Privacy-Preserving Collection of Power Consumption Data for Enhanced AMI Networks

Ahmad Alsharif*, Mahmoud Nabil*, Mohamed Mahmoud*, and Mohamed Abdallah**
*Department of Electrical and Computer Engineering, Tennessee Tech. University, TN, USA 38505.
** College of Science and Engineering, Hamad bin Khalifa University, Doha, Qatar.

***Abstract*—In this paper, we propose a privacy-preserving data collection scheme for enhanced AMI networks. The idea is that each cluster of meters is divided into members and heads. A cluster member should send encrypted subreadings with lifetime values to a number of cluster heads where the lifetime permits the cluster heads to reuse the received subreading for future reporting cycles. Then cluster heads should aggregate all the received/stored subreadings and send the result to a local aggregator which performs a further aggregation process and then send an aggregated reading for the cluster to the utility. If the reading of a cluster member does not change, it should run a countermeasure to traffic analysis to determine whether it needs to send a subreading to one of the cluster heads or not, whereas if the power reading changes or the lifetime of a subreading expires, the cluster member needs to update only one subreading to make the summation of all its subreadings gives the correct reading. In addition, the proposed scheme is more resistive to collusion attacks than existing schemes. Our analysis demonstrates that the proposed scheme can preserve consumers privacy and resist collusion attacks. Our measurements confirm that the proposed scheme can reduce the communication bandwidth by 30%.**

I. Introduction

The advanced metering infrastructure (AMI) networks are one part of the smart grid that provides two-way communication between the smart meters (SMs) installed at the consumer side and the utility company. AMI networks allow the utility to collect energy consumption data at high rates, e.g., every few minutes, for energy management and grid monitoring. However, the collection of such fine-grained data causes privacy concerns for consumers since this data can leak sensitive information about the consumers' activities [1].

Another problem that arises when sending power consumption data every few minutes is the consumption of too much bandwidth. Since cellular networks may be used to connect AMI networks to the utility, sending this large amount of data is cost prohibitive [2]. In order to reduce the bandwidth needed for sending power consumption readings to the utility, there is no need to send the same reading value when the power consumption does not change [3]. Thus, we define *enhanced* AMI networks in which each smart meter (SM) should send a power consumption reading in one power consumption collection cycle only if the reading value is different from the one sent in the previous cycle.

Recently, several schemes have been proposed to preserve consumers' privacy while enabling the utility to collect the fine-grained measurements by using data encryption to hide the measurements [4], [5], but the existing schemes consider only fixed-rate-transmission AMI networks. The use of encryption is not enough to preserve consumer privacy in enhanced AMI networks because the event of sending a reading by a meter implies a change in the power consumption. By observing the transmission pattern and using traffic analysis techniques, attackers can infer sensitive information about the consumers' activities without decrypting the encrypted readings [3]. For instance, when the power consumer is at home, he may use multiple appliances which can increase the rate of the power consumption change, and thus trigger sending more readings. On contrary, if the rate of sending readings is significantly lower than the normal level, this is an indication that the consumer may not be at home or sleeping.

The seriousness of traffic analysis attacks in enhanced AMI networks stem from the following facts: 1) the attacks can be launched in an undetectable way because the attackers usually work completely in the receiving mode without disrupting the communications; 2) the use of encryption schemes cannot thwart attacks; 3) the AMI networks use wireless communication, and thus the broadcast nature of this communication can facilitate intercepting meters' transmissions.

In this paper, we propose a privacy-preserving data collection scheme for enhanced AMI networks. In the proposed scheme, a group of meters, called a cluster, is divided into members and heads. A cluster member sends an encrypted subreading with a lifetime value to a number of cluster heads. In each data collection cycle, a meter's reading should equal to the summation of its subreadings used by the cluster heads. Cluster heads should aggregate all the encrypted subreadings of the cluster members and send an encrypted aggregated reading of the cluster meters to the utility. If the power reading of a cluster member changes or the lifetime expires, it needs to update only one of the subreadings so that the summation of all its subreadings gives the new power reading. If the power reading does not change, each cluster member should run a countermeasure to traffic analysis to determine whether it needs to send a subreading to one of the cluster heads or not. Cluster heads should reuse the subreadings as long as they are not expired or updated by cluster members. Our analysis demonstrates that the scheme is secure, can preserve privacy, and has high resistance to collusion attacks. Our measurements confirm that the scheme can significantly save the communication bandwidth.

To the best of our knowledge, this paper is the first attempt to propose a privacy-preserving data collection scheme for enhanced AMI networks.

978-1-5386-2321-3/18/$31.00 ©2018 IEEE

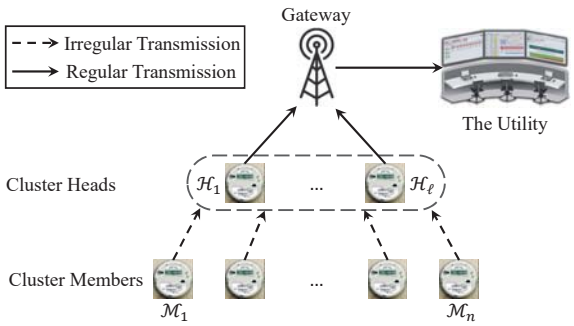

Fig. 1. The considered network model.

The remainder of the paper is organized as follows. The considered system models are presented in section II. The reading splitting technique and privacy-preserving data collection scheme are explained in sections III and IV respectively. Security analysis and performance evaluations are given in sections V and VI, respectively. The related works are discussed in section VII. Finally, conclusions are drawn in section VIII.

II. Network and Threat Models

A. Network Model

The considered network model has a number of consumers in a residential area, the utility, and the gateway which acts as a local collector. SMs are installed at consumers' houses to report real-time fine-grained power consumption readings to the utility through the gateway. SMs are connected through wireless communications, while the gateway can communicate with the utility through wired or long-distance communications. The SMs of a residential area, called cluster, are divided into members and heads. As shown in Figure 1, the AMI network has a cluster of $(\ell + n)$ SMs, with ℓ cluster heads $(\mathcal{H}_1 \dots \mathcal{H}_\ell)$ and n cluster members $(\mathcal{M}_1 \dots \mathcal{M}_n)$. The transmission pattern of cluster members is irregular since a cluster member may not send a packet in some cases, whereas cluster heads should regularly transmit packets to the gateway.

B. Threat Model

Security threats may arise from either internal or external attackers. Internal attackers, which could be the utility, the gateway, and SMs, are considered honest-but-curious, i.e., they follow the protocol and do not disrupt the communication, but they are curious to know the power consumption pattern of consumers. In addition, an external adversary $\mathcal{A}$ can eavesdrop all the exchanged messages between the network nodes to infer sensitive information about consumers. Attackers may try to launch traffic analysis attacks by analyzing the transmission rates of the SMs to extract information such as the appliances being used, whether consumers are at their houses or not [3]. Moreover, to launch stronger attacks, the attackers can collude instead of attacking the system individually.

III. Reading Splitting Technique

In this section, we explain the reading splitting technique which is used in the privacy preserving data collection scheme explained in next section.

Algorithm 1: Generate initial subreadings

Procedure: GENERATE INITIAL SUBREADINGS
Input : $R_i, \ell, \lambda_i, \tau_{i_{min}}, \tau_{i_{max}}$
Output : d, τ

$d[\mathcal{H}_k] \leftarrow 0 \ \forall \ 1 \leq k \leq \ell$
$\tau[\mathcal{H}_k] \leftarrow -1 \ \forall \ 1 \leq k \leq \ell$
$subreadings \leftarrow split(R_i, \lambda_i)$
for $j = 1$ **to** λ_i **do**
  repeat $\mathcal{H}_j \leftarrow randi(1, \ell)$ **until** $\mathcal{H}_j \neq \mathcal{H}_k \ \forall \ 1 \leq k < j$;
  $d[\mathcal{H}_j] \leftarrow subreadings[j]$
  repeat $\tau[\mathcal{H}_j] \leftarrow randi(\tau_{i_{min}}, \tau_{i_{max}})$ **until** $\tau[\mathcal{H}_j] \neq \tau[\mathcal{H}_k] \ \forall \ 1 \leq k < j$;
return (d, τ)

A. Overview

The idea of the proposed reading splitting technique is that each cluster member $\mathcal{M}_i, 1 \leq i \leq n$, should send ℓ subreadings, $\{d_{i1}, \dots, d_{i\ell}\}$, and associated ℓ lifetimes, $\{\tau_{i1}, \dots, \tau_{i\ell}\}$, to ℓ cluster heads. Each cluster head $\mathcal{H}_j, 1 \leq j \leq \ell$, should store the subreading d_{ij} and its lifetime in a list called *stored reports*, and remove the expired subreadings from the list. In each power consumption data collection cycle, the cluster heads should aggregate the subreadings in the list and send one aggregated reading to the gateway. For example, if $\tau_{ij} = 9$, then $\mathcal{M}_i$ asks $\mathcal{H}_j$ to use the associated subreading d_{ij} in the next 9 data collection cycles. If $\mathcal{H}_j$ receives a new subreading from $\mathcal{M}_i$, it should replace the stored subreading with the new one. This update can occur even before the lifetime of the existing subreading expires, as will be explained later in this section.

In the proposed technique, $\mathcal{M}_i$ should run two algorithms. The first should be run only once to send initial subreadings and associated lifetimes to the cluster heads. The second is used to enable the meter to ensure that the summation of the subreadings stored by the cluster heads is equal to its reading, by updating a subreading if needed and protect against traffic analysis as will be explained in details later.

B. Algorithm1: Generating Initial Subreadings

The input parameters of the algorithm are R_i, ℓ, λ_i, $\tau_{i_{min}}$, $\tau_{i_{max}}$ and the outputs of the algorithm are d and τ. For the input parameters, R_i is the current reading of $\mathcal{M}_i$. ℓ is the number of cluster heads. λ_i is the minimum number of cluster heads that should store/use subreadings for $\mathcal{M}_i$ $(\lambda_i \leq \ell)$. The value of λ_i can determine the protection level against collusion attacks as will be explained in section V. $\tau_{i_{min}}$ and $\tau_{i_{max}}$ are the minimum and maximum possible lifetime values of the subreadings. The range of lifetime values controls the reduction in the communication bandwidth and randomness of the transmission pattern as will be explained in section VI. To ensure the randomness of the transmission pattern, $\mathcal{M}_i$ can change the lifetimes limits $\tau_{i_{min}}$ and $\tau_{i_{max}}$ from time to time. For the outputs of the algorithm, d is a vector of size ℓ which contains the subreadings' values of the cluster heads, and τ is a vector of size ℓ which contains the lifetimes correspond-

Algorithm 2: Generate data for updated report

Procedure: GET DATA FOR UPDATED REPORT
Input : $\Delta, \ell, \lambda_i, \tau_{i_{min}}, \tau_{i_{max}}, d, \tau$
Output : d, τ

1 $\tau[\mathcal{H}_k] \leftarrow \tau[\mathcal{H}_k] - 1 \; \forall \; 1 \leq k \leq \ell$
2 $expired \leftarrow get\_expired(\tau)$
3 **if** $\Delta \neq 0$ *or* $expired \neq 0$ **then**
4 **if** $expired \neq 0$ **then**
5 $\Delta \leftarrow \Delta + d[expired]$
6 $d[expired] \leftarrow 0$
7 **if** $\Delta == 0$ *and* $ActiveHeads \geq \lambda_i$ **then**
8 **go to** 13
9 $selected \leftarrow randi(1, \ell, \tau, \lambda_i)$
10 $d[selected] \leftarrow d[selected] + \Delta$
11 **repeat** $\tau[selected] \leftarrow randi(\tau_{i_{min}}, \tau_{i_{max}})$ **until** $\tau[selected] \neq \tau[\mathcal{H}_k] \; \forall \; 1 \leq k \leq \ell \; \& \; selected \neq \mathcal{H}_k$;
12 **else**
13 Randmoly choose between sending a redundant packet to protrect against traffic analysis or send nothing to reduce the number of transmissions
14 **return** (d, τ)

ing to each subreadings. The algorithm uses two functions; $randi(min, max)$ and $split(R, \lambda)$. $randi(min, max)$ returns a random integer number between min and max inclusive, and $split(R, \lambda)$ splits the value of R into λ random values.

The algorithm works as follows. Initially, the output parameters are initialized by assigning 0 to all the subreadings and -1 to all the lifetimes where a negative lifetime value indicates that the corresponding cluster head should not receive a subreading. Then, R_i is split into λ_i subreadings. For each subreading, a cluster head is selected randomly and the subreading is assigned to the selected head. In addition, a lifetime value is randomly chosen and linked to each subreading. Finally, the algorithm outputs d and τ so that $\mathcal{M}_i$ can generate the messages it has to send to the selected cluster heads. Throughout the paper, cluster heads that store unexpired subreadings for $\mathcal{M}_i$ are named the active heads of $\mathcal{M}_i$.

C. Algorithm2: Updating Subreadings

AThe input parameters of the algorithm are Δ, ℓ, λ_i, $\tau_{i_{min}}$, $\tau_{i_{max}}$, d, and τ. Δ is the reading change, i.e., the difference between the current reading and reading of the previous data collection cycle. ℓ, λ_i, $\tau_{i_{min}}$, and $\tau_{i_{max}}$ are previously defined in algorithm 1. d and τ are the current subreadings and corresponding lifetimes. The output parameters are d and τ which are updated versions of the current subreadings and lifetimes after running the algorithm. The algorithm uses the following two functions; $get\_expired(\tau)$ and $randi(min, max, \tau, \lambda_i)$. $get\_expired(\tau)$ accepts a vector τ and returns the index of the head that has expired subreading if exist, and $randi(min, max, \tau, \lambda_i)$ returns a random integer between min and max such that the number of non-negative elements in τ is at least λ_i, i.e., this functions selects a random cluster head such that the total number of selected cluster heads is at least λ_i.

It should be noted that algorithms 1, and 2 should ensure that all the selected lifetimes are different, and hence only one lifetime can expire at a time. This is because if more than one lifetime expires at the same time, $\mathcal{M}_i$ has to send multiple subreadings to sustain the required number of active heads which can consume unnecessary bandwidth.

In each power consumption data collection cycle, based on the statuses of the reading (changed or not) and the subreading lifetimes (expired or not), we have the following cases.

(1) *Reading change without lifetime expiration*.
In this case, $\mathcal{M}_i$ should update a subreading of an active head, or send a new subreading to a non-active head so that the summation of all subreadings gives the current reading. This is achieved through steps 3, 9, 10, and 11.

(2) *No reading change and no lifetime expiration*.
In this case, $\mathcal{M}_i$ either sends a redundant subreading to a head to protect against traffic analysis attacks or does not send any packet to reduce the number of transmissions. The redundant subreading can be sent to an active cluster head or a non-active head. In all classes, the summation of the subreadings should equal to the fine-grained reading of $\mathcal{M}_i$, and the total number of active heads is at least λ. This is achieved through steps 12, and 13.

(3) *A lifetime expires and same or changed reading*.
In this case, $\mathcal{M}_i$ should select a cluster head and activate it by sending a subreading such that the total number of active heads is at least λ. The selected head could be the head of the expired subreading lifetime or any other head. The value of the subreading should be computed such that the summation of all the subreadings is equal to the fine-grained reading of the meter. This process is achieved through steps 2, 3, 4, 5, 6, 9, 10, and 11. In case the reading change is suppressed by the expired subreading, i.e. the new reading is equal to the summation of the existing subreadings after excluding the expired one, and the number of active heads is at least λ_i, then $\mathcal{M}_i$ can take the same actions as (2). This is achieved through steps 3, 4, 5, 6, 7, 8, and 13.

IV. PRIVACY-PRESERVING DATA COLLECTION

A. System Initialization

An offline trusted authority should bootstrap the system. First, it generates the Paillier cryptosystem's public key $(n = pq, g)$, and the corresponding private key (λ, μ) where p and q are two large prime numbers with $|p| = |q|$. Moreover, it generates the bilinear pairing parameters $(q_1, \mathbb{G}, \mathbb{G}_T, P, \hat{e})$. Furthermore, it chooses a secure cryptographic hash function H, where $H : \{0,1\}^* \rightarrow \mathbb{G}$. Finally, it publishes the system parameters as $pubs = \{n, g, q_1, \mathbb{G}, \mathbb{G}_T, P, \hat{e}, H\}$. In addition, each cluster member $\mathcal{M}_i$ chooses a secret key $x_i \in \mathbb{Z}_q^*$ and computes the corresponding public key $Y_i = x_i P$. Similarly, each cluster head $\mathcal{H}_j$ and the gateway possess private/public key pairs x_j/Y_j and x_{gw}/Y_{gw} respectively.

B. Subreading Reports by Cluster Members

1) Initial Report: Each cluster member $\mathcal{M}_i$ should run algorithm 1 to compute the required subreadings and lifetimes. Then, for each selected cluster head $\mathcal{H}_j$, $\mathcal{M}_i$ should send the subreading d_{ij} and the corresponding lifetime τ_{ij} by performing the following steps.

- Step 1: $\mathcal{M}_i$ chooses a random number $r_{ij} \in \mathbb{Z}_n^*$ and encrypts d_{ij} using Paillier cryptosystem as follows.

$$C_{ij} = g^{d_{ij}} \cdot r_{ij}^n \mod n^2$$

- Step 2: $\mathcal{M}_i$ encrypts τ_{ij} using AES encryption as follows.

$$E_{k_{ij}}(\tau_{ij} \parallel TS)$$

where TS is a timestamp and k_{ij} is a symmetric key shared between $\mathcal{M}_i$ and $\mathcal{H}_j$.

- Step 3: $\mathcal{M}_i$ uses its private key x_i to generate a signature σ_{ij} as

$$\sigma_{ij} = x_i H(C_{ij} \parallel \tau_{ij} \parallel TS)$$

- Step 4: $\mathcal{M}_i$ sends $\mathcal{H}_j$ the following tuple.

$$C_{ij} \parallel E_{k_{ij}}(\tau_{ij} \parallel TS) \parallel TS \parallel \sigma_{ij}.$$

2) Updated Reports: For the following report periods, $\mathcal{M}_i$ runs algorithm 2 to determine whether it needs to an update report to a cluster head or not. Based on algorithm 2 output, $\mathcal{M}_i$ either sends an updated report to the selected cluster head using same steps in subsection IV-B1 or $\mathcal{M}_i$ does not need to send any updated reports and hence, all the active cluster heads will use the previously reported subreadings.

C. Aggregation by Cluster Heads

1) Verification of the received reports: After receiving w reports $(w \leq n)$, each cluster head $\mathcal{H}_j$ should check whether the timestamps are fresh or not. Then, it decrypts the encryptions of the lifetimes to obtain τ_{ij}. Finally, it verifies the received signatures efficiently with a batch verification process as

$$\hat{e}\left(\sum_{i=1}^{w} \sigma_{ij}, P\right) \stackrel{?}{=} \prod_{i=1}^{w} \hat{e}\Big(H(C_{ij} \parallel \tau_{ij} \parallel TS), Y_i\Big)$$

If the batch verification process passes, then $\mathcal{H}_j$ moves to the next step, otherwise, it should verify the individual signatures and drop the report that has an invalid signature.

2) Updating the stored reports list: $\mathcal{H}_j$ should update the *stored reports* list by replacing the stored reports with the new ones. Also, if there is no stored report for $\mathcal{M}_i$, $\mathcal{H}_j$ should add to the list a new entry for $\mathcal{M}_i$.

3) Privacy-preserving report aggregation: Once the *stored reports* list is updated, $\mathcal{H}_j$ should generate a report for the aggregated subreadings stored in the list and send it to the gateway using the following steps

- Step 1: $\mathcal{H}_j$ chooses a random number $r_{jg} \in \mathbb{Z}_n^*$ and encrypts d_{jg} using Paillier cryptosystem as follows.

$$C_{jg} = g^{d_{jg}} \cdot r_{jg}^n \mod n^2$$

- Step 2: $\mathcal{H}_j$ computes the aggregated and encrypted subreadings (C_j) as follows.

$$C_j = C_{jg} \prod_{i=1}^{list} C_{ij} \mod n^2$$

- Step 3: $\mathcal{H}_j$ uses its private key x_j to generate the signature σ_j

$$\sigma_j = x_j H(C_j \parallel TS)$$

- Step 4: The report should have the following tuple.

$$C_j \parallel TS \parallel \sigma_j$$

D. Aggregation by the Gateway

1) Received reports verification: After receiving ℓ reports from the ℓ cluster heads, the gateway first checks the freshness of the timestamps, and then it uses a batch verification approach to verify the received signatures as follows.

$$\hat{e}\left(\sum_{j=1}^{\ell} \sigma_j, P\right) \stackrel{?}{=} \prod_{j=1}^{\ell} \hat{e}\Big(H(C_j \parallel TS), Y_j\Big)$$

2) Privacy-preserving report aggregation: Following the verification process, the gateway should compose a report containing the total aggregated readings and send it to the utility using the following steps.

- Step 1: Compute the encrypted aggregated fine-grained reading for the cluster C_{gw} as follows.

$$C_{gw} = \prod_{j=1}^{\ell} C_j \mod n^2$$

- Step 2: Use the private key x_{gw} to compute the signature

$$\sigma_{gw} = x_{gw} H(C_{gw} \parallel TS)$$

- Step 3: The report should have the following tuple.

$$C_{gw} \parallel TS \parallel \sigma_{gw}$$

E. Aggregated Reading Recovery by the Utility

Upon receiving the fine-grained aggregated reading report from the gateway, the utility checks the freshness of the timestamp and verifies the signature by checking

$$\hat{e}(\sigma_{gw}, P) \stackrel{?}{=} \hat{e}\Big(H(C_{gw} \parallel TS), Y_{gw}\Big)$$

Once the signature is verified, the utility uses the secret key (λ, μ) to decrypt C_{gw} and recover the total power consumption of the cluster R_T as

$$R_T = D(C_{gw}) = L\Big({C_{gw}}^{\lambda} \mod n^2\Big) \cdot \mu \mod n$$

V. Security and Privacy Analysis

Privacy Preservation. To reveal the fine-grained reading of $\mathcal{M}_i$, the attacker must obtain all the encrypted subreadings, use the utility's private key to decrypt them, and then add the subreadings. For internal attackers, a cluster head $\mathcal{H}_j$ could have one subreading at a time. However, $\mathcal{H}_j$ cannot decrypt the subreading since the private key needed for decrypting Paillier ciphertext is known only to the utility. Although the utility can decrypt any encrypted subreading, it has access only to the aggregated subreadings but not the individual ones. Therefore, neither the utility nor cluster heads can obtain the reading of $\mathcal{M}_i$, but they need to collude to be able to compute the subreading. On the other hand, an external adversary, $\mathcal{A}$, can eavesdrop all messages transmitted by $\mathcal{M}_i$. However, given the initial subreadings sent by $\mathcal{M}_i$ and the updated subreadings sent later by $\mathcal{M}_i$, $\mathcal{A}$ cannot learn which subreadings are active and which subreadings are expired because each lifetime τ_{ij} is encrypted with a key shared between $\mathcal{M}_i$ and $\mathcal{H}_j$. Therefore, $\mathcal{A}$ cannot get all the encrypted active subreadings at any time. In addition, since $\mathcal{A}$ does not have the utility private key used for decrypting Paillier ciphertext, decryption is impossible.

Moreover, attackers can monitor the transmissions of $\mathcal{M}_i$ to learn whether a transmission is due to reading change to infer sensitive information about the consumer activities. This attack may succeed if the transmission is triggered only by reading changes [3]. However, in our scheme, transmissions are triggered to respond to the following events: 1) a subreading lifetime expires; 2) sending a redundant subreading to protect against traffic analysis attacks; and 3) reading change, as illustrated in subsection III-C. Given that all messages are encrypted, lifetimes have random values, and sending redundant subreadings is random, it is hard for the attackers to differentiate between these events.

Collusion Resistance. The fine-grained power consumption reading of each meter is split between at least λ cluster heads. Therefore, to compute the reading of a meter, all the λ cluster heads must collude with the utility. The utility should use its private key to decrypt the subreadings and then add them to obtain the meter's reading. Obviously, the number of active cluster heads (λ) can determine the protection level against collusion attack, i.e., collusion attack is harder as λ increases because the utility has to collude with a larger number of heads. In the following paragraphs, we formally analyze how a proper value to λ can determine a satisfactory protection level against collusion attacks.

The probability that an $\mathcal{M}_i$ is secured against collusion attack is $1 - q^{\lambda_i}$, where q is the probability that a cluster head colludes with the utility. Using this probability, we define $\mathbb{P}$ as the probability that the utility can obtain at least one meter's readings from n cluster members is

$$\mathbb{P}(q, \lambda, n) = 1 - \prod_{i=1}^{n} (1 - q^{\lambda_i})$$

To clarify how our scheme can resist collusion attacks by the selection of proper value of λ, Figure 2 gives $\mathbb{P}$ versus λ for an AMI network with $n = 100$ cluster members and $q = 0.1$, 0.3, and 0.5. As the value of q increases, as it is likely that the cluster heads collude with the utility. As shown in the figure, the probability of successful collusion attack is less likely as the number of cluster heads increases. For instance, if the probability that a cluster head colludes with the utility (q) is 0.1, then at least four cluster heads should be active ($\lambda = 4$) to ensure that the probability of successful collusion attacks is less than 0.01. It can also be seen that the increase of q triggers increasing the number of active cluster heads to maintain the same probability to resist collusion attacks. In an extreme case, if $q = 0.5$, the number of active cluster heads should be at least 14 to ensure that $\mathbb{P} \leq 0.01$.

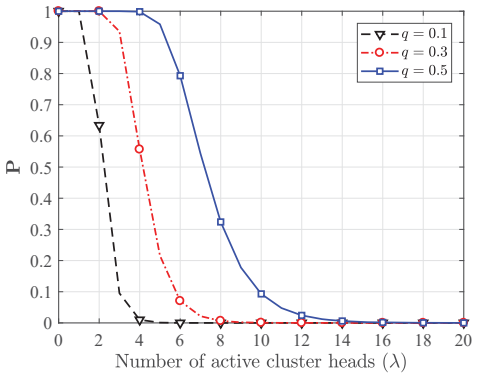

Fig. 2. Collusion analysis

VI. Performance Evaluation

A. Simulation Setup and Baseline

We used Matlab to evaluate the performance of our scheme. In our simulations, we used real power consumption data sets for 114 single-family apartments recorded over one year [6]. The power consumption collection is performed periodically every 1 minute [7]. The size of the encrypted subreading is 512 byte, the size of the encrypted lifetime is 16 byte, the timestamp and the signature sizes are 4 and 64 bytes respectively.

We compare the performance of the proposed scheme at different subreading lifetime values against fixed-transmission-rate AMI in which SMs send their encrypted power consumption readings every reporting period even if the reading is the same. As shown in as in subsection IV-B1, $\mathcal{M}_i$ transmits a homomorphically encrypted subreading, an encrypted lifetime, a timestamp and a signature. Therefore, the total packet size in our scheme is 596 bytes. For the fixed-rate AMI, the packet size is 580 bytes since no lifetime is reported in this baseline.

B. Simulation Results

We evaluate the proposed scheme in terms of communication overhead, defined by the average number of bytes transmitted in the network per SM. Figure 3 gives the communication overhead per SM measured in kilobytes versus a period of one week for fixed-rate AMI and our scheme with different ranges of lifetime values assuming the number of cluster heads $\ell = 8$. As shown in the figure, the communication overhead of our scheme is much better than that of fixed-rate AMI. For

978-1-5386-2321-3/18/$31.00 ©2018 IEEE

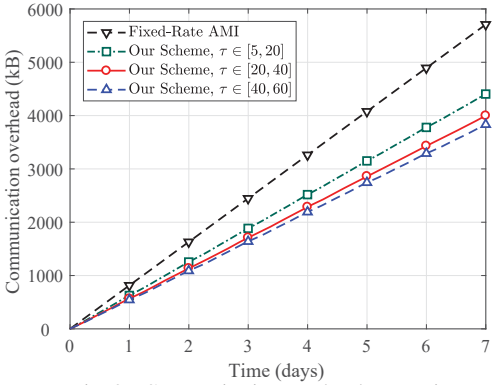

Fig. 3. Communication overhead comparison.

instance, with $\tau \in [5, 20]$ minutes, an average reduction in communication overhead of 19% can be achieved during the week. It can also be seen that increasing the lifetime value to be in $[40, 60]$ minutes results in an average reduction of 30% in the communication overhead. This is because fewer subreadings are needed to be sent to activate cluster heads when the subreading lifetimes expire.

VII. Related Work

Several privacy-preserving schemes have been proposed to preserve consumers' privacy and resist internal attackers in fixed-transmission-rate AMI networks [8], [9]. In [8], Mohammed et al. proposed a privacy-preserving data collection scheme based on one-time masking. However, this scheme cannot be used in enhanced AMI networks since it requires all the SMs to use new masks every data collection cycle. In [9], Fan et al. employed blinding factors to resist internal attackers. However, since the blinding factors are not changed, a simple collusion between the utility and the gateway can reveal meters' reading changes. Therefore, this scheme cannot be applied to the enhanced AMI networks since hiding the reading changes cannot be achieved. Different from periodic data collection schemes, Li et al. proposed in [3] a communication scheme to prevent external attackers from analyzing the presence of house owner by counting the number of transmitted packets assuming trusted internal nodes.

Several schemes have been developed to counter traffic analysis attacks in different wireless networks such as WSNs [10], [11].However, these schemes cannot be applied efficiently to counter traffic analysis in the enhanced AMI networks because WSNs have different characteristics, privacy requirements and objectives. We can summarize the main differences as follows: 1) Most of the proposed schemes aim to preserve the location privacy of the source/destination nodes, but we aim to preserve the consumers' activities in the enhanced AMI networks; and 2) In WSNs, there is a restriction on energy consumption because most of the nodes are battery powered, but this restriction does not exist in AMI networks.

To the best of our knowledge, this paper is the first attempt to propose a privacy-preserving data collection scheme for enhanced AMI networks.

VIII. Conclusion

In this paper, we proposed a privacy-preserving data collection scheme for enhanced AMI networks that can save the communication bandwidth and resist collusion attacks. In the proposed scheme, a cluster of meters is divided into members and heads. Each cluster member splits its reading between a number of cluster heads, permit them to reuse the subreading in the aggregation process by using a lifetime. In addition, each member should monitor the reading changes, lifetimes expiration to determine whether it needs to update one of the subreadings, send a redundant packet to countermeasure traffic analysis attack, or send nothing to save the communication bandwidth. Our analysis demonstrated that the proposed scheme can preserve consumers' privacy and the number of cluster heads can be controlled to achieve a satisfactory protection against collusion attacks. Finally, our simulation results confirmed that the proposed scheme can save the communication bandwidth as compared to fixed-transmission-rate AMI networks by around 30% based on the lifetime values and the redundant packets transmission rate.

Acknowledgement

This work is supported in part by US National Science Foundation under the grant number 1619250.

References

[1] C. Laughman, K. Lee, R. Cox, S. Shaw, S. Leeb, L. Norford, and P. Armstrong, "Power signature analysis," *IEEE power and energy magazine*, vol. 99, no. 2, pp. 56–63, 2003.

[2] M. Mahmoud, N. Saputro, P. Akula, and K. Akkaya, "Privacy-preserving power injection over a hybrid AMI/LTE smart grid network," *IEEE Internet of Things Journal*, pp. 870–880, 2017.

[3] H. Li, S. Gong, L. Lai, Z. Han, R. C. Qiu, and D. Yang, "Efficient and secure wireless communications for advanced metering infrastructure in smart grids," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1540–1551, 2012.

[4] F. Li and B. Luo and P. Liu , "Secure and privacy-preserving information aggregation for smart grids," *International Journal of Security and Networks*, vol. 6, no. 1, pp. 28–39, 2011.

[5] F. D. Garcia and B. Jacobs, "Privacy-friendly energy-metering via homomorphic encryption," in *Springer Security and Trust Management*, 2010, pp. 226–238.

[6] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht, "Smart: An open data set and tools for enabling research in sustainable homes," *SustKDD, August*, vol. 111, p. 112, 2012.

[7] A. Beussink, K. Akkaya, I. F. Senturk, and M. Mahmoud, "Preserving consumer privacy on ieee 802.11 s-based smart grid AMI networks using data obfuscation," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2014, pp. 658–663.

[8] H. Mohammed, S. Tonyali, K. Rabieh, M. Mahmoud, and K. Akkaya, "Efficient privacy-preserving data collection scheme for smart grid AMI networks," in *IEEE Global Communications Conference (GLOBECOM)*, Dec. 2016, pp. 1–6.

[9] C. Fan, S. Huang, and Y. Lai, "Privacy-enhanced data aggregation scheme against internal attackers in smart grid," *IEEE Transactions on Industrial informatics*, vol. 10, no. 1, pp. 666–675, 2014.

[10] S. Alsemairi and M. Younis, "Adaptive packet-combining to counter traffic analysis in wireless sensor networks," in *IEEE Wireless Communications and Mobile Computing Conference (IWCMC)*, 2015, pp. 337–342.

[11] A. Proano, L. Lazos, and M. Krunz, "Traffic decorrelation techniques for countering a global eavesdropper in WSNs," *IEEE Transactions on Mobile Computing*, vol. 16, no. 3, pp. 857–871, 2017.