Chameleon: A Blind Double Trapdoor Hash Function for Securing AMI Data Aggregation

Heng Chuan Tan[Ψ], Kelvin Lim[Ψ], Sye Loong Keoh[Φ], Zhaohui Tang[*], David Leong[Ψ], Chin Sean Sum[†]
[Ψ]School of Infocomm, Republic Polytechnic, Singapore
[Φ]School of Computing Science, University of Glasgow, UK
[*]Singapore Institute of Technology, Singapore
[†]Wireless Smart Utility Networks (Wi-SUN) Alliance, Singapore
Email: {Tan_Heng_Chuan, kelvin_lim_cy, David_Leong}@rp.edu.sg, SyeLoong.Keoh@glasgow.ac.uk,
Zhaohui.Tang@SingaporeTech.edu.sg, sum@wi-sun.org

***Abstract*— Data aggregation is an integral part of Advanced Metering Infrastructure (AMI) deployment that is implemented by the concentrator. Data aggregation reduces the number of transmissions, thereby reducing communication costs and increasing the bandwidth utilization of AMI. However, the concentrator poses a great risk of being tampered with, leading to erroneous bills and possible consumer disputes. In this paper, we propose an end-to-end integrity protocol using elliptic curve based chameleon hashing to provide data integrity and authenticity. The concentrator generates and sends a chameleon hash value of the aggregated readings to the Meter Data Management System (MDMS) for verification, while the smart meter with the trapdoor key computes and sends a commitment value to the MDMS so that the resulting chameleon hash value calculated by the MDMS is equivalent to the previous hash value sent by the concentrator. By comparing the two hash values, the MDMS can validate the integrity and authenticity of the data sent by the concentrator. Compared with the discrete logarithm implementation, the ECC implementation reduces the computational cost of MDMS, concentrator and smart meter by approximately 36.8%, 80%, and 99% respectively. We also demonstrate the security soundness of our protocol through informal security analysis.**

***Index Terms*— Double Trapdoor Chameleon hashing; Elliptic Curve Cryptography; Polynomial-based Key Management**

I. Introduction

Advanced Metering Infrastructure (AMI) is an integrated system consisting of smart meters, communication networks and Meter Data Management Systems (MDMSs) [1]. The smart meters collect energy reports from the household appliances and send them to the MDMS over the communication networks. The communication network provides a two-way communication between the smart meters and the MDMS and supports a wide range of wireless technologies such as cellular, WiSUN [2], WiMAX [3], Bluetooth [4], etc. To reduce the number of message transmissions and bandwidth consumption, a concentrator is deployed in the communication network to aggregate all the energy reports before forwarding the readings to the MDMS. On the other hand, the MDMS is responsible for storing and processing the collected readings for billing purposes. AMI improves the operational efficiency and cost savings related to metering, billing and labor costs. Since the energy consumption values are automatically read and sent to the MDMS, AMI can provide more accurate and timely readings than the current manual method, thus reducing the number of consumer disputes. The end-consumers can better track their usage to save energy and money. For the utility companies who are operating the MDMS, they can monitor the usage patterns of each household to drive more innovations on the type of tariffs they provide.

Despite the huge benefits, it is challenging to achieve secure data aggregation because the concentrators are typically deployed in unattended locations and can be easily compromised by adversaries. In particular, the compromised concentrator can be used to manipulate and tamper with the readings before sending the aggregated data back to the MDMS. This could lead to erroneous bills, energy thefts, and possible consumer disputes. In addition, the data source must be authenticated to ensure that the readings are originated from the intended sender to ensure proper operation.

In this paper, we propose an end-to-end data integrity protocol to secure data aggregation in AMI with the goal of providing data integrity and data source authentication. We adopt the idea of chameleon hashing presented in [5] and introduce several enhancements. Specifically, we propose a chameleon hash function based on the Elliptic Curve Cryptography (ECC) [6] technology to improve the implementation efficiency since the smart meters have limited storage and processing power. In order to prevent the exposure of the trapdoor key through node capture attack, our scheme uses double trapdoor keys to construct the ECC-based chameleon hash function, where each trapdoor key is held by a different entity i.e. the MDMS and the smart meters, respectively. However, the use of two trapdoor keys requires the smart meters to have the MDMS's trapdoor key so that they can compute a commitment to the MDMS to facilitate the reconstruction of the chameleon hash value for verification. To solve this problem, our scheme uses a polynomial-based key management scheme [7] to disseminate the blind copy of the MDMS's trapdoor key without exposing the actual key value. In short, our contributions are as follows:

1. Propose an efficient chameleon hash function based on ECC to provide end-to-end security.

2. Redesign the chameleon hash function using double trapdoor keys to prevent exposure of the chameleon trapdoor keys even if the smart meter is compromised.
3. Propose a polynomial-based key management scheme to facilitate the construction of commitment by the smart meter and the reconstruction of the chameleon hash value by the MDMS.
4. Conduct a performance comparison and security analysis of our protocol with existing work based on the Discrete Logarithmic (DL) assumption.

The rest of the paper is organized as follows: Section II provides a brief survey of relevant work on secure data aggregation. Section III outlines and details the proposed protocol. Section IV and V provides the security analysis and performance evaluation of our proposal. Section VI concludes the paper.

II. Related Work

Several works have been proposed in the literature to preserve integrity for AMI data aggregation. For instance, Li et al. [8] proposed a homomorphic signature scheme for homomorphically encrypted data that supports batch verification of the aggregated results. The basis of their approach is based on the bilinear map. Other similar homomorphic-based signature schemes include [9], [10] and [11]. While these homomorphic-based schemes are promising, they are not practical in that the smart meter must compute a homomorphic signature for each message it transmits before they are aggregated. Therefore, these approaches incur high computational and communication costs, resulting in performance degradation.

Unlike the above approach, Keoh et al. [5] proposed a novel end-to-end data integrity protocol for AMI based on the properties of chameleon hashing [12]. The basic idea is that the owner of the trapdoor key i.e. the smart meter must generate the same chameleon hash value as the concentrator by computing a commitment using its own energy readings so that the MDMS can use it to verify the hash value of the concentrator. In this case, the commitment that is sent to the MDMS need not be signed which makes the scheme very efficient. In [13], Keoh et al. formalized this idea by designing the chameleon hash function based on the DL assumptions. However, the DL approach incurs high computation cost which is not suitable for resource-constrained smart meters. Moreover, their approach is vulnerable to key exposure problem in which anyone with the knowledge of a hash collision can recover the private trapdoor key [14]. Inspired by [13], this paper proposes enhancements to the chameleon hash scheme in order to mitigate these shortfalls.

III. Proposed Scheme

In this section, we introduce our end-to-end data integrity protocol for securing data aggregation. Our protocol is based on a double trapdoor chameleon hash function [15] and is constructed using Elliptic Curve Cryptography (ECC) [6] to achieve better efficiency.

In essence, the concentrator aggregates the energy readings from the smart meters and calculates a chameleon hash value using the public keys that are associated to the two trapdoor keys of the chameleon hash function. To verify the chameleon hash value, a smart meter equipped with one of the trapdoor keys needs to calculate a commitment value using its own energy readings such that the resultant chameleon hash value is equivalent to the previous hash value sent by the concentrator. The commitment value is then forwarded to the MDMS where a second trapdoor key is applied to verify the correctness of the readings sent by the concentrator. To facilitate the construction of the commitment value and subsequently the chameleon hash value, the MDMS embeds a blind copy of its trapdoor key in the polynomial and distribute it to all the smart meters using a polynomial-based key distribution scheme [7]. The entire operation is divided into five phases: **setup, data aggregation, trapdoor collision, hash verification and key blinding**.

A. Setup Phase

In the setup phase, the MDMS generates the system parameters. We assume that the smart meters in the same geographic area form a group and share the same group trapdoor key. We assume that the smart meters are well-behaved and comply with the rules of the protocol, but they may be compromised. We also assume that the compromised smart meter or concentrator act alone, and the problem of collusion is out of the scope of this paper. The system parameters are generated as follows.

- **Generate ECC domain parameters (E, p, a, b, G, n)**
  MDMS determines the ECC domain parameters based on the elliptic curve E of the form $y^2\ (mod\ p) = x^3 + ax + b\ (mod\ p)$ over the finite field, Fp where p is a large prime number and a, b are the coefficients of the elliptic curve. G is a generator denoted by a point (G_x, G_y) selected from the elliptic curve and n is the order of the generator. The security of ECC is derived from the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP).
- **Generate chameleon hash function H_{HK}**
  The smart meter chooses a random value $x \in [1, n-1]$ as the group trapdoor key, computes the chameleon hash public key as $\mathrm{X} = xG$, and register the public key with the MDMS. Similarly, the MDMS selects the second trapdoor key $y \in [1, n-1]$ and computes the corresponding chameleon hash public key as $\mathrm{Y} = yG$. The chameleon trapdoor key is $TK = (x, y)$ and the chameleon public key is $HK = (\mathrm{X}, \mathrm{Y})$. Next, MDMS defines a double trapdoor chameleon hash function as follows.

$$H_{HK}(m, r) = h_2(h_1(m||\mathrm{X}), \mathrm{Y})(X + \mathrm{Y}) + rG\ (mod\ n) \quad (1)$$

  where $h_1: \{0,1\}^* \rightarrow \{0,1\}^k$ is a secure hash function that maps an arbitrary length string to a fixed string of length k and $h_2: Z_n \times \mathbb{G} \rightarrow Z_n$ is a keyed-hashed function that takes as input the result of h_1 and the public key of MDMS.
- **Generate polynomial $P_t(x)$ for smart meters in a group**
  Each smart meter sm_i in group t also receives a unique pre-shared key EK_i from the MDMS for secure unicast communications. Using the pre-shared keys of all the smart meters in the group, MDMS constructs a polynomial and

embeds its trapdoor key y in the polynomial as shown in (2). However, the MDMS selects a secret random value $r'' \in [1, n-1]$ to conceal the actual value of the trapdoor key y to prevent the exposure of the trapdoor key.

$$P_t(x) = (x - EK_1)(x - EK_2) \dots (x - EK_i) + r''y \qquad (2)$$

The polynomial is then preloaded by the MDMS to all the smart meters in a group. Different groups of smart meters will receive different polynomials.

Finally, MDMS publishes the system parameters $(E, p, a, b, G, n, h_1, h_2, HK, HK_{HK}, P_t(x))$ to all the smart meters and concentrators in the network where $P_t(x)$ refers to the polynomial of group t.

B. Data Aggregation Phase

For a group of smart meters $sm_1, sm_2, \dots, sm_n$ where n is the number of meters in the group, each smart meter periodically sends an energy report $m_i^{(t)}$ to the concentrator for aggregation where $m_i^{(t)}$ denotes the readings at time t for sm_i. Upon receiving the individual reports from the smart meters, the concentrator aggregates the individual reports according to equation (3).

$$m_{agg}^{(t)} = \sum_{i=1}^{n} m_i^{(t)} \qquad (3)$$

The concentrator then selects a random value $r^{(t)} \in [1, n-1]$, and calculates the chameleon hash value of the aggregated message using equation (1), with the aggregated readings $m_{agg}^{(t)}$ in (3) and the chameleon hash public key HK as inputs. After that, the concentrator sends the individual smart meter readings $m_i^{(t)}$, the chameleon hash value $H_{HK}^{(t)}$ and the random value $r^{(t)}$ to the MDMS for verification. The message tuple $\left((m_1^{(t)}, m_2^{(t)}, \dots, m_n^{(t)}), r^{(t)}, SIGN(H_{HK}^{(t)})\right)$ is signed by the concentrator's private key using any unforgeable signature scheme to prove authenticity and non-repudiation. At the same time, the concentrator sends $(H_{HK}^{(t)}, h_1(m_{agg}^{(t)}||\mathrm{X}), r^{(t)})$ back to the group of smart meters so that they can produce a chameleon hash collision during the trapdoor collision phase (described in next section). The aggregated message $m_{agg}^{(t)}$ is hashed using h_1 to prevent an adversary from extracting energy readings about other smart meters. The pseudocode of the data aggregation phase is shown in Algorithm 1.

When the MDMS receives the message tuple $\left((m_1^{(t)}, m_2^{(t)}, \dots, m_n^{(t)}), r^{(t)}, SIGN(H_{HK}^{(t)})\right)$ from the concentrator, the MDMS adds up all the received meter readings $m_i^{(t)}$ at time t and computes the chameleon hash value using equation (1) with the given $r^{(t)}$ value. After that, it uses the public key of the concentrator to verify the signature. If the verification is successful, MDMS accepts the integrity and authenticity of the received data, and stores the meter readings as well as the chameleon hash value $H_{HK}^{(t)}$ for end-to-end verification later.

Algorithm 1: Generate Chameleon Hash at time $1 \le t \le T$ (Concentrator)

Inputs:
Chameleon hash public key: $\mathrm{X} = \mathrm{xG}, \mathrm{Y} = \mathrm{yG}$,
Energy readings from different smart meters: $m_i^{(t)}\ \forall i \in (1, \dots n)$,
Random value: $r^{(t)}$

Output: $H_{hk}^{(t)}(m_{agg}^{(t)}, r^{(t)})$

(1) **For** i to n smart meters in a group **do**

$$m_{agg}^{(t)} = \sum_{i=1}^{n} m_i^{(t)}$$

End For

(2) Select a cryptographic secure random integer $r^{(t)}$ from $[1, n-1]$ and compute scalar multiplication $f_1 \leftarrow r^{(t)} \cdot G$

(3) Compute $H_{HK}^{(t)}(m_{agg}^{(t)}, r^{(t)}) = h_2(h_1(m_{agg}^{(t)}||\mathrm{X}), \mathrm{Y})(\mathrm{X} + \mathrm{Y}) + r^{(t)}G\ (mod\ n)$

(3.1) Compute $f_2 \leftarrow h_1(m_{agg}^{(t)}||\mathrm{X})$ using SHA-2

(3.2) Compute HMAC value $f_3 \leftarrow h_2(f_1, \mathrm{Y})$

(3.3) Compute sum of product $f_4 \leftarrow \mathrm{X}f_3 + \mathrm{Y}f_3$

(3.3) Output $H_{HK}^{(t)}(m_{agg}^{(t)}, r^{(t)}) \leftarrow f_4 + f_1$

(4) Send signed $(m_i^{(t)}, r^{(t)}, SIGN(H_{HK}^{(t)}))$ to MDMS

(5) Send $(H_{HK}^{(t)}, h_1(m_{agg}^{(t)}||\mathrm{X}), r^{(t)})$ to smart meters

C. Trapdoor Collision Phase

This phase is executed by the smart meters every T period. On receiving t copies of $(H_{HK}^{(t)}, h_1(m_{agg}^{(t)}||\mathrm{X}), r^{(t)})$ from the concentrator where $t \rightarrow (1 \le t \le T)$, each smart meter selects any one of them to calculate a r' value so that the generated chameleon hash value of its own energy readings m' is equivalent to the chameleon hash value stored by the MDMS at time t i.e. $H_{HK}(m', r') = H_{HK}^{(t)}(m_{agg}^{(t)}, r^{(t)})$ where $m' \neq m_{agg}^{(t)}$. In this case, the message m' is the sum of all the energy readings that the smart meter had sent during period T. To compute the r' value, the smart meter must solve the following.

$$r' = (\mathrm{x} + \mathrm{y})\left[h_2(h_1(m_{agg}^{(t)}||\mathrm{X}), \mathrm{Y}) - h_2(h_1(m'), \mathrm{Y})\right] + r^{(t)}\ (mod\ n) \qquad (4)$$

Since the smart meter does not have knowledge of MDMS's trapdoor key y, it is not able to compute the r' value. So, the smart meter splits the construction of r' value into two components, namely, r_1' and r_2' and sends them as commitments to the MDMS. First, the smart meter substitutes its pre-shared key EK_i into the preloaded polynomial in (2) to retrieve the concealed trapdoor key $r''y$. Using the $r''y$ value, the smart meter executes equation (5) to calculate the two components.

$$r_1' = x\left[h_2(h_1(m_{agg}^{(t)}||\mathrm{X}), \mathrm{Y}) - h_2(h_1(m'), \mathrm{Y})\right] + r^{(t)} \qquad (5)$$
$$r_2' = r''y\left[h_2(h_1(m_{agg}^{(t)}||\mathrm{X}), \mathrm{Y}) - h_2(h_1(m'), \mathrm{Y})\right]$$

Remark 1: If the smart meter is compromised by an adversary, the trapdoor key y is still safe because it is blinded and randomized by the secret value r''.

Algorithm 2: Generate Hash Collision (smart meter sm_i , $1 \le i \le n$)

Inputs:
Chameleon trapdoor key: x,
Smart meter i encryption key: EK_i,
Polynomial: $P(x)$,
Chameleon hash public key: $\mathrm{X} = \mathrm{xG}, \Upsilon = \mathrm{yG}$,
Chameleon hash value, hash of aggregated readings: $H_{HK}^{(t)}, h_1(m_{agg}^{(t)}||\mathrm{X})$,
Random value: $r^{(t)}$,

Output: r' every period T s.t. $H_{HK}(m', r') = H_{HK}^{(t)}(m_{agg}^{(t)}, r^{(t)})$ where $m' \neq m_{agg}^{(t)}$

$$P_i(x) = (x - EK_1)(x - EK_2) \dots (x - EK_i)(x - EK_n) + r''y$$

(1) Substitute EK_i of smart meter i in $P(x)$ to recover secret $r''y$
(2) **For** i to T **do**
(2.1) Sum up readings of smart meter i

$$m' \leftarrow \sum_{j=1}^{T} m_i^{(j)}$$

End For

$$r' = (x + y)\left[h_2(h_1(m_{agg}^{(t)}||\mathrm{X}), \Upsilon) - h_2(h_1(m'), \Upsilon)\right] + r^{(t)} \ (mod\ n)$$

(3) Split r' into two components, r_1' and r_2' where
(3.1) $r_1' \leftarrow x\left[h_2(h_1(m_{agg}^{(t)}||\mathrm{X}), \Upsilon) - h_2(h_1(m'), \Upsilon)\right] + r^{(t)}$
(3.2) $r_2' \leftarrow r''y\left[h_2(h_1(m_{agg}^{(t)}||\mathrm{X}), \Upsilon) - h_2(h_1(m'), \Upsilon)\right]$
(4) Encrypt (r_1', r_2') using EK_i and send it to MDMS

The derived r_1' and r_2' commitments are then sent to the MDMS, encrypted using the smart meter pre-shared key EK_i to provide confidentiality and privacy. Using the commitment values, MDMS will be able to reconstruct the r' value to calculate the chameleon hash value (described in next section) and verify that the previous aggregated messages sent by the concentrator are not tampered with, and that the readings truly originate from the smart meters. The detailed steps of this phase is summarized in Algorithm 2.

D. Hash Verification Phase

When the MDMS receives the commitments (r_1', r_2') from smart meter sm_i, it uses the pre-shared key EK_i of smart meter i to decrypt the message to recover (r_1', r_2'). The MDMS then divides r_2' by r'' to reconstruct the actual value as $\overline{r_2'}$. Using r_1' and the derived $\overline{r_2'}$ values, the MDMS reconstructs the true value of r'as $\overline{r'}$. Next, the MDMS computes m' by summing up all the meter readings m_i of smart meter i for T intervals and computes the chameleon hash value H_{HK}' using equation (6) and the derived $\overline{r'}$. More formally, MDMS calculates the following:

$$H_{HK}(m', \overline{r'}) = h_2(h_1(m'), \Upsilon)(\mathrm{X} + \Upsilon) + \overline{r'}\,G \ (mod\ n) \quad (6)$$

The MDMS compares the calculated $H_{HK}'(m', \overline{r'})$ with the previous value $\left(H_{HK}^{(t)}(m_{agg}^{(t)}, r^{(t)})\right)$ stored in the database at time t. If the two hash values match, it means that the reported readings from the concentrator are consistent with each other,

Algorithm 3: Verification (MDMS)

Inputs:
Components of r': r_1' and r_2'
Aggregated readings of smart meter i over a time period T: m'
Random value to blind MDMS's trapdoor key y: $r'' \in [1, n-1]$

Output: Check $H_{HK}(m', r') \stackrel{?}{=} H_{HK}^{(t)}(m_{agg}^{(t)}, r^{(t)})$

(1) Divide r_2' by r''
$\overline{r_2'} \leftarrow y\left[h_2(h_1(m_{agg}^{(t)}||\mathrm{X}), \Upsilon) - h_2(h_1(m'), \Upsilon)\right]$
(2) Derive $\overline{r'} \leftarrow r_1' + \overline{r_2'}$
(3) **For** i to T **do**
(3.1) Sum up readings of smart meter i

$$m' \leftarrow \sum_{j=1}^{T} m_i^{(j)}$$

End For
(4) Compute $H_{HK}(m', \overline{r'}) = h_2(h_1(m'), \Upsilon)(\mathrm{X} + \Upsilon) + \overline{r'}\,G \ (mod\ n)$
(5) **If** $H_{HK}^{(t)}(m', \overline{r'}) = H_{HK}^{(t)}(m_{agg}^{(t)}, r^{(t)})$ **Then**
Concentrator not compromised
Else
Concentrator is compromised

and are not tampered with. If the verification is unsuccessful, it means that either the concentrator or the reporting smart meter is compromised. To verify the status, the MDMS may request another smart meter in the group to send the commitments to validate the chameleon hash value at time t. If successful, the MDMS concludes that the concentrator is compromised. As long as the majority of the smart meters in the group is trusted, detection of compromised concentrator will always work. The verification process is shown in Algorithm 3.

E. Key Blinding Phase

After each T period, the MDMS randomizes its trapdoor key y to limit the vulnerability of key exposure to increase security. The key randomization interval can be configured to be updated every 6 or 12 hours, depending on the application specifications. To support this requirement, the polynomial-based key management mechanism is used.

First, the MDMS selects a new random blinding value $r_{new}'' \in [1, n-1]$ and computes a new concealed trapdoor key as $r_{new}''y$. After that, the MDMS constructs a new polynomial $P_i'(x)$ for group i using the pre-shared keys of all the smart meters in the group as $P_i'(x) = (x - EK_1)(x - EK_2) \dots (x - EK_n) + r_{new}''y$. The MDMS then broadcasts $P_i'(x)$ to the group smart meters and the concentrator, respectively[1]. Upon receiving the new polynomial, every smart meter in the group uses its secret key EK_i to retrieve the new concealed trapdoor key $r_{new}''y$ by computing $P_i'(EK_i)$. Once $r_{new}''y$ is known, the smart meter follows the hash collision procedure to generate the two commitments so that MDMS can verify all chameleon hash values issued between T and $T + 1$ intervals later. Using polynomial-based approach, updating of trapdoor keys require no further encryption/decryption by the smart meters and the MDMS.

[1] The authenticity of the issued chameleon hash public key Υ' can be verified using Public Key Infrastructure (PKI) and digital certificates

IV. SECURITY DISCUSSIONS

In this section, we analyze the security of our protocol.

A. Key Exposure Freeness

We assume the concentrator is malicious and attempts to intercept the two commitments sent out by the smart meters i.e. r_1' and r_2'. Key exposure freeness requires that on seeing the commitments, the concentrator is not able to extract the trapdoor keys x and y that belongs to the smart meter and MDMS, respectively. If the two trapdoor keys are exposed, the concentrator is able to impersonate as the smart meter to use any false aggregated readings to prove hash collisions. To demonstrate our protocol is safe from key exposure, we analyze our protocol under two cases:

- **Case 1: (Smart meter is not compromised).** The commitment values r_1' and r_2' are encrypted using the pre-shared key EK_i shared between the smart meter and the MDMS. To crack the trapdoor keys of the chameleon hash function, the concentrator needs to solve the ECDLP which is computationally infeasible based on the underlying point multiplication operation and the structure of elliptic curves. Thus, our protocol are protected against key exposure.
- **Case 2: (Smart meter is compromised).** If the smart meter is compromised by an adversary[2], it means that its pre-shared key EK_i and the trapdoor key x are exposed. However, the adversary is still unable to extract the actual value of MDMS's trapdoor key y because this value is blinded by a secret value r'' selected by the MDMS at random. In addition, the trapdoor key y is randomized after every T period. With the knowledge of one trapdoor x, impersonation attacks are not possible. Thus, we can conclude that our protocol satisfies the key exposure freeness property. To protect the smart meters against physical attacks, they can be equipped with Trusted Platform Module (TPM) which provides tamper-resistant hardware for keeping cryptographic keys safe [16].

B. Data Integrity

End to end data integrity is achieved based on the properties of the chameleon hash function, namely, trapdoor collision and collision resistant. This guarantee is conditioned upon the security of the key exposure freeness property that is, the trapdoor keys are not exposed.

- **Trapdoor collision property:** There exists an efficient probabilistic polynomial time (PPT) algorithm A that on input the smart meter's trapdoor key x, the concealed trapdoor key $r''y$ of MDMS, a message pair $\left(h_1(m_{agg}^{(t)}||\mathrm{X}), r^{(t)}\right)$ and an additional self-generated message m', the smart meter is able to output a value $r' \in Z_n$ such that hash collision occurs i.e. $H_{HK}(m', r') = H_{HK}^{(t)}(m_{agg}^{(t)}, r^{(t)})$. This r' value that is sent to the MDMS is represented by r_1' and r_2' and serve as a commitment to assist the MDMS in validating the integrity of the aggregated readings $m_{agg}^{(t)}$ that was reported by the concentrator at time t. If the concentrator modifies the aggregated readings, it can be detected without fail based on this property.
- **Collision resistant property:** There is no probabilistic polynomial time (PPT) algorithm A that on input $HK = (\mathrm{X}, \mathrm{Y})$ and without the knowledge of the trapdoor key pair $TK = (x, y)$, the concentrator is able to find pairs $(m_{agg}^{(t)}, r^{(t)})$ and (m', r') where $m_{agg}^{(t)} \neq m'$ such that $H_{HK}(m', r') = H_{HK}^{(t)}\left(m_{agg}^{(t)}, r^{(t)}\right)$ with a non-negligible probability. This is equivalent to solving the ECDLP problem which is known to be computationally hard. By this property of chameleon hash function, the concentrator is always forced to abide by the rules of the protocol because such forgery can adversely affect its credibility.

C. Data Authenticity

Data authenticity provides assurance that the received messages come from the authorized senders. We analyze our protocol in two aspects to show that it achieves authentication.

- **Case 1: Concentrator →MDMS:** The chameleon hash value $H_{HK}^{(t)}$ sent by the concentrator to the MDMS is signed using a digital signature scheme such as the Elliptic Curve Digital Signature Algorithm (ECDSA) [17]. Assuming a PKI is available and can verify the concentrator's public key using digital certificates, the authenticity of the origin and data can be validated after verifying the ECDSA signature. The ECDSA is secure under the assumptions that the ECDLP is hard and that the hash function is a random function. An unauthenticated concentrator cannot pass off as legitimate to perform data aggregation.
- **Case 2: Smart meter → MDMS:** Each smart meter i is preloaded with a secret pre-shared key EK_i that is shared between the smart meter and the MDMS. The smart meter uses this key to encrypt the commitment (r_1', r_2'). The use of pre-shared key provides guarantee that messages originate from authenticated and unique smart meters.

D. Security of Polynomial Exchange

Whenever a trapdoor key needs to be randomized, the MDMS will broadcast a new polynomial to all the smart meters in the group without encryption. We note that sending the polynomial in clear will not compromise security because the MDMS is sending the expanded form of the polynomial of degree n that is, $P_i(x) = x^n - Ax^{n-1} + \cdots - Bx^2 + Cx - D$ where n denotes the number of smart meters in the group and (A, B, C, D) denote the coefficients of the polynomial. If n is large, it is proven that finding the roots of the polynomial is NP-hard [18]. Therefore, it is not easy to recover the concealed trapdoor key. Moreover, the trapdoor key is blinded by a random r''. Thus, we conclude that the key blinding phase is secure against eavesdropping attacks.

[2] We assume that there is no collusion between the smart meter and the concentrator, and that they cannot be compromised at the same time.

V. Performance Evaluation

In this section, we evaluate and compare the performance of our ECC-based protocol with the DL-based protocol in [11].

A. Implementation Details and Results

Both protocols were implemented in C using OpenSSL 1.0.2k Crypto library. We are interested in the computational time and the CPU cycles with respect to the following:

- Generating a chameleon hash value by the concentrator
- Generating a trapdoor collision by the smart meter
- Verifying the chameleon hash value by the MDMS.

All tests were executed for 1000 times under Ubuntu 16.04 on an i5-3427U CPU@2.3GHz laptop. In our ECC-based chameleon implementation, we used a nistp-256 curve that provides 128-bits security. In the DL implementation, a 2048-bit field with 112-bits of security was chosen. **Table 1** compares the average computing time and CPU cycles for both protocols.

The simulation results show that ECC-based chameleon hashing is significantly more efficient than the DL implementation. The time taken to generate a chameleon hash on the concentrator based on the ECC construction is 1.096 ms while the DL approach requires 5.53 ms. In the case of generating a trapdoor hash collision, the ECC version took only 0.1 ms on the smart meter, while the DL method took nearly 11 ms. The DL implementation is more expensive because the smart meter needs to perform two modular exponentiations to compute the commitments which are computationally costly. In terms of performing hash verification by the MDMS, the ECC implementation improves the time efficiency by a factor of 1.6 over the DL approach. These results show that our protocol is very efficient and well suited for low-powered devices, especially smart meters. A lower computational cost means that more resources can be free up on the device to perform other tasks. It also indicates higher availability to service more requests, thereby improving the scalability.

Table 1: Timing comparison between ECC and DL implementation

Chameleon Hash Protocol	Concentrator	Smart Meter	MDMS
	Time taken/CPU (ms/megacycles)		
ECC-256 bits	1.096 /2.49	0.1/0.21	1.84/4.21
DL-2048 bits	5.53/12.92	10.7/25.85	2.91/6.68

VI. Conclusions

In this paper, we have proposed a novel end-to-end data integrity protocol for AMI to protect data aggregation against message tampering. Our protocol is based on an ECC-based double trapdoor chameleon hashing. Through informal security analysis, we show that our protocol is secure against key exposure problem and provides integrity and authenticity assurances. We also experimented and demonstrated the high efficiency of our ECC-based chameleon hashing by comparing it with the DL method. The simulation results show that the ECC-based implementation can reduce the computational cost of MDMS, concentrator, and smart meters by about 36.8%, 80%, and 99% respectively. Therefore, our protocol is highly suitable for AMI applications. For future work, we plan to implement our protocol on a real AMI testbed to validate the performance on a larger scale and analyze its security using a formal verification tool such as Proverif.

Acknowledgment

This work was supported by the Ministry of Education, Singapore under grant MOE2015-TIF-2-G-002.

References

[1] Y. Kabalci, "A survey on smart metering and smart grid communication," *Renewable and Sustainable Energy Reviews,* vol. 57, pp. 302-318, 2016.

[2] W.-S. Alliance. (2017, July 28). *Introduction to Wi-SUN Alliance* [Online]. Available: https://www.wi-sun.org/images/assets/docs/wi-sun-alliance-overview.pdf

[3] D. Bian, M. Kuzlu, M. Pipattanasomporn, and S. Rahman, "Analysis of communication schemes for Advanced Metering Infrastructure (AMI)," in *PES General Meeting| Conference & Exposition, 2014 IEEE*, 2014, pp. 1-5.

[4] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen, "How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4," in *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, 2012, pp. 232-237.

[5] S. L. Keoh and Z. Tang, "Towards secure end-to-end data aggregation in AMI through delayed-integrity-verification," in *Information Assurance and Security (IAS), 2014 10th International Conference on*, 2014, pp. 6-11.

[6] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation,* vol. 48, pp. 203-209, 1987.

[7] Y. Piao, J. Kim, U. Tariq, and M. Hong, "Polynomial-based key management for secure intra-group and inter-group communication," *Computers & Mathematics with Applications,* vol. 65, pp. 1300-1309, 2013.

[8] F. Li and B. Luo, "Preserving data integrity for smart grid data aggregation," in *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, 2012, pp. 366-371.

[9] N. Saxena, B. J. Choi, and S. Grijalva, "Secure and Privacy-Preserving Concentration of Metering Data in AMI Networks," 2017.

[10] H. J. Jo, I. S. Kim, and D. H. Lee, "Efficient and privacy-preserving metering protocols for smart grid systems," *IEEE Transactions on Smart Grid,* vol. 7, pp. 1732-1742, 2016.

[11] D. He, N. Kumar, and J.-H. Lee, "Privacy-preserving data aggregation scheme against internal attackers in smart grids," *Wireless Networks,* vol. 22, pp. 491-502, 2016.

[12] H. Krawczyk and T. Rabin, "Chameleon Signatures," in *NDSS*, 2000.

[13] S. L. Keoh, K. W. K. Au, and Z. Tang, "Securing Industrial Control System: An End-to-End Integrity Verification Approach," in *Proc. Industrial Control System Security Workshop, Los Angeles, CA, USA*, 2015.

[14] G. Ateniese and B. De Medeiros, "On the Key Exposure Problem in Chameleon Hashes," in *SCN*, 2004, pp. 165-179.

[15] T. Thakur, "An Access Control Protocol for Wireless Sensor Network Using Double Trapdoor Chameleon Hash Function," *Journal of Sensors,* vol. 2016, 2016.

[16] V. Ford, A. Siraj, and M. A. Rahman, "Secure and efficient protection of consumer privacy in Advanced Metering Infrastructure supporting fine-grained data analysis," *Journal of Computer and System Sciences,* vol. 83, pp. 84-100, 2017.

[17] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *International Journal of Information Security,* vol. 1, pp. 36-63, 2001.

[18] D. S. Roche, "Space-and time-efficient polynomial multiplication," in *Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, 2009, pp. 295-302.