Practical and Secure Machine-to-Machine Data Collection Protocol in Smart Grid

Suleyman Uludag[1], King-Shan Lui[2], Wenyu Ren[3], and Klara Nahrstedt[3]

[1]Department of Computer Science, University of Michigan - Flint, MI, USA
[2]Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong
[3]Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

***Abstract*—One of the most promising growth and deployment domains of Machine-to-Machine (M2M) communications is within the context of Smart Grid and its Advanced Metering Infrastructure. Data generation, collection, monitoring, and processing potentials are expected to be greatly expanded. An accompanying security for the communications infrastructure is generally agreed to be unmet by the legacy power grid networking protocols. In this paper, we present a new customized, secure, and scalable M2M data collection protocol for the Smart Grid. We use a hierarchical architecture with intermediary nodes collecting and relaying the data securely from measurement devices back to the power operator. While the data collectors verify the integrity, they are not given access to the content, which may also pave the way for third party providers to deliver value-added services or even the data collection itself. We also report some preliminary experimental results from a testbed developed from Raspberry Pi devices to emulate the resource-constrained measurement devices.**

I. Introduction

The Power Grid is set to include a substantially large number of sensors or measurement devices as part of the transitioning to the Smart Grid (SG). Collecting real-time information for monitoring, from generation to distribution of energy, will be a critical task for successful implementation. Security and scalability of the data collected must be ensured in such an environment. A hierarchical data collection framework is usually adopted to make data collection scalable. For example, in Advanced Meter Infrastructure (AMI), smart meters first report data to concentrators [1]–[3]. Collected data may further be processed by the concentrators before reporting to the power operator. With these intermediary devices, the power operator is relieved from the burden of maintaining separate connections with each measurement device or smart meter, which is neither practical nor scalable. Besides, data concentrators can aggregate the data reported by the smart meters to further reduce the message size.

Fig. 1 presents the data collection model we consider in this paper. The Measurement Devices (MDs) are sensors or smart meters that generate measurement data of the physical infrastructure. They are small and are constrained devices in terms of computational power, memory, and communication capabilities. While each MD may report its data to different Data Collectors (DCs), it is required that at least one DC is responsible for collecting each MD's data at any point in time. Each DC may connect to multiple MDs. Solid lines in

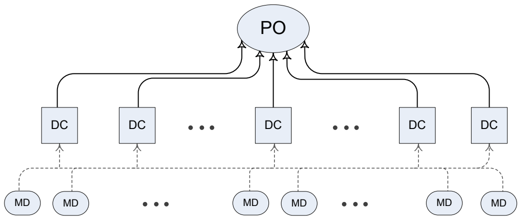

Fig. 1. Data Collection Model.

Figure 1 indicate that each DC will be communicating with the Power Operator (PO) on an ongoing basis, and dashed lines show that there are no predetermined pairings of MD-DC information exchange while any association is possible. The PO has a direct physical connection with each DC. PO and DCs are relatively more powerful than MDs. The data are reported to the PO via a set of DCs. If DCs are assumed to be within the security perimeter of the PO, then they may be trustworthy. However, due to the massive number of MDs and their dispersion over a large area, such an assumption may not be appropriate.In addition, one of the seven actors identified by the NIST in the SG Framework [4] is service providers which are to provide third party value-added services, including cloud computing [5], outsourcing [6], etc. For example, our approach makes DC outsourcing possible.

In some other applications [7], DCs are mobile and the connections between DCs and MDs are dynamic. Therefore, it would be desirable for MDs to encrypt their data in a way that DCs cannot have an access to, that is, each MD should encrypt its data using an appropriate key to keep its data private to DCs and other possible adversaries. On the other hand, due to limitations in memory and computational capabilities, the encryption algorithm used should be efficient. PO should also protect its commands appropriately.

In this paper, we develop a custom-tailored key establishment scheme and data collection protocol for a SG Machine-to-Machine (M2M) communications scenario. SG is one of the most relevant application areas for M2M communications [8], [9] and the setting we consider in this work falls within the same context. Our proposal protects the data and commands sent between PO and MDs via DCs in a scalable and efficient manner. Specifically, data reported by a certain MD can only be accessed by the PO, although the message is relayed by an untrusted DC. The protocol is light-weight in the sense that MDs do not have to perform expensive operation to report data and does not take much memory to remember key information.

To show the practicality and feasibility of our protocols, we have developed a small testbed of Raspberry Pi devices with our protocol implementation to mimic the constrained MDs. To the best of our knowledge, ours is the first in the literature in making use of a testbed of Raspberry Pi devices in SG M2M data collection. We report a preliminary results from our experiments in the testbed.

The rest of the paper is organized as follows: Sec. II presents the related work. Our secure data collection protocol with its operations is summarized in Sec. III. Sec. IV provides the details for the key establishment scheme. Secure data collection is presented in Sec. V. Testbed and experimental results are provided in Sec. VI. Sec. VII concludes the paper.

II. Related Work

[10] gives an overview of the security and privacy issues in the smart grid. Data integrity and confidentiality are the major security concerns. End-to-end data protection has been studied extensively in the Internet. However, most schemes, such as TLS [11], assume that the devices have abundant memory and computational power to perform expensive cryptographic operations. In smart grids, on the other hand, reporting devices have limited memory with a slow CPU. Traditional Internet security protocols are thus not suitable for data collection in smart grids. DNP3 [12] is a standard communication protocol used in SCADA (Supervisory Control And Data Acquisition). It assumes all components are within the security perimeter of the operator and is not designed to protect data forwarded by the DC as in our situation. A more recent standard for substation automation is the IP-based IEC 61850 [13]. Yet, IEC 61850 was also initially designed without security mechanisms [14]. It is thus generally agreed by the experts that new security protocols for data collection and command delivery need to be developed.

Some efforts have been put on key management of smart meters and sensors in smart grid. [15] describes a key management scheme for secure communication in smart grid. The scheme develops keys for unicast, multicast, and broadcast. Nodes are arranged as binary trees and the secret key of parent is the hash of the children keys. How different parties process or encrypt the data is not discussed. It is not clear whether the data reported by a certain meter can be hidden from the data collector. [16] also considers the key management problem for a massive number of smart meters. *Key graph* is used to manage unicast, multicast, and broadcast keys. Nevertheless, as DC is not considered in the architecture, the key management scheme cannot be applied in our scenario. The authors in [17] apply the elliptic curve public key technique to perform key management. Mutual authentication between different entities are studied. Nevertheless, there is no discussion on how to protect the data reported by a sensor.

Some protocols have been developed to establish shared keys when the two parties can establish direct communication. [18] describes how to established keys to secure unicast and multicast communications. The authors suggest keys to be established by direct connection between the two entities that need shared keys. [19] describes how to apply the Diffie-Hellman mechanism to establish a shared key for data authentication between two parties. [20], on the other hand, relies on identity-based cryptography. All these mechanisms cannot be applied in the hierarchical data collection model because the PO and the MDs cannot establish a direct connection. [21] studies how to reduce the storage needed when the control center needs to establish multiple sessions with the MDs. MDs are configured with a long-term shared key with the control center upon manufacturing. A function is used to generate this key. Thus, the control center does not have to remember a lot of keys but can derive the key when needed. Nevertheless, the key developed this way is not very secure. Besides, the protocol is not suitable for the hierarchical data collection architecture. [22] studies how data generators report to a *honest-but-curious* storage center for a user to retrieve later. The data collection trust model assumed in this paper is the most related to our scenario. The storage center is similar to the DC in our model that can be semi-trusted, and data should be hidden from it. MDs in our model are the data generators, while PO is a user in their model. However, the paper suggests to use expensive identity-based and public key encryption to protect data to incorporate policy consideration.

An approach presented in [23] is based on symmetric cryptography to provide data confidentiality and authentication between sensors and the base station. [24] describes a protocol for DC to collect data from an MD, requiring the PO and the MD to establish a new shared key for every data collection. The protocol in this paper smartly uses a group key to allow data to be collected with less overhead. Unlike [24], we provide an implementation in this paper. Another category for providing security takes advantage of in-network data processing (aka *aggregation*) to induce some masquerading behavior on the transmitted data [3], [25]–[27].

III. System and Protocol Overview

A. Operations and their requirements

Our communications architecture enables the PO to initiate data collection of all or a group of MDs in a timely and secure manner. This is a regular call-for-data from the PO which is performed periodically. Data reported by a certain MD should be authenticated and should be read only by the PO but not by other MDs or any DC. Besides, an MD may launch an urgent data reporting process. This is done when MD detects something abnormal and would like to report to the PO. Security requirement of this operation must be of the same level as the the regular data collection.

We develop our protocol to be secure from outsider attacks such as eavesdropping, impersonation, and message tampering, etc. There are three types of insiders in the protocol: PO, DCs, and MDs. As the PO is the main control of the whole system and the recipient of all data, we assume it is always trustworthy. The DCs, on the other hand, are *honest-but-curious*. They are honest in the sense that they would follow the protocol as specified. They would not impersonate other entity or tamper any data. On the other hand, they are curious and eager to read the data transmitted. The MDs should be trustworthy at the time of installation. However, as they are located in the field which may not be under a very secure physical environment, it is possible that an attacker takes over the MD some time after the installation. In this case, the attacker can read the key information kept in the device and can report fake data to the PO on behalf of the MD. Our protocol cannot identify whether the data reported using a legitimate key is generated by an attacker, but our protocol ensures this compromised MD cannot impersonate others based on the key information it has. To detect whether a certain MD is compromised, the data reported and behavior should be carefully analyzed, which is beyond our scope here.

B. System Parameters

PO, DCs, and MDs are assumed to have been equipped with the long-term secrets, a set of system parameters, and the required cryptographic functions in advance. A DC or MD should have all parameters and functions configured before it is installed in the field.

Long-term secrets/keys: We assume there is a key server that can generate a set of public and private keys for each entity in the system. The public/private key pair is configured into a DC or MD before it is installed in the field. PO, on the other hand, apart from keeping its own key pair, it also remembers the public keys of all MDs and DCs in the system. We denote the public key and private key of node A as A^+ and A^-, respectively. It is not likely the PO would like to publish the public keys of DCs and MDs in an open site as PO should not expect or allow any outsider to know the keys. However, our protocol is secure even if the attackers know the public key information of any DC or MD they want to attack.

Diffie-Hellman (DH) parameters: We adopt the Diffie-Hellman key exchange mechanism to develop shared keys. The DH key exchange works as follows: When Alice and Bob want to generate a shared key using DH, they first each generate a random number and keep it as a secret. Let a and b be the secret, also called the secret DH half key, of Alice and Bob, respectively. They, then, exchange $g^a mod\ p$ and $g^b mod\ p$, where p is prime and g is a primitive root $mod\ p$. $g^a mod\ p$ ($g^b mod\ p$) is called the public half key or public DH key of Alice (Bob). When Alice receives the public half key $g^b mod\ p$ sent by Bob, she can compute the shared key $g^{ab} mod\ p$ by $(g^b mod\ p)^a mod\ p$. Similarly, Bob can compute the shared key by $(g^a mod\ p)^b mod\ p$. Although eavesdroppers can overhear $g^a mod\ p$ and $g^b mod\ p$, they cannot compute $g^{ab} mod\ p$. Therefore, the shared key is secure. Through removing half keys and shared keys from memory after they are no longer used appropriately, DH keys also support *perfect forward secrecy* [28].
Before using the DH mechanism, the PO, DCs, and MDs have to agree on the g and p to be used. We assume PO picks g and p and configures them into DCs and MDs before installation. To make the discussion concise, we drop $mod\ p$ when the context is clear in the rest of the paper.

Cryptographic functions: To provide authentication, confidentiality, integrity, and other security protections, messages have to be encrypted, hashed, or signed. We assume the PO selects appropriate cryptographic algorithms for the purposes, and these functions are installed in the DCs and MDs. For example, PO may use AES for symmetric key encryption and SHA-256 for hash computation. Table I summarizes the

Name	Description
$PKE(K,M)$	apply public key encryption on M using K
$SKE(K,M)$	apply symmetric key encryption on M using K
$SIGN(A,M)$	The signature of M by A (created using A^-)
$HASH(K,M)$	compute the keyed-hash of M using key K
$GENKEY(X,Y)$	generate a key based on X and Y

TABLE I
SYSTEM FUNCTIONS

functions used in the protocol. Function $GENKEY(X,Y)$ is used when we need a key generated from two numbers X and Y. This function is very computationally inexpensive that the time needed is negligible when compared with any cryptographic function.

C. Protocol Overview

The long-term keys in our system should not be used for data encryption because, on one hand, public key encryption is expensive, and on the other hand, it is not a good practice to use the long-term keys to encrypt data. Just like other existing network security protocols, we first establish shared keys among PO, DCs, and MDs using the long-term keys, and then these shared keys are used for data protection. To ensure data reported by a certain MD can be decrypted by the PO only, we need to establish a key that is known by PO and that MD. We call a key that is known by exactly two parties a *pairwise shared key*. PO and each DC should also develop a pairwise shared key to protect their conversations. The key used for DC to talk to MD, on the other hand, is known by the PO too. We will explain later the reason of establishing this key in this way. Apart from keys shared between two or three parties, we also develop a set of *group keys* where each group key is shared among the PO, a certain DC, and MDs that connect to that DC. The group keys allow data collection to be done in a fast and light-weight manner.

The PO initiates the *Shared Key Generation Process* to establish the necessary keys. We adopt the Diffie-Hellman key exchange mechanism to develop all shared keys between two or three parties. We authenticate the DH half keys using the long-term public keys to avoid the man-in-the-middle attack. Once the shared keys and group keys are established, they will be used for data collection.

DH operations are expensive. We should not re-generate the DH shared keys for every data collection. However, it may not be very secure if we use the same shared keys to encrypt data collected at different times. To strike a balance of computational complexity and security, the data encryption key for each data collection depends on both the DH shared key and the timestamp. As the timestamp changes for every data collection, the data encryption key will be changed even though we do not re-generate the DH shared key. In the following, we will first describe in the Shared Key Generation process in Section IV. The detailed message exchanges of data collection will be provided in Section V.

IV. Shared Key Generation

Before the PO initiates the shared key generation process, PO has to determine the set of MDs that a particular DC has to talk to. We let $MDLIST_i$ be the set of MDs assigned to DC_i. For the ease of discussion, we denote K_B^A as the shared key between A and B. We refer the set $\{PO, DC_i\} \cup MDLIST_i$ as group G_i, and the group key of G_i is GK_i. We use $M1||M2$ to represent concatenating messages $M1$ and $M2$. The definitions of the functions used can be found in Table I.

Figure 2 presents a summary of the shared key generation process. When the procedure starts, the only keys an MD or a DC knows are its own public/private keys and the public key of the PO. After the procedure, MD_j should have established $K_{MD_j}^{PO}$, $K_{MD_j}^{DC_i}$, and GK_i if $MD_j \in MDLIST_i$. Through the procedure, DC_i knows GK_i, $K_{DC_i}^{PO}$ and $K_{MD_j}^{DC_i}$ for all $MD_j \in MDLIST_i$. The detailed procedure is as follows (For clarify, we drop subscript i and j in Figure 2.):

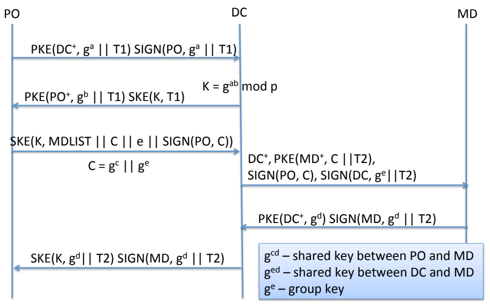
PO
DC
MD
PKE(DC⁺, gᵃ || T1) SIGN(PO, gᵃ || T1)
K = gᵃᵇ mod p
PKE(PO⁺, gᵇ || T1) SKE(K, T1)
SKE(K, MDLIST || C || e || SIGN(PO, C))
C = gᶜ || gᵉ
DC⁺, PKE(MD⁺, C ||T2),
SIGN(PO, C), SIGN(DC, gᵉ||T2)
PKE(DC⁺, gᵈ) SIGN(MD, gᵈ || T2)
SKE(K, gᵈ|| T2) SIGN(MD, gᵈ || T2)
gᶜᵈ – shared key between PO and MD
gᵉᵈ – shared key between DC and MD
gᵉ – group key

Fig. 2. Shared Key Generation

1) PO generates a DH secret a to talk to the DCs and captures the current timestamp $T1$. It then encrypts and signs g^a and $T1$. The message to DC_i is thus protected from message tampering and impersonation.

$$PO \rightarrow DC_i\text{: } PKE(DC_i^+, g^a||T1), SIGN(PO, g^a||T1)$$

2) DC_i verifies whether the message has been tampered by verifying whether $T1$ differs from DC's local clock within a reasonable amount. It then generates its DH secret b and computes $K_{DC_i}^{PO}$ as $g^{ab} mod\ p$. It also replies to PO with its public DH key. Note that an attacker cannot impersonate DC_i as he does not know DC_i^-.

$$DC_i \rightarrow PO\text{: } PKE(PO^+, g^b||T1), SKE(K_{DC_i}^{PO}, T1)$$

3) After verifying $SKE(K, T1)$ to ensure it was DC_i who sent the message, PO sends DC_i the list of MDs, together with the MDs' public keys, that it assigns DC_i to talk to. It also creates C, which is part of the message, for DC_i to talk to the MDs in the list. C contains two DH half keys: g^c and g^{e_i}. g^c is used for generating $K_{MD_j}^{PO}$, and g^{e_i} is used for generating $K_{MD_j}^{DC_i}$. It is worth noting that to allow DC_i to develop $K_{MD_j}^{DC_i}$ using the DH mechanism, DC_i has to know e_i. Therefore, PO sends e_i to DC_i. In other words, PO picks the DH secret key for DC_i instead of allowing DC_i to pick by itself. This can avoid curious DCs to work together to guess the DH secret of an MD [24]. As PO knows e_i, PO can also develop $K_{MD_j}^{DC_i}$. Apart from the DH half keys, the public keys of the MDs should also be sent (We assume they are included in $MDLIST_i$ in Figure 2). PO sends $SIGN(PO, C||DC_i^+)$ to avoid messages from being tampered.

$$PO \rightarrow DC_i\text{: } SKE(K, MDLIST_i||C||e_i||SIGN(PO, C||DC_i^+))$$
$$\text{where } C = g^c||g^{e_i}$$

As the PO, DC_i, and MD_j all know g^{e_i} and we protect g^{e_i} through encryptions and signatures, we assign the group key GK_i to be g^{e_i}.

4) After verifying the message sent from the PO, DC_i captures the current timestamp $T2$ and sends the information to MD_j in $MDLIST_i$. DC_i also needs to send its public key. To protect DC_i^+ and $C||T2$ from being tampered, $SIGN(PO, C||DC^+)$ and $SIGN(DC_i, g^{e_i}||T2)$ are sent as well.

$$DC_i \rightarrow MD_j\text{: } DC_i^+,\ PKE(MD_j^+, C||T2),\ SIGN(PO, C),$$
$$SIGN(DC_i, g^{e_i}||T2)$$

5) After retrieving the DH half keys from the message sent by DC_i, MD_j generates its DH secret d. It establishes $K_{MD_j}^{PO}$ to be g^{cd} and $K_{MD_j}^{DC_i}$ to be $g^{e_i d}$. As DC_i sends the same g^{e_i} and $T2$ to all other MDs in $MDLIST_i$, MD_j has to sign g^d to authenticate the reply.

$$MD_j \rightarrow DC_i\text{: } PKE(DC_i^+, g^d),\ SIGN(MD_j, g^d||T2)$$

6) When DC_i receives the message, it retreives g^d and verifies the signature. If so, it sends PO the key information.

$$DC_i \rightarrow PO\text{: } SKE(K_{DC_i}^{PO}, g^d||T2),\ SIGN(MD_j, g^d||T2)$$

7) If $g^d||T2$ encrypted using $K_{DC_i}^{PO}$ and signed by MD_j are the same, the message has not been tampered. PO can then compute $K_{MD_j}^{PO}$ to be g^{cd}. Note that as DC_i can only read g^c and g^d but neither c nor d, it cannot compute g^{cd}. g^{cd} is thus a key shared by PO and MD_j only.

V. Data Collection

A. Data Collection of a Group of or All MDs

It is a regular data collection initiated by the PO. Data is encrypted using $K_{MD_j}^{PO}$ so that no outsiders, as well as no DCs, can read the data. As MDs are limited in computational capability, they should not be requested to perform a lot of expensive operations. The whole data collection process is shown in Fig. 3. In the figure, $K1$ is $K_{DC_i}^{PO}$, $K2$ is $K_{MD_j}^{PO}$, and $K3$ is $K_{MD_i}^{DC_i}$.

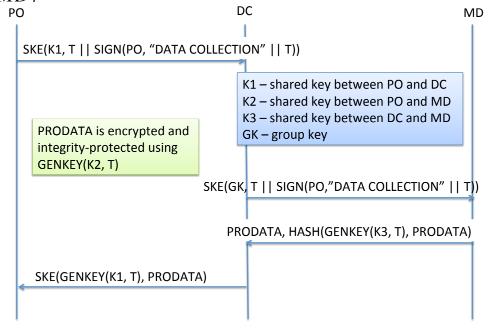
PO
DC
MD
SKE(K1, T || SIGN(PO, "DATA COLLECTION" || T))
K1 – shared key between PO and DC
K2 – shared key between PO and MD
K3 – shared key between DC and MD
GK – group key
PRODATA is encrypted and integrity-protected using GENKEY(K2, T)
SKE(GK, T || SIGN(PO,"DATA COLLECTION" || T))
PRODATA, HASH(GENKEY(K3, T), PRODATA)
SKE(GENKEY(K1, T), PRODATA)

Fig. 3. Data Collection.

1) When PO wants to collect data, it first captures the current timestamp T, signs it, and sends a message to DC_i. The signature avoids impersonation.

$$PO \rightarrow DC_i\text{:}$$
$$SKE(K^{PO}_{DC_i}, T||SIGN(PO, \text{"}DATA\ COLLECTION\text{"}||T))$$

2) After verifying the message from the PO, DC_i sends T to $MD_j \in MDLIST_i$.

$$DC_i \rightarrow MD_j\text{: } SKE(GK_i, T||SIGN(PO, \text{"}DATA\text{"}||T))$$

It is worth noting that the message is encrypted using the group key GK_i. In this way, DC_i only needs to create a single message for all MDs in its group. Since any member in G_i can encrypt something using GK_i, we need to include a signature of PO to authenticate the message.

3) After verifying T, MD_j generates a key $MK = GENKEY(K^{PO}_{MD_j}, T)$. An encryption key and an integrity key developed based on MK are used to protect the data. The protected data is denoted as $PRODATA$. MD_j also generates $DK = GENKEY(K^{DC_i}_{MD_j}, T)$. The hash of $PRODATA$ using DK is computed and sent to DC_i. By using MK and DK, which depend on T, we can ensure the actual keys to be used to protect the data would be different at different times even though the same shared keys ($K^{PO}_{MD_j}$ and $K^{DC_i}_{MD_j}$) are used.

$$MD_j \rightarrow DC_i\text{: } PRODATA, HASH(DK, PRODATA)$$

4) DC_i verifies the hash to ensure $PRODATA$ was generated by MD_j even it cannot decrypt $PRODATA$. It then forwards $PRODATA$ to PO.

$$DC_i \rightarrow PO\text{: } SKE(GENKEY(K^{PO}_{DC_i}, T), PRODATA)$$

Alternatively, DC_i can encrypt all the replies from MDs in a single message. In this case, only a single symmetric key encryption is needed, but PO may receive some data later.

5) Finally, PO develops MK on its own to extract the data from $PRODATA$.

We now analyze the complexity from the MD's perspective. To decrypt the message from the DC, MD has to perform one symmetric key decryption and one signature verification. To prepare the $PRODATA$, it has to perform one symmetric key encryption and one hash computation. It has to perform another hash computation for the integrity check for the DC. Therefore, MD only has to perform one single expensive public key operation in the whole data collection process. Our mechanism is thus very light-weight.

B. MD_j initiates an urgent data report

1) MD_j first identifies a certain DC_i to relay the message and records the current timestamp T. $PRODATA$ and DK are generated as in Step 3 in Section V-A.

$$MD_j \rightarrow DC_i\text{:}$$
$$SKE(K^{DC_i}_{MD_j}, T||PRODATA||HASH(DK, PRODATA))$$

2) DC_i verifies the hash and forwards $PRODATA$ to PO.

$$DC_i \rightarrow PO\text{: } SKE(K^{PO}_{DC_i}, T||PRODATA)$$

3) PO can then extract T using $K^{PO}_{DC_i}$ to develop the appropriate keys to decrypt $PRODATA$.

In reporting emergency information, latency and reliability are very important. In the protocol, MD_j does not need to perform any expensive public key operation before sending the data report. The latency is thus very small. To enhance reliability, MD_j can send the data to PO via multiple DCs. It has to compute $HASH(DK, PRODATA)$ and encrypt $T||PRODATA||HASH(DK, PRODATA)$ using different keys for different DCs in Step 1. As both operations are not expensive, MD_j can send out the reports promptly.

VI. Experiment Results

To evaluate the performance of our protocol, we have built a testbed to analyze the time consumption of data collection process of the protocol. We have used credit card sized Raspberry Pi devices in our testbed to emulate the resource-constrained MDs. Raspberry Pi is a tiny, single-board computer (or embedded device) with limited computational, storage and communications capabilities. The CPU is 700MHz and the memory available is 512MB.

Since MDs are highly resource-constrained in terms of computation power and storage compared to POs and DCs, the bottleneck of the protocol is on the MD side. Thus, we put our experiment's emphasis on the part between DC and MD of the protocol. Our testbed includes a laptop, which stands for the DC, and 15 Raspberry Pis, which act as multiple MDs. All of them are configured to run in an ad-hoc wireless mode. That is, all the MDs are able to communicate with the DC in an ad-hoc manner without any access point.

In our experiments, we focus on the message generation, communication, message processing between DC and MD. We measure the total time consumption between DC and MD for the data collection process in our protocol. The total time includes 5 parts: (1) the time for DC to generate the message to send to MD; (2) the transmission time for the message from DC to MD; (3) the time for MD to process the message from DC and generate the message to send to DC; (4) the transmission time for the message from MD to DC; (5) the time for DC to process the message from MD. In all of our experiments, we use DH keys of 1024 bits, RSA keys of 3072 bits and AES keys of 256 bits. Multi-threading is used when DC communicates with multiple MDs to take advantage of the parallelization. The time for multiple MDs is the total time consumption to finish all the threads.

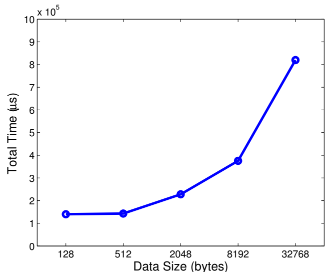

Fig. 4. Time for Data Collection with 1 MD.

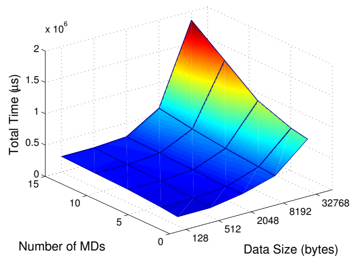

Fig. 5. Time for Data Collection with Different Data Sizes and number of MDs.

Fig. 4 shows the total time for data collection versus different data sizes when one DC collects data from one MD. As expected intuitively, larger data takes longer to collect. Fig. 5 plots the total time for data collection by one DC under varying data size and number of MDs. With even a small number of MDs, this shows that a high degree of coordination is needed to timely collect data when the number of MDs becomes very large. In this respect, association of MDs with DCs emerge as a critical problem, which we will be studying as part of our future work. Another aspect of the results rises about the frequency of the data collection. For high-frequency data collection, the volume of data becomes large, deserving a meticulous attention.

VII. Conclusion

Data collection from smart meters, sensors, and measurement devices is a critical component for delivering the desirable promises of the Smart Grid. Vastly increased data generation, monitoring, and processing capabilities in the Smart Grid are calling for equivalently high impact mechanisms to ensure the security while preserving the reliability and scalability. Automation becomes inevitable and contribution of the Machine-to-Machine communications manifests itself clearly. In this paper, we present a comprehensive and secure communications protocol to enable a power operator to collect data from measurement devices in a practical, scalable, and efficient manner under a hierarchical data collection model where intermediary devices relay data from measurement devices on behalf of the power operator securely without access to it. Our protocol paves the way for third party service provisioning, as envisioned by the NIST Smart Grid Framework. Examples of such third party services include outsourcing data collection by 3rd party DCs, utilizing cloud computing services for data storage and processing, etc. We have also developed a testbed of Raspberry Pi devices to mimic the resource-constrained measurement devices and fully deployed our protocol on them. Preliminary results from this testbed indicate that small message size and small group of MDs yield very low time for secure data collection.

VIII. Acknowledgment and Disclaimer

This material is based upon work supported by the Department of Energy under Award Number DE-OE0000097.

References

[1] N. Kayastha, D. Niyato, E. Hossain, and Z. Han, "Smart grid sensor data collection, communication, and networking: a tutorial," *Wireless Communications and Mobile Computing*, pp. n/a–n/a, 2012.

[2] B. Karimi, V. Namboodiri, and M. Jadliwala, "On the scalable collection of metering data in smart grids through message concatenation," in *Proc. of IEEE SmartGridComm*, 2013.

[3] C. Rottondi, G. Verticale, and C. Krauss, "Distributed privacy-preserving aggregation of metering data in smart grids," *IEEE JSAC*, vol. 31, no. 7, pp. 1342–1354, July 2013.

[4] National Institute of Standards and Technology. (2013, October) NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 3.0. Smart Grid Interoperability Panel (SGIP).

[5] S. Bera, S. Misra, and J. Rodrigues, "Cloud computing applications for smart grid: A survey," *IEEE Tran. on Par. and Dist. Sys.*, no. 99, 2014.

[6] X. Fang, S. Misra, G. Xue, and D. Yang, "Managing smart grid information in the cloud: opportunities, model, and applications," *Network, IEEE*, vol. 26, no. 4, pp. 32–38, July 2012.

[7] R. Tabassum, K. Nahrstedt, E. Rogers, and K.-S. Lui, "Scapach: Scalable password-changing protocol for smart grid device authentication," in *IEEE ICCCN*, July 2013, pp. 1–5.

[8] Z. Fadlullah, M. Fouda, N. Kato, A. Takeuchi, N. Iwasaki, and Y. Nozaki, "Toward intelligent machine-to-machine communications in smart grid," *IEEE Comm. Mag.*, vol. 49, no. 4, pp. 60–65, April 2011.

[9] Jesus Alonso-Zarate, Javier Matamoros, David Gregoratti, and Mischa Dohler, "Machine-to-machine communications in smart grid," in *Smart Grid Communications and Networking*, E. Hossain, Z. Han, and H. Poor, Eds. Cambridge University Press, 2012, Ch.6.

[10] J. Liu, Y. Xiao, S. Li, W. Liang, and C. Chen, "Cyber security and privacy issues in smart grids," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 4, Fourth Quarter 2012.

[11] *RFC 5246*, "The transport layer security (tls) protocol version 1.2," 2008.

[12] *IEEE 1815-2012*, "Dnp3 secure authentication version 5," 2011.

[13] International Electrotechnical Commission's (IEC) Technical Committee 57 (TC57). IEC 61850, Power Utility Automation .

[14] W. Wang and Z. Lu, "Cyber security in the smart grid: Survey and challenges," *Computer Networks*, vol. 57, no. 5, pp. 1344 – 1371, 2013.

[15] X. Long, D. Tipper, and Y. Qian, "An advanced key management scheme for secure smart grid communications," in *Proc. of IEEE SmartGridComm*, 2013.

[16] N. Liu, J. Chen, L. Zhu, J. Zhang, and Y. He, "A key management scheme for secure communications of advanced metering infrastructure in smart grid," *IEEE Tran. on Ind. Electr.*, vol. 60, no. 10, 2013.

[17] D. Wu and C. Zhou, "Fault-tolerant and scalable key management for smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, June 2011.

[18] Y. Law, G. Kounga, and A. Lo, "WAKE: Key management scheme for wide-area measurement systems in smart grid," *IEEE Comm. Mag.*, Jan. 2013.

[19] M. M. Fouda, Z. M. 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, Dec. 2011.

[20] C. Bekara, T. Luckenbach, and K. Bekara, "A privacy preserving and secure authentication protocol for the advanced metering infrastructure with non-repudiation service," in *Proc. of ENERGY*, 2012.

[21] Y.-J. Kim, V. Kolesnikov, H. Kim, and M. Thottan, "SSTP: a scalable and secure transport protocol for smart grid data collection," in *Proc. of IEEE SmartGridComm*, 2011.

[22] J. Hur, "Attribute-based secure data sharing with hidden policies in smart grid," *IEEE Tran. on Par. & Dist. Sys.*, vol. 24, no. 11, 2013.

[23] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: Security protocols for sensor networks," *Wirel. Netw.*, vol. 8, no. 5, pp. 521–534, Sep. 2002.

[24] G. Dan, K.-S. Lui, R. Tabassum, Q. Zhu, and K. Nahrstedt, "Selinda: A secure, scalable and light-weight data collection protocol for smart grids," in *IEEE SmartGridComm*, Oct 2013, pp. 480–485.

[25] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "Pda: Privacy-preserving data aggregation in wireless sensor networks," in *IEEE INFOCOM 2007*, May 2007, pp. 2045–2053.

[26] T. Feng, C. Wang, W. Zhang, and L. Ruan, "Confidentiality protection for distributed sensor data aggregation," in *IEEE INFOCOM*, April 2008.

[27] K. Kursawe, G. Danezis, and M. Kohlweiss, "Privacy-friendly aggregation for the smart-grid," in *Privacy Enhancing Technologies*. Springer, 2011, vol. 6794, pp. 175–191.

[28] W. Diffie, P. Oorschot, and M. Wiener, "Authentication and authenticated key exchanges," *Designs, Codes and Cryptography*, vol. 2, no. 2, pp. 107–125, 1992.