Noname manuscript No.
(will be inserted by the editor)

Pragmatic Authenticated Key Agreement for IEEE Std 802.15.6

Haibat Khan · **Benjamin Dowling** · **Keith M. Martin**

Received: date / Accepted: date

Abstract The IEEE Std 802.15.6 is the latest international standard for Wireless Body Area Networks (WBANs). The security of communication in this standard is based upon four elliptic-curve based key agreement protocols. These protocols have been shown to exhibit serious security vulnerabilities but surprisingly, do not provision any privacy guarantees. To date, no suitable key agreement protocol has been proposed which fulfills all the requisite objectives for IEEE Std 802.15.6. In this paper two key agreement protocols are presented which, in addition to being efficient and provisioning advance security properties, also offer the essential privacy attributes of anonymity and unlinkability. The protocols are also quantum-safe as they are independent of any public-key based operations. We develop a formal security and privacy model in an appropriate complexity-theoretic framework and prove the proposed protocols secure in this model.

Keywords anonymity · key-exchange protocols · forward-secrecy · privacy · unlinkability

1 Introduction

Wireless Body Area Networks (WBANs) consist of miniaturized computing devices which can be fitted inside or around the human body [7]. Through use of short range communication technologies, these devices talk to a designated centralized node (Hub) which further communicates with external networks via a Gateway [23]. The general layout of a typical WBAN is illustrated in Fig 1. Note that the Hub and Gateway are functionally two separate entities, but are usually combined into a single physical node. Mindful of the peculiarities of communicating in and around the human body, the IEEE published IEEE Std 802.15.6 [3] for WBAN communications in 2012. As high power transmissions are harmful to humans and nodes in a WBAN are energy constrained, this standard provisioned an optional two-hop communication architecture to enable resource-constrained nodes to minimize transmissions when communicating with the Hub.

In addition to conventional security guarantees, privacy is of utmost importance for typical target application areas such as healthcare and the military [29]. The elliptic-curve based session key agreement methods of IEEE Std 802.15.6 have been shown to have security weaknesses [28], but also do not provide the privacy features that should be expected of a WBAN [20]. In this paper, we present two key agreement protocols which render a comprehensive range of security and privacy properties, which are regarded as essential [20] for WBANs. We start by presenting a network and adversary model for WBAN key agreement and elaborating upon the desired security, privacy and functional objectives.

H. Khan
Information Security Group, Royal Holloway, University of London
E-mail: Haibat.Khan.2016@live.rhul.ac.uk
ORCID: https://orcid.org/0000-0002-2948-4964

B. Dowling
Information Security Group, Royal Holloway, University of London
E-mail: Benjamin.Dowling@rhul.ac.uk

K. M. Martin
Information Security Group, Royal Holloway, University of London
E-mail: Keith.Martin@rhul.ac.uk

1.1 Network and Adversary Model

We begin by describing a system model suitable for the deployment scenarios of WBANs. In this model, a System Administrator (SA) initializes the network. The network is composed of three types of nodes; a Hub Node (HN), Intermediary Nodes (IN) and Normal Nodes (N). As the HN

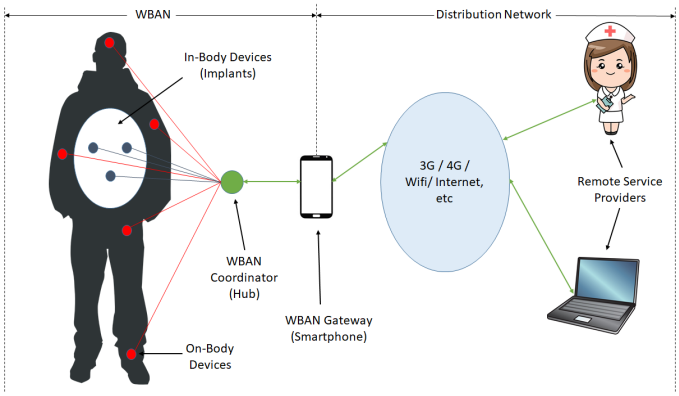

Fig. 1: Generic architecture of a typical WBAN

is usually a resourceful device with better hardware protection mechanisms in place, we assume it to be trusted and its long term secret *Master Key* to be protected. As the role of *HN* is usually undertaken by a modern smart phone in a generic WBAN, this argument is supported aptly by the real world example of the *FBI-Apple encryption dispute* [2] where even for resourceful parties like government agencies it is not easy to crack into a smart phone. Normal nodes *N* are resource constrained and their transmission range is assumed to be limited; in particular, they are not always able to communicate directly with *HN*. Intermediary nodes *IN* are also located in and around the body but, at a particular time instance, are in direct communication with both *N* and *HN*, thus acting as intermediary nodes for the purpose of relaying traffic between *HN* and *N* when required. We assume a Dolev-Yao [13] adversary $\mathcal{A}$ who can listen, modify and synthesize any message of his choice in this model.

1.2 Desired Objectives

The security of traffic in IEEE Std 802.15.6 is protected using authenticated encryption, which requires the establishment of symmetric session keys. The procedure for agreeing these keys is thus critical to the overall security and privacy of a WBAN. Next we list down the requisite properties (and where required the associated rationale) of a *Privacy-Preserving Key Agreement (PPKA)* protocol to be executed between a node *N* and *HN*:

1.2.1 Security Properties

Mutual Entity Authentication. Entity authentication is the process by which one entity (the verifier) is assured of the identity of a second entity (the claimant) [27]. The PPKA should provision mutual entity authentication between *N* and *HN*.

Mutual "Implicit" Key Authentication. The assurance that only a particularly identified other party may possibly know the negotiated key [27]. Mutual "implicit" key authentication is required between *N* and *HN*.

Known Key Security. An adversary compromising a session key in a single session should not impose any threat to the session key security in any other sessions.

Key Randomness. The assurance that any successful key agreement should output a uniformly distributed session key among the set of all possible session keys [26].

Partial Forward Secrecy. The compromise of the long-term secret of a node *N* should not enable an adversary to compromise previously established session keys of that node. *Partial Forward Secrecy (PrFS)* is crucial as client nodes (unlike *HN*) in typical WBAN deployment scenarios are not tamper-proof and their internal storage can be accessed by an adversary easily. Note that as already explained earlier in Section 1.1, we do not consider the compromise of the long term secret of *HN*. This enables us to consider a more pragmatic version of forward secrecy for WBANs. PrFS is a well documented [6] and discussed [5,9,10,24] security notion for key exchange protocols which considers the compromise of the long-term keys of a subset of protocol participants. We remark that PrFS is distinct from the related notion of *Weak Forward Secrecy (WFS)* [17] where the concerned adversary is a passive one. PrFS considers an active adversary.

Key Compromise Impersonation (KCI) Resilience. Suppose *N*'s long term secret gets disclosed. Clearly an adversary that knows this value can now impersonate *N*, since it is precisely this value that identifies *N*. However, it is highly desirable that this loss should not enable an adversary to impersonate other entities to *N* [19]. Consider the scenario where a cardiac pacemaker is part of a WBAN deployed upon a chronic patient by a hospital for remote administration and monitoring purposes. The leakage of the pacemaker's long term secret should not enable the adversary to issue "stop" commands to the pacemaker by impersonating as the hospital administrator. Such a case could potentially lead to a life threatening situation.

Replay Prevention. An adversary should not be able to successfully replay previously captured copies of legitimate messages between the protocol participants.

Desynchronization Resistance. If the authentication parameters get updated during the protocol execution, then usually the participants need to have the same updated values at the end of a protocol run. Otherwise, they will not authenticate each other in later sessions and we say they have been desynchronized. In a desynchronization attack, the adversary forces the protocol participants to update their authentication parameters to different values. A PPKA needs to be resistant to these types of attacks.

1.2.2 Privacy Properties

We focus on two privacy aspects:

Node Anonymity. An adversary $\mathcal{A}$, who is observing all communications, should not be able to learn the identity of any node N who is participating in a PPKA protocol with HN. The privacy attribute of anonymity is a necessity for typical application scenarios of WBANs, such as healthcare and military.

Session Unlinkability. An adversary $\mathcal{A}$, who is observing communications, should not be able to link one successfully executed PPKA session of node N to another successfully completed session of the same node. Session unlinkability is imperative in addition to anonymity. Although the PPKA sessions could be anonymous, if the adversary is able to link various PPKA sessions and group them together then $\mathcal{A}$ would be able to attribute a group to a particular node with high probability, due to his knowledge of the operations of the WBAN. For example, consider a medical WBAN in which a pacemaker is supposed to communicate with the remote healthcare providers every five minutes, while the body temperature sensor communicates only three times per day.

1.2.3 Functional Requirements

Support for Multi-Hop Communication. As discussed in Section 1.1, depending upon the network topology, nodes would either be communicating directly with the Hub Node HN or via an Intermediary Node IN. Therefore, the PPKA protocol should be designed to be suitable for both single-hop and two-hop communication modes of [3].

Energy Consumption. As nodes in a WBAN are severely energy constrained, the PPKA protocol needs to be minimalistic in terms of computation, communication and storage overhead. Energy consumption in WBANs is dominated by radio communications [11], which mainly depends on the number of bits to be transmitted within the network. Consequently, the PPKA protocol should be designed such that the number of bits to be exchanged between the protocol participants and the computational overhead for nodes N should be minimal.

Stateless HN. HN is the consistent nucleus of the network whose lack of accessibility will have devastating effects on the complete WBAN. As the network topology in WBANs is dynamic where client nodes join and leave the network on a frequent basis; it is imperative for HN's accessibility that it be independent of such dynamism. Consequently, an important requirement is that the PPKA protocol should not require HN to maintain a state of the WBAN nodes.

1.3 Design Principles

Offloading of Expensive Operations. As nodes in a WBAN are resource constrained, it makes sense to offload energy-expensive operations to more resourceful entities such as SA and HN. An example of this is discussed in more detail in Section 4.2.

Minimizing the Implementation Footprint. Ideally, the proposed solution should not introduce new cryptographic primitives as this will adversely affect the implementation footprint (hardware and memory). Specifically, we aim to use the already specified block cipher function in [3] for achieving the various security and privacy objectives. A more detailed discussion is given in Section 4.2.

Reducing Management Costs. A PPKA solution should not place management costs on the WBAN nodes after the network initialization. Consider the situation where a third party wants to add its node (for example a fitness tracker) to an already deployed WBAN. The third party should be able to contact the SA, who (after registration of the new node) would dispatch it to the WBAN owner, who begins using the new device upon receipt. Note that all this was done without interacting with the currently operational WBAN.

1.4 Related Work

Toorani [28] discovered various security weaknesses in the key agreement methods of IEEE Std 802.15.6, all of which were susceptible to Key Compromise Impersonation attacks as well as attacks on forward secrecy. Wang and Zhang [30] proposed a key agreement scheme for WBANs that claimed to provide anonymity and unlinkability in addition to the requisite security guarantees. However, Jiang et al. [15] show that [30] is vulnerable to client impersonation attack and thus lacks mutual authentication. They proposed an authenticated key agreement scheme which rectified this flaw. However, their scheme was based on computing bilinear pairings; which is not suitable for deployment in resource constrained WBANs. To avoid the overhead of managing public-key certificates, He et al. proposed a certificateless authentication scheme [14], which provides anonymity and unlinkability. However, the computation and communication overheads associated with their scheme also render it unsuitable for WBAN deployment. Recently, Li et al. [21] presented an authenticated key agreement scheme based only upon symmetric cryptographic primitives. This is an attractive proposal since there is no requirement of any additional infrastructure and the associated computation and communication overheads are negligible. The authors claimed that this scheme achieved almost all of the security and privacy objectives defined in Section 1.2.

1.5 Contributions and Paper Organization

Previous Version. This manuscript is full version of the paper presented at IEEE TrustCom, 2018 [16]. The main differences from the conference version are as follows: Firstly, the list of required security objectives is more comprehensive after inclusion of PrFS and KCI resilience. Moreover, the introduction section has been further enhanced by addition of the design principles of the proposed protocols. Furthermore, a new section (Section 4) providing discussion about various aspects of the proposed protocols is also part of this manuscript. This manuscript successfully answers the open question put forward in [16] regarding the feasibility of a privacy-preserving authenticated key agreement protocol for IEEE Std 802.15.6 offering PrFS and KCI resilience without any dependence on public key cryptography. The second key agreement protocol (termed PPKA2 within this manuscript) is one such protocol satisfying all the requisite security and privacy requirements. Further additional contributions of this manuscript are as below:

- We enhance our previous analysis [16] of [21] which, in addition to showing that Li et al.'s scheme does not provide session unlinkability and forward secrecy, also exhibits its vulnerability to KCI attacks.
- In addition to the key agreement protocol (PPKA-1) proposed in [16], which provided session unlinkability and resolved the privacy flaws found in [21], we present another protocol (PPKA-2) that additionally provisions PrFS and KCI resilience. Table 1 lists the security and privacy features provisioned by each protocol.
- We develop a formal security and privacy model in an appropriate complexity-theoretic framework and prove the proposed protocols secure in this model.

Table 1: Comparison of Security and Privacy Features

Security/Privacy Feature	Li et al.	PPKA-1	PPKA-2
PrFS	✗	✗	✓
KCI Resilience	✗	✗	✓
Session Unlinkability	✗	✓	✓
Anonymity	✓	✓	✓

The remainder of this paper is organized as follows:

- Section 2, provides an overview and analysis of a WBAN key agreement scheme proposed in [21].
- The proposed protocols are detailed in Section 3.
- Section 4 discusses pertinent aspects of the proposed protocols.
- Sections 5 and 6 explain the formal security model and the associated analysis, respectively.

Table 2: Notations used in [21]

Symbol	Description
$h(.)$	Cryptographic hash function
(a,b)	Concatenation of a and b
$\oplus$	Bitwise XOR operation k
SA	System Administrator (initializes the WBAN)
N	Normal Node
HN	Hub Node
IN	Intermediary Node
id_N	Long term secret/identity of node N
id'_{IN}	Relay identity of node IN
tid_N	Temporary identity of node N
k_{HN}	Long term master secret key of HN
k_N, f_N	Temporary secret parameters chosen by HN/SA
r_N	Temporary secret parameter chosen by N
a_N, b_N	Authentication parameters stored in N
x_N, y_N	Auxiliary authentication parameters
α, β, η, μ	Authentication parameters computed by HN
k_S	Shared session key
t_N	Timestamp generated by node N
$X \rightarrow Y : Z$	Entity X sends message Z to entity Y

- Finally, Section 7 provides future research directions and concludes the paper.

2 Li et al.'s Scheme

In this section we present an overview and analysis of Li et al.'s scheme [21]. For ease of comparison we use the same notation (details in Table 2) as in [21].

2.1 The Key Agreement Protocol

Li et al.'s PPKA protocol between the Hub node (HN) and a node (N) consists of three phases. For a pictorial overview of the protocol see Fig. 2.

2.1.1 Initialization Phase

The (SA) generates a master secret key k_{HN} and stores it in HN.

2.1.2 Registration Phase

The SA generates a unique secret identity id_N for node N. It then randomly chooses the temporary secret parameter k_N and calculates $a_N = id_N \oplus h(k_{HN}, k_N)$ and $b_N = k_{HN} \oplus a_N \oplus k_N$. A unique relay identity id'_{IN} for the intermediary node (IN) is chosen and the parameters $\langle id_N, a_N, b_N \rangle$ and $\langle id'_{IN} \rangle$ are stored in N and IN respectively, while id'_{IN} is stored by HN as the identity of IN when communicating in relay mode.

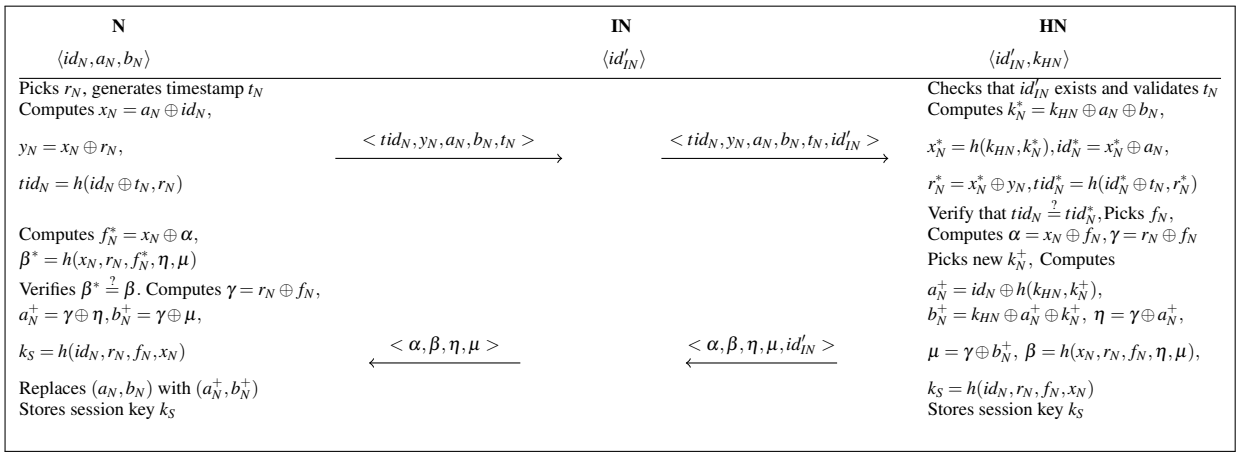

Fig. 2: Li et al.'s Protocol

2.1.3 Authentication Phase

We can think of the authentication phase of Li et al.'s scheme as a two-pass protocol. The individual steps are outlined below:

Step 1: $N \rightarrow IN : \langle tid_N, y_N, a_N, b_N, t_N \rangle$. N picks a random r_N and creates timestamp t_N. Then it computes $x_N = a_N \oplus id_N$, $y_N = x_N \oplus r_N$ and $tid_N = h(id_N \oplus t_N, r_N)$ and forwards the tuple $\langle tid_N, y_N, a_N, b_N, t_N \rangle$ to IN.

Step 2: $IN \rightarrow HN : \langle tid_N, y_N, a_N, b_N, t_N, id'_{IN} \rangle$. IN adds its relay identity id'_{IN} to the tuple and forwards it to HN. Note that IN when operating in relay mode uses id'_{IN} not id_{IN}.

Step 3: $HN \rightarrow IN : \langle \alpha, \beta, \eta, \mu, id'_{IN} \rangle$. After receiving the parameters from IN, HN verifies the relay identity id'_{IN} from its database and substantiates the validity of the timestamp t_N. Upon success of these checks, it computes $k^*_N = k_{HN} \oplus a_N \oplus b_N$, $x^*_N = h(k_{HN}, k^*_N)$, $id^*_N = x^*_N \oplus a_N$, $r^*_N = x^*_N \oplus y_N$ and $tid^*_N = h(id^*_N \oplus t_N, r^*_N)$. It then verifies whether $tid_N \stackrel{?}{=} tid^*_N$. Then, a random f_N is chosen and $\alpha = x_N \oplus f_N$ and $\gamma = r_N \oplus f_N$ are computed. Then a new k^+_N is picked and $a^+_N = id_N \oplus h(k_{HN}, k^+_N)$, $b^+_N = k_{HN} \oplus a^+_N \oplus k^+_N$, $\eta = \gamma \oplus a^+_N$, $\mu = \gamma \oplus b^+_N$, $\beta = h(x_N, r_N, f_N, \eta, \mu)$ are computed. The shared session key is computed as $k_S = h(id_N, r_N, f_N, x_N)$ and is stored in memory. Finally, HN forwards the tuple $\langle \alpha, \beta, \eta, \mu, id'_{IN} \rangle$ to IN.

Step 4: $IN \rightarrow N : \langle \alpha, \beta, \eta, \mu \rangle$. IN removes the relay identity id'_{IN} from the received tuple and forwards $\langle \alpha, \beta, \eta, \mu \rangle$ to N.

Step 5: Upon receipt of the response from IN, N computes $f^*_N = x_N \oplus \alpha$ and $\beta^* = h(x_N, r_N, f^*_N, \eta, \mu)$ and verifies that $\beta^* \stackrel{?}{=} \beta$. If true, N computes $\gamma = r_N \oplus f_N$, $a^+_N = \gamma \oplus \eta$ and $b^+_N = \gamma \oplus \mu$. The shared session key k_S is computed as $h(id_N, r_N, f_N, x_N)$ and the authentication parameters (a_N, b_N) are replaced by (a^+_N, b^+_N).

2.2 Analysis of the Li et al.'s Scheme

In this section we discuss vulnerabilities and attacks on the security of the Li et al. scheme.

2.2.1 Security Analysis

In addition to provisioning of mutual "direct" authentication [12], Li et al.'s scheme fulfills all the security criteria as defined in Section 1.2 except KCI resilience and PrFS. Moreover, the scheme also protects the master secret (k_{HN}) in the event of compromise of various nodes of the WBAN. For sake of brevity, we will restrict our security analysis to highlight only the vulnerabilities of Li et al.'s scheme.

Discussion about Forward Secrecy. Li et al. claimed a forward security property of their scheme. Their definition of forward secrecy varies from the generally accepted one. According to Li et al., the goal of forward secrecy is to protect other (past / future) session keys in the event of compromise of the current session key k_S. However, the conventional definition of forward secrecy states that in the event of compromise of the long term secrets of the protocol participant(s), an adversary should not be able to obtain any of the past session keys [22]. While Li et al.'s scheme is forward secure according to their own definition, it is not forward secure in a conventional sense.

KCI Attack. We demonstrate a KCI attack on Li et al.'s scheme. $\mathcal{A}$ observes the first pass of the protocol and notes the message contents. As the value id_N is known to $\mathcal{A}$, he calculates the following values as follows:

$$x_N = a_N \oplus id_N; \quad r_N = y_N \oplus x_N.$$

$\mathcal{A}$ chooses a random f_N and calculates $\alpha = f_N \oplus x_N$. $\mathcal{A}$ then chooses arbitrary values of η and μ and calculates β as:

$$\beta = h(x_N, r_N, f_N, \eta, \mu).$$

Finally, $\mathcal{A}$ sends out the tuple $\langle \alpha, \beta, \eta, \mu \rangle$ back to node N. N cannot detect this KCI attack as N's computed value β is the same as in the received tuple. As a result, node N would be sharing the session key $k_S = h(id_N, r_N, f_N, x_N)$ with $\mathcal{A}$, incorrectly believing itself to be sharing k_S with HN.

2.2.2 Privacy Analysis

The Anonymity Dilemma. It is known apriori to the attacker that all nodes ultimately communicate with HN. As the node identifier id_N is always masked (by taking an XOR of it with a fresh random value), anonymity in Li et al.'s protocol is preserved from "direct" privacy attacks. However, now consider the situation depicted in Fig. 3, where an intermediary node IN is providing the relaying service to various nodes N. In the second pass of Li et al.'s scheme, it is not

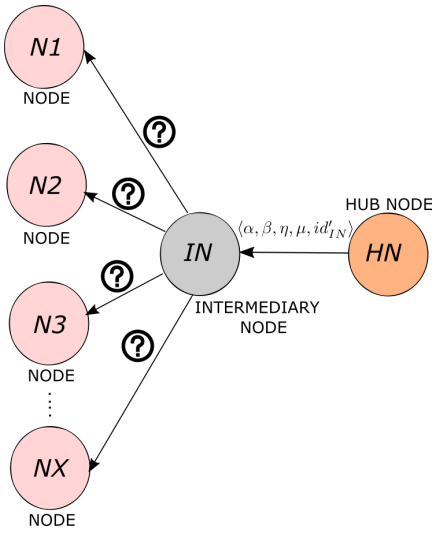

Fig. 3: The privacy dilemma of Li et al.'s scheme

clear how the intermediary node IN would be able to identify the original node N out of the "anonymity set" [25] for onward forwarding of the tuple $\langle \alpha, \beta, \eta, \mu \rangle$ received from HN. One naive way to resolve this is to allow IN to broadcast the second pass of protocol for all nodes. However, this approach is unsuitable for already energy-constrained WBAN nodes as they will need to perform additional communication (radio reception) and computational steps for each transmission.

Session Unlinkability. While Li et al. claim their scheme provides session unlinkability, we show this to be untrue. We highlight a weakness in Li et al.'s key agreement protocol, which allows a passive attacker to easily link two or more sessions of the same node N. The attack proceeds as follows:

Session # 1. Suppose that a run of Li et al.'s key agreement protocol is carried out between node N and HN. A passive attacker $\mathcal{A}$ observes the contents of the messages being exchanged. From Step 1 of Section 2.1.3, $\mathcal{A}$ records the value $y_N = x_N \oplus r_N$. Then, from Step 3 of Sec 2.1.3, $\mathcal{A}$ records $\alpha = x_N \oplus f_N$. Now, $\mathcal{A}$ obtains the value $\gamma = r_N \oplus f_N = \alpha \oplus y_N$. Further, $\mathcal{A}$ records the values η and μ from Step 3 of Section 2.1.3 and uses γ to compute:

$$a_N^+ = \gamma \oplus \eta; \quad b_N^+ = \gamma \oplus \mu.$$

Session # 2. Now, $\mathcal{A}$ observes key exchange protocol sessions between various nodes and HN. $\mathcal{A}$ compares the values of the parameters a_N and b_N from Step 1 of the protocol with the saved values of a_N^+ and b_N^+. When $\mathcal{A}$ finds a match, $\mathcal{A}$ concludes with almost certainty that another key exchange session has been initiated by the same node N. This is correct because node N uses the updated authentication parameters a_N^+ and b_N^+ in its next run of the protocol. In this way, $\mathcal{A}$ can track and link sessions of node N, demonstrating that Li et al.'s scheme does not achieve session unlinkability.

Table 3: Overheads associated with Li et al.'s scheme

Index	Node N	Hub Node HN
Computation Overhead	$3h+7\oplus$	$5h+12\oplus$
Communication Overhead	$5B$ bits	$4B+16$ bits
Storage Overhead	$3B$ bits	$(B+16m)$ bits

2.2.3 Functional Requirements

Li et al.'s scheme can be easily adapted for direct communication between N and HN without the involvement of IN. Since this scheme employs only symmetric cryptographic primitives, it is extremely efficient from a computation, communication and storage overhead perspective and there is no requirement of any additional network infrastructure. Assuming a hash function with a digest length of B bits and 16 bit intermediary node IDs (i.e. id'_{IN}), Table 3 highlights the communication, computation and storage overhead of Li et al.'s scheme. In this table, h denotes one hash operation, $\oplus$ denotes an XOR operation and m denotes the number of intermediary nodes in the WBAN. Note that, contrary to the assumption made by Li et al. in Section 5.4 of [21] about the arbitrary length of the timestamp field, it is implicitly the same length as the hash function digest because, as described earlier in Section 2.1.3, $tid_N = h(id_N \oplus t_N, r_N)$. This is not commensurate with the length of the timestamp field as defined in IEEE Std 802.15.6, which is three octets or 24 bits. Regarding state maintenance by HN, in case of [21], HN needs to maintain states concerning the relay nodes IN, which is an undesirable feature as already explained in Section 1.2.3.

Table 4: Detail of additional symbols

Symbol	Description
id'_N	Session identity chosen randomly by N
z_N	Security parameter stored in memory of N by HN/SA
$\mathsf{Enc}(k,m)$	Encryption of message m under symmetric key k
$\mathsf{Dec}(k,c)$	Decryption of ciphertext c under symmetric key k
γ	Additional authentication parameter computed by HN

3 Our PPKA Protocols

In this section we propose two PPKA protocols which rectify the problems highlighted in Section 2.2. While devising these PPKA protocols, we have tried to preserve the original elegance, simplicity and efficiency of the scheme in [21]. The first PPKA protocol addresses the privacy flaws of unlinkability and anonymity dilemma faced by IN (Section 2.2.2) in Li et al.'s scheme. The second protocol, additionally provides PrFS and KCI resilience (in case of compromise of the long term secret of node N). Note that though in our protocols the intermediary node IN is not an active participant from a cryptographic standpoint (this was a conscious design consideration), we have included IN in our protocol description for verification of the resolution of the anonymity dilemma of IN. Detail of additional notation used in our PPKA protocols is given in Table 4.

3.1 PPKA Protocol 1

The phases of PPKA Protocol 1 are separated into three distinct phases. An *Initialization Phase*, that generates the long-term secret values of the Hub Node HN. A *Registration Phase*, that generates the long-term values of the end-nodes N and stores them with HN. Finally, an *Authentication Phase* where the nodes N and HN generate an authenticated shared secret key, and update the authentication parameters.

3.1.1 Initialization Phase

This is identical to the Initialization Phase as presented in [21]. Specifically, the (SA) generates a master secret key k_{HN} and stores it in HN.

3.1.2 Registration Phase

The intermediary node (IN) is not provided with a relay identity id'_{IN}. Parameters $\langle id_N, a_N, b_N \rangle$ are stored in N.

3.1.3 Authentication Phase

The various steps of the authentication phase are depicted in Fig. 4 and are as follows:

Step 1: $N \rightarrow IN : \langle tid_N, y_N, a_N, b_N, t_N, id'_N \rangle$. N picks a random r_N and creates timestamp t_N. It then computes $x_N = a_N \oplus id_N$, $y_N = x_N \oplus r_N$. It further picks a random pseudonym id'_N to be used as a temporary identifier for this session only, calculates $tid_N = h(id_N, id'_N, t_N, r_N)$ and sets the "Relay Field" of the underlying "MAC Header" to value 1, according to sub-clause 6.10 of [3].
Step 2: $IN \rightarrow HN : \langle tid_N, y_N, a_N, b_N, t_N, id'_N \rangle$. IN checks the value of "Relay Field" and forwards the tuple to HN.
Step 3: $HN \rightarrow IN : \langle \alpha, \beta, \eta, \mu, id'_N \rangle$. After receipt of the tuple from IN, HN verifies the validity of the timestamp t_N. Upon success of this check, it computes $k^*_N = k_{HN} \oplus a_N \oplus b_N$, $x^*_N = h(k_{HN}, k^*_N)$, $id^*_N = x^*_N \oplus a_N$, $r^*_N = x^*_N \oplus y_N$ and $tid^*_N = h(id^*_N, id'_N, t_N, r^*_N)$. It then verifies whether $tid_N \stackrel{?}{=} tid^*_N$. Then, a random f_N is chosen and $\alpha = x_N \oplus f_N$, $\gamma = r_N \oplus f_N \oplus h(id_N, t_N)$ and $\gamma' = r_N \oplus f_N \oplus h(id_N, t_N, r_N, id'_N)$ are computed. Then a new k^+_N is picked and $a^+_N = id_N \oplus h(k_{HN}, k^+_N)$, $b^+_N = k_{HN} \oplus a^+_N \oplus k^+_N$, $\eta = \gamma \oplus a^+_N$, $\mu = \gamma' \oplus b^+_N$, $\beta = h(x_N, r_N, f_N, \eta, \mu, id'_N)$ are computed. Finally, the shared session key $k_S = h(id_N, r_N, f_N, x_N)$ is computed and stored in memory, and the value of the underlying "Relay Field" is set to 1.
Step 4: $IN \rightarrow N : \langle \alpha, \beta, \eta, \mu, id'_N \rangle$. IN checks the "Relay Field" of the message received from the Hub node. If "Relay Field" value is set to 1, then it notes the identifier id'_N received in the tuple for onward forwarding of the tuple to node N.
Step 5: Upon receiving a response from IN, N computes $f^*_N = x_N \oplus \alpha$ and $\beta^* = h(x_N, r_N, f^*_N, \eta, \mu, id'_N)$ and verifies that $\beta^* \stackrel{?}{=} \beta$. If so, N computes $\gamma = r_N \oplus f_N \oplus h(id_N, t_N)$, $\gamma' = r_N \oplus f_N \oplus h(id_N, t_N, r_N, id'_N)$, $a^+_N = \gamma \oplus \eta$ and $b^+_N = \gamma' \oplus \mu$. The shared session key k_S is computed as $h(id_N, r_N, f_N, x_N)$, and the authentication parameters (a_N, b_N) are updated by being replaced with (a^+_N, b^+_N).

3.2 PPKA Protocol 2

The second PPKA protocol is structurally similar to PPKA Protocol 1, with three phases and similar goals. We describe the execution of PPKA Protocol 2 below.

3.2.1 Initialization Phase

This phase is unchanged from [21].

3.2.2 Registration Phase

The registration phase is mostly identical to PPKA Protocol 1. However, SA additionally computes $z_N = h(k_{HN}, id_N, k_N)$. Parameters $\langle id_N, a_N, b_N, z_N \rangle$ are stored in N.

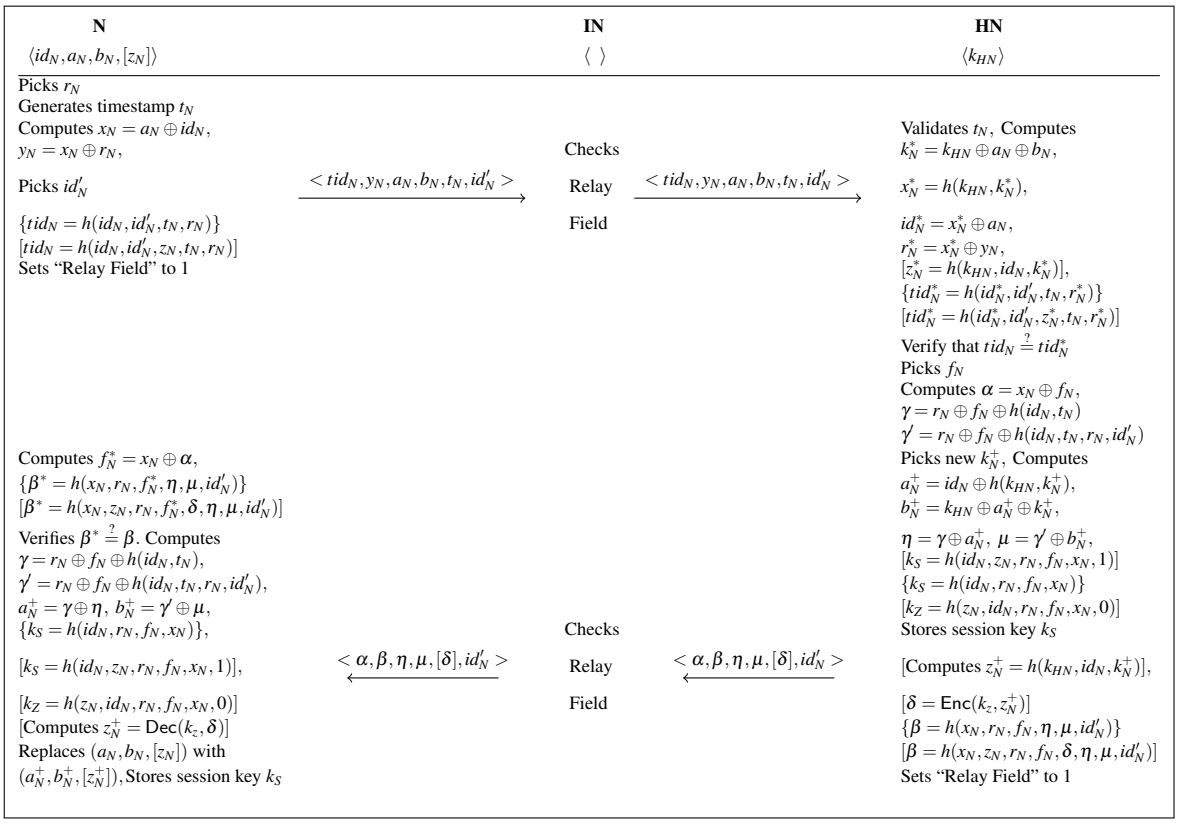

Fig. 4: PPKA Protocols 1 and 2 (Values / operations defined within curly brackets { } are valid only for PPKA 1, while those defined within square brackets [ ] are valid only for PPKA 2)

3.2.3 Authentication Phase

The authentication phase of PPKA Protocol 2 is depicted in Fig. 4 and detailed as follows:

Step 1: $N \rightarrow IN : \langle tid_N, y_N, a_N, b_N, t_N, id'_N \rangle$. This is identical to Step 1 in PPKA 1 except that the value of tid_N is calculated as $h(id_N, id'_N, z_N, t_N, r_N)$.

Step 2: $IN \rightarrow HN : \langle tid_N, y_N, a_N, b_N, t_N, id'_N \rangle$. This is identical to Step 2 in PPKA 1

Step 3: $HN \rightarrow IN : \langle \alpha, \beta, \eta, \mu, \delta, id'_N \rangle$. After receipt of the tuple from *IN*, *HN* proceeds identically to Step 3 in PPKA 1. Additionally z^*_N is calculated as $h(k_{HN}, id_N, k_N)$ and tid^*_N as $h(id^*_N, id'_N, z^*_N, t_N, r^*_N)$. It then verifies whether $tid_N \stackrel{?}{=} tid^*_N$. Then, α, η and μ are computed as in PPKA 1. k_S is computed as $h(id_N, z_N, r_N, f_N, x_N, 1)$ while an additional key k_Z is computed as $h(z_N, id_N, r_N, f_N, x_N, 0)$. *HN* then computes $z^+_N = h(k_{HN}, id_N, k^+_N)$ and encrypt it with k_z as $\delta = \mathsf{Enc}(k_z, z^+_N)$. Lastly, β is calculated as $h(x_N, z_N, r_N, f_N, \delta, \eta, \mu, id'_N)$.

Step 4: $IN \rightarrow N : \langle \alpha, \beta, \eta, \mu, \delta, id'_N \rangle$. This is identical to Step 4 in PPKA 1

Step 5: This is identical to Step 5 in PPKA 1, except that β^* is calculated as $h(x_N, z_N, r_N, f^*_N, \delta, \eta, \mu, id'_N)$ and the shared session key k_S is computed as $h(id_N, z_N, r_N, f_N, x_N, 1)$. Additionally, node *N* decrypts $z^+_N = \mathsf{Dec}(k_z, \delta)$ and replaces z_N with z^+_N.

4 Discussion

4.1 Why a Bespoke Solution?

If we consider the scenario of direct communication between *N* and *HN* (without the involvement of *IN*), at first glance, it seems to be similar to that of RFID; where a *tag* needs to be authenticated in a secure and private manner by the *reader*. However, there is a fundamental distinction between the two scenarios. As discussed in Section 1.2.3, in our case the *HN* does not maintain any state about the network nodes and is oblivious to the identity management of the network, while in the RFID setting, the *reader* has access to the back-end database server(s) which maintain nodes' status in the RFID network. This means that, in the case of RFID, *SA* needs to update the status at the back-end servers whenever it introduces a new node or removes an old one from the system. As explained in Section 1.2.3, this is problematic for WBANs.

4.2 Random Number Generation on WBAN Nodes

A WBAN Cryptographically Secure Pseudo Random Number Generator (CSPRNG) needs to be computationally inexpensive and there should be no requirement for entropy collection from environmental resources, as this would entail extra communication. We recommend the approach outlined in [18]. During the "Registration Phase", SA can allocate each node N with a unique (randomly chosen) secret key K. Thereafter, node N can encrypt the sequence $\{0,1,2,3,...\}$ under key K using AES (already available for message security purposes) as the block cipher. This arrangement can securely generate 2^{60} bytes without the need for re-seeding the key K.

4.3 Post-Quantum Significance

Given recent progress towards achieving practical universal quantum computers [8], it is imperative that proposals for any standard should also cater for this future threat. Our PPKA protocols avoid any public key cryptography and are thus well suited to post-quantum deployment scenarios.

4.4 Why Timestamps?

Timestamps are generally avoided in key agreement protocols as they present various practical problems, such as the need for a reliable source of time. Our setting does not face these difficulties, as a comprehensive mechanism already exists in Clause 6.11 of [3] which provisions for HN to act as the central time source for the WBAN and regularly broadcasts time-synchronization beacons.

5 Security Model

We now introduce our security models for the analysis of privacy-preserving key agreement (PPKA) protocols. Our first security experiment is based on standard key-exchange models in the tradition of Bellare-Rogaway [4] key indistinguishability games. This allows our model to easily capture known key secrecy, as well as generically capture key randomness notions, since our adversary is tasked merely with the goal of distinguishing the targeted session key from a random session key from the same distribution. Our second security experiment allows us to capture privacy notions of sessions, by challenging an adversary to determine which of two previously selected nodes ran a given protocol execution. Our cleanness predicates (see Section 5.4) allows us to model KCI attacks by allowing the adversary to reveal the long-term key of the node running the PPKA protocol, as well as the notions of partial forward secrecy. We begin by describing the execution environment for our security frameworks.

5.1 Execution Environment

Consider an experiment $\mathsf{Exp}^{\mathsf{PPKA\text{-}IND}}_{\Pi,n_N,n_S,\mathcal{A}}(\lambda)$ played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. $\mathcal{C}$ maintains a single node HN, running a number of instances of the PPKA protocol Π, and a set of (up to) n_N nodes $N_1,\ldots,N_{n_N}$ (representing nodes communicating with the hub node HN), each potentially running one stage of (up to) n_S consecutive stages of Π. The PPKA protocol Π is represented as a tuple of algorithms $\Pi = (\mathsf{HKeyGen},\mathsf{HF},\mathsf{NKeyGen},\mathsf{NF},\mathsf{StateGen})$. We abuse notation and use π^{stid}_{id} to refer to both the identifier of the $stid$-th stage of Π being run by node N_{id} and the collection of per-session variables maintained for this stage. We describe the algorithms below:

$\Pi.\mathsf{HKeyGen}(\lambda) \xrightarrow{\$} (k_{HN})$ is a probabilistic symmetric key generation algorithm taking as input a security parameter λ and outputting a long-term hub node secret key (k_{HN}).

$\Pi.\mathsf{HF}(\lambda,k_{HN},m) \xrightarrow{\$} (m')$ is a (potentially) probabilistic algorithm that takes a security parameter λ, the long-term key of the hub node k_{HN}, and an arbitrary bit string $m \in \{0,1\}^* \cup \{\emptyset\}$, and outputs a response $m' \in \{0,1\}^* \cup \{\emptyset\}$ and an updated per-session state π'.

$\Pi.\mathsf{NKeyGen}(\lambda) \xrightarrow{\$} (ltk)$ is a probabilistic symmetric key generation algorithm taking as input a security parameter λ and outputting a long-term hub node secret key (ltk). Note that in our proposed PPKA protocols, we denote this long-term secret key with id_N.

$\Pi.\mathsf{NF}(\lambda,\pi,m) \xrightarrow{\$} (m',\pi')$ is a probabilistic algorithm taking a security parameter λ, the set of per-session variables π and an arbitrary bit string $m \in \{0,1\}^* \cup \{\emptyset\}$, and outputs a response $m' \in \{0,1\}^* \cup \{\emptyset\}$ and an updated per-session state π'.

$\Pi.\mathsf{StateGen}(\lambda,k_{HN},ltk) \xrightarrow{\$} (psstate)$ is a probabilistic symmetric key generation algorithm taking as input a security parameter λ and the long-term secret keys of the hub node and the "normal" node, outputting secret state information for node N $(psstate)$. In PPKA Protocol 1, this per-stage secret state is $\langle a_N,b_N\rangle$. In PPKA Protocol 2, this is $\langle a_N,b_N,z_N\rangle$.

$\Pi.\mathsf{StateUpdate}(\lambda,\pi) \xrightarrow{\$} (psstate)$ is a probabilistic symmetric key generation algorithm taking as input a security parameter λ and a set of per-session variables, outputting the next stage's per-stage secret state $(psstate)$ for node N.

The experiment begins with $\mathcal{C}$ running $\Pi.\mathsf{HKeyGen}$ once to generate a long-term secret key for the hub node (k_{HN}), and randomly sampling a bit $b \in \{0,1\}$. $\mathcal{A}$ then interacts with $\mathcal{C}$ via the queries listed in Section 5.2, eventually terminating and outputting a guess bit b' of $\mathcal{C}$'s bit b. $\mathcal{A}$ wins the key-indistinguishability game if $b' = b$ *and* the session π^{stid}_{id} such

that $\mathcal{A}$ issued $\textbf{Test}(id, stid)$ satisfies the cleanness predicate clean, which we discuss in Section 5.4. Each session maintains the following set of per-session variables:

- $ltk \in \{0,1\}^{\lambda}$ - the long-term symmetric-secret of N_{id}.
- $id \in \{1,\ldots,n_N\}$ - the index of the node N_{id}.
- $m_s \in \{0,1\}^* \cup \{\perp\}$ - the concatenation of messages sent by the node, initialised by $\perp$.
- $m_r \in \{0,1\}^* \cup \{\perp\}$ - the concatenation of messages received by the node, initialised by $\perp$.
- $psstate \in \{0,1\}^* \cup \{\perp\}$ - the per-stage secret state of the node, initialised by $\perp$.
- $sk \in \{0,1\}^* \cup \{\perp\}$ - the session key, initialised by $\perp$.
- $stid \in \{1,\ldots,n_S\}$ - the index of the most recently completed stage, initialised by 1 and increased monotonically.
- $\alpha \in \{\texttt{active}, \texttt{accept}, \perp\}$ - the current status of the node, initialised by $\perp$.

Finally, the challenger manages the following set of registers, which indicate $\mathcal{A}$'s compromise of secrets:

- long-term symmetric keys $\{\mathsf{LSKflag}_1, \ldots, \mathsf{LSKflag}_{n_N}\}$, where $\mathsf{LSKflag}_i \in \{\texttt{corrupt}, \texttt{clean}, \perp\}\ \forall\ i \in [n_N]$.
- per-stage secret state $\{\mathsf{PSSflag}_1^1, \mathsf{PSSflag}_2^1, \ldots, \mathsf{PSSflag}_{n_S}^1, \ldots \mathsf{PSSflag}_1^{n_N}, \mathsf{PSSflag}_2^{n_N}, \ldots, \mathsf{PSSflag}_{n_S}^{n_N}\}$ where $\forall i \in n_N$, $j \in n_S$, $\mathsf{PSSflag}_j^i \in \{\texttt{corrupt}, \texttt{clean}, \perp\}$.
- session keys $\{\mathsf{SKflag}_1^1, \mathsf{SKflag}_2^1, \ldots, \mathsf{SKflag}_{n_S}^1, \ldots \mathsf{SKflag}_1^{n_N}, \mathsf{SKflag}_2^{n_N}, \ldots, \mathsf{SKflag}_{n_S}^{n_N}\}$ where $\forall i \in n_N$, $j \in n_S$, $\mathsf{SKflag}_j^i \in \{\texttt{corrupt}, \texttt{clean}, \perp\}$.

5.2 Adversarial Interaction

In the game, the adversary $\mathcal{A}$ is able to communicate with the challenger and thus interact with the parties/sessions via the following set of queries:

$\textbf{Register}(\lambda) \rightarrow id$: Allows $\mathcal{A}$ to register a new node with security parameters λ and gives $\mathcal{A}$ an identifier for the node id (which we denote N_{id}). For protocols where nodes do not have a public identifier, the index of the node is given to $\mathcal{A}$.

$\textbf{NextKey}(\lambda, id) \rightarrow m$: Allows $\mathcal{A}$ to indicate that the node with public identifier id should attempt a new key agreement using (potentially) the new/updated security parameters λ. The challenger then returns any protocol messages m.

$\textbf{Corrupt}(id) \rightarrow ltk$: Allows $\mathcal{A}$ to compromise the long-term key of the node $\pi_{id}.ltk$ with public identifier id.

$\textbf{Reveal}(id, stid) \rightarrow sk$: Allows $\mathcal{A}$ to compromise the session key established between the hub node and the node N_{id} in stage $stid$. Note that $stid$ indicates the index of the session key established between the node id and the hub node. The challenger responds with the session key $\pi_{id}^{stid}.sk$.

$\textbf{StateReveal}(id, stid) \rightarrow psstate$: Allows $\mathcal{A}$ to compromise the per-stage secret state $psstate$ of the node with public identifier id. Note that $stid$ indicates the index of the stage-specific state, and the challenger responds with $\pi_{id}^{stid}.psstate$.

$\textbf{Send}(id, m) \rightarrow m'$: Allows $\mathcal{A}$ to send a message m to the node with identifier id currently running a protocol execution. Note that the node will update its per-session variables and potentially output a new message m'.

$\textbf{Test}(id, stid) \rightarrow sk$: If the node N_{id} has completed its $stid$-stage key agreement, then the challenger uses the randomly-sampled bit $b \in \{0,1\}$. If $b = 0$ the challenger responds with $\pi_{id}^{stid}.sk$, otherwise the challenger responds with a random key from the same distribution.

We now formalise the advantage of a PPT algorithm $\mathcal{A}$ in winning the PPKA key-indistinguishability game:

Definition 1 (Key Indistinguishability) Let Π be a PPKA protocol and n_N, $n_S \in \mathbb{N}$. For a given cleanness predicate clean, and a PPT algorithm $\mathcal{A}$, we define the advantage of $\mathcal{A}$ in the key-indistinguishability game to be:

$$\mathsf{Adv}^{\mathsf{PPKA\text{-}IND,clean}}_{\Pi,n_N,n_S,\mathcal{A}}(\lambda) = |2 \cdot (\Pr[\mathsf{Exp}^{\mathsf{PPKA\text{-}IND,clean}}_{\Pi,n_N,n_S,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|).$$

We say that Π is PPKA-IND-secure if, for all $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{PPKA\text{-}IND,clean}}_{\Pi,n_N,n_S,\mathcal{A}}(\lambda)$ is negligible in security parameter λ.

5.3 Session Unlinkability

The experiments for PPKA key-indistinguishability and session unlinkability are mostly identical. However, instead of using the $\textbf{Test}(id, stid)$ query, at some point $\mathcal{A}$ will stop and output (id_0, id_1). When $\mathcal{A}$ outputs (id_0, id_1), $\mathcal{C}$ runs $\textbf{NextKey}(\lambda, id_0)$ and responds to queries as before. We will refer to this as the "challenge" node. However, when $\pi_{id_0}^{stid}.\alpha \leftarrow \texttt{accept}$, $\mathcal{C}$ then refers to the random bit b sampled at the beginning of the experiment and:

- if $b = 0$, then $\mathcal{C}$ runs $\textbf{NextKey}(\lambda, id_0)$
- if $b = 1$, then $\mathcal{C}$ runs $\textbf{NextKey}(\lambda, id_1)$ instead.

$\mathcal{A}$ now uses the $\textbf{SendTest}(m)$ query to send messages to the node N_{id_b} in order to avoid trivial identification. We will refer to this as the "unnamed node". $\mathcal{A}$ at some point terminates and outputs a guess bit b'. If $b' = 0$, then $\mathcal{A}$ is indicating that the unnamed node N_{id_b} *was* linked to the challenge node N_{id_0}. If $b' = 1$, then $\mathcal{A}$ is indicating that the the unnamed node N_{id_b} *was not* linked to the challenge node N_{id_0}.

We now formalise the advantage of a PPT algorithm $\mathcal{A}$ in winning the PPKA session-unlinkability game:

Definition 2 (Session Unlinkability) Let Π be a PPKA protocol, and n_N, $n_S \in \mathbb{N}$. For a given cleanness predicate clean, and a PPT algorithm $\mathcal{A}$, we define the advantage of $\mathcal{A}$ in the session-unlinkability game to be:

$$\mathsf{Adv}^{\mathsf{PPKA\text{-}SU,clean}}_{\Pi,n_N,n_S,\mathcal{A}}(\lambda) = |2 \cdot (\Pr[\mathsf{Exp}^{\mathsf{PPKA\text{-}SU,clean}}_{\Pi,n_N,n_S,\mathcal{A}}(\lambda) = 1] - \frac{1}{2})|.$$

We say that Π is PPKA-SU-secure if, for all $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{PPKA\text{-}SU,clean}}_{\Pi,n_N,n_S,\mathcal{A}}(\lambda)$ is negligible in the security parameter λ.

5.4 Cleanness Predicates

The cleanness predicates are used in the security experiments to define the exact combination of secrets that $\mathcal{A}$ is able to compromise without trivially breaking the PPKA protocol. In order to capture key-compromise-impersonation (KCI) attacks and PrFS notions, we allow $\mathcal{A}$ to leak the long-term secret key of the "normal" nodes if $\mathcal{A}$ has not also leaked any previously established per-stage secret state. Our analysis is focused primarily on the normal nodes, and we do not allow the compromise of the hub node secrets, as all security in all stages is lost in this scenario. We additionally describe a cleanness predicate for PPKA protocols that do not achieve PrFS or KCI resilience.

Definition 3 (PrFS-KCI-clean) A session π^{stid}_{id} such that $\pi^{stid}_{id}.\alpha = \texttt{accept}$ in the PPKA-IND experiment defined in Figure 5 is PrFS-KCI-clean if $\mathsf{SKflag}^{stid}_{id} \neq \texttt{corrupt}$ and if $\mathsf{LSKflag}_{id} = \texttt{corrupt}$ then $\forall\, s \leq stid\ \mathsf{PSSflag}^{s}_{id} \neq \texttt{corrupt}$.

Definition 4 (nPrFS-clean) A session π^{stid}_{id} such that $\pi^{stid}_{id}.\alpha = \texttt{accept}$ in the PPKA-IND experiment defined in Figure 5 is nPrFS-clean if $\mathsf{SKflag}^{stid}_{id} \neq \texttt{corrupt}$.

Finally, we describe a cleanness predicate for our session-unlinkability game. It is straightforward to realise that if **Corrupt**(id_0) or **Corrupt**(id_1) were to be issued, it would trivially allow $\mathcal{A}$ to win in either of our PPKA protocols by simply reconstructing the tid_N field sent by the unnamed node. Similarly, we cannot allow the adversary to reveal the per-stage secret state for the current stage $stid$ of the unnamed node N_{id_b}.

Definition 5 (SU-clean) A session π^{stid}_{id} in the PPKA-SU experiment defined in Figure 5 is SU-clean if $\mathsf{LSKflag}_{id} \neq \texttt{corrupt}$ and $\mathsf{PSSflag}^{stid}_{id} \neq \texttt{corrupt}$.

6 Analysis of our proposed PPKA Protocols

6.1 Security and Privacy Analysis

Before we begin, we show that an adversary $\mathcal{A}$ is unable to recover the hub node secret k_{HN} (with non-negligible probability) even if $\mathcal{A}$ reveals all long-term secrets id_N of all nodes and all per-stage secret states $psstate$. In our proofs we work within the random oracle model, and $\mathcal{A}$ cannot learn anything about k_{HN} from hash outputs $h(k_{HN},X)$ (where X is any concatenation of arbitrary values). We turn to $\mathcal{A}$ attempting to learn k_{HN} that has been "blinded" through exclusive-or (XOR) operations. We give below the generic construction of messages that include k_{HN}:

- $b_N = k_{HN} \oplus k_N \oplus id_N \oplus h(k_{HN},k_N)$
- $\mu = k_{HN} \oplus k^+_N \oplus id_N \oplus h(k_{HN},k^+_N) \oplus h(id_N,t_N,r_N,id'_N) \oplus f_N \oplus r_N$

Taking μ first, we note that k^+_N (independently sampled by the hub node, uniformly-at-random, in each stage) acts as the key in a one-time-pad, perfectly hiding the long-term secret key k_{HN} of the hub node, the long-term secret key id_N of the normal node and the value $h(k_{HN},k_N)$. k^+_N is an internal value that is known only to the challenger implementing the Hub Node, as it cannot be compromised by $\mathcal{A}$ via **Reveal**, **Corrupt** or **StateReveal** queries. For b_N, we note that k_N (randomly sampled by the hub node in a previous stage) is still acting as the same key k^+_N in a one-time-pad, and thus still perfectly hiding the same message, i.e. the long-term secret key k_{HN} of the hub node, the long-term secret key id_N of the normal node and the value $h(k_{HN},k_N)$. We argue then that $\mathcal{A}$ cannot recover the hub node secret key k_{HN}. We can further conclude that an adversary that compromises fewer internal states and long-term secret keys will also be unable to recompute k_{HN}. We can continue our proof knowing that the best strategy for $\mathcal{A}$ to recover the long-term secret key of the hub node k_{HN} is to attempt to brute-force the value.

We now show that an adversary $\mathcal{A}$ that does not issue a **Corrupt**(id) query cannot recover the long-term secret key id_N of node N_{id}. As before, we note that since we instantiate the hash function as a random oracle, that the adversary cannot invert hash outputs of the form $h(id_N,X)$ (where X is some arbitrary concatenation of values) in order to learn id_N. We can now focus on the adversary attempting to learn id_N from "blinded" values by XORing them with other values. In each stage of the protocol execution, this is available to $\mathcal{A}$ in the following generic ways:

- $a_N = id_N \oplus h(k_{HN},k_N)$
- $b_N = k_{HN} \oplus k_N \oplus id_N \oplus h(k_{HN},k_N)$
- $\eta = r_N \oplus f_N \oplus id_N \oplus k_{HN} \oplus k_N \oplus h(id_N,f_N) \oplus h(k_{HN},k^+_N)$
- $\mu = r_N \oplus f_N \oplus id_N \oplus k_{HN} \oplus k^+_N \oplus h(id_N,t_N,r_N,id'_N) \oplus h(k_{HN},k^+_N)$

If this is the first stage of the protocol execution for node N_{id}, then a_N and b_N are established in some out-of-band way. Thus $h(k_{HN},k_N)$ and k_N act as uniformly random and independent keys in a one-time pad, perfectly hiding id_N and $k_{HN} \oplus id_N \oplus h(k_{HN},k_N)$ (for a_N and b_N respectively). Since, by the previous argument, the best strategy for $\mathcal{A}$ to recover k_{HN} is simply to guess, (and we instantiate the hash function with a random oracle), in order to recompute $h(k_{HN},k_N)$ $\mathcal{A}$ must either guess k_{HN} or to guess $h(k_{HN},k_N)$. Since they are

$\mathsf{Exp}^{\text{PPKA-IND,clean}}_{\Pi,n_N,n_S,\mathcal{A}}(\lambda)$:

$b \xleftarrow{\$} \{0,1\}$
tested $\leftarrow$ `false`
$k_{HN} \xleftarrow{\$} \mathsf{HKeyGen}(\lambda)$
$\mathsf{LSKflag}_i, \ldots, \mathsf{LSKflag}_{n_N} \leftarrow$ `clean`
$\mathsf{PSSflag}^1_1, \ldots, \mathsf{PSSflag}^{n_N}_{n_S} \leftarrow$ `clean`
$\mathsf{SKflag}^1_1, \ldots, \mathsf{SKflag}^{n_N}_{n_S} \leftarrow$ `clean`
$ctr \leftarrow 0$
$b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Send},\mathsf{Register},\mathsf{NextKey},\mathsf{Corrupt},\star\mathsf{Reveal},\mathsf{Test}}(\lambda)$
For $(id, stid)$ **such that Test**$(id, stid)$ **was issued**:
if $\mathsf{clean}(\pi^{stid}_{id})$ **then**
  return $(b = b')$
else
  return $b' \xleftarrow{\$} \{0,1\}$
end if

Reveal$(id, stid)$:

if $\pi^{stid}_{id}.\alpha \neq$ `accept` **then**
  return $\perp$
end if
$\mathsf{SKflag}^{id}_{stid} \leftarrow$ `corrupt`
return $\pi^{stid}_{id}.sk$

StateReveal$(id, stid)$:

if $\pi^{stid}_{id}.psstate = \perp$ **then**
  return $\perp$
end if
$\mathsf{PSSflag}^{id}_{stid} \leftarrow$ `corrupt`
return $\pi^{stid}_{id}.psstate$

NextKey(λ, id):

let $stid = max\{s : \pi^s_{id}.\alpha \neq \perp\}$
if $(\pi^{stid}_{id}.\alpha \neq$ `accept`$)$ **then**
  return $\perp$
end if
$stid \leftarrow stid + 1$
$\pi^{stid}_{id}.\alpha \leftarrow$ `active`
$\pi^{stid}_{id}, m' \leftarrow \Pi.\mathsf{NF}(\lambda, \pi^{stid}_{id}, \perp)$
return m'

Send(id, m):

if $id = HN$ **then**
  return $\Pi.\mathsf{HF}(\lambda, k_{HN}, m)$
end if
let $stid = max\{s : \pi^s_{id}.\alpha \neq \perp\}$
if $\pi^{stid}_{id}.\alpha \neq$ `active` **then**
  return $\perp$
end if
$\pi^{stid}_{id}.m_r \leftarrow \pi^{stid}_{id}.m_r \| m$
$(\pi^{stid}_{id}, m') \leftarrow \Pi.\mathsf{NF}(\lambda, \pi^{stid}_{id}, m)$
$\pi^{stid}_{id}.m_s \leftarrow \pi^{stid}_{id}.m_s \| m'$
if $\pi^{stid}_{id}.\alpha \leftarrow$ `accept` **then**
  $\pi^{stid+1}_{id}.psstate \leftarrow \mathsf{StateUpdate}(\lambda, \pi^{stid}_{id})$
end if
return m'

Corrupt(id):

$\mathsf{LSKflag}_{id} \leftarrow$ `corrupt`
return $\pi_{id}.ltk$

Test$(id, stid)$:

if $(\text{tested} =$ `true`$) \vee (\pi^{stid}_{id}.\alpha \neq$ `accept`$)$ **then**
  return $\perp$
end if
tested $\leftarrow$ `true`
if $b = 0$ **then**
  return $\pi^{stid}_{id}.sk$
else
  $sk \xleftarrow{\$} \mathcal{K}$
  return sk
end if

Register(λ):

$ctr \leftarrow ctr + 1$
$\pi.stid \leftarrow 1$
$\pi.ltk \leftarrow \Pi.\mathsf{NKeyGen}(\lambda)$
$\pi.id \leftarrow ctr$
$\pi.psstate \leftarrow \Pi.\mathsf{StateGen}(\lambda, k_{HN}, \pi.ltk)$
return $\pi.id$

$\mathsf{Exp}^{\text{PPKA-SU,clean}}_{\Pi,n_N,n_S,\mathcal{A}}(\lambda)$:

$b \xleftarrow{\$} \{0,1\}$
$k_{HN} \xleftarrow{\$} \mathsf{HKeyGen}(\lambda)$
$\mathsf{LSKflag}_i, \ldots, \mathsf{LSKflag}_{n_N} \leftarrow$ `clean`
$\mathsf{PSSflag}^1_1, \ldots, \mathsf{PSSflag}^{n_N}_{n_S} \leftarrow$ `clean`
$\mathsf{SKflag}^1_1, \ldots, \mathsf{SKflag}^{n_N}_{n_S} \leftarrow$ `clean`
$ctr \leftarrow 0$
$(id_0, id_1) \xleftarrow{\$} \mathcal{A}^{\mathsf{Send},\mathsf{Register},\mathsf{NextKey},\mathsf{Corrupt},\star\mathsf{Reveal}}(\lambda)$
NextKey$(\lambda, id_0) \rightarrow m$
$\emptyset \leftarrow \mathcal{A}^{\mathsf{Send},\mathsf{Register},\mathsf{NextKey},\mathsf{Corrupt},\star\mathsf{Reveal}}(\lambda, m)$
if $\pi^{stid}_{id_0}.\alpha \leftarrow$ `accept` **then**
  NextKey$(\lambda, id_b) \rightarrow m'$
end if
$b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Send},\mathsf{Register},\mathsf{NextKey},\mathsf{Corrupt},\star\mathsf{Reveal},\mathsf{SendTest}}(\lambda, m')$
if $\mathsf{clean}(\pi^{stid_b}_{id_0}) \wedge \mathsf{clean}(\pi^{stid_b}_{id_1})$ **then**
  return $(b = b')$
else
  return $b' \xleftarrow{\$} \{0,1\}$
end if

SendTest(m):

$\mathsf{Send}(id_b, m) \rightarrow m'$
return m'

Fig. 5: An algorithmic description of the PPKA-IND and PPKA-SU security experiments.

the same bit-length, the probability of $\mathcal{A}$ doing either is the same: $2^{-\lambda}$.

If this is not the first stage of the protocol execution, then a_N and b_N they were sent as "sub-XOR" of a previous stage η and μ. We argue that $h(k_{HN}, k^+_N)$ and k^+_N act as keys to one-time-pads for η and μ respectively, and *remain* the keys to the one-time-pad perfectly hiding id_N and $k_{HN} \oplus id_N \oplus h(k_{HN}, k_N)$ (for a_N and b_N respectively) in the following stage. It follows then that the best strategy $\mathcal{A}$ has in recovering id_N is to merely guess $id_N : 2^{-\lambda}$.

We now prove the key-indistinguishability of our PPKA protocols given in Figure 4. We begin with PPKA-2, as it captures the strongest notions of security, capturing PrFS, KCI resilience, key randomness, known key security and

hub-node authentication. Afterwards, we turn to proving the session unlinkability of PPKA-2. We then prove key indistinguishability and session unlinkability of PPKA-1. As PPKA-1 is essentially a truncated version of PPKA-2, this allows us to omit the most repetitive details of the proofs.

Theorem 1 (Key Indistinguishability of PPKA-2) *The privacy preserving key agreement protocol* PPKA-2 *given in Figure 4 is* PPKA-IND*-secure with cleanness predicate* PrFS-KCI-clean *(capturing PrFS and KCI resilience) and assuming all hash functions are random oracles. For any PPT algorithm $\mathcal{A}$ against the* PPKA-IND *key indistinguishability game,* $\mathsf{Adv}^{\mathsf{PPKA\text{-}IND,PrFS\text{-}KCI\text{-}clean}}_{\mathsf{PPKA\text{-}2},n_N,n_S,\mathcal{A}}(\lambda)$ *is negligible in the security parameter λ.*

Proof. For our proof, we assume that a test query **Test**$(id,stid)$ has been issued, and seperate into the following three cases:

- π_{id}^{stid} has accepted such that $\pi_{id}^{stid}.m_r \neq \mathsf{PPKA\text{-}2.HF}(\lambda, k_{HN}, \pi_{id}^{stid}.m_s)$.
- π_{id}^{stid} has accepted such that $\pi_{id}^{stid}.m_r = \mathsf{PPKA\text{-}2.HF}(\lambda, k_{HN}, \pi_{id}^{stid}.m_s)$ and **Corrupt**(id) has not been issued.
- π_{id}^{stid} has accepted such that $\pi_{id}^{stid}.m_r = \mathsf{PPKA\text{-}2.HF}(\lambda, k_{HN}, \pi_{id}^{stid}.m_s)$ and **Corrupt**(id) has been issued. By the definition of the cleanness predicate PrFS-KCI-clean, we assume that the per-stage secret state has not been revealed for *any* stage $s \leq stid$.

Case 1. In this case, we show that the probability the session π_{id}^{stid} such that **Test**$(id,stid)$ was issued set $\pi_{id}^{stid}.\alpha \leftarrow$ `accept` such that $\pi_{id}^{stid}.m_r \neq \mathsf{PPKA\text{-}2.HF}(\lambda, k_{HN}, \pi_{id}^{stid}.m_s)$ is negligible.

Game 0

This is a normal PPKA key-indistinguishability game. Thus we have: $\mathsf{Adv}^{\mathsf{PPKA\text{-}IND},C_1}_{\mathsf{PPKA\text{-}2},n_N,n_S,\mathcal{A}}(\lambda) = \Pr(break_0)$.

Game 1

In this game, we guess the index $(id,stid)$ of the session π_{id}^{stid}, and abort if during the execution of the experiment, a query **Test**(i^*,s^*) is received and $(i^*,s^*) \neq (id,stid)$. Thus we have: $\Pr(break_0) \leq n_N n_S \cdot \Pr(break_1)$.

Game 2

In this game, we replace the $h(k_{HN},k_N)$ value computed within π_{id}^{stid} (and, potentially, in the hub node processing $\pi_{id}^{stid}.m_s$) with a uniformly-random value $\widetilde{h(k_{HN},k_N)}$. We note that since we instantiate the hash function with a random oracle that the distribution is identical to $h(k_{HN},k_N)$. Thus, the only way that $\mathcal{A}$ can detect this change is to query (k_{HN},k_N) to the random oracle. Since the only way for $\mathcal{A}$ to do this is to recover k_{HN} fully, and we argued previously that $\mathcal{A}$'s probability of success in this endeavour is $2^{-\lambda}$, we have: $\Pr(break_1) \leq 2^{-\lambda} + \Pr(break_2)$.

Game 3

In this game we argue that the adversary $\mathcal{A}$ has a negligible probability of producing a value $\hat{\beta} = h(\widetilde{h(k_{HN},k_N)}, \hat{z_N}, \hat{r_N}, \hat{f_N}, \hat{\delta}, \hat{\eta}, \hat{\mu}, \hat{id'_N})$. Note that for $\pi_{id}^{stid}.\alpha$ to reach `accept`, $\mathcal{A}$ must produce such a value $\hat{\beta}$. We know by the definition of Case 1 that the following must be true:

- $\pi_{id}^{stid}.m_r = \langle \hat{\alpha}, \hat{\beta}, \hat{\eta}, \hat{\mu}, \hat{\delta}, \hat{id'_N} \rangle \neq \mathsf{PPKA\text{-}2.HF}(\lambda, k_{HN}, \pi_{id}^{stid}.m_s)$

Since all message fields are included in the computation of $\hat{\beta}$, and the message received by the test session does not match any output from an honest hub node, we know that the only way that $\mathcal{A}$ can cause π_{id}^{stid} to reach `accept` is to query $\widetilde{h(k_{HN},k_N)}, \hat{z_N}, \hat{r_N}, \hat{f_N}, \hat{\delta}, \hat{\eta}, \hat{\mu}, \hat{id'_N}$ to the random oracle. However, since by Game 2, $\widetilde{h(k_{HN},k_N)}$ is a uniformly-random value sampled independently from the protocol flow, the only way for $\mathcal{A}$ to produce such an input is to guess $\widetilde{h(k_{HN},k_N)}$. Thus we have: $\Pr(break_2) \leq 2^{-\lambda} + \Pr(break_3)$.

It is clear that if the session π_{id}^{stid} such that **Test**$(id,stid)$ must be issued (by Game 1) cannot reach $\pi_{id}^{stid}.\alpha \leftarrow$ `accept`, then in Game 3 the experiment proceeds identically regardless of the bit b sampled by the challenger. Thus: $\Pr(break_3) = 0$. We can now begin treating Case 2: that **Corrupt**(id) has not been issued for the appropriate node.

Case 2. In this case, we show that an adversary who issues a **Test**$(stid,id)$ query (and does not also issue a **Corrupt**(id) query) cannot win the key-indistinguishability game with non-negligible probability.

Game 0

This is a normal PPKA key-indistinguishability game. Thus we have: $\mathsf{Adv}^{\mathsf{PPKA\text{-}IND},C_2}_{\mathsf{PPKA\text{-}2},n_N,n_S,\mathcal{A}}(\lambda) = \Pr(break_0)$.

Game 1

In this game, we guess the index $(id,stid)$ of the session π_{id}^{stid}, and abort if during the execution of the experiment, a query **Test**(i^*,s^*) is received and $(i^*,s^*) \neq (id,stid)$. Thus we have: $\Pr(break_0) \leq n_N n_S \cdot \Pr(break_1)$.

Game 2

In this game, we replace the session key k_S computed by the node N_{id} in stage $stid$ with a uniformly-random and independent value $\widetilde{k_S}$. First we note that k_S is computed as $k_S =$

$h(id_N, z_N, r_N, f_N, x_N)$. Since we instantiate the hash function as a random oracle, the distribution of $\widetilde{k_S}$ and k_S is identical. In order to distinguish this change, $\mathcal{A}$ must be able to query the random oracle with the input (id_N, r_N, f_N, x_N). Since we argued previously that in order to recover id_N (the long-term secret key of the node N_{id}), $\mathcal{A}$'s only strategy to distinguish this change would be to guess the long-term secret id_N. The probability of $\mathcal{A}$ distinguishing this replacement is $2^{-\lambda}$ where λ is the bit-length of id_N.

After this change, the session key returned to $\mathcal{A}$ as the response to the **Test**$(stid, id)$ query is a uniformly-random value independent of the protocol execution regardless of the bit b sampled by the challenger. Thus we have: $\Pr(break_1) \leq 2^{-\lambda}$.

Case 3. In this case, we show that an adversary who issues a **Test**$(stid, id)$ query (and does not issue **StateReveal** queries for all per-stage secret states established before stage $stid$) cannot win the key-indistinguishability game.

Game 0

This is a normal PPKA key-indistinguishability game. Thus we have: $\mathsf{Adv}^{\mathsf{PPKA\text{-}IND},C_3}_{\mathsf{PPKA}\text{-}2,n_N,n_S,\mathcal{A}}(\lambda) = \Pr(break_0)$.

Game 1

In this game, we guess the index $(id, stid)$ of the session π^{stid}_{id}, and abort if during the execution of the experiment, a query **Test**(i^*, s^*) is received and $(i^*, s^*) \neq (id, stid)$. Thus we have: $\Pr(break_0) \leq n_N n_S \cdot \Pr(break_1)$.

Game 2

In this game, we replace the $z_N = h(k_{HN}, id_N, k_N)$ value held in secret stage by the node N_{id} with a uniformly random value $\widetilde{z_N}$ independent from the protocol execution. Since we instantiate the hash function with a random oracle, the distributions of z_N and $\widetilde{z_N}$ are identical. Thus, in order to detect this change, $\mathcal{A}$ must query the random oracle with the input k_{HN}, id_N, k_N. Since, by earlier arguments, the best strategy $\mathcal{A}$ has to recover k_{HN} is simply to guess k_{HN}, the probability that $\mathcal{A}$ is able to do this is $2^{-\lambda}$. Thus $\Pr(break_1) = 2^{-\lambda} + \Pr(break_2)$.

Game 3

In this game, we replace the computation of the z^+_N encryption key $k_z = h(\widetilde{z_N}, id_N, r_N, f_N, 0)$ with a uniformly-random and independent value value $\widetilde{k_z}$. We note that $\widetilde{z_N}$ (by Game 2) is already a uniformly random value, and the hash function is instantiated with a random oracle, this replacement is sound and indistinguishable from the perspective of $\mathcal{A}$. Thus $\Pr(break_2) = \Pr(break_3)$.

Game 4

In this game, we replace the contents of ciphertext δ with a random string of the same length, and abort if the ciphertext δ sent by the hub node HN is not the ciphertext recieved by N_{id}, but the output of decrypting δ is not $\perp$. We do so by constructing an algorithm $\mathcal{B}$ that interacts with an IND-CCA challenger in the following way: $\mathcal{B}$ acts identically as in Game 3, except for the hub node protocol execution that computes $\widetilde{k_z}$. Instead, when $\mathcal{B}$ computes δ, $\mathcal{B}$ selects a uniformly-random string $\widetilde{z^+_N}$ (of the same length as z^+_N) and submits $(z^+_N, \widetilde{z^+_N})$ to the IND-CCA encryption oracle Enc.

When the random bit b sampled by the IND-CCA challenger is 0, then δ contains the encryption of z^+_N, so $\mathcal{B}$ is a perfect simulation of Game 3. However, when the bit b sampled by the IND-CCA challenger is 1, then δ contains a random string $\widetilde{z^+_N}$ and thus $\mathcal{B}$ is a perfect simulator of Game 4. Since in Game 3, the z^+_N encryption key $\widetilde{k_z}$ is uniformly-random and independent of the protocol execution, this replacement is sound. Any adversary capable of distinguishing this change can break the confidentiality of the IND-CCA encryption scheme and guess b with perfect success. Thus $\Pr(break_3) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{Enc}} + \Pr(break_4)$.

Game 5

We now note that by Game 4, z^+_N has been established in an out-of-band way, reminiscent of the first stage run by node N_{id}. We now repeat the process of Games 2, 3, and 4 $(stid - 2)$ times to establish a z_N value for stage $stid$ run by node N_{id} that is indistinguishable from establishing z_N in some out-of-band way. Thus $\Pr(break_4) \leq (stid - 2) \cdot (2^{-\lambda} + \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{Enc}}) + \Pr(break_5)$.

Game 6

We replace z_N with a uniformly-random and independent value $\widetilde{z_N}$ in stage $stid$ of node N_{id} by the same argument as Game 2. Thus $\Pr(break_5) = 2^{-\lambda} + \Pr(break_6)$.

Game 7

In this game, we replace the computation of the session key $k_s = h(id_N, \widetilde{z_N}, r_N, f_N, 1)$ with a uniformly-random and independent value $\widetilde{k_s}$. We note that $\widetilde{z_N}$ (by Game 6) is already a uniformly random value, and the hash function is instantiated with a random oracle, this replacement is sound and indistinguishable from the perspective of $\mathcal{A}$. Thus $\Pr(break_6) = \Pr(break_7)$. We finally note that the session key established by π^{stid}_{id} is now uniformly random and independent of the protocol flow, and of the bit b sampled by the PPKA-IND challenger. Thus $\Pr(break_7) = 0$. □

We follow our proof of the key-indistinguishability of PPKA-2 by proving the session-unlinkability of PPKA-2.

Theorem 2 (Session Unlinkability of PPKA-2**)** *The PPKA* PPKA-2 *given in Figure 4 is* PPKA-SU*-secure with cleanness predicate* SU-clean *and assuming all hash functions are random oracles. For any PPT algorithm $\mathcal{A}$ against the* PPKA-SU *session-unlinkability game described in Figure 5,* $\mathsf{Adv}^{\mathsf{PPKA\text{-}SU,SU\text{-}clean}}_{\mathsf{PPKA\text{-}2},n_N,n_S,\mathcal{A}}(\lambda)$ *is negligible in the security parameter λ.*

Proof. We begin by restating the SU-clean cleanness predicate, and reiterating the impact upon our proof. For both nodes N_{id_0} and N_{id_1}, we know that the queries **Corrupt**(id_0) and **Corrupt**(id_1) have not been issued. In addition, for the stage $stid_b$ run by the unnamed node N_{id_b}, we know that a **StateReveal**$(id_b)(stid_b)$ query has not been issued.

Game 0

This is a normal PPKA session-unlinkability game. Thus we have: $\mathsf{Adv}^{\mathsf{PPKA\text{-}SU}}_{\mathsf{PPKA\text{-}2},n_N,n_S,\mathcal{A}}(\lambda) = \Pr(break_0)$.

Game 1

In this game, in the unnamed session $\pi^{stid_b}_{id_b}$, we replace the hash outputs of the form $h(id_N,X)$ (where X is a concatenation of arbitrary stings) with a uniformly random values $\widetilde{h(id_N,X)}$ chosen independently of the protocol flow. As before, since we instantiate (in our proof) the hash function with a random oracle, the distribution of this change is indistinguishable. In order to detect this change then, $\mathcal{A}$ must query the random oracle with the input (id_N,X). As per our previous arguments, in order to query id_N to the random oracle, $\mathcal{A}$ must first recover id_N. Since the best strategy to recover id_N is to simply guess the value of id_N, the probability of $\mathcal{A}$ distinguishing this change is $2^{-\lambda}$. Thus we have: $\Pr(break_0) = 2^{-\lambda} + \Pr(break_1)$.

Game 2

In this game, in the unnamed session $\pi^{stid_b}_{id_b}$, we replace the hash outputs of the form $h(k_{HN},X)$ (where X is either k_N or k^+_N) with a uniformly random values $\widetilde{h(k_{HN},X)}$ chosen independently of the protocol flow. As before, since we instantiate (in our proof) the hash function with a random oracle, the distributions of Game 1 and Game 2 are indistinguishable. In order to detect this change then, $\mathcal{A}$ must query the random oracle with the input (k_{HN},X). As per our previous arguments, in order to query k_{HN} to the random oracle, $\mathcal{A}$ must first recover k_{HN}. Since the best strategy to recover k_{HN} is, to simply guess the value of k_{HN}, the probability of $\mathcal{A}$ distinguishing this change is $2^{-\lambda}$. Thus we have: $\Pr(break_1) = 2^{-\lambda} + \Pr(break_2)$.

Game 3

In this game, in the message output by the hub node for the unnamed session $\pi^{stid_b}_{id_b}$, we replace the hash outputs $\beta = \widetilde{h(h(k_{HN},k^+_N),z_N,r_N,f_N,\delta,\eta,\mu,id'_N)}$ with a uniformly random value $\widetilde{\beta}$ chosen independently of the protocol flow. As previous arguments, the distributions of Game 2 and Game 3 are indistinguishable. In order to detect this change then, $\mathcal{A}$ must query the random oracle with the input $(\widetilde{h(k_{HN},k^+_N)},z_N,r_N,f_N,\delta,\eta,\mu,$ Since $\widetilde{h(k_{HN},k^+_N)}$ is already a uniformly random value independent of the protocol flow (by Game 2), the best strategy to distinguish this change is to simply guess the value of $\widetilde{h(k_{HN},k^+_N)}$. Thus we have: $\Pr(break_2) = 2^{-\lambda} + \Pr(break_3)$.

Game 4

In this game, in the unnamed session $\pi^{stid_b}_{id_b}$ we replace the computation of the z^+_N key $k_z = h(z_N,id_N,r_N,f_N,0)$ with a uniformly-random and independent value value $\widetilde{k_z}$. We note that since we instantiate the hash function with a random oracle, that the distribution of $\widetilde{k_z}$ and k_z is indistinguishable. Thus, in order to detect this change, $\mathcal{A}$ must query the random oracle with the input $z_N,id_N,r_N,f_N,0$. By earlier arguments, the best strategy $\mathcal{A}$ has to recover id_N is simply to guess id_N. Thus $\Pr(break_3) = 2^{-\lambda} + \Pr(break_4)$.

Game 5

In this game we replace the value δ send by the hub node to the unnamed session $\pi^{stid_b}_{id_b}$ with a uniformly random and independent values $\widetilde{\delta} \xleftarrow{\$} \{0,1\}^\lambda$. We do so by constructing an algorithm $\mathcal{B}$ that interacts with a PRF challenger in the following way: $\mathcal{B}$ acts identically as in Game 4, expect for the hub node protocol execution that computes $\widetilde{k_z}$. Instead, $\mathcal{B}$ initialise a PRF challenger and queries (z^+_n), and uses the output $\widetilde{\delta}$ from the PRF challenger to replace the computation of δ. Since by Game 4, $\widetilde{k_z}$ is a uniformly random and independent value, this replacement is sound. If the test bit sampled by the PRF challenger is 0, then $\widetilde{\delta} \leftarrow \mathsf{Enc}(\widetilde{k_z},z^+_N)$ and we are in Game 4. If the test bit sampled by the PRF challenger is 1, then $\widetilde{\delta} \xleftarrow{\$} \{0,1\}^\lambda$ and we are in Game 5. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful adversary against the PRF security of the encryption scheme Enc, and we find: $\Pr(break_4) \leq \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{Enc},\mathcal{A}}(\lambda) + \Pr(break_5)$

We pause here to reflect on the consequences of these changes. The first message sent by the unnamed node is $\langle \widetilde{tid_N},y_N,a_N,b_N,t_N,id'_N\rangle$. Since t_N is a timestamp and id'_N is sampled identically regardless of the identity of the unnamed node, the distributions of these fields is similarly identical independent of the choice of the randomly sampled bit

b. $\widetilde{tid_N}$ is a uniformly-random valued and independent of the protocol flow (by Game 1), as it is the output of a random oracle query that is of the form $(id_{N_b}, id'_N, t_n, r_n)$. This is true regardless of the choice of the randomly sampled bit b of the challenger. For y_N we remark that r_N is a uniformly-random value sampled identically from the same distribution regardless of the node identity. This value acts as the key in a one-time-pad, perfectly hiding $\widetilde{h(k_{HN}, k_N)}$. r_N is not reused (as a key) in any message in any stage, and thus y_N is a uniformly-random value, regardless of node identity. a_N is also a uniformly random value. Here, $\widetilde{h(k_{HN}, k_N)}$ acts as the key in a one-time-pad, perfectly hiding the long-term secret key id_N of the node by Game 2. Since $\widetilde{h(k_{HN}, k_N)}$ is not reused (as a key) in any message in any stage, a_N is a uniformly random value, regardless of the node identity, or the bit b randomly sampled by the challenger. Finally, we turn to b_N. We note that this time, k_N (randomly sampled by the hub node in a previous stage, uniformly-at-random) acts as the key in a one-time-pad, perfectly hiding the long-term secret key k_{HN} of the hub node, the long-term secret key id_N of the node and the value $\widetilde{h(k_{HN}, k_N)}$. k_N is not reused (as a key) in any message in any stage, and thus b_N is a uniformly-random value, regardless of node identity.

We examine the first message received by the unnamed node, $\langle \alpha, \beta, \eta, \mu, \delta, id'_N \rangle$. Again, id'_N is sampled identically regardless of the identity of the unnamed node; the distributions of the fields are similarly identical independent of the choice of the randomly sampled bit b. For α we remark that f_N is a uniformly-random value sampled identically from the same distribution regardless of the node identity. This value acts as the key in a one-time-pad, perfectly hiding $\widetilde{h(k_{HN}, k_N^+)}$. f_N is not reused (as a key) in any message in any stage, and thus α is a uniformly-random value, regardless of node identity. η is also a uniformly random value. Here, $\widetilde{h(id_N, t_N)}$ acts as the key in a one-time-pad, perfectly hiding the values r_N, f_N and a_N^+ by Game 1. Since $\widetilde{h(id_N, t_N)}$ is not reused (as a key) in any message in any stage, η is a uniformly random value, independent of the node identity, or the bit b randomly sampled by the challenger. A similar argument apples for μ, substituting $\widetilde{h(id_N, t_N, r_N, id'_N)}$ for $\widetilde{h(id_N, t_N)}$. $\widetilde{\beta}$ is a uniformly-random valued and independent of the protocol flow (by Game 3), as it is the output of a random oracle query that is of the form $(\widetilde{h(k_{HN}, k_N^+)}, z_N, r_N, f_N, \delta, \eta, \mu, id'_N)$. This is true regardless of the choice of the randomly sampled bit b of the challenger. Finally, we rely on the PRF security of the encryption scheme Enc to replace the δ field returned by the hub node. By Game 5, the value $\widetilde{\delta}$ is uniformly-random and independent of the protocol regardless of the node identity id_b. We note then that all message fields have the same distribution regardless of the challenger's randomly-sampled bit b; thus we have: $\Pr(break_5) = 0$. □

We now prove key-indistinguishability of our proposed PPKA-1, capturing known key security, and key randomness, but not forward-secrecy. It follows identically from *Case 2* of the proof of PPKA-2 key-indistinguishability, as it does not capture PrFS or KCI resilience. However, it still captures known key security, and key randomness and (obviously) key-indistinguishability.

Theorem 3 (Key Indistinguishability of PPKA-1) *The PPKA Protocol 1 PPKA-1 given in Figure 4 is PPKA-IND-secure with cleanness predicate nPrFS-clean (capturing neither PrFS nor KCI resilience) and assuming all hash functions are random oracles. For any PPT algorithm $\mathcal{A}$ against the PPKA-IND key-indistinguishability game,* $\mathsf{Adv}^{\text{PPKA-IND,nPrFS-clean}}_{\text{PPKA-1},n_N,n_S,\mathcal{A}}(\lambda)$ *is negligible in the security parameter λ.*

Proof. For our proof, we note that we cannot prove partial-forward-secrecy or key-compromise-impersonation resilience for the proposed PPKA Protocol 1. Thus, unlike PPKA-2, the cleanness predicate nPrFS-clean ensures that **Corrupt**(id) has not been issued. In this case, we assume that the per-stage secret state has been compromised at any (or perhaps, at all) previous stages. Since PPKA-1 sends the per-stage secret state $\langle a_N, b_N \rangle$ in the clear, this has no bearing on our security proof of PPKA-1.

Similarly to the proof for PPKA-2, we begin by showing that the adversary is unable to recover the Hub Node secret key k_{HN} (with non-negligible probability) even if $\mathcal{A}$ completely reveals the long-term secret keys of every normal node and the per-stage secret states of the nodes. This argument follows identically to the argument for the secrecy of k_{HN} in the proof of PPKA-2, and we can continue our proof knowing that the best strategy $\mathcal{A}$ has in recovering k_{HN} is to merely guess k_{HN}.

In this proof, we show that an adversary that issues a **Test**$(stid, id)$ query (and does not also issue a **Corrupt**(id) query) cannot win the key-indistinguishability game with negligible probability. Before we begin in earnest, we wish to show that an adversary that does not issue a **Corrupt**(id) query cannot recover the long-term secret key id_N of node N_{id}. This argument follows identically to the argument for the secrecy of id_N in the proof of PPKA-2, and we can continue our proof knowing that the best strategy $\mathcal{A}$ has in recovering id_N is to merely guess id_N.

Game 0

This is a normal PPKA key-indistinguishability game. Thus we have: $\mathsf{Adv}^{\text{PPKA-IND},C_1}_{\text{PPKA-1},n_N,n_S,\mathcal{A}}(\lambda) = \Pr(break_0)$.

Game 1

In this game, we guess the index $(id, stid)$ of the session π_{id}^{stid}, and abort if during the execution of the experiment, a query **Test**(i^*, s^*) is received and $(i^*, s^*) \neq (id, stid)$. Thus we have: $\Pr(break_0) \leq n_N n_S \cdot \Pr(break_1)$.

Game 2

In this game, we replace the session key k_S computed by the node N_{id} in stage $stid$ with a uniformly-random and independent value $\widetilde{k_S}$. First we note that k_S is computed as $k_S = h(id_N, r_N, f_N, x_N)$. Since we instantiate the hash function as a random oracle, the distribution of $\widetilde{k_S}$ and k_S is identical, thus in order to distinguish this change, $\mathcal{A}$ must be able to query the random oracle with the input (id_N, r_N, f_N, x_N). Since we argued previously that in order to recover id_N (the long-term secret key of the node N_{id}), $\mathcal{A}$'s only strategy in distinguishing this change would be to guess the long-term secret key id_N. Thus the probability of $\mathcal{A}$ in distinguishing this replacement is $2^{-\lambda}$ where λ is the bit-length of id_N.

After this change, the session key returned to $\mathcal{A}$ as the response to the **Test**$(stid, id)$ query is a uniformly-random value independent of the protocol execution regardless of the bit b sampled by the challenger. Thus we have: $\Pr(break_1) \leq 2^{-\lambda} + 0$. □

Finally, we finish our security analysis by proving the session-unlinkability of PPKA-1.

Theorem 4 (Session Unlinkability of PPKA-1**)** *The PPKA* PPKA-1 *given in Figure 4 is* PPKA-SU*-secure with cleanness predicate* SU-clean *and assuming all hash functions are random oracles. For any PPT algorithm $\mathcal{A}$ against the* PPKA-SU *session-unlinkability game described in Figure 5,* $\mathsf{Adv}^{\text{PPKA-SU,SU-clean}}_{\text{PPKA-1},n_N,n_S,\mathcal{A}}(\lambda)$ *is negligible in the security parameter λ.*

Proof. The proof of the session-unlinkability of PPKA-1 follows near-identically to the proof of session-unlinkability for PPKA-2, (with the exception of Game 4 and Game 5, since PPKA-1 does not have z_N state, nor a δ field in the hub node's response) and so we omit repeating it here. □

6.2 Functional Analysis

The proposed PPKA protocols can easily be adapted for direct communication between N and HN by removal of Steps 2 and 4 out of their respective *Authentication Phases*. As our PPKA protocols are also based on symmetric cryptographic primitives, they preserve the efficiency of the original scheme from a computation, communication and storage perspective without the aid of any additional network infrastructure. Moreover, in our protocols the timestamp field can be of any arbitrary length to suit the underlying protocol layers, unlike [21]. Assuming a B bit hash digest and 16 bit pseudo identity id'_N for node N, Tables 5 and 6 depict the various overheads associated with PPKA Protocols 1 and 2, respectively. In these tables, h denotes an instance of a hash operation and $\oplus$ denotes an XOR operation. From a computational perspective, single instances of hash operation and encryption operation have been considered equal [1].

Table 5: Overheads associated with PPKA Protocol 1

Index	Node N	Hub Node HN
Computation Overhead	$5h+9\oplus$	$7h+14\oplus$
Communication Overhead	$5B+16$ bits	$4B+16$ bits
Storage Overhead	$3B$ bits	B bits

Table 6: Overheads associated with PPKA Protocol 2

Index	Node N	Hub Node HN
Computation Overhead	$6h+9\oplus$	$10h+14\oplus$
Communication Overhead	$5B+16$ bits	$5B+16$ bits
Storage Overhead	$4B$ bits	B bits

7 Conclusion and Future Research Directions

We have proposed two authenticated key agreement protocols suitable for WBANs. The protocols are based upon symmetric cryptographic components only and are thus highly efficient and avoid the additional burden of deploying and managing an associated public key infrastructure. Our protocols are suitable for any application scenario where efficiency is of essence and the network can be initialized by a *System Administrator*. In addition to the requisite security guarantees, the proposed protocols also offer appropriate privacy attributes suitable for a wide variety of application scenarios. In order to ensure confidence in our proposals, we introduce formal security frameworks for the analysis of privacy-preserving key agreement protocols, and analyze our constructions. The proposed protocols emerge as attractive alternatives to the current key exchange methods described in the IEEE 802.15.6 standard, which are based upon legacy public key based primitives and do not offer any privacy features. One of the protocols offers the advance security properties of partial forward secrecy and KCI resilience in case of compromise of the long term secret of the sensor/client node. It would be interesting to investigate whether future research can yield a scheme which is based on symmetric primitives and still offers (full) forward secrecy and KCI resilience in the (additional) event of compromise of the long term secret of the Hub node.

8 Compliance with Ethical Standards

Conflict of Interest: Haibat Khan and Keith M. Martin declare that they has no conflict of interest. Benjamin Dowling has conflict of interest with IJIS editorial board member Joseph Piperzyk.
Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Crypto++ 5.6.5 Benchmarks. URL https://www.cryptopp.com/benchmarks.html. [Online; accessed 01-November-2018]
2. FBI-Apple encryption dispute. URL https://en.wikipedia.org/wiki/FBI%E2%80%93Apple_encryption_dispute. [Online; accessed 11-March-2019]
3. 802.15.6-2012 - IEEE Standard for Local and Metropolitan Area Networks - Part 15.6: Wireless Body Area Networks (2012). URL https://doi.org/10.1109/IEEESTD.2012.6161600
4. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: D.R. Stinson (ed.) Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings, *Lecture Notes in Computer Science*, vol. 773, pp. 232–249. Springer (1993). DOI 10.1007/3-540-48329-2_21. URL https://doi.org/10.1007/3-540-48329-2_21
5. Boyd, C., Cliff, Y., Nieto, J.M.G., Paterson, K.G.: Efficient one-round key exchange in the standard model. In: Y. Mu, W. Susilo, J. Seberry (eds.) Information Security and Privacy, 13th Australasian Conference, ACISP 2008, Wollongong, Australia, July 7-9, 2008, Proceedings, *Lecture Notes in Computer Science*, vol. 5107, pp. 69–83. Springer (2008). DOI 10.1007/978-3-540-70500-0_6. URL https://doi.org/10.1007/978-3-540-70500-0_6
6. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Information Security and Cryptography. Springer (2003). DOI 10.1007/978-3-662-09527-0. URL https://doi.org/10.1007/978-3-662-09527-0
7. Cavallari, R., Martelli, F., Rosini, R., Buratti, C., Verdone, R.: A survey on wireless body area networks: Technologies and design challenges. IEEE Communications Surveys and Tutorials **16**(3), 1635–1657 (2014). DOI 10.1109/SURV.2014.012214.00007. URL https://doi.org/10.1109/SURV.2014.012214.00007
8. Chen, L., Jordan, S., Liu, Y.K., Moody, D., Peralta, R., Perlner, R., Smith-Tone, D.: Report on post-quantum cryptography. US Department of Commerce, National Institute of Standards and Technology (2016)
9. Chen, L., Kudla, C.: Identity based authenticated key agreement protocols from pairings. In: 16th IEEE Computer Security Foundations Workshop (CSFW-16 2003), 30 June - 2 July 2003, Pacific Grove, CA, USA, pp. 219–233. IEEE Computer Society (2003). DOI 10.1109/CSFW.2003.1212715. URL https://doi.org/10.1109/CSFW.2003.1212715
10. Chien, H.: Authenticated diffie-hellman key agreement scheme that protects client anonymity and achieves half-forward secrecy. Mobile Information Systems **2015**, 354586:1–354586:7 (2015). DOI 10.1155/2015/354586. URL https://doi.org/10.1155/2015/354586
11. Deepak, K.S., Babu, A.V.: Energy efficiency analysis of IEEE 802.15.6 based wireless body area networks in scheduled access mode. Wireless Networks **22**(5), 1441–1459 (2016). DOI 10.1007/s11276-015-1041-x. URL https://doi.org/10.1007/s11276-015-1041-x
12. Diffie, W., van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Des. Codes Cryptography **2**(2), 107–125 (1992). DOI 10.1007/BF00124891. URL https://doi.org/10.1007/BF00124891
13. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Trans. Information Theory **29**(2), 198–207 (1983). DOI 10.1109/TIT.1983.1056650. URL https://doi.org/10.1109/TIT.1983.1056650
14. He, D., Zeadally, S., Kumar, N., Lee, J.H.: Anonymous authentication for wireless body area networks with provable security. IEEE Systems Journal **PP**(99), 1–12 (2016). DOI 10.1109/JSYST.2016.2544805
15. Jiang, Q., Lian, X., Yang, C., Ma, J., Tian, Y., Yang, Y.: A bilinear pairing based anonymous authentication scheme in wireless body area networks for mhealth. J. Medical Systems **40**(11), 231:1–231:10 (2016). DOI 10.1007/s10916-016-0587-1. URL https://doi.org/10.1007/s10916-016-0587-1
16. Khan, H., Dowling, B., Martin, K.M.: Highly efficient privacy-preserving key agreement for wireless body area networks. In: 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications / 12th IEEE International Conference On Big Data Science And Engineering, TrustCom/BigDataSE 2018, New York, NY, USA, August 1-3, 2018, pp. 1064–1069. IEEE (2018). DOI 10.1109/TrustCom/BigDataSE.2018.00149. URL https://doi.org/10.1109/TrustCom/BigDataSE.2018.00149
17. Krawczyk, H.: HMQV: A high-performance secure diffie-hellman protocol. In: V. Shoup (ed.) Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings, *Lecture Notes in Computer Science*, vol. 3621, pp. 546–566. Springer (2005). DOI 10.1007/11535218_33. URL https://doi.org/10.1007/11535218_33
18. Lampert, B., Wahby, R.S., Leonard, S., Levis, P.: Robust, low-cost, auditable random number generation for embedded system security. In: P. Levis, S. Eglash, L. Nachman, A. Rowe (eds.) Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems, SenSys 2016, Stanford, CA, USA, November 14-16, 2016, pp. 16–27. ACM (2016). DOI 10.1145/2994551.2994568. URL http://doi.acm.org/10.1145/2994551.2994568
19. Law, L., Menezes, A., Qu, M., Solinas, J.A., Vanstone, S.A.: An efficient protocol for authenticated key agreement. Des. Codes Cryptography **28**(2), 119–134 (2003)
20. Li, M., Lou, W., Ren, K.: Data security and privacy in wireless body area networks. IEEE Wireless Commun. **17**(1), 51–58 (2010). DOI 10.1109/MWC.2010.5416350. URL https://doi.org/10.1109/MWC.2010.5416350
21. Li, X., Ibrahim, M.H., Kumari, S., Sangaiah, A.K., Gupta, V., Choo, K.R.: Anonymous mutual authentication and key agreement scheme for wearable sensors in wireless body area networks. Computer Networks **129**, 429–443 (2017). DOI 10.1016/j.comnet.2017.03.013. URL https://doi.org/10.1016/j.comnet.2017.03.013
22. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1996). URL https://www.crcpress.com/Handbook-of-Applied-Cryptography/Menezes-van-Oorschot-Vanstone/p/book/9780849385230
23. Movassaghi, S., Abolhasan, M., Lipman, J., Smith, D.B., Jamalipour, A.: Wireless body area networks: A survey. IEEE Communications Surveys and Tutorials **16**(3), 1658–1686 (2014). DOI 10.1109/SURV.2013.121313.00064. URL https://doi.org/10.1109/SURV.2013.121313.00064
24. Park, D., Boyd, C., Moon, S.: Forward secrecy and its application to future mobile communications security. In: H. Imai, Y. Zheng (eds.) Public Key Cryptography, Third International Workshop on Practice and Theory in Public Key Cryptography, PKC 2000,

Melbourne, Victoria, Australia, January 18-20, 2000, Proceedings, *Lecture Notes in Computer Science*, vol. 1751, pp. 433–445. Springer (2000). DOI 10.1007/978-3-540-46588-1_29. URL https://doi.org/10.1007/978-3-540-46588-1_29

25. Pfitzmann, A., Hansen, M.: A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management (2010). URL http://www.maroki.de/pub/dphistory/2010_Anon_Terminology_v0.34.pdf
26. Tang, Q.: Key establishment protocols and timed-release encryption schemes. Ph.D. thesis, Department of Mathematics, Royal Holloway, University of London, Royal Holloway Research Online (2007). URL https://repository.royalholloway.ac.uk/file/605ee4f4-4adb-f265-dfc1-d15e12f6919e/1/RHUL-MA-2007-9.pdf
27. van Tilborg, H.C.A., Jajodia, S. (eds.): Encyclopedia of Cryptography and Security, 2nd Ed. Springer (2011). DOI 10.1007/978-1-4419-5906-5. URL https://doi.org/10.1007/978-1-4419-5906-5
28. Toorani, M.: On vulnerabilities of the security association in the IEEE 802.15.6 standard. In: M. Brenner, N. Christin, B. Johnson, K. Rohloff (eds.) Financial Cryptography and Data Security - FC 2015 International Workshops, BITCOIN, WAHC, and Wearable, San Juan, Puerto Rico, January 30, 2015, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 8976, pp. 245–260. Springer (2015). DOI 10.1007/978-3-662-48051-9_18. URL https://doi.org/10.1007/978-3-662-48051-9_18
29. Ullah, S., Higgins, H., Braem, B., Latré, B., Blondia, C., Moerman, I., Saleem, S., Rahman, Z., Kwak, K.S.: A comprehensive survey of wireless body area networks - on phy, mac, and network layers solutions. J. Medical Systems **36**(3), 1065–1094 (2012). DOI 10.1007/s10916-010-9571-3. URL https://doi.org/10.1007/s10916-010-9571-3
30. Wang, C., Zhang, Y.: New authentication scheme for wireless body area networks using the bilinear pairing. J. Medical Systems **39**(11), 136:1–136:8 (2015). DOI 10.1007/s10916-015-0331-2. URL https://doi.org/10.1007/s10916-015-0331-2