SELINDA: A Secure, Scalable and Light-Weight Data Collection Protocol for Smart Grids

György Dán[1], King-Shan Lui[2], Rehana Tabassum[3], Quanyan Zhu[4], and Klara Nahrstedt[3]

[1]School of Electrical Engineering, KTH, Royal Institute of Technology, Stockholm, Sweden
[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
[4]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, IL, USA

***Abstract*—Security in the smart grid is a challenge as an increasing number of sensors and measurement devices are connected to the power grid. General purpose security protocols are not suitable for providing data security to devices with limited memory, computational power and network connectivity. In this paper, we develop a secure and light-weight scalable security protocol that allows a power system operator (PO) to collect data from measurement devices (MDs) using data collectors (DCs). The security protocol trades off between computations and device memory requirements and provides flexible association between DC and MDs. These features allow data to be securely transferred from MDs to PO via mobile or untrustworthy DCs. We analyze the complexity and security of the protocol and validate its performance using experiments. Our results confirm that our proposed protocol collects data in a secure, fast and efficient manner.**

I. Introduction

The proper operation of the smart grid relies on the quality of data collected from a vast number of sensors and measurement devices (MDs). The data collection needs to be efficient and secure in order for smart grids to be economical and dependable. Efficiency and scalability arguably require a hierarchical data collection framework to be adopted [1]. Fig. 1 shows a simplified hierarchical data collection model. Data collectors (DCs) collect data from MDs, and send the data to a control center, typically owned by the power operator (PO). Each DC is responsible for collecting data from multiple MDs, and therefore the PO only needs to communicate with a few DCs directly, keeping the number of connections to maintain manageability.

A DC can be assumed to be trusted if it is within the operator's security perimeter, and therefore can be used to read the collected data. Extending the security perimeter to a vast number of DCs is, however, expensive, and can even be infeasible when the data collection architecture is changeable or DCs are mobile. If the DCs cannot be trusted, the MDs need to encrypt and authenticate the data so that the DC does not have the access to data even if it carries them. In this way, the data are secure even if the DC is compromised. In the event that the DC needs to aggregate the data, homomorphic encryption [2] can be used to maintain data confidentiality. This allows the PO to outsource the data collection and reduce operation cost without sacrificing security. While various protocols have been proposed for trusted DCs [3]–[6], there is no efficient protocol yet for secure data collection via untrusted DCs.

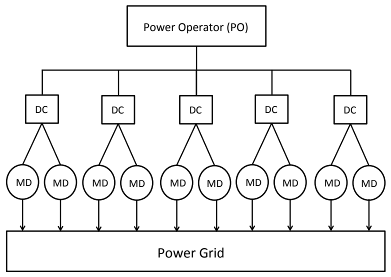

Fig. 1. A three-level hierarchical data collection model

In this work, we propose and analyze a key establishment and data collection protocol, SELINDA, that allows a PO to establish shared keys with multiple MDs via an untrusted DC. The DC behaves like a relay for data communications although it is not continuously connected to the PO. Besides, the DC has no access to the keys established between the PO and the MDs. Therefore, the DC can potentially be mobile and untrusted, which makes our scheme essential for ensuring the security of community aided data collection in the smart grid.

Our protocol has four key features that distinguish it from existing protocols. First, the protocol is computationally light-weight for the MDs as it requires the MDs to perform very few expensive cryptographic operations and to send few messages. Thus the protocol supports resource constrained MDs with limited memory and a slow CPU. Second, the protocol allows to trade off computation for memory requirements in the MDs. There is one long-term secret per entity, its private key. Thus, the PO needs to maintain the public key of all MDs in the system. It also maintains state information for the current session of data collection. The MD only needs to remember its own private key and the public key of the PO. They can recalculate or store the session keys, depending on their computational and memory constraints. Third, the protocol provides flexible association between DCs and MDs. As an MD does not need to know the public key of any DC, the PO can assign different DCs to collect data from the same MD at different times. This feature is particularly important in scenarios where the DCs are mobile or the infrastructure is evolving so that different mappings or assignments between DCs and MDs can be used at different times after the deployment of the MDs. Finally, the protocol protects the collected data from a compromised DC. Thus, an attacker cannot access the collected data even if it is in control of a DC. This allows the PO to outsource the data collection procedure without sacrificing security.

The paper is organized as follows. In Section II, we discuss related works. Section III describes a two-phase protocol that enables secure data collection via untrusted DCs. In Section IV, we analyze the complexity and security of the protocol. Section V provides experimental results of the protocol and we conclude the paper in Section VI.

II. Related Work

Security in smart grids is a challenging problem for many reasons [7], [8]. One of the biggest challenges comes from connecting the grid with a plethora of devices with limited memory, computational power and network connectivity [7]. Furthermore, interoperability and legacy-compliance are also key concerns [8]. General purpose Internet security protocols are thus not suitable, and new security protocols tailored for the smart grid need to be developed.

[5] describes a lightweight and scalable transport protocol for establishing multiple sessions among MDs to the control center. The study focuses on how to reduce the storage needed in maintaining the state information of the massive amount of sessions. The session keys used can be derived so that the control center does not have to remember a lot of keys. Nevertheless, the number of sessions maintained is still proportional to the number of MDs. The protocol is also not suitable for the hierarchical data collection architecture in which a DC is responsible for collecting and/or processing data from multiple MDs before sending the data to the control center. The authors in [4] develop a transport protocol for reporting data through a data collector. Two separate TCP connections are maintained: one between the control center and the data collector, and another between the MD and the data collector. Each connection can be protected independently. This approach assumes that the data collector is trustworthy, which may not be the case when the data collector is outsourced or compromised.

Secure authentication for smart grids has been considered in [9], [10]. The focus of these mechanisms is on how to establish a shared key for data authentication between two entities. The two entities need to establish a session, which may be infeasible in the hierarchical data collection model. [6] studies how to remotely access power system devices for substation monitoring. The substation controller (or data concentrator) is used as the central point of access control. The substation controller authenticates users and keeps access logs, and is assumed to be trusted. The issue of securing data between the devices and the PO is not discussed.

Another line of work deals with key management. The authors in [3] propose to have a trust anchor for performing mutual authentication and key establishment between a device and a collector. This approach may not be scalable when there are many devices. [11] studies which unicast and multicast sessions need to be secured in a wide-area measurement system. The authors suggest keys to be established by direct connection between the two entities that need shared keys. Thus, the scheme is not suitable for secure data collection via a data collector. The work in [12] describes a key management scheme for unicast, multicast, and broadcast messages in AMI. The keys form a graph so that keys are easily stored and derived. Session keys are generated based on previously read data. Nevertheless, this scheme cannot be applied in a hierarchical data collection architecture.

TABLE I
Summary of Notations

Symbol	Meaning
PO	Power Operator
DC	Data Collector
MD	Measurement Device
$K\{M\}$	Encrypt message M using shared key K
$[M]_A$	Sign message M using the private key of entity A
$\{M\}_A$	Encrypt message M using the public key of entity A
$G(p)$	Multiplicative group over prime p
$G_q(p)$	Subgroup of $G(p)$ of order q
g	Generator of subgroup $G_{q_0}(p)$ for a large prime q_0

To the best of our knowledge, there is no existing work that allows a PO to generate different shared keys with different MDs in a scalable manner. Existing standard protocols such as DNP3 [13] and TLS [14] are not suitable for the scenario when the data collectors are untrusted and potentially mobile with intermittent connectivity. DNP3 [13] is a standard communication protocol used for telecontrol and telemetry in SCADA systems. Its security model is not designed to provide data integrity and confidentiality against compromised relay nodes, as it assumes that all components are within the security perimeter of the operator. TLS [14], on the other hand, involves multiple phases of handshakes and is therefore not suitable if the data collector is off-line when communicating the measurement devices. In this paper, we present a secure and light-weight protocol that allows a power operator to collect data from measurement devices using potentially multiple mobile and non-trustworthy data collectors.

III. Protocol for Secure Data Collection

Before we describe our protocol in detail, we first explain the security and scalability objectives that we want to achieve. Table I summarizes the notations used in the paper.

Confidentiality and integrity: The data reported by the MD to the PO should remain secret to a potential eavesdropper, an active attacker, and a compromized DC. The PO should be able to tell whether the data have been tampered with. In addition, our protocol should allow the DC to perform an integrity check right after it receives the message containing the data from the MD, even though it cannot decrypt the message to retrieve the raw data. This enables a trustworthy DC to immediately detect potential data corruption/tampering, so that a remedial action can be taken. Without this feature, corrupted/tampered data cannot be detected until delivered to the PO. We establish different session keys between different pairs of parties to achieve this objective.

Perfect forward secrecy (PFS): Our protocol protects the confidentiality of the data sent in earlier sessions even if the long-term secrets are obtained by the attacker later. We adopt the Diffie-Hellman (DH) mechanism for key establishment to support PFS.

Low memory and computational requirement: The data reported by different MDs must be protected by different keys. To reduce the memory needed to maintain this key information, the PO uses the same DH public key to develop different shared keys with different MDs. This approach may be subject to the *small subgroup attack*, i.e., an attack in which several compromised MDs try to guess the PO's DH secret key. In Section IV, we will describe the attack and analyze the probability of such an attack to be successful.

MDs are computationally-constrained devices, and they should not perform too many expensive operations or handle too many messages. In our protocol, the MD only has to exchange two messages with the DC. This simple handshake not only reduces complexity, but also allows data to be reported quickly. To further understand the complexity of our protocol, we conduct simulations in Section V.

We assume that the PO, the DCs, and the MDs are initially configured with the following set of parameters:

Long-term keys: The PO, every DC, and every MD have their public and private key pairs; the PO knows the public keys of all MDs and DCs, while the MDs know the public key of the PO. MD does not store the public key of DC so that PO can flexibly assign different DCs to collect data at different time.

Diffie-Hellman (DH) parameters: The PO and the MDs agree on parameters of the prime order digital signature algorithm subgroup $G_{q_0}(p)$ with generator g of group $G(p)$. The length of p is the same order of magnitude as that of a public key, but q_0 is substantially smaller.

The Diffie-Hellman key exchange works as follows: when Alice and Bob want to establish a shared key, Alice picks a natural number a and keeps as secret. Similarly, Bob picks a natural number b as his secret. Alice sends $g^a mod\ p$ to Bob, and Bob sends $g^b mod\ p$ to Alice. When Alice receives $g^b mod\ p$, she computes the shared key by $(g^b mod\ p)^a\ mod\ p = g^{ab} mod\ p$. Bob, on the other hand, computes the shared key by $(g^a mod\ p)^b\ mod\ p = g^{ab} mod\ p$. Note that although an eavesdropper knows p, g, $g^a mod\ p$, and $g^b mod\ p$, it is very difficult for him to compute $g^{ab} mod\ p$. In this paper, we call $g^a mod\ p$ and $g^b mod\ p$ DH half keys or DH public keys.

Cryptographic functions: The PO and the MDs have a common set of cryptographic schemes for encryption (e.g., AES) and for hash computation (e.g., SHA-256). We use a keyed hash for data authentication. In the rest of the paper, we call the key for encrypting data the *encryption key*, and the key for providing integrity the *integrity key*.

In total, every MD has to store one private key, one public key, and the parameters of the group $G_{q_0}(p)$, i.e., g, q_0 and p.

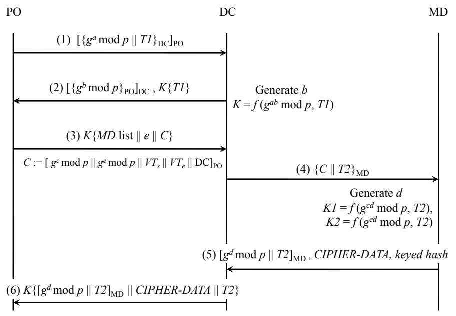
PO
DC
MD
(1) $[\{g^a \bmod p \parallel T1\}_{DC}]_{PO}$
(2) $[\{g^b \bmod p\}_{PO}]_{DC}$, $K\{T1\}$
Generate b
$K = f(g^{ab} \bmod p, T1)$
(3) $K\{MD \text{ list} \parallel e \parallel C\}$
$C := [g^c \bmod p \parallel g^e \bmod p \parallel VT_s \parallel VT_e \parallel \mathrm{DC}]_{PO}$
(4) $\{C \parallel T2\}_{MD}$
Generate d
$K1 = f(g^{cd} \bmod p, T2)$,
$K2 = f(g^{ed} \bmod p, T2)$
(5) $[g^d \bmod p \parallel T2]_{MD}$, *CIPHER-DATA, keyed hash*
(6) $K\{[g^d \bmod p \parallel T2]_{MD} \parallel CIPHER\text{-}DATA \parallel T2\}$

Fig. 2. The SELINDA Protocol

Following the hierarchical data collection model, we describe our protocol based on the two sessions that it consists of: the PO-DC session and the DC-MD session. Fig. 2 illustrates the complete data collection process.

A. PO-DC Session

The purpose of the PO-DC session is to provide to DC the information needed to be able to collect the data: the list of MDs that the DC has to collect data from, the public keys of the MDs and the DH half keys. The DH half keys will be used later for establishing the encryption/integrity keys between the PO and the MD, and between the DC and the MD. The session is composed of the following messages between the PO and the DC.

(1) PO to DC: $[\{g^a mod\ p||T1\}_{DC}]_{PO}$
$T1$ is the current timestamp of PO. DC checks whether $T1$ is close to its own time. To detect replay attacks, DC should keep the previous $T1$ received. The current $T1$ is accepted only when it is later than the previous one.

(2) DC to PO: $[\{g^b mod\ p\}_{PO}]_{DC}$, $K\{T1\}$
If $T1$ is valid, DC generates its DH half key $g^b mod\ p$ and computes the shared key K. K depends on $g^{ab} mod\ p$, $T1$, and the identity of DC. Recall that both the DC and the PO know the function to generate K, thus the PO can generate K after receiving $g^b mod\ p$. DC sends $K\{T1\}$ to authenticate itself to the PO, which the PO does by verifying $T1$ carried in $K\{T1\}$. If the authentication is successful, the PO can send to the DC the list of MDs, together with the public key information of the MDs.

(3) PO to DC:
$K\{\text{MD list}||e||C\}$ where
$C = [g^c mod\ p||g^e mod\ p||VT_s||VT_e||DC]_{PO}$
We call C a *token* from PO to DC for data collection. The token, and the DH half keys $g^c mod\ p$ and $g^e mod\ p$ inside, will be valid during the time period $[VT_s, VT_e]$, and VT_s must be later than $T1$. $g^c mod\ p$ is used for establishing shared keys between PO and MDs. $g^e mod\ p$, on the other hand, is for DC to establish a session key with the MDs. Note that e is contained in the message so that DC can retrieve it using K. It is important to note the fact that $g^e mod\ p$ is provided by the PO assures that a compromised DC cannot perform a small subgroup attack on the MDs' DH keys, as we will see in Section IV-A.

To be able to decrypt the data collected by the DCs, the PO has to remember one K per DC, but DH secret c can be the *same* for all DCs. Therefore, the memory requirement at the PO is very low. To further reduce the state information kept at the PO, it is possible to close the session after the DC has received the token, and resume the session when the DC is about to report data. Whether or not this is done does not affect the rest of the protocol.

B. DC-MD Session

The DC-MD session happens when the DC is able to communicate to one of the MDs, and its purpose is the actual data collection. It contains two messages between the DC and every MD, and one message from the DC to the PO.

(4) DC to MD: $\{C,T2\}_{MD}$ where
$C = [g^c mod\ p||g^e mod\ p||VT_s||VT_e||DC]_{PO}$
$T2$ is the current timestamp of the DC, and C is the token received in Step (3). When the MD receives the message, it verifies it by checking whether $T2$ is close to its current time and whether $T2$ falls in the range $[VT_s,VT_e]$. $T2$ should be also later than the previous timestamp used for the same purpose. The MD then generates its reply.

(5) MD to DC: $[\{g^d mod\ p\}_{PO}||T2]_{MD}$, *CIPHER-DATA*, keyed hash
Let *DATA* be the data measured by the MD in plaintext, i.e., the data to be collected. As mentioned earlier, we want to encrypt *DATA* in a way that only the PO can read it, but at the same time, we want to allow DC to check the integrity of *DATA*, so it can detect if an attacker between MD and DC tampers with the data. We achieve this by establishing one session key between the PO and the MD, and one session key between the DC and the MD. Both keys are established through DH, and to reduce complexity, the MD uses the *same* DH public key ($g^d mod\ p$) for both keys.
Given d (MD's DH private key), the MD can derive the session key $K1$ shared with the PO based on $g^{cd} mod\ p$ and based on $T2$. By using $T2$ to establish $K1$, even if the DH half keys are reused to save computation, $K1$ will be different in every DC-MD session. The MD can then use a standard mechanism to develop the encryption key and the integrity key from $K1$, e.g., using a hash function to hash $K1$ with other information as done in IPSec. Similarly, the MD obtains $K2$ based on $g^{ed} mod\ p$ and $T2$.
The MD then uses the keys derived from $K1$ to encrypt and to authenticate *DATA* towards the PO; we denote the result by *CIPHER-DATA*. Observe that *CIPHER-DATA* is *piggybacked* in the same message as the DH public key used to generate the session key $K1$. As we show later, this is important for security. Finally, the MD uses $K2$ as a key to generate a keyed hash of *CIPHER-DATA*.

Upon receiving the message from the MD, the DC verifies $T2$ from the signed message. DC can then compute $K2$ using $g^d mod\ p$ and $T2$, and can verify the integrity of *CIPHER-DATA*. If the integrity check is successful, DC encrypts *CIPHER-DATA* and $T2$ using the session key K it established with the PO in the PO-DC session, and send it to the PO. The PO uses $g^d mod\ p$ and $T2$ to compute $K1$ and the encryption and integrity keys needed to decrypt and to validate *CIPHER-DATA*. The data collection process is complete once the PO receives and validates *DATA*.

C. Protocol complexity

We now study the number of expensive operations the MD has to perform every time data are collected. Both public key operations and DH operations are regarded as expensive, while shared key operations and hash operations are not. Upon receiving Message 4 in Fig. 2, MD has to perform a public key decryption to retrieve C and $T2$. To verify it is PO who has signed C, a signature verification operation is needed. Three DH operations are needed to generate $g^d mod\ p$, $g^{cd} mod\ p$, and $g^{ed} mod\ p$. Finally, a signature operation is needed to sign $g^d mod\ p$ and $T2$ in Message 5. There are altogether three public key operations and three DH operations.

It is worth noting that the DH operations become unnecessary if we reuse the DH keys. Suppose after the first data collection, MD keeps $g^d mod\ p$, $g^{cd} mod\ p$, and $g^{ed} mod\ p$ in its memory. PO also keeps c and e. In the next collection, PO can send $[\text{“}REUSE\ DH\text{”}||VT_s||VT_e||DC]_{PO}$ as a token to DC. When MD receives the token, it does not have to generate a new d or recompute $g^{cd} mod\ p$ and $g^{ed} mod\ p$. Note that because $T2$ in the new session must be different from the last one, $K1$ and $K2$ will be different from the last session even though the DH keys are the same. In principle, reusing c several times could make it easier for an attacker to guess c. In the following, we show that for SELINDA this is not true.

IV. Security Analysis

In this section we show that the use of a single DH public key $g^c mod\ p$ to agree on a session key with many MDs (c.f. Fig. 2) does not make the protocol subject to simultaneous small subgroup attacks [15] because of the fact that MDs have to *piggyback* data with their public keys in Message 5. We consider an attacker that compromises a set $\mathscr{M}^C$ of MDs with the goal of obtaining PO's DH secret c. It does so by modifying the MDs' public DH keys sent to the PO.

We start with introducing the necessary notation. Let us denote by $G(p)$ the multiplicative group of integers modulo p, where p is a prime, and by $G_{q_i}(p)$ a subgroup of $G(p)$ of order q_i, where q_i, $0 \le i \le I$ is the i^{th} prime factor of $p-1$, and $p-1 = \prod_{i=0}^{I} q_i$. For some computational power B, let $\mathscr{I}_B = \{i : \log_2 q_i \le B\}$ be the set of small prime factors, and let $\mathscr{I}_B^* \subseteq \mathscr{I}_B$ be the least cardinality set of small subgroups for which $\sum_{i\in\mathscr{I}_B^*} \log_2 q_i \ge \log_2 q_0 - B$. Let q_0 be the large prime factor of $p-1$ used for DH key agreement, and let g be the generator of the subgroup $G_{q_0}(p)$. Congruence is defined using “mod p” unless otherwise noted.

We first explain the attack without piggybacking, we then show how *piggybacking* in Message 5 mitigates the attack.

A. Mitigation without piggybacking

Without piggybacking every compromised MD $m \in \mathscr{M}^C$ generates a DH public key $D_m = \beta_m g^{d_m}$, where $\beta_m \in G_{q_i}(p)$ for some $i > 0$. Since q_i and q_0 are relative primes $D_m \in G_{q_0 q_i}(p)$, the product group of $G_{q_0}(p)$ and $G_{q_i}(p)$. Assume that the PO uses D_m to compute the session secret $K_m^{PO} = (D_m)^c = \beta_m^c (g^d)^c$ and then uses a key derived from K_m^{PO} to encrypt or to authenticate data sent to the compromised MDs. If q_i is small, i.e., $i \in \mathscr{I}_B$, then the attacker can use a brute force attack to compute $c \, mod \, q_i$. This reveals approximately $\log_2 q_i$ bits of information about c. The attacker could perform $\min(|\mathscr{M}^C|, |\mathscr{I}_B^*|)$ attacks simultaneously with different small primes, and once it performs the attack $\forall i \in \mathscr{I}_B^*$ it could recover $c \, mod \, p$ using the Chinese remainder theorem [15].

In general, small order subgroup attacks can be mitigated in two ways. First, by choosing p such that $p-1$ does not have small prime factors, hence $\mathscr{I}_B = \emptyset$. Second, by verifying that the received DH public key has order q_0, i.e., $\forall m \; (D_m)^{q_0} = 1 (mod \; p)$. If there are thousands of MDs, this verification is computationally demanding. In the following we analyze the trade-off between computational load and security by introducing subset order verification defined as follows.

Definition 1: Subset Order Verification (SOV) is the following procedure. Pick $1 \le V \le |\mathscr{M}|$, the number of verification sets. Assign every MD to one of V verification sets with probability $1/V$. Let $\mathscr{M}_v \subseteq \mathscr{M}$, $1 \le v \le V$ be the verification sets. Verify $(\prod_{m \in \mathscr{M}_v} D_m)^{q_0} = 1 \pmod p$ for $\mathscr{M}_v \neq \emptyset$.

To prove our results, we introduce the following notation. For a non-empty verification set $\mathscr{M}_v \subseteq \mathscr{M}$ let $\mathscr{M}_v^C = \mathscr{M}^C \bigcap \mathscr{M}_v$ be the compromised MDs in the verification set. Let $\mathscr{M}_{v,i}^C = \{m \in \mathscr{M}_v^C : D_m \in G_{q_0 q_i}(p)\}$ be the set of compromised MDs that use subgroup $G_{q_i}(p)$. Define $\beta_i = \prod_{m \in \mathscr{M}_{v,i}^C} \beta_m$ if $\mathscr{M}_{v,i}^C \neq \emptyset$, and $\beta_i = 1$ if $\mathscr{M}_{v,i}^C = \emptyset$.

Lemma 1: The following statement holds:

$$(\prod_{m \in \mathscr{M}_v} D_m)^{q_0} = 1 \pmod p \iff \forall i, \quad \beta_i = 1 (\bmod p).$$

Proof: Here we prove "⇒" of the statement, as the other direction is straightforward. Since $(g^{d_m})^{q_0} = 1 (mod \; p)$ we have

$$(\prod_{m \in \mathscr{M}_v} D_m)^{q_0} = \prod_{m \in \mathscr{M}_v} (g^{d_m})^{q_0} \prod_{m \in \mathscr{M}_v^C} (\beta_m)^{q_0} = \prod_{i=0}^{I} (\beta_i)^{q_0}.$$

Observe that $\beta_i \in G_{q_i}(p)$, and the q_i are pairwise relative primes and are relative primes with q_0. Thus the product of the groups $G_{q_i}(p)$ is isomorphic to a group $G_{q'}(p)$, whose order is $q' = \prod_i q_i$. This implies that $\prod_{i=0}^{I} (\beta_i)^{q_0} = 1 \iff \forall i, \; (\beta_i)^{q_0} = 1$. Due to the relative primeness of q_0 and q_i we also have $(\beta_i)^{q_0} = 1 \iff \beta_i = 1 (mod \; p)$, which proves the lemma. ■

Lemma 2: Let g_i be an arbitrary generator of $G_{q_i}(p)$, and let γ_m be such that $\beta_m = g_i^{\gamma_m}$. Then, $\beta_i = 1 (\bmod p)$ implies $\sum_{m \in \mathscr{M}_{v,i}^C} \gamma_m = 0 (\bmod q_i)$.

Proof: Since q_i and q_0 are relative primes, the result is immediate. ■

The PO performs the verification $(\prod_{m \in \mathscr{M}_v} D_m)^{q_0} = 1 \pmod p$ for every non-empty verification set $\mathscr{M}_v \subseteq \mathscr{M}$. Since $\forall m \in \mathscr{M}^C$, $\beta_m^{j_m} \neq 1 \pmod p$, by Lemma 2 the attacker has to ensure that if $|\mathscr{M}_{v,i}^C| > 0$, then $|\mathscr{M}_{v,i}^C| \ge 2$ and furthermore $\sum_{\mathscr{M}_{v,i}^C} \gamma_m = 0 (\bmod q_i)$. As a consequence, an attacker that compromises one MD to perform a small subgroup attack for a subgroup will be detected with probability one by SOV. Also, if c is only used once, then an attacker cannot compute c if $|\mathscr{M}^C| < 2|\mathscr{I}_B^*|$.

Otherwise, it is easy to see that it is optimal for the attacker to use two compromised MDs per subgroup, and in this case the probability that the compromised MDs using the same subgroup fall in the same verification group is $1/V$. Hence the following results can be obtained.

Proposition 1: If the attacker wants to use the set $\mathscr{I}_B^*$ of small subgroups for the attack, the probability of not being detected is

$$(1/V)^{|\mathscr{I}_B^*|} \le (1/V)^{\frac{\log_2 q_0}{B} - 1} = V \, (1/q_0)^{\log_2(V)/B}. \qquad (1)$$

This probability can be unacceptably high for small V.

In general, the small subgroup attack can be launched by an attacker that establishes a shared secret using DH with the target of its attack, and its target would send messages to the attacker using the established shared secret. In our protocol, DC and MD establish a shared key using DH, and MD sends the key-hash of *CIPHER-DATA* to DC. A compromised DC could thus perform a small subgroup attack on the MDs. To avoid this, instead of allowing a DC to pick its DH half key $g^e mod \; p$, the key is assigned by the PO in Message 3. Our protocol is thus secure against compromised DCs.

B. Mitigation with piggybacking

Let us consider now SELINDA: every MD has to use its session secret to encrypt or to authenticate data sent to the PO. We assume that the attacker can detect, e.g., based on the application layer behavior, whether or not the PO was able to decrypt the message sent by a compromised MD.

To perform the attack, the attacker would still send $D_m = \beta_m g^{d_m} \in G_{q_0 q_i}(p)$ to the PO, but it would use $K_m = \beta_m^{j_m} (g^c)^{d_m}$ for some $1 \le j_m < q_i$ to establish the session secret used for encrypting or authenticating the message. If the PO was able to decrypt the message despite the compromised D_m, then the attacker knows that the session secrets are identical, i.e., $K_m^{PO} = (D_m)^c = \beta_m^{j_m} (g^c)^{d_m} = K_m$, which means $c = j_m \bmod q_i$. Thus, the attacker essentially has to guess $c = j_m \bmod q_i$. If the PO uses SOV, then due to Lemma 2 the attacker has to make two guesses per subgroup, and thus it can perform the guessing simultaneously for up to $\lfloor |\mathscr{M}^C|/2 \rfloor$ subgroups $G_{q_i}(p)$. The distribution of the number of trials until success is uniform on $\{1, (q_i - 1)/2\}$. We can obtain the following result by combining with SOV in Definition 1.

Proposition 2: The probability of an attacker guessing $c = j_m \bmod q_i$ before being detected by SOV is given by and upper bounded by

$$\frac{2}{q_i - 2} \sum_{k=1}^{(q_i-1)/2} \frac{1}{V^k} \le \frac{2}{q_i - 2} \left(\frac{1}{1 - 1/V} \right). \qquad (2)$$

It can be seen that (2) is comparable to the probability of guessing $c = j_m \bmod q_i$ in one shot. Hence there is small but non-zero probability that the attacker can compute c despite SOV. In principle, if c is reused, then on average after $\frac{1}{2|\mathscr{M}^C|} \sum_{i \in \mathscr{I}_B^*} q_i$ uses of c the attacker can compute c.

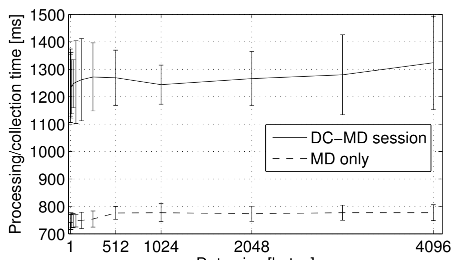

Fig. 3. Time Performance: total time of computation in MD and total time for data collection for messages of different sizes

Observe, however, that the PO can detect a bad guess $j_m \neq c \bmod q_i$ because it fails to decrypt the message received from m. If this happens, the PO can verify $(D_m)^{q_0} = 1$. For a bad guess not to happen before c is computed, the attacker has to correctly guess $j_m = c \bmod q_i$ for the first try for all $\mathscr{I}_B^*$ prime factors, which happens with probability proportional to $\prod_{i \in \mathscr{I}_B^*} 1/q_i \approx 2^B/q_0$, which is comparable to directly guessing c itself. Interestingly, the probability of a successful attack does not increase thanks to piggybacking although c is reused by the PO. Thus, *piggybacking* in Message 5 in SELINDA provides strong security against compromised MDs without the need for individual group order verification.

V. Numerical results

To further understand the computational performance of the SELINDA protocol, we measure the time needed for DC to collect data from MD. The MD and DC are simulated using two laptops with Intel Core i5 2.4GHz processor and 4GB read-only memory. The prototype is implemented in Java. The laptops communicate through a Wifi 802.11n wireless network. We use RSA with key length 2048 bits as the public-key cryptography and AES with key length 256 bits for symmetric key cryptography. We use SHA-256 to compute hash-based message authentication (HMAC), and use Diffie-Hellman with a prime order of length 1024 bits.

To understand the computational complexity on the MDs, we have measured the time needed to decrypt Message 4 and create Message 5 (c.f., Fig. 2). We have also measured the total time needed for the DC to collect data from a MD in the DC-MD session, which includes the time it takes to create Message 4, the round-trip network delay to send and receive Messages 4 and 5, as well as the time required to verify the integrity of *CIPHER-DATA*. Fig. 3 shows the measurement results as a function of the size of *DATA*. Each data point is the average time of 30 different trials, and we show the 95 percent confidence intervals.

For MD, the difference in computational time between data sizes of 1 byte and 4096 bytes is less than 60ms, and the difference between 1K and 4K is less than 1ms. We can conclude that our mechanism scales well for typical amounts of data to be reported in the considered smart grid context. The average total times it takes for the DC to collect data of sizes 1 byte and 4K bytes are 1241ms and 1324ms, respectively. These results help to explore feasible mobility patterns for a mobile DC. In a scenario where an automobile is used as a DC. An automobile moving at 50km/h can advance less than 20m in this amount of time. Since the communication range of 802.11 is around 250m, even a mobile DC should have enough time to complete the data collection as long as it is not very far from the MD. We thus conclude that the proposed SELINDA protocol fulfills the goal of being scalable and efficient.

VI. Conclusion

In this paper, we develop a secure, scalable, and light-weight protocol for smart grid data collection. Our protocol allows a measuring device to report data securely to the power operator via a data collector that may not be trustworthy. It is thus suitable for data collection using mobile data collectors, and can be used for community-aided data collection. We provided a formal analysis of the security of the protocol. We implemented the protocol and provided measurement results that show that the protocol indeed has low computational complexity and makes mobile smart grid data collection possible.

Acknowledgment

The work was funded in part by the TCIPG funding of the US DoE and by the Small Project Funds of the University of Hong Kong. Dán was a Fulbright research scholar at UIUC while the work was done.

References

[1] J. Zhou, Q. Hu, and Y. Qian, “Scalable distributed communication architectures to support advanced metering infrastructure in smart grid,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, Sep. 2012.

[2] N. Saputro and K. Akkaya, “Performance evaluation of smart grid data aggregation via homomorphic encryption,” in *Proc. of IEEE WCNC*, 2012.

[3] 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.

[4] T. Khalifa, K. Naik, M. Alsabaan, A. Nayak, and N. Goel, “Transport protocol for smart grid infrastructure,” in *Proc. of IEEE International Conference on Ubiquitous and Future Networks*, 2010.

[5] 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.

[6] B. Vaidya, D. Makrakis, and H. T. Mouftah, “Authentication and authorization mechanisms for substation automation in smart grid network,” *IEEE Network*, pp. 5–11, Jan/Feb 2013.

[7] 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.

[8] G. Dán, H. Sandberg, G. Björkman, and M. Ekstedt, “Challenges in power system information security,” *IEEE Security & Privacy Magazine*, vol. 10, no. 4, 2012.

[9] M. M. Fouda, Z. M. Fadlullah, N. Kato, R. Lu, and X. Shen, “A lightweight message authentication scheme for smart grid communications,” *IEEE Transactions on Smart Grid*, vol. 2, no. 4, Dec. 2011.

[10] 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.

[11] Y. Law, G. Kounga, and A. Lo, “WAKE: Key management scheme for wide-area measurement systems in smart grid,” *IEEE Communications Magazine*, January 2013.

[12] N. Liu, J. Chen, L. Zhu, J. Zhan, and Y. He, “A key management scheme for secure communications of advanced metering infrastructure in smart grid,” *IEEE Transactions on Industrial Electronics*, to appear.

[13] *IEEE 1815-2012*, “Dnp3 secure authentication version 5,” 2011.

[14] *RFC 5246*, “The transport layer security (tls) protocol version 1.2,” 2008.

[15] C. Lim and P. Lee, “A key recovery attack on discrete log-based schemes using a prime order subgroup,” in *Proc. of CRYPTO*, 1998.