Authentication and Key Management for Advanced Metering Infrastructures Utilizing Physically Unclonable Functions

Mohamed Nabeel, Sam Kerr, Xiaoyu Ding, Elisa Bertino

Department of Computer Science, Purdue University
305 N. University Street, West Lafayette, IN, 47906, USA
{nabeel, stkerr, ding55, bertino}@cs.purdue.edu

Abstract— **Conventional utility meters are increasingly being replaced with smart meters as smart meter based AMIs (Advanced Metering Infrastructures) provide many benefits over conventional power infrastrucutures. However, security issues pertaining to the data transmission between smart meters and utility servers have been a major concern. With large scale AMI deployments, addressing these issues is challenging. In particular, as data travels through several networks, secure end-to-end communication based on strong authentication mechanisms and a robust and scalable key management schemes are crucial for assuring the confidentiality and the integrity of this data. In this paper, we propose an approach based on PUF (physically unclonable function) technology for providing strong hardware based authentication of smart meters and efficient key management to assure the confidentiality and integrity of messages exchanged between smart meters and the utility. Our approach does not require modifications to the existing smart meter communication. We have developed a proof-of-concept implementation of the proposed approach which is also briefly discussed in the paper.**

I. Introduction

End-to-end secure communication between utility servers and smart meters is a key requirement for the overall security of the Advanced Metering Infrastructure (AMI). The messages exchanged between the utility servers and smart meters travel through multiple hops before reaching the destination. Collector nodes and, sometimes smart meters act as routing nodes. In other words, a message between the utility server and a smart meter may travel through one or more collector nodes and other smart meters. Different hops use different communication protocols. For example, the hop between the utility servers and collectors may use a 3G network whereas the hop between the collectors and smart meters may use a radio link. Even though most of these communication protocols provide link level security, this is not sufficient to protect messages traveling between the utility servers and smart meters as compromised/malicious intermediate nodes may not be trusted for the confidentiality and integrity of the messages. Therefore end-to-end message level security is essential to protect messages from atttacks carried through the communication channels and intermediate nodes.

In initial AMI deployments, the data formats, security measures, and protocols for smart meter security were proprietary. However, because of the difficulties in achieving secure communication when proprietary methods are adopted and because of interoperability issues, the industry is moving towards a common standard developed by ANSI and known as the ANSI C12 standard [1]. ANSI C12 standard compliant smart meters are required to store a symmetric key used to encrypt and create message authentication codes (MAC), and the passwords used to provide different access privileges in specific tables in the smart meter. These keys and passwords must in turn be protected by a secure mechanism. Approaches by which data is encrypted with a set of keys which are in turn encrypted with other keys are very common; for exampe such an approach is used for SQL Server Database Encryption. In the AMI, the application of such an approach requires a scalable, efficient, and robust key management scheme able to support very large number of smart meters and also able to support smart meter authentication. In the absence of a strong authentication mechanism, smart meters are vulnerable to man-in-the-middle attacks. An impostor may persuade the utility server that it is communicating with a valid smart meter and may cause damages.

In this work, we address the problem of designing a key management scheme able to achieve secure end-to-end communication in the AMI. Specifically, our solution provides an efficient approach to manage keys and a strong authentication mechanism. Our solution is based on the use of PUF (physically unclonable function) devices which are inexpensive to manufacture and provide a hardware based strong authentication mechanism resistant to spoofing attacks. We utilize the PUF devices' hardware based one-way function to generate and re-generate the symmetric keys and access level passwords for smart meters. The PUF based secret generation mechanism provides strong protection against key leakage as the master key is never stored in memory.

The rest of the paper is organized as follows. Section II provides an overview of the AMI and smart meters. Section III describes the key building blocks used in our approach. Section IV describes our overall scheme. Section V provides an overview a proof-of-concept implementation we develop based on the proposed approach. Section VI briefly discusses the related work, before concluding in Section VII.

II. Background

In this section we provide a high-level overview of the AMI and smart meters. This overview focuses on the aspects that are relevant for the presentation of our protocols and therefore we abstract away several components and details.

A. Advanced Metering Infrastructure

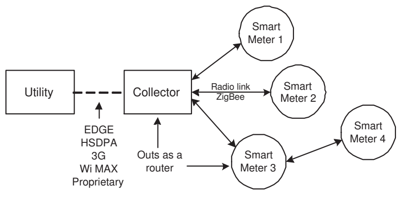

Fig. 1. A simplified AMI

The Advanced Metering Infrastructure (AMI) consists of four main components: the utility company (utility, for short), data collectors, often located in the neighborhood, smart meters, and the home or office appliances. The communication between smart meters and appliances can use several communication protocols such as ZigBee, Wi-Fi, and Ethernet. In this work, we focus only on the communication between smart meters and the utility (see Figure 1). The messages between these two components are transmitted across multiple networks. These messages go through one or more collectors and possibly through other smart meters which act as routing nodes. Long distance communication protocols such as 3G, EDGE or HSPDA, are used between the utility and collectors. Short distance communication protocols such as radio links are used between collectors and smart meters. Different network segments use different communication protocols and have their own transport level security.

B. Smart Meters

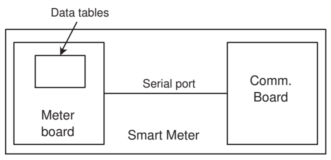

Fig. 2. Two components of a smart meter

An ANSI C12 standard compliant smart meter consists of two internal components (see Figure 2): the meter board and the communication board connected through a serial port. The meter board contains a set of tables storing various information including keys and passwords used for secure communication and privilege levels. It also performs power consumption measurements. The communication board is responsible for communications with the outside nodes such as collectors, other smart meters, or home/office appliances, and performing any required computation. Using an interrupt based mechanism, the communication board fetches data and other necessary information such as keys from the meter board whenever it needs to send data to the utility. A common scenario is to have the firmware in the communication board to periodically, typically every 15 minutes, fetch meter readings from the meter board and send it to the utility.

C. The ANSI C12 Standard

Security protocols for communication between smart meters and the utility are defined by the ANSI C12 standards. According to the ANSI C12.22 standard, smart meters are recommended to use a authenticated symmetric encryption algorithm to assure confidentiality and a message authentication code (MAC) to assure authenticity and integrity. The authenticated encryption algorithm specified uses a variant of EAX mode [2] called *EAX'* along with AES for encryption and a symmetric block cipher, called *CMAC* for authentication and message integrity.

The ANSI C12.19 standard outlines the format and purpose of the tables available in the meter board. For example, table 42 (the *password table*) stores passwords for the different privilege levels, and table 45 (the *the key table*) stores the keys for encryption and authentication. Smart meters provide an optical port through which an operator can execute various commands based on the operator's security level. ANSI C12.18 standard defines six security levels, $L0$ to $L5$, with different privileges and with the highest privilege being $L5$. The security level $L0$ requires no password. For each of other security levels, the password table contains a password which is initially set at the time of manufacture or installation. The operator's security level is equal to the password entered to gain access to the smart meter through the optical port. Similar to the concept of multi-level security (MLS), the operator can execute any command that requires a security level less than or equal to its security level.

III. Building Blocks

In this section, we give an overview of the key hardware and cryptographic constructs we use in our scheme.

A. Physically Unclonable Functions

PUFs (Physically unclonable functions) are one-way functions that are embodied in a physical structure [3]. A PUF takes an input challenge $C_i \in C$, where C is the set of all possible challenges and produces a response $R_i \in R$, where R is the set of all possible responses. Mathematically, a PUF can be represented as a function $PUF : C \rightarrow R$. The function is based on the intrinsic randomness that exists in the integrated circuit used to generate the response and cannot be controlled. As PUF relies on the random variations during the integrated circuit fabrication process, even two PUFs with the same layout results in two different functions. In other words, it is physically impossible to make two PUFs behave identically.

We exploit the following characteristics of PUFs in our work.

- Given a PUF device, and a challenge C_i as input produces approximately the same response R_i. In practice, error correction codes, such as Reed-Solomon, are used to remove the noise from the response and make it stable and identical.
- Given a PUF device, and a response R_i, it is difficult to find the corresponding challenge C_i.
- Given a PUF device, two different challenges $C_1 \neq C_2$ produces two different responses $R_1 \neq R_2$.
- Given a challenge C_i, two different PUFs produces two different responses $R_i \neq R_i'$.

As introduced in previous research [4], we incorporate the PUF device in a feedback loop for a system-on-chip (SoC) design. Figure 3 shows a high-level view of the PUF SoC. The *ECU* (Error Correcting Unit) performs error correction on the PUF response with noise so that, in a real setting, every time the same challenge is given the PUF along with the ECU produces the same response. The *CC* (Cryptographic Core) is a stand-alone hardware component that provides cryptographic services to the communication board of the smart meter. The operations of the CC depends on the required functionality. In our approach, Section IV, the CC implements a secure hashing, and encryption operations for use by smart meters. The *Reg* (register) stores the initial challenge and then later get overwritten by the subsequent responses from PUF-ECU component. Notice that PUF-ECU-Reg forms a feedback loop. We utilize this feedback mechanism to generate keys chaining responses in a sequence. This technique is similar to using hash chains to generate one-time keys where a cryptographic hash function is applied repeatedly to obtain a new key. An adversary cannot derive the old keys from the new key due to the pre-image resistance property of the hash function. However, PUF based implementations are known to be more secure than software based implementations [3].

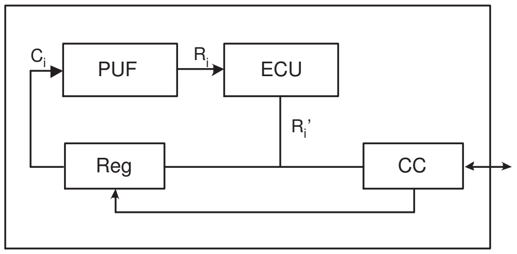

Fig. 3. PUF SoC design

B. Pedersen commitment

A cryptographic "commitment" is a piece of information that allows one to commit to a value while keeping it hidden, and preserving the ability to reveal the value at a later time. The *Pedersen commitment* [5] is an unconditionally hiding and computationally binding commitment scheme which is based on the intractability of the discrete logarithm problem.

Definition 1 (Pedersen Commitment): It consists of the following three algorithms.

Setup A trusted third party T chooses a multiplicatively written finite cyclic group G of large prime order $\mathfrak{p}$ so that the computational Diffie-Hellman problem is hard in G.[1] T chooses two generators g and h of G such that it is hard to find the discrete logarithm of h with respect to g, i.e., an integer x such that $h = g^x$. It is not required that T know the secret number x. T publishes $(G, \mathfrak{p}, g, h)$ as the system parameters.

Commit The domain of committed values is the finite field $\mathbb{F}_\mathfrak{p}$ of $\mathfrak{p}$ elements, which can be represented as the set of integers $\mathbb{F}_\mathfrak{p} = \{0, 1, \ldots, \mathfrak{p} - 1\}$. For a party U to commit a value $\alpha \in \mathbb{F}_\mathfrak{p}$, U chooses $\beta \in \mathbb{F}_\mathfrak{p}$ at random, and computes the commitment $c = g^\alpha h^\beta \in G$.

Open U shows the values α and β to open a commitment c. The verifier checks whether $c = g^\alpha h^\beta$.

C. Zero-knowledge proof of knowledge (Schnorr's scheme)

The *zero-knowledge proof of knowledge (ZKPK)* protocol used in this paper can be viewed a natural extension of Schnorr's scheme [6]. In our proposed approach, we use ZKPK as a secure means of smart meter authentication to the utility.

As in the case of the Pedersen commitment scheme, a trusted party T generates public parameters $G, \mathfrak{p}, g, h$. A Prover which holds private knowledge of values α and β can convince a $\mathsf{Verifier}$ that Prover can open the Pedersen commitment $c = g^\alpha h^\beta$ as follows.

1) Prover randomly chooses $y, s \in \mathbb{F}_\mathfrak{p}^*$, and sends $\mathsf{Verifier}$ the element $d = g^y h^s \in G$.
2) $\mathsf{Verifier}$ picks a random value $e \in \mathbb{F}_\mathfrak{p}^*$, and sends e as a challenge to Prover.
3) Prover sends $u = y + e\alpha, v = s + e\beta$, both in $\mathbb{F}_\mathfrak{p}$, to $\mathsf{Verifier}$.
4) $\mathsf{Verifier}$ accepts the proof if and only if $g^u h^v = d \cdot c^e$ in G.

IV. Our Scheme

Figure 4 shows the overall block diagram of smart meters we utilize. As it can be seen from the figure, without modifying the existing communication model of smart meters, we integrate a PUF SoC with each smart meter. PUF SoC modules act similar to a TPM but it neither utilizes public and private key pair nor store any master keys.

In our threat model, we assume that the intermediate nodes in the AMI and the network is not trusted for the confidentiality and the integrity of the messages. Further, smart meters are susceptible to spoofing and key invasion attacks [7].

[1]For a multiplicatively written cyclic group G of order q, with a generator $g \in G$, the *Computational Diffie-Hellman problem (CDH)* is the following problem: Given g^a and g^b for randomly-chosen secret $a, b \in \{0, \ldots, q-1\}$, compute g^{ab}.

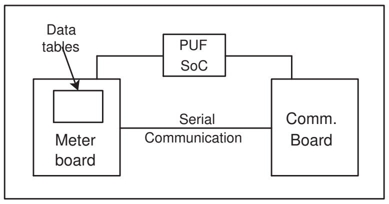

Fig. 4. Smart Meter with a PUF SoC

A. Procedures and Algorithms

Our secure AMI system undergoes the following procedures and algorithms in order to provide end-to-end security between smart meters and the utility.

Initialization:

The utility initializes the system by executing the **Setup** of the Pedersen Commitment scheme. It defines the cryptographic hash functions: $H_1 : \{0,1\}^t \to \{0,1\}^m$, $H_2 : \{0,1\}^t \to \mathbb{F}_p$, and $H_1 : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}^m$, where t is the length of PUF responses and m is a security parameter. These cryptographic functions are implemented in the CC of the PUF SoC. The utility also defines two challenges C_a and C_k in order to generate responses using each smart meter PUF device for authentication and smart meter secret generation respectively. Let the AMI consists of n smart meters, M_i, $i = 1, 2, \cdots, n$.

Smart Meter Registration:

Using out-of-band communication, preferably before the installation of smart meters, the utility executes the following procedure with each smart meter M_i. We assume that the utility is in physical contact with M_i so that the communication channel is private and authentic.

The utility generates the PUF responses R^i_a and R^i_k giving C_a and C_k as input. It computes the Pedersen commitment of R^i_a as describe in Section III-B. It stores the Pedersen commitment of the R^i_a, denoted by com_i and the hashed value $H_1(R^i_k)$, denoted by s_i, in its database. Since $R^i_a \in \{0,1\}^t$, before creating the commitment, R^i_a is first converted to a unique value in $\mathbb{F}_p$ by applying H_2. By storing only the commitments, com_i, $i = 1, 2, \cdots, n$, the actual responses, R^i_a, $i = 1, 2, \cdots, n$, are never leaked even if the utility database is compromised. The hashed value of the responses R^i_k are taken as the secrets s_i in order to prevent leaking challenge-response pairs and guarantee strong secrets.

Refreshing PUF Secrets:

In order to improve the security, the com_i and s_i values are updated periodically. We utilize the feedback loop of the PUF SoC to perform the update and use the previous response to obtain a new response. As shown in Algorithm 1, in order to update s_i of the smart meter M_i, the utility sends the tuple (C_a, r), where r is the number iterations to be performed at the PUF SoC to generate the new R^i_k. Note that the current R^i_k is produced with $r-1$ iterations. Let $s^{(r)}_i$ be the secret produced by executing the PUF's feedback loop r times and applying the hash function H_1. The CC of the PUF SoC computes both $s^{(r-1)}_i$ and $s^{(r)}_i$, encrypts the latter with the former and sends it to the utility. The utility can decrypt the message using the current secret for the smart meter and update the s_i and r values in its database. A similar approach is followed to refresh com_i. $\mathsf{com}^{(r)}_i$ is the commitment of the value $R^i_a(r)$ produced by executing the PUF's feedback loop r times and applying the hash function H_2.

Algorithm 1 PUF Feedback loop to refresh secrets

```
1: Refresh($C_i, r$)
2: rounds = 0
3: while ++rounds $\leq r$ do
4:     $R^{i\,\prime}_k \leftarrow$ PUF($C_i$)
5:     $R^i_k \leftarrow$ error-correction($R^{i\,\prime}_k$)
6:     if rounds $== r - 1$ then
7:         $R^i_k(r-1) = H_1(R^i_k)$
8:     end if
9:     $C_i \leftarrow R^i_k$
10: end while
11: $s_i(r) = H_1(R^i_k)$
12: Return $\mathrm{E}_{s^{(r-1)}_i}(s^{(r)}_i)$
```

Smart Meter Authentication:

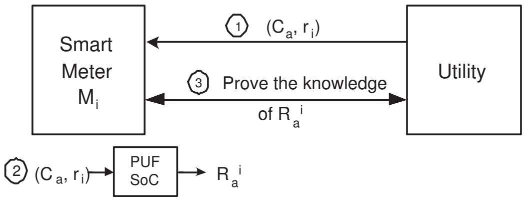

Fig. 5. Smart Meter Authentication

Now we explain how the utility authenticates smart meters before initiating any further communication. Recall that during the registration, the utility stores a commitment com_i for each smart meter M_i. As shown in Figure 5, the utility sends the tuple (C_a, r) to the smart meter M_i. M_i uses C_a as the initial challenge to the PUF and generates the the response $R^i_a(r)$ by executing the feedback loop r times and applying H_2 on the last response. M_i provides a ZKPK of the ability to open the commitment $\mathsf{com}^{(r)}_i$ using the algorithm described in Section III-C. Notice that the ZKPK of the commitment does not reveal the PUF response $R^i_a(r)$ to the utility and, in fact, the response never leaves out of the PUF SoC. Since PUF produces volatile responses, it is extremely difficult to carry out invasive attacks such as extracting the responses used in the ZKPK. Therefore, the same challenge-response pair can be used authenticate smart meters multiple times. Further, as different PUF devices produce unique responses, it is very difficult to clone a PUF and launch spoofing attacks. Therefore, PUF integrated smart meters provide strong authentication.

Smart Meter Key/Password Generation and Re-Key:

Recall that in an ANSI compliant smart meter, symmetric keys and access level passwords are stored in two specific tables in the meter board of the smart meter. A symmetric key in the key table is used to provide confidentiality and integrity to the messages communicated between the smart meter and the utility. PUF SoC is utilized to generate the symmetric key and store it in the key table. For the smart meter M_i, given (C_k, r), PUF SoC generates s_i as explained earlier, selects a random label l_i^0 and generates an m-bit random value k_i using $H_3(s_i, l_i^0)$. k_i is used as the EAX' encryption/decryption and CMAC key and stored in the key table of the meter board. Similarly, five more random labels l_i^j, $j = 1, 2, \cdots, 5$ are selected and an access level password, pwd_i^j $(= H_3(s_i, t_i^j))$, is created for each access level Lj, $j = 1, 2, \cdots, 5$. The random labels l_i^j, $j = 0, 1, \cdots, 5$ are sent to the utility. The utility can also generate the same key k_i and the access level passwords as it knows s_i and t_i^j values. In order to update a smart meter symmetric key and access level passwords, that is to re-key, the same procedure is repeated with a new set of random labels. Note that each smart meter has a unique key as it is generated from a unique response of PUF. It makes revocation and addition of smart meters to the AMI easy.

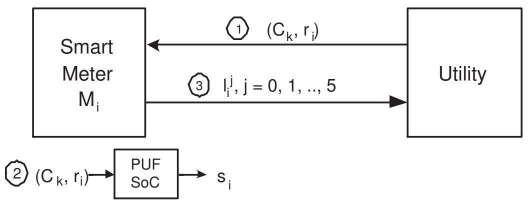

Fig. 6. Re-keying a smart meter

Compromise of the smart meter keys k_j's or access level passwords does not leak the PUF keys s_i's due to the pre-image resistance property of H_3. Therefore, the above re-key approach can update smart meter keys even if the current smart meter keys are compromised.

Adding/Revoking Smart Meters:

As the secret key for each smart meter is generated independently, adding a new smart meter does not affect the secret keys generated for the existing smart meters. The utility executes the registration and key/password generation procedures. In order to revoke a smart meter, the utility simply removes the corresponding record from its database.

As shown earlier in this section, our secret generation procedures use simple operations such as PUF challenge-repsponse function, error correction function and cryptographic hashing, and therefore our scheme can easily scale to large number of smart meters.

B. *Secure End-to-End Messaging*

After generating keys/passwords or re-keying all smart meters, the utility database looks like in Table I. The notations and symbols used in the table are defined earlier in this section.

TABLE I
DATABASE AT THE UTILITY

Meter	$r\ (C_a)$	s	$r\ (C_k)$	com	l^0	$\ldots$	l^5
M_1	r_a^1	s_1	r_k^1	com$_1$	l_1^0	$\ldots$	l_1^5
M_2	r_a^2	s_2	r_k^2	com$_2$	l_2^0	$\ldots$	l_2^5
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
M_n	r_a^n	s_n	r_k^n	com$_n$	l_n^0	$\ldots$	l_n^5

Whenever the utility wants to communicate with a smart meter, M_i, it fetches the record corresponding to M_i in Table I and executes the authentication procedure described earlier in this section using the parameters C_a, r_a^i and com$_i$. M_i can successfully authenticate itself only if it can dynamically generate the response $R_a^i(r_i)$ using the integrated PUF SoC. After successfully authenticating M_i, the utility sends authenticated encrypted messages using the AES algorithm with the EAX' mode. Notice that the utility derives the smart meter secret key $k_i^{r_i}$ from s_i and t_i^0 values in its database. Since only M_i can derive this key, unauthorized access or modification of the messages by intermediate nodes, such as collectors or other smart meters, is prevented. Similarly, smart meters send authenticated encrypted messages, such as meter readings, to the utility by fetching the symmetric key from its key table and encrypting using the EAX' cipher.

C. *Supporting Secure Broadcast Messaging*

In addition to unicast messages, the utility at times need to broadcast messages, such as firmware updates and control messages, to a set of smart meters.

A naive approach to broadcast a message is to choose a random session key, encrypt the message using the session key, and finally encrypt the session key using each of the smart meter secret keys. However, such an approach is inefficient, as it requires multiple encryptions and decryptions.

A better approach is to utilize a broadcast group key management scheme [8]. Using smart meter keys as secrets, such an approach can efficiently and securely broadcast a message to any subset of smart meters. Due to space limitation, we omit the details.

V. IMPLEMENTATION

To demonstrate a proof-of-concept for our PUF integrated smart meters for secure end-to-end communication, we developed a prototype implementation. Figure 7 shows the layout of our implementation setup. We utilized an ANSI standard compliant smart meter produced by Meter-ON which is usually used for sub-metering. Additionally, the communication board of the smart meter provides a way to communicate through Wi-Fi. We implemented the PUF feedback loop using the Xilink's Spartan-6 FPGA board which is connected to a PC through a serial port and the remaining functions of error correction and cryptographic operations in the PC. The PC has an Intel Core i5-2430 2.4Gz processor and runs a 64-bit Linux kernel 2.6.32. The PC is connected to the smart meter through the ANSI C12.19 optical port. The utility communicates with the smart meter through the Wi-Fi link. We used OpenSSL for the cryptographic functions. Each execution of the PUF produced a 32-bit output. We observed that our implementation executed the PUF in 2.4 ms and the SHA-1 on the 32 bit PUF response in 0.2 ms on average. We report the performance analysis of our prototype in the extended version of our paper.

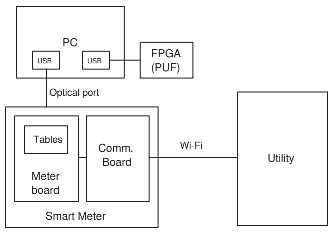

Fig. 7. Proof-of-concept Implementation Setup

VI. Related Work

In this section, we briefly compare previous research work on PUFs and smart grid security.

Physically Unclonable Functions:

PUFs have been applied in many other applications. Gassend et al. [3] analyzes the security of PUF constructions and apply it to several applications including smartcard authentication and certified execution. Suh et al. [9] shows how to enable low-cost authentication of integrated circuits and volatile secrets for cryptographic operations. Atallah et al. [10] utilizes PUF technology to bind software to native hardware in a virtualization environment. Kirkpatrick et al. [4] presents a hardware based approach to create read once keys which, for example, could be applied to construct one-time programs and perform program obfuscation.

Smart Grid Security:

Fouda et al. [11] proposes an approach to authenticate smart grid nodes based on Diffie-Hellman key exchange protocol. Li et. al [12] proposes multicast authentication using one-time signatures. There has been research efforts to support key management schemes for secure unicast communication [13], [14] and broadcast communication [15], [16]. All such schemes are based on non-volatile memory technologies and vulnerable to spoofing/invasive attacks.

VII. Conclusions

We have proposed an approach to secure end-to-end communication in an AMI integrating the PUF technology with ANSI standard compliant smart meters. Our approach protects the confidentiality and the integrity of the messages and strongly authenticates smart meters. Furthermore, by exploiting the intrinstic characteristics of PUF devices, we prevent the leakage of secret keys utilized by smart meters. Our prototype implementation shows that it is practical and efficient to utilize PUF-enabled smart meters to provide secure end-to-end communication.

As future work, we plan to design and implement techniques to defend against denial of service attacks and to integrate our protocols with OASIS Key Management Interoperability Protocol (KMIP) [17]. We also plan to develop an FPGA-based smart meter that will integrate the PUF and the smart meter functions on a single device.

Acknowledgements

We would like to thank Athula Kulatunga and Uditha S. Navaratne from the Smart Meter Integration Lab (SMIL) of Purdue University for their valuable information on smart meters and providing smart meters for the proof-of-concept implementation.

References

[1] "ANSI C12 smart grid meter package," http://goo.gl/PQxkW.
[2] M. Bellare, P. Rogaway, and D. Wagner, "The eax mode of operation," in *FSE '04*, 2004, pp. 389–407.
[3] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *CCS '02*. New York, NY, USA: ACM, 2002, pp. 148–160.
[4] M. S. Kirkpatrick, S. Kerr, and E. Bertino, "Puf roks: a hardware approach to read-once keys," in *ASIACCS '11*. New York, NY, USA: ACM, 2011, pp. 155–164.
[5] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *CRYPTO '91*. London, UK: Springer-Verlag, 1992, pp. 129–140.
[6] C. Schnorr, "Efficient identification and signatures for smart cards," in *CRYPTO '89*. New York, NY, USA: Springer-Verlag New York, Inc., 1989, pp. 239–252.
[7] R. J. Anderson and M. G. Kuhn, "Low cost attacks on tamper resistant devices," in *Proceedings of the 5th Int. Workshop on Security Protocols*. London, UK, UK: Springer-Verlag, 1998, pp. 125–136.
[8] N. Shang, M. Nabeel, F. Paci, and E. Bertino, "A privacy-preserving approach to policy-based content dissemination," in *ICDE '10*, 2010.
[9] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *DAC '07*. New York, NY, USA: ACM, 2007, pp. 9–14.
[10] M. J. Atallah, E. D. Bryant, J. T. Korb, and J. R. Rice, "Binding software to specific native hardware in a vm environment: the puf challenge and opportunity," in *VMSec '08*. New York, NY, USA: ACM, 2008, pp. 45–48.
[11] M. Fouda, Z. Fadlullah, N. Kato, R. Lu, and X. Shen, "A lightweight message authentication scheme for smart grid communications," *IEEE Trans. on Smart Grid*, vol. 2, no. 4, pp. 675 –685, dec. 2011.
[12] Q. Li and G. Cao, "Multicast authentication in the smart grid with one-time signature," *IEEE Trans. on Smart Grid*, vol. 2, no. 4, pp. 686 –696, dec. 2011.
[13] D. Wu and C. Zhou, "Fault-tolerant and scalable key management for smart grid," *IEEE Trans. on Smart Grid*, vol. 2, no. 2, pp. 375 –381, june 2011.
[14] A. Saxena, O. Pal, S. Saiwan, and Z. Saquib, "Token based key management scheme for scada communication," *Int. J. on Distributed and Parallel Systems*, vol. 2, no. 4, pp. 69–86, july 2011.
[15] D. Choi, H. Kim, D. Won, and S. Kim, "Advanced key-management architecture for secure scada communications," *IEEE Trans. on Power Delivery*, vol. 24, no. 3, pp. 1154 –1163, july 2009.
[16] D. Choi, S. Lee, D. Won, and S. Kim, "Efficient secure group communications for scada," *IEEE Trans. on Power Delivery*, vol. 25, no. 2, pp. 714 –722, april 2010.
[17] "KMIP key management interoperability protocol," http://www.oasis-open.org/committees/kmip/.