Secure and Scalable Data Collection With Time Minimization in the Smart Grid

Suleyman Uludag, *Member, IEEE*, King-Shan Lui, *Senior Member, IEEE*, Wenyu Ren, and Klara Nahrstedt, *Fellow, IEEE*

***Abstract*—Deployment of data generation devices such as sensors and smart meters have been accelerating toward the vision of smart grid. The volume of data to be collected increases tremendously. Secure, efficient, and scalable data collection becomes a challenging task. In this paper, we present a secure and scalable data communications protocol for smart grid data collection. Under a hierarchical architecture, relay nodes [also known as data collectors (DCs)] collect and convey the data securely from measurement devices to the power operator. While the DCs can verify the integrity, they are not given access to the content, which may pave the way for third party providers to deliver value-added services or even the data collection itself. We further present optimization solutions for minimizing the total data collection time.**

***Index Terms*—Data collection with time minimization in the smart grid, scalable data collection in the smart grid, secure data collection in the smart grid.**

I. Introduction

IN THE smart grid, a massive number of sensors or measurement devices (MDs) will be installed to collect real-time information. The generated data should be collected in a secure, efficient, and scalable manner. To make it scalable, a hierarchical data collection framework is usually adopted. For example, in advanced metering infrastructure (AMI), smart meters first report measurements to data concentrators [1]. Thereby, the power operator (PO) does not have to maintain a separate, expensive connection with each smart meter. Apart from data collection, this hierarchical communication structure should also allow the PO to send an instruction to the devices. To maintain fast response, messages, data, or instructions, should be delivered efficiently and as fast as possible. The messages should be protected to prevent from information leak and launch of attacks. In this paper, we develop a comprehensive protocol that allows a PO to collect data, as well as send commands to MDs in a secure, scalable, and efficient manner.

Fig. 1 presents the data collection architecture considered in this paper. The MDs are sensors or smart meters that generate power-grid specific data. They are small telemetric devices and computationally constrained. Each MD is connected to at least one data collector (DC), and each DC may connect to multiple MDs. The PO has a direct connection with each DC. PO and DCs are relatively more powerful than MDs. The data are reported to PO via a set of DCs. PO may also issue commands to the MDs via the DCs. Theoretically, a DC is trustworthy if it is within the security domain of the PO.

However, due to the massive number of MDs and their dispersion over a large area, it may not be appropriate to assume DCs can be completely trusted. In addition, one of the seven actors identified by the National Institute of Standards and Technology (NIST) in the smart grid framework [2] is third party service providers, which are to furnish value-added services. We assume honest-but-curious model for DCs. Thus, the data collection tasks may be outsourced to third party service providers [3]. Besides, the benefits of cloud computing [4] may be accrued for storage and processing of the data collected. Data sharing to others to provide services like energy management services can be facilitated as well.

In some other applications [5], 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 do not have access to them. In other words, each MD should encrypt its data using an appropriate key to keep its data private from DCs and other possible adversaries. On the other hand, due to limitation in memory and computational capability, the encryption algorithm used should be efficient. PO should also protect its commands appropriately. Apart from ensuring the security of these commands, it is also crucial to deliver data and commands promptly because fast actions of MDs are necessary to maintain the stability and health of the smart grid. Because the network delay for different DCs to collect data from a certain MD would be different, to make the data collection more efficient, we also study how to assign DCs to MDs to minimize the time for data collection.

Our contributions in this paper may be summarized as follows.

1) Under a hierarchical infrastructure, we have proposed a scalable, secure, and lightweight data collection scheme in the smart grid.

Manuscript received July 16, 2014; revised December 8, 2014; accepted January 15, 2015. This work was supported by the Department of Energy under Award DE-OE0000097. Paper no. TSG-00726-2014.

S. Uludag is with the Department of Computer Science, Engineering, and Physics, University of Michigan–Flint, Flint, MI 48502 USA (e-mail: uludag@umich.edu).

K.-S. Lui is with the Department of Electrical and Electronic Engineering, University of Hong Kong, Hong Kong.

W. Ren and K. Nahrstedt are with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61820 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSG.2015.2404534

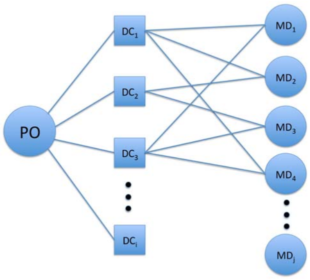


Fig. 1. Hierarchical data collection structure.

2) Our secure data collection scheme does not assume trusted DCs, rather they are considered to be honest-but-curious entities. This may pave the way for outsourcing the data collection and third party service providers as envisioned by the NISTs smart grid framework [2].
3) We have coupled the secure date collection with an optimization problem formulation with the objective function of time minimization, which is a first in the literature as a joint security and optimization approach.
4) We have shown the NP-hardness of the problem and developed an efficient heuristic solution.
5) We have also provided a solution to the assignment of DCs to MDs as part of the optimization framework.

The rest of this paper is organized as follows. Section II describes existing efforts on data collection in smart grids. We provide the system and protocol overview in Section III. The details of the protocol are described in Sections IV and V. In Section VI, we analyze the time performance of our mechanism and present the DC-MD assignment problem as an optimization problem. We conclude this paper in Section VIII.

II. Related Work

Data integrity and confidentiality of end-to-end data have been studied extensively in the Internet. However, most schemes, such as TLS [6], assume 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 [7].

Distributed network protocol (DNP3) [8] is a standard communications protocol used in supervisory control and data acquisition, the data collection subsystem of power grids. 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 the International Electrotechnical Commission (IEC) 61850 [9]. Yet, IEC 61850 was also initially designed without security mechanisms [10]. It is thus generally agreed by the experts that new security protocols for data collection and command delivery of smart grids need to be developed.

Our proposed approach comprises security aspect of the smart grid data collection as well as the time minimization. In what follows, we provide synopses of related work from these aspects as well as with respect to a variety of other relevant subtopics of our holistic approach.

Collecting data by means of transport layer protocols from a massive number of MDs has been studied in the literature as an option. Kim *et al.* [11] studied how to reduce the storage needed when the control center needs to establish multiple sessions with the MDs. Long-term shared keys are generated by a function so that the control center only needs to memorize the function but not individual keys. Nevertheless, the key developed this way is not very secure. Besides, the protocol is not suitable for the hierarchical data collection architecture. Data collection through a DC is considered in [12]. The authors propose to maintain two separate transmission control protocol (TCP) connections, and the two connections can be protected using different mechanisms independently. Nevertheless, the DC is assumed to be trustworthy, that is, it can read the data sent by the MD.

Another important aspect of secure data collection is concerned with key management. Many assume trusted DC, that is, they do not consider hiding the data from the DCs, such as [13]–[15]. Some others assume a direct connection between the PO and MDs [16]–[18] and some others are not suitable for hierarchical data collection as we consider in this paper [16], [17], [19].

The SAKE protocol [20] allows two neighboring sensor nodes to establish keys using hash chains. However, the authors assume the attackers are of limited computational capability as another sensor. Wu and Zhou [15] applied the elliptic curve public key technique to perform key management. Mutual authentication between different entities is 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. Law *et al.* [18] described how to establish keys and secure unicast and multicast communications. Kim *et al.* [7] proposed long-term keys to be given to the different parties for protecting messages. Fouda *et al.* [16] described how to apply the Diffie–Hellman (DH) mechanism to establish a shared key for data authentication between two parties. Reference [17], 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. Nicanfar and Leung [21] described how a device establishes shared keys with different controllers at different hierarchical levels. However, it is assumed that a shared key exists between two adjacent controllers. Another approach presented in [22] is based on symmetric cryptography to provide data confidentiality and authentication between sensors and the base station. Again, a master key is assumed with a preagreed pseudo-random function in the scheme.

Another category for providing security and privacy [23] exploits the aggregate statistics of the sensed data, such as summation, average, minimum, maximum, etc. These approaches take advantage of in-network data processing (also referred to as aggregation) to induce some obfuscating operations on the transmitted data [24]–[31]. Examples of this category include cluster-based private data aggregation [24] and its integrity enhanced version [25], secret perturbation [26], *k*-indistinguishable privacy-preserving data aggregation [27], a centralized authentication server based in-network aggregation for AMI [28], [29], a secure architecture for distributed aggregation of additive data [30], and a network coding-based encryption between smart meters and aggregators [31]. Unlike these techniques, our problem formulation does not assume any statistical property for in-network processing and deliver the MD data unaltered to the PO. There are also data aggregation schemes without any security schemes, such as [32] and [33].

There are also homomorphic encryption-based approaches to hide the collected information from the MDs, such as [34]–[39]. However, homomorphic encryption is a compute-intensive operation.

Hur [40] studied how data generators report data to a honest-but-curious storage center for a user to retrieve later. To the best of our knowledge, 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 it is 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, this paper suggests to use expensive attribute-based and public key encryption to protect data to incorporate policy consideration. The experimental computational time for a decryption on a message of size less than 1000 bits in a low-end smart meter (TinyPBC library on a 32-bit ARM XScale PXA271 processor) is around 140 ms, while the encryption is supposed to be a few times more expensive. Our protocol, on the other hand, encrypts data using the much more light-weighted symmetric key cryptography, which is more suitable for computationally-constrained MDs.

There are some approaches with optimization for the data collection process. Cost minimization of data collection by means of wireless channel selection and transmission scheduling has been reported in [41]. A delay minimization of overhead transmission lines over unreliable wireless links is studied in [42]. These and other similar approaches lack any security mechanism as part of their approaches, unlike our proposal in this paper.

An interesting approach in [43] considers the tradeoff between the strength of security and energy consumption jointly for both phasor measurement unit and AMI data over energy-constrained devices. However, only a generic comparison of different cryptographic algorithms over CrossBow and Ember sensor platforms is reported without any attention to the overall data collection scheme.

To the best of our knowledge, no other paper in the literature appears to be proposing a holistic approach for hierarchical data collection with curious-but-honest DCs with a joint goal of minimizing the overall data collection time and assignment of MDs to DCs. Further, we also perform experiments to study the time performance of our mechanism.

III. System and Protocol Overview

A. Operations and Their Requirements

TABLE I
System Operations and Their Requirements

	Operation	Security Requirement	Time Requirement
Op 1	PO initiates data collection of all MDs or a group of MDs	Data reported should be authenticated and should be read only by the PO, not by other MDs or any DC	The total time to collect all data should be minimized
Op 2	PO requests data from a certain MD	Same as Op 1	The time needed should be kept minimal
Op 3	MD initiates an urgent data report	Same as Op 1	The data should be delivered to the PO as soon as possible
Op 4	PO issues an urgent command to a group of MDs	The command should be authenticated appropriately	Time for each MD to receive and read the command should be minimized

As mentioned in Section I, our communication architecture supports MDs to report data and PO to deliver commands in a timely and secure manner. Table I describes each operation. Op 1 is a regular call-for-data from the PO which is performed periodically. Op 2 is performed when PO detects something abnormal and would like a data report from a particular MD. Time is more critical than a regular data reporting. Op 3 is done when MD detects something abnormal and would like to report to the PO. Op 4 is issued when PO needs a group of MDs to perform a certain action as soon as possible.

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: 1) PO; 2) DCs; and 3) MDs. They are all given with a corresponding pair of public and private key pair (see Section III-B). Similar to other secure communication protocols, we basically assume a signature, which is created by the private key, can be an identity authenticator. That is, if somebody can prove that she has the knowledge of Alice's private key, we assume this person is Alice. If the working environment is very insecure that even private keys could be stolen easily, our protocol does not work. The system, in this case, probably require another form of authentication such as token-based, bio-metric, instead of key-based.

We assume the PO (the entity that possesses the private key of the PO) is always trustworthy because it is the control of the whole system and it decides how to use the data collected. The DCs, on the other hand, are honest-but-curious that they would follow the protocol as specified but would like to read the data and share with others if they could. That is, they would not impersonate another entity in the system, nor actively tamper the data, but would like to learn as much as possible based on the information they can access according to the normal operation of the protocol. As the MDs are devices located in the field (for example, on power grid poles), it is not likely they are under the same physically secure environment as the PO. The chance that leaking private keys is higher. When an attacker gets the private key of a certain MD, it can report fake data to the PO on behalf of the MD. The PO can analyze the data reported to detect whether they were legitimate data or not. Even though this attacker can report fake data on behalf of its victim, it cannot impersonate other entities (other MDs, PO, or DCs) and read the data sent by other MDs.

TABLE II
SYSTEM FUNCTIONS

Name	Description	Name	Description
$PKE(K_p,M)$	encrypt M using K_p	$PKD(K_p,C)$	decrypt C using K_p
$SKE(K_s,M)$	encrypt M using K_s	$SKD(K_s,C)$	decrypt C using K_s
$SIGN(A,M)$	sign of M by A	$SIGV(A,M)$	verify M signed by A

B. System Parameters

Before any communication, PO, DCs, and MDs are equipped with a set of system parameters. We assume necessary parameters are configured in a DC or MD before they are installed in the field.

1) Long-Term 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. Under normal circumstances, PO would not publish the public keys of DCs and MDs to the general public. However, our protocol is secure even if the attackers know the public key information of any DC or MD they want to attack.

2) DH Parameters: We adopt the DH key exchange mechanism to develop shared keys between two parties. Due to space limitation, we refer readers to [44] for the details. Generally speaking, DH allows the two parties to develop a secret shared key even eavesdroppers can read the half keys they exchange with each other. Through forgetting half keys and shared keys appropriately, DH keys also support perfect forward secrecy.

C. 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 the advanced encryption standard (AES) for symmetric key encryption and secure hash algorithm with 256 bits of key length (SHA-256) for hash computation. Table II summarizes the functions used in the protocol. In the table, PKE is public key encryption, PKD is public key decryption, SKE is symmetric key encryption, SKD is symmetric key decryption, SIGV is signature verification, K_p is a public key while K_s is a shared key.

Some cryptographic functions run much slower than others. As some smart grid operations are time sensitive, it is very crucial to identify efficient cryptographic functions appropriately to protect the communication. To further understand the computational time of the cryptographic functions on computationally constrained devices, we measure the time needed to execute some representative cryptographic functions on Raspberry Pi. Raspberry Pi is a tiny computer with a size similar to a credit card. The CPU is 700 MHz and the memory available is 512 MB. Due to space limitation, we only present some of the results. More details can be found in [45].

Table III presents the time needed to create an Rivest–Shamir–Adleman cryptosystem (RSA) signature and verify an RSA signature using different key sizes. The time spent on encrypting a message using public key is similar to the time needed in verifying a signature. The time needed on decrypting a message using private key is similar to the time needed on signature creation. It can be observed that the time needed does not grow with message size but with key size. Column ratio in the table gives the time ratio of signature computation/signature verification. The time spent on a private key operation (signing a message) is much longer than that on a public key operation (verifying a signature). An efficient protocol should not require MDs to sign a lot of messages, especially when a long RSA key is used.

TABLE III
RSA COMPUTATIONAL TIME

RSA	1024 bits			3072 bits		
Message Size (bits)	Sign. (ms)	Ver. (ms)	sign/ver ratio	Sign. (ms)	Ver. (ms)	sign/ver ratio
128	64.01	3.91	15.12	1048.37	11.49	91.23
256	64.97	4.01	16.19	1033.46	11.65	88.68
512	64.27	4.00	16.08	1047.96	11.69	89.67

We also measured the time needed to generate different DH keys with different key sizes [45]. A DH shared key generation is more expensive than an RSA signature verification. It implies that it may not be appropriate to regenerate DH shared key for each data collection instance. By adopting different cryptographic functions and techniques carefully based on their security features and computational complexities, our protocol facilitates efficient and secure data collection.

D. Protocol Overview

To detect replay attacks, we adopt a similar way as the widely used Kerberos protocol (RFC4120) that uses timestamps. The parties who talk directly should first synchronize their clocks. When a timestamp is included in a message, the receiver should check whether the carried timestamp is within a certain amount of difference from its local clock. The default threshold in Kerberos is 300 s. In our protocol, the threshold would depend on the expected delay in transmitting the message and granularity of clock synchronization.

Because encrypting data using public key cryptography is very expensive, before any data collection, we should first develop shared keys among PO, DCs, and MDs 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 same applies to DC with each MD it will talk to. Apart from pairwise keys, to facilitate a certain command or instruction to be delivered to a group of MDs in a secure and efficient manner, we also develop a set of group keys that each group key is shared between the PO, a DC, and the MDs that connect to that DC.

The PO initiates the shared key generation process to establish the necessary pairwise shared keys and group keys. We adopt the DH key exchange mechanism to develop all pairwise shared keys. We authenticate the DH half keys using the long-term public keys to avoid the man-in-the-middle attack.

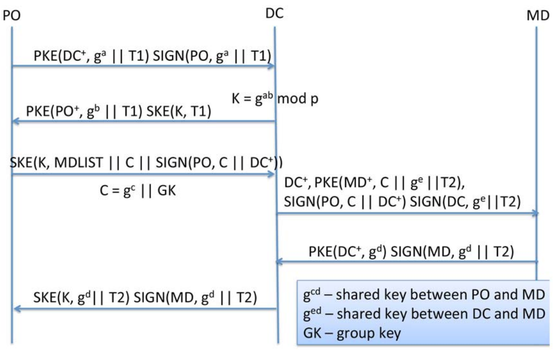

PO
DC
MD
PKE(DC⁺, gᵃ || T1) SIGN(PO, gᵃ || T1)
K = gᵃᵇ mod p
PKE(PO⁺, gᵇ || T1) SKE(K, T1)
SKE(K, MDLIST || C || SIGN(PO, C || DC⁺))
C = gᶜ || GK
DC⁺, PKE(MD⁺, C || gᵉ ||T2),
SIGN(PO, C || DC⁺) SIGN(DC, gᵉ||T2)
PKE(DC⁺, gᵈ) SIGN(MD, gᵈ || T2)
SKE(K, gᵈ|| T2) SIGN(MD, gᵈ || T2)
gᶜᵈ – shared key between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

Fig. 2. Initial shared key generation.

Once the pairwise shared keys and group keys are established, they will be used for data collection and command delivery.

As DH operations are expensive, we should not regenerate 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 instance depends on both the DH shared key and the timestamp. As the timestamp changes for every data collection instance, the data encryption key will be changed even though we do not regenerate the DH shared key. In the following, we will first describe the shared key generation process in Section IV. The detailed message exchanges of the four operations mentioned in Section III-A will be provided in Section V.

IV. Shared Key Generation

Let the set of MDs be $\mathbb{MD}$ and the set of DCs be $\mathbb{DC}$. Before the PO initiates the process, PO has to assign a set of MDs for DC to connect to. We let $\text{MDLIST}_i \subseteq \mathbb{MD}$ be the set of MDs that are assigned to DC_i. Definitely, $\cup_{\text{DC}_i \in \mathbb{DC}} \text{MDLIST}_i = \mathbb{MD}$. However, $\text{MDLIST}_i \cap \text{MDLIST}_j$, where $i \neq j$, may not necessarily be $\emptyset$. It is possible that PO would like multiple DCs to collect data from the same MD to enhance reliability. In fact, different assignments between MDs and DCs would differ in data security, cost, and data collection time. In Section VI, we will formulate the assignment problem to minimize the data collection time.

In the rest of this paper, for the ease of discussion, we use shared key to refer to pairwise shared key. We further denote K_B^A as the shared key between A and B. We refer to the set $\{\text{PO}, \text{DC}_i\} \cup \text{MDLIST}_i$ 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 II.

Fig. 2 presents a summary of the initial 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 $K_{\text{MD}_j}^{\text{PO}}$, $K_{\text{MD}_j}^{\text{DC}_i}$, and GK_i if $\text{MD}_j \in \text{MDLIST}_i$. Through the procedure, DC_i knows GK_i, $K_{\text{DC}_i}^{\text{PO}}$ and $K_{\text{MD}_j}^{\text{DC}_i}$ for all $\text{MD}_j \in \text{MDLIST}_i$. The detailed procedure is as follows.

1) PO starts the key generation process. It first generates a DH secret a to talk to the DCs. PO also captures the current timestamp $T1$ and sends the following message to DC_i. $T1$ should be kept until the whole key generation process is done

$$\text{PO} \rightarrow \text{DC}_i : \text{PKE}\big(\text{DC}_i^+, g^a||T1\big), \text{SIGN}\big(\text{PO}, g^a||T1\big).$$

g^a is encrypted and so an eavesdropper cannot read g^a. Because PO signs $g^a||T1$ and $T1$ is a timestamp, an attacker cannot change $T1$ or g^a easily without being detected. Suppose an attacker also knows DC_i^+, although he can create $\text{PKE}(\text{DC}_i^+, g^{a'}||T1')$ using his own $g^{a'}$ and $T1'$, he cannot forge the signature $\text{SIGN}(\text{PO}, g^a||T1)$. To further enhance security, PO can use different a's for different DCs, but it has to generate different signatures for different a's and remember which is used for which DC.

2) When DC_i receives the message, it uses DC_i^- to retrieve g^a and $T1$. It checks whether the received signature $\text{SIGN}(\text{PO}, g^a||T1)$ is correct. If so, DC_i checks whether $T1$ is within an acceptable range. If so, it generates its DH secret b and computes K as $g^{ab} \text{mod } p$. K is then the shared key between PO and DC_i ($K_{\text{DC}_i}^{\text{PO}}$). It encrypts its public DH key (g^b) using POs public key and send it to PO. It also proves it knows K by providing $\text{SKE}(K, T1)$

$$\text{DC}_i \rightarrow \text{PO} : \text{PKE}\Big(\text{PO}^+, g^b\Big|\Big|T1\Big), \text{SKE}(K, T1).$$

The timestamp $T1$ is used to detect replay attack as mentioned in Section III-D. When a message is accepted, DC_i should record $T1$. When another message arrives that carries a time stamp T, T is accepted only when it is not exactly the same as $T1$ and the time difference between T and it local clock is acceptable. Note that an attacker, who does not know DC_i^-, cannot retrieve g^a and $T1$ from $\text{PKE}(\text{DC}_i^+, g^a||T1)$. Therefore, when PO receives a correct reply, he knows that it was DC_i who sent him the message.

3) When PO receives the message, it can retrieve g^b using PO^- to compute K. It then uses K to decrypt $\text{SKE}(K, T1)$ and retrieves $T1$. If this is the same as the one he sent earlier, PO can confirm that it was DC_i who sent the message. It then 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 for DC_i to talk to the MDs in the list. C contains g^c, which is used for establishing shared keys between PO and MDs, and GK_i, which is the group key of G_i. The public keys of the MDs should also be sent (we assume they are included in MDLIST_i in Fig. 2)

$$\begin{aligned}\text{PO} \rightarrow \text{DC}_i : \ & \text{SKE}(K, \text{MDLIST}_i||C|| \\ & \text{SIGN}\big(\text{PO}, C||\text{DC}_i^+\big)\big) \text{ where } C = g^c||\text{GK}_i.\end{aligned}$$

It is worth noting that PO also sends $\text{SIGN}(\text{PO}, C||\text{DC}_i^+)$ and further encrypts it using K. This allows DC_i to detect whether the message has been tampered. As C is encrypted using K, it should be safe from eavesdroppers. GK_i, which should be known to PO, DC_i, and those MD in MDLIST_i, is protected then.

4) When DC_i receives the message, it first uses K to retrieve C and $\mathrm{SIGN}(\mathrm{PO}, C||\mathrm{DC}_i^+)$. It verifies whether the signature is correct. If so, DC_i can then generate its DH half key, g^{e_i} for establishing shared keys with the MDs. DC_i also captures the current timestamp $T2$, which must be larger than $T1$, and sends the information to MD_j in MDLIST_i using the public keys provided. DC_i also needs to send its public key. To allow MD_j to detect whether the message has been tampered, two signatures, $\mathrm{SIGN}(\mathrm{PO}, C||\mathrm{DC}^+)$ and $\mathrm{SIGN}(\mathrm{DC}_i, g^{e_i}||T2)$, are sent as well

$$\begin{aligned}\mathrm{DC}_i \rightarrow \mathrm{MD}_j :\ & \mathrm{DC}_i^+, \mathrm{PKE}\left(\mathrm{MD}_j^+, C||g^{e_i}||T2\right)\\ & \mathrm{SIGN}\left(\mathrm{PO}, C||\mathrm{DC}^+\right), \mathrm{SIGN}\left(\mathrm{DC}_i, g^{e_i}||T2\right).\end{aligned}$$

As DC_i^+ is sent in plaintext and $C||g^{e_i}||T2$ is encrypted using the public key of MD_j, it is possible for an attacker to create its own DC_i^+, $\mathrm{PKE}(\mathrm{MD}_j^+, C||g^{e_i}||T2)$ and $\mathrm{SIGN}(\mathrm{DC}_i, g^{e_i}||T2)$. That is, let the attacker be AK. He can send AK^+, $\mathrm{PKE}(\mathrm{MD}_j^+, C'||g_{ak}^{e_i}||T2')$ and $\mathrm{SIGN}(AK, g_{ak}^{e_i}||T2')$ to MD_j, trying to pretend to be DC_i. However, he cannot forge PO to create $\mathrm{SIGN}(\mathrm{PO}, C'||AK^+)$ to cheat MD_j. This message is thus safe from message tampering and an attacker cannot impersonate DC_i.

5) Upon receiving the message, MD_j first decrypts $\mathrm{PKE}(\mathrm{MD}_j^+, C||g^{e_i}||T2)$ using his private key to retrieve $C||g^{e_i}||T2$. It then verifies the two signatures to ensure the message has not been tampered. It should also check whether $T2$ is acceptable in a similar way that DC_i verifies $T1$ to detect replay attacks. If the message passes the tests, MD_j creates a DH secret key d to establish the shared key between itself and PO ($K_{\mathrm{MD}_j}^{\mathrm{PO}}$), which is g^{cd}, and the shared key with $\mathrm{DC}_i(K_{\mathrm{MD}_j}^{\mathrm{DC}_i})$, which is $g^{e_i d}$. It then encrypts g^d using the public key of DC_i so that g^d is safe from eavesdroppers. It also signs g^d and $T2$ to defend against impersonation and replay attacks

$$\mathrm{MD}_j \rightarrow \mathrm{DC}_i : \mathrm{PKE}\left(\mathrm{DC}_i^+, g^d\right), \mathrm{SIGN}\left(\mathrm{MD}_j, g^d||T2\right).$$

If an attacker eavesdropped an earlier communication, it cannot simply replay the message from the previous session because $T2$ carried in the new message should be different. By signing g^d, we can defend against attackers who want to impersonate MD_j in replying to DC_i.

6) When DC_i receives the message, it decrypts $\mathrm{PKE}(\mathrm{DC}_i^+, g^d)$ using its private key and retrieves g^d. It can then verify the signature to detect whether the message has been tampered. If not, it sends g^d to PO by encrypting it using K. It also sends the signature by MD_j it received to PO

$$\mathrm{DC}_i \rightarrow \mathrm{PO} : \mathrm{SKE}\left(K, g^d||T2\right), \mathrm{SIGN}\left(\mathrm{MD}_j, g^d||T2\right).$$

As only DC_i and PO know K, only PO can read g^d in $\mathrm{SKE}(K, g^d||T2)$. By checking whether $T2$ is later than $T1$ kept in memory, PO can detect whether it is a replay. The signature of MD_j on g^d and $T2$ authenticates that it was MD_j who created g^d.

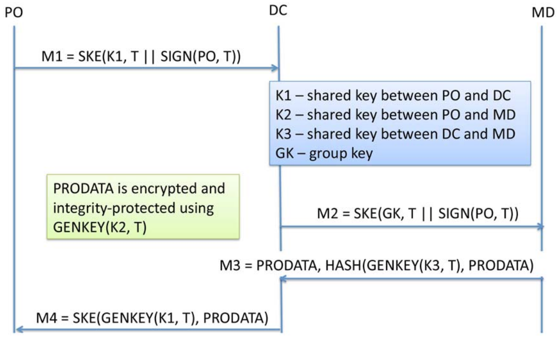

PO
DC
MD
M1 = SKE(K1, T || SIGN(PO, T))
K1 – shared key between PO and DC
K2 – shared key between PO and MD
K3 – shared key between DC and MD
GK – group key
PRODATA is encrypted and integrity-protected using GENKEY(K2, T)
M2 = SKE(GK, T || SIGN(PO, T))
M3 = PRODATA, HASH(GENKEY(K3, T), PRODATA)
M4 = SKE(GENKEY(K1, T), PRODATA)

Fig. 3. Data collection.

7) If $g^d||T2$ encrypted using K and signed by MD_j are the same, PO can assume the message has not been tampered. PO can then compute $K_{\mathrm{MD}_j}^{\mathrm{PO}}$ 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.

We now analyze the memory needed for each entity to keep the shared keys. The PO needs to keep a shared key for each DC, a shared key for each MD, and a group key for each group. The total number of keys is $2 \times |\mathbb{DC}| + |\mathbb{MD}|$. DC_i has to keep $K_{\mathrm{DC}_i}^{\mathrm{PO}}$, a shared key with each MD belongs to its group, and a group key. The total is $2 + |\mathrm{MDLIST}_i|$. For MD_j, for each group G_i it belongs to, it has to keep a shared key with DC_i and the group key GK_i. It is worth noting that MD_j can establish different shared keys with PO through different DCs. If PO provides different g^c's for different DCs, the shared keys developed via different DCs must be different. Even when PO provides the same g^c through different DCs, MD_j can also establish different shared keys by replying different g^d's for different DCs. Therefore, MD_j has to keep at most 3 × number of groups it belongs to keys in total. PO decides how many groups an MD is associated with and can thus establish keys according to the memory available in different MDs.

V. Data Collection and Command Delivery

A. *PO Initiates Data Collection of Group of or All MDs*

It is a regular data collection initiated by the PO. We want the data collection to be secure, scalable, and efficient. To ensure data confidentiality and integrity, data reported by MD_j is encrypted using $K_{\mathrm{MD}_j}^{\mathrm{PO}}$, a key shared by the PO and MD_j only. Our data collection protocol is scalable because a single DC would collect data from multiple MDs. PO no longer needs to establish a single session to each MD. To achieve efficiency, we do not require computationally-constrained MDs to perform a lot of expensive operations. We also reduce the number of messages exchanged. To further reduce the time of data collection, we study how to assign DCs to collect data from the MDs in Section VI. In the following, we first present the data collection procedure in a step by step manner. Fig. 3 shows the whole process. In the figure, $K1$–$K3$ are $K_{\mathrm{DC}_i}^{\mathrm{PO}}$, $K_{\mathrm{MD}_j}^{\mathrm{PO}}$, and $K_{\mathrm{MD}_j}^{\mathrm{DC}_i}$, respectively.

1) PO first identifies all the DCs to talk to according to a certain optimization criterion. It captures the current

timestamp T, signs it, encrypts T and the signature using $K_{\mathrm{DC}_i}^{\mathrm{PO}}$, and sends the encrypted message to DC_i. Note that it is possible that PO does not want to collect data from some MDs in MDLIST_i. If so, PO should also include the list of intended MDs. We omit that in our protocol to simplify the discussion

$$\mathrm{PO} \to \mathrm{DC}_i : \mathrm{SKE}\Big(K_{\mathrm{DC}_i}^{\mathrm{PO}}, T||\mathrm{SIGN}(\mathrm{PO}, T)\Big).$$

2) Upon receiving the message, DC_i can retrieve $T||\mathrm{SIGN}(\mathrm{PO}, T))$ using the shared key. It first verifies the signature to ensure the message has not been tampered. To detect whether the message is a replayed one, it checks whether T is acceptable. It then encrypts $T||\mathrm{SIGN}(\mathrm{PO}, T))$ using the group key GK_i and sends the message to $\mathrm{MD}_j \in \mathrm{MDLIST}_i$ (or only the MDs PO wants to collect data from)

$$\mathrm{DC}_i \to \mathrm{MD}_j : \mathrm{SKE}(\mathrm{GK}_i, T||\mathrm{SIGN}(\mathrm{PO}, T)).$$

By encrypting the message using the group key, DC_i only needs to create a single message for all MDs in its group. However, the group key cannot authenticate it was PO who requested the data collection because it is a key shared by many entities. We thus need to include a signature of PO to facilitate authentication. This message should work fine if DC_i has to collect data from every MD in MDLIST_i. However, when some MDs are not supposed to report data, those are not reporting can also read T in the message. As T is only a timestamp and is not a secret, knowing T would not allow MDs to launch any attack. However, if this is a serious concern, DC_i can send $\mathrm{SKE}(K_{\mathrm{MD}_j}^{\mathrm{DC}_i}, T||\mathrm{SIGN}(\mathrm{PO}, T))$ to the involved MDs instead. The disadvantage of this approach is DC_i needs to create a different message for different MD and possibly incurs more delay in the data collection process. As an attacker does not know GK_i and cannot forge $\mathrm{SIGN}(\mathrm{PO}, T)$, the message is safe from impersonation.

3) When MD_j receives the message, it retrieves the content using GK_i. It first verifies the signature and whether T is within an acceptable range. If so, MD_j generates keys for protecting the data and allows DC_i to perform integrity check. Let the message key (MK) be $\mathrm{MK} = \mathrm{GENKEY}(K_{\mathrm{MD}_j}^{\mathrm{PO}}, T)$. An encryption key and an integrity key developed based on MK are used to protect the data. The protected data is denoted as PRODATA. As MK depends on T, different MKs will be used for different data collection instances even $K_{\mathrm{MD}_j}^{\mathrm{PO}}$ is not changed. MD_j also generates $DK = \mathrm{GENKEY}(K_{\mathrm{MD}_j}^{\mathrm{DC}_i}, T)$ to protect from message tampering. The hash of PRODATA using DK is computed and sent to DC_i

$$\mathrm{MD}_j \to \mathrm{DC}_i : \mathrm{PRODATA}, \mathrm{HASH}(DK, \mathrm{PRODATA}).$$

Note that as PRODATA is encrypted using a derivative of $K_{\mathrm{MD}_j}^{\mathrm{PO}}$, DC_i cannot decrypt and read it. PRODATA is thus secure against honest-but-curious DCs. The hash of PRODATA, on the other hand, is computed using a derivative of $K_{\mathrm{MD}_j}^{\mathrm{DC}_i}$. DC_i can thus check whether an attacker has tampered the message before relaying the data back to the PO. As an attacker does not know $K_{\mathrm{MD}_j}^{\mathrm{DC}_i}$, he cannot impersonate MD_j to send DC_i the data.

4) When the encrypted data arrives, DC_i verifies the hash to ensure PRODATA was generated by MD_j even it cannot decrypt PRODATA. It then forwards PRODATA to PO by encrypting it $\mathrm{GENKEY}(K_{\mathrm{DC}_i}^{\mathrm{PO}}, T)$. 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

$$\mathrm{DC}_i \to \mathrm{PO} : \mathrm{SKE}\Big(\mathrm{GENKEY}(K_{\mathrm{DC}_i}^{\mathrm{PO}}, T), \mathrm{PRODATA}\Big).$$

5) Upon receiving the message, PO retrieves PRODATA by decrypting the message using $\mathrm{GENKEY}(K_{\mathrm{DC}_i}^{\mathrm{PO}}, T)$. It also develops MK to extract the data from PRODATA. Because $K_{\mathrm{DC}_i}^{\mathrm{PO}}$ is a shared secret between PO and DC_i, an attacker cannot forge the message. If the message is tampered, say, a bit is flipped, PRODATA decrypted would be scrambled and would not pass the integrity check using MK. The data sent from MD_j are thus remain confidential and secure.

It can be observed that each MD and DC and the PO need to perform one public key operation only no matter how many messages it has to handle. Besides, the signature verification that MDs and DCs have to perform is not very expensive when compared with signature creation. Our protocol is thus very light-weight and scalable.

B. PO Requests Data From MD_j

We list the steps PO can take to request date from MD_j.

1) PO first identifies a certain DC_i such that $\mathrm{MD}_j \in G_i$. T is the timestamp. Apart from signing the timestamp, PO also encrypts the timestamp using $K_{\mathrm{MD}_j}^{\mathrm{PO}}$

$$\mathrm{PO} \to \mathrm{DC}_i : \mathrm{SKE}\Big(K_{\mathrm{DC}_i}^{\mathrm{PO}}, T||\mathrm{SIGN}(\mathrm{PO}, T)|| \mathrm{SKE}\Big(K_{\mathrm{MD}_j}^{\mathrm{PO}}, T\Big)\Big).$$

2) DC_i sends the information to MD_j after verifying the signature on T

$$\mathrm{DC}_i \to \mathrm{MD}_j : \mathrm{SKE}\Big(K_{\mathrm{MD}_j}^{\mathrm{DC}_i}, T||\mathrm{SKE}\Big(K_{\mathrm{MD}_j}^{\mathrm{PO}}, T\Big)\Big).$$

Steps 3–5 are the same as in Section V-A.

Similar mechanism can be used for PO to issue an urgent command to MD_j. MD_j should respond with an acknowledgement instead of PRODATA.

C. MD_j *Initiates Urgent Data Report*

The following are the steps by MD_j to report unsolicited urgent data.

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

$$\mathrm{MD}_j \to \mathrm{DC}_i : \mathrm{SKE}\Big(K_{\mathrm{MD}_j}^{\mathrm{DC}_i}, T||\mathrm{PRODATA}|| \mathrm{HASH}(DK, \mathrm{PRODATA})\Big).$$

2) DC_i verifies the hash and forwards PRODATA to PO

$$\mathrm{DC}_i \rightarrow \mathrm{PO} : \mathrm{SKE}\Big(K^{\mathrm{PO}}_{\mathrm{DC}_i}, T||\mathrm{PRODATA}\Big).$$

3) PO can then extract T using $K^{\mathrm{PO}}_{\mathrm{DC}_i}$ 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 $\mathrm{HASH}(DK, \mathrm{PRODATA})$ and encrypt $T||\mathrm{PRODATA}||\mathrm{HASH}(DK, \mathrm{PRODATA})$ using different keys for different DCs in step 1. As both operations are not expensive, MD_j can send out the reports promptly.

D. PO Issues Urgent Command to Group of MDs

When PO invokes a group of MDs, it employs the following.

1) Similar to requesting data, PO should first identify the DCs that cover all the MDs that it wants to send the urgent command to. Let the command be COMD. MDLIST_i contains the MDs that DC_i should talk to

$$\mathrm{PO} \rightarrow \mathrm{DC}_i : \mathrm{SKE}\Big(K^{\mathrm{PO}}_{\mathrm{DC}_i}, \mathrm{SIGN}(\mathrm{PO}, \mathrm{COMD})|| \\ \mathrm{MDLIST}_i||\mathrm{COMD}\Big).$$

2) DC_i sends to each MD_j in MDLIST_i the urgent command

$$\mathrm{DC}_i \rightarrow \mathrm{MD}_j : \mathrm{SKE}(\mathrm{GK}_i, \mathrm{SIGN}(\mathrm{PO}, \mathrm{COMD})||\mathrm{COMD}).$$

The signature of the command by the PO provides authentication check to all MDs and DCs. By using a group key in step 2, we share the same issue as in step 2 of Section V-A. The administrator can thus select the most appropriate way to strike a balance of security and efficiency.

VI. Grouping Optimization

A. Deriving the Optimized Data Collection Time

We now consider how to minimize the time to perform data collection from a group of MDs by selecting a single appropriate DC to collect data from each MD. To compute the total time needed for PO to collect the data, we first define some notations to represent the time needed to perform a single cryptographic operation defined in Table II. Theoretically speaking, the time needed for a cryptographic operation depends on the size of the message. As we only perform public key operations on small-sized messages, we ignore this factor and denote $T^p(OP, A)$ as the time needed for A to execute public key cryptographic operation PKE, PKD, SIGN, and SIGV. For example, the time for PO to sign a message is $T^p(\mathrm{SIGN}, \mathrm{PO})$. To capture the effect of message size on the computational time of symmetric key and hash operations, we denote the time needed as $T^s(OP, A, \mathrm{size})$. As symmetric key encryption and decryption take roughly the same time, we use SK to represent both SKE and SKD. We also use HASH to denote both hash computation and verification. To simplify our discussion, we assume the size of $T||\mathrm{SIGN}(\mathrm{PO}, T)$ in Section V-A as 1 unit. That is, the time needed for DC_i to develop message $\mathrm{SKE}(\mathrm{GK}_i, T||\mathrm{SIGN}(\mathrm{PO}, T))$ is $T^s(SK, \mathrm{DC}_i, 1)$. The one-way network delay between A and B is $T^n(A, B)$. We also let $x_{ij} = 1$ if MD_j belongs to G_i.

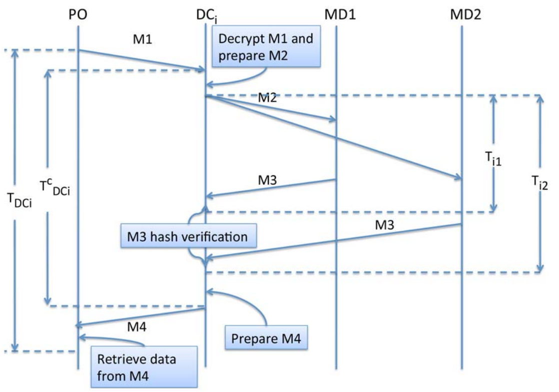


Fig. 4. Time for data collection.

To simplify our discussion, we use $M1$–$M4$ to represent the four messages exchanged between PO, DCs, and MDs as shown in Fig. 3. We only consider the situation where a DC reports all data collected in a single message to PO. To illustrate the process of time analysis, we present Fig. 4 to explain the different time components in the whole data collection process. In the picture, we assume there are only two MDs.

We first develop the time needed for DC_i, after having prepared $M2$, to send message $M2 = \mathrm{SKE}(\mathrm{GK}_i, T||\mathrm{SIGN}(\mathrm{PO}, T))$ to MD_j and verify the hash of MD_j's reply, which is denoted as T_{ij}. T_{ij} is the sum of the following components:

1) round-trip network delay between DC_i and MD_j: $2T^n(\mathrm{DC}_i, \mathrm{MD}_j)$;
2) time needed for MD_j to generate reply $M3$ (step 3): $T^s(SK, \mathrm{MD}_j, 1) + T^p(\mathrm{SIGV}, \mathrm{MD}_j) + T^s(SK, \mathrm{MD}_j, \mathrm{size}) + 2T^s(\mathrm{HASH}, \mathrm{MD}_j, \mathrm{size})$ where size is the size of the data in terms of number of units;
3) time needed for DC_i to verify the hash: $T^s(\mathrm{HASH}, \mathrm{DC}_i, \mathrm{size})$.

Before DC_i can send message $M2 = \mathrm{SKE}(\mathrm{GK}_i, T||\mathrm{SIGN}(\mathrm{PO}, T))$ to MD_j, DC_i needs to decrypt $M1$ and prepare $M2$. As described in step 2 in Section V-A, DC_i has to spend $2T^s(SK, \mathrm{DC}_i, 1) + T^p(\mathrm{SIGV}, \mathrm{DC}_i)$ time to prepare $M2$. We now study the time needed for DC_i to prepare the reply ($M4$) to PO after verifying the hashes of the replies from all MDs. Let N_i be $\sum_j x_{ij}$. That is, N_i is the number of MDs in G_i. The total amount of data received by DC_i is $N_i \times \mathrm{size}$. The time to prepare $M4$ is $T^s(SK, \mathrm{DC}_i, N_i \times \mathrm{size})$. Therefore, the total time needed for DC_i from the moment it receives $M1$ from PO to the moment it sends out $M4$ to the PO is

$$T^c_{\mathrm{DC}_i} = 2T^s(SK, \mathrm{DC}_i, 1) + T^p(\mathrm{SIGV}, \mathrm{DC}_i) + \max_j\{x_{ij}T_{ij}\} \\ + T^s(SK, \mathrm{DC}_i, N_i \times \mathrm{size}).$$

We now study the time from the moment that PO sends out $M1$ until the moment that PO successfully decrypts and verifies the data carried in $M4$ sent by DC_i. We denote this time as T_{DC_i}. To retrieve the raw data from $M4 = \mathrm{SKE}(\mathrm{GENKEY}(K^{\mathrm{PO}}_{\mathrm{DC}_i}, T), \mathrm{PRODATA})$, PO first needs to decrypt $M4$ using GENKEY$(K_{\mathrm{DC}_i}^{\mathrm{PO}}, T)$. It then needs to decrypt and verify the hash carried in PRODATA. Therefore, T_{DC_i} is

$$\begin{aligned} T_{\mathrm{DC}_i} &= 2T^n(\mathrm{PO}, \mathrm{DC}_i) + T_{\mathrm{DC}_i}^c + 2T^s(SK, \mathrm{PO}, N_i \times \mathrm{size}) \\ &\quad + T^s(\mathrm{HASH}, \mathrm{PO}, N_i \times \mathrm{size}) \\ &= f(i) + \max_j \{x_{ij} T_{ij}\} \end{aligned} \tag{1}$$

where

$$\begin{aligned} f(i) &= 2T^s(SK, \mathrm{DC}_i, 1) + T^p(\mathrm{SIGV}, \mathrm{DC}_i) + 2T^n(\mathrm{PO}, \mathrm{DC}_i) \\ &\quad + 2T^s(SK, \mathrm{PO}, N_i \times \mathrm{size}) + T^s(\mathrm{HASH}, \mathrm{PO}, N_i \times \mathrm{size}). \end{aligned}$$

B. Problem Formulation

When PO wants to collect all data as soon as possible, we should assign each MD to an appropriate DC such that the maximum T_{DC_i} over all $i \in \mathbb{D}$ is minimized. Such an objective leads to what is known in the literature as a minimax problem. From (1), we can simplify the terms into two major categories for the minimax optimization. One is the maximizing component $(\max_j\{x_{ij}T_{ij}\})$ and the other is the summative part $(f(i))$. The former consists mostly of the network delay whose maximum value will determine the total completion time for data collection by a DC. The latter includes the processing time, including the cryptographic computation, whose total time will be a summation operation. In what follows, we will ignore the maximization components, as it is rather trivial to address alone, and concentrate on the summative part. Under a realistic data collection scenario, summative component will likely be the dominant term to determine the overall performance.

When the summative part is considered, the problem looks very similar to the makespan minimization problem from the scheduling theory [46], [47]. Scheduling theory considers problems where a set of jobs (tasks) are to be assigned to a set of machines or processors to satisfy an objective. One machine can only work on one job at a time. The well-established three-field classification introduced in [48] uses $\alpha|\beta|\gamma$ notation, where job, machine, and scheduling characteristics are denoted by α, β, and γ, respectively. The summative part of our objective function is denoted by $Q||C_{\max}$, where arbitrary number of machines operating at different speeds must be used to complete a given set of tasks with the minimum makespan objective. This problem setting is also known in the literature as uniform parallel machines [49]. In our problem, machines are DCs, and tasks are MDs whose data need to be collected.

The integer linear programming (ILP) formulation for our summative part may be formulated as follows:

$$\min \quad \max \quad \sum_j x_{ij} t_{ij} \tag{2}$$

$$\text{s.t.} \quad \sum_j x_{ij} = 1, \quad \forall i \in \mathbb{D} \tag{3}$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in \mathbb{D}, \quad \forall j \in \mathbb{M} \tag{4}$$

where x_{ij} represents whether DC i is assigned to collect data from MD j, and t_{ij} is the amount of the summative part of the total data collection time of MD j's data by DC i.

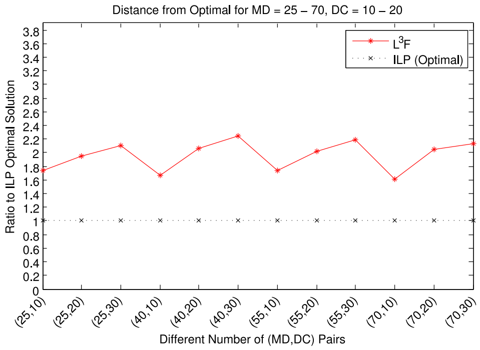


Fig. 5. Ratio of total data collection time for L^{3}F to optimal ILP.

When we let $C_{\max}$ represent the maximum data collection time, the above formulation can be rewritten in a standard form as follows:

$$\min \quad C_{\max} \tag{5}$$

$$\text{s.t.} \quad \sum_j x_{ij} = 1, \quad \forall i \in \mathbb{D} \tag{6}$$

$$\sum_j x_{ij} t_{ij} \le C_{\max}, \quad \forall i \in \mathbb{D} \tag{7}$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in \mathbb{D}, \quad \forall j \in \mathbb{M}. \tag{8}$$

The above problem can be shown to be strongly NP-hard [50], [51] by a reduction from a three-partition problem [52]. Also note that this problem is a kind of the dual of the bin packing problem [50], [53].

As solving the ILP of minimum makespan is NP-hard by reduction from a three-partition problem, and thus making it unlikely that a polynomial algorithm exists, we develop a greedy heuristic, least loaded DC first (L^{3}F), to solve the problem. We find the largest time for data collection for any (DC, MD) pair, say δ, μ. We assign MD μ to a DC that will complete in the least time. Next, we pick the next largest time and assign it to the least loaded DC for the corresponding MD. We iterate until we deplete unassigned MDs. It is obvious that the complexity of the algorithm is $O(d)$. Due to the page limitation, we omit the details of the full algorithm and refer the reader to [45].

VII. Performance Evaluation

We have used CPLEX to solve the ILP formulations and implemented our approaches in C++. Since the problem is NP-Hard, the ILP formulation that can be solved by CPLEX hits a wall rather quickly. After about 70 MDs and 35 DCs, CPLEX started taking very long to yield any results. Thus, we have run some simulations up to 70 MDs and 35 DCs each with 30 runs to get an idea of the comparative performance results. The time for collecting data from MDs by DCs are randomly generated from a uniform probability distribution in the range of 10–100. The number of DCs took the values of 10, 20, and 30 while the number of MDs were assigned 25, 40, 55, and 70. All possible combinations were run for 30 times for statistical significance.

Fig. 5 shows the performance of ILP and L^{3}F for all 12 combinations of the number of MDs and DCs. It plots the total

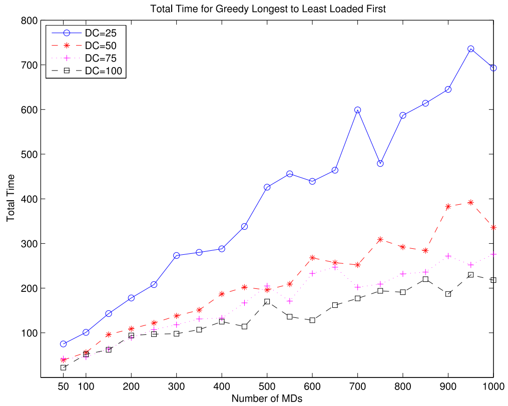


Fig. 6. Performance of L^{3}F in terms of total data collection time over changing the number of MDs with four different number of DCs.

time values returned by the ILP from CPLEX as the reference point and hence shows it as a straight line on bottom. L^{3}F, being a greedy algorithm, performed worse with an average distance ratio to the optimum of approximately 1.96.

For more MDs and DCs, ILP cannot yield results. Thus, we only report L^{3}F in extensive simulations with the number of MDs going up to 1000 in increments of 50 starting from 50 and number of DCs at 25, 50, 75, and 100. We had a total of 80 unique (MD, DC) pairs. Again, in order to attain statistical significance, each combination pair was run 30 times. The time values for the data collection from MDs by DCs were generated using a uniform density function in the range of 10–100. Fig. 6 displays the total time of data collection for L^{3}F over the number of MDs from 50 to 1000 for 25, 50, 75, and 100 DCs as separate lines. Except for when the number of DCs was equal to 25, the total time increases with respect to larger number of MDs is with moderate slope. When DC is equal to 25, the increase is rather steep but still linear. This behavior might indicate that when there is significant imbalance between the number of DCs and MDs the total time to collect data may adversely affected. This point of operating overload is hard to have a threshold value to associate with but nevertheless should be considered.

VIII. Conclusion

The bidirectional power and information flow of the smart grid vision has led to the proliferation of a variety of MDs. These devices generate unprecedented amounts of data. The existing, legacy protocols are not capable of addressing this new phenomenon. In order to address this challenge, we propose a comprehensive and secure communications protocol to enable a PO to collect data from MDs in a practical, scalable, and efficient manner under a hierarchical data collection model. Intermediary nodes are assumed to follow the honest-but-curious model in relaying the data. Thus, our protocol paves the way for third party service provisioning, as envisioned by the NIST smart grid framework. Examples of such services include outsourcing data collection by third party DCs, utilizing cloud computing services for data storage and processing, etc. We formulate an optimization problem for associating the intermediary relay nodes with MDs for data collection in order to minimize the total data collection time. The problem is intractable and thus we present a heuristic algorithm with good approximation and fast convergence.

Acknowledgment

This report was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.

References

[1] N. Kayastha, D. Niyato, E. Hossain, and Z. Han, "Smart grid sensor data collection, communication, and networking: A tutorial," *Wireless Commun. Mobile Comput.*, vol. 14, no. 11, pp. 1055–1087, 2012.

[2] *NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 3.0, Smart Grid Interoperability Panel (SGIP)*, NIST Standard 1108R3, Oct. 2013.

[3] X. Fang, S. Misra, G. Xue, and D. Yang, "Managing smart grid information in the cloud: Opportunities, model, and applications," *IEEE Netw.*, vol. 26, no. 4, pp. 32–38, Jul./Aug. 2012.

[4] S. Bera, S. Misra, and J. Rodrigues, "Cloud computing applications for smart grid: A survey," *IEEE Trans. Parallel Distrib. Syst.*, to be published.

[5] R. Tabassum, K. Nahrstedt, E. Rogers, and K.-S. Lui, "SCAPACH: Scalable password-changing protocol for smart grid device authentication," in *Proc. 22nd Int. Conf. Comput. Commun. Netw.*, Nassau, The Bahamas, 2013, pp. 1–5.

[6] *The Transport Layer Security (TLS) Protocol Version 1.2*, RFC Standard 5246, 2008.

[7] Y.-J. Kim, V. Kolesnikov, and M. Thottan, "Resilient end-to-end message protection for large-scale cyber-physical system communications," in *Proc. IEEE 3rd Int. Conf. Smart Grid Commun. (SmartGridComm)*, Tainan, Taiwan, 2012, pp. 193–198.

[8] *DNP3 Secure Authentication Version 5*, IEEE Standard 1815-2012, 2011.

[9] *IEC Power Utility Automation, Technical Committee 57 (TC57)*, IEC Standard 61850, 2003.

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

[11] Y.-J. Kim, V. Kolesnikov, H. Kim, and M. Thottan, "SSTP: A scalable and secure transport protocol for smart grid data collection," in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Brussels, Belgium, 2011, pp. 161–166.

[12] T. Khalifa, K. Naik, M. Alsabaan, A. Nayak, and N. Goel, "Transport protocol for smart grid infrastructure," in *Proc. IEEE Int. Conf. Ubiquitous Future Netw.*, Jeju Island, Korea, 2010, pp. 320–325.

[13] X. Long, D. Tipper, and Y. Qian, "An advanced key management scheme for secure smart grid communications," in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Vancouver, BC, Canada, 2013, pp. 504–509.

[14] 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 Trans. Ind. Electron.*, vol. 60, no. 10, pp. 4746–4756, Oct. 2013.

[15] D. Wu and C. Zhou, "Fault-tolerant and scalable key management for smart grid," *IEEE Trans. Smart Grid*, vol. 2, no. 2, pp. 375–381, Jun. 2011.

[16] M. M. Fouda, Z. M. Fadlullah, N. Kato, R. Lu, and X. Shen, "A lightweight message authentication scheme for smart grid communications," *IEEE Trans. Smart Grid*, vol. 2, no. 4, pp. 675–685, Dec. 2011.

[17] 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. 2nd Int. Conf. Smart Grids Green Commun. IT Energy-Aware Technol. (ENERGY)*, St. Maarten, The Netherlands, 2012, pp. 60–68.

[18] Y. Law, G. Kounga, and A. Lo, "WAKE: Key management scheme for wide-area measurement systems in smart grid," *IEEE Commun. Mag.*, vol. 51, no. 1, pp. 34–41, Jan. 2013.

[19] 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 Trans. Ind. Electron.*, vol. 60, no. 10, pp. 4746–4756, Oct. 2013.

[20] A. Seshadri, M. Luk, and A. Perrig, "SAKE: Software attestation for key establishment in sensor networks," in *Proc. Int. Conf. Distrib. Comput. Sensor Syst.*, Santorini Island, Greece, 2008, pp. 372–385.

[21] H. Nicanfar and V. Leung, "Multilayer consensus ECC-based password authenticated key-exchange (MCEPAK) protocol for smart grid system," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 253–264, Mar. 2013.

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

[23] S. Uludag, S. Zeadally, and M. Badra, "Techniques, taxonomy, and challenges of privacy protection in smart grid," in *Privacy in a Digital, Networked World*. London, U.K.: Springer, Feb. 2015.

[24] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "PDA: Privacy-preserving data aggregation in wireless sensor networks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Anchorage, AK, USA, 2007, pp. 2045–2053.

[25] W. He, H. Nguyen, X. Liu, K. Nahrstedt, and T. Abdelzaher, "iPDA: An integrity-protecting private data aggregation scheme for wireless sensor networks," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, San Diego, CA, USA, 2008, pp. 1–7.

[26] T. Feng, C. Wang, W. Zhang, and L. Ruan, "Confidentiality protection for distributed sensor data aggregation," in *Proc. IEEE 27th Conf. Comput. Commun. (INFOCOM)*, Phoenix, AZ, USA, 2008, pp. 1–9.

[27] M. Groat, W. He, and S. Forrest, "KIPDA: k-indistinguishable privacy-preserving data aggregation in wireless sensor networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Shanghai, China, 2011, pp. 2024–2032.

[28] Y. Yan, Y. Qian, and H. Sharif, "A secure and reliable in-network collaborative communication scheme for advanced metering infrastructure in smart grid," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Cancun, Mexico, 2011, pp. 909–914.

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

[30] C. Rottondi, G. Verticale, and C. Krauss, "Distributed privacy-preserving aggregation of metering data in smart grids," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 7, pp. 1342–1354, Jul. 2013.

[31] H. Nicanfar, A. Alasaad, P. Talebifard, and V. Leung, "Network coding based encryption system for advanced metering infrastructure," in *Proc. IEEE Int. Conf. Comput. Commun. Netw. (ICCCN)*, Nassau, Bahamas, 2013, pp. 1–7.

[32] D. Niyato and P. Wang, "Cooperative transmission for meter data collection in smart grid," *IEEE Commun. Mag.*, vol. 50, no. 4, pp. 90–97, Apr. 2012.

[33] S. Shao *et al.*, "Traffic scheduling for wireless meter data collection in smart grid communication network," in *Proc. Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT)*, Hefei, China, 2014, pp. 1–7.

[34] F. Li, B. Luo, and P. Liu, "Secure information aggregation for smart grids using homomorphic encryption," in *Proc. 1st IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Gaithersburg, MD, USA, 2010, pp. 327–332.

[35] N. Yukun *et al.*, "A security privacy protection scheme for data collection of smart meters based on homomorphic encryption," in *Proc. IEEE EUROCON*, Zagreb, Croatia, 2013, pp. 1401–1405.

[36] J. Kamto, L. Qian, J. Fuller, J. Attia, and Y. Qian, "Key distribution and management for power aggregation and accountability in advance metering infrastructure," in *Proc. IEEE 3rd Int. Conf. Smart Grid Commun. (SmartGridComm)*, Tainan, Taiwan, 2012, pp. 360–365.

[37] F. Li and B. Luo, "Preserving data integrity for smart grid data aggregation," in *Proc. IEEE 3rd Int. Conf. Smart Grid Commun. (SmartGridComm)*, Tainan, Taiwan, 2012, pp. 366–371.

[38] T. Chim, S. Yiu, V. Li, C. Hui, and J. Zhong, "PRGA: Privacy-preserving recording & gateway-assisted authentication of power usage information for smart grid," *IEEE Trans. Depend. Secure Comput.*, vol. 12, no. 1, pp. 85–97, Jan./Feb. 2015.

[39] L. Chen, R. Lu, and Z. Cao, "PDAFT: A privacy-preserving data aggregation scheme with fault tolerance for smart grid communications," in *Peer-to-Peer Networking and Applications*. New York, NY, USA: Springer, 2014, pp. 1–11.

[40] J. Hur, "Attribute-based secure data sharing with hidden policies in smart grid," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 11, pp. 2171–2180, Nov. 2013.

[41] P. Li, S. Guo, and Z. Cheng, "Joint optimization of electricity and communication cost for meter data collection in smart grid," *IEEE Trans. Emerg. Topics Comput.*, vol. 1, no. 2, pp. 297–306, Dec. 2013.

[42] P. Li and S. Guo, "Delay minimization for reliable data collection on overhead transmission lines in smart grid," in *Proc. Comput. Commun. IT Appl. Conf. (ComComAp)*, Hong Kong, 2013, pp. 147–152.

[43] M. Qiu, H. Su, M. Chen, Z. Ming, and L. Yang, "Balance of security strength and energy for a PMU monitoring system in smart grid," *IEEE Commun. Mag.*, vol. 50, no. 5, pp. 142–149, May 2012.

[44] 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 *Proc. IEEE Smart Grid Commun. (SmartGridComm)*, Vancouver, BC, USA, 2013, pp. 480–485.

[45] S. Uludag, K.-S. Lui, W. Ren, and K. Nahrstedt, "Secure and scalable communications protocol for data collection with time minimization in the smart grid," Dept. Comput. Sci., Univ. Illinois Urbana-Champaign, Urbana, IL, USA, Tech. Rep. 2014-07-16, Jul. 2014. [Online]. Available: http://hdl.handle.net/2142/49985

[46] J. Y.-T. Leung, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. London, U.K.: Chapman and Hall, 2004.

[47] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. New York, NY, USA: Springer, 2008.

[48] R. Graham, E. Lawler, J. Lenstra, and A. Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Ann. Discrete Math.*, vol. 5, no. 2, pp. 287–326, 1979. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016750600870356X

[49] P. Brucker, *Scheduling Algorithms*. Berlin, Germany: Springer-Verlag, 2007. [Online]. Available: http://books.google.com/books?id=FrUytMqlCv8C

[50] J. Y.-T. Leung, "Bin packing with restricted piece sizes," *Inf. Process. Lett.*, vol. 31, no. 3, pp. 145–149, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0020019089902238

[51] D. S. Hochbaum and D. B. Shmoys, "Using dual approximation algorithms for scheduling problems theoretical and practical results," *J. ACM*, vol. 34, no. 1, pp. 144–162, Jan. 1987. [Online]. Available: http://doi.acm.org/10.1145/7531.7535

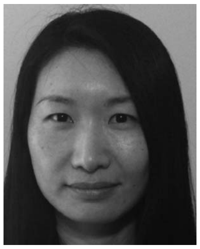
[52] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: W. H. Freeman, 1979.

[53] E. Coffman, Jr., M. Garey, and D. Johnson, "An application of bin-packing to multiprocessor scheduling," *SIAM J. Comput.*, vol. 7, no. 1, pp. 1–17, 1978. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/0207001

Suleyman Uludag (M'09) received the B.A. degree in business administration from Marmara University, Istanbul, Turkey, in 1991; the M.B.A. degree from the Illinois Institute of Technology, Chicago, IL, USA, in 1992; and the M.Sc. degree in telecommunications and the Ph.D. degree in computer science from DePaul University, Chicago, in 1997 and 2007, respectively.

He is currently an Associate Professor of Computer Science with the University of Michigan–Flint, Flint, MI, USA. His current research interests include network quality of service, routing in wireless and wired networks, microgrids, network security, intelligent transportation system, and smart grid privacy and security.

Dr. Uludag is a member of the Association for Computing Machinery.



KingShan Lui (S'00–M'03–SM'14) received the B.Eng. and M.Phil. degrees from the Hong Kong University of Science and Technology, Hong Kong, in 1994 and 1995, respectively, and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 2002.

She joined the Department of Electrical and Electronic Engineering, University of Hong Kong, Hong Kong, in 2002, where she is currently an Associate Professor. Her current research interests include network protocol design and analysis, smart grids, wireless networks, and quality-of-service issues.

Wenyu Ren received the B.S. degree in electronic engineering from Tsinghua University, China, in 2013. He is currently pursuing the Ph.D. degree in computer science from the University of Illinois at Urbana Champaign, Champaign, IL, USA.

His current research interests include secure and scalable communication framework in the smart grid.

Klara Nahrstedt (F'08) received the undergraduate degree in mathematics and the Mathematics Diploma in numerical analysis from Humboldt University, Berlin, Germany, in 1984 and 1985, respectively, and the Ph.D. degree in computer science from the University of Pennsylvania , Philadelphia, PA, USA.

She is the Ralph and Catherine Fisher Professor with the Department of Computer Science, and the Director of the Coordinated Science Laboratory, College of Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA. Her current research interests include trustworthy power grid, 3-D teleimmersive systems, mobile systems, quality of service and resource management, quality of experience in multimedia systems, and real-time security in mission-critical systems. She is the co-author of the widely used multimedia books *Multimedia: Computing, Communications, and Applications* (Prentice Hall, 1995), and *Multimedia Systems* (Springer-Verlag, 2004).

Prof. Nahrstedt was a recipient of the IEEE Communication Society Leonard Abraham Award for Research Achievements, University Scholar, the Humboldt Award, and the IEEE Computer Society Technical Achievement Award. She was the Chair of the Association for Computing Machinery (ACM) Special Interest Group in Multimedia. She was the General Chair of ACM Multimedia 2006, the ACM Network and Operating System Support for Digital Audio and Video (NOSSDAV) Conference 2007, and IEEE Percom 2009. She is a Fellow of ACM and a Member of the Leopoldina German National Academy of Sciences.