symmetry

MDPI

Article

Implication of Lightweight and Robust Hash Function to Support Key Exchange in Health Sensor Networks

Mishall Al-Zubaidie

Department of Computer Sciences, Education College for Pure Sciences, University of Thi-Qar, Nasiriyah 64001, Iraq; mishall_zubaidie@utq.edu.iq; Tel.: +964-61469869029

Abstract: Internet of Things (IoT) applications are critical for the fast delivery of health information/data in different environments. The wireless sensor network (WSN) can be used within IoT applications to collect this information in the electronic-health sector. However, the essential drawback of WSN and health applications is ensuring that patient and healthcare provider data/information is protected. In addition, exchanging keys and joining the network is the first/most important line of defense to protect health information. Amid all this, the previous search has introduced many key exchange protocols but still suffers from security and performance issues for WSNs and user devices. In this research, we propose a new protocol for exchanging keys and joining the network using security algorithms that are Elliptic-curve Diffie–Hellman (ECDH) and QUARK hash (qh). We focused on applying lightweight and high-security techniques to reduce the burden on WSN resources, by adopting a solid methodological approach to support security first and performance second. The security analysis is simulated with the Scyther tool, and the results indicate that our protocol is able to block key exchange attacks known in the existing research. Furthermore, we carried out a comparison with the results of the recent search in terms of performance, our protocol provides better performance results than the results of the existing search.

Keywords: ECDH-QUARK; e-health; IoT; key exchange attacks; QUARK; Scyther; sensors; WSN security

Citation: Al-Zubaidie, M. Implication of Lightweight and Robust Hash Function to Support Key Exchange in Health Sensor Networks. *Symmetry* **2023**, *15*, 152. https://doi.org/10.3390/sym15010152

Academic Editors: Takeshi Koshiba, Milan Milosavljević, Yuan Ping and Yuri Borissov

Received: 26 November 2022
Revised: 22 December 2022
Accepted: 30 December 2022
Published: 4 January 2023

1. Introduction

As part of the Internet of Things (IoT), small, dynamic, smart, fast, and wireless nodes communicate with online platforms that provide various applications, including electronic-health (e-health), electronic-banking (e-banking), smart cities, industry 4.0, and smart homes. Recently, there has been a big surge in the use of IoT applications in the health sector (home, clinic, health center, hospital or even health institutions that include the whole city or country). One feature of the IoT which remains to be studied is the technology of wireless sensor networks (WSNs). IoT has increased the level of investigation into WSNs even more than before. The recent advancements in the field of telecommunications and various services have made it possible to develop low-cost, low-power, compact, high-performance sensors that are small enough to fit a variety of uses. Basically, WSNs consist of four basic components: sensors (Ss), cluster heads nodes (CHs), base stations (BSs) nodes, and interfaces (Is) [1,2]. The widespread use of WSNs that have been composed of resource-limited sensors in connection with BSs in open-space environments and health sectors focused on the necessity of a fast yet secure connection approach between users, Ss, CHs, and BSs. Security is one of the essential controversies in WSNs, as weak precautions may disrupt the whole WSN and may have a serious impact on data storage/transmission and detection of secret keys. Additionally, security procedures for health records and databases can significantly affect WSN performance. In the interests of balance, WSNs require safe key exchange at a high rate while ensuring data integrity and lightweight processes. In light of WSN limitations, elliptic-curve cryptography (ECC) sounds like the most attractive

option since it uses fewer resources than most other alternatives. For instance, standard asymmetric algorithms are impractical. Most IoT devices in the health sector currently handle the use of assorted types of symmetric algorithms such as shared and predistribution keys algorithms. Cryptographic algorithms are compelling in maintaining the protection of WSN data to curb the impact of such malicious threats in terms of key exchange [3–5].

WSN nodes are increasingly using asymmetric key cryptography to exchange and authenticate keys while dealing with the Internet. Nonetheless, because of the resource requirements of these sensors, it is frequently difficult to execute public key and hash algorithms as these algorithms are computationally costly. Previously, some key exchange administration implementations on WSN nodes depend on preshared symmetric keys. Before the sensors are deployed, keys are pre-introduced on these implementations. This manner does not provide adaptability to organizations that include a huge number of sensors (tens of hundreds of thousands or even millions). For instance, pre-installing 10,000 advanced encryption standard (AES) keys with 128 bits length on a sensor needs storage memory of 160 KB, and furthermore presents with critical look-up latency. Furthermore, WSNs are decentralized and especially ad hoc, making customary key exchange algorithms difficult to implement. An ad hoc structure has decentralized sensor placement, uncontained fixed positions, and furthermore does not provide information about adjacent sensors [1,6]. Numerous applications in WSNs should be secured in assorted aspects, including pre-deployment, registration, distribution, routing, session key agreements, management, and key update. Cryptography approaches undertake a fundamental part in the security construction of sensor networks. Hence, WSNs cannot adapt to conventional cryptographic key exchange algorithms. Subsequently, there are three crucial difficulties related to WSN's protection solutions. First, the memory limit should be decreased, i.e., the length and size of the key as well as the size of the encrypted and explicit data. Second, key information over-burden ought to be limited since each data item moved consumes energy, decreasing the sensors' lifetime. Because of the nature of the radio frequency for wireless, anybody can eavesdrop or take part in the connection [7]. Thirdly, the randomization of the key should be appropriate to secure the network connection [8].

Indeed, even keys agreement protocols, which are a fundamental requirement of specific security schemes, are as yet the protocols that are not fully secure and not addressed accurately. Because of the resource constraints in terms of computing cost, battery power, and memory of low cost sensors and other limitations introduced in previous studies, WSN is a predestined target for security threats and common security protocols are hard to execute [9]. Protecting keys is a significant point in WSNs since messages are communicated/encrypted through channels using these keys wirelessly where adversaries ($\mathcal{A}$s) might gain penetration to collected data if the keys are hacked. Some current solutions do not take into account WSN limitations, such as constrained resources when addressing critical resource-constrained nodes [10,11]. Guaranteeing the security of WSN collected data and transmitted remains a real test for developers and researchers [10,12]. Certainly, the absence of a trustworthy key management protocol does not support applying symmetric cryptographic procedures. The low calculating and data storage capabilities of WSN components complicate asymmetric cryptography execution. Furthermore, lightweight computational sensors, which were deployed in the WSN framework, make security difficulties for many existing health applications. The heuristic function obtained is adjusted to decrease the divergence issue [10]. Because of such conditions, it is important to determine the requirement for comprehensive and intact protection to guarantee WSNs are settled and secured. Elliptic Curve Diffie Hellman (ECDH) is one of the most popular key exchange protocols. In Aikins-Bekoe and Hayfron-Acquah [13] and Al Zubaidie et al. [14], the ECDH algorithm for key exchange is analyzed and studied.

Henceforward, the fundamental issue with WSNs is to provide a more suitable and protected network, for which the state-of-the-art needs to pay attention to the method of setting up simple, reliable, robust, and secure networks. Due to the unusual/unpredictable nature of wireless channels and network security, conventional key exchange mechanisms

are unable to secure WSNs from threats. This comes up as a result of the many constraints of resources such as limited energy, running time cost, and lower memory [13,15]. Cryptography includes the procedures for protecting connections in the existence of third parties, which are categorized into a hash function (one-way cryptography), session key, secret key and public key cryptography based on the keys employed [13]. Asymmetric cryptography relies on mathematical methods that can be calculated simply but are comparably strenuous to calculate its opposite. Among the asymmetric algorithms used at present for key generation/exchange, ECC is found to be the latest method to provide an elevated level of protection [13].

The problem of key exchange is crucial in WSNs. The unattended/decentralized nature of some WSNs makes them susceptible to sensor compromise attacks. The network and resource limitations alongside exceptionally various threats impose many severe requirements for the security manner in WSNs. However, the reliable construction to perform the best security is usually not performed during WSN which often relies on the engaged keys and their management operation. In addition, longer cryptographic keys need more space, more bandwidth and extra processor power. Furthermore, these keys take time for key generation, registering and distributing, which is mostly associated with most modern used cryptographic keys such as the RSA and ECDH [13]. Wireless sensors and their data must be secured using a lightweight key exchange method to enable real-time data access [16]. Many cryptography protocols have been designed using symmetric and asymmetric algorithms. According to the security perspective, an asymmetric algorithm is more attractive due to its improved security level. Digital Signature Algorithm (DSA), Elgamal cryptosystem (ECS), ECC, Rivest–Shamir–Adleman (RSA), and other methods have been designed for realizing asymmetric cryptosystems. These algorithms are basically used to guarantee a reliable and secure connection. A smaller key size and the same level of security are provided by ECC [17]. In this case, ECC-ECDH is used and both parties will exchange keys based on the points that have been agreed upon [17].

Data connections are secure with a public-key cryptosystem since a secret/private key is not shared between the client and server. A symmetric key cryptosystem is quicker than an asymmetric one because of producing two keys. A lot of academics/developers have designed many manners to increase the quickness and performance of asymmetric public keys. ECDH has been designed with a less key size that supports high security. ECDH is the best substitute for an RSA cryptosystem. RSA uses a 1024-bit key size while ECDH uses only 160-bit for equivalent security. Among these different cryptographies (RSA, Elgamal, ECDH, etc.), ECDH is effective in view of its higher security and more modest key length [17].

1.1. Significance of Proposed

The exchange/collection of health data via IoT applications using WSN is a significant/critical issue, as any manipulation of this data will generate negative consequences for the health of patients and the reputation of the health institution. Any institution that does not apply strict security measures in protecting patient data will be rejected by individuals and society. It is well known that the exchange of keys in WSN is carried out through insecure channels that are vulnerable to hacking at any time. In addition, the first line of defense in cryptographic systems is to secure the exchange and management of keys for user devices and sensors. If the adversary can hack and gain access to the keys, this means that this adversary may be able to access the network and modify the health data, which negatively affects the health of patients, especially in emergency, critical situations, and online surgeries. Therefore, the security of key exchange is an important and crucial issue, especially in health systems, as the motive of this research is to build a secure and robust key exchange protocol and to overcome the shortcomings of previous protocols in terms of key exchange.

1.2. Main Contributions

We propose a new protocol for addressing security drawbacks in key exchange schemes that is based on the following major contributions:

- Support for parameter security and public key exchange is through the establishment of strict rules in the protocol methodology. Completely hiding public key exchange is to prevent any information or parameters from being leaked to adversaries.
- Integration of QUARK with ECDH is to provide robust keys for both user devices and WSN devices in the health sector.
- Providing a security analysis that tests the proposed protocol using the Scyther tool, in addition to comparing performance and security with the results of recent research.

1.3. Paper Structure

Following is a description of how the research is structured: Section 1 provides a comprehensive introduction. In Section 2, we provide a critique of recent research on key exchange security. Basic concepts are introduced in Section 3. Section 4 provides details of the proposed protocol. The proposal results and findings discussion are described in Section 5. Limitations of the proposed protocol are presented in Section 6. In Section 7, we briefly present the conclusions of this study and future work.

2. Keys Exchange Studies of Related Work

In this section, we will cover a set of recent protocols that suggested by existing research around key exchange in IoT-WSN applications.

Wu et al. [18] provided a key agreement scheme using fog nodes secure authentication for wearable health sensor application. The cloud server and the patients of smart wearable medical devices (SWMDs) are connected by fog nodes. Through the connection protocol, these patients' sensors send data pertaining to their bodies to the fog node. However, our protocol does not deal with the wireless body sensor network (WBSN). Furthermore, the authors did not locate sensors on the patient's body to assess performance in terms of transmitted packets. In addition, the key exchange and authentication phase of their protocol performs heavy operations, for example, a wearable medical device performs seven standard hashes. A data integrity scheme based on signatures and hashes was proposed to alleviate security issues associated with wireless sensor networks for health applications [4]. The researchers focused on signatures for the integrity of patient data and did not focus on important security issues in key exchange. In addition, Kim et al. [19] revisited the three-party password-authenticated key exchange. Their protocol provides patients' anonymity to prevent privileged insider and impersonation attacks. They presented an authentication key agreement protocol in order to protect patient data and support efficient performance. However, their protocol does not provide protection for most key exchange threats such as key recovery and known session key. Alzahrani [20] designed a telecare medical information system protocol using symmetric key exchange procedures. Their protocol applied in electronic-health WSNs with three phases, which are Initialization, User Registration, and Mutual Authentication. They claimed that their protocol was validated with the ProVerif tool. The authors indicated that their protocol addresses preventing attacks such as replay. However, their protocol does not address the issue of protecting keys stored on users' devices, sensors, and servers. Butpheng et al. [21] investigated the use of intelligent security and privacy techniques in health applications and its evolution over time (from 2017 to 2020), with a particular focus on IoT sensor devices and cloud computing. They pointed out the issues of security, privacy and key exchange and the importance of their application in health-WSN projects. In their review, they stated that security solutions in health applications still are insufficient to protect health data and services.

Abdulhameed et al. [7] proposed a hybrid algorithm. The authors claimed that their protocol provides confidentiality, authentication and integrity. They relied on the algorithms Blowfisth, Rivest Cipher 4 (RC4) for encryption, ECDH for authentication, and Message Digest 5 (MD5) for integrity. However, the algorithms Blowfish, RC4, and

MD5 are vulnerable to attacks. Their proposed also does not have an accurate analysis of the security keys of ECDH. ECDH-AES protocol [10] is proposed for key exchange in WSN. The authors relied on classic ECDH which may not provide sufficient security to protect data collected/transmitted by sensors. Furthermore, their protocol does not provide perfect forward secrecy. Aikins-Bekoe and Hayfron-Acquah [13] implemented an analytical study of key exchange protocols (ECDH and RSA) using the PyCryptodome package. The obtained analysis results indicate that ECDH provides higher security with smaller keys than RSA which makes it applicable in WSN. Kardi and Zagrouba [22] proposed a dynamic hybrid protocol based on ECDH and RSA. This protocol takes advantage of the advantages of ECDH and uses them to improve RSA. However, their protocol still uses long RSA keys which are a significant cause of sensors running out of storage space and computational complexity. Another paper Nagesh and S Naresh [17] improved the ECDH protocol to Modified-ECDH (MOD-ECDH) to reduce the key size and improve performance. However, there is no security analysis of their proposed protocol and nor did the authors provide an in-depth analysis of key sizes and lengths.

Phimphinith et al. [23] proposed an ECDH-key agreement based on cureve25519 in IoT applications for communications between ESP866. They indicated that their scheme resists man in the middle (MITM) attacks, replay and modification. Nonetheless, their protocol requires additional hardware/software to support its use ESP8266. Moreover, their protocol relies on a key of length 192 which is not very safe against attacks. In addition, there are some clearly published parameters such as ID that can lead to hacking and exposing the keys. Another paper Hasan and Farhan [24] proposed a scheme for key distribution protection using logistic map Diffie Hellman (LMDH), SubMAC to prevent replay and MITH attacks, and RSA to improve key cryptographic security in WSN-ZigBee. However, their scheme does not provide enough randomness for key generation. Additionally, recent studies have indicated that RSA is not very suitable for application in a WSN. In addition, the authors did not discuss key-bargaining attacks. Lin [25] introduced a key synchronization protocol to support hierarchy in WSN-IoT by proposing a hierarchy-based cluster elliptic curve key agreement named (HCECKA). They claimed that their protocol provides efficiency and speed in key agreement and data transfer. Their protocol relies on an elliptic curve key agreement to accomplish group key agreements. They indicated that RSA and Diffie Hellman are inappropriate for group key agreements in WSN. In addition, they pointed out that ECDH provides keys that are safe and small compared to previous algorithms. However, the authors do not describe a threat model for their proposed protocol, as it is not possible to determine how their protocol is able to fend off key exchange attacks. Rangwani et al. [26] proposed a lightweight authentication and session key negotiation scheme that uses ECDH and SHA256 for agricultural WSN. They claimed that their scheme is able to fend off attacks such as the impersonation, key compromise, known session-specific temporary information, sensor node capture, and session key leakage attacks. However, they did not specify the key length used for ECDH. In addition, using SHA256 will perform heavy computations on the sensors, which can lead to a rapid loss of sensor power. Recently, another paper Thabit et al. [27] proposed a lightweight cryptographic algorithm to secure shared keys and data. Their algorithm consists of a 128-bit block cipher and a 128-bit key. They pointed out that their symmetric algorithm requires efficient execution time for key generation and data encryption in cloud computing applications but that their algorithm is not suitable for large WSNs, as symmetric algorithms are known to suffer from scalability problem.

Gope and Sikdar [28] proposed a key exchange protocol based on the principle of reconfigurable physical unclonable functions. They indicated that their protocol provides efficient security in smart grids and that it provides low communication and computation costs. They claimed that their protocol counteracts several threats such as session-key and impersonation. However, their protocol does not provide protection against critical attacks for key exchange protocols such as stolen verify, key recovery and session key computation. Mehra et al. [29] proposed a lightweight routing protocol based on attested

key exchange in wireless sensor networks. The authors pointed out that performance efficiency greatly affects security and makes it more challenging, especially on source-restricted devices. Their protocol relies on codeword, hash function and one-time password authentication to fend off WSN key exchange attacks. However, their protocol does not provide randomness when calculating the hash function in the registration phase of the user's device and sensor, which makes their protocol vulnerable to session key computation threat. Another paper Wu et al. [30] proposed the three factors (password, biometric and smart card) authentication key exchange protocol to prevent impersonation threats and known session specific temporary information. They indicated that their protocol provides perfect forward secrecy and anonymity for WSN health applications. However, their protocol does not provide a security condition for session key and random parameters detection. Wu et al. [31] proposed an authentication key exchange (AKE) that relies on relay sensors to improve the performance and security of their protocol. The authors pointed out that their protocol uses elliptic curve, bitwise operation, and hash function techniques to ensure network security. However, their protocol suffers from the threat of sensor impersonation and does not provide the user/sensor session key verification property.

3. Preliminary Techniques for Key Exchange

In this section, we will cover some basic concepts/algorithms in key exchange protocols.

3.1. Elliptic Curve Diffie-Hellman (ECDH)

Standard Diffie-Hellman (DH) was proposed in 1976 by Whitfield Diffie and Martin Hellman. An elliptic curve with DH was used to solve the problem of large key sizes and lengths. A secret key is generated using an elliptic curve Diffie-Hellman analog. With ECDH, Diffie-Hellman key agreement is enhanced using elliptic curves instead of conventional Diffie-Hellman analog. Through the Diffie-Hellman method, two parties create secret keys that are shared between them, keeping the secrets from third parties. An elliptic curve Diffie-Hellman algorithm is a key agreement protocol that permits network entities, Alice (A) and Bob (B), to create shared secret keys that are used for public key protocols. Public information is exchanged between the two parties. A shared secret key could be generated by two parties using public and private information. It is impossible for third parties without any knowledge of both parties' private information to calculate the shared secret key from publicly accessible parameters [13,22]. The ECDH protocol presents parties with public and private keys for encrypting data they transact among one another in an insecure environment. As a result, these derived keys can be used as keys to secure subsequent data transactions between the channel's parties. For sending and receiving keys, ECDH algorithms follow the following steps [17] (Figure 1 describes these steps).

- First, all parties should agree publicly on elliptic curve parameters (p: prime, a and b: define a curve, G: curve's generator, n: large integers, h: cofactor) and generate an elliptic curve together.
  - Elliptic curve equation values and p,
  - Elliptic group from elliptic curve equation values
  - G is calculated from that equation of elliptic group
- Next, each party should generate a pair of keys: a private key (PrK), a random point on the curve, and a public key (PuK) derived from $PrK.G$.
  - PrK is a random integer, n, chosen from [1, $p-1$]
  - PuK is the product of base point and private key (i.e., $PuK = PrK.G$)
- Suppose the keys of sender (Alice) be (PrK_A, PuK_A) and keys of the receiver (Bob) be (PrK_B, PuK_B). In order for a message to be encrypted/decrypted, the sender must share their or her PuK with others during communication.
- In the ECDH protocol, a message or data denotes a point in the elliptic curve (x, y).
- The receiver can calculate the point (x, y) and decrypt the message using $PuK_B.PrK_A$ or $PuK_A.PrK_B$.

- Asymmetrical property of ECDH encrypted messages is as follows: $PrK_A.PuK_B = PrK_A.PrK_B.G = PrK_B.PrK_A.G = PrK_B.G$ [3,17].

With ECDH, two parties can share secrets over an insecure channel by having a pair of public and private keys. Assuming A has a private key (PrK_A) and a public key (PuK_A) pair based on an elliptic-curve of characteristic p, and B has a private key (PrK_B) and public key (PuK_B) pair. The two parties are then able to come to an agreement on a secret key ($PrK_A.PrK_B.G$) or some function of it, since: $PrK_A.PuK_B = PrK_B.PuK_A = PrK_A.PrK_B.G$ [3].

Using the public key, $PuK = PrK.G$, each party creates a secret key by multiplying PuK by the secret integer (i.e., $Prk.PuK$) [13]. ECDH keys can be static (ECDHS) or ephemeral (ECDHE). Static keys can be vulnerable to a variety of attacks as they allow an adversary to perform analytic attacks. ECDH security is determined by the difficulty of solving the discrete logarithm problem on elliptic curves. Solving the ECDLP is more difficult than solving many problems such as the discrete logarithm problem (DLP) and the integer factorization problem (IFP). ECDH is more lightweight as compared to classic DH because ECDH depends on the elliptic curve. Many recent studies have implemented ECDH with source-restricted devices such as WSN because this protocol provides relatively small keys compared to existing public-key protocols. Figure 2 shows a hypothetical use of the ECDH protocol within hospital WSN. After ECDH keys are safely generated and distributed to network entities, encryption (symmetric or asymmetric) and signature algorithms can be applied.

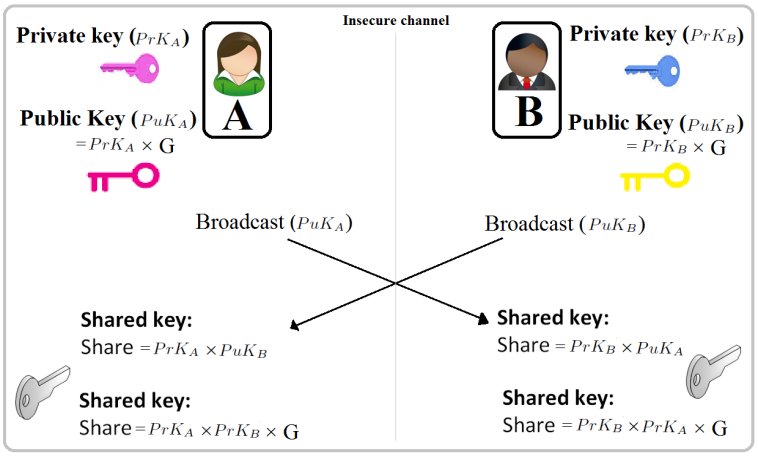

Figure 1. ECDH protocol.

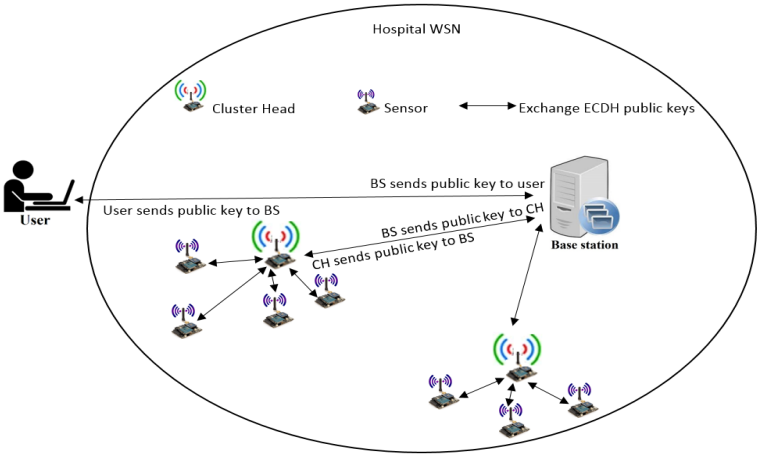

Figure 2. Using ECDH protocol in hospital WSN.

3.2. QUARK Hash Function

QUARK was proposed by Aumasson et al. in 2010. The stream ciphers Grain and KATAN are used in QUARK's permutation function (Pb), depending on sponge construction. In QUARK, the permutation function is implemented using feedback shift registers (FSRs). QUARK uses three FSRs: two of which are non-linear, and one of which is linear. Three versions are included in this hash function: the u-QUARK (136 bits), the d-QUARK (176 bits), and the s-QUARK (256 bits). In this algorithm, shift registers replace the s-box in SPONGENT and PHOTON hashes, and it is inspired by the Grain (stream) and KATAN (block) ciphers. It also uses sponge construction to improve operation and core permutation. Researchers pointed out that there are devices constrained source (such as sensors, radio frequency identification (RFID) . . . etc.) that uses little power at the same time, these devices need protection from attacks. Results discussed by researchers that this algorithm implements short light calculations, leading to reduced time (reduction of consumption energy) to generate keys in addition to the QUARK algorithm's ability to repel the attacks [32].

According to the inputs of all three FSRs, QUARK used the "h" function. To update the first non-linear FSR (NFSR), the "h" function is used in conjunction with another "f" function and the second NFSR values. In addition, the second NFSR updates are performed based on "h" function and "g" function. A linear FSR (LFSR) is updated based on an independent "p" function. Inputs to the first NFSR are derived from the "f" function, and inputs to the second NFSR are derived from the "g" function. A digest with the same length as the input state is generated after four rounds of updates. The highest area requirement for this hash function is 4640 GE, in contrast, 1379 GE is the lowest requirement. Figure 3 shows the permutation treatment of a function.

There are three properties of QUARK: length, capacity, and rate. This can be referred to as QUARK-n/c/r. The internal state has a size of b bits, where b equals the sum of capacity and rate. This hash function provides three different versions representing three different levels of security: u-QUARK (64-bit security), d-QUARK (80-bit security), and s-QUARK (112-bit security). An initialization phase begins with padding the message and dividing it into r-bit blocks, as the last bits in the internal state of the absorbing phase are permutated, it is XORed with the last bit in the internal state of the absorbing phase. After that comes the squeezing phase, which involves making r-bits until each n-bit hash is returned [33].

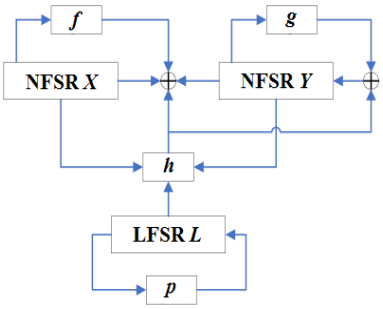

Figure 3. QUARK permutation function.

It incorporates three FSRs that include two NFSRs that contain f, g (such as the Grain algorithm), and h; and LFSR that contains p. The Pb function goes through three phases: initialization, state updating, and result computation. Each version has different non-linear capabilities f, g, and h [33]. The last version is c-QUARK (160-piece security) that developed in 2012. It is designed for the situation of key exchange and authentication protocols with related information [32]. Our research focuses on the version of s-QUARK (QUARK-256) and c-QUARK (QUARK-384). Abed et al. [33] pointed out that QUARK is better than the popular lightweight hashes (SPONGENT and PHOTON) in terms of power and energy and therefore may be better to implement in WSN. Figure 4 shows the advantage of

QUARK over SPONGENT and PHOTON in terms of power and energy consumption. In addition, QUARK hash function is designed to resist slide resynchronization, side channel, simple truncated differential, conditional differential, cube threats and cube testers, and is supported by the principles of resistance for collision, pre-image and second pre-image [34].

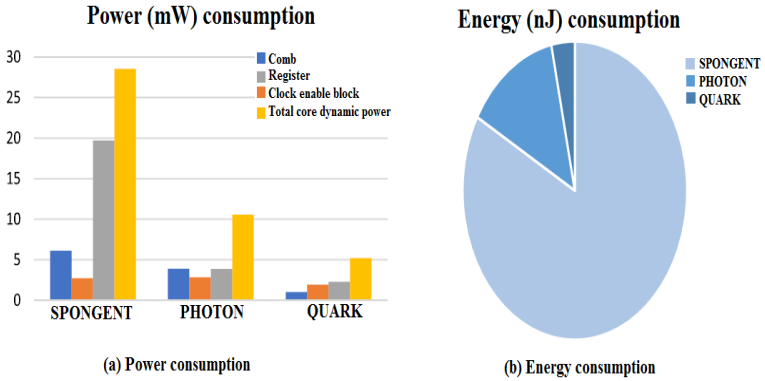

Figure 4. Power and energy consumption in QUARK, SPONGENT and PHOTON.

4. The Proposed ECDH Protocol

This section will describe the proposed protocol for secure key exchange that ensures that subsequent procedures such as authentication, authorization, access control, data availability . . . , etc. are performed reliably. The proposed protocol contains five phases which are pre-deployment, registration, key update, session key distribution and session key protection, which will be described in the following subsections.

4.1. Trust Model and Threat Model

In terms of the trust model, we considered that our protocol has confidence aspects, including that the sensors have a specific distance to transmit the signal, such, for instance, that the sensors' signal does not extend beyond the boundaries of the area of specific fields such as a hospital or health center. In addition, all sensors are placed in specific locations, and any sensor that sends data from different locations will be rejected. If the sensor locations are changed legitimately, this should be specified in the BS. Furthermore, requests to send and receive between BS and remote servers in IoT applications are secure. The security of remote servers is behind this study.

In terms of the threat model, we defined a range of attacks/threats tested under the proposed key exchange protocol. We considered attacks that are known session key, stolen verify, condition for the session key, forward secrecy, exposing the random parameters, key recovery, known session-specific temporary information, user-impersonation, sensor node impersonation, postdeployment, session key verification property, session key computation, offline password guessing and jamming to be within our field of study. These attacks target key exchange protocols to hack either by changing them or using them to access the network legitimately. Adversaries ($\mathcal{A}$s) try to penetrate session keys, or even private keys. These threats are considered serious due to the frequency explained in much research. In addition, adversaries use analytical methods, mathematical algorithms and deception methods to penetrate these keys and destroy key exchange protocols.

4.2. Network Model

In this section, we will explain the network model of our proposed protocol. Our proposed protocol goes through several phases until the key exchange is completed securely. It is designed to operate in health environments (hospitals, health centers . . . , etc.), our model is made up of sensors, BS and users' devices as well as remote servers. It achieves five phases before the legitimate keys are approved. First, the pre-deployment phase takes place where both sensors and users' devices are configured with security parameters, then the registration phase is completed to ensure that the devices use legitimate security parameters

with validation of these parameters and proof of registration. Next, the protocol uses the key update phase to ensure freshness and randomness. Then our protocol covertly implements public key distribution. Furthermore, session keys are generated to be protected by the session key protection phase. Figure 5 shows the network model of the proposed key exchange protocol.

As shown in Figure 5, the sensors are only directly connected to the BS. Users' devices are connected with BS and remote servers, for example, a user (patient, caregiver) can access the old medical records of a particular patient after passing the authentication and authorization procedures, whereas BS is associated with sensors and users as well as remote servers to store medical reports and data collected by WSN (in the case of sending from a BS to medical database servers) or to receive medical instructions from a doctor or health caregiver (in the case of receiving from medical servers to BS).

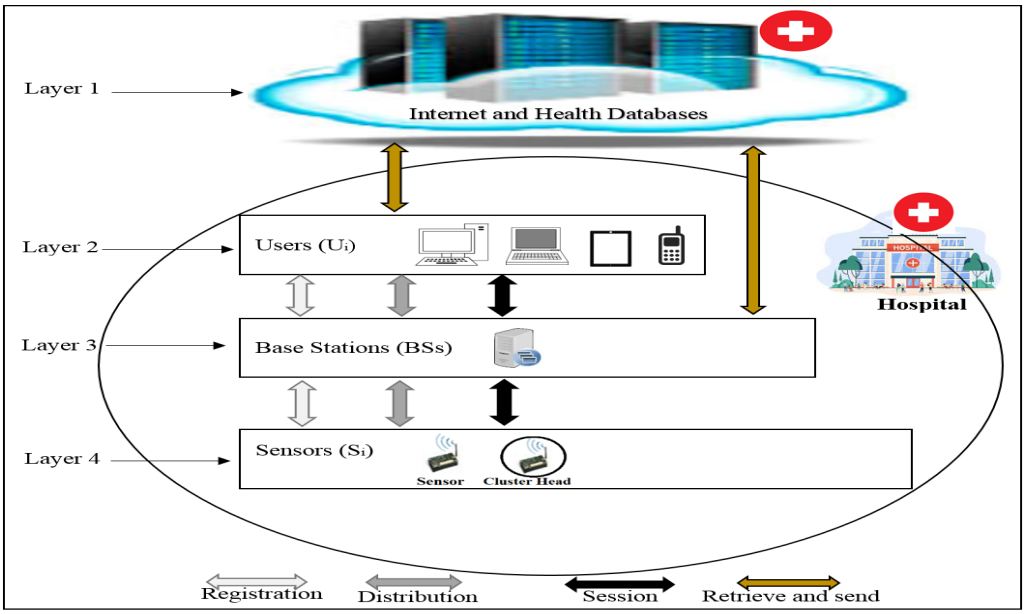

Figure 5. Proposed network model.

4.3. The Proposed Phases

A detailed description of the phases of the proposed protocol will be presented in this section.

4.3.1. Pre-Deployment Phase

In this phase, all network model devices are loaded with default security parameters to allow sensor devices and users' devices to complete the registration process in BS. All sensors have equipped sensor identifier (SID), location (Loc), default public (PuK_S) and private (PrK_S) keys of 224-bit length, and a secret key domain (which consists of a file containing a 60 message digest) of 256 s-qh length. Users' devices (UDs) are equipped with the user identifier (UID), password (PW), biometric (BM) and default public (PuK_U) and private (PrK_U) keys of 384-bit length. After the sensors are deployed and the parameters are uploaded to the users' devices, the network is ready to start the registration phase.

At this point, the security and network parameters will be uploaded to the sensors and users' devices before communicating and exchanging public or session keys. For example, each node (user device) will have UID, elliptic curve, points, PW, BM, default public and private keys, and a file (pool of message digest (MD)) of 708 MD. Each node (sensor) will have a SID, location, elliptic curve, points, a default public and private key, and a file of 60 MD. All devices require these parameters to securely complete the registration and key exchange phase. Because sensors have limited resources compared to users' devices, we

use s-qh-256 with sensors while we use c-qh-384 with users' devices. Furthermore, the proposed protocol uses ECDH-224 with sensors while ECDH-384 with users' devices.

4.3.2. Registration Phase

After loading the initial parameters of the user devices (UDs) and sensors (Ss), the sensor devices are placed in fixed locations, whether in the hospital or health center. All devices, whether users' devices or sensors, must complete the registration phase to become legitimate in the network (the registration phase is shown in Figure 6), thus allowing these devices to perform the remaining key exchange phases. To complete the registration phase on the UD side, it performs the following steps:

1. UD retrieves default parameters received from BS.
2. The user's BM is computed by one of the input devices either a finger scanner or a camera for IRIS capture. The UD calculates a timestamp (TS) to use in the following steps.
3. The UD executes a QUARK hash (qh or $c-qh$) on ($UID||PW||BM||TS$) to obtain a 384-bit MD.
4. The user application retrieves the MD from the file stored by default in the user application. This file contains 708 MD (which represents the largest value of time multiplied by 12×59). One MD is retrieved depending on the time. For example, if the time is 3:20, the hours are multiplied by minutes to obtain the required input (60). The user application will extract the MD at entry 60 from the $c-qh_s$ file and perform the operation $qh_1 \oplus qh_{60}$ to obtain qh_2 and to support randomness in the registration phase.
5. Then, the user application generates a user registration request (URR) by calculating $P_uK \oplus qh_2 \oplus UN||TS \oplus qh$ (where UN is a user name and $c-qh$ is QUARK hash with 384-bit length to specify the user in registration phase), and then sends the URR to BS.
6. When BS receives the URR from the user's device, initially, it truncates $TS \oplus qh$ to distinguish the user.
7. BS specifies UN, P_uK_U and TS based on qh.
8. Furthermore, it retrieves qh_i based on TS (the number of hours is multiplied by the number of minutes to specify the qh_i entry in the stored file).
9. Then BS extracts qh_2 by calculating $P_uK_U \oplus UN \oplus URR$.
10. Next, BS extracts qh_1 based on $qh_2 \oplus qh_i$, and then calculates qh'_1 through $qh'_1(UID||PW||BM||TS)$. At this point, BS tests $qh'_1 = qh_1$. If the result matches, it accepts the registration process for that user otherwise the registration process is invalid.
11. If the registration is successful, BS generates new public and private keys PuK'_B and PrK'_B and calculating a new TS' time to check these parameters in the UD. In addition, it prepares two new qh' and qh'_i.
12. BS computes confirmation of user registration (CUR) through $qh' \oplus qh||qh'_i \oplus qh_i||PuK'_B \oplus PuK_U \oplus TS'$ and then sends CUR to the intended UD.
13. The user device receives CUR from BS and then extracts qh' by truncating the first 384 bits of CUR and compute it with qh to obtain qh'. In the same way, qh'_i is obtained, but with the second 384-bit truncation of CUR.
14. The user's device extracts PuK'_B from the third 384 bits in CUR.
15. The user's device calculates the current time TS and then checks that CUR has reached within the permissible limits of TS. Then the user's device stores qh' and qh'_i in the qh file instead of the old ones.

To complete the registration phase on the sensor's device side, it performs the following steps:

1. Each sensor uses its stored default parameters (SID, Loc, PuK_S and PrK_S) to start the registration process.
2. Each sensor calculates the current timestamp to prevent many threats such as replay attacks, as well as time is important in our proposal to specify the qh_i entry in the hashes file.
3. The sensor obtains qh_i by calculating the number of hours (Hs) multiplied by the number of minutes (Ms) to arrive at an entry in the file qh.
4. It uses the public key and qh ($qh_i \oplus PuK_S$) to obtain qh_1.
5. The sensor calculates qh_2 by calculating the QUARK hash ($s - qh$) 256-bit for the input $(SID||Loc||TS)$ since all sensor parameters (SID, Loc and TS) are hidden and not clearly transmitted in the unsecured channel.
6. The sensor performs the operation $qh_3 = qh_2 \oplus PuK_B$ to hide the public key of BS.
7. At this point, the sensor computes a sensor registration request (SRR) by executing $qh_1 \oplus qh_3||qh \oplus TS$, and then sends it to BS. It is clear that all parameters are hidden when the SRR request goes from the sensor to BS.
8. BS Receives an SRR request from the sensor. In the same way, it truncates $qh \oplus TS$ from SRR and characterizes the sensor by qh.
9. In addition, it checks TS for attacks.
10. BS retrieves the stored PuK_S to extract qh_1 via $qh_i \oplus P_uK_S$ where qh_i depends on TS.
11. Then BS calculates qh_3 by executing the first 256 bits of SRR and qh_1 ($SSR(256^{1st} \oplus qh_1)$. Next, use $qh_3 \oplus PuK_S$ to calculate qh_2.
12. BS computes a QUARK hash (qh_2' 256-bit) by executing $SID||Loc||TS$. Then it checks $qh_2 = qh_2'$ If the result matches, then BS completes the rest of the registration procedures. Furthermore, matching the result proves that SID is for a legitimate sensor, the sensor location has not changed and the time is within the specified time period of BS.
13. Furthermore, BS generates new parameters (PuK_B', PrK_B' and TS) to be used in the next phases.
14. BS prepares new MDs (qh' and qh_i') to be stored in BS instead of the old ones, and also sent to the sensor to prevent old MD from being used.
15. BS prepares a new confirmation of sensor registration (CSR) containing qh', qh_i', PuK_B' and TS' and sends them to the sensor where all parameters are hidden and inaccessible to the adversary ($\mathcal{A}$).
16. When the sensor receives a CSR request from BS, it uses temporary parameters such as qh_1, qh_2 and the quasi-permanent parameters SID, Loc and PuK_S to extract new parameters such as qh', qh_i', PuK_B' and TS'.
17. Then BS checks TS' to make sure that the CSR request was sent within the time specified on the sensor.
18. BS stores the new parameters qh', qh_i' and PuK_B' to be used in the next phases.

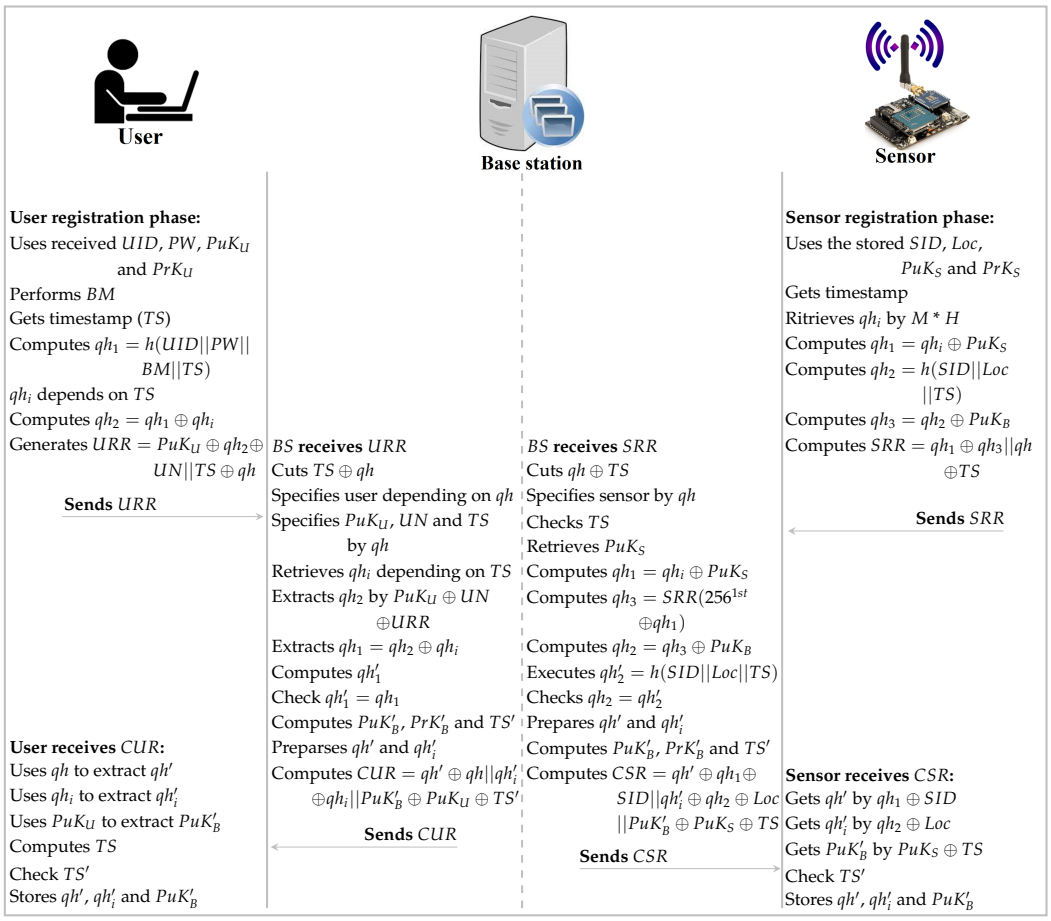

Figure 6. User and sensor registration phase.

4.3.3. Keys Update Phase

After legitimate users and trusted sensors are properly and legally registered, the key update phase comes to ensure that fresh/new keys are used and that safe keys are used, Figure 7 describes the keys update phase. User device side:

1. Initially the UD generates keys (private PrK'_U and public PuK'_U) using the ECDH-384 bits protocol.
2. Then the UD computes qh_1 by $h(BM)$ which is used to hide the new public key through user key update (UKU)= $PuK'_U \oplus PuK_B \oplus qh_1$.
3. Next, the UD computes qh_2 to produce a MD that depends on the password. This operation is important to hide the new public and private keys on the UD and store them on the user's device through the two operations ($qh_2 \oplus PuK'_U$ and $qh_2 \oplus PrK'_U$).
4. At this point, the user's device sends UKU to BS.
5. BS receives the update request UKU from the UD. Then it performs the operation qh_1 (because BS has databases of BM, UID ... etc.) to extract the new public key (PuK'_U) associated with a given UD and store it instead of the old one.
6. Then BS generates public PuK'_B and private PrK'_B to connect to the intended user's device.
7. BS computes BS key update (BKU) to hide the public key of BS by $PuK'_B \oplus PuK_U \oplus qh_1$.
8. Then, BS computes qh_2 through the operation $h(UID)$ to hide the public and private keys of the BS by $qh_2 \oplus PuK'_B$ and $qh_2 \oplus PrK_B$ and then stores them on the BS device. At this point, BS sends BKU to the intended UD.
9. Finally, the UD receives BKU from BS and retrieves PuK'_U to extract PuK'_B by the operation $BKU \oplus PuK_U \oplus qh_1$ and store it on the UD for use in future connections.

Sensor device side:

1. First, each sensor uses ECDH-224 to generate the public PuK'_S and the private PrK'_S.
2. Then each sensor uses qh_1 through $h(Loc)$ to hide the PuK'_S public key associated with the sensor using the $SKU = PuK'_S \oplus PuK_S \oplus qh_1$ operation.
3. Next, each sensor uses $qh_2 = h(SID)$ to hide the sensor's public and private keys, then each sensor stores the new keys and sends sensor key update (SKU) to BS to update the public key.
4. The BS receives SKU and retrieves the location of the sensor to extract the new public key by the operation $SKU \oplus PuK_S \oplus qh_1$ and then stores it in place of the old public key.
5. Furthermore, BS generates the public keys PuK'_B and the private PrK'_B and stores them after they are hidden by the qh_2 operation.
6. BS computes BKU by $BKU = PuK'_B \oplus PuK_S \oplus qh_1$ and then sends BKU to sensor.
7. Sensor receives BKU and retrieves PuK'_S to extract PuK'_B by the operation $BKU \oplus PuK_S \oplus qh_1$ and then stores the public key PuK'_B for the BS.

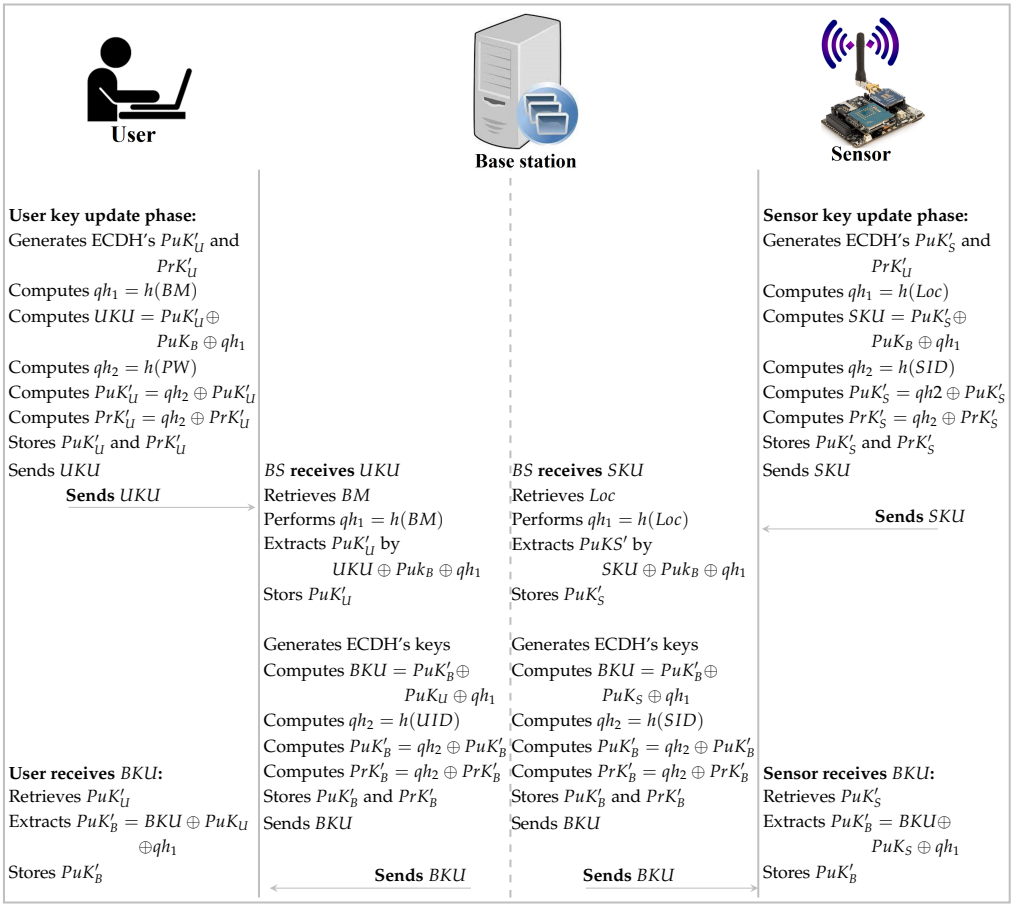

Figure 7. User and sensor key update phase.

4.3.4. Session Key Distribution Phase

After the legitimate devices (users' devices and sensors) have completed the registration and update/exchange of public keys phases securely, the session key distribution phase begins through which data and information packets can be transmitted securely, Figure 8 depicts the steps of session key distribution phase.

1. Initially, the UD enters UID, BM and PW and generates RN_{U_i} to increase the randomness of the session key SK. It then extracts the public and private keys (PrK_U and PuK_U) using $\oplus qh_2$.
2. The UD calculates SK_U by $h(PuK_U||BM||PW||RN_{U_i})$, where SK_U can be divided into SK_{U_1} and SK_{U_2} to be used in the final SK production. Then, the user's device performs

three processes (P_1, P_2 and P_3) to hide the initial session key, the random number and the user ID (UID). Then the UD sends user's session key request (SK_UR) containing P_1, P_2 and P_3 to BS.

3. At this point, BS receives the SK_UR request from the UD. BS uses the public key (PuK_B) to obtain the UID, then uses the latter to obtain RN_{U_i}. Then BS calculates SK_U by $P_1 \oplus PuK_U \oplus RN_{U_i}$. Next, BS uses PuK_U, BM and PW stored plus RN_{U_i} obtained to find SK'_U, and then checks the obtained SK_U equals SK'_U computed, if they match, then BS completes the procedures of the session key distribution phase, otherwise BS refuses to complete the distribution of the SK and stops the connection.
4. If SK'_U is validated, BS generates a new random number RN_{B_i} and calculates the temporary session key SK_B by $h(PuK_B||SID||UID||RN_{B_i})$. Then BS divides both SK'_U and SK_B into two parts ($SK_{U'_1}$ and $SK_{U'_2}$, SK_{B_1} and SK_{B_2}) to obtain a temporary session key through $SK_{tmp} = SK_{U_1}||SK_{B_2}$. Finally, BS completes three processes to hide the temporary session key, UID, SID and RN_{B_i} and to create $SK_{B_1}R$ by $P_1||P_2||P_3$ then send it to the intended sensor.
5. When the sensor receives a $SK_{B_1}R$ request, it uses both the public key (PuK_S) and the sensor identifier (SID) to obtain UID and RN_{B_i}. This procedure also enables the sensor to validate both PuK_S and SID. Then the sensor extracts the temporary session key SK_{tmp} by $P_1 \oplus PuK_B \oplus RN_{B_i}$, where SK_{tmp} consists of SK_{U_1} and SK_{B_2}. Then, the sensor computes SK'_B by $h(PuK_B||SID||UID||RN_{B_i})$ which consists of SK'_{B_1} and SK'_{B_2}. The sensor tests for a match of $SK_{tmp_2} = SK'_{B_2}$, if it matches it means that the temporary SK is valid otherwise the sensor will refuse to connect and consider the SK to be fake.
6. If they are identical, the sensor generates a new RN_{S_i} and calculates the initial session key SK_S by $h(PuK_S||Loc||SID||RN_{S_i})$ which consists of SK_{S_1} and SK_{S_2}. At this point, the sensor SK between the sensor and BS is SK_{SB} consisting of SK_{tmp_2} and SK_{S_1}. It will be used in subsequent encryption and signatures operations that are beyond the scope of this research. Finally, the sensor hides the final session key SK_{SB}, the sensor location Loc, and the SID by P_1, P_2 and P_3 and then sends these processes after being combined in SK_SR into BS.
7. BS receives a SK_SR request and uses PuK_B to extract SID and Loc to extract RN_{S_i}. Then BS calculates SK_{BS} (where $SK_{BS} = SK_{SB}$) by $P_1 \oplus PuK_S \oplus RN_{S_i}$ consisting of SK_{S_1} and SK_{B_2}. Then BS tests $SK_{B_2} = SK'_{B_2}$. If they match then $SK_{BS} = SK_{S_1}||SK_{B_2}$ is the final session key between BS and the sensor otherwise it refuses to connect. Then BS generates a new RN_{B_i} with the final session key SK_{BU} (that consists of SK_{B_1} and SK_{U_2}) between BS and user, then hides final SK_{BU} and RN_{B_i} by the processes P_1, P_2 and $P3$ to prevent adversaries from exposing the exchanged keys and network security parameters. Finally, BS sends a $SK_{B_2}R$ request containing $P_1||P_2||P_3$ to the UD to complete the session key distribution phase.
8. On the UD side, the UD receives the $SK_{B_2}R$ request and extracts UID to validate the request and extracts RN_{B_i} to calculate the session key. The user's device extracts SK_{UB} (where $SK_{UB} = SK_{BU}$) by $P_1 \oplus PuK_B \oplus RN_{B_i}$. Then the UD splits SK_{UB_2} into SK_{UB_1} and SK_{UB_2} and tests that $SK_{U_2} = SK_{UB_2}$ if identical then SK_{UB} is the final session key between the user and BS, otherwise the key is modified by adversaries as the UD will reject it.

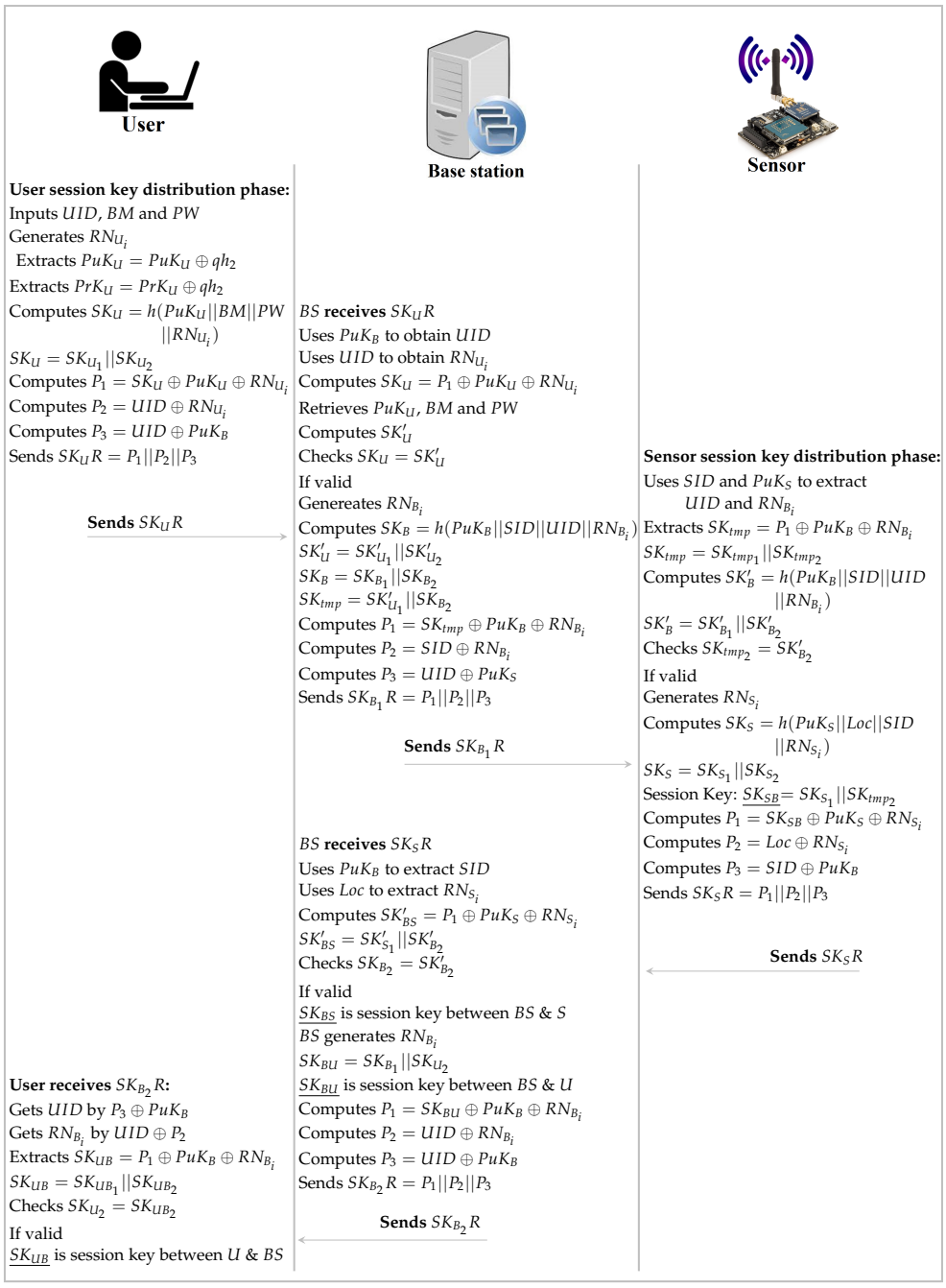

Figure 8. User and sensor session key distribution phase.

4.3.5. Session Key Protection Phase

To make a session key secure against attacks, the keys stored on the devices must be protected. On the user's device, it first requires extracting the private key in the process $PrK_U = PrK_U \oplus qh_2$ and then doing the operation $SK'_{UB} = SK_{UB} \oplus PrK_U \oplus qh_2 \oplus PW$ to hide the SK on the device the user. As for the sensor, it performs the operation $PrK_S = PrK_S \oplus qh_2$. Next, the sensor performs the operation $SK'_{SB} = SK_{SB} \oplus PrK_S \oplus qh_2 \oplus SID$ to hide the SK in the sensors. Implementing these operations provides security robustness to protect exchanged keys, especially session keys.

5. Proposal Results and Findings Discussion

This part of the research will deal with the results obtained from the proposed key exchange protocol in addition to evaluating the results during their discussion with the results of the existing research.

5.1. Analysis of the Proposed Protocol Security

For security analysis, this section will include an extensive formal and informal security analysis.

5.1.1. Informal Analysis of the Proposed Protocol Security

This section will address the attacks identified in the threat model (scope of the study) section and demonstrate the ability of the proposed key exchange protocol to resist these threats. In addition, Table 1 shows a comparison of the threat resistance (TR) between the proposed key exchange protocol and the existing protocols.

- Known session key threat (KSKT) resistance: In insecure environments, the adversary tries to use the keys of the previous sessions to gain access to the keys of the current session. If the adversary can derive the keys of the current session, it becomes legitimate and can perform malicious activities. In the proposed key exchange protocol, all previous and updated session keys are hidden (e.g., $P_1 = SK_{tmp} \oplus PuK_B \oplus RN_{B_i}$) and not sent explicitly. Furthermore, there is no correlation or relationship between the previous and updated keys. Additionally, there is high randomness using RN_{B_i} to support robust session key protection. Therefore, the proposal resists this attack with high rigidity.
- Resistance of stolen verify threat (SVT): In this type of attack, the adversary tries to steal security parameters, whether related to users' devices, sensors, or the network. Stolen validation parameters can simply give the adversary authentication and verification in the network. The adversary tries to access the database containing confidential information by hacking users' devices and sensors. In the proposed key exchange protocol, all security parameters such as PW and keys are either session or private keys, even public keys are not explicitly stored. For example, the public key of a particular user's device is hidden by $PuK'_U = qh_2 \oplus PuK'_U$, also, the password is not stored on the user's device ($qh_2 = h(PW)$). Therefore, the adversary cannot derive the keys and security parameters. The proposed protocol thus successfully counters the threat of a stolen verifier.
- Session key and security condition (SKSC): In order for a session key to be secure, it must be protected from accessing its analysis and derivation by the adversary. In our protocol, the adversary cannot access, derive or analyze security parameters or keys because first the session key is not repeated in subsequent sessions and second, the session key between the user's device and BS such as $SK_{BU} = SK_{B_1}||SK_{U_2}$ is different from the session key between BS and S as $SK_{SB} = SK_{S_1}||SK_{tmp_2}$. Furthermore, all security parameters such as sensor location $P_2 = Loc \oplus RN_{S_i}$ and keys such as the public key are sent to the recipient anonymously as $P_3 = SID \oplus PuK_B$. Therefore, deriving or computing the session key by the adversary is very difficult in our protocol. Thus, the security condition of the session key in our protocol is high and suitable for protecting patients' data.
- Providing perfect forward secrecy (PFS): In this attack, the adversary tries to take advantage of the current session key and long-term security parameters (assuming that it can access the session key); it cannot take advantage of this key in subsequent sessions because in our protocol the keys for subsequent sessions do not depend on the keys of the current sessions. Furthermore, the session key in our protocol is unique and random through the execution of $SK_B = h(PuK_B||SID||UID||RN_{B_i})$, even if the adversary can access the long-term parameters such as SID, UID ..., etc. It cannot be used to derive subsequent (future) session keys because the RN_{B_i} value in our

proposed protocol provides randomness within the implementation of the qh function. Hence our proposed protocol efficiently provides PFS.

- Random parameters detection threat (RPDT) resistance: The goal of this attack is to obtain random parameters to help expose the session key and other keys to network entities. The adversary tries to detect random parameters such as RN_{U_i}, RN_{B_i}, and RN_{S_i} to obtain the other security parameters. Even if the adversary exposes the random parameters, he/she cannot access the session key because the initial session key for each entity consists of two parts such as $SK_U = SK_{U_1}||SK_{U_2}$ and the final session key is agreed as $SK_{UB} = SK_{UB_1}||SK_{UB_2}$ between entities depending on multiple processes as shown in Figure 8. Moreover, random parameters are not explicitly sent over the network such as $SK_U = P_1 \oplus PuK_U \oplus RN_{U_i}$. Therefore, our proposed protocol withstands RPDT.
- Key recovery threat (KRT) resistance: In this type of attack, the adversary tries to extract the key assuming that the attacker has the plaintext and ciphertext. This attack is not viable if the security parameters are chosen appropriately. The proposed ECDH protocol generates security keys of sufficient length (384 bits) to block this attack. Furthermore, our protocol does not suffer from information leakage or security parameters that help to extract the keys. Moreover, random key generation in addition to the regular generation of these keys prevents hacking of our protocol. Therefore, our proposed protocol successfully counteracts this threat.
- Known-session-specific temporary threat (KSSTT) resistance: Detection of some temporary information can lead to session key exposure in vulnerable key exchange protocols. For example, the adversary tries to reveal temporary random values or parameters to obtain the session key. In our proposed protocol, it is very difficult for the adversary to obtain the random values, such as RN_{U_i}, but if we assume that the adversary can obtain the random values in the session, he/she cannot obtain the session key ($SK_U = h(PuK_U||BM||PW||RN_{U_i})$) because it relies on other permanent parameters such as BM or semi-permanent parameters such as PW and not stored by users' devices. Even if the adversary can obtain the random value and the qh value, it will not be able to reveal the session key. Thus, our protocol reliably combats a KSSTT.
- User impersonation threat (UIT) resistance: To accomplish the impersonation threat, the adversary needs to prepare a login request that contains valid information (such as keys and security parameters) for a legitimate user. Fortunately, our protocol relies on the security parameters BM and PW (two factors authentication), the adversary does not have access to these security parameters because these parameters are not stored on UDs and are not explicitly transmitted from sender to recipient. Thus, the adversary cannot imitate the user's device, i.e., our protocol aptly resists this threat.
- Sensor node impersonation threat (SIT) resistance: Similar to the previous attack, the adversary needs secret information to imitate the sensor. In our protocol, the adversary can only imitate sensors when secret parameters such as SID, Loc, PuK_S and PrK_S are obtained. As all these parameters are hidden securely, whether in storage or transmission. For instance, even if the adversary penetrates the sensor, he/she cannot extract the private key $PrK'_S = qh_2 \oplus PrK'_S$. Therefore, our proposed protocol successfully resists the SIT.
- Post-deployment keys threat (PDKT) resistance: Sometimes the adversary exploits key exchange vulnerabilities post-deployment of sensors, adding a user device to the network or in the case of updating the public/private keys of users' devices and sensors. In all of these cases, the adversary tries to derive, alter or block/intercept the keys. Our protocol generates public and private keys on a regular basis. In the phase of user and sensor key update (Figure 7), the adversary cannot derive the keys because they are hidden ($UKU = PuK'_U \oplus PuK_B \oplus qh_1$) when transmitting via the unsecured channel and protected ($PuK'_U = qh_2 \oplus PuK'_U$) when stored in users' devices or sensors. Furthermore, the adversary cannot prevent the user/sensor update request for the keys from reaching the intended target because the user/sensor device will wait

for the BKU request which means the key update is completed. Thus, our protocol successfully resists PDKT.

- User/sensor session key verification (U/SSKV) threat resistance: In this attack, the adversary attempts to fail the session key verification completion to prevent the use of a legitimate session key between entities. In our protocol, all entities (UD, BS and S) produce primary keys (SK_U, SK_B and SK_S) to eventually obtain final keys (SK_{SB}, SK_{BS}, SK_{BU} and SK_{UB}) after completion of $P_1 = SK_{BU} \oplus PuK_B \oplus RN_{B_i}$ in verification request $SK_{B_2}R$. The adversary does not know the parameters of the verification request $SK_{B_2}R$ and cannot falsify it because it is hidden and any manipulation of it will be clear to the receiver. As a result, our protocol worthily provides U/SSKV property.
- Session key computation threat (SKCT) resistance: This attack relies on obtaining a session key creation/updating request. The adversary tries to analyze the session key request to compute it. In our protocol, each session key consists of two parts ($SK_{U_1}||SK_{U_2}$, $SK_{B_1}||SK_{B_2}$ and $SK_{S_1}||SK_{S_2}$), one part is used to produce the final session key and the other part is used for verification (see Figure 8). For example, SK_{U_1} (on the user's device side) is used to produce the session key while SK_{U_2} is used to verify the session key. In our protocol, the adversary cannot compute the session key because first, all the parameters are hidden, secondly, the adversary does not know how to generate the initial and final keys for the session keys. Thus, our protocol ably resists this threat.
- Resistance of off-line password guessing threat (OPGT): In this threat, the adversary tries to guess the passwords in the requests transmitted between the sender and the recipient. After the adversary obtains some key exchange or registration request, he/she tries different methods, such as a dictionary and brute force, to guess the correct passwords. In our protocol, first, the passwords are not stored on UDs. Furthermore, users' passwords are not explicitly transmitted over the network. Throughout our protocol phases (Figures 6–8), passwords are used within a QUARK hash of length 256 MD with S_i or 384 MD with UD_i (i.e., $qh_1 = h(UID||PW||BM||TS)$) which it is impossible for an adversary to extract the password from qh_1 even if the adversary has some parameters such as UID and PuK_U. Therefore, our protocol successfully resists the OPGT.
- Resistance of jamming threat (JT): This is a type of service obfuscation threat that mainly targets servers and BSs. The adversary sends unnecessary packets at the same network frequency to stop services. The adversary can use it in many forms such as constant, deceptive, random and reactive. Our protocol uses authentication procedures that greatly mitigate this threat. Since our protocol uses $TS \oplus qh$ in BSs to authenticate the registration of both the user and sensor. Thus, our protocol provides resistance to this type of threat.

Table 1. Resistance of threats to both the proposed protocol and existing protocols.

TR	Ametepe et al. [10]	Kardi and Zagrouba [22]	Lin [25]	Rangwani et al. [26]	Thabit et al. [27]	Gope and Sikdar [28]	Mehra et al. [29]	Wu et al. [30]	Wu et al. [31]	Proposed
KSKT										✓
SVT				✓			✓	✓		✓
SKSC			✓				✓			✓
PFS				✓		✓	✓	✓	✓	✓
RPDT				✓		✓				✓
KRT	✓				✓					✓
KSSTT						✓		✓	✓	✓
UIT				✓		✓	✓	✓	✓	✓
SIT				✓				✓		✓
PDKT										✓
U/SSKV			✓				✓			✓
SKCT	✓			✓	✓	✓				✓
OPGT				✓			✓	✓	✓	✓
JT		✓								✓

5.1.2. Formal Analysis of the Proposed Protocol Security

To verify the ability of key exchange protocols to fend off security threats, researchers and academics use formal tools such as Scyther to demonstrate that designed security protocols are immune to known attacks and are free from vulnerabilities. In this section, we will provide a Scyther summary, proposed ECDH-QUARK protocol with Scyther and results.

Scyther Summary:

Among the formal tools for checking vulnerabilities is the Scyther checker tool. This tool is effective in testing and analyzing vulnerabilities of security protocols [35]. It is very popular for testing key exchange and security protocols. This tool has the ability to analyze many threats such as password guessing, insider, impersonation, session key and established key that cannot be analyzed in other analysis tools such as ProVerif and Akiss. Scyther uses specific language to describe protocols, roles, parameters, send/receive requests, and claims. Scyther language is a security protocol description language (SPDL) that is similar to the syntax of Java. Furthermore, it provides a graphical interface to check results and test threats easily and simply [36]. The security protocol is designed based on the roles of network entities and implemented a set of claim events on the security parameters in requests to send () and recv (). Among these claims are:

- Secret: Keeping secret terms
- Alive: Proof that key exchange protocol steps are implemented in the network entity.
- Weakagree: Ensuring that an impersonation threat will not be carried out by an adversary
- Nisynch: Verifying that send and receive requests are between two connected entities
- Niagree: Ensuring that transmission and receiving requests are transmitted in the correct order and integrated data between network entities
- Match: Testing the equality of two variables or assigning a value to a variable

Scyther provides pattern discipline, shortened representations and analyzed traces. This checker helps in the analysis of classes of threats and expected protocol behaviors [37], or to prove correctness for an unbounded number of established sessions for protocol. The checker has been swimmingly used in both teaching and scientific research.

Proposed ECDH-QUARK Protocol with Scyther:

Our protocol was analyzed by the Scyther checker after writing the protocol code in SPDL. As noted earlier, the design of a security protocol in SYTHER depends on the roles of that protocol. Therefore, our proposed protocol is based on three roles (UID, BS and S). Furthermore, our protocol contains four "send" events and four "recv" events to safely complete the session key distribution. It sends and receives four requests (SK_UR, $SK_{B2}R$, $SK_{B2}R$ and SK_SR) between the designed protocol roles. All security procedures and parameters in Figure 8 have been applied to analyze the security of our proposed ECDH-QUARK protocol. In our protocol, the Secret, Nisynch, Niagree and Match claims are implemented and the confidentiality and concealing of information (such as SID and UID) and security parameters (such as SK_{ub1} and SK_{ub2}) are implemented.

Results:

After implementing our protocol in the Scyther tool, the results indicate that our ECDH-QUARK protocol is safe against attacks and threats (shown in Figure 9). Where all session keys (SK_{ub}, SK_{bu}, SK_{sb} and SK_{bs}) are between roles, and public ($PuKu$, $PuKb$ and $PuKs$), random parameters (RN_u, RN_b and RN_s) and hardware information (Loc, SID and UID) are secure and confidential. In Figure 9, we notice that the "Comments" section indicates "no attacks" which means that our protocol is secure against known threats.

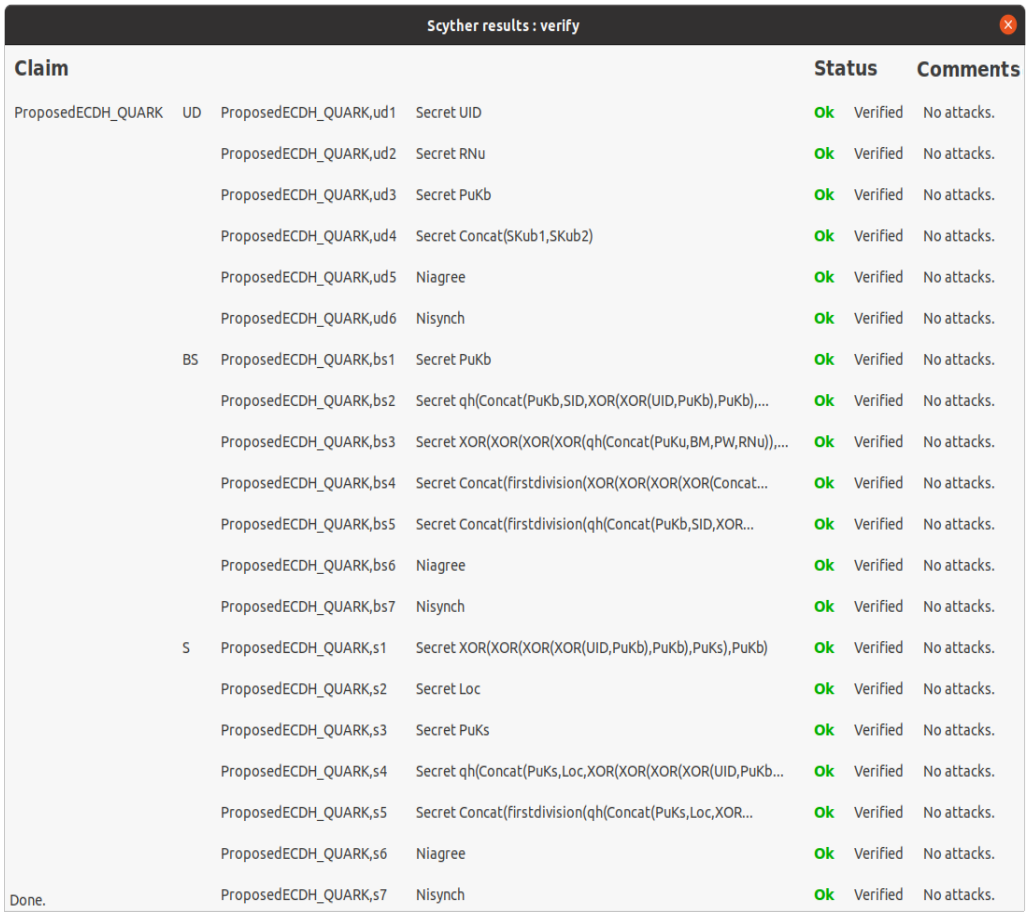

Figure 9. Proposed ECDH-QUARK results by SYTHER checker.

5.2. Performance Analysis of the Proposed Protocol

In this section, we will analyze the performance of the proposed key exchange protocol. We will demonstrate how our protocol provides lightweight mechanisms of performance while providing trustworthy security.

- **Computational cost**
  All key exchange protocols require a computational cost evaluation of the embedded security measures to demonstrate their applicability. Therefore, we tested the execution of security processes to obtain the execution time for all these processes, Table 2 shows the execution time for key exchanges such as T_{qh}, T_{KE} and T_{SM}. To illustrate the computational costs, we compare our protocol with existing key exchange protocols ([3,23,26,28,30,31,38]), where costs are evaluated across all network components (UD, BS and S) and then the total cost and time cost are calculated (see Table 3). For comparing costs of computation, there are some acronyms that should be defined and some of which may not be used in our protocol.
  - Execution time for symmetric encryption (T_S)
  - Execution time for fuzzy extractor (T_{fe})
  - Execution time for hash function (T_h)
  - Standard hash function (SH) and lightweight hash function (LH)

  We tested the computation costs in the user device and sensor session key distribution phase (Figure 8) with existing key exchange protocols, because the registration and update phases are not applied continuously. Furthermore, the costs of the RN and XOR operations are small and negotiable so we do not include them in the computation costs comparison. We can see from Table 3 that the computational costs in our protocol are $UD = 1T_{qh} + 1T_{KE} + 1T_{SM}$, $BS = 1T_{qh} + 1T_{KE} + 1T_{SM}$ and $S = 1T_{qh} + 1T_{KE} +$

$1T_{SM}$, as well as the total cost, is $3T_{qh} + 3T_{KE} + 3T_{SM}$ that is the best of all recent key exchange protocols. Furthermore, our protocol implements a lightweight hash (LH) algorithm that is QUARK to generate light session keys compared to existing protocols that implement standard hash (SH). Moreover, the cost of execution time for existing protocols such as [3] = 0.1453 s, [26] = 8.9730 ms, [28] = 94.903 ms, [31] = 110.50 s and [38] = 421.25 ms consumes a big execution time (which will negatively affect resource-constrained sensors) compared to our protocol which needs ≈0.012816 ms. In Baghbanijam and Sanaei [3], the key exchange phase requires encryption and therefore the execution time cost will be very large compared to our proposed protocol. Phimphinith et al. [23] and Wu et al. [30] did not specify execution time costs and thus it is difficult to make a comparison with our protocol. Rangwani et al. [26] and Gope and Sikdar [28] used many parameters such as using more than six hashes, which makes key exchange complicated to calculate. In Wu et al. [31], the authors used the AKE protocol which performs complex operations due to the execution of 11 T_{SM} which greatly affects the performance. Similarly, another paper Qi and Chen [38] relied on AKE with ECC to apply key exchange with two factors authentication. However, AKE is not compatible with WSN-based health networks. The authors of this paper indicated that the sensor computation cost is 136.35 ms, while in our protocol the sensor computation cost is 0.004272 ms. After evaluating the computation costs, we note that the proposed key exchange protocol offers the best performance out of the existing protocols.

Table 2. Cryptography key exchange operations.

Notation	Description	Time Cost
T_{qh}	c-QUARK hash function	0.002892 ms
T_{KE}	Key exchange (ECDH)	0.000847 ms
T_{SM}	Scalar multiplication	0.000533 ms
T_S	Symmetric encryption	-
T_{fe}	Fuzzy extractor	-
T_h	One-way hash function	-

Table 3. Comparison of computational costs between key exchange protocols.

Protocol	*UD*	*BS*	*S*	Total	Exchange Type	Hash	Time Cost
Baghbanijam and Sanaei [3]	$3T_h + 4T_{KE} + 2T_S$	$4T_h + 3T_{KE} + 2T_S$	$3T_h + 2T_{KE}$	$10T_h + 7T_{KE} + 4T_S$	ECDH	SH	0.1453 s
Phimphinith et al. [23]	$4T_h + 2T_{KE} + 2T_{SM}$	$4T_h + 1T_{KE} + 1T_{SM}$	-	$8T_h + 3T_{KE} + 3T_{SM}$	ECDH	-	-
Rangwani et al. [26]	$4T_h + 2T_{KE} + 2T_{SM}$	$7T_h + 1T_{KE} + 1T_{SM}$	$4T_h + 1T_{KE} + 1T_{SM}$	$15T_h + 4T_{KE} + 4T_{SM}$	ECDH	SH	8.9730 ms
Gope and Sikdar [28]	$6T_h + 2T_S$	$5T_h + 2T_S$	-	$11T_h + 4T_S$	AES-CBC	SH	94.903 ms
Wu et al. [30]	$1T_{fe} + 11T_h + 1T_S$	$14T_h + 1T_S$	$6T_h + 1T_S$	$1T_{fe} + 31T_h + 3T_S$	-	-	-
Wu et al. [31]	$1T_{fe} + 8T_h + 1T_{KE} + 2T_{SM}$	$13T_h + 2T_{KE} + 4T_{SM}$	$5T_h + 1T_{KE} + 3T_{SM}$	$1T_{fe} + 26T_h + 4T_{KE} + 9T_{SM}$	ECDH	SH	110.50 s
Qi and Chen [38]	$7T_h + 1T_{KE} + 3T_{SM}$	$6T_h + 1T_{KE} + 1T_{SM}$	$3T_h + 2T_{KE} + 2T_{SM}$	$16T_h + 4T_{KE} + 6T_{SM}$	ECC + AKE	SH	421.25 ms
Proposed	$1T_{qh} + 1T_{KE} + 1T_{SM}$	$1T_{qh} + 1T_{KE} + 1T_{SM}$	$1T_{qh} + 1T_{KE} + 1T_{SM}$	$3T_{qh} + 3T_{KE} + 3T_{SM}$	ECDH	LH	0.012816 ms

- **Communication cost** Another important aspect of evaluating key exchange protocols is the calculation of communication costs. In our user and sensor session key distribution phase (Figure 8), each entity has a communication cost, for example, UD needs PuK_U = 384 bits, RN_{U_i} = 64 bits and UID = 32 bits, where the cost of P_1 = 384 bits, P_2 = 64 and P_3 = 384 bits. Thus, the communication cost of UD becomes 832 bits,

which is similar to the costs of BS = 832 bits and S = 832 bits. As a result, the total cost of communication in our protocol is 2496 bits. By comparing our protocol with the communication costs in Table 4, we notice that our protocol outperforms the protocols in [30,31] in terms of data size. Although the protocol in [38] performs three handshakes and a data size of 2368 bits, compared to our protocol that performs four handshakes and a data size of 2496 bits; this does not reflect the overall performance of the protocol [38], as our protocol (which uses $1T_{qh}$) outperforms the protocol in Qi and Chen [38] (which uses $6T_h$) in computation cost. In addition, their protocol contains a clear security breach when it reduces the number of handshakes from 4 to 3, as there are no parameters checked (third handshake) in BS between S and UD. Thus, our protocol balances security and performance to outperform existing protocols in terms of communication cost.

Table 4. Comparison of communication costs between key exchange protocols.

Protocol	Handshake	Bit per Data
Wu et al. [30]	4	2944
Wu et al. [31]	4	4448
Qi and Chen [38]	3	2368
Proposed	4	2496

6. Limitations of Proposed Protocol

The design of a key exchange protocol, no matter how precise the design, may contain some limitations and open issues. Therefore, we present some of the limitations that may be encountered in implementing our protocol. For example, if the network is large (a huge number of users and sensors) there can be use of identical parts of the same session key for different users. Moreover, it is possible (in some cases) that the proposal establishes a session key for a specific user without establishing a session key for the sensors and vice versa if there is a hardware failure of the network components or because the proposal has not been tested against various attacks and threats. Furthermore, any failure that affects the BS device can be the cause of the network downtime, especially since the key exchange phase is highly dependent on the verification procedures in BS.

7. Conclusions and Future Work

Providing a trusted key exchange protocol gives users (patients and providers) confidence that their information and data are secure. However, providing a security methodology for exchanging keys without caring about protocol performance would leave a big loophole in the security protocol. This protocol will not be acceptable to health institutions and individuals due to its inapplicability in those health institutions. Therefore, we proposed a key exchange (ECDH-QUARK) protocol that balances security and performance. We tested our protocol against a range of malicious attacks targeting key exchange. The results of the analyses indicate that our protocol is able to efficiently fend off these threats. Furthermore, our protocol provided a balanced performance in terms of computation and communication costs when compared with the existing search results. Among the expected future works of this study are:

1. Test the proposed protocol on actual health applications such as surgeries that require high speed and security to exchange keys in joining and leaving the network. Especially when the surgeon is not present in the surgical procedure, but rather supervises it online.
2. This proposal deals with the exchange of keys and the distribution of session keys securely between network entities, we intend to develop this proposal in the use of these keys to accomplish the processes of encryption and signing of patient data and information.
3. Test our proposal on wearable and implantable health sensors in the human body to assess energy consumption and calculate the computational cost and complexity.

Funding: This research received no external funding.

Conflicts of Interest: The author declares no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

$\mathcal{A}$, BS, S_i, UD_i	Adversary, base station, sensor, user device
PuK_U, PuK_S, PuK_B	Public keys of users ,sensors, and BS
PrK_U, PrK_S, PrK_B	Private keys of users ,sensors, and BS
SK_{SB}, SK_{UB}	Session keys between sensors and BS, users and BS
SK_{tmp}	Temporary session key
UID, SID	Identifiers of users and sensors
Loc	Sensors locations
PW, BM	Users' passwords and biometrics
UN	User name
RN_{U_i}, RN_{S_i}, RN_{B_i}	Random numbers for sensors, users and BS
$qh(.)$ or $h(.)$	One-way QUARK hash function
$\|$, $\oplus$	Concatenation and exclusive or operations

References

1. Frimpong, E.; Michalas, A. IoT-cryptodiet: Implementing a lightweight cryptographic library based on ECDH and ECDSA for the development of secure and privacy-preserving protocols in contiki-NG. In Proceedings of the IoTBDS 2020, Prague, Czech Republic, 7–9 May 2020; pp. 101–111.
2. Awaad, M.H.; Jebbar, W.A. Prolong the lifetime of WSN by determining a correlation nodes in the same zone and searching for the best not the closest CH. *Int. J. Mod. Educ. Comput. Sci.* **2014**, *6*, 31. [CrossRef]
3. Baghbanijam, S.; Sanaei, H. An improved authentication & key exchange protocol based on ECDH for WSNs. *arXiv* **2021**, arXiv:2109.11450.
4. Al-Zubaidie, M.; Zhang, Z.; Zhang, J. REISCH: Incorporating lightweight and reliable algorithms into healthcare applications of WSNs. *Appl. Sci.* **2020**, *10*, 2007. [CrossRef]
5. Awaad, M.H. The use of dynamic sliding window with IPSec. *J. Educ. Pure Sci.* **2014**, *4*, 278–289.
6. Awaad, M.H. Improve the effectiveness of sensor networks and extend the network lifetime using 2BSs and determination of area of CHs choice. *J. Comput. Sci. Control. Syst.* **2014**, *7*, 15.
7. Abdulhameed, H.A.; Abdalmaaen, H.F.; Mohammed, A.T.; Mosleh, M.F.; Abdulhameed, A.A. A lightweight hybrid cryptographic algorithm for WSNs tested by the diehard tests and the raspberry Pi. In Proceedings of the 2022 International Conference on Computer Science and Software Engineering (CSASE), Duhok, Iraq, 15–17 March 2022; pp. 271–276.
8. Al-Zubaidie, M.; Zhang, Z.; Zhang, J. RAMHU: A new robust lightweight scheme for mutual users authentication in healthcare applications. *Secur. Commun. Netw.* **2019**, *2019*, 3263902. [CrossRef]
9. Heigl, M.; Schramm, M.; Fiala, D. A lightweight quantum-safe security concept for wireless sensor network communication. In Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kyoto, Japan, 11–15 March 2019; pp. 906–911.
10. Ametepe, A.F.-X.; Ahouandjinou, A.S.; Ezin, E.C. Robust encryption method based on AES-CBC using elliptic curves Diffie–Hellman to secure data in wireless sensor networks. *Wirel. Netw.* **2022**, *28*, 991–1001. [CrossRef]
11. Marhoon, A.F.; Awaad, M.H.; Jebbar, W.A. A new algorithm to improve LEACH protocol through best choice for cluster-head. *Int. J. Adv. Eng. Sci.* **2014**, *4*, 1–12.
12. Al-Zubaidie, M.; Zhang, Z.; Zhang, J. PAX: Using pseudonymization and anonymization to protect patients' identities and data in the healthcare system. *Int. J. Environ. Res. Public Health* **2019**, *16*, 1490. [CrossRef]
13. Aikins-Bekoe, S.; Hayfron-Acquah, J.B. Elliptic curve diffie-hellman (ECDH) analogy for secured wireless sensor networks. *Int. J. Comput. Appl.* **2020**, *176*, 1–8. [CrossRef]
14. Al-Zubaidie, M.; Zhang, Z.; Zhang, J. Efficient and secure ECDSA algorithm and its applications: A survey. *Int. J. Commun. Netw. Inf. Secur. (IJCNIS)* **2019**, *11*, 7–35. [CrossRef]
15. Al-Zubaidie, M.; Zhang, Z.; Zhang, J. User authentication into electronic health record based on reliable lightweight algorithms. In *Handbook of Research on Cyber Crime and Information Privacy*; IGI Global: Hershey, PA, USA, 2021; pp. 700–738.
16. Almansoori, M.N.; Elshamy, A.A.; Mustafa, A.A.M. Secure Z-MAC protocol as a proposed solution for improving security in WSNs. *Information* **2022**, *13*, 105. [CrossRef]
17. Nagesh, O.; Naresh, V.S. Comparative analysis of MOD-ECDH algorithm with various algorithms. *Int. J. Ind. Eng. Prod. Res.* **2020**, *31*, 301–308.
18. Wu, T.-Y.; Yang, L.; Meng, Q.; Guo, X.; Chen, C.-M. Fog-driven secure authentication and key exchange scheme for wearable health monitoring system. *Secur. Commun. Netw.* **2021**, *2021*, 8368646. [CrossRef]

19. Kim, M.; Moon, J.; Won, D.; Park, N. Revisit of password-authenticated key exchange protocol for healthcare support wireless communication. *Electronics* **2020**, *9*, 733. [CrossRef]
20. Alzahrani, B.A. Secure and efficient cloud-based IoT authenticated key agreement scheme for e-health wireless sensor networks. *Arab. J. Sci. Eng.* **2021**, *46*, 3017–3032. [CrossRef]
21. Butpheng, C.; Yeh, K.-H.; Xiong, H. Security and privacy in IoT-cloud-based e-health systems—A comprehensive review. *Symmetry* **2020**, *12*, 1191. [CrossRef]
22. Kardi, A.; Zagrouba, R. Hybrid cryptography algorithm for secure data communication in WSNs: DECRSA. In *Congress on Intelligent Systems*; Springerr: Berlin/Heidelberg, Germany, 2020; pp. 643–657.
23. Phimphinith, A.; Anping, X.; Zhu, Q.; Jiang, Y.; Shen, Y. An enhanced mutual authentication scheme based on ECDH for IoT devices using ESP8266. In Proceedings of the 2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN), Chongqing, China, 12–15 June 2019; pp. 490–496.
24. Hasan, N.A.; Farhan, A.K. Security improve in ZigBee protocol based on RSA public algorithm in WSN. *Eng. Technol. J.* **2019**, *37*, 67–73. [CrossRef]
25. Lin, H.Y. Integrate the hierarchical cluster elliptic curve key agreement with multiple secure data transfer modes into wireless sensor networks. *Connect. Sci.* **2022**, *34*, 274–300. [CrossRef]
26. Rangwani, D.; Sadhukhan, D.; Ray, S.; Khan, M.K.; Dasgupta, M. An improved privacy preserving remote user authentication scheme for agricultural wireless sensor network. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4218. [CrossRef]
27. Thabit, F.; Alhomdy, S.; Al-Ahdal, A.H.; Jagtap, S. A new lightweight cryptographic algorithm for enhancing data security in cloud computing. *Glob. Transitions Proc.* **2021**, *2*, 91–99. [CrossRef]
28. Gope, P.; Sikdar, B. A privacy-aware reconfigurable authenticated key exchange scheme for secure communication in smart grids. *IEEE Trans. Smart Grid.* **2021**, *12*, 5335–5348.
29. Mehra, P.S.; Doja, M.N.; Alam, B. Codeword authenticated key exchange (cake) light weight secure routing protocol for WSN. *Int. J. Commun. Syst.* **2019**, *32*, 1–27.
30. Wu, T.-Y.; Yang, L.; Lee, Z.; Chu, S.-C.; Kumari, S.; Kumar, S. A provably secure three-factor authentication protocol for wireless sensor networks. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 5537018. [CrossRef]
31. Wu, T.-Y.; Lee, Z.; Yang, L.; Luo, J.-N.; Tso, R. Provably secure authentication key exchange scheme using fog nodes in vehicular ad hoc networks. *J. Supercomput.* **2021**, *77*, 6992–7020. [CrossRef]
32. Lu, X.; Li, B.; Liu, M.; Lin, D. Improved conditional differential attacks on lightweight hash family QUARK. *Cybersecurity* **2022**, 5, 1–16. [CrossRef]
33. Abed, S.; Jaffal, R.; Mohd, B.J.; Al-Shayeji, M. An analysis and evaluation of lightweight hash functions for blockchain-based IoT devices. *Clust. Comput.* **2021**, *24*, 3065–3084. [CrossRef]
34. Gupta, D.N.; Kumar, R. Sponge based lightweight cryptographic hash functions for IoT applications. In Proceedings of the 2021 International Conference on Intelligent Technologies (CONIT), Hubli, India, 25–27 June 2021; pp. 1–5.
35. Cremers, C. Scyther User Manual-Draft 18 February 2014. 2014. Available online: https://people.cispa.io/cas.cremers/scyther/ (accessed on 3 August 2022).
36. Mohammad, Z. Cryptanalysis and improvement of the yak protocol with formal security proof and security verification via Scyther. *Int. J. Commun. Syst.* **2020**, *33*, e4386. [CrossRef]
37. Amin, R.; Lohani, P.; Ekka, M.; Chourasia, S.; Vollala, S. An enhanced anonymity resilience security protocol for vehicular ad hoc network with Scyther simulation. *Comput. Electr. Eng.* **2020**, *82*, 106554. [CrossRef]
38. Qi, M.; Chen, J. Secure authenticated key exchange for WSNs in IoT applications. *J. Supercomput.* **2021**, *77*, 13897–13910. [CrossRef]