending RMK to the new MGR. Therefore, when a MGS wants to send a packet to MGRs, MGS encrypts the messages by SMK. On the other side, a MGR uses group's RMK to decrypt the received message. In order to have a multicast packet's source authentication and because a SMK can be compromised, MGS signs the messages using its own entity (original) private key ($PrvK^{i}_{ID}$).

The other solution provided by EIBC is having a Muticast Group Pseudo Random Number Generator $\widetilde{sm}_i$, similar to $\widetilde{s}_i$, with its own setup values am & bm and initial value $\widetilde{sm}_0$. Receivers use $\widetilde{sm}_i$ to refresh the group's RMK.

B. Secure Remote Password (SRP) Protocol

SRP in [13] is an authentication and key-exchange protocol for secure password verification and session key generation over an insecure communication channel. The SRP protocol utilizes Authenticated Key Exchange (AKE), and stores verifiers instead of the passwords. AKE uses a one-way (hash) function to compute the verifier and stores it in the server. Also compromising the server and finding the verifier is not enough, since the password is still required.

First of all in SRP, the user enters a password and then a verifier is computed from the password among with a randomly generated password salt. Then, the user name, salt

and verifier are stored in the server database. Finally, the client can now be authenticated to the server, as it is depicted in Fig. 1. All of the calculation are in modulo p, and in some of the steps, one-way hash function is being used. If M_1 and M_2 in the seventh and eighth steps are the same in both ends, the mutual authentication is successful on the client as well as the server sides.

C. Smart Grid network structure

Our study shows that SG most probably will have IPv6 technology in a mesh based topology for outside of the HAN domain. For instance, [15] assumes system has an IP-Based communication between a SM and the utility's Meter Data Management. One of the latest research in this area is proposed by H. Gharavi et al. in [16]. They designed a mesh based architecture for the last mile SG, compromises two domains. One of the domains, which is in the Neighbourhood Area Network (NAN), supports communication between HAN and AMI head-end via data aggregation point and mesh-relay-station if required. This mesh based topology is in-charge of expanding the coverage area of the network by using multiple hops connection.

III. Smart Grid Mutual Authentication (SGMA)

In this paper, we concentrate on communication outside of the HAN domain. These data communication could be between SMs of different homes, or between SM and an aggregator or a controllers like a NAN controller, and for sure between SM and security server SAS which provides the required authentication and key management supports. We also cover the key management for the unicast, multicast and broadcast communication desired by any application in these scope.

Followings are our assumptions used in this paper in order to design the mechanisms:

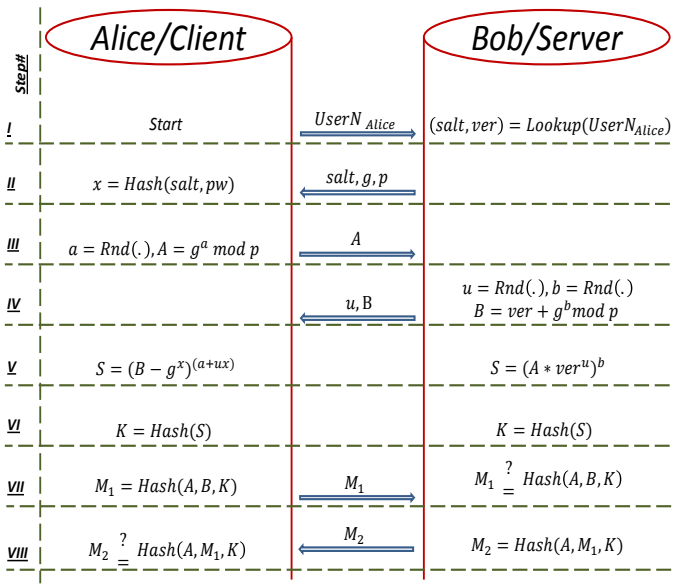

Fig. 1. Original Secure Remote Password Protocol

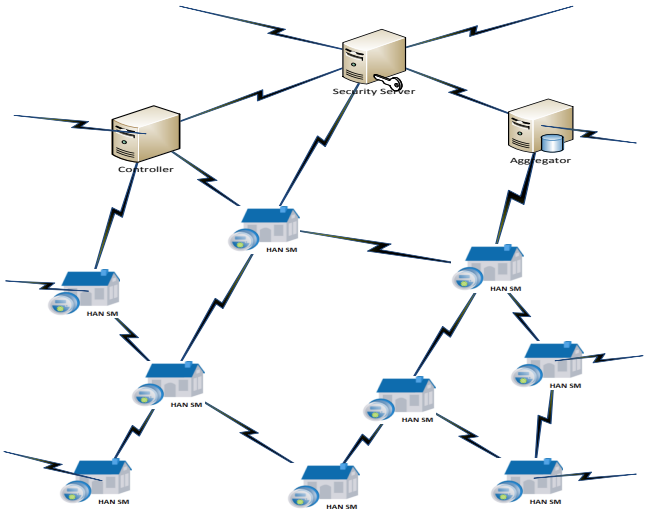

Fig. 2. Smart Grid Topology for Outside HAN domain

- Network topology between nodes is a wireless mesh network topology.
- Nodes have the required unicast technology support to be able to communicate with each other via multiple hops.
- Each node has a unique IP address (most likely IPv6), which can be used as node's ID. This ID can be assigned by a technician at the set up time.
- SM acts as a HAN gateway which separates outside and inside HAN domain communication.
- SM has a unique serial number SN and an initial secret password pw for authentication. On the other hand, SAS holds the appropriate ver and $salt$ (referring SRP protocol) for each SM.
- Each node initially comes with a $Hash(.)$ function, and values g & p to be used in SRP algorithm, or can receive them from the technician at the set up time.
- Nodes are all synchronized respect to the current time, and the new installed SM would be able to synchronize itself with others, using a valid synchronization algorithm.
- SAS server is responsible for the authentication as well as key management.

Fig. 2 presents topology of our study domain area, which we used the researches in [16]. Referring Section I, we assume a new SM (mutually) authenticates itself to the SAS server, which will receive its private key as well.

Definition: Let us define system state (i, j):

Dimension "i": This dimension represents the system functions $f_i(.)$ & $F_i(.)$ as well as random values s_i & $\widetilde{s}_i$.

Dimension "j": This dimension represents PRNG set up values a_j & b_j used in (4b). Note that in (4b), we only showed a & b for simplicity.

A. Mutual Authentication Scheme

Depicted by Fig. 3, our SPR based mutual authentication scheme consists of three following steps:

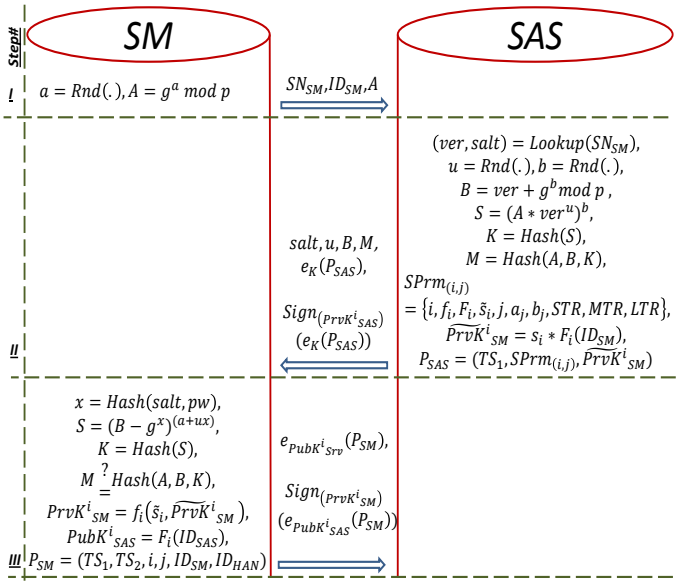

Fig. 3. Smart Grid Secure Remote Password Protocol

1) Step I: Firstly, new SM select a random value a and calculates $A = g^a \bmod p$. Then, SM sends A along with its own SN and ID to the SAS server.

2) Step II: SAS performs the following steps as per receiving the first packet:

- SAS lookups the values ver & $salt$ respect to the serial number SN.
- SAS picks random values u & b, and calculates $B = ver + g^b \bmod p$.
- Then, SAS computes $S = (A * ver^u)^b$ followed by $K = Hash(S)$ and verifier value M as $M = Hash(A, B, K)$.
- Furthermore, SAS computes the SM's private key seed value $\widetilde{PrvK}^i_{SM}$.
- Finally, SAS sends values $salt$, u, B & M along with encrypted and signed system's parameters (like pseudo random number generator) to SM.

3) Step III: SM performs the following steps when receives packet II from SAS:

- SM calculates $x = Hash(salt, pw)$, and then $S = (B - g^x \bmod p)^{(a+u.x)}$.
- Then, SM calculates K as $K = Hash(S)$, and then verifies K based on the received M, by comparing M with $Hash(A, B, K)$.
- If condition holds, SM has the valid symmetric key K shared by the server. So, SM is able to decrypts received values, as well as is capable of checking the signature.
- Finally, SM obtains its own private key and sends an encrypted and signed acknowledgement to the SAS server.

At this point, SM and SAS are mutually authenticated to each other, and SM has received system parameters as well as its own private key.

IV. Smart Grid Key Management (SGKM) Protocol

Our proposed key management follows EIBC key management algorithm. Thus far (assuming Section III is done), nodes have the appropriate PKI-Based keys to be used for unicast and node-to-node secure communication. In this section, we introduce our key refreshment mechanism as well as a solution for required multicast and broadcast keys (including their related key refreshment).

A. Key refreshment

Referring EIBC presented in Section II and reference [14], system requires three timers STR, MTR and LTR values to be set. We manage transferring these timers as part of the system parameter in our authentication scheme algorithm.

1) Short term refreshment process: Our short term refreshment process is presented by Fig. 4. Referring to the STR timer value, system runs this process in a regular basis to change the system state from (i, j) to $(i + 1, j)$.

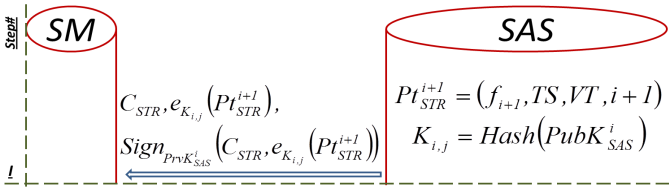

Fig. 4. Broadcasting an encrypted and signed packet in the STR process

a) SAS duties: First of all, SAS generates a new function f_{i+1} according to the new system state $i + 1$. Then, SAS prepares a packet Pt^{i+1}_{STR} containing the f_{i+1} function, Time Stamp TS of the f_{i+1} production, Valid Time VT of the new system state's dimension i and its new value $i+1$. Then, SAS applies the original $Hash(.)$ function to its own live public key to obtain a symmetric key $K_{i,j}$ (8):

$$K_{i,j} = Hash(PubK^i_{SAS}) \tag{8}$$

Note: We will use this technique to handle the broadcasting key in the broadcast key management part at the end of this section.

Finally, SAS broadcasts the encrypted packet Pt^{i+1}_{STR} utilizing the $K_{i,j}$ key, along with the controlling STR command C_{STR}. SAS also signs this values with its own live private key in order to have source authentication.

b) SMs duties: As soon as a SM receives the broadcasting information identified by C_{STR}, obtain the SAS's live public key to verify the signature. Assume the signature is valid, SM calculates the symmetric key $K_{i,j}$ following (8) and decrypts the received packet Pt^{i+1}_{STR}. Then, SM controls the system iteration $i+1$ making sure is one after the current iteration, along with the TS checking to prevent the replay attack. Finally and prior to the VT time, SM utilizes the f_{i+1} and follows the defined EIBC's short period refreshment process (Section II) steps and uses the equations (7a)-(7d) to refresh the appropriate keys and starts using them from VT.

2) Medium term refreshment process: Our medium term refreshment process is presented by Fig. 5. Based on the MTR timer value, system runs this process in a regular basis to change the system state from (i, j) to $(i, j + 1)$.

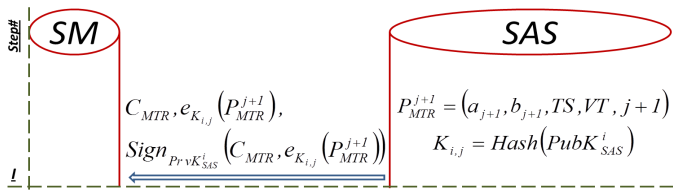

Fig. 5. Broadcasting an encrypted and signed packet in the MTR process

a) SAS duties: Referring EIBC definitions in Section II, first of all SAS generates a new PRNG's pair values a_{j+1} & b_{j+1} for the new system state $(i, j+1)$. Then, SAS prepares a packet Pt_{MTR}^{j+1} containing the a_{j+1} & b_{j+1} values, Time Stamp TS of the pair values a_{j+1} & b_{j+1} production, Valid Time VT of the new setup values plus the new system state's dimension value $j+1$. Then, SAS applies the original $Hash(.)$ function to its own live public key to obtain a symmetric key $K_{i,j}$ (8). Finally, SAS broadcasts the encrypted packet Pt_{STR}^{j+1} utilizing the $K_{i,j}$ key, along with the controlling MTR command C_{MTR}. SAS also signs this values with its own live private key in order to provide source authentication.

b) SMs duties: When a SM receives the broadcasting information identified by C_{MTR}, obtains the SAS's live public key to verify the signature. Assume the signature is valid, SM calculates the symmetric key $K_{i,j}$ following (8) and decrypts the received packet Pt_{MTR}^{j+1}. Then, SM controls the system state's dimension $j+1$ making sure is one after the current one (j), along with the TS checking to prevent the replay attack. Finally, starting by the VT time, SM updates its $\widetilde{s}_i$ setup parameters.

3) Long term refreshment process: Our long term refreshment process is presented by Fig. 6. Referring to the LTR timer value, system follows this process to go from the (i, j) state to the $(0, 0)$ state. It obvious the SAS needs to generates the system parameters as well as each node private key and inform them one by one.

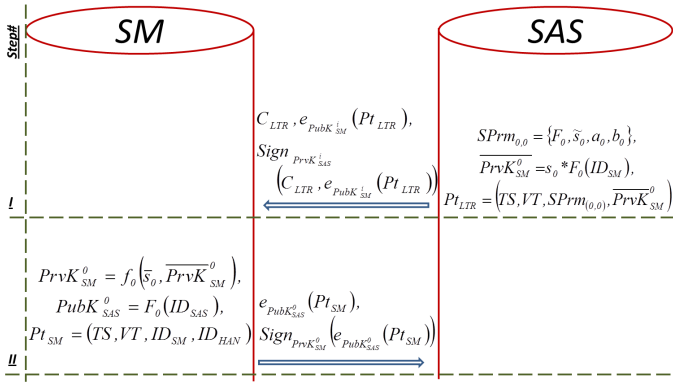

Fig. 6. Unicasting an encrypted and signed packet in the LTR process

B. Multicast key mechanism

We propose multicast group key source/receiver pair key, referring to Section II. We define Multicast Group Source (MGS) key that is being used by each group source to encrypt the multicast packets. Furthermore, we define Multicast Group Receiver (MGR) for each group that is being used by group receivers to decrypt the messages that are encrypted by MGS. Our assumptions are:

- Multicasting group is group/source based, and joining is initiated by receiver.
- Each group is identified by a unique Multicast Group ID (MID).
- SAS is in charge of the multicast group key management.

Beside the MGS and MGR keys, each group has a public/private key pair as well, to be used in multicast join algorithm. Like any party and since each group has a MID, system manages this key pair referring to (5a), (5b), (6a) and (6b).

For MGS and MGR, we define multicast group state $(k \ \& \ l)$ similar to the $(i \ \& \ j)$ state. Furthermore, $g_k(.)$ & $G_k(.)$ similar to the $f_i(.)$ & $F_i(.)$ functions, and finally m_k & $\widetilde{m}_k$ along with c_l & d_l similar to the s_i & $\widetilde{s}_i$ and a_j & b_j items in our original system design for the unicast communication.

$$\begin{cases} G_{k+1}(.) = g_k(G_k(.)) & \text{(9a)} \\ m_{k+1} = g_{k+1}(m_k) & \text{(9b)} \\ \widetilde{m}_{k+1} = c_l * \widetilde{m}_k + b_l & \text{(9c)} \\ SMK_k = G_k(MID) & \text{(9d)} \\ RMK_k = (\widetilde{m}_k, (m_k * G_k(MID)) & \text{(9e)} \end{cases}$$

1) Establishing a multicast group: A MGS that wants to form a multicast group (i) sends the request to SAS. (ii) SAS provides MGS with the group initial parameters set including MID, $\widetilde{m}_0$, RMK_0 & $G_0(.)$ along with the group's private key seed value as per (5b) and (6b) respect to MID. (iii) Then, MGS picks c_0, d_0 & $g_0(.)$ and makes the group parameters set completed for the multicast group $(0, 0)$ state. From this point, MID will be publicly accessible by the parties which wants to join. Note that MGS is in charge of $g_k(.)$ production.

2) Joining multicast group: Our join algorithm presented by Fig. 7 follows the following steps:

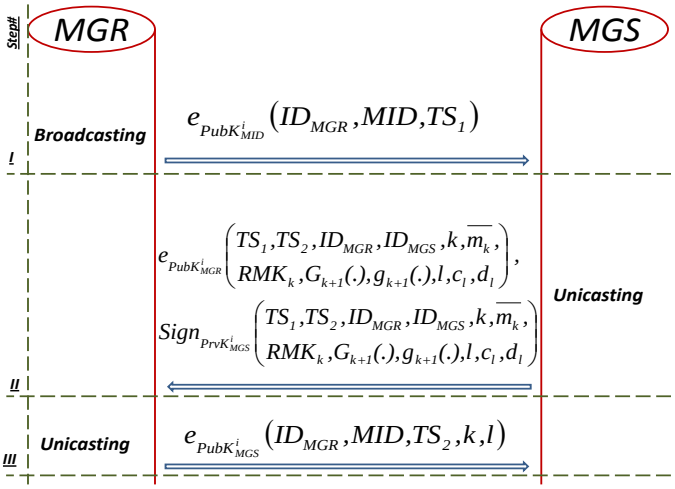

Fig. 7. Joining a Multicast Group

a) Join request (Step I): The new MGR applies the current system state function $F_i(.)$ to MID and obtain appropriate public key (3b). Then, MGR sends (broadcasts) its join request encrypted by the group's public key, including its own ID.

b) Grant membership (Step II): Since only MGS has the group's private key, MGS decrypts the packet and replies with the membership grant, which consists of the group parameter set $\widetilde{m}_k$, RMK_k, $G_{k+1}(.)$, $g_{k+1}(.)$, c_l, d_l and at the same time, sends the $g_{k+1}(.)$ to the entire (current) group members to support forward secrecy. For source authentication purposes, MGS signs this packet with its own node private key.

c) Acknowledgement of membership (Step III): Firstly, MGR verifies the signature, and then accept the information and joins the group if it is a valid one. Then, MGR sends an acknowledgement to the source notifying the source that MGR has successfully joined the group.

3) Key refreshment process: The reason for the key refreshments could be different for the multicasting keys. For instance, joining and leaving a member initiates demand to system refresh the keys in order to support forward and backward secrecy, as well as providing multicast overall key secrecy. However, we suggest a similar algorithms in both mentioned demands. To be more precise, multicast group has a similar timers set by the system administrator as per group establishment purposes and application requirements. Referring to our unicast refreshment processes, we only describe the multicasting's relevant points.

- For multicasting forward and backward secrecy concerning nodes join/leave situation, we follow a short term refreshment process.
- Recalling our aforementioned discussion, MGS is in charge of $g_k(.)$ generation and distribution (a similar short term key refreshment), proceeding from the (k, l) to $(k + 1, l)$ state.
- MGS is in charge of $\widetilde{m}_k$ set up values c_l & d_l, addressing a similar medium term key refreshment, moving from the (k, l) to $(k, l + 1)$ state.
- SAS is in charge of similar long term key refreshment process, moving from the (k, l) to $(0, 0)$ state. SAS provides appropriate parameters including keys to the MGS, and then MGS unicasts them to the members utilizing their unicast public/private key system.

C. Broadcast key mechanism

Refer to our unicast medium term key refreshment process, we apply the system original $Hash(.)$ function to the SAS's public key to obtain a symmetric key. Since the SAS' public key is dynamic and changes periodically according to the $f_i(.)$ function, only the system members would have the live value of the SAS's public key.

V. Evaluation

In this section, we present our mechanism evaluation using Automated Validation of Internet Security Protocols and Application (AVISPA) security analyzer. Furthermore, we review the adversary model, his/er interests and capabilities to attack the system. Then we review the system security against attacks. At the end, we study our system security overhead cost and improvements, concerning key management.

A. Authentication scheme evaluation using AVISPA

We use the original model already existed in the AVISPA library and then modified it based on our needs respect to our mechanism, which part of the codes are shown by Fig. 8:

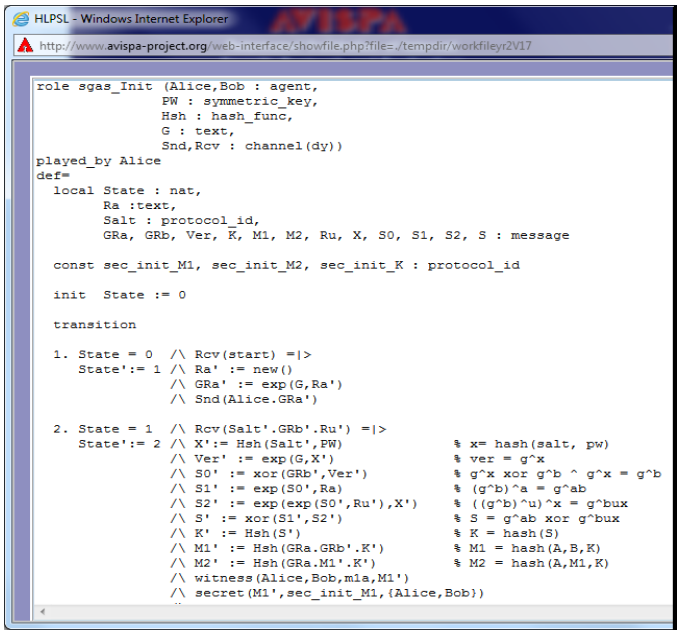

Fig. 8. AVISPA code

Since AVISPA does not support the arithmetic calculation, we are ended with xor & exp (raise to a power) operators besides other security functions. Therefore, we used xor instead of " + & − " (addition and subtraction) that we required for our authentication, which is shown by the figure. The result of the evaluation presented in Fig. 9) shows the mechanism is secure/safe. To be more precise, the symmetric key that we prepare at the end of our authentication to be used for sending the system parameter by SAS to SM is a valid and safe key. The system parameters are consists of the PRNG and its setup values a & b, as well as SM's seed private key.

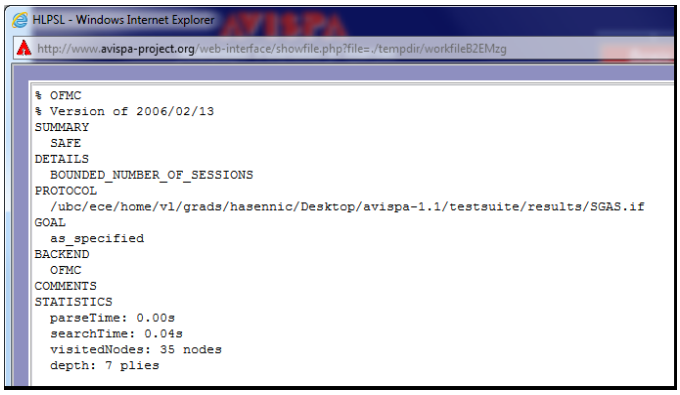

Fig. 9. AVISPA evaluation result

B. Adversary analysis

The adversary objective to attack the system can be defined as gaining access to the system resources, for instance SAS or any of the SMs. Prior to the attack, s/he may have different situation. For instance, we consider two overall situations: s/he does not have (first case) or has (second case) a full control on one of the SM devices including the SM's password.

1) First case: In this case, all adversary may know or be able to find, is the server ID. Thus, if s/he wants to attack the server, like performing a Denial of Service (DoS) attack, s/he can send several requests (Fig. 3) to the server for the authentication. As soon as server receives the requests, will check the database for the $(ver, salt)$ pair associated with every request. If SAS does not find any, won't take any further actions and assumes it is an attack. If s/he sends the request with valid ID & SN (can be stolen from a SM), SAS may find the $(ver, salt)$ values and proceeds by sending the response and goes to the next step of the scheme. Since the adversary does not have the appropriate password, s/he is not able to obtain the key and decrypt the packet. However, SAS is remained by an open session. Note that SAS sends a time stamp (TS_1) among other information. SAS can close the session if the appropriate acknowledge is not receiving by a time period. Furthermore, even for preventing attack up to this point, SAS can allow only a limit number of the authentication request. So, sending a high volume of requests does not cause any issue on the server.

The adversary may try a replay attack by forwarding any other previous SM's acknowledgement answer to the server. This solution does not help the adversary since the acknowledgement should encrypted and signed utilizing the valid and appropriate system public and private keys. Also, the acknowledgement consists of the time stamp and ID of SM, which is not the valid one for the adversary's authentication session.

The next option for the adversary is performing a brute-force attack and gets access to the encrypted packet. Brute-force attack takes time. If it takes more than session expiry time, the attack is not causing any issue. In the best situation, s/he can move to the on-line dictionary attack to speed up, or performs an off-line dictionary attack and find the session key, and finally obtain an expired private key for a not valid SM. However, s/he would gain access to the system parameters, and if SAS has not run LTR process yet, s/he can keep going and makes the system parameters valid and fresh. At the end of the day and using any of the aforementioned exercises, the adversary is not able to compromise the server, since s/he can only communicate with others and if others sends information to her/im, s/he is able to decrypt the packets. Furthermore, since our mechanism follows SRP protocol technique (hash function), our authentication provides forward secrecy and the adversary is not able to find out the original password.

To perform a Man-In-The-Middle (MIMT) attack as another option for the adversary, s/he may receive the first packet and change the value of A. However, s/he won't be able to decrypt the second packet coming from the server, because s/he requires the victim's password to obtain the symmetric key K.

The other option is compromising the server by an attack like social engineering. Attacking a node with this attack is almost equivalent to gaining control on the SM, which we will study it as part of the second case. So, attacking the server does not give the adversary access to the SMs' passwords since we only keep the verifier (and salt). However, if SAS records and keeps the nodes' private key (to be more precise, the nodes' private key seed value), s/he will have the entire SMs' private keys. This attacks is a costly attack on the server and unfortunately works in almost all of the situations. If SAS only generates the private keys and not keeping them, in some extent will prevent the attack harms on the previous generated keys. However, the adversary will be able to attack the new SMs. The best solution in this situation is managing the server security very well. For instance by changing the server password more often or similar solutions.

2) Second case: In this case, an adversary has full control on one of the nodes (SM). Hence, s/he can proceed the authentication and receive his/er own private key from the server along with the system parameters, like PRNG and system's functions. In this case, the adversary may communicate with others, including SAS, without any issue. However, it does help her/im to gain access to the server. General speaking, being in this case does not help an adversary to improve she/his chance to attack, referring to the aforementioned discussion in the first case. For instance, s/he can run a brute-force attack by having a valid private key and communicate with others to brute-force their private key. Although refer to our above discussion, SAS may change the keys before the attacker finds a valid one. In this case, off-line dictionary can work because the adversary has the system parameters and like $f_i(.)$ and PRNG and can find the live private key. However just by performing one LTR process by SAS, system prevents the adversary of a successful attack.

C. Other security characteristics

Recall our discussion in Section III, a mutual authentication is performed since SAS needs to know the password verifier, and on the other side, SM needs to know the password. Both ends require one of these values to calculate the session key. In terms of attacks resilience, we refer to study in previous subsection, about the most well-known attacks such as Brute-force, DoS, Replay on-line & off-line dictionary and MITM attack, which covers part of the attacks resilient summary presented by TABLE I. We also refer to the above section about the social engineering attack that may work partially on the server; however, compromising a SM does not help the adversary to attack the whole system.

Unknown key-share attack: The second packet of the authentication scheme presented in Fig. 3 is encrypted by symmetric key K. Encryption of this packet by SAS shows SAS has the key, and decryption the packet by SM and acknowledging the SAS proves that SM has the key as well.

Compromised impression resilience: Referring to our analysis at the first of this section, finding any SM's private

TABLE I
SUMMARY OF ATTACKS STUDY

Attack	Resilience
Social engineering attack	✔ & ✖
Brute-force attack	✔
Replay attack	✔
DoS attack	✔
MITM attack	✔
On-line dictionary attack	✔
Off-line dictionary attack	✔
Unknown key share attack	✔
Compromised impression attack	✔
Denning-Sacco attack	✔
Key privacy & insider attack	✔
Ephemeral key compromise impersonation	✔

key does not help an intruder to obtain any other node or SAS's private key .

Denning-Sacco attack resilience: If an intruder somehow finds a symmetric key in middle of the authentication, since the key is the product of a hash, which is a one-way function, s/he would not able to find the original password or even the verifier. Furthermore, finding a private key does not help the adversary to find a symmetric key of the authentication session.

privacy & insider attack resilience: Since we are using the PKI, each key is only known by the owner (and maybe server). Other nodes only know the SM's public key, which in fact required by them to communicate to the SM. Even if other nodes in between relay the packets, since the packets are encrypted and signed, they cannot have access to the SM's private key.

Ephemeral key compromise impersonation: Let us assume an adversary perform an off-line dictionary attack or brute-force or even social engineering and obtains a SM's password. Because the password is only one of the required values for the key formation, the adversary still is not able to find the session key, or the private key.

D. Communication and network performance analysis

In our design, we took advantage of the SPR, PKI and IBC approaches. Each one brings some benefits to the mechanism. Beside, our improvements on each of those have brought more benefits to the system.

Firstly, we have improvement on the required packet delivery in our authentication scheme. To be more precise, we reduce the number of packet delivery from six to three packets. Furthermore and during this three packets handling, we manage delivery of the entire system parameters as well as the new SM's private key. Our design shows that the authentication scheme presented here is fast and secure.

Secondly, implementing private key infrastructure in a distributed system causes providing a symmetric key between every two nodes that may need to communicate to each other. From other point of view, if the number of the nodes that wants to have communication with a single node increases, it causes the node to keep and manage a high number of keys (one per each), which is the case in the SG system. However, PKI requires one pair key per entity in spite of a higher key size. In fact, while a node has its own private/public pair key, it is sufficient for the node and others to perform a secure communication.

Also, IBC improves the public key distribution overhead cost in a PBI, which we are taking advantage of this technique in our design. Furthermore, we have designed and utilize an improved version of the IBC so-called EIBC for this model. The most important benefit of using EIBC in this design, and in any other system, is private key distribution and refreshment overhead cost improvement. In EIBC, in most of the time the PKG entity only broadcasts a packet instead of unicasting it that causes the system overhead cost improvement. Indeed, in two (STR and MTR) of three key refreshment processes the broadcasting is being used, also the unicast (LTR) model is required less often comparing to STR and MTR processes.

Last but not the least, our mechanism can be easily implemented in any system and platform. Since nodes only require to have their own private key, and only know with whom they want to have communication with, expanding this model does not require further action. Based on the nodes population and application that are going to be run on the system, a system administrator can tune up the security and overhead utilizing the timers value as well as size of the key. Furthermore and referring to our EIBC design in [14], the system administrator even can turn any of the features off, like PRNG. All s/he requires to do is for instance sets $a = 0$ & $b = 1$. On the other hand, if s/he wants to turn the periodic distribution function $f_i(.)$ off, s/he can set $f_i(x) = x$. This flexibilities makes our mechanisms applicable to the variety of systems and platforms.

VI. CONCLUSION

In this paper we stated our tailored mutual authentication and key management mechanisms for the smart grid system. The proposed flexible design addresses the system's required security aspects, and at the same time, handles the process in en efficient fashion. The saving resources caused by our mechanism can be used to handle more data and/or to increase the security of the system by refreshing the keys more often. Consequently, more key refreshment can give us this opportunity to use smaller key size, which again brings more improvement on the entities' resources.

To improve and develop our design more, we will apply the Elliptic Cure Cryptography (ECC) approach to our mechanism. ECC-based approach decreases the size of the keys without decreasing the system security level. Furthermore, we will adapt this design to the inside customer domain in smart grid system.

ACKNOWLEDGEMENT

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada through grant STPGP 396838.

REFERENCES

[1] NIST Smart Grid, CSWG, *Introduction to NISTIR 7628 Guidelines for Smart Grid Cyber Security*, Published by www.nist.gov/smartgrid, Sept. 2010.

[2] P. McDaniel and S. McLaughlin, "Security and Privacy Challenges in the Smart Grid", *IEEE Security Privacy,* Vol. 7, No. 3, Pages 75-77, May/Jun. 2009.
[3] H. Nicanfar, P. Jokar and V. C.M. Leung, "Smart Grid Authentication and Key Management for Unicast and Multicast Communications", in Proc. IEEE PES ISGT, Perth, Australia, Nov. 2011.
[4] W. Diffie and M.E. Hellman, "New direction in cryptography", *IEEE Transaction on Information Theory,* Vol IT-11, Pages 644-654, Nov. 1976.
[5] S. M. Bellovin and M. Merritt, "EKE: Password-based protocols secure against dictionary attacks", in Proc. Research in Security and Privacy, Oakland, CA, May 1992.
[6] D.H. Seo and P. Sweeney, *Simple Authenticated Key Agreement Algorithm*, an Electronic Letter, pp 1073-1074, Vol. 35, Issue 13, June 1999.
[7] H. Yeh and H. Sun, "Simple Authenticated Key Agreement Protocol Resistant to Password Guessing *ACM SIGOPS Operating Systems Review,* Vol. 36, Issue 4, Pages 14-22, Oct. 2002.
[8] W. Juang, S. Chen and H. Liaw, "Robust and Efficient Password-Authenticated Key Agreement Using Smart Cards", *IEEE Transaction on Industrial Electronics,* Vol. 55, No. 6, Pages 2551-2556, Jun. 2008.
[9] D. Xiao-fei, M. Chuan-gui and C. Qing-feng, "Password Authenticated Key Exchange Protocol with Stronger Security", in Proc. ETCS, Wuhan, Hubei China, Mar. 2009.
[10] L. Liu and Z. Cao, "Improvement of One Password-Based Authenticated Key Exchange Protocol", in Proc. ISISE, Wuhan, Hubei China, Mar. 2009.
[11] M. B. MBarka, L. Granboulan and F. Krief, "Using OTP with PAKE: An Optimized Implementation of a Synchronization Window", in Proc. IFIP NTMS, Paris, France, Feb. 2010.
[12] Z. Zhang and Q. Zhang, "Verifier-based password authenticated key exchange protocol via elliptic in Proc. ICISIT, Beijing, China, Dec. 2010.
[13] T. Wu, "The Secure Remote Password protocol", *in Internet Society Network and Distributed Systems Security Symposium (NDSS)* pages 97111, San Diego, CA, Mar. 1998.
[14] H. Nicanfar and V. C.M. Leung, "EIBC: Enhanced Identity-Based Cryptography, a Conceptual Design", in Proc. IEEE SysCon, Vancouver, BC, Mar. 2012.
[15] J. Wang and V. C. M. Leung, "A Survey of Technical Requirements and Consumer Application Standards for IP-based Smart Grid AMI Network", in Proc. ICOIN, Kuala Lumpur, Malaysia, Jan. 2011.
[16] H. Gharavi and B. Hu, "Multigate Communication Network for Smart Grid", *Proceedings of the IEEE*, Vol. 99, No. 6, pages 1028-1045, Jun. 2011.

Hasen Nicanfar (GS'11) received his BASc. degree in electrical engineering from Sharif University of Technology in 1993, and his MASc. degree in Computer Networks (electrical and computer engineering) from Ryerson University in 2011. In 2011, he has started his PhD program at graduate school of the University of British Columbia in electrical and computer engineering.

From 1993 to 2010, he served market in different positions including IT & ERP manager & consultant, project manager, production engineer & manager and business & system analyst.

Currently, he holds the position of research assistant in WiNMoS Lab. His research interests are in the areas of system & network design and management, security and privacy, cryptography, routing protocol design for wireless communication and Smart Grid system.