Article

Efficient and Secure Pairing Protocol for Devices with Unbalanced Computational Capabilities

Xin Huang 1, Haotian Yin 1*, Xin Zhang 1*, Di Zhang 1, Sheng Chai 1, Bin Xing 1, Jie Zhang 2*, Xiaoling Yu 1, Yu Zhou 1 and Haixia Zheng 1,*

1 College of Data Science, Taiyuan University of Technology, Taiyuan 030024, China; xin@huangstudio.org (X.H.); yinhaotian1438@link.tyut.edu.cn (H.Y.); zhangxin072318@163.com (X.Z.); zhanggw0714@outlook.com (D.Z.); hzltbs@163.com (S.C.); xingbin1436@link.tyut.edu.cn (B.X.); yuxiaoling@tyut.edu.cn (X.Y.); zhouyu@tyut.edu.cn (Y.Z.)

2 Department of Computer Science and Software Engineering, Xi’an Jiaotong-Liverpool University, Suzhou 215123, China

* Correspondence: jie.zhang01@xjtlu.edu.cn (J.Z.); zhenghaixia@tyut.edu.cn (H.Z.)

Abstract: Wearable devices that collect data about human beings are widely used in healthcare applications. Once collected, the health data will be securely transmitted to smartphones in most scenarios. Authenticated Key Exchange (AKE) can protect wireless communications between wearables and smartphones, and a typical solution is the Bluetooth Secure Simple Pairing (SSP) protocol with numeric comparison. However, this protocol requires equivalent computation on both devices, even though their computational capabilities are significantly different. This paper proposes a lightweight numeric comparison protocol for communications in which two parties have unbalanced computational capabilities, e.g., a wearable sensor and a smartphone, named UnBalanced secure Pairing (UB-Pairing for short). The security of UB-Pairing is analyzed using the modified Bellare–Rogaway model (mBR). The analysis results show that UB-Pairing achieves the security goals. We also carry out a number of experiments to evaluate the performance of UB-Pairing. The results show that UB-Pairing is friendly to wearable devices, and more efficient than standard protocols when the computation capabilities of the two communication parties are highly unbalanced.

Keywords: device pairing; Secure Simple Pairing; Bluetooth 5.00; OOB channel; COVID-19

1. Introduction

Obtaining valuable information through big data analysis has become an important trend [1]. In particular, under the influence of Coronavirus Disease 2019 (COVID-19), much related data need to be collected, processed, and analyzed, such as datasets of nucleic acid test results, chest X-rays, and computed tomography scans [2,3]. Big data will be collected and processed in distributed or centralized servers through various terminal devices [4]. More broadly, in the field of healthcare, with the growing demand for health monitoring and vital sign tracking, the market for wearable devices has tripled in the last few years [5]. Nevertheless, the transmission of these data usually requires security mechanisms to prevent leakage [6]. Authenticated Key Exchange (AKE) negotiates shared secrets over insecure channels to protect the wireless communications [7]. In the abovementioned device-pairing scenarios, Bluetooth is a widely adopted technology [8–10], using three Secure Simple Pairing (SSP) protocols as its AKE solutions [11]. One of them is the numeric comparison protocol, which combines the human–computer interaction Out-Of-Band (OOB) channel and Elliptic Curve Cryptography (ECC) to assist authentication between two pairing devices. This method has been proven to be secure and easy to use [12].
However, the numeric comparison protocol assumes that the capabilities of the two communicating parties are “computationally balanced”. This is often not the case in Internet of Things (IoT) applications [13,14]. For example, a paired link is established between a wearable device equipped with a body temperature sensor and a hospital server.

Thus, this paper proposes a lightweight pairing protocol, named the UnBalanced secure Pairing (UB-Pairing for short) protocol, for devices with different computational capabilities. UB-Pairing is based on the OOB channels and specifically designed for unbalanced computational requirements. It can outperform the OOB-based protocols in terms of computational costs and has a much more comprehensive range of applications in IoT systems. The main contributions are summarized as follows.

- The UB-Pairing protocol is proposed and designed: it allows the powerful device to undertake a portion of the computing tasks on behalf of the wearable device and enables mutual authentication for devices with the help of OOB channels.
- The security of UB-Pairing is analyzed using the modified Bellare–Rogaway (mBR) model [15] via the computational No Reveal-mBR (cNR-mBR) game [16]. Analysis results show that UB-Pairing achieves the security goals of an AKE protocol. Moreover, we also identify some security vulnerabilities in parallel lightweight protocols.
- We demonstrate the advantages of UB-Pairing through experimental results and use cases in COVID-19 scenarios. Results demonstrate that UB-Pairing is friendly to wearable devices, and the overall performance of UB-Pairing is better than that of benchmark protocols.

The rest of this paper is organized as follows. In Section 2, related work is reviewed and discussed. In Section 3, preliminaries are introduced. In Section 4, the design of the UB-Pairing protocol is explained. In Section 5, security analysis is presented. In Section 6, a number of experiments are conducted and the performance of the UB-Pairing protocol is studied. In Section 7, a use case and corresponding application are simulated. Finally, in Section 8, the paper is concluded.

2. Related Work

Secure Simple Pairing (SSP) protocols can establish secure communications between devices. Depending on the input and output capabilities of the device, a series of Manual Authentication (MANA) protocols have been designed by Gehrmann, Mitchell, and Nyberg [17]. Vaudenay designed a protocol that requires only one message on the OOB channel and pointed out the efficiency problems of MANA protocols that require two OOB channel messages [18]. In [19], the protocol in [18] was improved by reducing one message sent over the wireless channel. In the same year, Laur and Nyberg proposed the three-round protocols MANA IV and MA-DH with a formal security analysis [20]. Later, a series of SSP protocols were proposed using a theory called hash commitment before knowledge [21–24], which forms the basis of our security analysis. In 2015, Nguyen and Leneutre [25] proposed a lightweight scheme for device pairing with only four messages. However, a brute-force attack was found by Khalfaoui et al. [26]. Later, in 2017, Taparia, Panigrahy, and Jena [27] proposed AKE based on a commitment scheme and authentication strings to withstand MITM (man-in-the-middle) attacks. Unfortunately, we found that their protocol had a design flaw leading to an MITM attack, which will be explained in the security discussion section. In recent years, Hou, Zhang, and Man tried to resist side-channel attacks in SSP protocols [28]. Groza et al. [29] used accelerometer data to establish OOB channels.

IEEE 802.15.6 employs the SSP protocols to establish initial trust in wearable applications [30]. In terms of security, Huang, Liu, and Zhang [31] identified that impersonation and MITM attacks were found against password-authenticated association protocol IEEE 802.15.6. Later, an improved protocol was proposed to solve the security defects [32], including off-line attacks and the lack of forward secrecy. In terms of light weight, Zhang, Xue, and Huang [33] proposed a lightweight version of the display association protocol in IEEE 802.15.6 for pervasive social networks.
Bluetooth also uses SSP as the authentication protocol in wearable applications [11]. Recently, researchers have carried out intensive research on the security of numeric comparison protocols [28,34,35] as well as the Passkey Entry protocols [36] in the Bluetooth standard. However, the requirement of being lightweight for these protocols has not been adequately studied. The proposed protocol aims to address this requirement. To evaluate the performance, we chose the numeric comparison protocol (named Pairing-Bluetooth) in the Bluetooth standard [11], the display-authenticated association protocol (named Pairing-IEEE 802.15.6) in IEEE 802.15.6 [30], and Zhang, Xue, and Huang’s protocol (named ZXH) [33] as the benchmark protocols. Pairing-Bluetooth and Pairing-IEEE 802.15.6 are two standard protocols that have been widely used in many applications; ZXH is a lightweight solution for IEEE 802.15.6.

3. Preliminaries

This section introduces the system model, security goals, security models, and relevant security mechanisms. Table 1 summarizes the notations used in this paper.

Table 1. Notations and corresponding descriptions.

<table>
<thead>
<tr>
<th>Notations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC</td>
<td>Elliptic curve group</td>
</tr>
<tr>
<td>G</td>
<td>A generator point of EC</td>
</tr>
<tr>
<td>q</td>
<td>The order of G</td>
</tr>
<tr>
<td>$\mathbb{Z}_q^*$</td>
<td>Prime finite field</td>
</tr>
<tr>
<td>$k$</td>
<td>Security parameter</td>
</tr>
<tr>
<td>$(SK_i, PK_i)$</td>
<td>Private and public key pair of device $i$</td>
</tr>
<tr>
<td>ID$_i$</td>
<td>Identity string of device $i$</td>
</tr>
<tr>
<td>$N_i, r_i$</td>
<td>Random number of device $i$</td>
</tr>
<tr>
<td>$K_i$</td>
<td>Shared secret of device $i$</td>
</tr>
<tr>
<td>$LK_i$</td>
<td>Session key (link key) of device $i$</td>
</tr>
<tr>
<td>MAC(_: ·)</td>
<td>Message authentication code function</td>
</tr>
<tr>
<td>MAC$<em>{16}(</em>: ·)$</td>
<td>16-bit short digest function</td>
</tr>
<tr>
<td>$\times$</td>
<td>Elliptic curve point scalar multiplication</td>
</tr>
<tr>
<td>$U_i$</td>
<td>Sum of temporary secret and long-term private key of device $i$</td>
</tr>
<tr>
<td>$T_i$</td>
<td>Sum of temporary secret and long-term public keys of device $i$</td>
</tr>
<tr>
<td>$f_1(·)$</td>
<td>Function that generates 128-bit commitment (used in Pairing-Bluetooth)</td>
</tr>
<tr>
<td>$f_2(·)$</td>
<td>Function that generates link key (used in Pairing-Bluetooth)</td>
</tr>
<tr>
<td>$f_3(·)$</td>
<td>Function that generates check values (used in Pairing-Bluetooth)</td>
</tr>
<tr>
<td>$g_1(·)$</td>
<td>Function that generates numeric check values (used in Pairing-Bluetooth)</td>
</tr>
<tr>
<td>DHKey</td>
<td>Diffie-Hellman key</td>
</tr>
<tr>
<td>$D_i$</td>
<td>Confirmation value on device $i$</td>
</tr>
<tr>
<td>$C_i$</td>
<td>Commitment value of device $i$</td>
</tr>
<tr>
<td>$mac_i$</td>
<td>Check value from device $i$</td>
</tr>
<tr>
<td>$M_i$</td>
<td>The $i$th message exchanged in protocol</td>
</tr>
<tr>
<td>$\oplus$</td>
<td>Bitwise XOR operation</td>
</tr>
<tr>
<td>$||$</td>
<td>Concatenation operation</td>
</tr>
<tr>
<td>$\rightarrow m$</td>
<td>Message $m$ is sent via wireless channel</td>
</tr>
<tr>
<td>$\rightarrow \rightarrow$</td>
<td>Message $m$ is sent via the OOB channel</td>
</tr>
</tbody>
</table>

3.1. System Model and Security Goals

Figure 1 shows the system model. UB-Pairing can be applied to scenarios where two parties with unbalanced computational capabilities intend to generate a shared link key through Bluetooth connections. Without loss of generality, we assume that a wearable device ($A$) intends to establish a common secret key with a relatively computationally powerful smartphone ($B$) via UB-pairing. Then, using the shared link key, the health data can be securely collected and transmitted to the smartphone according to the Bluetooth protocol stack [11].
User equipped with wearable device
UB-Pairing
Security manager protocol
Host
Host Controller Interface
Link layer
Physical layer
Smartphone

Figure 1. System model of wearable device pairing.

A and B can both display a 6-digit number and allow the users to enter confirmation signals (yes or no). If “yes” is entered on both devices, the pairing is successful. An attacker cannot alter, insert, delay, or delete the 6-digit numbers and confirmation signals.

UB-Pairing has two security goals:
• To be secure under passive eavesdropping—the adversary cannot infer any useful information from eavesdropping;
• To be secure against MITM attacks (active eavesdropping)—it prevents a third party from establishing independent contacts with both communication parties without being detected.

3.2. Security Model

Kudla and Paterson in [16] proposed modular security proofs and the mBR model. We use their cNR-mBR game to prove the security of UB-Pairing based on the mBR model.

3.2.1. cNR-mBR Game

A cNR-mBR game is a simulation of a protocol being attacked by an adversary. Kudla and Paterson mainly followed the spirit of the BR model [15,37] to construct the security game and removed the Reveal-query of the mBR model.

Formally, a protocol is denoted by $\Pi$, the security of which is modelled by a cNR-mBR game between an adversary $A$ and a challenger $C$. $k$ is the security parameter; $P$ is the set of participants; oracle $\Pi_{ij}^i$ is a session of $U$ with session number $i$; $SK_{U}$ is the private key of $U$, and $PK_{U}$ is the public key. The adversary $A$ is given all $\{PK_{U}\}$ and has access to any $\Pi_{ij}^i$ together with random oracles, which are set by $C$ to simulate the attack of $A$ using inquiries and responses:

• $Send(\Pi_{ij}^i, M): A$ sends an arbitrary message $M$ to the oracle $\Pi_{ij}^i$, and $\Pi_{ij}^i$ responds according to $\Pi$. If $M = \lambda$, the oracle $\Pi_{ij}^i$ initiates a protocol run with its partner $U'$ and $\{role_{ij} = \text{initiator}, role_{ij'} = \text{responder}\}$.
• $Corrupt(U): A$ obtains the private key $SK_{U}$ of $U$.
• $Test(\Pi_{ij}^i): \Pi_{ij}^i$ must be accepted, and its partner must not be queried by the Corrupt-oracle. Then, a random element in the key space of the protocol is returned to $A$.

3.2.2. cNR-mBR-Secure Protocol

For any adversary $A$, let $Advantage^E(k)$ be the probability that the session key $sk$ guessed by $A$ is equal to $sk_{\Pi_{ij}^i}$, where $\Pi_{ij}^i$ is the oracle responding to the Test query.

Definition 1. $\Pi$ is a cNR-mBR-secure protocol if:

• Two oracles running the protocol both accept holding the identical session key and session ID, and the session key is distributed uniformly on $\{0, 1\}^k$ for a benign adversary;
• $Advantage^E(k)$ is negligible for any adversary $A$ in the cNR-mBR game.

3.2.3. Partnership

The partnership between two participants is defined as follows.
Assume that adversary $A$ has a non-negligible advantage over protocol $\Pi$ in the cNR-mBR game. $\Pi$ has weak partnering if $A$ could cause any two oracles to accept holding the same session key when they are not partners. Otherwise, $\Pi$ has strong partnering.

3.3. Hardness Assumptions

Suppose that $EC$ is an elliptic curve, $G$ is its generator in the order of $q$, and $a, b, c \in \mathbb{Z}_q^*$.

- Computational Diffie–Hellman (CDH) assumption: given $a \times G, b \times G$, the advantage of computing $ab \times G$ in probability polynomial time is negligible.
- Decisional Diffie–Hellman (DDH) assumption: given $a \times G, b \times G, c \times G$, the advantage of determining whether $c \times G = ab \times G$ in probability polynomial time is negligible.
- Gap Diffie–Hellman (GDH) assumption: given $a \times G, b \times G$ and an oracle that solves the DDH problem, the advantage of computing $ab \times G$ in probability polynomial time is negligible.

3.4. Security Mechanisms

Several security mechanisms are worth mentioning here.

3.4.1. Commitment Scheme

A commitment scheme is defined as follows.

Definition 4. The following two algorithms constitute a commitment scheme:

- The probabilistic algorithm $\text{Commit}(\text{pub}, x) \rightarrow (c, t)$, where $x$ is the $n$-bit private value to be committed, $c$ is the commitment value, $t$ is the corresponding opening value, and $\text{pub}$ is some public value.
- The deterministic algorithm $\text{Open}(\text{pub}, c', t') \rightarrow x' \in \{1, 0\}^n \cup \{\bot\}$. This algorithm returns the $n$-bit private value $x'$ if the commitment $c'$ is valid, or $\bot$ otherwise.

The commitment scheme shall satisfy the following two properties:

- $(\epsilon_b, \tau_b)$-Hiding property, where $\epsilon_b$ is the upper bounded probability of the case where, given $(c, \text{pub})$, an adversary correctly guesses the private value $x$ without $t$, and $\tau_b$ is the corresponding running time of the adversary.
- $(\epsilon_v, \tau_v)$-Binding property, where $\epsilon_v$ is the upper bounded probability of the case where an adversary correctly opens a commitment $c$ of $x$ to a different private value $x'$, and $\tau_v$ is the corresponding running time of the adversary.

3.4.2. $\text{MAC}_{16}(\cdot)$ Function

The $\text{MAC}_{16}(\cdot)$ function is defined as follows.

Definition 5. $\text{MAC}_{16}(k, m) \rightarrow d$, where $k$ is the $l$-bit key, $m$ is the $n$-bit message, and $d$ is the 16-bit output. It satisfies two properties:

- $\epsilon_u$-(key-based uniformity) property: given $(m, d)$, $\Pr_{k \in \{0,1\}^l}[\text{MAC}_{16}(k, m) = y] = \epsilon_u$.
- $(\epsilon_u, \text{key-based uniformity})$ for any fixed $m$ and $y$, $\Pr_{k \in \{0,1\}^l}[\text{MAC}_{16}(k, m) = y] = \epsilon_u$. 

• \( \epsilon_r \)-property: given \((\theta, m \neq m')\), \( \Pr_{k \in \{0,1\}}\) \(\text{MAC}_{16}(k, m) = \text{MAC}_{16}(k \oplus \theta, m') \) = \( \epsilon_r \).

4. The Proposed Protocol

In this section, we first provide a brief review on the Pairing-Bluetooth in [11]; then, the UB-Pairing protocol design is explained in detail and its advantages are described afterwards.

4.1. Review of Pairing-Bluetooth

The Pairing-Bluetooth protocol is composed of four phases. Figure 2 shows the integrated process of Pairing-Bluetooth.

4.2. UB-Pairing

UB-Pairing transfers one scalar multiplication from the wearable device \(A\) to the more powerful smartphone \(B\). First, both \(A\) and \(B\) compute their private–public key pairs; then, they start the two authentication stages and, finally, the Link Key Calculation. The process is demonstrated in Figure 3. After these, \(A\) and \(B\) will share a fresh and secure session key (or link key [11]) \(LK_A = LK_B\).
**Figure 3.** UB-Pairing.

4.2.1. Initiation

Both A and B share the public parameters of elliptic curve EC, generator G, prime finite field $\mathbb{Z}_q^*$, and security parameter $k$. Then, A generates its private and public key pair $SK_A$ and $PK_A = SK_A \times G$; B generates its private and public key pair $SK_B$ and $PK_B = SK_B \times G$.

4.2.2. Authentication Stage 1

- A sends the message $M_1$ to B as (1):
  
  $$A \rightarrow B : M_1 = PK_A. \quad (1)$$

- Upon receiving $M_1$, B generates a random value $R_B \in \mathbb{Z}_q^*$ and computes $U_B$, $T_B$, and a commitment $C_B$ as (2)–(4):
  
  $$U_B = R_B + SK_B, \quad (2)$$
  
  $$T_B = U_B \times G, \quad (3)$$
  
  $$C_B = MAC(T_B, PK_A \parallel PK_B). \quad (4)$$
Then, $B$ sends the message $M_2$ to $A$ as (5):

$$B \rightarrow A : M_2 = \{ C_B, PK_B \}. \tag{5}$$

- Upon receiving $M_2$, $A$ generates a random value $R_A \in \mathbb{Z}_q^*$ and computes $U_A$ as (6):

$$U_A = R_A + SK_A. \tag{6}$$

$A$ sends the message $M_3$ to $B$ as (7).

$$A \rightarrow B : M_3 = U_A. \tag{7}$$

- Upon receiving $M_3$, $B$ computes digest $D_B$ as (8):

$$D_B = \text{MAC}_{16}(U_A, T_B \| PK_A \| PK_B). \tag{8}$$

$B$ then displays the six-digit decimal number converted from $D_B$. Meanwhile, $B$ sends the message $M_4$ to $A$ as (9):

$$B \rightarrow A : M_4 = T_B. \tag{9}$$

- Upon receiving $M_4$, $A$ verifies $C_B$ as (10):

$$\text{MAC}(T_B, PK_A \| PK_B) = C_B. \tag{10}$$

If the verification succeeds, $A$ computes a digest $D_A$ as (11):

$$D_A = \text{MAC}_{16}(U_A, T_B \| PK_A \| PK_B). \tag{11}$$

$A$ then displays the six-digit decimal number converted from $D_A$.

- The human user checks if $D_A = D_B$. If the digests are equal, the user confirms on each device. The two devices compute the shared key as (12) and (13):

  - $A$ computes

  $$K_A = R_A \times (T_B - PK_B)$$
  $$= R_A \times R_B \times G. \tag{12}$$

  - $B$ computes

  $$K_B = R_B \times (U_A \times G - PK_A)$$
  $$= R_A \times R_B \times G$$
  $$= K_A. \tag{13}$$

4.2.3. Authentication Stage 2

- $A$ firstly computes $mac_A$ as (14):

$$mac_A = \text{MAC}(K_A, U_A \| T_B \| A \| B). \tag{14}$$

Then, $A$ sends the message $M_5$ to $B$ as (15):

$$A \rightarrow B : M_5 = mac_A. \tag{15}$$

- $B$ firstly checks the following Equation (16):

$$\text{MAC}(K_B, U_A \| T_B \| A \| B) = mac_A. \tag{16}$$
When the verification succeeds, $B$ computes $mac_B$ as (17):

$$mac_B = MAC(K_B, T_B || U_A || B || A).$$

(17)

Then, $B$ sends the message $M_6$ to $A$ as (18):

$$B \rightarrow A : M_6 = mac_B.$$

(18)

- $A$ checks the following Equation (19):

$$MAC(K_A, T_B || U_A || B || A) = mac_B.$$  

(19)

4.2.4. Link Key Calculation

If the verification of $mac_B$ and $mac_A$ succeeds, $A$ and $B$ derive the link key $LK$ from the shared key as (20):

$$LK_A = MAC(K_A, A || B || PK_A || PK_B) = MAC(K_B, A || B || PK_A || PK_B) = LK_B.$$  

(20)

4.3. Advantages

The Bluetooth Core Specification [11] stipulates that the device should replace the public–private key pair in each round for protecting the device’s private key. This leads to recalculating a new public–private key pair every time. However, UB-Pairing can generate fresh shared secrets while using the long-term public key. Since UB-Pairing shifts the computational load (one scalar multiplication) from the wearable device to the smartphone, it significantly reduces the computational requirements of the wearable device.

The computational complexity required for UB-Pairing and related protocols is shown in Table 2, where $S_A$ and $S_B$ represent the computational complexity on $A$ and $B$, respectively. In Pairing-Bluetooth and Pairing-IEEE 802.15.6, the time complexity of $A$ and $B$ is the same, and both $A$ and $B$ undertake two scalar multiplications; meanwhile, in the ZXH and UB-Pairing protocols, the time complexity difference between $A$ and $B$ is mainly in scalar multiplication, where $A$ undertakes one scalar multiplication, and $B$ undertakes three scalar multiplications.

Table 2. Number of cryptography operations required on the two parties in UB-Pairing and related protocols.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>$S_A$</th>
<th>$S_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pairing-Bluetooth [11]</td>
<td>$2T_m + 3T_b + T_a$</td>
<td>$2T_m + 3T_b + T_a$</td>
</tr>
<tr>
<td>DPE [36]</td>
<td>$2T_m + 6T_h$</td>
<td>$2T_m + 6T_h$</td>
</tr>
<tr>
<td>SPEKE [29]</td>
<td>$2T_{mod} + lT_h$</td>
<td>$2T_{mod} + lT_h$</td>
</tr>
<tr>
<td>HDK [28]</td>
<td>$2T_m + 5T_b + T_a$</td>
<td>$2T_m + 5T_b + T_a$</td>
</tr>
<tr>
<td>Pairing-IEEE 802.15.6 [30]</td>
<td>$2T_m + 5T_h$</td>
<td>$2T_m + 5T_h$</td>
</tr>
<tr>
<td>ZXH [33]</td>
<td>$T_m + 3T_b + T_a$</td>
<td>$3T_m + 3T_b + T_a$</td>
</tr>
<tr>
<td>UB-Pairing</td>
<td>$T_m + 5T_b + T_a$</td>
<td>$3T_m + 5T_b + T_a$</td>
</tr>
</tbody>
</table>

Note: $T_h$: HMAC operation; $T_{add}$: elliptic curve point addition; $T_{m}$: elliptic curve point scalar multiplication; $T_x$: xor operation; $T_{mod}$: Modulo exponentiation; $l$ is typically larger than 56.

The ZXH protocol is based on IEEE 802.15.6 (which is not widely used as Bluetooth) and lacks explicit key confirmation. In addition, an inattentive user may confirm two unequal values, but ZXH is unable to detect this kind of human error (will be explained in the next section).

Note: Pairing-Bluetooth, Pairing-IEEE 802.15.6, and ZXH are selected as benchmark protocols in the experiments. The reason is that these three protocols are more efficient than other protocols, as shown in Table 2.
5. Security Analysis

We analyze the security of UB-Pairing in this section.

5.1. Formal Security Proof

To start, we first define a similar protocol, UB-Pairing', from UB-Pairing. UB-Pairing' is identical to UB-Pairing except for the session key: UB-Pairing' uses the string \( \{ K, A \parallel B \parallel PK_A \parallel PK_B \} \), while UB-Pairing uses \( \text{MAC}(K, A \parallel B \parallel PK_A \parallel PK_B) \).

**Theorem 1.** Suppose that there is an adversary \( \mathcal{A} \) to protocol UB-Pairing' that can win the cNR-mBR game with non-negligible probability \( \eta(k) \) in polynomial time \( \tau(k) \), where \( k \) is the security parameter of UB-Pairing'. The number of participants is \( n_P \) and the number of sessions that each participant may be involved in is \( n_S \), where \( n_P \) and \( n_S \) are both polynomial functions of \( k \). Then, the CDH problem can be solved with non-negligible probability \( \eta(k) \frac{1}{n_P n_S} \) within time \( \tau(k) \).

The proof of Theorem 1 is given in Appendix A.

Based on Definition 4 (commitment scheme) and Definition 5 (MAC\(_{16}(\cdot)\)), we redefine them here:

**Definition 6.** Commit\((\cdot)\) algorithm modelled by MAC\(_{Cmt}\)-oracle is a MAC\(_{Cmt}\)-query, where the input value is composed of the public keys and private data, and the output of MAC\(_{Cmt}\)-oracle is the commitment value \( c \). Then, the Open\((\cdot)\) algorithm in Definition 4 is a MAC\(_{Cmt}\)-query, too, where the output value is compared with the commitment value \( c \) held by the query executor. If the output value equals \( c \), \( c \) is a valid commitment; otherwise, \( c \) is an invalid commitment.

The commitment scheme has the following two properties:

- Suppose that there is a record \( \{ T_j, PK_I, PK_J, C_j \} \) in MAC\(_{Cmt}\)-list. \( \mathcal{A} \) queries \( C_j \) to Guess-oracle and obtains a \( T'_j \). The probability of \( T'_j = T_j \) is upper-bounded by \( \epsilon_b \) in time \( \tau_b \).
- Suppose that there is a record \( \{ T_j, PK_I, PK_J, C_j \} \) in MAC\(_{Cmt}\)-list. \( \mathcal{A} \) queries \( \{ T'_j, PK'_I, PK'_J \} \) to MAC\(_{Cmt}\)-oracle and receives a \( C'_j \), where \( \{ T'_j, PK'_I, PK'_J \} \neq \{ T_j, PK_I, PK_J \} \). The probability of \( C'_j = C_j \) is upper-bounded by \( \epsilon_b \) in time \( \tau_b \).

**Definition 7.** MAC\(_{16}(\cdot)\) modeled by MAC\(_{16}\)-oracle in the cNR-mBR model has the following properties:

- Suppose that there is a record \( \{ U_j, T_j, PK_I, PK_J, D_{ij} \} \) in MAC\(_{16}\)-list. \( \mathcal{A} \) generates a number \( U'_j \) and queries \( \{ U'_j, T_j, PK_I, PK_J \} \) to MAC\(_{16}\)-oracle to obtain a digest \( D'_{ij} \). The probability of \( D'_{ij} = D_{ij} \) is \( \epsilon_u \).
- Suppose that there is a record \( \{ U_j, T_j, PK_I, PK_J, D_{ij} \} \) in MAC\(_{16}\)-list. \( \mathcal{A} \) queries \( \{ U'_j, T'_j, PK'_I, PK'_J \} \) to MAC\(_{16}\)-oracle to obtain \( D'_{ij} \) where \( \{ U'_j, T'_j, PK'_I, PK'_J \} \neq \{ U_j, T_j, PK_I, PK_J \} \). The probability of \( D'_{ij} = D_{ij} \) is \( \epsilon_r \).

**Theorem 2.** Assume that MAC\((\cdot)\) and MAC\(_{16}(\cdot)\) are modelled by MAC\(_{Cmt}\)-oracle and MAC\(_{16}\)-oracle, respectively, and protocol UB-Pairing has strong partnering in the cNR-mBR model.

The proof of Theorem 2 is given in Appendix B.

Here, we draw Theorem 3 from [16]:

**Theorem 3.** Suppose that MAC\((\cdot)\) is a random oracle MAC\(_{LinkKey}\). If UB-Pairing uses a session key via the MAC\((\cdot)\) function and has strong partnering, and UB-Pairing' is probabilistic polynomial time reducible to the hardness of the CDH problem, and the session string decisional problem for UB-Pairing is polynomial time reducible to the DDH problem, then the mBR security of UB-Pairing is probabilistic polynomial time reducible to the hardness of the GDH problem of \( f \).

According to Theorems 1–3, UB-Pairing is secure in the mBR model if the CDH, DDH, and GDH assumptions hold.
5.2. Security Discussion

We discuss the security of UB-Pairing and other protocols [11,25,27,28,30,33] with regard to several security threats involving human users [38]. The results are summarized in Table 3.

Table 3. Security and functionality comparison of UB-Pairing and benchmark protocols.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
<th>F6</th>
<th>F7</th>
<th>F8</th>
<th>F9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pairing-IEEE 802.15.6 [30]</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Nguyen and Leneutre [25]</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>ZXH [33]</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>Taparia, Panigraphy, and Jena [27]</td>
<td>×</td>
<td>✓</td>
<td>√</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Hou, Zhang, and Man [28]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>UB-Pairing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Note: F1: resistance to MITM attacks; F2: resistance to relay attacks; F3: resistance to brute-force attacks; F4: inattentive or dishonest user; F5: rushing behavior; F6: user observation; F7: honest-but-curious party; F8: unbalanced computation; F9: formal security analysis; ✓: the protocol preserves the feature; ×: the protocol does not preserve the feature.

5.2.1. Man-in-the-Middle Attack

The attacker can modify the messages transmitted on the wireless channel. For example, to obtain the session key $L_K$, the attacker can replace the values of $U_A$, $PK_A$, and $PK_B$. Nevertheless, with the commitment scheme and OOB checking, the attacker must commit an unknown value, and pass the verification of honest party $A$ or the confirmation of the user who compares the two short numbers. In such cases, the attacker’s advantage is negligible. Therefore, UB-Pairing can resist MITM attacks.

However, in Taparia, Panigraphy, and Jena’s protocol [27], as shown in Figure 4 (where $D_a = m_a$ w.l.o.g.), an MITM attacker could replace $g^{X_b}$ with its own generated $g^c$ in message $m_b$. Since $A$ accepts $m_b$ without authentication, and furthermore, $S_a$ and $S_b$ exclude $g^{X_b}$, $A$ will then generate the shared secret key $K_a = (g^c)^{X_a} = g^{c_X_a} \mod p$. Therefore, the attacker could calculate the key $K_c = (g^{X_a})^c = g^{c_X_a} = K_a \mod p$. Hence, Taparia, Panigraphy, and Jena’s protocol [27] could not resist MITM attacks.

5.2.2. Replay Attack

The attacker may replay the messages transmitted in the previous sessions. In UB-Pairing, both $A$ and $B$ generate a fresh nonce or a random number and $DHKey$ in every session. Moreover, the attacker cannot reserve the secure hash function to obtain the random value. Thus, the attacker can only replay the commitment, while the human
user’s confirmation can prevent the session from proceeding to the next step. Therefore, UB-Pairing can resist replay attacks.

5.2.3. Brute-Force Attack

The attacker may use brute force to extract the values of nonce or random numbers from the hashes. In UB-Pairing, $R_A$ and $R_B$ are used as ephemeral secrets, the length of which is the same as the private keys (e.g., 160 bits). Therefore, the advantage of an attacker using brute force is negligible. Meanwhile, in Nguyen and Leneutre’s protocol [25], their exchanged commitment contained a short nonce used for subsequent OOB checking, but they ignored the ability of an attacker that could extract the nonce by brute-force search. If the nonce is revealed, the attacker could obtain the value used for OOB checking in advance, rendering the protocol insecure. Therefore, their protocol cannot resist the brute-force attack [26].

5.2.4. Inattentive or Dishonest User

The human user may be careless when confirming two different values. UB-Pairing needs key confirmation with the MAC exchanging after the careless user confirms, which is not given in ZXH [33] and Pairing-IEEE 802.15.6 [30]. In UB-Pairing, if any MAC verification ($mac_A$ or $mac_B$) fails, the honest party will abort. Therefore, UB-Pairing can tolerate an inattentive or dishonest user.

5.2.5. Rushing Behavior

The human user may forget the confirmation step. UB-Pairing cannot proceed without receiving the confirmation signals from the two negotiating parties, as the session key cannot be generated. Hence, the attacker cannot utilize the rushing behavior of human users to crack UB-Pairing.

5.2.6. User Observation

The attacker may learn the six-digit number through a hidden camera. $D_A$ and $D_B$ are computed from the messages transmitted on the wireless channel, on which the attacker may eavesdrop. Therefore, the attacker could compute the value of $D_A$ or $D_B$. However, the attacker cannot obtain $D_B$ or $D_A$ in advance; hence, observing the six-digit number does not pose a threat to the security of UB-Pairing.

5.2.7. Honest-but-Curious Party

The negotiating parties may be interested in obtaining some additional information from each other. In UB-Pairing, the additional information may be the long-term private keys. If $B$ wants to derive $A$’s private key $SK_A$ through $U_A$, it is no different from obtaining a random number. Moreover, $T_B$ gives no advantage to $A$ to derive the private key $SK_B$. Therefore, honest-but-curious parties cannot obtain additional information in UB-Pairing.

6. Performance Analysis

The performance of UB-Pairing is studied via a series of experiments. Details are elaborated below.

6.1. Setup

The experimental setup is shown in Table 4. Experiment I and III used a Raspberry Pi and a virtual machine, and Experiment II used two identical virtual machines. The CPU of the Raspberry Pi was 1.2 GHz ARM, and the CPU of the virtual machine was i7-6700HQ 2.6 GHz, and the programming language was Python. The Raspberry Pi was used as the wearable device $A$, with the virtual machine in a laptop as the powerful device $B$. The elliptic curves were P-192, P-224, P-256, P-384, and P-521, which are recommended by Federal Information Processing Standards (FIPS). The HMAC based on SHA-256 was used as the default MAC algorithm. The MAC$_{16}$ algorithm in experiments II, III, and IV was
implemented based on the MAC algorithm. The output of MAC_{16} was the first 16 bits of the MAC output. In the rest of the section, we use $T_y^x$ to represent the time of running $y$ on device $x$. For all experiments, we repeated them 10 times and used the average value as the final result.

Table 4. Experimental setup.

<table>
<thead>
<tr>
<th>Device</th>
<th>CPU</th>
<th>Programming Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expt. I</td>
<td>Raspberry Pi</td>
<td>1.2 GHz ARM</td>
</tr>
<tr>
<td></td>
<td>Virtual Machine</td>
<td>i7-6700HQ 2.6 GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Python</td>
</tr>
<tr>
<td>Expt. II</td>
<td>Virtual Machine</td>
<td>i7-6700HQ 2.6 GHz</td>
</tr>
<tr>
<td></td>
<td>Virtual Machine</td>
<td>i7-6700HQ 2.6 GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Python</td>
</tr>
<tr>
<td>Expt. III</td>
<td>Raspberry Pi</td>
<td>1.2 GHz ARM</td>
</tr>
<tr>
<td></td>
<td>Virtual Machine</td>
<td>i7-6700HQ 2.6 GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Python</td>
</tr>
</tbody>
</table>

Note: The operating systems of virtual machines are Ubuntu 16.04 32-bit; “Expt.” means “Experiment”.

The scope of the experiment and expected results are described below.

- Experiment I tested the computing time required for major cryptographic operations. The expected result: the computation time of scalar multiplication is much longer than that of other operations.
- Experiment II tested the computing time of UB-Pairing on two virtual machines. The expected result: UB-Pairing reduces the communication loads of one party.
- Experimental III tested the running time of UB-Pairing on devices with unbalanced computation capabilities. The expected result: the protocol running time of UB-Pairing is shorter than that of other protocols.

6.2. Experiment I

In experiment I, we evaluated the computing time of HMAC, CMAC, point addition, and scalar multiplication on a Raspberry Pi and a virtual machine. The elliptic curve used was P-192. The average computing time is shown in Table 5.

Table 5. Average computing time on Raspberry Pi and virtual machine in experiment I.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Raspberry Pi</th>
<th>Virtual Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMAC</td>
<td>0.0011805 s</td>
<td>0.000263 s</td>
</tr>
<tr>
<td>CMAC</td>
<td>0.0009493 s</td>
<td>0.0002671 s</td>
</tr>
<tr>
<td>Point addition</td>
<td>0.0000263 s</td>
<td>0.0000039 s</td>
</tr>
<tr>
<td>Scalar multiplication</td>
<td>0.0410498 s</td>
<td>0.0185570 s</td>
</tr>
</tbody>
</table>

- On both devices, $T_{\text{scalar multiplication}} \gg T_{\text{HMAC}} \approx T_{\text{CMAC}}$. This result verified that the computation time of scalar multiplication is much longer than the MAC computation time.
- On both devices, $T_{\text{scalar multiplication}} \gg T_{\text{point addition}}$. This result verified that the computation time of scalar multiplication is much longer than that of point addition.
- $T_{\text{A scalar multiplication}} \gg T_{\text{B scalar multiplication}}$. This result verified that computing a scalar multiplication on Raspberry Pi is much more time-consuming than that on the laptop.

Thus, we should try to reduce the time of scalar multiplication on the wearable devices.

6.3. Experiment II

In experiment II, we evaluated the computing time of UB-Pairing on two communication parties, $B1$ and $B2$, which were two virtual machines. The average computing time is shown in Figure 5. We can see that $T_{\text{UB-Pairing}}^A \ll T_{\text{UB-Pairing}}^B$. This result verified that

- UB-Pairing transferred one scalar multiplication from $B1$ to $B2$;
• The UB-Paring protocol can reduce the computational loads on one communication party, and it is more friendly to wearable devices. Thus, we can conclude that UB-Paring can reduce the computational loads on one communication party. It is more friendly to wearable devices.

![Figure 5. Average computing time on two virtual machines in experiment II.](image)

6.4. Experiment III

In this experiment, we implemented and evaluated the running and computing time of UB-Pairing on two communication parties: Raspberry Pi A and a virtual machine B in the laptop. Paring-Bluetooth, Pairing-IEEE 802.15.6, and ZXH were used as the benchmarks. Results are shown in Figure 6. Moreover, the communication overhead is shown in Table 6.

- On Raspberry Pi, from the perspective of the running time or computing time, $T_{A-UB-Pairing} \approx T_{A-ZXH} \ll T_{A-Paring-Bluetooth} \approx T_{A-Paring-IEEE 802.15.6}$.
- From the perspective of the total running time or computing time, $T_{UB-Pairing} \approx T_{ZXH} \ll T_{Paring-Bluetooth} \approx T_{Paring-IEEE 802.15.6}$.
- When the elliptic curve parameter changed from P-192 to P-521, $T_{Paring-Bluetooth} - T_{UB-Pairing}$ and $T_{Paring-IEEE 802.15.6} - T_{UB-Pairing}$ became larger.
- The differences between running time and computing time are small. In other words, the dominant factor that influences the protocol performance is the time of computing scalar multiplications.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Wireless Messages (bits)</th>
<th>OOB Messages (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pairing-IEEE 802.15.6 [30]</td>
<td>2624</td>
<td>16</td>
</tr>
<tr>
<td>ZXH [33]</td>
<td>2176</td>
<td>16</td>
</tr>
<tr>
<td>UB-Pairing</td>
<td>1728</td>
<td>16</td>
</tr>
</tbody>
</table>

Note: Under curve P-192, the communication payloads of an identity, elliptic curve point, random number, MAC, and MAC$_{16}$ are 160, 384, 192, 128, and 16 bits, respectively.
Figure 6. Average protocol running and computing time of UB-Pairing and benchmark protocols on Raspberry Pi and virtual machine in experiment III.

In summary, the results verified that UB-Pairing performs better than Pairing-Bluetooth and Pairing-IEEE 802.15.6 when the computational capabilities of the two devices are unbalanced, and UB-Pairing is more friendly to wearable devices. The performance of UB-Pairing and ZXH is almost the same; however, UB-Pairing is more secure (see Table 3), and has more application scenarios since Bluetooth has been widely used.

7. Use Case and Advantages

This section illustrates the usages and advantages of UB-Pairing via a use case in emergency scenarios. It verifies whether UB-Pairing is more applicable for emergency scenarios than symmetric-AKE protocols [15] and general asymmetric encryption-based AKE protocols [39].

7.1. Application of UB-Pairing

In temporary mobile cabin hospitals and nucleic acid PCR testing areas (Figure 7), the public key infrastructure is not always available, and information sharing between devices must be carried out in ad hoc ways. The following steps explain how to use UB-Pairing.

Suppose that there are two characters, Alice and Bob, in our scenario. Their devices should support UB-Pairing. In particular, Alice’s device is a computationally limited device, e.g., a mobile phone A, and Bob’s is a powerful device, e.g., a server B. Moreover, both A and B should support encryption/decryption algorithms such as AES and MAC algorithms such as HMAC.

Suppose that there are N devices in emergency scenarios.

- N1 represents the number of devices in N that have master keys (mk) with each other. When devices A and B have mk with each other, the symmetry-AKE protocol in [15] will be run. The probability that the symmetric-AKE protocol is successfully executed is \( P_1 = \left( \frac{|N_1|}{|N|} \right)^2 \).

- N2 represents the number of devices in N that have public keys pk (and no mk) with each other. When devices A and B have public keys with each other, the TLS-AKE protocol in [39] will be run. The probability of the TLS-AKE protocol being successfully executed is \( P_2 = \left( \frac{|N_2|}{|N|} \right)^2 \).

- N3 (N4, N5, or N6) represent the number of devices that have neither mk nor pk. In the situation wherein devices have neither master key nor public key, we will run UB-Pairing (N3), ZXH (N4), Pairing-Bluetooth (N5), or Pairing-IEEE 802.15.6 (N6). The probability of the protocol being successfully executed is \( P_3 = P_4 = P_5 = P_6 = 1 - P_1 - P_2 = 1 - \left( \frac{|N_1|}{|N|} \right)^2 - \left( \frac{|N_2|}{|N|} \right)^2 \).

When A and B generate the link key, they can use it to send messages securely.
7.2. Advantages

Here, we analyze the advantages of UB-Pairing. The availability results are shown in Figures 8 and 9. We find that the connection probability of UB-Pairing ($P_3$) is roughly the same as that of the ZXH protocol ($P_4$), Pairing-Bluetooth ($P_5$), and Pairing-IEEE 802.15.6 ($P_6$), and better than the case in which devices only have shared mk and pk. In addition, when the proportion of $N_3$ ($N_4$ or $N_5$ or $N_6$) increases, the advantages of UB-Pairing, ZXH, Pairing-Bluetooth, or Pairing-IEEE 802.15.6 are much more obvious.

However, UB-Pairing is much more efficient than Pairing-Bluetooth and Pairing-IEEE 802.15.6. Although the efficiency of UB-Pairing and ZXH is almost the same, the use of Bluetooth is more extensive than that of IEEE 802.15.6, and UB-Pairing is more secure than ZXH (see Table 3).

In the extreme case, when protocols run between devices with unbalanced computational capabilities in experiment III, the accumulated authentication time is as shown in Figure 10. In the case in which the elliptic curve is P-521, $P_3/P_4/P_5/P_6 = 90\%$, and $N = 100/540/1000$, UB-Pairing can save 16 min more than Pairing-Bluetooth and 14 min more than Pairing-IEEE 802.15.6. This is extremely important for emergency medical treatment.
8. Conclusions

Secure Simple Pairing (SSP) protocols are simpler and more elegant than PKI (Public Key Infrastructure)-based solutions; thus, they have become useful in applications in the mobile computing era. Many researchers have proposed SSP protocols in the past few years, and MANA protocols are one of the representative SSP protocols [17–20]. Bluetooth [11] and IEEE 802.15.6 [30] also use SSP as their authentication protocols, and some improved versions have been proposed [28,32–36]. Currently, the security, performance, and usability of SSP protocols are still the main research questions.

This paper presented the design of the UB-Pairing protocol, which is specifically proposed for improving the protocol performance where the two communicating parties have unbalanced computational capabilities. The security of UB-Pairing was analyzed using the modified Bellare–Rogaway (mBR) model via the computational No Reveal-mBR (cNR-mBR) game. The analysis results showed that UB-Pairing achieves the security goals of an AKE protocol. Experimental results showed that UB-Pairing is more friendly to wearable devices and more efficient than standard protocols [11,30] when the computation capabilities of the two communication parties are highly unbalanced. Moreover, compared
with ZXH [33], UB-Pairing is more secure and has more application scenarios since Bluetooth has been widely used. Further comparison regarding the performance and security can be found in Tables 2 and 3, and experimental results in Section 6.

In the next few years, when healthcare applications based on Blockchain [40] and Metaverse [41] enter the market, more and more Bluetooth devices will be used to generate health data, and UB-Pairing can play an important role in these applications since it performs better than standard protocols in many cases.

UB-Pairing is the authentication procedure of the whole Bluetooth protocol; thus, if the Bluetooth protocol is used for transmitting large files, the performance improvement is not as obvious as transmitting many small files. In the future, we will study more lightweight AKE protocols, as well as lightweight secure transmitting protocols, which can cover more scenarios.

Author Contributions: Conceptualization, methodology, X.H. and J.Z.; validation, formal analysis, X.H., H.Y., and D.Z.; software, data curation, X.Z. and D.Z.; investigation, S.C., B.X., X.Y., Y.Z., and H.Z.; writing—original draft preparation, X.H. and J.Z.; writing—review and editing, H.Y., X.Z., and D.Z.; funding acquisition, X.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by the Shanxi Scholarship Council of China (No. 2021-038), the Applied Basic Research Project of Shanxi Province (No. 20210302123130, No. 20210302124273), Scientific and Technological Innovation Project of Universities in Shanxi Province (No. 20211038), National Natural Science Foundation of China (No. 62002255, No. 62002296), the Natural Science Foundation of Jiangsu Province (No. BK20200250), and the XJTLU Key Programme Special Fund (No. KSF-E-54).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

- AKE: Authenticated Key Exchange
- SSP: Secure Simple Pairing
- mBR: modified Bellare–Rogaway
- cNR-mBR: computational No Reveal-mBR
- ECC: Elliptic Curve Cryptography
- OOB: Out-Of-Band
- IoT: Internet of Things
- CDH: Computational Diffie–Hellman
- DDH: Decisional Diffie–Hellman
- GDP: Gap Diffie–Hellman
- MAC: Message Authentication Code

Appendix A

Proof of Theorem 1. We construct a challenger C from A. C is a CDH problem solver: given \((SK_A \times G, SK_B \times G)\), C computes \(SK_A \times SK_B \times G\), where \(SK_A, SK_B \in Z_q^*\). C sets the following preparation:

- For the participant B, C sets B's public key as \(PK_B = SK_B \times G\); for the participant A, C sets A's public key as \(PK_A = SK_A \times G\).
- For other participants P, C generates P's private keys as \(SK_P \in Z_q^*\) and sets their public keys as \(PK_P = SK_P \times G\).
• All public keys are sent to $A$.

Suppose that $\{I, J\}$ are two arbitrary participants. $C$ picks a session number $t \in \{1, \ldots, n_S\}$, starts $A$, and answers $A$’s queries as follows:

Send($\Pi_t^I$; $M$):
- If $M = C_B$, $I = A$ and role$_A$ = initiator, $C$ picks a random number $U_A \in Z_q^n$ and sends $U_A$ to $A$.
- If $M = U_A$, $\Pi_I^1 = \Pi_B^1$, and role$_B$ = responder, $C$ returns $T_B = PK_B + PK_B$.
- Otherwise, $C$ acts according to the UB-Pairing protocol.

Corrupt($I$): If $I = A$ or $I = B$, then $C$ aborts; otherwise, $C$ returns $SK_I$ to $A$.

$MAC_{Cmt}(T_I, PK_I, PK_J)$: If $\{T_I, PK_I, PK_J, C_I\}$ is in $MAC_{Cmt}$-list, $C$ returns with $C_I$; otherwise, $C$ randomly chooses $C'_I \in \{0, 1\}^{256}$, returns with $C'_I$, and adds $\{T_I, PK_I, PK_J, C'_I\}$ to $MAC_{Cmt}$-list.

$MAC_{LinkKey}(K_I, I, J, PK_I, PK_J)$: If $\{K_I, I, J, PK_I, PK_J, LK_I\}$ is in $MAC_{LinkKey}$-list, $C$ returns with $LK_I$; otherwise, $C$ randomly chooses $LK'_I \in \{0, 1\}^{256}$, returns with $LK'_I$, and adds $\{K_I, I, J, PK_I, PK_J, LK'_I\}$ to $MAC_{LinkKey}$-list.

$MAC_{16}(U_I, T_I, PK_I, PK_J)$: If $\{U_I, T_I, PK_I, PK_J, D_{IJ}\}$ is in $MAC_{16}$-list, $C$ returns with $D_{IJ}$; otherwise, $C$ randomly chooses $D'_{IJ} \in \{0, 1\}^{16}$, returns with $D'_{IJ}$, and adds $\{U_I, T_I, PK_I, PK_J, D'_{IJ}\}$ to $MAC_{16}$-list.

Guess($C$): $C$ returns a random $T$ on $A$.

Test($\Pi^I_t$): $C$ returns with a random string in the form of $\{K, A \mid B \mid PK_A \mid PK_B\}$.

The probability that $A$ queries the oracle $\Pi^I_t$ for the test session and $pid_A = B$ is $\frac{1}{n^p \cdot n^S}$.

Suppose that $A$ outputs a valid session key in the form of $\{K, A \mid B \mid PK_A \mid PK_B\}$ with probability $\eta(k)$. In this case,

$$
K = R_B \times (U_A \times G - PK_A)
= R_B \times U_A \times G - R_B \times PK_A
= R_B \times U_A \times G - SK_B \times PK_A
= R_B \times U_A \times G - SK_B \times SK_A \times G
= R_B \times U_A \times G - SK_A \times SK_B \times G
= SK_B \times U_A \times G - SK_A \times SK_B \times G
= SK_A \times U_A \times G - SK_A \times SK_B \times G
= U_A \times SK_B \times G - SK_A \times SK_B \times G
= U_A \times SK_B \times G - SK_A \times SK_B \times G
$$

Thus, $C$ can output $SK_A \times SK_B \times G = U_A \times PK_B - K$ as the solution of the CDH problem with non-negligible probability $\eta(k) \frac{1}{n^p \cdot n^S}$ within time $\tau(k)$.  \(\square\)

### Appendix B

**Proof of Theorem 2.** We use $M_{SG}$ to indicate the message sent by $A$ or $B$. $\hat{M}_{SG}$ is the message transmitted by $A$, which may not be equal to $M_{SG}$.

Suppose that $\Pi^I_A$ and $\Pi^I_B$ are two legal oracles, and $A$ could make oracles $\Pi^s_A$ and $\Pi^s_C$ accept holding the same session key but without being partners. In the end, $\Pi^s_A$, $\Pi^s_B$, and $\Pi^s_C$ have accepted; $\Pi^s_A$ and $\Pi^s_B$ hold $\{K_A, A \mid B \mid PK_A \mid PK_B\}$ and $\{K_C, A \mid B \mid PK_A \mid PK_B\}$, respectively. To obtain the same session key, $\Pi^s_A$ and $\Pi^s_C$ must make the same $MAC_{LinkKey}$ query. This means $K_A = K_C$.  \(\square\)
Let $S$ denote event “$A$ succeeds”:

$$S = \{S_1 \land S_2\},$$  \hspace{1cm} (A2)

$$S_1 = \{A-B-C\text{-accept} \land K_A = K_C\},$$  \hspace{1cm} (A3)

$$S'_1 = \{A-B\text{-accept}\},$$  \hspace{1cm} (A4)

$$S_2 = \{\text{Not-partners}\},$$  \hspace{1cm} (A5)

where $\text{Not-partners}$ refers to the event in which $A$ and $B$ are not partners, $A-B\text{-accept}$ refers to the event in which $A$ and $B$ have successfully verified $\tilde{C}_B$ (Event $X_C$), digests (Event $X_D$), $\tilde{m}_A$ and $\tilde{m}_B$ (Event $X_M$), and $A-B-C\text{-accept}$ refers to the event in which $A$, $B$, and $C$ have successfully verified $\tilde{C}_B$, digests, $\tilde{m}_A$ and $\tilde{m}_B$. With (A2)–(A5), then we have

$$\Pr[S] = \Pr[S_1 \land S_2]$$

$$\leq \Pr[S'_1 \land S_2]$$

$$\leq \Pr[W],$$  \hspace{1cm} (A6)

where $W = \{X_C \land X_D \land S_2\}$. In addition, let

$$\text{view}_A = \{PK_A, \tilde{PK}_B, \tilde{C}_B\},$$  \hspace{1cm} (A7)

$$\text{view}_B = \{\tilde{PK}_A, PK_B, C_B\},$$  \hspace{1cm} (A8)

$$m_A = \{PK_A, \tilde{PK}_B\},$$  \hspace{1cm} (A9)

$$m_B = \{\tilde{PK}_A, PK_B\}.$$  \hspace{1cm} (A10)

**Lemma A1.** Suppose $\text{view}_A = \text{view}_B$ (Event $V$), $\Pr[W|V] \leq \epsilon_b + \epsilon_h + \epsilon_r$.

**Proof of Lemma A1.** There are the following cases:

Event $V_{1.1}$: $\tilde{T}_B = T_B$:

- Event $V_{1.1.1}$: $A$ knows $T_B$ before $B$ sends $T_B$, breaking the hiding property with probability $\epsilon_h$;
- Event $V_{1.1.2}$: $A$ does not know $T_B$ before $B$ sends $T_B$. $A$ can adjust $U_A$ and make $D_A = D_B$ with probability $\epsilon_r$.

Event $V_{1.2}$: $\tilde{T}_B \neq T_B$, breaking the binding property with probability $\epsilon_b$.

From the above analysis,

$$\Pr[W|V] = \Pr[W|(V_{1.1} \lor V_{1.2})]$$

$$= \Pr[W|V_{1.1}] + \Pr[W|V_{1.2}]$$

$$\overset{\text{independent}}{=} \Pr[W|V_{1.1}] + \Pr[W|V_{1.1} \lor V_{1.2}] + \Pr[W|V_{1.2}]$$

$$\overset{\text{independent}}{=} \Pr[W|V_{1.1}] + \Pr[W|V_{1.1}] + \Pr[W|V_{1.2}]$$

$$\leq \epsilon_b + \epsilon_h + \epsilon_r.$$  \hspace{1cm} (A11)

where $\text{independent}$ means that two events are independent (e.g., $V_{1.1}$ and $V_{1.2}$). Lemma A1 is proven. $\Box$

**Lemma A2.** Suppose $\text{view}_A \neq \text{view}_B$ (Event $\tilde{V}$), $\Pr[W|\tilde{V}] \leq 2\epsilon_b + 2\epsilon_h + 2\epsilon_r$.

**Proof of Lemma A2.** There are the following cases:

Event $V_{2.1}$: $\tilde{C}_B = C_B$ but $m_A \neq m_B$:

- Event $V_{2.1.1}$: $\tilde{T}_B \neq T_B$, the binding property is broken with probability $\epsilon_b$;
- Event $V_{2.1.2}$: $\tilde{T}_B = T_B$, the binding property is broken with probability $\epsilon_b$;

Event $V_{2.2}$: $\tilde{C}_B \neq C_B$ but $m_A = m_B$:
• Event $V_{2.2.1}$: $\hat{T}_B = T_B$: It means that with the same queries, $MAC_{Cmt}$-oracle returns different output. The probability of this case is negligible;

• Event $V_{2.2.2}$: $\hat{T}_B \neq T_B$:
  - Event $V_{2.2.2.1}$: $A$ knows $T_B$ before $B$ sends $T_B$, breaking the hiding property with probability $\epsilon_b$;
  - Event $V_{2.2.2.2}$: $A$ does not know $T_B$ before $B$ sends $T_B$. $A$ can adjust $U_A$ and make $D_A = D_B$ with probability $\epsilon_r$.

Event $V_{2.3}$: $\hat{C}_B \neq C_B$ and $m_A \neq m_B$:

• Event $V_{2.3.1}$: $A$ knows $T_B$ before $B$ sends $T_B$, breaking the hiding property with probability $\epsilon_b$;

• Event $V_{2.3.2}$: $A$ does not know $T_B$ before $B$ sends $T_B$. $A$ can adjust $U_A$ and make $D_A = D_B$ with probability $\epsilon_r$.

As a reminder, the situation $\{\hat{C}_B = C_B, m_A = m_B\}$ has already been discussed in Lemma A1.

Thus,

$$\Pr[W|\bar{V}] = \Pr[W|V_2.1 \lor V_2.2 \lor V_{2.3}]$$

$$\equiv \Pr[W|V_2.1] + \Pr[W|V_2.2] + \Pr[W|V_{2.3}]$$

$$= \Pr[W|V_2.1,1 \lor V_2.1,2] + \Pr[W|V_2.2,1 \lor V_2.2,2] + \Pr[W|V_{2.3,1} \lor V_{2.3,2}]$$

$$\equiv \Pr[W|V_2.1,1] + \Pr[W|V_2.1,2] + \Pr[W|V_2.2,1] + \Pr[W|V_2.2,2]$$

$$\leq \epsilon_b + \epsilon_b + 0 + \Pr[W|V_{2.2.2}] + \epsilon_h + \epsilon_r$$

$$= 2\epsilon_b + \epsilon_h + \epsilon_r + \Pr[W|V_{2.2.2.1} \lor V_{2.2.2.2}]$$

$$\equiv 2\epsilon_b + \epsilon_h + \epsilon_r$$

in Lemma A2 is proven. ☐

In summary, from Lemmas A1 and A2, and Equations (A11) and (A12), we can obtain

$$\Pr[S] \leq \Pr[W]$$

$$\equiv \Pr[W|V] \Pr[V] + \Pr[W|\bar{V}] \Pr[\bar{V}]$$

(a)

$$\leq \Pr[W|V] + \Pr[W|\bar{V}]$$

(A13)

(b)

$$\leq \epsilon_b + \epsilon_h + \epsilon_r + 2\epsilon_b + 2\epsilon_h + 2\epsilon_r$$

$$= 3\epsilon_b + 3\epsilon_h + 3\epsilon_r$$

where \texttt{tpp} means the total probability principle, operation \texttt{(a)} omits two probabilities less than \(1 (\Pr[V] \text{ and } \Pr[\bar{V}])\), and operation \texttt{(b)} substitutes the conclusions of Lemmas A1 and A2.

$A$ makes two oracles accept while they are not partners with each other with advantage $3\epsilon_b + 3\epsilon_h + 3\epsilon_r$. UB-Pairing has strong partnering. Theorem 2 is proven. ☐
References


32. Zhang, J.; Huang, X.; Craig, P.; Marshall, A.; Liu, D. An improved protocol for the password authenticated association of IEEE 802.15. 6 standard that alleviates computational burden on the node. Symmetry 2016, 8, 131. [CrossRef]


