# Group Device Pairing based Secure Sensor Association and Key Management for Body Area Networks

Ming Li*, Shucheng Yu*, Wenjing Lou* and Kui Ren†
*Dept. of ECE, Worcester Polytechnic Institute, Worcester, MA 01609
†Dept. of ECE, Illinois Institute of Technology, Chicago, IL 60616

***Abstract*—Body Area Networks (BAN) is a key enabling technology in E-healthcare such as remote health monitoring. An important security issue during bootstrap phase of the BAN is to securely associate a group of sensor nodes to a patient, and generate necessary secret keys to protect the subsequent wireless communications. Due to the the ad hoc nature of the BAN and the extreme resource constraints of sensor devices, providing secure, fast, efficient and user-friendly secure sensor association is a challenging task. In this paper, we propose a lightweight scheme for secure sensor association and key management in BAN. A group of sensor nodes, having no prior shared secrets before they meet, establish initial trust through *group device pairing* (GDP), which is an authenticated group key agreement protocol where the legitimacy of each member node can be visually verified by a human. Various kinds of secret keys can be generated on demand after deployment. The GDP supports batch deployment of sensor nodes to save setup time, does not rely on any additional hardware devices, and is mostly based on symmetric key cryptography, while allowing batch node addition and revocation. We implemented GDP on a sensor network testbed and evaluated its performance. Experimental results show that that GDP indeed achieves the expected design goals.**

## I. Introduction

In recent years, wireless body area networks (BAN) have emerged as an enabling technique for E-healthcare systems, which will revolutionize the way of hospitalization [1]–[3]. A BAN is composed of small wearable or implantable sensor nodes that are placed in, on or around a patient's body, which are capable of sensing, storing, processing and transmitting data via wireless communications. In addition, a controller (a hand-held device like PDA or smart phone) is usually associated with the same patient, which collects, processes, and transmits the sensor data to the upper tier of the network for healthcare records. A typical structure of the BAN and its relationship with the E-healthcare system is depicted in Fig. 1.

The BAN is designed to satisfy a wide range of applications, such as ubiquitous health monitoring (UHM) [3] and emergency medical response (EMS) [1]. The UHM features long-term and consistent monitoring of a patient's health status and surrounding environment, while the EMS requires real-time medical data collection and reporting.

Unlike conventional sensor networks, a BAN deals with more important medical information which has more stringent requirements for security. Especially, secure BAN bootstrapping is essential since it secures the very first step. In this paper we focus on the *secure sensor association* problem during BAN bootstrapping (before the BAN is actually deployed). In

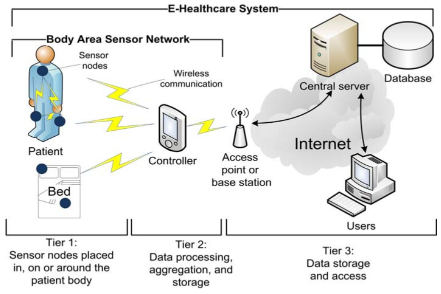
E-Healthcare System
Body Area Sensor Network
Sensor nodes
Wireless communication
Patient
Bed
Controller
Access point or base station
Central server
Database
Internet
Users
Tier 1: Sensor nodes placed in, on or around the patient body
Tier 2: Data processing, aggregation, and storage
Tier 3: Data storage and access

*Fig. 1:* A typical body area sensor network and its relationship with the E-healthcare system.

order to prevent wrong medical data from being collected, a group of BAN devices should be correctly and securely associated to an intended patient. In particular, the sensor nodes must authenticate to each other and form a group with the controller. Secret keys which only belong to the intended group are generated, so as to protect the subsequent communications. Since the wireless communication is imperceivable by human, during this process it is desirable to let a user *physically make sure that the devices ultimately forming a group includes and only includes the intended devices that s/he wants to associate* (group demonstrative identification). Since the time spent in BAN bootstrapping is a critical concern in many applications (e.g., in EMS where 5 minutes may result in a difference between life and death), the protocol must be fast while being user-friendly, i.e., involving less human interactions. Moreover, overhead is another issue since the medical sensor nodes are extremely resource-constrained.

A unique challenge for secure sensor association in BAN is, the sensor nodes may not share any prior common security contexts, since they may come from different manufacturers or have never met each other before. A secure communication channel shall be established our of an insecure channel, for all the BAN devices upon their first meet. This can be achieved by the device pairing technique that "pairs" two devices [4]. A straightforward solution is to apply device pairing between the controller and each of the $N$ sensors to establish individual keys, based on which the pairwise keys and group key can be derived. However, this requires about $N$ human interactions while each one needs tens of seconds. Current device pairing techniques are almost all designed for pairing two devices (except GAnGS [5] which still require $N$ interactions), which are not fast enough.

In this paper, we propose *group device pairing* (GDP) to address the above problems. The key observation is that, *agreeing on a group key requires fewer human interactions than establishing individual and pairwise keys between nodes one at a time*. In particular, GDP refers to the process that a group of BAN devices establish a common group key based on no prior shared secrets and no public key infrastructure (PKI), where each device authenticates itself to the whole group as a legitimate member[1], which is verified visually by a human. With a group key, standard cryptographic methods can be applied to generate other secret keys on demand after BAN deployment.

**Our contributions**. We propose a novel scheme for secure sensor association and key management in BAN. First, we put forward GDP that associates a group of BAN devices with a patient. GDP leverages device pairing and group key agreement in an unique way, in that only one simultaneous comparison of synchronous LED blinking sequences is required for a batch of at most 10 nodes, which lasts less than 30 seconds. GDP is fast, efficient, user-friendly, and also error-proof. Second, GDP enables efficient key management after network deployment. Multiple types of keys can be derived on-demand based on the group key. Also, dynamic operations, such as regular key updates, batch node addition and revocation are supported naturally by GDP. Our scheme is mostly based on symmetric key cryptography (SKC), thus having low communication and computation overhead. Third, we implement GDP on a 10 node sensor network testbed to evaluate its performance. Experimental results show that group sensor association can be done within 30 seconds with low overhead, and is intuitive-to-use.

## II. Related Work

The problem of secure sensor association in BAN has received little attention so far. In BAN, most previous works focus on security issues such as key management [1], [6], [7], encryption [1], [7], [8], and access control [8]. However, it is a non-trivial issue to securely associate sensors to a patient before any data communication happens in the BAN. Biometrical methods [9] have been adopted to establish a secure channel on human body, however this kind of channel is not always available.

Only until recently, the secure sensor association in BAN is explicitly studied by Keoh *et.al* [10]. Each sensor node is associated with the controller one-by-one using public key based authentication, where a user compares LED blinking patterns to verify each association. However, the total associating time is long since it does not support batch deployment. Also, it is impractical to assume that sensor nodes are pre-distributed with the public key of a trusted authority (TA), since each sensor would need to be registered at the TA before use. In "message-in-a-bottle" [11] and KALwEN [12], a closed faraday-cage is employed as a secure channel, in which keying materials are pre-distributed to all the intended sensor nodes before deployment. Sensor association is achieved in the sense that the user is assured no attackers out of the cage can associate to the same patient. However, costly additional hardware is required and it is cumbersome to add new nodes.

Device pairing is a promising technique to generate a common secret between two devices that shared no prior-secrets with minimum or without additional hardware. It employs some out-of-band (OOB) secure channel to exchange authenticated information. Examples include the "resurrecting duckling" [13], "talking-to-strangers" [14], and "seeing-is-believing" [15] and short string comparison based key agreement [16], [17].

The idea of device pairing has been adopted in group message authentication protocols, where each group member wants to deliver an authenticated data copy to each other. For example, GAnGS [5] requires $O(N)$ human interactions while using digital signature which increases the computation complexity. Recently, in SPATE [18], this is done through comparing T-flags. Each group member carries out $N$ comparisons in parallel to authenticate other members' data. However, SPATE is specifically designed for message exchange and is not for group key agreement. Laur et.al. [19] proposed a group message authentication and key agreement protocol (SAS-GAKA) based on comparison of short authentication strings (SAS). However, it does not achieve group demonstrative identification. Moreover, SAS and T-flags are not applicable for sensor nodes. Therefore, none of SPATE and SAS-GAKA is suitable for secure, fast, efficient and user-friendly sensor association in BAN. In GDP, the whole group is authenticated and group key is generated in one shot (i.e., requires only 1 visual comparison of synchronized LED blinking patterns).

## III. Problem Definition

### A. Network Model

A BAN consists of a controller (gateway node) and a group of sensor nodes. The size of the network may range from a few to the order of tens. The sensor nodes are variable in their functionalities; nevertheless, we assume they are all low-end nodes such as Tmote. All of them are equipped with the same wireless communication interface, say ZigBee, and so does the controller. The sensors are scarce in energy, computation and storage capabilities, while the controller is more sufficient in energy and computation resources.

The sensors may be placed in, on or around the patient's body. Although there is no consensus on the communication technologies in BAN, the communication ranges in most current proposals are larger than 3m (e.g. ZigBee), which is enough to assume that all nodes can be reached in one-hop after deployment, thus a star topology is assumed. Each BAN has an owner (patient), and a user who sets up the network (may be a nurse or patient herself).

### B. Design Requirements

#### 1) Security Goals

We shall first establish a group key through key agreement, which can be used for the controller to broadcast messages such as queries to the BAN. For the design of the authenticated group key agreement, we have the following security goals:

I. *Key secrecy* and *key confirmation* [20]. For key secrecy, each group member is assured that no non-member can obtain

[1]Note that, GDP is different from "device pairing in a group", where every pair of devices mutually authenticate each other.

the group key. Key confirmation means each member is assured that the peers actually possess the same key.

II. *Group demonstrative identification* (GDI). The user of the BAN can physically verify that the group $\mathcal{G}'$ that derives the same group key includes and only includes the nodes in $\mathcal{G}$, which is the group intended to associate with a specific patient. Actually, this includes two properties: 1) key authenticity or consistency, each legitimate group member derives the same key; 2) exclusiveness, the group only includes legitimate members but not any attackers. This extends demonstrative identification (DI) in [14], [15], but is different from PAALP in GAnGS [5].

III. *Forward secrecy*. Compromise of a group key for one session should not give adversary any information about previous group keys.

Apart from that, the individual keys shared between each sensor and the controller are needed. We shall also establish pairwise keys between sensor nodes, so that they can securely distribute their data to other sensors. Sometimes, cluster keys are also needed in BAN.

*2) Usability Goals*

I. *Efficiency*. BAN is often consisted of low-end sensors, rely on battery energy and is intended to last at least for several days [1]–[3]. To match the low-capabilities of the sensors in BAN and to minimize energy consumption, it is important to minimize computation, communication and storage overhead. Therefore, expensive cryptographic functions such as public-key operations are to be avoided whenever possible.

II. *Fast association and user-friendliness*. The sensor association of the BAN should be fast, while involving as few and intuitive human interactions as possible. Especially, batch-deployment of sensors should be supported.

III. *Error-proof*. Since human may make mistakes during the association process, the process must be easy-to-follow. Also, the system should be able to detect errors or attackers and alert the user.

IV. *Requires no additional hardware*. In order to reduce the cost of the system, it is essential to use commercial-off-the-shelf (COTS) products, and use less hardware components. For example, there should be no auxiliary devices. Also, the sensors usually do not have physical interfaces like USB, because their size may be form-factor.

In addition, because the devices may be manufactured by different vendors which are hard to inter-operate, we assume there are no pre-loaded public keys, certificates, or pre-shared secrets among the devices in BAN. The sensors are used in a plug-and-play manner. Note that, we do not preclude the existence of a PKI in Tier 3 (Fig. 1).

### C. Threat Model

First, we assume the user is trusted by the owner of the BAN; also the controller is trusted since the user can recognize his/her controller by password, and the controller is usually better kept and protected. The sensor nodes that the user wants to group together are assumed to be benign during pre-deployment (bootstrap) phase, since they are reset before use, are placed in close proximity and are under the control of the user. Yet they can be compromised after deployment. The same applies to new nodes that join the BAN afterwards. Note that, sensor nodes do not trust each other before association.

The attacker can be an outsider located in the wireless range of the sensor nodes; either in the same room or different one from the owner. The attacker's device is able to eavesdrop, intercept, modify, replay or inject the wireless communication between any devices in range. The attacker can also compromise a certain number of sensor nodes after deployment.

The main goals of an attacker are: obtain the secret keys by eavesdropping; impersonate as a legitimate group member to join the group; prevent one or more legitimate group member to join the group; act as man-in-the-middle and try to split the intended group into two or more subgroups; maliciously modify the information contributed by legitimate group members so as to violate key authentication and disrupt the group. The attacker can also pose as multiple identities to join the group, which is a Sybil attack. We do not consider denial of service (DoS) attacks in this paper.

## IV. Preliminaries

*TABLE I:* Frequently used notations

| | |
|---|---|
| $H()$ | Cryptographic hash function |
| $\mathcal{H}_{\mathbf{r}}()$ | Universal hash function; $\mathbf{r}$: keys |
| $x \overset{R}{\leftarrow} S$ | Choose $x$ uniformly from set $S$ |
| $E_K\{\cdot\}$ | Symmetric encryption with key $K$ |
| $\hat{x}$ | The unauthenticated version of $x$ |
| $a\|b$ | Concatenation of $a$ and $b$ |
| $M_i$ | A node or a group member |
| $\mathcal{G}$ | The group of devices intended to associate to a patient |
| $K_G$ | The group key |
| $\mathcal{S}_k$ | A subgroup of index $k$ |
| $N$ | Total number of devices in the group |
| $n_{max}$ | Maximum subgroup size |
| $n$ | Size of a subgroup |
| $s$ | The length of both $K_G$ and nonce |
| $\rho$ | Length of the short authentication string |

### A. Device Pairing

A "Pairing method" refers to the type of auxiliary OOB channel used. When selecting a pairing method, practical factors need to be considered. In BAN, sensor nodes may only have LED lights, beepers and buttons, but no interfaces such as camera, displays or keyboards; yet the controller may have all of them. Under this asymmetric setting, the methods in [14], [15] are unable to achieve mutual authentication. Fortunately, the "Blink-Blink" (BB) pairing method proposed in [21] was shown to be a practical approach. Briefly, both devices encode a short authentication string (SAS) obtained from a protocol run to a synchronized LED blinking pattern, where a '1' bit encodes to a "blink" (on) period and '0' bit encodes to an "off". Then the user compares the patterns and accepts the results if they are the same.

### B. Commitment Schemes

Commitment schemes are important cryptographic primitives that have been widely used in message authentication [22] and authenticated key agreement protocols [16], [17], [19]. Basically, a commitment scheme should have two properties: 1) *hiding*: the committed value is hidden from the receiver until the sender reveals it. 2) *binding*: it is infeasible to find another value $m' \neq m$ that commits to the same $c$.

The existing commitment schemes that aim at achieving strictly proven security are all based on heavy asymmetric cryptography [22], which are inefficient to implement on low-end sensor nodes like Tmote. For practical reasons, one-way hash functions (OHF) has been adopted as an alternative [18], [23]. Ideally, an OHF has three properties: 1) pre-image resistant; 2) second pre-image resistant; 3) collision resistant. In this paper, we follow the same approach and call it *hash commitments*.

### *C. Group Key Agreement Scheme*

A contributory group key agreement establishes a group key based on no pre-shared secret, where every member equally contributes one share of the group key. In this paper, we choose the unauthenticated group key agreement protocol (UDB) proposed by Dutta and Barua [24] as a primitive. It is based on Diffie-Hellman (DH) key agreement and is provably secure, and only requires 2 rounds of communication. However, its authenticated version uses digital signatures, which requires PKI and is unsuitable for BAN. We describe the UDB protocol for completeness in Fig. 2. $\mathbb{Z}_q^*$ is a multiplicative group of prime order $q$, where $g$ is a generator. Note that, $K_G = g^{x_1 x_2 + x_2 x_3 + \ldots + x_n x_1}$. Each node spends 2 broadcast messages, 3 modular exponentiations, $2n-2$ modular multiplications and 1 modular division.

*Fig. 2:* Unauthenticated DB Key Agreement Protocol. ($1 \le i \le n$)

```
   /* Round 1:                                                        */
1  $M_i$:            $x_i \xleftarrow{R} \mathbb{Z}_q^*$; $X_i \leftarrow g^{x_i}$;
2  $M_i \rightarrow M_{i-1}, M_{i+1}$: $X_i$  //can be achieved by a broadcast;
   /* Round 2:                                                        */
3  $M_i$:            $K_i^L \leftarrow X_{i-1}^{x_i}$; $K_i^R \leftarrow X_{i+1}^{x_i}$; $Y_i \leftarrow K_i^R / K_i^L$;
4  $M_i \longrightarrow *$:     $Y_i$ //"$\longrightarrow *$" stands for broadcast in the wireless channel;
5  $M_i$:            $\hat{K}_{i+1}^R \leftarrow Y_{i+1} K_i^R$;
6  for $j = 2$ to $n-1$ do
7      $M_i$:        $\hat{K}_{i+j}^R \leftarrow Y_{i+j} \hat{K}_{i+j-1}^R$;
8  end
   /* Key computation:                                                */
9  $M_i$:            verifies $K_i^L = \hat{K}_{i+n-1}^R$; if fails, abort;
10 $M_i$:            group key: $K_G \leftarrow \hat{K}_1^R \hat{K}_2^R \ldots \hat{K}_n^R$;
```

## V. Secure Sensor Association and Key Management for BAN

### *A. Overview*

Our scheme spans three phases: 1) Pre-deployment. In this phase, the sensor nodes are bootstrapped for the first time after purchased by the user or owner. This phase is assumed to be immune of node compromise, which allows the user to securely associate the sensor nodes to a patient. Group device pairing is performed among the sensor nodes and the controller to setup a group key. Also, keying materials are distributed by the controller to each sensor node using the group key. 2) Deployment. Nodes are actually deployed to designated places on/in/around the human body. Neighbor discovery is performed to form a BAN topology, pairwise keys are computed, and a logical key hierarchy is established. 3) Working phase, when the regular functions (e.g. collecting and reporting medical data) are executed. Our scheme updates all the keys periodically, and handles node join/leave/revocation efficiently.

| **Controller (A)** | | **Sensor Node (B)** |
|---|---|---|
| Input: $ID_A, x_A \xleftarrow{R} \mathbb{Z}_q^*, X_A = g^{x_A}$ | | $ID_B, x_B \xleftarrow{R} \mathbb{Z}_q^*, X_B = g^{x_B}$ |
| Pick $r_A \xleftarrow{R} \{0,1\}^s$ | | Pick $r_B \xleftarrow{R} \{0,1\}^s$ |
| $m_A \leftarrow ID_A \vert X_A$ | | $m_B \leftarrow ID_B \vert X_B$ |
| $c_A \leftarrow H(m_A \vert r_A)$ | $\xrightarrow{c_A}$ | |
| | $\xleftarrow{c_B}$ | $c_B \leftarrow H(m_B \vert r_B)$ |
| | $\xrightarrow{m_A, r_A}$ | Verify $\hat{c}_A = H(\hat{m}_A \vert \hat{r}_A)$ |
| Verify $\hat{c}_B = H(\hat{m}_B \vert \hat{r}_B)$ | $\xleftarrow{m_B, r_B}$ | $\mathsf{sas}_B : \mathtt{trunc}(\mathcal{H}_{\hat{r}_A, r_B}(\hat{m}_A \vert m_B))$ |
| $\mathsf{sas}_A : \mathtt{trunc}(\mathcal{H}_{r_A, \hat{r}_B}(m_A \vert \hat{m}_B))$ | | |
| LED blinking pattern$\leftarrow \mathsf{sas}_A$ | $\Longleftrightarrow$ | LED blinking pattern$\leftarrow \mathsf{sas}_B$ |
| User accepts, if patterns match | | |
| //**Key confirmation**: | | |
| $K_{AB} \leftarrow \hat{X}_B^{x_A}, s_A \leftarrow H(K_{AB} \vert ID_A)$ | $\xrightarrow{ID_A, s_A}$ | $K'_{AB} \leftarrow \hat{X}_A^{x_B}$ |
| | | Check if $\hat{s}_A = H(K'_{AB} \vert ID_A)$ |
| | | Beep if not equal |
| Check if $\hat{s}_B = H(K_{AB} \vert ID_B)$ | $\xleftarrow{ID_B, s_B}$ | $s_B \leftarrow H(K'_{AB} \vert ID_B)$ |
| Beep if not equal | | |
| "$\Longleftrightarrow$: OOB secure channel; | | "$\longleftrightarrow$": wireless channel. |

*Fig. 3:* Authenticated key agreement between the controller and a sensor node in Scheme I.

### *B. Pre-deployment Phase*

In this phase, a group of sensor nodes and a controller picked by the user must be uniquely associated to the patient they will serve for. We first present a straightforward scheme (Scheme I) and then describe our main scheme based on GDP.

#### *1) The Straightforward Scheme*

In Scheme I, the controller first performs authenticated key agreement to establish a secret key with each sensor one-by-one (Fig. 3). It combines DH key agreement with device pairing. $A$ and $B$ first both generate a DH public value ($X_A$ and $X_B$), and a random nonce ($r_A$, $r_B$), respectively. Then they compute hash commitments ($c_A, c_B$) to their messages ($m_A$, $m_B$) and nonces, and exchange the messages after sending the hash commitments. Later, $A$ and $B$ both compute a SAS in order to authenticate $m_B$ and $m_A$. The SAS is the truncation of a universal hash function to the leading $\rho$ bits: $\mathtt{trunc}(\mathcal{H}(\cdot))$, whose inputs are $r_A$, $r_B$, $m_A$ and $m_B$. After that, the SASes are encoded into LED blinking patterns which are displayed synchronously. The user indicates "authentication accepted" to the controller if the patterns are the same. Thus, $K_{AB} = \hat{X}_B^{x_A} = \hat{X}_A^{x_B} = g^{x_A x_B}$. However, since there is no user interface on a sensor, we add an additional round to let the sensor "know" that the controller is authenticated through key confirmation.

After that, a group key $K_G$ is generated by the controller. To distribute the group key, the controller simply encrypts it $N$ times using the shared keys between it and each sensor node, and then unicasts to each sensor node. Now, the user enters the ID of the patient into the controller, and associates the individual keys and the group key with this ID, which is also the ID of the BAN.

However, associating sensor nodes one-by-one is very time-consuming, since each pair of LED blinking requires tens of seconds. Therefore, a more scalable and efficient method must be developed.

### *2) Sensor Association via Group Device Pairing*

In contrast to Scheme I, the GDP directly establishes a group key through authenticated group key agreement. The idea is to authenticate the protocol transcript of a group key agreement scheme by simultaneously comparing LED blinking patterns for a group of nodes.

We first give an overview of the sensor association process. If the size of the intended group $N = |\mathcal{G}| \leq n_{max}$, the user carries out one GDP for $\mathcal{G}$ to setup the group key $K_G$. If $N > n_{max}$, the user randomly picks nodes from $\mathcal{G}$ in a batch to form smaller subgroups whose sizes are equal to $n_{max}$ when possible, where $n_{max}$ is a parameter. The GDP protocol is then executed for each subgroup $\mathcal{S}_k$. The controller is in each subgroup, so that it can establish a subgroup key $K_{\mathcal{S}_k}$ with each of them through GDP. When the last subgroup has only one sensor node left, Scheme I is automatically used to establish a pairwise key (however, it makes no difference to the user). After that the controller generates the final group key $K_G$ and distributes it by encryption to each subgroup: $E_{K_{\mathcal{S}_k}}\{K_G|\mathcal{G}\}, H(K_G|\mathcal{G})$, where $\mathcal{G} = \cup_k \mathcal{S}_k$ and $|\mathcal{G}| = N$.

The GDP for a subgroup $\mathcal{S}_k$ is outlined in Fig. 4. It can be roughly divided into four phases. In the counting & initialization phase, the user $U$ first randomly picks $m$ sensor nodes and places them in close proximity, powers them on and begins the association process. Then $U$ enters the group member count information ($n = m + 1$) into the controller and indicates to start the protocol (steps 2-3). Each member device broadcasts its original identity to determine its unique ID in the group (only needed for GDP): $ID_i \in \{1, 2, 3, ..., n\}$, and sets $M_i = ID_i$ (step 4)[2]. Now each node knows the group size by listening to the broadcasts. The controller aborts if the group size does not equal to $n$.

Next the UDB protocol is started for key agreement. In steps 6-9, each member $M_i$ first computes $X_i$ and broadcasts. Based on collected $X_j$s, each $M_i$ computes $Y_i$. Then $X_i$ and $Y_i$ are treated as the message to be authenticated. In the authentication phase, a random nonce $r_i$ is first generated which will be used as hash keys afterwards. Then $r_i$ is committed along with the message and member ID (step 11). After all members sent their commitments (step 12), they reveal the committed message and nonces so that each member can verify (steps 14-15). The controller, upon collecting all the other members' commitments and messages, checks if the number of group members, the commitments and messages all equal to $n$ (steps 16-17). The SAS is truncated from a keyed universal hash function, with inputs being the protocol transcript. Since the nonces are hidden to the attacker before revealing, they provide the randomness required for security. Next, SASes are encoded into synchronized LED blinking patterns for user comparison (step 18). Whether or not the patterns are the same, $U$ indicates the results to the controller (step 19).

The key derivation and confirmation phase tells the sensor nodes about the result of LED blinking pattern comparison. All members derive the group key as in the UDB protocol (step 20), then the controller encrypts the above result using the $K_{\mathcal{S}_k}(1)$ derived by itself, and broadcasts to all sensor nodes (step 21). Nodes will alert the user if no confirmation message is received in a short period (typically less than 1s) (step 22). A sensor node, decrypts this message using its own $K_{\mathcal{S}_k}(i)$, and alerts the user if the keys don't match. User will abort the operation and reset all sensors if an alert is heard (step 23-24). Finally, if the protocol does not abort, a group key $K_G$ is derived by all members, and $\forall M_i \in \mathcal{S}_k$ retains the intermediate key $K_{\mathcal{S}_k}$ for key management purposes.

```
// **Counting and initialization**
1. $U$ picks $m \leq n_{max}$ sensors randomly to form a subgroup $\mathcal{S}_k$
2. $U \xrightarrow{user\ interface} M_1$        : Enter group size $n = m + 1$
3. Controller $\rightarrow *$                   : "$M_1$ wants to initiate the protocol"
4. Others $\xrightarrow[identity]{broadcast} *$ : Determine $ID_i$, $M_i \leftarrow ID_i$
5. $M_1$                                        : $\hat{\mathcal{S}}_{k_1} \leftarrow ID_1 \cup \{ID_i\}$; if $|\hat{\mathcal{S}}_{k_1}| \neq n$, abort
// **Start the UDB protocol**
6. $M_i$                   :$x_i \overset{R}{\leftarrow} \mathbb{Z}_q^*$; $X_i \leftarrow g^{x_i}$
7. $M_i \rightarrow *$     :$X_i$
8. $M_i$                   :Compute $K_i^L, K_i^R, Y_i$
9. $M_i$                   :$m_i \leftarrow \{X_i|Y_i\}$ (to be authenticated)
// **Authentication steps**
10. $M_i$                  :$r_i \overset{R}{\leftarrow} \{0,1\}^s$,$\hat{\mathcal{S}}_{k_i} \leftarrow \varnothing$
11. $M_i$                  :$c_i \leftarrow H(ID_i|m_i|r_i)$
12. $M_i \rightarrow *$    :$ID_i, c_i$
13. $M_i$                  :$\hat{\mathbf{C}}_i \leftarrow c_i \cup \{\cup_{j\neq i}\hat{c}_j\}, \hat{\mathcal{S}}_{k_i} \leftarrow ID_i \cup \{\cup_{j\neq i}\hat{ID}_j\}$
14. $M_i \rightarrow *$    :$m_i, r_i$
15. $M_i$                  :Verify that $\forall j \neq i, \hat{c}_j = H(\hat{ID}_j|\hat{m}_j|\hat{r}_j)$
                              If wrong, abort
16. $M_i$                  :$\hat{\mathbf{m}}_i \leftarrow m_i \cup \{\cup_{j\neq i}\hat{m}_j\}, \hat{\mathbf{r}}_i \leftarrow r_i \cup \{\cup_{j\neq i}\hat{r}_j\}$
17. $M_1$                  :If any of $|\hat{\mathbf{C}}_1|, |\hat{\mathbf{m}}_1|, |\hat{\mathbf{r}}_1|, |\hat{\mathcal{S}}_{k_1}| \neq n$
                           Broadcast an "abort" message to all sensors
18. $M_i$                  :$\mathsf{SAS}_i \leftarrow \mathtt{trunc}(\mathcal{H}_{\hat{\mathbf{r}}_i}(\hat{\mathbf{C}}_i|\hat{\mathbf{m}}_i|\hat{\mathcal{S}}_{k_i}))$
                           Display LED blinking pattern from $\mathsf{SAS}_i$
19. $U \xrightarrow{user\ interface} M_1$   :$\mathtt{result}$="APM" or "SD"
// **Key derivation and confirmation**
20. $M_i$                  :Compute $\hat{K}_j^R, j \in \hat{\mathcal{S}}_{k_i} \backslash i$, $K_{\mathcal{S}_k}(i) \leftarrow \hat{K}_1^R \hat{K}_2^R ... \hat{K}_n^R$
21. $M_1 \rightarrow *$    :$E_{K_{\mathcal{S}_k}(1)}\{\mathtt{result}|ID(M_1)\}$
22. $M_i$                  :Wait for timeout, beep if no msg received.
23. $M_i, i \neq 1$        :Decrypt using $K_{\mathcal{S}_k}(i)$
                           If obtain meaningful plaintext
24. (This includes $M_1$)      If $\mathtt{result}$ = "APM", wait for $T_{timeout}$
                               Else abort, delete $K_{\mathcal{S}_k}(i)$
                           Else $M_i \rightarrow *$:"FA", delete $K_{\mathcal{S}_k}(i)$
25. $M_i$ (upon timeout)   :If no "FA", $K_{\mathcal{S}_k} = H(K_{\mathcal{S}_k}(i))$
                           Else, beep to alert the user and abort.
Note: APM="All patterns match"; SD="some differ"; FA="fail alert".
```

*Fig. 4:* Group device pairing protocol for a subgroup for secure sensor association in BAN. Group member are represented as $M_i, 1 \leq i \leq n$. The group includes one controller (indexed by $M_1 = 1$) and $n - 1$ sensor nodes.

### *3) Distribution of Keying Materials*

After the sensor association is successfully done, we use the Blundo's polynomial based key pre-distribution method [25] to enable nodes derive pairwise keys afterwards. The controller first randomly generates a bivariate $t$-degree symmetric polynomial $f(x, y) = \sum_{i,j=0}^{t} a_{i,j}x^i y^j$ defined over a finite field $\mathbb{F}_p$ with $p$ being a large prime number[3]. The controller computes a univariate *polynomial share* for each node $u$: $f_u(y) = f(u, y)$.

[2]This can be achieved distributively by letting each device broadcast after a random backoff delay, and count the number of identities received before itself.

[3]For example, we can use $p \approx 2^{80}$ to provide a 80 bit symmetric key.

Then it encrypts and unicasts this to each sensor node:

$$(msg1)\ C \longrightarrow u:\ u, E_{K_G}\{f_u(y)\}|H(f_u(y)).$$

where the hash provides message authentication. After node $u$ receives the share for itself, $u$ stores it in the memory and ignores all other nodes' shares. Now, the pairwise key between $u$ and $v$ is: $K_{uv} = f_u(v) = f_v(u) = K_{vu}$. The individual keys shared between each sensor node and the controller are also derived and saved.

In addition, in order for the controller to authenticate itself afterwards, the controller generates a *one-way hash chain* [26]: $\bar{k}_n, \bar{k}_{n-1}, ..., \bar{k}_0$, where $\bar{k}_i = H(\bar{k}_{i+1}), 0 \leq i \leq n-1$. The controller distributes the commitment of the chain ($\bar{k}_0$) to all sensor nodes:

$$(msg2)\ C \longrightarrow u:\ E_{K_G}\{\bar{k}_0\}|H(\bar{k}_0).$$

## C. Deployment and Thereafter

The deployment phase establishes the pairwise and logical keys. Upon deployment, each node $M_i$ first performs neighbor discovery, and we consider a star topology. For each neighbor $M_j$, $M_i$ computes the pairwise key $K_{ij}$ as previously mentioned. In practice, in order to save storage space, a node can merely store the pairwise keys that it uses frequently, while computing the other pairwise keys on-demand.

Then, the logical keys are derived naturally from the subgroup keys in GDP, which are used to form a logical key hierarchy (LKH). The LKH [27] has been proposed to achieve efficient key revocation. Since the LKH is a balanced binary tree, the message overhead for key revocation is $O(log_2(N))$. However, it is not very efficient for batch node addition or removal.

To avoid this drawback, we use a constant depth ($d = 3$), variable branch and balanced key tree (Fig. 5). Each internal node stands for a logical key, and each leaf node corresponds to the individual key of a sensor node. We have $k_{0,0} = K_G$ and $k_{2,i} = k_{M_{i+2},M_1}$. The keys $k_{1,i} = K_{\mathcal{S}_k}$ which are the subgroup keys derived in the end of GDP. The branch of the root $\mu_{0,0} = |\{\mathcal{S}_k\}|$, while the branch of a second level node $\mu_{1,i} = |\mathcal{S}_i|$. The controller $M_1$ has the information of the entire key tree. Note that, no messages are needed to transmit the logical keys for the tree in our scheme.

Note that, our scheme can be easily extended to BANs with clusters, since we can predict which nodes will form a cluster and thereby a subgroup by looking at their functionalities. For example, the use of several sensor nodes connected to 30 motion sensors are reported in [28] to detect patient's acceleration and gait. In this case, the cluster keys will be the logical keys and the subgroup keys at the same time.

After that, the BAN is ready to function. In summary, now a sensor node $M_i$ has the following key (material)s: $K_G, k_{M_i,C}, K_{\mathcal{S}_k}, f_i(y), \bar{k}_0$. Since the keys may be compromised by cryptanalysis afterwards, we need to introduce sessions for the working phase —- time periods across which keys are updated regularly. The above keys are all treated as keys in session 0. A key $K$ in session $i$ is denoted as $K(i)$.

### 1) Session Key Update

Periodically, the controller broadcasts a update message to the network. It is authenticated using the local broadcast

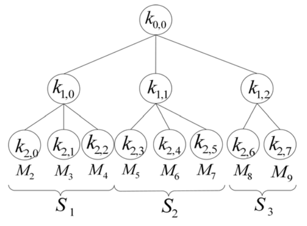

*Fig. 5:* A logical key tree for a BAN of 9 nodes ($n_{max} = 3$). A key is indexed by its level $\lambda$ and branch number $\mu$. $\mathcal{S}_k$ refers to a subgroup.

authentication method [29], since we assume the BAN is one-hop. The controller first updates $f(x, y)$: $f_{i+1}(x, y) = f_i(x, y) + \Delta_{i+1}$, where $\Delta_{i+1} \overset{R}{\leftarrow} \mathbb{F}_p$. Then, it updates the logical keys as $K_{0,0}(i + 1) = H(K_{0,0}(i)), k_{1,\mu}(i + 1) = H(k_{1,\mu}(i))$, and broadcasts the following:

$$msg3 \leftarrow \texttt{Update to session } i + 1|\Delta_{i+1}$$
$$C \longrightarrow *: E_{K_{0,0}(i)}\{msg3\}, \bar{k}_{i+1},\ H_{\bar{k}_{i+1}}(msg3).$$

Then, each sensor can authenticate $C$ by verifying that $H(\bar{k}_{i+1}) = \bar{k}_i$.

Next, all sensor nodes update all the keys in its memory as the controller does. For the pairwise keys, node $u$ computes $f_{u,i+1}(y) = f_{u,i}(y) + \Delta_{i+1}$. This achieves the update of all $\frac{n(n-1)}{2}$ pairwise keys through only one broadcast message.

## D. Membership Management

### 1) Node Join

Adding one node is easy; we can just perform one device pairing using Scheme I. We will elaborate on how GDP supports efficient batch node addition.

Step 1. Before $l > 1$ new nodes $M_{N+1}, ..., M_{N+l+1}$ join the BAN during session $i$, they are reset by the user (all dynamic memories are lost) and assumed to be benign.

Step 2. Before they are deployed, the same steps in GDP are performed by treating them as a new group, where the controller obtains the temporary group key $K_G^T$ and all the logical keys.

Step 3. The controller advances the existing BAN to session $i + 1$ without waiting until the end of session $i$. To this end, all nodes do the same thing as in session key update.

Step 4. The controller pre-distributes new polynomial shares $f_{v,i+1}(y)$ for each new node $v$. Also, $C$ encrypts $K_G(i + 1)$ and $\bar{k}_{i+1}$ using $K_G^T$ and broadcasts to the new nodes. A new key tree can then be derived that includes the new nodes. Then, the new nodes are deployed.

### 2) Node Leave/Revocation

Upon single node leave or revocation during session $i$, the group key, logical keys and pairwise keys are renewed to exclude the leaving node. The controller randomly generates a new group key $K_G(i + 1)$. All the logical keys on the tree path of the leaving node are refreshed. For example, in Fig. 5, say $M_2$ is revoked. Then, the controller sends the following messages:

$$C \to M_3 : E_{k_{M_3,M_1}}\{k_{1,0}(i+1)\};$$
$$C \to M_4 : E_{k_{M_4,M_1}}\{k_{1,0}(i+1)\};$$
$$C \to M_3, M_4 : E_{k_{1,0}(i+1)}\{k_{0,0}(i+1)\};$$
$$C \to M_5, M_6, M_7 : E_{k_{1,1}(i+1)}\{k_{0,0}(i+1)\};$$
$$C \to M_8, M_9 : E_{k_{1,2}(i+1)}\{k_{0,0}(i+1)\};$$

where $k_{1,1}(i+1) = H(k_{1,1}(i)), k_{1,2}(i+1) = H(k_{1,2}(i))$. After that, the controller sends the updated polynomial share ($\Delta_{i+1}$) to all nodes using authenticated broadcast. Thus, the revoked node cannot obtain the new group key and the updated polynomial share. It is straightforward to see how the above is done when batch node leave event happens, for which we will analyze the efficiency in Sec. VII.

## VI. Security Analysis

### A. Security of GDP

#### 1) Key Secrecy

It is proved in [24] that the UDB protocol is secure against passive adversary under the Decisional Diffie-Hellman (DDH) assumption. In GDP, all information sent over wireless channel that are related to the group key are the $X_i$s and $Y_i$s, from which an eavesdropping attacker cannot derive $K_G$ except with negligible probability.

#### 2) Key Authentication

Compromising key authentication involves active attacks. We argue that this is infeasible. The attacker $\mathcal{A}$'s goals are: prevent legitimate members from obtaining the correct group key while joining the group itself ($\mathcal{A}$ interacts with a subset of members), or act as Man-in-the-Middle (MitM) and try to split the group into two ($\mathcal{A}$ interacts with both subgroups).

First, we consider the first case. In order to result in different keys in legitimate group members, $\mathcal{A}$ must break the consistency of exchanged messages in GDP. Assume the user correctly compares the LED blinking patterns, if $\mathcal{A}$ modifies the key shares of a legitimate group member, she must make the final SAS of each legitimate member look the same. Now, suppose $\mathcal{A}$ wants to impersonate a member $M_i$ and prevents $M_i$ from joining the group. $\mathcal{A}$ generates $X_{\mathcal{A}}$ as its key share, intercepts $M_i$'s key share $X_i$ and replace it by $X_{\mathcal{A}}$. Also, $\mathcal{A}$ computes its own $Y_{\mathcal{A}}$ and replaces $Y_i$ by $Y_{\mathcal{A}}$. Without the commitments to the random nonces $r_1, ..., r_n$ (step 11), $\mathcal{A}$ can wait until all members (including $i$) broadcast their nonces and group key shares, then computes a $r_{\mathcal{A}}$ to satisfy: $\texttt{trunc}\ \mathcal{H}_{...,r_i,...}(\{..., X_i, Y_i, ...\}, \mathcal{G}) = \texttt{trunc}\ \mathcal{H}_{...,r_{\mathcal{A}},...}(\{..., X_{\mathcal{A}}, Y_{\mathcal{A}}, ...\}, \hat{\mathcal{G}})$, which requires $2^{\rho}$ hash computations (the output space of SAS is $2^{\rho}$). And then $r_{\mathcal{A}}$ is sent to all other members instead of $r_i$. If $\rho = 16$, this can be done efficiently in seconds with a commercial computer. Then $\mathcal{A}$ derives the group key, by including itself but excluding $M_i$.

However, by first sending the commitments of all nonces, $\mathcal{A}$ cannot precompute the value of $r_{\mathcal{A}}$. This is due to the pre-image resistant (hiding) property of the hash commitment to $r_i$, which leaks no information about $r_i$. Also, the second pre-image resistant (binding) property prevents the attacker from finding another pair of $m_{\mathcal{A}}$ and $r_{\mathcal{A}}$ that hashes to the same $c_i$. After $\mathcal{A}$ commits to a random $r_{\mathcal{A}}$, she cannot change it. Without knowing at least one of the legitimate group member's nonce and key shares, $\mathcal{A}$ cannot precompute the values needed to make all the SASes equal. Therefore, the best $\mathcal{A}$ can do is to guess randomly for $c_{\mathcal{A}}$, $m_{\mathcal{A}}$ or $r_{\mathcal{A}}$, hoping that $\forall j, l \in \mathcal{G}, \texttt{trunc}\ \mathcal{H}_{\hat{\mathbf{r}}_j}(\hat{\mathbf{C}}_j|\hat{\mathbf{m}}_j|\hat{\mathcal{G}}_j) = \texttt{trunc}\ \mathcal{H}_{\hat{\mathbf{r}}_l}(\hat{\mathbf{C}}_l|\hat{\mathbf{m}}_l|\hat{\mathcal{G}}_l)$. Because the probability to find a collision of the universal hash function is smaller than $1/|\mathcal{Y}|$ where $\mathcal{Y}$ is the output space, such attack will successes with probability $2^{-\rho}$. Therefore, $\rho = 20$ gives $1.5^{-5}$ and is believed to provide enough security strength in practical scenarios.

Second, for the MitM attack, $\mathcal{A}$ will try to split the group into two by intercepting/modifying/relaying information between the two, and then establish group keys with the two groups separately. This extends the impersonation attack against one group member. Now the equation becomes $\forall \mathcal{S}, \mathcal{T} \subset \mathcal{G}, \texttt{trunc}\ \mathcal{H}_{\mathbf{r}_{\mathcal{S}},\hat{\mathbf{r}}_{\mathcal{T}}}(\mathbf{C}_{\mathcal{S}}|\hat{\mathbf{C}}_{\mathcal{T}}|\mathbf{m}_{\mathcal{S}}|\hat{\mathbf{m}}_{\mathcal{T}}|\mathcal{G}_{\mathcal{S}}|\hat{\mathcal{G}}_{\mathcal{T}}) = \texttt{trunc}\ \mathcal{H}_{\hat{\mathbf{r}}_{\mathcal{S}},\mathbf{r}_{\mathcal{T}}}(\hat{\mathbf{C}}_{\mathcal{S}}|\mathbf{C}_{\mathcal{T}}|\hat{\mathbf{m}}_{\mathcal{S}}|\mathbf{m}_{\mathcal{T}}|\hat{\mathcal{G}}_{\mathcal{S}}|\mathcal{G}_{\mathcal{T}})$. However, the advantage of $\mathcal{A}$ is still the same, assuming an ideal hash commitment and an universal hash function. The above ensures that GDP achieve key authenticity. Note that, similar arguments apply to the Scheme I.

#### 3) Exclusiveness

If there is no member count information, exclusiveness cannot be achieved, as is the case in [19]. This is because before the group of sensor meets with each other, they do not know the member list in advance. An attacker $\mathscr{A}$ can thus claim it is one of the group members and inject her key share, trying to obtain the group key. Then the actual group becomes $\hat{\mathcal{G}} = \mathcal{G} \cup \mathscr{A}$, while for members in $\mathcal{G}$, they still have the same SAS values. While the only sign that the user perceives is the LED blinking patterns on the sensor nodes, s/he will accept $\hat{\mathcal{G}}$ as authenticate. However, with the count information, this attack can be defeated. First, if $n+1$ key shares are received by the controller, GDP will abort, assuming that the user counts correctly. Second, if $C$ only receives $n$ $X_i$s and $Y_i$s from $\mathcal{G}$, but $\mathcal{G} \backslash C$ all receive $n+1$ key shares from $\mathcal{G} \cup \mathcal{A}$, $\mathcal{A}$ will not be able to derive the same key with $\forall j \in \mathcal{G}$, thus have no gain. Even if $\mathcal{A}$ carries out such attack to disrupt the group, it will not be able to make all the SASes equal since $\hat{\mathbf{m}}_C \neq \hat{\mathbf{m}}_i, \forall i \in \mathcal{G} \setminus C$, and the same argument for key authenticity applies. Even if the SASes appear to be the same for $\mathcal{G}$, GDP can detect attacks through the key confirmation steps. Note that, it does not matter how many key shares $\mathcal{A}$ injects (sybil attack).

#### 4) Key confirmation

Only when all the $M_i$s derive the same key will all the members accept the key. Otherwise, there are three possible cases: (1) $K_{\mathcal{S}_k}(i) \neq K_{\mathcal{S}_k}(1)$, so that $M_i$ will decrypt to meaningless plaintext and abort. (2) $K_{\mathcal{S}_k}(i) = K_{\mathcal{S}_k}(1)$ but decrypts to "some differ". Then $M_i$ knows the association is not successful and will also abort. (3) $K_{\mathcal{S}_k}(i) = K_{\mathcal{S}_k}(1)$ and decrypts to "all patterns match", but $M_i$ receives a "fail alert". This indicates the attacker is successful in that some $K_{\mathcal{S}_k}(j) \neq K_{\mathcal{S}_k}(i) = K_{\mathcal{S}_k}(1)$, which means the group is split into more than one groups. For all the above cases, all the group members will detect and abort the operation. Note that, if a sensor node does not receive any confirmation from $M_1$ within a certain period, it will also alert the user.

In summary, group demonstrative identification is achieved

| Decomposition | Commu. | Comput. | LED blink. | Idle | Total |
|---|---|---|---|---|---|
| Time (ms) | 409 | 8905 | 15360 | 3187 | 27861 |
| Energy (mJ) | 24.5 | 48.1 | 1152 | 1.5 | 1226.1 |

*TABLE II:* Decomposition of overhead in GDP ($N$=10).

in GDP, in that the user is assured all legitimate member are in the final group and derive the same group key, while no attackers are included in the final group.

### *B. Security of Key Management*

#### *1) The polynomial based key distribution scheme*

This is ensured to be unconditionally secure and resists up to $t$ colluding attackers [25]. If more than $t$ polynomial shares are collected, $f(x, y)$ can be reconstructed using bivariate Lagrange interpolation. Therefore, we set $t$ as the maximum number of nodes in the BAN. For example, $t = 50$ is usually enough. In this case, even if all the sensors are compromised, $f(x, y)$ is secure and we can replace compromised nodes by new ones, as long as the total number of nodes is smaller than $t$.

#### *2) Forward secrecy*

It is infeasible to break the forward secrecy of all the keys, since it requires to break the pre-image resistance property of hash functions.

#### *3) Key update and revocation*

Because the value $\Delta(i+1)$ is randomly chosen from $\mathbb{F}_p$ and is encrypted thus is not known to non-legitimate members, an attacker can only guess it randomly. The success probability is $1/p$. For a revoked node $v$, without knowing $\Delta(i+1)$, even if it possesses $f_{v,i}(y)$, it cannot derive $f_{v,i+1}(y)$, therefore cannot obtain pairwise keys with any legitimate node.

## VII. Evaluation

### *A. Implementation*

We implemented GDP on a sensor network platform consisting of 10 Tmote-Sky nodes, each with 8MHz TI-MSP430 microcontroller, 10KB RAM and 48KB Flash (ROM). We let one of the sensor nodes be the controller, which does not improve the performance of GDP protocol. For preliminary experiments, We implement steps 3-20 in Fig. 4. The counting step is omitted, by programming the group IDs of sensor nodes and the group size into them in advance.

We convert the Diffie-Hellman based group key agreement (UDB) to its elliptic curve cryptography (ECC) version, where the modular exponentiation and modular multiplication correspond to point multiplication and point addition, respectively. We use the primitive operations provided by TinyECC [30], including point multiplication and point addition, with all optimizations enabled. To provide 80-bit key security, we derive a 160-bit group key; so the size of the finite field used is 160-bits. Then, $s = 160$. For implementation of the universal hash function in GDP, we employ a cryptographic hash function instead. Thus, 160-bit SHA-1 is used for both $H()$ and $\mathcal{H}()$ in GDP. Finally, $\rho = 16$.

### *B. Results*

In the following, we choose $n_{max} = 10$.

#### *1) Time required for sensor association*

In our experiments, $N \le n_{max}$. So we plot the time for one GDP run ($T_{gdp}(N)$) against the group size $N$ in Fig. 6. It can be seen that $T_{gdp}$ is almost constant (increases linearly but very slowly) when $N$ increases. This is because all nodes display LED blinking patterns simultaneously, while the computations are quite fast. $T_{gdp}$ consists of time spent in computation ($T_{cp}$), communication ($T_{cm}$) and human interaction ($T_I$). We then decompose $T_{gdp}$ in Table. II. For $\rho = 16$ bits, $T_I \approx 16$s (one bit for 1s). Obviously, the LED blinking time takes a major portion, and then the computation time, and finally the communications. The idle time is needed for nodes to wait to receive all other's broadcasts in each round and to resolve collisions.

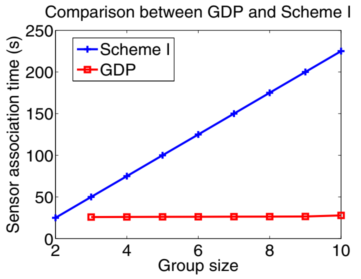

*Fig. 6:* Sensor association time.

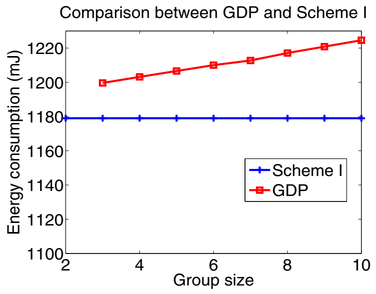

*Fig. 7:* Energy consumption per sensor node.

When $N > n_{max}$, the number of subgroups $k = \lceil \frac{N-1}{n_{max}-1} \rceil$. Then the total sensor association time $T_{gdp}(N) \approx (k-1)T_{gdp}(n_{max}) + T_{gdp}(N - k(n_{max} - 1))$, which increases linearly with $k$, and repeats the almost constant pattern when $N \le n_{max}$. The above time can be approximated theoretically, based on the experimental values $T_{gdp}(N), N \le n_{max}$. For $N = 20$, $T_{gdp} \approx$56s.

We also compare GDP with Scheme I, in which $T_{sc1}(N) = (N-1)T_{sc1}(2)$, where $T_{sc1}(2)$ is the estimated time for pairwise device pairing. From Fig. 6, $T_{sc1}(N)$ is linear with $N$. For $N = 20$, this is 475s. Obviously, when $N \ge 3$ the time of GDP is far less than Scheme I, which is also the case for [10] that uses one-by-one sensor association.

#### *2) Energy Consumption*

From the data sheet of Tmote, we obtain the normal voltage and current of the mote under different conditions, based on which we compute the energy consumption (EC). We plot the average EC for each sensor node in GDP against the group size ($N \le 10$) in Fig. 7, and compare it to the estimated EC of Scheme I (based on the EC break down for each primitive operation). The EC of GDP is a little higher than that of Scheme I, since it uses extra ECC point multiplication and addition operations. However, the difference is small (below

| | Comparison criteria | GDP | MiB [11] | KALwEN [12] | Keoh *et.al.* [10] | SAS-GAKA [19] |
|---|---|---|---|---|---|---|
| Security | Key secrecy, authenticity | √ | √ | √ | √ | √ |
| | Key confirmation | √ | √ | × | √ | × |
| | Exclusiveness | √ | √ | √ | √ | × |
| Usability | Fast batch deployment | √ | √ | √ | × | √ |
| | Error-proof | √ | √ | √ | √ | × |
| | # of human interactions | $k \ll N$ | / | / | $N$ | $N$ |
| | Human effort | L | M | M | H | M |
| Cost | Requires no PKI | √ | √ | √ | × | √ |
| | No additional hardware | √ | × | × | √ | √ |
| | No interface on sensors | √ | √ | √ | √ | × |
| | Involvement of PKC | L | NA | NA | M | H |

*TABLE III:* Comparison of GDP with related previous schemes: secure sensor association [10]; key deployment [11], [12]; group key agreement [19]. L: low; NA: none; M: medium; H: high.

50 mJ). Note that, for the controller, the EC of Scheme I is linear to $N$ which is much larger than that of GDP due to GDP's grouping mechanism.

Then we break down the EC of GDP in Table. II. It can be seen that the LED blinking takes major part in the EC, since its time is the longest and the required power is among the largest. Although the communication needs the largest power, it consumes the smallest energy since the time of it is quite small. Finally, note that the energy spent in computation is very small too, because the required power is small.

### *3) Usability*

GDP supports batch deployment. From the experiments, we found it is practical for a human to watch $n \leq 10$ LED blinking patterns simultaneously, when the nodes are put close to each other. The watch-and-compare is easy to follow, and differences can be identified with high probability. While MiB [11] and KALwEN [12] also achieve batch deployment, they require additional hardware (a faraday cage (FC), a keying device and a keying beacon). These devices add cost to the BAN and a FC is cumbersome to carry by the user. The SAS-GAKA [19] does not use additional device, however string comparison needs a user to remember strings which requires $N$ interactions. The results are summarized in Table. III. We do not compare with SPATE because it has different goals; notably, it requires $N$ comparisons of T-flags for each user.

## VIII. Conclusion

In this paper, we propose a novel protocol, group device pairing (GDP), for secure sensor association and key management in BAN. A group of nodes and a controller that may have never met before and share no pre-shared secrets, form a group securely to associate to the correct patient. For each subgroup, GDP achieves authenticated group key agreement by simultaneously and manually compare the LED blinking patterns on all nodes, which can be done within 30 seconds with enough security strength in practical applications. GDP helps the user of BAN to visually make sure that the BAN consists only of those nodes that s/he wants to associate with the patient. The resulting group keys enable efficient key management after network deployment. Experimental results show that GDP greatly reduces the total time and complexity of human interactions, while being efficient both in communication and computation.

**Acknowledgements.** This work was supported in part by the US National Science Foundation under grants CNS-0716306, CNS-0831628, CNS-0746977, and CNS-0831963.

## References

[1] K. Lorincz, D. Malan, T. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton, "Sensor networks for emergency response: challenges and opportunities," *IEEE Pervasive Computing*, vol. 3, no. 4, pp. 16–23, Oct.-Dec. 2004.

[2] M. Hanson, H. Powell, A. Barth, K. Ringgenberg, B. Calhoun, J. Aylor, and J. Lach, "Body area sensor networks: Challenges and opportunities," *Computer*, vol. 42, no. 1, pp. 58–65, Jan. 2009.

[3] E. Jovanov, A. Milenkovic, C. Otto, and P. C. de Groen, "A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation." *J Neuroengineering Rehabil*, vol. 2, no. 1, March 2005.

[4] M. Li, W. Lou, and K. Ren, "Secure device pairing," *in Encyclopedia of Cryptography and Security (2nd Ed.)*, H. Tilborg and S. Jajodia Ed., Springer, to appear, 2010.

[5] C.-H. O. Chen, C.-W. Chen, C. Kuo, Y.-H. Lai, J. M. McCune, A. Studer, A. Perrig, B.-Y. Yang, and T.-C. Wu, "Gangs: gather, authenticate 'n group securely," in *MobiCom '08*, 2008, pp. 92–103.

[6] O. Morchon, H. Baldus, and D. Sanchez, "Resource-efficient security for medical body sensor networks," in *BSN '06*, April 2006, p. 83.

[7] K. Malasri and L. Wang, "Addressing security in medical sensor networks," in *HealthNet '07*, 2007, pp. 7–12.

[8] C. C. Tan, H. Wang, S. Zhong, and Q. Li, "Body sensor network security: an identity-based cryptography approach," in *ACM WiSec '08:*, 2008, pp. 148–153.

[9] C. Poon, Y.-T. Zhang, and S.-D. Bao, "A novel biometrics method to secure wireless body area sensor networks for telemedicine and m-health," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 73–81, April 2006.

[10] S. L. Keoh, E. Lupu, and M. Sloman, "Securing body sensor networks: Sensor association and key management," *IEEE PerCom '09*, pp. 1–6, 2009.

[11] C. Kuo, M. Luk, R. Negi, and A. Perrig, "Message-in-a-bottle: user-friendly and secure key deployment for sensor nodes," in *SenSys '07*, 2007, pp. 233–246.

[12] Y. Law, G. Moniava, Z. Gong, P. Hartel, and M. Palaniswami, "Kalwen: A new practical and interoperable key management scheme for body sensor networks," in *BodyNets '09*, April 1-3 2009.

[13] F. Stajano and R. J. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *IWSP '00*, 2000, pp. 172–194.

[14] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong, "Talking to strangers: authentication in ad-hoc wireless networks," in *NDSS '02*, 2002.

[15] J. M. McCune, A. Perrig, and M. K. Reiter, "Seeing-is-believing: Using camera phones for human-verifiable authentication," in *IEEE S & P*, 2005, pp. 110–124.

[16] M. Cagalj, S. Capkun, and J.-P. Hubaux, "Key agreement in peer-to-peer wireless networks," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 467–478, Feb. 2006.

[17] S. Pasini and S. Vaudenay, "SAS-based Authenticated Key Agreement," in *Public Key Cryptography - PKC '06*, ser. LNCS, vol. 3958, 2006, pp. 395 – 409.

[18] Y.-H. Lin, A. Studer, H.-C. Hsiao, J. M. McCune, K.-H. Wang, M. Krohn, P.-L. Lin, A. Perrig, H.-M. Sun, and B.-Y. Yang, "Spate: small-group pki-less authenticated trust establishment," in *Mobisys '09*, 2009, pp. 1–14.

[19] S. Laur and S. Pasini, "SAS-Based Group Authentication and Key Agreement Protocols," in *Public Key Cryptography - PKC '08*, ser. LNCS, 2008, pp. 197–213.

[20] G. Ateniese, M. Steiner, and G. Tsudik, "New multiparty authentication services and key agreement protocols," *IEEE JSAC*, vol. 18, no. 4, pp. 628–639, Apr 2000.

[21] R. Prasad and N. Saxena, "Efficient device pairing using human-comparable synchronized audiovisual patterns," in *Applied Cryptography and Network Security (ACNS)*, ser. LNCS, 2008, pp. 328–345.

[22] S. Laur, N. Asokan, and K. Nyberg, "Efficient mutual data authentication using manually authenticated strings," in *Cryptology and Network Security*. Springer, 2005, pp. 90–107.

[23] P. Zimmermann, A. Johnston, and J. Callas, "Zrtp: Extensions to rtp for diffie-hellman key agreement for srtp draft-zimmermann-avt-zrtp-01," in *Internet-draft*, March. 2006.

[24] R. Dutta and R. Barua, "Provably secure constant round contributory group key agreement in dynamic setting," *IEEE Trans. on Inf. Theory*, vol. 54, no. 5, pp. 2007–2025, May 2008.

[25] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," in *CRYPTO '92*. Springer-Verlag, 1993, pp. 471–486.

[26] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[27] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 68–79, 1998.

[28] K. Van Laerhoven, A. Schmidt, and H.-W. Gellersen, "Multi-sensor context aware clothing," in *Wearable Computers, 2002. (ISWC 2002)*, 2002, pp. 49–56.

[29] S. Zhu, S. Setia, and S. Jajodia, "Leap: efficient security mechanisms for large-scale distributed sensor networks," in *CCS '03*, 2003, pp. 62–72.

[30] A. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," in *IPSN '08*, 2008, pp. 245–256.