REISCH: Incorporating Lightweight and Reliable Algorithms into Healthcare Applications of WSNs

Mishall Al-Zubaidie 1,2,* , Zhongwei Zhang 2 and Ji Zhang 2

1 Department of Computer Science, Education College for Pure Science, Thi-Qar University, Nasiriyah 64001, Iraq
2 Faculty of Health, Engineering and Sciences, University of Southern Queensland, Toowoomba, QLD 4350, Australia; Zhongwei.Zhang@usq.edu.au (Z.Z.); Ji.Zhang@usq.edu.au (J.Z.)
* Correspondence: u1070801@umail.usq.edu.au or Mishall.Al-Zubaidie@usq.edu.au; Tel.: +61-469-869-029

Received: 3 February 2020; Accepted: 11 March 2020; Published: 15 March 2020

Abstract: Healthcare institutions require advanced technology to collect patients’ data accurately and continuously. The tradition technologies still suffer from two problems: performance and security efficiency. The existing research has serious drawbacks when using public-key mechanisms such as digital signature algorithms. In this paper, we propose Reliable and Efficient Integrity Scheme for Data Collection in HWSN (REISCH) to alleviate these problems by using secure and lightweight signature algorithms. The results of the performance analysis indicate that our scheme provides high efficiency in data integration between sensors and server (saves more than 24% of alive sensors compared to traditional algorithms). Additionally, we use Automated Validation of Internet Security Protocols and Applications (AVISPA) to validate the security procedures in our scheme. Security analysis results confirm that REISCH is safe against some well-known attacks.

Keywords: ECDSA; healthcare; homomorphic; integrity; pseudonym; WSN

1. Introduction

Medical records of patients require accurate, secure, and efficient electronic systems to be managed and organized. Electronic medical record (EMR) systems are extremely useful for managing patients’ data. These systems are widely disseminated in the health sector [1]. Moreover, EMR systems need patients’ data collection technology such as a wireless sensor network (WSN). This technology consists of a group of sensing nodes that communicate wirelessly with each other to gather data about a particular environment in various applications. A WSN often has limited resources such as energy and memory, but it provides comfort, speed, accuracy and safety to humans by monitoring a specific area without human intervention or presence [2]. An important application that has brought the attention of sensor networks to many researchers is healthcare (HC), because of great importance in our lives to reduce the effects of diseases on the health of patients. Providing better HC quality of lower cost will be a key aim of all health industries over the next decades [3].

These applications relying on the use of WSNs are known as healthcare wireless sensor networks (HWSNs) [4]. By using HWSNs, healthcare providers including physicians, doctors, and nurses can access data and information about patients on an ongoing basis, whether at clinics or in hospitals. Therefore, this medical record leads to more accurate diagnosis and thus is likely to lead to an improvement in the patient’s condition. For many diseases that require constant monitoring and precise care, HWSN is the best method used by doctors to get patients’ data, as this technology provides patients with comfort and more care at less cost [2]. Since these applications monitor patients’ activities without interruption, accurately and continually, they should lead to an improvement in their health condition [5].
Disclosure of medical records for patients in the HC systems results in weak security in these systems. In addition, some security mechanisms such as public key signing significantly affect WSN performance. Therefore, several issues need to be addressed when designing schemes for collecting data in HC applications. These issues are critical to the acceptance and success of HC applications in the healthcare sector. These issues listed are as follows:

- **Communications security:** To protect data and information between source and destination, security mechanisms, such as signatures should be applied to prevent an attacker from accessing records transferred between network entities (for example, sensor, Cluster Head (CH), and Base Station (BS)/Local Server (LS)). These mechanisms should resist attacks such as disclosure, alteration, replication, collision, preimage, and impersonation of medical records transmitted. The communication channel should be protected end-to-end at both the wireless level and on the Internet through the integration of a set of security mechanisms and privacy [6,7].

- **Datasets security:** Medical records stored on the EMR server as a repository become the target of malicious attacks. In particular, if a HC application is based on a single server, the process of hacking this server results in both data and information being detected [8]. Besides, access to datasets without pseudonyms and signature mechanisms makes it easy for attackers to detect users’ real identities (IDs). To protect users’ medical records, the EMR server should not contain real information for users to prevent detection of users’ identities or tampering with datasets. For instance, an EMR server contains only signatures and pseudonyms and users’ identifiers are stored on the remote server, such as an Attributes Server (AS). Furthermore, the database should be available to legitimate users at any time and from anywhere, and should support authorization requests for access to partial data from an EMR repository and patients’ history from a remote server, such as a Data Server (DS) [9].

- **Performance of collection devices:** WSN requires efficient security algorithms to work efficiently. EMR systems use WSN to collect patient data. However, a WSN is source-constrained in terms of energy, computing and memory. Therefore, when using signature mechanisms, security and performance should be efficient. The efficiency of these algorithms is a major challenge in HC applications. Namely, the sensor nodes should be very efficient to collect patient data accurately and for a long time while protecting the data collected from the penetration [10,11].

Many attacks undermine the security of the WSN of collecting patients’ data and threaten the privacy of the EMR repository. These attacks have been classified into passive, active, internal, and external attacks [12-14]. For instance, potential attacks on data transferred or stored in an EMR repository by WSN are a serious risk to HC systems. As shown in Figure 1, Intruder 1 can listen to data as they are transferred from the patients’ sensors or CHs to the server (LS/BS). When the attacker intercepts the message, he/she can obtain information about the physical location of the patient, ID, timestamps, source address, target address, and the medical report sent by the sensors or directed by medical staff devices. The patients’ data transmitted through the sensor networks requires complete security, especially when movement through the network does not require the consent of the patient, such as moving the data of an emergency case [15]. Additionally, Intruder 2 can perform an attack on the LS/remote Central Server (CS) to penetrate the datasets to obtain patient information. An attacker can also get information from the datasets, such as the patient’s name, age, address, type of disease, and the seriousness of the disease. This information allows the attacker to harm the patient in different ways, such as changing or destroying data [16]. However, designing a patients’ data collection scheme with strong and heavy signing mechanisms without regard to network performance in data collection is useless and infeasible for HWSN systems. Therefore, performance and security issues are essential to provide care services in HC applications.
1.1. Our Contributions

We propose Reliable and Efficient Integrity Scheme for Data Collection in HWSN (REISCH) to ensure that patient data is transferred/stored to the LS/BS securely and efficiently. The REISCH is characterized as follows:

- REISCH applies the Elliptic Curve Digital Signature Algorithm (ECDSA) with BLAKE2bp instead of ECDSA with Secure Hash Algorithm 1 (SHA1) to improve HWSN lifetime and prevent intruders from altering/changing patients’ data.
- REISCH used the homomorphic mechanism with CHs to reduce energy consumption when aggregating patient data from sensors.
- REISCH hides the sensor’s identification (SID) and location (SL) by using random pseudonyms. This mechanism prevents intruders from detecting sensors information transmitted between network terminals.
- Formal security analysis in REISCH is simulated by an automated validation of Internet security protocols and applications (AVISPA). This tool is dramatically accepted as an effective way to validate threat models in HWSN. AVISPA is used to check that our scheme is secure against both passive and active attacks.

1.2. Paper Structure

The rest of this research is organized as follows. Section 2 discusses existing research related to our study. The trust model, threat model, and an overview of techniques used in REISCH are explained in Section 3. Section 4 provides details about the proposed data collection scheme. Section 5 discusses security and performance analysis of the REISCH scheme. Finally, Section 6 presents the conclusion and future work directions.

2. Related Existing Research

This section briefly discusses existing studies [17–25] designed to secure patient data in the EMR and highlights their drawbacks.

Fan and Gong [17] implemented ECDSA on Micaz motes with the binary field (163-bit). They improved signature verification via cooperation of the adjacent nodes. ECDSA’s implementation was also presented in the sensor node (IRIS) [18]. However, because this node supported 8-bit of the microcontroller, the author modified the SHA1 code from the 32 bits original to 8 bits. Through implementation, the original algorithm is better in size and time than the modified algorithm. The author explained the possibility of using the ECDSA algorithm with the sensor node (IRIS) held 8-bit microcontroller.

To store patient data accurately, data collection schemes should rely on reliable and fast hash algorithms in ECDSA. The authors of [19] applied the ECDSA algorithm as a lightweight authentication scheme in the WSN, demonstrating the effectiveness and efficiency of using ECDSA in WSN in terms of...
security and performance. Staudemeyer et al. [20] designed an ECC/ECDSA-based scheme to provide privacy in WSN. However, they did not provide a performance analysis of the security algorithms during exchange of data in WSN. Malathy et al. [21] focused on the efficiency of transmission in WSN to extend the lifetime of sensor nodes with the use of ECDSA and generated message digest (MD) with data. Their scheme relied on a colony optimization scheme to save energy in the WSN. However, it did not support privacy parameters during data transfer. Sharavanan et al. [22] proposed a scheme to monitor the heterogeneous network environments in WSN and protect the medical information of patients using ECDSA. Unfortunately, their scheme addressed only the computation processes of transport. It did not address the complicated computation processes that generate and verify the signature in ECDSA.

Recently, Sui and de Meer [23] designed a data aggregation scheme that focused on computation in demand-response management to improve performance and security efficiency. Their scheme was based on the identity signature (Bilinear Map) to protect information and data aggregated by integration and authentication. Hathaliya et al. [24] proposed an elliptic curve cryptography (160 bits) scheme to encrypt and authenticate patients’ biometric properties. They used wearable sensors to collect patient data and used a mobile device to send and store these data in the medical repository (cloud server). Finally, Furtak et al. [25] designed a framework based on RSA-2048 bits and trusted modules to secure the sensors’ domain and prevent unauthorized threats. They organized sensors into master, replica, and gateway categories in the network area and data structure in the sensor memory. In their framework, security procedures for the domain and sensor were used to support both integrity and authentication. Moreover, many researchers [26–30] have pointed out that ECDSA is particularly appropriate for authentication and authorization schemes because it performs lightweight processes during security procedures. Many recent studies [31–35] have also pointed out that SHA1 suffers from collision, preimage, and second preimage attacks. However, no schemes addressed SHA1 performance and security (collision, preimage, and second preimage) problems in ECDSA.

3. Preliminary Techniques for Our Data Collection Scheme

Data collection technology should be efficient and secure to meet the requirements of health institutions. The HWSN requires techniques to perform the data collection procedures before storing patients’ data on the EMR server. To guarantee that only legitimate sensors are associated with the trusted LS, our scheme uses a set of security techniques to integrate and authenticate the collected data and detect false data in LS. In our scheme, we depend on algorithms that provide efficient lightweight operations and a high-security level for signature operations. This section presents the trust model, the threat model and the basic review of REISCH’s techniques.

3.1. Trust Model

We assume that the history of patient data and information is stored on remote and safe servers (CS, AS, and DS). LS does not contain patients’ real information. It contains only signatures and pseudonyms. Additionally, LS is not associated with CH outside a certain range. Moreover, the authorization provider of the network cannot know the correlation between patient information and data, times, and locations. Legitimate users can access partial data at LS without disclosing confidential patient information.

3.2. Threat Model

Building a threat model in HWSN is important to identify serious attacks on patients’ data and subsequent disclosure. HWSN provides important services to the health sector compared to traditional computer networks, such as LAN and MAN, but they are more vulnerable than the latter. These networks rely on self-organization and synchronization to increase the flexible communications of sensor nodes, but HWSN suffers from a security vulnerability. Because of the wireless radio signals in WSN, it is easier for attackers to intercept data transmitted among sensors nodes, CHs, and LS. These networks are targeted for many attacks that exploit resource-constrained, untrustworthy communication and unattended processes. HWSN threats are as follows:
• The attacker performs a man in the middle (MITM) attack to modify or replay attack to resend the data to the CH/LS. The attacker’s aim is to use his/her device as a legitimate sensor in the network.
• The attacker can execute a denial of service (DoS) attack on the CH/LS. This attack exploits a heavy transmission of duplicate or counterfeit data to destroy the HWSN.
• The attacker can apply several types of localization attacks such as Sybil, Wormhole, and Sinkhole to intercept of network communications.
• The attacker performs an attack to penetrate the EMR repository in the LS, to access the patient’s data and reveal their identities.
• The attacker can launch an eavesdropping attack to obtain patients’ data, and then perform an analysis of these data to detect the linkability among data, information, and pseudonyms.
• The attacker can copy a legitimate sensor ID in more than one counterfeit sensor. These counterfeit nodes send modified data to the network (node replication attack).
• Collision, preimage, and second preimage attacks can be implemented to change signatures and data transferred between a network’s devices.

3.3. Overview of Techniques Used in REISCH

• **Electronic medical record (EMR)**
  A medical record is a communication entity used to record and review patients’ health status for members of the medical staff and patients themselves. Medical records are divided into two categories: paper and electronic records. The paper record is a traditional method used to check and record patient information. This type of medical record suffered from many problems when dealing with patient data. These problems include delay, errors, lack of coordination of care equality at different levels, management of health information and data, integration of scientific evidence into HC services and decision-making practices [36], and security issues. The electronic medical record rapidly processes and transmits data across digital devices. EMRs provide HC services continuously and accurately. It has attracted the attention of both the HC industry and researchers because it provides advantages in efficiency and effectiveness. Consequently, recent studies [36–38] have indicated that the electronic medical record (EMR) reduces adverse effects among patients and providers because of its many advantages:

1. It is easy for the patient to review his/her information/data; users can review the medical record at the same time, auto-update, and quickly retrieve information.
2. Patient understanding of care services is improved; it facilitates patient participation and cooperation in decision-making.
3. It reduces errors in documents and reduces the embarrassment of the patient with a professional.
4. It increases transparent cooperation and improves the interaction between the patient and the providers.
5. The use and quality of health information, quality of care, efficiency, cost of care, facilitate data collection, retrieval, and use of patients’ data are improved.

EMR is efficient because it provides many features and supports the use of WSN. EMR is defined as a one-organization system [39]. Currently, most Australian professionals use an EMR, which is rated similarly in several countries such as Germany, New Zealand, and the Netherlands [40]. EMR stores patient health data within a single institution and uses WSN to store patient data in a local repository for use in reports, disease diagnosis, and treatment. However, an EMR only contains a partial patient medical history [40]. For example, doctors may use an EMR to identify a patient’s prescription and avoid errors, and the nurse may use an EMR to monitor tests and reports for a patient. However, if the doctor needs complete data about a patient’s medical history, he/she needs to send a request to the CS. However, performance and security efficiency are the main issues when using WSN with an EMR.
• Security properties of EMRs

Security of EMRs relies on the elliptic curve discrete logarithm problem (ECDLP) [41]. ECDSA utilizes small parameters which improve the performance of computations, thus diminishing process time and storage. These features are essential for large institutions and limited-resource devices such as WSN because these networks require intensive/complex processes, memory, or power consumption [42].

Since the intruder can tamper with the collected data (d) when they are transferred from sensors to LS, patients’ data integrity is important in the HWSN environments [43]. Many reputable organizations such as NIST and IEEE use ECDSA as standard [44]. ECDSA with a 160-bit key achieves the equivalent for symmetric cryptography with a 80-bit key [45]. It is suitable for limited-resource devices because it produces small keys and provides computation speed in the integrity process. Furthermore, ECDSA uses four-point multiplication (PM) operations: one PM each for public key and signature generation and two for verification. Besides, it comprises three procedures: key generation, signature, and verification. These procedures are described as follows:

– Key generation:

1 Select a pseudorandom integer private key (K_{pr}) and compute public key (K_{pu} = K_{pr}G)

– Signature generation:

1 Select a pseudorandom integer k, 1 \leq k \leq n-1.
2 Compute e = SHA1 (d) and kG = (x_1, y_1).
3 Compute r = x_1 \mod n. If r = 0 then go to Step 1.
4 Compute k^{-1} \mod n and s = k^{-1}(e + K_{pr}r) \mod n. If s = 0 then go to Step 1, else signature for the message m is (r, s).

– Signature verification:

1 Verify that r and s are integers in the interval [1,n-1].
2 Compute e = SHA1 (d).
3 Compute w = s^{-1} \mod n, u_1 = ew \mod n, u_2 = rw \mod n and X = u_1G + u_2K_{pu}.
4 If X = \theta, then reject the signature. Otherwise, convert the x-coordinate x_1 of X to an integer \bar{x}_1, and compute v = \bar{x}_1 \mod n, accept the signature if and only if v = r.

ECDSA becomes inappropriate to sign d if applied poorly and incorrectly. It becomes reliable if the parameters are validated effectively [46]. In REISCH, we use ECDSA-256 bit to add a high-security level and take care to consume system resources.

• Integrity and authentication of EMRs

In this subsection, we explain the two one-way hash functions, both of which are related to our study.

– SHA family

Secure Hash Algorithm (SHA) is one of the traditional hash algorithms that provides integrity and authentication when used with digital signatures. For instance, SHA1 was used in the ECDSA algorithm to perform the signature process. SHA consists of several varieties: SHA0, SHA1, SHA2, and SHA3. Both SHA2 and SHA3 consist of SHA224, SHA256, SHA384, and SHA512, but SHA3 uses a different structure than the rest of the SHA family. SHA0, SHA1 and SHA2 are built on the basis of the Merkle–Damgard structure, as shown in Figure 2 [47], and were designed by the National Security Agency (NSA). SHA3 is also known as KECCAK and is built on sponge construction and uses two-stage absorbing and squeezing. Since 2007, NIST has adopted KECCAK because of the practical attacks on SHA0, SHA1, and SHA2. KECCAK became the rival standard in 2015 [47]. However, some research [48,49] has indicated that SHA3 can suffer from fault injection threats.
SHA is a one-way function consisting of two phases that divide the message into blocks of the same size (such as 512 or 1024). A set of zeros is added and followed by one at the end of the last block of the message [18]. This phase is called preprocessing or padding. The second stage is the MD computation. At this stage, all message blocks are entered into the iterations (SHA1 (80), SHA2 (64), and SHA3 (256)) one by one, containing constants and logic operations (OR, AND, and XOR) in the compression function \( F \) to produce MD. Each hash algorithm produces a fixed length of MD such as 160 for SHA1 and 224, 256, 384, and 512 for SHA2 and SHA3 [50]. Table 1 shows the comparison between SHA1, SHA2, and SHA3 [51].

Many existing schemes to collect data in WSN [18,52–54] have used the SHA algorithm to support integrity and authentication. However, these schemes do not address the collision and preimage problems in the SHA algorithm [31–35].

**Table 1.** Comparison of SHA family.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SHA1</th>
<th>SHA2</th>
<th>SHA3</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD</td>
<td>160</td>
<td>224</td>
<td>256</td>
</tr>
<tr>
<td>Block size</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>Message size</td>
<td>(&lt;2^{64})</td>
<td>(&lt;2^{64})</td>
<td>(&lt;2^{64})</td>
</tr>
<tr>
<td>Iterations</td>
<td>80</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>Security</td>
<td>80</td>
<td>64</td>
<td>64</td>
</tr>
</tbody>
</table>

**Weak security**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SHA1</th>
<th>SHA2</th>
<th>SHA3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes, practical such as Collision and preimage</td>
<td>Yes, practical such as preimage and length extension</td>
<td>Yes, theoretical such as fault injection</td>
<td></td>
</tr>
</tbody>
</table>

**Performance**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SHA1</th>
<th>SHA2</th>
<th>SHA3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast</td>
<td>Less</td>
<td>Lowest</td>
<td></td>
</tr>
</tbody>
</table>

**Year**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SHA1</th>
<th>SHA2</th>
<th>SHA3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>2004</td>
<td>2015</td>
<td></td>
</tr>
</tbody>
</table>

**Designer**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SHA1</th>
<th>SHA2</th>
<th>SHA3</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSA</td>
<td>Guido Bertoni and et al.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Construction**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SHA1</th>
<th>SHA2</th>
<th>SHA3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merkle–Damgård</td>
<td>Sponge</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**BLAKE family**

Aumasson et al. [55] proposed a BLAKE hash algorithm to overcome the efficiency problems in previous hash algorithms. This algorithm offers several features such as simplicity, speed, and parallel operations in hardware and software implementations. It is immune to second preimage, side-channel and length-extension attacks. BLAKE implements HAIFA construction which is an enhanced version of Merkle–Damgård. This development of construction is accomplished by adding a salt and a counter to the algorithm stages to prevent security vulnerability for second preimage attacks in Merkle–Damgård. BLAKE’s local wide-pipe structure also makes collision attacks impossible [47]. BLAKE uses the LAKE hash algorithm and compresses the message blocks in hash-tree constructions with Bernstein’s stream cipher ChaCha, which is a variation of Salsa20-256. Skein and Grostl [56] considered NIST, a BLAKE
of competing algorithms, in the final round of hash algorithms such as KECCAK. BLAKE supports several versions: 244, 256, 384, and 512. Subsequently, Aumasson et al. [57] developed BLAKE2 to improve the speed in software implementation and to reduce memory. BLAKE2 has 32% less memory than BLAKE. In addition, BLAKE2 contains two versions, BLAKE2s and BLAKE2b, to be used with 32-bit and 64-bit platforms, respectively. Moreover, the authors developed the latest versions, BLAKE2bp and BLAKE2sp, to improve the speed of MD production during parallel processes. Table 2 shows the BLAKE family [50].

Table 2. Versions of BLAKE hash function.

<table>
<thead>
<tr>
<th>BLAKE Version</th>
<th>Word</th>
<th>Message</th>
<th>Block</th>
<th>MD</th>
<th>Salt</th>
<th>Round</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAKE-28</td>
<td>32 bits</td>
<td>&lt;2&lt;sup&gt;64&lt;/sup&gt;</td>
<td>512</td>
<td>224</td>
<td>128</td>
<td>14</td>
</tr>
<tr>
<td>BLAKE-32</td>
<td>32 bits</td>
<td>&lt;2&lt;sup&gt;64&lt;/sup&gt;</td>
<td>512</td>
<td>256</td>
<td>128</td>
<td>14</td>
</tr>
<tr>
<td>BLAKE-48</td>
<td>64 bits</td>
<td>&lt;2&lt;sup&gt;128&lt;/sup&gt;</td>
<td>1024</td>
<td>384</td>
<td>256</td>
<td>16</td>
</tr>
<tr>
<td>BLAKE-64</td>
<td>64 bits</td>
<td>&lt;2&lt;sup&gt;128&lt;/sup&gt;</td>
<td>1024</td>
<td>512</td>
<td>256</td>
<td>16</td>
</tr>
<tr>
<td>BLAKE2s/BLAKE2sp</td>
<td>32 bits</td>
<td>-</td>
<td>256</td>
<td>128-256</td>
<td>64</td>
<td>10</td>
</tr>
<tr>
<td>BLAKE2b/BLAKE2bp</td>
<td>64 bits</td>
<td>-</td>
<td>512</td>
<td>160-512</td>
<td>128</td>
<td>12</td>
</tr>
</tbody>
</table>

Figure 3 shows the architecture of the BLAKE hash function. In BLAKE, the message is divided into blocks, and the last block is padded with 1 followed by zeros to complete the last block size to 512 or 1024 bits. BLAKE consists of two parts: the compression function and iteration mode. The compression function consists of chain value, message blocks, salt value, and counter value. The BLAKE compression function uses three phases: initialization, round functions, and finalization. The initialization phase uses the chain value, salt, and counter to create a 4 × 4 matrix, and produces a 16-word value for different initializing states (V). These states are entered into the round function (r) with parallel rounds in the phase of the round functions. The output of this phase is a new V that is used to generate the chain value for the finalization phase. In the finalization phase of the chain, salt and new state values are applied with ⊕ operations to produce a new chain value. BLAKE is one of the fastest hash algorithms and has strong security [50,58]. Recent research has pointed out that BLAKE is a suitable algorithm for source limited devices [59,60].

![Architecture of BLAKE hash function.](image)

- **De-identification mechanism**

  Encryption and k-anonymity mechanisms are applied to hide patients’ data. However, these mechanisms suffer from serious shortcomings. For instance, encryption of collected data [61] has the following drawbacks:
1. A temporary HC provider such as a researcher doctor will not benefit from the encrypted data, and, if he/she is able to get the collected data by the decryption process, this is a security weakness in the HWSN system.

2. Huge datasets encryption is dramatically burdening for the LS system, which causes complexity of operations and processor power consumption [62].

3. The datasets of collected data perform intensive and continuing operations on medical records such as add, delete, and edit, and, if the records are encrypted, this will multiply the burden on the LS [63].

4. Encryption can contain implicitly direct information about the HC patients. The breach of this encryption will expose the patients’ information to intruders [64].

The k-anonymity of collected data suffers from the following:

1. The removal process of all the patients’ information obstructs the HC provider from dealing with the linked patients’ data [61].

2. Inserting a large set of false medical records would greatly reduplicate the dataset size. Consequently, this process consumes LS resources, particularly with the intensive and continuous access of the datasets by HC providers.

To address these disadvantages, we use random pseudonyms in REISCH’s requests to hide the correlation of patients’ information with data. The medical records transmitted/stored among the sensors, CHs and LS, do not contain any patients’ real information. This mechanism prevents the intruders from identifying patients’ IDs. In addition, this mechanism is fast and does not need complex operations. When the EMR system wants to add a new HC provider/patient, the REISCH sends a request to the remote servers (CS and AS), which provides LS with the required information about updating random pseudonyms. These random pseudonyms are linked with the users’ IDs. This mechanism enables sensors to access and store a specific patients’ data without exceeding granted privileges.

- **One time passcode (OTP)**

OTP is a forceful way of validating sensors in HWSN environments if applied with reliable signature technologies. Using a static passcode/nonce without other validation mechanisms is a security weakness with respect to attacks. Thereupon, OTP presents significant support to the validation process. This mechanism is a countermeasure against replay, MITM, and DoS threats [65]. The intruder cannot utilize this passcode/nonce to authenticate the HWSN later. The sensor sends OTP within a validation request. If the validation process is achieved, the LS will delete OTP from memory and it will be unacceptable to use it again. OTP provides a strong mechanism to relieve the intruders’ risks in the HWSN communications. In REISCH, we apply OTP to get a random nonce with each link to sensors in HC applications to guarantee that only legitimate sensors are communicated to the HWSN.

- **Efficient HC data management using XML**

The other important part of the proposed EMR system is the repository. The repositories contain data in various contexts since these systems have difficulties dealing with these different coordinates for data. The extensible access control (XML) is considered convenient for the exchange of various data via different environments. XML is the symbolic, simple, and flexible language designed to manage, describe, and exchange data across the Internet. It divides the data into the form of useful information through data organization, for the purpose of sharing data across different systems and storing them in the dataset. Moreover, XML has several features that make it suitable for data management, such as support for unicode, the representation of computer data structures (trees, records, and lists), and using a formula read by both humans and computer. However, XML should support the security mechanisms to provide different levels of protection of
sensitive data in the whole or part of the XML document [66]. Access to the data is a challenge in big data management systems that use different techniques. In addition, the exchange of information over an insecure environment has become essential, particularly in HC applications. However, this information needs mechanisms to identify the arrival of unauthorized users to protect patient data. Patient data transmitted between sensors (nodes and CHs) and network devices (such as a nurse and a LS device) need data management algorithms to maintain both performance and security at the same time. EMR including patients’ confidential data and private information needs to be accessed by HC professionals. Thus, sharing such EMR without breaching a patient’s privacy requires EMR management in an efficient and secure manner. XML technology has begun showing its superiority in the exchanging of complex data over different systems.

- **Homomorphic scheme**

A homomorphic is a mechanism for merging all messages and signatures together to improve both performance efficiency and security. This mechanism consists of many types such as linear, polynomial, full, and aggregate signature [67]. In this study, we focus on the aggregate signature because it deals with multi-sensors signatures, messages, and different private keys depending on different devices such as sensors. Furthermore, this process is extremely suitable for multihop-based networks during the integration of signatures into a single signature. We assume that we have a range of messages $M = \{m_1, \ldots, m_n\}$ and a range of signatures $S = \{s_1, \ldots, s_n\}$. $M$ contains all group’s messages together, where $S$ is one signature for all signatures, $A$ is an aggregate function, and $V$ is a verification function. The process of homomorphic signatures is as follows:

- Each device generates $K_{pr}$ and $K_{pu}$ keys and broadcasts the $K_{pu}$ keys to network members.
- Each device signs the $m$ by the signature algorithm, which includes the device’s ID, message and private key $s(K_{pr}, m_i, ID)$.
- The aggregation procedure in the intermediate nodes such as CH relies on $A$ to collect all public keys, messages and signatures $A(K_{pu1}, \ldots, K_{pu_i}; m_1, \ldots, m_n; s_1, \ldots s_n)$.
- The verification procedure will be in the final entity such as LS, which uses $V$ to validate the signatures $V(K_{pu1}, \ldots, K_{pu_i}; m_1, \ldots, m_n; s_1, \ldots s_n)$. If the verification process fails, it means that the data integrity operation is incorrect.

The homomorphic aggregate signature scheme is important to support the performance of network devices by making the intermediate nodes such as CH perform a single signature process for all members’ signatures of the group rather than the signature verification process (the ECDSA verification process consumes more time and energy than the signature process) [68]. In addition, homomorphic increases security measures in preventing the tracking of patients’ information and data or changing signatures of legitimate network devices [69].

4. The Proposed Data Collection Scheme

In this section, we provide details about REISCH that possesses security and performance efficiency features in HWSN. The section consists of three parts: the network model, security goals, and proposed data collection protocols.

4.1. Network Model

Figure 4 shows the network model in which our proposed REISCH scheme is based:

1. Sensor (SN): This entity collects raw data related to a specific patient. It sends these data to the CH.
2. Cluster Head (CH): This entity aggregates data from the sensors that followed it. Then, it sends these data to the LS.
3. Local Server (LS): This entity receives data from all CHs in each round and stores it in EMR’s repository. These data are subsequently sent periodically to the Central Server (CS).
4. Central Server (CS): This entity is a gateway accessing remote servers such as the Attributes Server (AS) and the Data Server (DS). It receives data from the LS and sends data to the DS after being authenticated by the AS. Security procedures in AS and DS are left for future directions.

Among the WSNs, the low-energy adaptive clustering hierarchy (LEACH) protocol is used. LEACH uses clustering architecture to improve the WSN lifetime. More details about this protocol are available in [70]. Each group of SNs collects raw data for a specific patient [71]. These SNs sign data before sending them to CHs. Each CH aggregates data and signatures from his followers. Then, each CH uses homomorphic property with all data and signatures without verifying the signatures to reduce energy consumption on the CH and send them to the LS. As the LS has unlimited resources, it verifies and validates collected data from SNs. The LS sends data stored on the EMR’s repository to the central repository to allow HC users (patients and providers) to access them by sending authentication/authorization requests to the CS, AS, and DS. This paper focuses on performance and security issues in SNs, CHs, LS, and CS. Security issues for datasets and transferred data in CS, AS, and DS are left for future works.

### 4.2. Security Goals of REISCH Scheme

The REISCH has the following security services:

- **Information confidentiality**: This service is achieved to hide SNs/patients’ identities and to protect patients secrecy from disclosure by intruders.
- **Data integrity**: This service is required to protect the patient data from tampering by intruders. The collected data should arrive at the intended target without alteration to provide a reliable communication channel among SNs, CHs, LS, and CS [72].
- **Non-repudiation**: This is a feature to prove that the m is sent by a particular SN in the HWSN. If a legitimate entity in HWSN performs internal attacks, he/she cannot deny his/her messages while availing the privileges granted to him/her.
- **Forward secrecy/Backward secrecy**: This requirement is performed when network entities use new keys and parameters temporarily without depending on old ones in the future. While backward secrecy prevents the newly joined sensors from accessing previous messages before entering the HWSN.
• **Freshness**: It indicates that the data collection message is new and updated to guarantee that the intruder cannot replay the previous message at a later time. This goal is accomplished by a checking of time, a random passcode, and random signatures within each data collection round to counteract spoofing risks such as replay, MITM, and impersonation.

• **Security of Localization**: This feature ensures that the patient/sensor’s real location is protected from detection, or sends error messages to the LS by an intruder.

• **Scalability**: HWSN applications elaborate in a scalable environment in both data and devices. Thus, these applications need data collection schemes capable of processing and adapting to the ever-increasing number of devices of the HWSN. This feature indicates the ability of the data collection scheme to properly handle huge HWSN devices. Public key signature schemes are convenient to provide this requirement [73].

• **Survivability**: It provides a certain level of services in patient data collection or network capability to withstand failure/threats in an appropriate manner and continue to provide services between SNs and LS for as long as possible.

• **Accountability**: This property means tracking the behaviour of malicious threats/suspicious activities by legitimate users/counterfeiting attacks in accessing EMR repository.

• **Efficiency**: HWSN sources such as energy, storage, and processor should be within the design objectives of security protocols in HWSN.

4.3. REISCH’s Scheme

In this subsection, we explain the details of REISCH in terms of entities preparation, using ECDSA-BLAKE2bp, applying a camouflage signature, and implementing homomorphic and REISCH protocols.

4.3.1. Entities Preparation

To start collection and storage processes, the HWSN network should be prepared with the following points:

• Each sensor \( (SN_i) \) and LS server provides \( SN_i \) pseudonym \( (SN_{Pseud}) \), \( SN_i \) pseudonym signature \( (Sig_{LS_i}(SN_{Pseud})) \) and \( SN_i \) location \( (SNSL) \).

• All entities \( (SN_i, CH_i, LS \ and \ CS) \) generate \( K_{pu_i} \) and \( K_{pr_i} \) to apply asymmetric cryptographic.

• Each entity broadcasts \( K_{pu_i} \) to network members.

• Each \( SN_i \) uses ECDSA signatures \( (Sig_{SN} \ and \ Sig_{CH}) \) to achieve collected data integrity.

• Each server \( (LS \ and \ CS) \) uses ECDSA signatures \( (Sig_{LS} \ and \ Sig_{CS}) \) to achieve storage data integrity.

4.3.2. Using ECDSA-BLAKE2bp

REISCH implements ECDSA-BLAKE2bp (NIST prime 256-bit) to sign all messages \( (m) \) among HWSN entities \( (SN_i, CH_i, LS \ and \ CS) \). The collected data are formatted as XML-enabled files to allow different devices in the HWSN network to deal easily and flexibly with these records. We use the BLAKE2bp algorithm instead of SHA1 to perform the hash function on collected and stored data (Section 3.3; in the second point in both signature generation and signature verification, we use BLAKE2bp \((d)\) instead of SHA1 \((d)\)). In REISCH, we use ECDSA-BLAKE2bp to ensure data integrity as well as add \( SN_{Pseud} \) within \( Sig_{SN} \) to prevent changing data. \( LS \ and \ CS \) accept only valid signature after verification. The high performance and security of the ECDSA-BLAKE2bp algorithm makes it an appropriate choice to protect EMR health records. Using ECDSA-BLAKE2bp with XML also adds the feature of managing medical records in HWSN.

4.3.3. Applying Camouflage Signature

REISCH uses the camouflage process to hide the data signature and completely prevent traceability, analysis or alteration of data. The camouflage process starts by signing the data to obtain a 64-bit MD
and then adding a 64-bit counterfeit signature to a total length of 64-bit + 64-bit = 128-bit. In addition, each \(SN_i\) adds padding (0000) to become the total length of the 132-bit signature, as shown in Figure 5. \(SN_i\) performs the process of exchanging data signature segments based on Parity (even/odd) value. It receives this value invisibly from the \(LS\) because this value is included in the ephemeral random value (\(SigLsE_i\)). \(SN_i\) tests \(SigLsE_i\); if “even”, it divides the 132-bit into four segments (each segment to 33-bit) and exchanges the segments. Then, \(SN_i\) truncates the 32 bit from the first segment and divides it into four segments (each segment to 8-bit). If \(SigLsE_i\) is “odd”, it divides the 132 bit into three segments (each segment to 44-bit) and then exchanges the segments. It then truncates the 42-bit and divides the first segment into three segments (each segment to 14 bit). Because the exchanging operation is based on Parity sent from the \(LS\), this process prevents the detection of the original signature of the data and prevents the data from being changed. Thus, this process protects patient data from tampering or alteration.

**Figure 5. Camouflage signature.**

### 4.3.4. Implementing Homomorphic

REISCH uses the homomorphic property with the ECDSA-BLAKE2bp algorithm to increase network performance. Because the verification process in ECDSA consumes more time and processing than the signature process, it is convenient to use the homomorphic property in HWSN to support both performance and security. The LEACH protocol is based on the principle of clustering to reduce energy consumption, thus REISCH uses the aggregate signature to allow \(CH_i\) to aggregate signatures and data without using verification. To double security in REISCH, \(CH_i\) performs the process of aggregating temporary signatures such as \(SigSnT_{36}\) and \(SigSnT_{46}\) in addition to random numbers (\(SN_{RNS}\)) and data. Temporary signatures contain unclear original signatures that prevent an intruder from penetrating patient data. The homomorphic procedure reduces energy consumption and thus increases the possibility of using the ECDSA algorithm with HWSN for as long as possible.

### 4.3.5. REISCH’s Protocols

The REISCH scheme consists of three protocols. During these protocols, REISCH provides reliable data collection processes to protect collected patients’ data.

**Protocol 1 between \(SNs\) and \(CHs\):**

This protocol performs the data collection process (Figure 6 shows the first protocol processes between \(SN_i\) and \(CH_i\) in the data collection). The process is as follows:

\(SN_i\) Side

- At the beginning of each round, each \(SN_i\) receives a one-time passcode (\(LS_{OTP}\)) and a random number (\(SN_{RNS}\)). This \(LS_{OTP}\) contains an ephemeral random value (\(SigLsE_i\)) of the same length as the signature. \(SN_i\) extracts \(SigLsE_i(SN_{Pseud})\) from the dataset and executes \(\oplus\) to extract the secret value \(SigLsE_i\).
• Then, $SN_i$ executes the Parity (as shown in Section 4.3.3) process based on $SigLsE_i$ to get the temporary signature ($SigSnT_1$).

• After that, $SN_i$ generates an ephemeral random value ($SigSnE_i$) with the same signature length and uses it with $SigSnT_1$ to compute the $SigSnT_2$ value.

• Next, $SN_i$ computes the $Di f$ value that represents the subtraction value of the distance between $CH_i$ and $SN_i$ ($SN_iCHD$) and the distance between $LS$ and $SN_i$ ($SN_iLSD$). $Di f$ specifies that $SN_i$ is within the HWSN framework (1000 m $\times$ 1000 m).

• Additionally, $SN_i$ computes a new timestamp ($SN_{TS}$) and one time passcode ($SN_{OTP}$). $SN_i$ also performs a hidden process for $SN_{TS}$ and $SN_{OTP}$ at a temporary value ($SN_{TS_i}$) with the addition of a value of only seconds (SS) at the end of the $SN_{TS_i}$.

• Furthermore, $SN_i$ uses $SN_p$ to concatenate secret parameters such as $SN_{TS_i}$, $SN_{OTP}$, $SN_{RN_i}$, $SN_{Pseud}$ and $SN_{SL}$ to match them at the $LS$. To protect both $SN_p$ and $SigSnE_i$, $SN_i$ uses the $\oplus$ operation to hide them by calculating the temporary values of $SigSnT_3$ and $SigSnT_4$. At this point, $SN_i$ computes the message ($SN_m$) and sends it to $CH_i$ which is a sequence of $SigSnT_3$, $SigSnT_4$, $SN_{RN_i}$, $Di f$, $SN_{TS_i}$ and data collection.

$CH_i$ Side

• $CH_i$ also receives $LS_{OTP}$ of the $LS$ and $SN_m$ of $SN_i$.

• Afterwards, $CH_i$ truncates $Di f_i$ and tests its value within the HWSN framework by computation $Di f_i \leq$ Maximum value, where the maximum value should be less than or equal to 707.1068.

• Then, $CH_i$ computes the timestamp ($CH_{TS_i}$) to prevent late messages.

• $CH_i$ truncates $SS$ from $SN_{TS_i}$ to obtain $SN_{TS_i}$. If the difference between $CH_{TS_i}$ and $SN_{TS_i}$ is less than the $\Delta T$ delay rate (we assumed that $\Delta T = 3$), namely, that the message is fresh.

![Collection protocol:](image)

- $SN_r$ to $CH_i$
- From $SN_i$ (data collection message)
- Receives $LS_{OTP}$ from $LS$ and $SN_m$ from $SN_i$
- Extracts $LS_{OTP} \oplus SigLS(SN_{Pseud})$
- Computes $SigSN = SigSN(Data)\mid SN_{Pseud}$
- Specifies $SN_{Pseud}$ for $SigLS_i$
- $SigSnT_1 = exchange SigSN segments by $SN_{Pseud}$
- $SigSnT_2 = SigSnT_1 \oplus SigSnE_i$
- $SN_{CLD} = (CH_{TS_i} - SN_{TS_i})^2 + (CH_{TS_i} - SN_{TS_i})^2$
- $SN_{CLD} = (LS_{TS_i} - SN_{TS_i})^2 + (LS_{TS_i} - SN_{TS_i})^2$
- Computes $Di f = SN_{CLD} - SN_{CLD}$
- Generates $SN_{TS_i}$ and $SN_{OTP}$
- Computes $SN_{TS_i} = Date \oplus Time \oplus Sn_{OTP} \oplus Di f$
- $\mid SS$
- $SN_r = SN_{TS_i} \mid SN_{TS_i} \mid SN_{TS_i} \mid SN_{OTP} \mid SN_m$
- $SigSnT_3 = SN_r \oplus SigSnT_2$
- $SigSnT_4 = SigSnE_i \oplus SigSnT_2$
- $SN_m = SigSnT_3 \mid SigSnT_4 \mid SN_m \mid Di f \mid SN_{TS_i} \mid Data$

![Figure 6. Data collection protocol.](image)

Protocol 2 between $CHs$ and $LS$:

This protocol performs the data aggregation process (Figure 7 shows the second protocol processes between $CH_i$ and $LS$ in the data aggregation). The process is as follows:
• Each CH receives temporary signatures, random numbers and collected data from its SN followers.
• Then, CH executes the signature process SigCH for the temporary signatures received (SigSnT3) of its SN followers.
• Thereafter, CH extracts the SigLsEi unique value from LSTOP, similar to the first protocol based on SigLS1(CHpseud) stored.
• Next, CHi Performs CHParity process based on SigLsEi extracted (as described in Section 4.3.3) to compute SigChT1. Moreover, CHi computes SigChT2 depending on the SigChT1⊕SigLsEi operation.
• After that, CHi generates CHTS2 and CHOTP to prevent the problem of replaying messages later. CHi calculates CHp which represents the sequence of secret parameters. In addition, CHi computes CHA to complete the process of aggregating temporary signatures (SigSnT3a and SigSnT4a), random numbers (SNRNA), and collected data (Data).
• Finally, CHi computes CHm and sends it to the LS.

LS Side
• After LS sends LSTOP for all SNi, it waits to receive CHm of all CHi per round. The LS truncates CHRN, SSi and CHA from each CHm received. It uses SSi to reconfigure CHTS2; in addition, the LS generates a timestamp (LSTSi) and tests △T between LSTSi and CHTS2 to confirm the freshness of the message.
• Then, it tests whether CHRN matches the value previously sent. If CHRN is correct, it is used to determine CHpseud and the latter is used to determine CHi location (CHSL).
• Afterwards, the LS retrieves temporary signatures and random numbers (SigSnT3, SigSnT4 and SNRNA) from CHA. The LS uses the SigLsEi value to specify a Parity (even/odd) value for all SNi and CHi. It computes a signature (SigLS1i) for all SNi signatures that followed a specific CHi (SigSnT3) and exchange the SigLS1i segments based on CHParity.
• After that, the LS calculates SigLsT1i which equals SigChT2 in CHi based on SigLS1i⊕SigLsEi. To ensure the legitimacy of CHi, the LS extracts the secret parameters at CHp and tests the match CHpseud and CHSL in the datasets.
• At this point, the LS checks for data integrity collected by SNi. Similarly, the LS uses SNRNA to determine SNpseud and performs data signature (SigLS2) that equals the SigSN in SNi and exchanges the SigLS2 segments based on SNParity.
• Next, the LS uses SigSnT4 and SigLsEi to extract SigSnEi. Thereafter, the LS uses SigSnT3 and SigSnEi to compute SigLsT2i.
• Finally, the LS extracts the secret parameters for SNi from SNp and tests matching SNpseud and SNSL in datasets. If all signatures and parameters are validated correctly, then that the data collected by SNi are legitimate and correct and have not been tampered with by the intruder.
Appl. Sci. 2020, 10, 07631

Figure 7. Data aggregation protocol.

Protocol 3 between LS and CS:

This protocol performs the data storage process (Figure 8 shows the third protocol processes between the LS and CS in the data storage). The process is as follows:

**LS Side**

- In sending case to CS, the LS initially generates a new pseudonym (LS_pseud) and timestamp (LT_{2,5}) to prepare for the process of sending data to the CS.
- Then, the LS computes the SigLS signature based on the CS’s old pseudonym (CS_pseud).
- After that, the LS generates and sends LSOTP to the CS, which is based on the SigLS, LS_pseud, LT_{2,5}, as well as appending SS at the end of LSOTP.
- In receiving case from CS, LS truncates parameters embedded within CM. Thereafter, the LS generates LT_{3,5} to check the arrival time of CM.
- Furthermore, the LS computes CSOTP that relying mainly on LS_pseud. Afterwards, the LS extracts CS_pseud to calculate SigLS3. The LS tests matching SigLS3 and SigCS, and if the result is identical, this means that mutual authentication process between the LS and CS is performed correctly and legitimately.
- After this stage, the LS prepares the data storage request to CS. First, the LS generates LT_{3,5} and LS_{RN} to ensure randomness and freshness.
- After that, the LS computes the SigLS4 signature that depends on the LS_{T1} temporary parameters.
- Then, the LS computes the SigLS5 data signature that depends on temporary parameters such as LS_{T2}, LS_{T3}, and SigLS4 as well as the Data.
- Finally, the LS sends LS_m which includes SigLS5, SS, LS_{RN} and Data to CS.
CS Side

- In sending case to LS, CS generates $C_{TS_1}$, $C_{Pseud_1}$, $C_{OTP}$, and $C_{RN}$. CS uses $C_{TS_1}$ to test the message arrival time of the LS. Depending on generated secret parameters, such as $C_{OTP}$, CS computes $C_{T1}$ and $C_{T2}$ temporarily.
- In addition, the CS generates a $Sig_{CS}$ signature that includes the temporary value ($C_{T2}$).
- At this point, the CS computes and sends $C_{Sm}$ to LS containing the sequence of parameters such as $Sig_{CS}$, $C_{T1}$, $C_{T2}$, $SS$ and $C_{RN}$.
- In receiving case from LS, CS receives $L_{Sm}$ of LS. The CS generates $C_{TS_2}$ new to test access time $L_{Sm}$. The CS calculates $Sig_{CS_1}$ and $Sig_{CS_2}$ similarly to $Sig_{LS_4}$ and $Sig_{LS_5}$ respectively.
- At this point, the CS checks matching $Sig_{CS_2}$ and $Sig_{LS_5}$, and, if the result is identical, it means that CS received patients’ data from the LS correctly and integrated without any changes by malicious attacks.

5. Discussion

In this section, we discuss the security and performance analysis for the REISCH scheme. Analyses demonstrate that REISCH is efficient for use in patient data collection within the HWSN environment in terms of security and performance.

5.1. Security Analysis

In this section, the theoretical and experimental security analysis is provided to examine REISCH protocols in repelling known attacks.

5.1.1. Theoretical Analysis

In this section, we examine the REISCH scheme theoretically with the set of threats mentioned in the threat model. We provide a theoretical analysis of REISCH resistance to known attacks as follows:
• MITM and replay

Proof 1: An intruder tries to change or delete part of data/information when transferred between the network’s entities. This situation is not possible because REISCH applies the ECDSA algorithm to sign data as well as some information such as $SN_{Pseud}$. Additionally, an intruder cannot replay a message late due to the REISCH’s entities use of timestamps such as $SN_{TS}$ and $CH_{TS}$. Consequently, REISCH resists MITM and replay attacks successfully. □

• DoS

Proof 2: An intruder applies a DoS attack to destroy availability service in servers such as the $LS$ and $CS$. The servers in REISCH initially check lightweight parameters such as $SigLsE_i$ in $LS$ and $CS_{Pseud}$ in $CS$ before completion of the authentication process. Moreover, these parameters change randomly in the communication process between entities. This procedure allows servers to check small parameters and prevent DoS duplicate messages. Therefore, REISCH withstands DoS threats. □

• Localization

Proof 3: An intruder tries to use the Sybil attack by using many legitimate SN IDs with fake data. Since $SN_i$ waits for random $SigLsE_i$ from $LS$ each round, the intruder cannot deceive $LS$ with fake data. Additionally, an intruder uses Wormhole attack by using many $SN_i$ to camouflage communications between network entities. Each $SN_i$ sends implicitly $SN_{Sl}$ to $LS$ and $Diff$ to $CH_i$ as well as a timestamp. These parameters prevent counterfeit communications. Furthermore, if an intruder aims to apply a Sinkhole attack using node as a sink to attract all patient data from $SN_i$, it cannot apply to REISCH because $LS$ sends a unique $LS_{OTP}$ including $SigLS_{i}(SN_{Pseud})$ for all $SN_i$. That intruder fails to detect $SigLS_i$ and $SN_{Pseud}$. Hence, REISCH strongly overcomes localization attacks. □

• EMR repository

Proof 4: Assume that an intruder can penetrate datasets in $LS$. First, $LS$ does not contain real patient information (real information such as the name is stored in $AS$). When the intruder gets these data, he/she cannot disclose that they belong to a particular patient. Second, the $LS$ ‘datasets tremendously are difficult to penetrate. Furthermore, $LS$ contains partial data for patients because the total data and patient history are transferred to $DS$ by $CS$ periodically. Thereupon, REISCH resists the EMR repository attack. □

• Eavesdropping

Proof 5: When an intruder eavesdrops and gets some of the messages transferred among $SN_i$, $CH_i$, $LS$, and $CS$, this intruder will not benefit from these messages that are being trapped because these messages contain no real information. Furthermore, the secret parameters, are completely hidden. Thus, REISCH prevents eavesdropping attacks from revealing patient information. □

• Node replication

Proof 6: An intruder applies a node replication attack using more than one $SN_i$ with the the same legitimate ID. In REISCH, we suppose that all $SN_i$ are inside a specific area in the hospital or clinic. Therefore, any $SN_i$ outside this area finds it extremely difficult to send messages from fake $SN_i$ with same legitimate ID. In addition, $LS$ waits $SN_{m}$ by $CH_i$ at the same number of $SN_i$ and the $LS$ removes replicated $SN_{m}$ or $SigSnT_3$. In addition, when $SN_i$ dies, $LS$ records this situation in the dataset to prevent replication risks. As a result, REISCH effectively resists replication attacks. □

• Collision and preimage

Proof 7: An intruder tries to implement a collision (the generation of two different messages that produce the same MD $=h(m) = h(m')$), preimage (the generation of a message that produces the same existing MD value as $h(m) = MD$), and second preimage (the generation of a different message from the received message and produce the same existing MD value) attacks when messages and signatures are transferred between REISCH’s entities. These attacks cannot be
implemented on REISCH protocols because our protocols use the BLAKE2bp hash instead of SHA1, which resists these attacks. Consequently, REISCH successfully prevents collision and preimage attacks. □

5.1.2. Experimental Analysis

In this section, we use the AVISPA tool to simulate the REISCH’s protocols. This tool is extremely important to examine/check applicability passive and active attacks on security protocols. We tested the exchanging of SNs’ data/information with network entities (CHi, LS and CS) and analyzed the results, as shown following subsections.

AVISPA Summary

AVISPA is a formal verification and validation tool that is used to trace and analyze threats on HWSN’s security protocols. This tool depends on high-level protocol specification language (HLPSL) to achieve its functions. In addition, AVISPA includes backends to trace/detect attacks in many ways, intermediate format (IF) to read HLPSL’s codes and output format (OF) to produce simulation results. In this paper, we depend on the On-the-Fly Model-Checker (OFMC) and the Constraint-Logic-based Attack Searcher (CL-AtSe) backends because our scheme deals with XoR operations. It presents a simple and easy way (push-button) to run HLPSL codes (the readers can get more information about AVISPA details in [72,74]). Additionally, the communication channel in AVISPA is Dolev–Yao (dy) that is used to transfer the sensors’ data/information during HWSN’s simulation. Moreover, AVISPA has been used in recent research because this tool has significant advantages such as threats tracking, implementation robustness, simplicity, analysis of results and statistics [15,72,75,76].

REISCH Scheme with AVISPA

In this subsection, we explain the REISCH scheme in AVISPA. REISCH includes four roles, namely localServer (LS), sensori (SNi), clusterHead (CHi), and centralServer (CS), as well as supporting roles, namely session and environment. Moreover, there are three sections to complete communication properly and securely: transition, composition, and goal specification. The transition section is used in the essential roles to keep a correct communication sequence. The composition section is used in the supporting roles to connect essential roles in specific sessions. The goal specification section includes security goals such as secrecy and authentication. Secrecy means known secrets only for specific entities while authentication depends on witness (freshness claim) and request (validation) processes to perform strong authentication. In addition, our scheme uses parameters such as RCV (receiving process), SND (sending process), _inv (private key), dy (communication channel by Dolev–Yao model), and intruder _knowledge (known information for an intruder). We assume that the intruder uses the public key (ki) and knows public keys for REISCH entities (kSNpu, kCHpu, and kLS). Figure 9 shows the REISCH framework in AVISPA. Figures 10–13 show the REISCH roles in AVISPA.

Figure 9. REISCH’s framework.
role sensori (SNi, CHi, LS:agent, KSNpu, KCHpu, KLSpu:public_key, ECDSA:hash_func, SNpseud, SNsl:Data:text, SND, RCV:channel (dy))
played_by SNi def=
local
State:nat,
SNrni, SigLsEi, SigSnEi, SNparity: text, MAXv:message, SigSN, SigSnT1, SigSnT2, SigSnT3, SigSnT4, SigLsEi: text,
Dif, SNts, SNtst, SN otp, Date, Time, SS, SNp, LSotpi: text
const
sec1, sec2, sec3, auth3: protocol_id
init
State := 0
transition
% SNi receives (LSotpi) from LS
1. State=0 /
   /\ RCV (SNi.LSotpi'.SNrni') =|> State':=1
   /\ SigSnEi':=new () /\ SigLsEi':=ECDSA (SNpseud) \_inv (KSNpu) /\ SigSnEi' := xor (LSotpi', SigLsEi)
   /\ SigSnT2':=ECDSA (Data.SNpseud) \_inv (KLSpu) /\ SNparity' := SigSN' /\ SigSnT1':=SNparity'
   /\ SigSnT3':=xor (SigSnT1', SigSnEi') /\ SNparity':=new () /\ SN otp':=new ()
   /\ SNtst':=xor (Date, xor (Time, xor (SN otp', xor (Dif')))) /\ SN otp' := (SN otp'.SN otp'.SN parity'.SNpseud.SNsl)
   /\ SigSnT4':=xor (SigSnT3'.SigSnT4') /\ SigLsEi':= xor (SigLsEi', SigSnEi')
   /\ secret ({SigSN', SigLsEi', SigSnEi'}, sec1, (SNi, LS)) /\ secret ({SNpseud, SNsl}, sec2, (SNi, LS))
   /\ secret ({MAXv, SNts'}, sec3, (SNi, CHi))
% SNi sends (SNm) to CHi
/\ SND (CHi.SigSnT3'.SigSnT4'.SNrni'.Dif.SNtst'.SS.Data) /
   /\ witness (SNi, CHi, auth3, {SNtst', SNrni', Dif})
end role

Figure 10. SNi role in HLPSL.

role clusterHeadi (CHi, LS, SNi:agent, KCHpu, KLSpu, KSNpu:public_key, ECDSA:hash_func, CHpseud, CHsl:Data:text, MAXv:message, SND, RCV:channel (dy))
played_by CHi def=
local
State:nat,
SNrni, SNrns, SigLsEi, SNparityi, SNs, LSotpi:text, SigCH, SigChT1, SigChT2, SigSnT3s, SigSnT4s, SigSnT3i, SigSnT4i, SigLsEi, Difi, SNtst, CHts1, CHts2, SS, SSI, CHp, CHparity, CHotp, CHrni, Datai, Datas, CHa:text
const
sec3, sec4, sec5, auth4: protocol_id
init
State := 0
transition
% CHi receives (LSotpi) from LS
1. State=0 /
   /\ RCV (CHi.LSotpi'.CHrni') =|> State':=1
   /\ SigSnEi':=new () /\ SigLsEi':=(ECDSA (SNpseud)) \_inv (KSNpu) /\ SigLsEi' := xor (LSotpi', SigLsEi)
   /\ SigSnEi':=(ECDSA (Data.SNpseud)) \_inv (KLSpu) /\ SNparity':=SigSN' /\ SigSnEi':=SNparity'
   /\ SigSnT2':=xor (SigSnEi', SigSnEi') /\ SN parity':=new () /\ SN otp':=new ()
   /\ SNtst':=xor (Date, xor (Time, xor (SN otp', xor (Dif')))) /\ SN otp' := (SN otp'.SN otp'.SN parity'.SNpseud.SNsl)
   /\ SigSnT3':=xor (SigSnT3'.SigSnT4') /\ SigSnT4':= xor (SigSnT3', SigSnEi')
   /\ secret ({SigSN', SigLsEi', SigSnEi'}, sec1, (SNi, LS)) /\ secret ({SNpseud, SNsl}, sec2, (SNi, LS))
   /\ secret ({MAXv, SNts'}, sec3, (SNi, CHi))
% CHi sends (CHm) to LS
/\ SND (LS. (xor (CHt2'.CHotp', CHrni.CHpseud.CHsl), SigChT1'). CHrni.SS. (SigSnT3s.SigSnT4s . SNrns'. Datas')) /
   /\ witness (CHi, LS, auth4, {CHt2', CHotp', CHrni, SNrns'})
end role

Figure 11. CHi role in HLPSL.
As shown in Figure 12, the LS receives the start signal. Then, the LS generates and sends new LSotp for all sensors (SNi and CHi). LSotp includes new SigLSi and pseudonym signature. Figures 10 and 11 both show that SNi and CHi receive LSotp from the LS. Furthermore, SNi and CHi use freshness nonces, timestamp, and signature to support reliable security. For instance, SNi uses SigLSi, SigSnEi, SNts, and SigSN to achieve security processes with the CHi and LS. SNi collects data and uses one ECDSA signature with XoR operations to protect collected data and send it to the CHi. At this stage, the CHi aggregates data and adds security parameters. The CHi sends aggregation data to the LS. After that, the LS uses LSotp, SigLS5, and LSrn to connect with the CS securely. Figure 13 shows the CS with the storage process. The CS receives LSotp and uses ECDSA (CsT2), CsT1, CsT2, and CSrn to secure communication with the LS. Figure 14 shows session and environment roles as well as security goals (secrecy and authentication). REISCH applies seven secrecy and seven authentication goals. For instance, Sec1 represents secrets between SNi and the LS such as SigSN, SigLSi, and SigSnEi. In addition, the authentication goal, such as auth4 proves freshness between the CHi and LS such as CHts2, CHotp, CHrn, and SNrn. Additionally, the environment role includes many attacks (replay, MITM, and impersonating) to test the security level in the REISCH scheme.
Results

The AVISPA tool describes the simulation results. We applied AVISPA with OFMC and CL-AtSe backends. The results of both of OFMC (Figure 15) and CL-AtSe (Figure 16) demonstrate that the REISCH scheme is safe against passive and active attacks (as in the SUMMARY Section). Furthermore, Figures 15 and 16 show analysis details about simulation reports such as number of sessions, goals and statistical numbers. Moreover, the goals of authentication and secrecy in Figure 14 are applied to prevent the penetration of sensors’ data/information in the network. These results prove that REISCH is reliable in combatting known attacks such as replay, MITM, and impersonating.

5.1.3. Security Comparison

In this section, we discuss the superiority of REISCH over existing schemes in terms of security (Table 3 shows a comparison of security features between our scheme and existing schemes). Compared with the scheme in [17] that uses a small key ($F_{216}$) and is extremely vulnerable to attacks, REISCH uses a key with 256 bit that resists attacks (reputable organization recommendations). REISCH also uses BLAKE2bp to get rid of hash attacks (collision and preimage) while the scheme in [18] focused on SHA1 performance without attention to the collision/preimage threats. In addition, all security parameters in REISCH such as $SN_i$’s location are completely hidden, while the scheme in [19] transfers some information explicitly, such as ID (the elliptic curve parameters), in the registration and authentication phases.

5.1.4. Protocol Implementation

The protocol implementation and simulation results are shown in Figures 15 and 16. The simulation results demonstrate that the REISCH scheme is safe against passive and active attacks (as in the SUMMARY Section). Furthermore, Figures 15 and 16 show analysis details about simulation reports such as number of sessions, goals and statistical numbers. Moreover, the goals of authentication and secrecy in Figure 14 are applied to prevent the penetration of sensors’ data/information in the network. These results prove that REISCH is reliable in combatting known attacks such as replay, MITM, and impersonating.

5.2. Performance Evaluation

The performance evaluation of REISCH is shown in Figure 16. The results demonstrate that the REISCH scheme is safe against passive and active attacks (as in the SUMMARY Section). Furthermore, Figures 15 and 16 show analysis details about simulation reports such as number of sessions, goals and statistical numbers. Moreover, the goals of authentication and secrecy in Figure 14 are applied to prevent the penetration of sensors’ data/information in the network. These results prove that REISCH is reliable in combatting known attacks such as replay, MITM, and impersonating.

5.3. Comparative Study

In this section, we discuss the superiority of REISCH over existing schemes in terms of security (Table 3 shows a comparison of security features between our scheme and existing schemes). Compared with the scheme in [17] that uses a small key ($F_{216}$) and is extremely vulnerable to attacks, REISCH uses a key with 256 bit that resists attacks (reputable organization recommendations). REISCH also uses BLAKE2bp to get rid of hash attacks (collision and preimage) while the scheme in [18] focused on SHA1 performance without attention to the collision/preimage threats. In addition, all security parameters in REISCH such as $SN_i$’s location are completely hidden, while the scheme in [19] transfers some information explicitly, such as ID (the elliptic curve parameters), in the registration and authentication phases.

**Figure 13. CS server role in HLPSL.**

```plaintext
role centralServer (CS,LS:agent, KCSpu,KLSpu:public_key, ECDSA:hash_func,SN,RCV:channel (dy))
played_by CS def=
local
State:nat,
SigCS1,SigCS2,LSts4:text, LSpseudo,LSpseudo,CSpseudo,CSpseudo:text,
CsT1,CsT2,CsT3,CsT4,LSts2,CSts1,CSts2,CSts,Cb,LSrn,SS,Datas,LSotp,SigLSi,SigLS5:text
const
sec6,sec7,auth5,auth6,auth7:protocol_id
init
State := 0
transition
% CS receives from LS
1.State=0
/\RCV (CS.LSotp'.SS)=|> State'=1 /\CSts1'=new () /\CSpseudn'=new ()
/\CSts'=new () /\CSts'=new () /\CSts1'=xor (CSts1',xor (CSotp',xor (CSts,Cb)))
/\CSts2'=xor (CSts1',xor (CSotp',xor (CSts,Cb))) /\request (CS,LS,auth5,{CSpseudn,LSpseudo,LSts2})
/\secret ((LSpseudo,CSpseudo),sec6,{CS,LS}) /\secret ((SigLSi,CSotp'),sec7,{CS,LS})
% CS sends to LS
/\SND (LS.(ECDSA (CsT2')_inv (KLSpu).CsT1'.CsT2'.SS.CSrn'))
/\witness (CS,LS,auth6,{CSpseudn,LSpseudo,CSts1'})
% CS receives from LS
2.State=1
/\RCV (CS.SigLS5'.SS'.LSrn'.Datas')=|> State'=2 /\CSts2'=new ()
/\SigCS2'=xor (ECDSA (xor (LSrn4,xor (LSpseudo,xor (LSrn,Cb))))_inv (KCSpu))
/CsT3'=xor (CSts1,xor (CSotp,CSpseudo)) /\CsT4'=xor (LSpseudo,CsT3')
/\SigCS1'=xor ((ECDSA (Datas,CsT4')_inv (KCSpu),SigCS1'))
/\request (CS,LS,auth7,{SigLS5,LSts4,LSrn'})
end role
```
role session (SNi, CHi, LS, CS: agent, KSNpu, KCHpu, KLSpu, KCSpu: public_key, ECDSA: hash_func, MAXv: message, SNpseudi, SNSli, Datai: text)
def =
local SND1, RCV1, SND2, RCV2, SND3, RCV3, SND4, RCV4: channel (dy)
composition
sensori (SNi, CHi, LS, KSNpu, KCHpu, KLSpu, ECDSA, SNpseudi, SNSli, Datai, SND3, RCV3)
\clusterHeadi (CHi, LS, SNi, KCHpu, KLSpu, KSNpu, ECDSA, SNpseudi, SNSli, MAXv, SND2, RCV2)
\localServeri (LS, CHi, SNi, KCHpu, KLSpu, KSNpu, KCSpu, ECDSA, SNpseudi, SNSli, SND1, RCV1)
\centralServeri (CS, LS, KCSpu, KCSpu, ECDSA, SNpseudi, SNSli, SND4, RCV4)
end role

role environment ()
def =
const kSNpu, kCHpu, kLS, kCS, ki: public_key, ecdsa: hash_func, datai, snpseudi, snsli: text,
sec1, sec2, sec3, sec4, sec5, sec6, sec7, auth1, auth2, auth3, auth4, auth5, auth6, auth7: protocol_id
intruder_knowledge = {sn1, ch1, ls1, cs1: agent, maxV: message,
sec1, sec2, sec3, sec4, sec5, sec6, sec7, auth1, auth2, auth3, auth4, auth5, auth6, auth7: protocol_id}
composition
session (sn1, ch1, ls1, cs1, kSNpu, KCHpu, KLSpu, KCSpu, ecdsa, maxV, datai, snpseudi, snsli)
% Check replay attack
%\ session (sn1, ch1, ls1, cs1, kSNpu, KCHpu, KLSpu, KCSpu, ecdsa, maxV, datai, snpseudi, snsli)
% Check MITM attack
%\ session (ch1, sn1, ls1, cs1, kSNpu, KCHpu, KLSpu, KCSpu, ecdsa, maxV, datai, snpseudi, snsli)
% Check impersonate SNi
%\ session (i, ch1, ls1, cs1, kSNpu, KCHpu, KLSpu, KCSpu, ecdsa, maxV, datai, snpseudi, snsli)
% Check impersonate CHi
%\ session (sn1, i, ls1, cs1, kSNpu, KCHpu, KLSpu, KCSpu, ecdsa, maxV, datai, snpseudi, snsli)
% Check impersonate LS
%\ session (sn1, ch1, i, cs1, kSNpu, KCHpu, KLSpu, KCSpu, ecdsa, maxV, datai, snpseudi, snsli)
end role

goal secrecy_of sec1, sec2, sec3, sec4, sec5, sec6, sec7
authentication_on auth1, auth2, auth3, auth4, auth5, auth6, auth7
end goal

environment ()

Figure 14. Supporting roles in HLPSL.

%% OFMC
%% Version of 2006/02/13
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
/home/span/span/testsuite/results/REIESCH_new11.if
GOAL
as_specified
BACKEND
OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 532.61s
visitedNodes: 1899 nodes
depth: 7 plies

Figure 15. Simulation result using OFMC.

%% TYPED_MODEL
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
/home/span/span/testsuite/results/REIESCH_new11.if
GOAL
as_specified
BACKEND
CL-AtSe
COMMENTS
STATISTICS
Analysed : 325 states
Reachable : 325 states
Translation: 0.12 seconds
Computation: 0.01 seconds

Figure 16. Simulation result using CL-AtSe.
Table 3. Comparison of security features.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Anti MITM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Anti replay</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Availability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Anti Sybil</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Anti Wormhole</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Anti fake sink</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Anti repository attack</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Anti eavesdropping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Anti node replication</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Anti collision/preimage</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Pseudonym</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Homomorphic</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Mutual authentication</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
This allows intruders to distinguish a specific \( SN_i \). Additionally, this scheme did not address the problem of hiding the \( SN_i \) location. Although the authors of \([20]\) addressed privacy in their scheme’s architecture to protect the \( SN_i \) parameters. Their scheme did not use the signatures camouflage or \( SN_{OTP} \) that are used in REISCH to support the privacy of data signing. This makes the privacy parameters in their scheme vulnerable to analysis and easy tracking. Furthermore, REISCH outperforms the scheme in \([21]\), which did not use the signature aggregation scheme to support security and hide signatures. The scheme in \([22]\) uses ECDSA to secure heterogeneous network environments. However, their scheme gives medical evaluators privileges to modify the medical parameters in the monitoring environment, \( SN_i \)’s locations and even creates keys that could be the cause of an internal attack. Moreover, some information sent from \( SN_i \) to the server can clearly leak to intruders. Fortunately, REISCH does not suffer from these problems. REISCH adds sufficient randomization to hide security parameters, and patient records are protected even after \( LS \) is penetrated, while the scheme in \([23]\) needs to support randomization and protect user information when a demand–response management unit is penetrated by an intruder. Besides, an intruder can send messages from a forged unit and deceive users after penetrating this module and revealing information. REISCH is robust against information leakage, while the scheme in \([24]\) uses a 160-bit key that is vulnerable to attacks. It explicitly sends patient identities within the encrypted message in the login and authentication phases. If an intruder can break the encryption, he/she can use this information in data disclosure. REISCH uses ECDSA-BLAKE2bp and random pseudonyms to secure data signing. The scheme in \([25]\) is based on SHA1 and HMAC, which are vulnerable to attacks in signing and authenticating collected data. It also does not include a pseudonym mechanism to protect \( SN_i \) parameters from misbehaving.

5.2. Performance Analysis

In this section, the theoretical and experimental performance analysis is presented to examine the computation processes of REISCH in improving the performance of the HWSN lifetime.

5.2.1. Theoretical Analysis

REISCH uses several features that qualify it to be efficient in HWSN performance. First, it relies on the ECDSA algorithm that integrates data collected by small keys compared to public key cryptography algorithms (RSA, DSA and Elgamal). For instance, ECDSA produces 256-bit equivalent keys in security for 3072-bit keys produced by RSA, DSA, and Elgamal. Second, REISCH implicitly uses BLAKE2bp with ECDSA, which is dramatically efficient in the operation of a hash function instead of SHA1. Third, REISCH uses the homomorphic property to combine signatures in \( CHs \) and significantly reduces energy dissipation. Fourth, REISCH relies on the LEACH routing protocol, which is the most efficient energy-saving protocol in WSN. Fifth, REISCH relies on rapid random pseudonyms to protect medical records rather than complex and costly processes of encryption and anonymity. Finally, REISCH uses XML to support efficient patient data management. Therefore, these features allow REISCH to maintain the energy of the \( SNs \) as long as possible.

5.2.2. Experimental Analysis

In this section, we evaluate the performance of REISCH in the execution of security operations in conjunction with the collected and saved data. As noted in previous sections, \( SNs \) require performance-efficient signatures to perform services for as long as possible in patients’ monitoring and care. We provide tests on hash algorithms (SHA and BLAKE) and the signature algorithm (ECDSA). Additionally, we applied these algorithms to HWSN to analyze performance properties such as time, storage, and energy. Table 4 shows all the simulation parameters used in HWSN, while Table 5 shows computational operations in the REISCH scheme. All hash and signature algorithms were implemented by C language while WSN was designed in Octave under Ubuntu 16.04 LTS, processor Intel Core i5 2.6 GHz, OS type 64-bit, Memory 4 GiB, and disk 32.0 GB.
Table 4. REISCH’s simulation parameters.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area of WSN</td>
<td>1000 m × 1000 m</td>
</tr>
<tr>
<td>Number of SNs</td>
<td>200</td>
</tr>
<tr>
<td>Number of CHs</td>
<td>5%</td>
</tr>
<tr>
<td>Number of hops</td>
<td>2</td>
</tr>
<tr>
<td>Node type</td>
<td>Homogeneous</td>
</tr>
<tr>
<td>Node distribution</td>
<td>Random</td>
</tr>
<tr>
<td>LS location</td>
<td>(500, 500)</td>
</tr>
<tr>
<td>Df</td>
<td>Maximum value (707.1068)</td>
</tr>
<tr>
<td>Initial energy</td>
<td>25 J</td>
</tr>
<tr>
<td>Size of packet</td>
<td>200 K, 400 K, 800 K and 1 M</td>
</tr>
<tr>
<td>Control packet size</td>
<td>50 B</td>
</tr>
<tr>
<td>Rounds</td>
<td>1000</td>
</tr>
<tr>
<td>Routing protocol</td>
<td>LEACH</td>
</tr>
<tr>
<td>Propagation energy</td>
<td>10 nJ/bit/m²</td>
</tr>
<tr>
<td>Multi-hop propagation energy</td>
<td>0.0013 pl/bit/m²</td>
</tr>
<tr>
<td>Aggregation energy</td>
<td>5 nJ/bit/signal</td>
</tr>
<tr>
<td>Number of runs</td>
<td>100</td>
</tr>
<tr>
<td>Simulation time</td>
<td>300 s</td>
</tr>
<tr>
<td>Simulator</td>
<td>Octave</td>
</tr>
</tbody>
</table>

Table 5. REISCH’s computational processes.

<table>
<thead>
<tr>
<th>Process Type</th>
<th>Number of Process</th>
<th>Running Time</th>
<th>Storage</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SN</td>
<td>CH</td>
<td>LS</td>
<td></td>
</tr>
<tr>
<td>SHA1 hash</td>
<td>1</td>
<td>1</td>
<td>Many</td>
<td>0.05529</td>
</tr>
<tr>
<td>BLAKE2bp hash</td>
<td>1</td>
<td>1</td>
<td>Many</td>
<td>0.040606</td>
</tr>
<tr>
<td>Keys generation</td>
<td>2</td>
<td>2</td>
<td>Many</td>
<td>0.000859</td>
</tr>
<tr>
<td>Point multiplication</td>
<td>2</td>
<td>2</td>
<td>Many</td>
<td>0.000543</td>
</tr>
<tr>
<td>ECDSA-SHA1 signature</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>0.072838</td>
</tr>
<tr>
<td>ECDSA-SHA1 verification</td>
<td>-</td>
<td>-</td>
<td>Many</td>
<td>0.073103</td>
</tr>
<tr>
<td>ECDSA-BLAKE2bp signature</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>0.050046</td>
</tr>
<tr>
<td>ECDSA-BLAKE2bp verification</td>
<td>-</td>
<td>-</td>
<td>Many</td>
<td>0.052076</td>
</tr>
</tbody>
</table>

The computation of energy in our scheme is based on the Micaz sensor specification. This process uses parameters such as current (0.0567), voltage (2.7), and time to extract both power and energy using \( \text{power} = \text{current} \times \text{voltage} \) and \( \text{energy} = \text{time} \times \text{power} \). We relied on real data provided by the City of Melbourne that is licensed under CC 4.0 [77]. These data were generated by sensors to monitor environmental parameters such as humidity, temperature, and light, as well as to include some information such as timestamp and ID. We divided these data into different sizes (200 K, 400 K, 800 K, and 1 M) and then converted them into an XML context. We used a large data size such as 1 M to test signature processes and security parameters in consuming sensor energy and thus the applicability of WSN. Furthermore, there are no communication channels between patients and SNs. To check performance, we implemented the SHA1-160, SHA2-256, BLAKE2s-256, BLAKE2b-512, BLAKE2sp-256, and BLAKE2bp-512 algorithms with 1 MB data size, as shown in Figure 17. Moreover, Figure 18 shows that ECDSA-BLAKE2bp gives the best execution time of ECDSA-SHA1. In addition, Figures 19, 20, and 21 show execution time (minimum, maximum, and average) for hash functions when using 200 K, 400 K, 800 K, and 1 M data. We also notice that BLAKE2bp has the best performance in terms of execution time in all figures. Additionally, Figures 22, 23, and 24 show the execution time (minimum, maximum, and average) for the ECDSA algorithms when using 200 K, 400 K, 800 K, and 1 M data. Thus, the amendment to the ECDSA algorithm is entirely appropriate for the use of security measures with the longest life of the SNs from the original algorithm.
Figure 17. Comparison of SHA and BLAKE2 with 1 MB data.

Figure 18. Execution time of ECDSA-SHA1 and ECDSA-BLAKE2bp with 1 MB data.

Figure 19. Minimum execution time of hash functions with different data sizes.
Figure 20. Maximum execution time of hash functions with different data sizes.

Figure 21. Average execution time of hash functions with different data sizes.

Figure 22. Minimum execution time of ECDSA algorithms with different data sizes.

Figure 23. Maximum execution time of ECDSA algorithms with different data sizes.
Figure 24. Average execution time of ECDSA algorithms with different data sizes.

We computed message complexity which is the number of messages transmitted between network entities. For each round, SN and CH send one message while CH and LS receive a set of aggregated messages. Thus, the message complexity with modified algorithms for SNs is (156,972), CHs is (8313), and LS is (165,285), while with original algorithms for SNs is (142,541), CHs is (7572), and LS is (150,113). Message overhead is to calculate the message size between network entities. In each round, the message overhead of SN is (1024 + 32) bytes while CH is (15,360 + 32) bytes. Figure 25 demonstrates that REISCH-ECDSA-BLAKE2bp is better than REISCH-ECDSA-SHA1 in terms of alive SNs, namely, HWSN will have a longer life span to collect patient data when it uses REISCH-ECDSA-BLAKE2bp. We noticed that REISCH with the modified algorithm (ECDSA-BLAKE2bp) has more alive SNs by 24% than the original algorithm (ECDSA-SHA1). Furthermore, the first SN dies when using the modified algorithm in round 322, while in the original algorithm is in round 295.

Figure 25. Comparison of alive SNs.

5.2.3. Performance Comparison

In this section, we discuss the superiority of REISCH to existing schemes in terms of performance (Table 6 shows a comparison of the ECDSA’s signature and verification (running time) between our scheme and existing schemes). Due to different environments, security parameters and network parameters such as key length, number of SNs, etc., it is difficult to compare schemes’ performances. However, we made some comparisons to illustrate the superiority of REISCH on the existing schemes in terms of performance. The scheme in [17] focused on accelerating ECDSA’s verification based on computation results for neighboring SNs. These computations consume additional energy. In addition, this scheme is vastly expensive if applied to a cluster scheme because CH needs to accomplish one PM
in each signature for each SN$_i$ and will thus consume energy in the intermediate SNs. REISCH does not need these computations because signatures’ verification is performed in LS. Schemes in [18,19,21] used ECDSA to sign data without using homomorphic property. Consequently, the performance of the SNs would be remarkably low due to signature and verification processes in each round. The scheme in [18] addressed the bits (8 and 32) of data processing in SHA1 but did not address the cost of energy consumption by SHA1. In addition, the scheme in [19] did not support the clustering environment to reduce energy consumption and the computation time to generate and verify the signature which was not clearly indicated. Furthermore, the scheme in [22] used SHA2, which is more secure than SHA1 but performs heavy processes that significantly affect the energy of SNs. It also addresses only computations in transport while REISCH addresses computations in transport and processing using BLAKE2bp and homomorphic property. The schemes in [20,24,25] rely on the use of encryption to protect data without a homomorphic property, since encryption processes extremely consume SNs resources (as mentioned in Section 3.3, Point 3), while REISCH uses signatures and homomorphic property to improve HWSN network performance. Although the scheme in [23] uses encryption and signature of data with homomorphic properties, encryption can particularly affect network performance, especially through a burden on the servers. Moreover, the scheme in [25] has implemented the RSA-2048 bits algorithm, which is significantly expensive in encryption operations. In addition, it uses several parameters such as many keys, 2048-bit key length, and SN$_i$ addresses (master, replica, and gateway) that cause storage problems in the pre-deployment and registration phases (consumption of SNs resources). It uses a random routing of the sensor network without relying on a specific routing protocol such as LEACH. This scheme considers the structure of the data in the SN$_i$ memory and does not pay attention to the structure of the data as they are transferred to the servers. REISCH uses XML to support performance of the LS and CS without having to convert data formats between network devices. In terms of alive SNs, REISCH provides more than 24% while the method in [78] 17.5%, the method in [79] 18.26% (100 nodes), the method in [80] 16% (100–700 nodes), and the method in [81] 7.14% (100 nodes) and 4% (50 nodes). Thus, REISCH provides longer network lifetime than the schemes in [78–81]. Recent research (e.g., [26–30]) has used different ways to improve ECDSA’s procedures. However, REISCH provides better performance in terms of ECDSA’s signature and verification than existing schemes (as shown in Table 6).
Table 6. Comparison of ECDSA’s procedures.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature</td>
<td>0.38</td>
<td>0.941</td>
<td>0.59</td>
<td>0.078</td>
<td>0.3472</td>
<td>0.434</td>
<td>0.084</td>
<td>0.051</td>
<td>0.050</td>
</tr>
<tr>
<td>Verification</td>
<td>0.65</td>
<td>-</td>
<td>-</td>
<td>0.079</td>
<td>-</td>
<td>0.429</td>
<td>0.088</td>
<td>0.105</td>
<td>0.052</td>
</tr>
</tbody>
</table>
6. Conclusions and Future Work

Wireless sensor networks provide unique and important care services when used with EMRs. Unfortunately, these networks suffer from performance and security problems, as mentioned in the previous sections. Therefore, we propose a REISCH scheme to address performance and security problems and cover gaps in existing research. As a result, REISCH uses ECDSA-BLAKE2bp and provides the best performance from using the original ECDSA-SHA1 algorithm. REISCH with the modified algorithm saves more than 24% alive SNs. In addition, the results of the security analysis prove that REISCH is safe against attacks in the threat model. Future directions planned for the development of this scheme are as follows:

1. Our scheme requires security mechanisms to support authentication requests (such as encryption and mutual authentication) and authorization (access control models) and thus allow legitimate users (patients and providers) to access medical records on remote servers (AS and DS).
2. Support for our scheme is by using ECDSA-BLAKE2bp with efficient curves such as the Edward curve and efficient PM methods such as Frobenius to improve the efficiency of patients’ data signing in HWSN.
3. We intend to integrate our scheme into a real HWSN environment to evaluate the efficiency and feasibility of REISCH algorithms to improve the lifetime of SNs in patients’ data collection as long as possible.

Author Contributions: Conceptualization, M.A. and Z.Z.; methodology, M.A.; software, M.A.; formal analysis, Z.Z.; writing—original draft preparation, M.A.; writing—review and editing, M.A., Z.Z., and J.Z.; supervision, Z.Z. and J.Z.; and project administration, M.A.

Funding: This research received no external funding.

Acknowledgments: We would like to acknowledge and thank the efforts of Barbara Harmes who revised our paper as well as the valuable feedback of the reviewers.

Conflicts of Interest: The authors declare that they have no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SN, CH</td>
<td>Sensor, Cluster Head</td>
</tr>
<tr>
<td>LS, CS</td>
<td>Local Server, Central Server</td>
</tr>
<tr>
<td>K_{pu}, K_{pr}</td>
<td>Public and private keys</td>
</tr>
<tr>
<td>OTP</td>
<td>One time passcode</td>
</tr>
<tr>
<td>Pseud</td>
<td>Pseudonym generated by entities (SN, CH, LS, CS)</td>
</tr>
<tr>
<td>Parity</td>
<td>The value specifies the signature of even/odd</td>
</tr>
<tr>
<td>P</td>
<td>Entity parameters</td>
</tr>
<tr>
<td>RN</td>
<td>The random number generated by entities</td>
</tr>
<tr>
<td>TS</td>
<td>Timestamp generated by entities</td>
</tr>
<tr>
<td>Sig_{SN}, Sig_{CH}</td>
<td>Signatures generated by SN, CH</td>
</tr>
<tr>
<td>Sig_{LS}, Sig_{CS}</td>
<td>Signatures generated by LS, CS</td>
</tr>
<tr>
<td>Sig_{SnEi}, Sig_{LsEi}</td>
<td>Random ephemeral value the same length as the signature generated by SN, LS</td>
</tr>
<tr>
<td>S_{NC}D</td>
<td>Distance between SN and CH</td>
</tr>
<tr>
<td>S_{NL}D</td>
<td>Distance between SN and LS</td>
</tr>
<tr>
<td>Diff</td>
<td>Value proves SN in the HWSN’s area</td>
</tr>
<tr>
<td>SL</td>
<td>Sensor location</td>
</tr>
<tr>
<td>m</td>
<td>Message sent by entity</td>
</tr>
<tr>
<td>A</td>
<td>Aggregation function</td>
</tr>
<tr>
<td>h(.)</td>
<td>One-way hash function</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
References


© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).