# Secure and Efficient Data Storage in Unattended Wireless Sensor Networks

Yi Ren, Vladimir Oleshchuk, and Frank Y. Li
Dept. of Information and Communication Technology, University of Agder, Norway
Email:{yi.ren, vladimir.oleshchuk, frank.li}@uia.no

***Abstract*—Providing forward and backward secrecy is still a big challenge in Unattended Wireless Sensor Networks (UWSNs), though some storage schemes have been proposed. Additionally, high storage requirement needs efficient storage techniques. In this paper, we propose a novel homomorphic encryption and key-evolution based scheme for efficient and secure data storage, which supports both forward and backward secrecy in UWSNs. We show that the stored data based on our scheme can be used to efficiently compute statistic values, e.g., expected value and variance of the sensed data, and at the same time the storage cost is significantly reduced using our scheme. Detailed analysis has been conducted to evaluate the scheme in terms of efficiency and security.**

## I. Introduction

Research aspects of security in Wireless Sensor Networks (WSNs) has gained some progress [1] recently in the research community. Most proposed security schemes, however, are based on assumption that a trusted party (sink or base station) is always available, leading to that unnormal actions of compromised sensors can be detected by the on-line sink [2]. However, in Unattended WSNs (UWSNs) [3], there is no static sink but a mobile sink periodically accesses the UWSN to collect data instead, and the sensor nodes store their sensed data locally or at some special designated nodes within the network till a mobile sink visit, instead of immediately forwarding the data to a centralized location out of the network.

In such a network, a mobile adversary [3] (we denote it as $ADV$ hereafter), which behaves in UWSNs when the mobile sink is absent, can steal data stored in compromised sensors without modifying any sensed data, and interfering with communications of any sensors. In other words, the mobile adversary is *read-only* data during the mobile sink visiting interval, which is impossible to detect by the mobile sink. When a sensor is compromised, as the corresponding secret key is also revealed, the mobile adversary can obtain the encrypted data using the secret key. Therefore, providing data confidentiality is a big challenge. Additionally, in the unattended areas, such as environment monitoring in human-being unfriendly areas, the users of UWSNs usually interested in some "history" data rather "live" data. For instance, what is the average temperature in one month, what is the average activities of volcano in half a year, etc,. Furthermore, in contrast with traditional WSNs, the sensors cannot offload data to an on-line sink at will, instead they will store the data locally till a mobile sink accesses it to retrieve the data. Transmission happens only when the mobile sink visits the UWSN and broadcasts data retrieval requests. Consequently, each sensor must accumulate sensed data and have ability to wait *long enough* until a mobile sink visits it. Since the memory size of sensor nodes is limited, the storage efficiency problem has to be solved.

In this paper, we propose a novel homomorphic encryption and key evolution based scheme for efficient and secure data storage. In our scheme, secret keys of sensors are updated in each round according to one-way hash function. In other words, previous keys cannot be derived from current keys. Forward secrecy is guaranteed. Additionally, storing data collected by sensors are encrypted by using homomorphic encryption and stored as sum of encrypted data and sum of encrypted data squares in sensor memory. Thus, statistic values, such as expected value, variance value, can be easily computed based on those encrypted data values, which can significantly reduce the storage needs. Furthermore, combined with homomorphic encryption and key evolution, our scheme can guarantee both forward secrecy and backward secrecy. We show through detailed analysis that our scheme has low storage cost and low computational cost as compared to other existing schemes and can be suitable for resource-constraint UWSNs.

The rest of the paper is organized as follows. Section II introduces the background and related work. In Section III, the network model and threat model are addressed. Section IV provides the detailed description of our proposed schemes. Section V analyzes performance of the scheme. Finally, Section VI concludes the paper.

## II. Background and Related work

In this section, we introduce related work and some necessary background for our proposed scheme.

### A. Related Work

Achieving data confidentiality in UWSNs is a big challenge because there is no an on-line sink presented. The authors in [4] divide the data collected by sensors into three cases, based on the time when data were compromised: a) before compromise, b) during compromise, and c) after compromise. Since, in case b), sensors are fully controlled, data generated in interval b) are revealed no matter what security schemes are adopted.

To guarantee both forward secrecy and backward secrecy, [4] proposed Distributed Self-Healing scheme (DISH) based on nodes reactive cooperation that provides both forward and

978-1-4244-6273-5/09/$26.00 ©2009 IEEE

(probabilistically) backward secrecy. However, the reactive node cooperative scheme causes extra communication overhead. The authors in [5] proposed a proactive node cooperative scheme which solves the aforementioned problem of [4]. However, both [4] and [5] provided merely certain probabilistically backward secrecy, in an application scenario where reliable communication channels or reliable sensors are needed. To solve this problem, [5] suggested that some suitable public key encryption technique can be used to encrypt round-specific sensor key. However, sensors are known as computation constrained devices that are not very suitable for public key encryption.

### B. Homomorphic Encryption

A privacy homomorphism [6] is an encryption function which allows the encrypted data to be operated on without the knowledge of the decryption function.

The authors in [7] proposed an additive homomorphic encryption scheme such that the addition of the encrypted messages equals the encryption of those messages' summation. The addition of $n$ ciphertexts, $\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} Enc(m_i, k_i, M) = \sum_{i=1}^{n} m_i + \sum_{i=1}^{n} k_i (mod\ M)$. Thus, $Dec(\sum_{i=1}^{n} c_i) = \sum_{i=1}^{n} Dec(Enc(c_i, k_i, M)) = \sum_{i=1}^{n} Enc(m_i, k_i, M) - \sum_{i=1}^{n} k_i (mod\ M)$.

## III. Network Model, Threat Model and Notations

This section presents our network model, threat model and notations.

### A. Network Model

We consider a UWSN that consists of $N$ sensor nodes. A sensor node in UWSN is denoted as $s_i (1 \leq i \leq N)$. There is a mobile sink that visits the UWSN periodically to collect data, (It can be more than one sink in practice. For simplicity, we consider only one mobile sink in the paper.). The sensor $s_i$ generates data at every round, and the data generated at round $r$ is denoted as $d_i^r$. Since sensors have specific sensing task, we assume that data generated in each round requires the same memory where the data are operated in $GF(2^p)$, that is $p = log_2(d_i^r), (r \in (0, R])$, where $R$ is the number of rounds between successive sink visits. We further assume that the data $d_i^r$ generated in each round is not a large value (e.g., $p = 64$ or 160, etc, here, we don't consider audio WSN and multimedia WSN.). Once a data value $d_i^r$ is generated, $d_i^r$ is stored locally, and waits until an authorized mobile sink requires them offload it. Each sensor has the ability to perform one-way hashing and symmetric key encryption. We assume that the mobile sink is a trusted party which cannot be compromised. Additionally, the mobile sink will offload the data accumulated in sensor nodes, re-initializes the secret keys and reset the round counters when the mobile sink visits the network.

### B. Threat Model

In this paper, we investigate that $ADV$ that is only interested in stealing (read-only) the data from the compromised sensor nodes and cannot be detected by the mobile sink. $ADV$ can compromise sensor nodes during the time interval when the mobile sink is absent. Consequently, the mobile adversary can visit the UWSN again and again until the whole network is compromised by it without knowing by the mobile sink. Here, we assume that an adversary can compromise up to $v (v < N)$ sensors in one interval.

### C. Notations

Our notation is listed in Table I.

TABLE I: Notation

| | |
|---|---|
| $K_m$ | master key of a mobile sink |
| $K_i^r$ | secret key of sensor $s_i$ at round $r$ |
| $Sum$ | sum of data, $Sum = \sum d$ |
| $SumEnc$ | sum of encrypted data, $SumEnc = \sum Enc(d)$ |
| $EncSum$ | encryption of data sum, $EncSum = Enc(\sum d)$ |
| $SumS$ | sum of data square, $SumS = \sum d^2$ |
| $SumEncS$ | sum of encrypted data square, $SumEncS = \sum Enc(d^2)$ |
| $EncSumS$ | encryption of data square sum, $EncSumS = Enc(\sum d^2)$ |
| $d_{max}$ | the maximum value of $d_i^r$ |
| $p$ | storage cost for $d_i^r$, $p = log_2(d_i^r)$ (required storage size) |

## IV. Proposed Scheme

In this section, we propose a family of four schemes for efficient and secure data storage. Our goal is to guarantee both forward and backward secrecy while maintaining low storage cost and low computation cost.

### A. Scheme 1 ($S1$)

Initially, the mobile sink picks a master key denoted as $K_m$. The initial key of a sensor node $s_i$ can be computed as $K_i = h(K_m||i)$ which is setup before the deployment of the UWSN. Here, $||$ stands for the concatenation operator. The mobile sink needs to store only one key, because it can use $K_m$ to compute all the secret keys deployed in $s_i$ $(1 \leq i \leq N)$. Then, at round $r \geq 1$, as it collects data, sensor $s_i$ encrypts data $d_i^r$ with $K_i$ by $Enc(K_i, d_i^r)$.

*Computation of expected value and variance*. 1) To compute expected value, the mobile sink needs to retrieve the encrypted data $Enc(d_i^1), Enc(d_i^2), \cdots, Enc(d_i^r), \cdots, Enc(d_i^R)$ from the sensor $s_i$, and then decrypts the encrypted data using its master key $K_m$ by generating corresponding secret key, that is $K_i = h(K_m||i)$. For plaintext data $d_i^1, d_i^2, \cdots, d_i^r, \cdots, d_i^R$, the expected value of data can be easily computed as

$$EX_i = \frac{\sum_{r=1}^{R} d_i^r}{R}. \quad (1)$$

2) The variance of data can be computed as

$$Var_i = \frac{\sum_{r=1}^{R} (d_i^r - EX_i)^2}{R}. \quad (2)$$

*Discussion*. Before the mobile sink retrieves the data, the sensor $s_i$ generates $R$ rounds data. So the storage cost of encrypted data in sensor memory can be denoted as $\sum_{r=1}^{R} log_2(Enc(d_i^r))$. If the value $R$ tends to be a very large value, e.g., 100000, the storage cost tends to a large value as well. Since our goal is to get expected value and variance of data, we transform Eq.(2) as following:

$$\begin{aligned} Var_i &= \frac{\sum_{r=1}^{R} (d_i^r)^2}{R} - \left( \frac{\sum_{r=1}^{R} d_i^r}{R} \right)^2 \\ &= \frac{\sum_{r=1}^{R} (d_i^r)^2}{R} - \frac{(\sum_{r=1}^{R} d_i^r)^2}{R^2}. \end{aligned} \quad (3)$$

Given $Sum_i^R = \sum_{r=1}^{R} d_i^r$ and $SumS_i^R = \sum_{r=1}^{R} (d_i^r)^2$, Eq. (1) and Eq. (3) can be further presented as

$$EX_i = \frac{Sum_i^R}{R}, \tag{4}$$

and

$$Var_i = \frac{SumS_i^R}{R} - \frac{(Sum_i^R)^2}{R^2}. \tag{5}$$

In other words, to compute $EX_i$ and $Var_i$, the mobile sink need to get $Sum_i^R$, $SumS_i^R$, and $R$ instead. Note that the storage cost of sensors can be largely reduced if $Sum_i^R$ and $SumS_i^R$ are stored, instead of storing the encryptions of data generated in each rounds.

*Proposition 4.1:* The storage cost of $Sum_i^R$ and $SumS_i^R$ is less than storing the encryptions of data generated in each rounds, if $R > 1$ and $p > 1$.

Since in UWSN setting, given $R$ tends to a very large value, e.g., 10000, and $p$ is a fixed small value, the storage cost reduce significantly (as shown in Fig. 1) if $Sum_i^R$ and $SumS_i^R$ are stored instead. Thus, these motivate us to design a data summation scheme that can reduce storage cost.

### B. Scheme 2 ($S2$)

The $S2$ is detailed as follows:

Initially, the sensor $s_i$ allocates two parameters $Sum_i^0$ and $SumS_i^0$ in its memory, which denote summation of data and summation of data square, respectively. The main process is shown in Algorithm 1. At the end of the first round, the sensor $s_i$ generates the data $d_i^1$, computes its square $(d_i^1)^2$, initializes $Sum_i^1 = d_i^1$ and $SumS_i^1 = (d_i^1)^2$, encrypts them as $EncSum_i^2$ and $EncSumS_i^2$, and then stores them in its memory. At the end of each round $r$, the sensor $s_i$ decrypts the $EncSum_i^{r-1}$ and $EncSumS_i^{r-1}$ to get the summation of previous data $Sum_i^{r-1} = \sum d_i^{r-1}$ and the summation of previous data square $SumS_i^{r-1} = \sum (d_i^{r-1})^2$, adds current data and data square to them, and stores the encrypted results $EncSum_i^r$ and $EncSumS_i^r$ on local memory.

*Computation of expected value and variance*. After the mobile sink retrieves the $EncSum_i^R$ and $EncSumS_i^R$, it can decrypt them to get summation of data $Sum_i^R$ and data square $SumS_i^R$. Then, the expected value and variance can be easily computed by Eq. (4) and Eq. (5).

*Discussion*. The scheme reduces the storage cost. However, as shown in Algorithm 1 (steps 6,7 and 11,12), in order to obtain $Sum_i^r$ and $SumS_i^r$, the sensor $s_i$ needs to encrypts and decrypts data at each round, which causes very high computation burden. This limitation motivates us to design next scheme that can reduce computation cost and yet maintain low storage cost.

### C. Scheme 3 ($S3$)

To deal with the limitation in $S2$, we now propose a homomorphic encryption based scheme. In this scheme, we utilize additive homomorphic encryption to encrypt new sensed data such that sum of encrypted data corresponds to encryption of those message summation, that is

$$Enc(d_i^r) = Enc(d_i^r, K_i, M) = d_i^r + K_i (mod\ M) \tag{6}$$

and

$$Enc[(d_i^r)^2] = Enc[(d_i^r)^2, K_i, M'] = (d_i^r)^2 + K_i (mod\ M').$$

Due to the properties of additive homomorphic encryption, we can obtain $Enc(d_i^r + d_i^{r+1}) = Enc(d_i^r) + Enc(d_i^{r+1})$. Thus, $EncSum$ and $EncSumS$ can be obtained by

$$EncSum = Enc(\sum_{r=1}^{R} d_i^r) = \sum_{r=1}^{R} Enc(d_i^r) = SumEnc, \tag{7}$$

and

$$EncSumS = Enc(\sum_{r=1}^{R} (d_i^r)^2) = \sum_{r=1}^{R} Enc[(d_i^r)^2] = SumEncS.$$

Therefore, the sensor $s_i$ does not need to decrypt and encrypt at each round to get $Sum$ and $SumS$, but simply add the encrypted data up: $SumEnc_i^R = \sum_{r=1}^{R} Enc(d_i^r)$ and $SumEncS_i^R = \sum_{r=1}^{R} Enc((d_i^r)^2)$.

In the end of round $r$, as shown in Algorithm 2, the sensor $s_i$ encrypt the data $d_i^r$ by using homomorphic encryption and its secret key $K_i$, that is, $Enc(d_i^r) = Enc(K_i, d_i^r, M) = d_i^r + K_i (mod\ M)$. Each encrypted data is added with previous encrypted data and stored locally, $SumEnc_i^R = \sum_{r=1}^{R} Enc(d_i^r) = \sum_{r=1}^{R} d_i^r + R * K_i (mod\ M)$, and $SumEncS_i^R = \sum_{r=1}^{R} Enc((d_i^r)^2) = \sum_{r=1}^{R} (d_i^r)^2 + R * K_i (mod\ M')$. Here, $M$ and $M'$ are defined as $M = 2^{log_2(d_{max} * R_{max})}$ and $M' = 2^{log_2[(d_{max})^2 * R_{max}]}$. We can see that, actually, $M$ and $M'$ are the storage costs of $EncSum$ and $EncSumS$, respectively. To decrypt the $EncSum$ and $EncSumS$, (by computing the corresponding secret key by $R * K_i = R * h(K_m || i)$), the mobile sink can get $SumEnc$ and $SumEncS$ as following:

```
   ; /* sensor s_i starts round r                                   */
 1 collect new sensed data d_i^r;
 2 if r = 1 then
 3 |   set Sum_i^r = d_i^r;
 4 |   set SumS_i^r = (d_i^r)^2;
 5 else
 6 |   compute Sum_i^r = Dec(EncSum_i^{r-1}) = Σ d_i^{r-1};
 7 |   compute SumS_i^r = Dec(EncSumS_i^{r-1}) = Σ (d_i^{r-1})^2;
 8 |   compute Sum_i^r ← Sum_i^{r-1} + d_i^r;
 9 |   compute SumS_i^r ← SumS_i^{r-1} + (d_i^r)^2;
10 end
11 compute EncSum_i^r = Enc(Sum_i^r, K_i, r, ···);
12 compute EncSumS_i^r = Enc(SumS_i^r, K_i, r, ···);
13 store EncSum_i^r, EncSumS_i^r on local storage;
   ; /* end round r                                                 */
```

**Algorithm 1**: Scheme 2

```
   ; /* sensor s_i starts round r                                   */
 1 collect new sensed data d_i^r;
 2 compute Enc(d_i^r) = Enc(d_i^r, K_i, M) = d_i^r + K_i(mod M);
 3 compute
   Enc((d_i^r)^2) = Enc((d_i^r)^2, K_i, M') = (d_i^r)^2 + K_i(mod M');
 4 if r = 1 then
 5 |   SumEnc_i^r = Enc(d_i^r);
 6 |   SumEncS_i^r = Enc((d_i^r)^2);
 7 else
 8 |   compute SumEnc_i^r ← SumEnc_i^{r-1} + Enc(d_i^r);
 9 |   compute SumEncS_i^r ← SumEncS_i^{r-1} + Enc((d_i^r)^2);
10 end
11 store SumEnc_i^r, SumEncS_i^r on local storage;
   ; /* end round r                                                 */
```

**Algorithm 2**: Scheme 3

$$
\begin{aligned}
Sum &= Dec(EncSum) = Dec(SumEnc) \\
&= Dec(\sum_{r=1}^{R} Enc(d_i^r)) = \sum_{r=1}^{R} Enc(d_i^r) - R * K_i (mod\ M) \\
&= \sum_{r=1}^{R} d_i^r \qquad (8)
\end{aligned}
$$

and

$$
\begin{aligned}
SumS &= Dec(EncSumS) = Dec(SumEncS) \\
&= Dec(\sum_{r=1}^{R} Enc((d_i^r)^2)) \\
&= \sum_{r=1}^{R} Enc((d_i^r)^2) - R * K_i (mod\ M') \\
&= \sum_{r=1}^{R} (d_i^r)^2. \qquad (9)
\end{aligned}
$$

*Computation of expected value and variance*. After the mobile sink retrieves the $SumEnc$ and $SumEncS$, it can decrypt them according to Eq. (8) and Eq. (9) to get $Sum$ and $SumS$ by using key $R * K_i$. Then, the expected value and variance can be easily computed according to Eq. (4) and Eq. (5).

*An Example*. For simplicity, we assume that the sensor $s_5$ generates 3 rounds data, $d_5^1 = 11$, $d_5^2 = 23$, and $d_5^3 = 18$, and sets its corresponding secret key $K_5 = 80$ with $M = 100$, and $M' = 1000$. Based on Eq.(6), $s_5$ can compute the encryption of data at the end of round 1, that is, $Enc(d_5^1) = d_5^1 + K_5 (mod\ M) = 11 + 80 (mod\ 100) = 91$, then $s_5$ stores 91 on its local memory. At the end of round 2, $s_5$ generates new data $d_5^2 = 18$; then it gets the encryption of new data by $Enc(d_5^2) = 23 + 80 (mod\ 100) = 3$. Next, $s_5$ does the additive homomorphic encryption to get sum of data $SumE = Enc(d_5^1) + Enc(d_5^2) (mod\ 100) = 91 + 3 (mod\ 100) = 94$. And then, it stores the $SumEnc = 94$ on its memory. Following the same procedure, at the end of round 3, the $s_5$ get $SumEnc = \sum_{r=1}^{3} Enc(d_5^r) (mod\ 100) = 82$. If the mobile sink retrieves the $SumEnc$ from the $s_5$, it can decrypt it by $Dec(SumEnc) = Dec(EncSum) = Sum$ for $SumEnc = EncSum$ based on Eq. (7), that is $Sum = Dec(94) = 94 - 3 * 80 (mod\ 100) = 42$. The $SumS$ can be obtained following the same approach. Therefore, the mobile sink can easily compute the $EX_5$ and $Var_5$ based on Eq.(4) and Eq.(5).

*Discussion*. Comparing to the $S2$, the $S3$ supports both low storage cost and low computation cost. However, a problem is raised if a $ADV$ compromises the sensor $s_i$ at round $r'$. The secret key $K_i$ of $s_i$ is hold by the $ADV$. Consequentially, all the data generated in previous rounds $r \in [0, r')$ will be revealed, because the $ADV$ can use $K_i$ to compute corresponding secret key $(r' - 1) * K_i$. Furthermore, new generated encrypted data are also encrypted by the same key, which means generated data can be decrypted by $ADV$ by using the key $K_i$ as well. To guarantee both forward secrecy and backward secrecy, we propose the next scheme.

### D. Scheme 4($S4$)

If the sensor keeps the secret unchanging, all encrypted data can be read by $ADV$, no matter the data are generated before or after the compromised period, for it only needs to compromise the sensor once and gets the secret key consequently. Therefore, we need to change the secret key after each round to guarantee that the $ADV$ who holds the secret key $K_i^r$ in compromised period $r \in [r', r'']$ cannot derive the secret key $K_i^{\hat{r}}$ in the previous rounds $\hat{r} \in [0, r')$. To solve this problem, we utilize key evolution approach [8], that is, the secret key of a sensor node is updated by its owner. The secret key of $s_i$ in round $r$ is computed as $K_i^r = h^{r-1}(K_i^1)$, where $h(\cdot)$ is an one-way hash function ($K_i^{r-1}$ is then securely erased.). After the secret key $K_i^r$ updates itself at the end of round $r$, the $ADV$ cannot derive the previous round's key before the sensor was compromised (due to $h(\cdot)$ one-way property). Additionally, the mobile sink can easily compute the corresponding secret key of sensor $s_i$ in round $r$ for $K_i^r = h^{r-1}(K_i) = h^{r-1}(K_m||i)$. To decrypt the $EncSum$ and $EncSumS$, similar to Eq. (8) and Eq. (9), after computing the corresponding secret key as $\sum_{r=1}^{R} K_i^r = \sum_{r=1}^{R} h^{r-1}(K_m||i)$, the mobile sink can get $SumEnc$ and $SumEncS$ as following:

$$
\begin{aligned}
Sum &= Dec(EncSum) = Dec(SumEnc) \\
&= Dec(\sum_{r=1}^{R} Enc(d_i^r)) = \sum_{r=1}^{R} Enc(d_i^r) - \sum_{r=1}^{R} K_i^r (mod\ M) \\
&= \sum_{r=1}^{R} d_i^r \qquad (10)
\end{aligned}
$$

and

$$
\begin{aligned}
SumS &= Dec(EncSumS) = Dec(SumEncS) \\
&= Dec(\sum_{r=1}^{R} Enc((d_i^r)^2)) \\
&= \sum_{r=1}^{R} Enc((d_i^r)^2) - \sum_{r=1}^{R} K_i^r (mod\ M') \\
&= \sum_{r=1}^{R} (d_i^r)^2. \qquad (11)
\end{aligned}
$$

Similar to $S3$, the $S4$ has the same homomorphic encryption and decryption process but the secret key. Algorithm 3 shows the operation process in sensor $s_i$.

```
   ; /* sensor s_i starts round r                                   */
 1 collect new sensed data d_i^r;
 2 compute Enc(d_i^r) = Enc(d_i^r, K_i^r, M) = d_i^r + K_i^r (mod M);
 3 compute
   Enc((d_i^r)^2) = Enc((d_i^r)^2, K_i^r, M') = (d_i^r)^2 + K_i^r (mod M');
 4 if r = 1 then
 5 |   SumEnc_i^r = Enc(d_i^r);
 6 |   SumEncS_i^r = Enc((d_i^r)^2);
 7 else
 8 |   compute SumEnc_i^r ← SumEnc_i^{r-1} + E_i^r;
 9 |   compute SumEncS_i^r ← SumEncS_i^{r-1} + E2_i^r;
10 end
11 store SumEnc, SumEncS on local storage;
12 compute K_i^{r+1} = h(K_i^r);
13 erase K_i^r securely;
   ; /* end round r                                                 */
```

**Algorithm 3**: Scheme 4

*Computation of expected value and variance*. After the mobile sink retrieves the $SumEnc$ and $SumEncS$, it can decrypt them by it can decrypt them by Eq. (10) and Eq. (11) to get $Sum$ and $SumS$ using the corresponding secret key $\sum_{r=1}^{R} K_i^r$. Then, the expected value and variance can be easily computed by Eq. (4) and Eq. (5).

*Proposition 4.2:* The $S4$ can provide forward secrecy as well as backward secrecy.

*Proof:* Since key evolution is adopted in the $S4$, the $ADV$ cannot derive the previous key from the current key it hold due to the one way property of hash function, thus the $ADV$ cannot decrypt the data encrypted in previous rounds, which means the forward secrecy is guaranteed. Moreover, the data generated in previous rounds are stored as $SumEnc$ and $SumEncS$ in sensor nodes. To decrypt them, the corresponding secret key $\sum_{r=1}^{R} K_i^r$ is needed, which consists of all keys generated in each rounds. Since the previous keys cannot be derived from current key, the corresponding key $\sum_{r=1}^{R} K_i^r$ cannot be obtained by $ADV$, that means backward secrecy is guaranteed. Therefore, the $S4$ can guarantee both forward secrecy and backward secrecy. ■

## V. Analysis

In this section, we investigate numeric performance of the proposed schemes with respect to storage cost, and computational cost. Security analysis is also given in this section.

### A. Storage Cost

Fig.1 (left) shows the analysis results of proposed schemes in term of storage cost. We can observe that the $S1$ has the highest storage cost in all four proposed schemes no matter $p = 64$ or 128, and that the rest three scheme has the lowest storage cost, that means the storing summation of data can significantly reduce the storage cost in sensors. We further observe that the storage cost of $S2$, $S3$, and $S4$ are almost zero when they are compared with $S1$, and the influence of $p$ cannot be identified. Therefore, as shown in Fig.1 (right), we ignore the storage cost of $S1$, where only $S2$, $S3$, and $S4$ are plotted in terms of $p = 64$ and $p = 128$, respectively. We observe that $p$ can increases with a little influence on the storage cost.

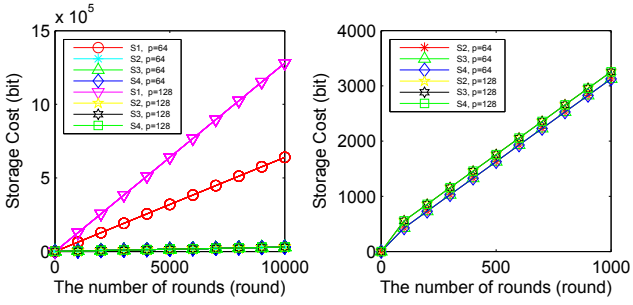


Fig. 1: The storage comparison between proposed schemes.

### B. Computation Cost

In $S1$, sensors encrypts data sensed in each round and stores them locally. In $S3$ and $S4$, sensors encrypts data by doing simply modulo arithmetic in each round, which based on homomorphic encryption. In $S2$, however, sensors decrypts encrypted data sum, updates new data sum, and encrypts it again which causes extra computational cost.

### C. Security Analysis

In $S1$, $S2$, and $S3$, both forward secrecy and backward secrecy cannot be guaranteed for the secret key is the same in each round. Once the secret key is compromised, all the encrypted data are revealed. As discussed in Section IV-D Proposition 4.2, the $S4$ can guarantee both forward secrecy and backward secrecy.

Finally, we summarize the comparison results between $S1$, $S2$, $S3$ and $S4$ in TABLE II in terms of storage cost, computation cost, Forward Secrecy (FSe) and Backward Secrecy (ASe). TABLE II shows the conclusion that $S4$ has the best conformance compared with the rest three schemes in terms of storage cost and computation cost, and that $S4$ can further guarantee forward secrecy and backward secrecy.

TABLE II: Performance comparison results between proposed schemes in terms of Storage Cost, Computation Cost, Forward Secrecy (FSe) and Backward Secrecy (ASe).

| Scheme | Storage Cost | Computation Cost | FSe | ASe |
|---|---|---|---|---|
| $S1$ | High | High | No | No |
| $S2$ | Low | Very High | No | No |
| $S3$ | Low | Low | No | No |
| $S4$ | Low | Low | Yes | Yes |

## VI. Conclusion

In this paper, we have proposed a novel homomorphic encryption and key evolution based scheme for efficient and secure data storage in UWSN. Detailed analysis demonstrates that our schemes accomplish the goals of data confidentiality and efficiency with respect to storage and computation.

## References


[1] V. Giruka, M. Singhal, J. Royalty, and S. Varanasi, "Security in wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 8, no. 1, pp. 1–24, 2008.

[2] B. Sun, L. Osborne, Y. Xiao, and S. Guizani, "Intrusion detection techniques in mobile ad hoc and wireless sensor networks," *IEEE Wireless Communications*, vol. 14, no. 5, p. 56, 2007.

[3] R. Di Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik, "Catch me (if you can): Data survival in unattended sensor networks," in *Proc. IEEE PerCom '08*, Hong Kong, 2008, pp. 185–194.

[4] D. Ma and G. Tsudik, "DISH: Distributed Self-Healing," in *Proc. 10th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS '08*, Detroit, MI, USA, 2008, pp. 47–62.

[5] R. Di Pietro, D. Ma, C. Soriente, and G. Tsudik, "POSH: Proactive co-Operative Self-Healing in Unattended Wireless Sensor Networks," in *Proc. IEEE Symposium on Reliable Distributed Systems, SRDS '08.*, Napoli, Italy, Oct. 2008, pp. 185–194.

[6] E. Brickell and Y. Yacobi, "On privacy homomorphisms," *Advances in Cryptology, Eurocrypt '88*, vol. 87, pp. 117–125, 1988.

[7] C. Castelluccia, E. Mykletun, and G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks," in *Proc. ACM/IEEE MobiQuitous '05*, San Diego, CA, USA, July 2005, pp. 109–117.

[8] M. Bellare and B. Yee, "Forward-security in private-key cryptography," in *Proc. CT-RSA '03*, San Francisco, CA, USA, 2003, pp. 1–18.