Abstract—Mobile healthcare social networks (MHSNs) have arisen as a very promising brandnew healthcare system, which will greatly improve the quality of life. Moreover, with the help of Software Defined Networking (SDN) paradigm, it can enhance the user experience. To achieve personal health information (PHI) sharing and the access control among parities, a similar symptoms matching process should be executed before that. However, the matching process requires users to exchange symptoms information, conflicting with the ever-increasing privacy concerns on protecting private symptoms from strangers. To realize privacy-preserving symptoms matching, in this paper, we design two blind signature-based symptom matching schemes in SDN-based MHSNs, which can achieve the coarse-grained symptom matching and fine-grained symptom matching, respectively. Moreover, our schemes do not relay on any Trusted Third Party (TTP). Security analysis and detailed simulations show that our proposed schemes can realize efficient privacy-preserving symptom matching. Finally, we do comprehensive experimental evaluation on real-world smartphones to demonstrate the practicality of our proposed schemes.

Index Terms—Mobile healthcare social networks; Symptoms matching; Coarse/Fine-grained; Blind signature

I. INTRODUCTION

With the rapid expanding of mobile computing, sensor systems, and wireless communication technologies, MHSNs [2, 3] have attracted plenty of interest in both academia and industry. Different from the conventional electronic healthcare system [4], patients can enjoy full freedom moving outside only by wearing body sensors to keep monitoring their PHI in MHSNs [5]. The monitored PHI can then be aggregated using smartphones via ZigBee or bluetooth and transmitted to the remote healthcare center subsequently via heterogeneous wireless networks. Besides, with the help of SDN architecture [6], the flexibility and performance of data flows between smartphones and healthcare center can be improved. With this method, the conventional centralized e-health system is improved into a decentralized and self-organized way which enables the recognition, searching and social interaction of the authorized mobile patients (i.e., those having similar symptoms and creating a social group).

In such mobile social networks [7, 8], patients, who have the same health conditions, are more likely to share their medical experience for mutual support and comfort, or support opportunistic computing [9]. Nevertheless, concerns on security and privacy are the stumbling blocks which should be cleared before the whole realization. As sensitive information is contained in PHI, patients might only want the medical users having some similar symptoms to participate in the sharing operation [10]. Thus, a major challenge is how to achieve privacy-preserving symptom matching process before establishing social interaction with other patients in MHSNs. Besides, as smartphones are also used for other applications in addition to healthcare monitoring, the cost and efficiency should both be considered for the proposed scheme. More specifically, the privacy-preserving. More specifically, the privacy-preserving symptom matching scheme should be efficient and time-saving [9].

To achieve this purpose, researchers present their schemes [9–12] with different methods to protect the privacy during the symptom matching process. Apart from that, privacy-preserving profile matching schemes seem to be a good alternative solution which help a user to find other users whose profiles best match his/hers without breaching their privacy [1, 13–21]. However, most of these schemes employ the cryptography tools which incur large overhead and are not practical for mobile devices.

Motivated by the above issues, in this paper, we propose blind signature-based privacy-preserving symptom matching schemes for SDN-based MHSNs. Our proposed schemes can support the coarse-grained matching (without the degree of a certain symptom) and fine-grained matching. The basic idea of our schemes is that the same symptom has the same signature, and we use RSA-based blind signature to protect the privacy during the matching process. The main contributions of our work are listed as follows:

- We propose two RSA-based blind signature schemes to achieve the coarse-grained and fine-grained privacy-preserving symptom matching, respectively, without any TTP. In the coarse-grained symptom matching, we use the RSA-based blind signature and Bloom filter to achieve the efficient and privacy-preserving symptom matching; while in the fine-grained symptom matching, we use RSA-based blind signature to privately compute the similarity degree of two users according to their weighted Euclidean distance.

- We provide the security and performance analysis for our schemes, which indicate that our schemes can achieve the privacy-preserving goal with a higher efficiency. Moreover, we design APPs based on our schemes and demonstrates the practicality of our solutions.

A part of this work [1] was accepted by ICC 2015. The new technical contribution is that we extend our scheme by introducing another blind signature-based symptoms matching scheme. Moreover, we design APPs according to our schemes to attest to the practicality.
The rest of this paper is organized as follows. Section II introduces the related work. Section III illustrates our system model and preliminaries. The details of our proposed scheme are discussed in the following Section IV. Section V and Section VI provide the security and performance analysis, respectively. Section VII presents the detailed process of the implementation of our schemes. Finally, Section VIII brings our work to a conclusion.

II. RELATED WORK

Chen et al. [10] adopt symptom matching method to design session keys for secure communication. In their scheme, they use the predicate encryption to hide symptoms and patients with the symptom are able to generate the session key to get private data. Lu et al. [9] propose a privacy-preserving opportunistic computing framework for mobile-healthcare emergency situation. They adopt Privacy-Preserving Scalar Product Computation (PPSPC) to achieve private matching between medical users. Based on [9], Zhu et al. [20] propose an efficient confusion matrix transformation algorithm to achieve secure and efficient fine-grained matching. Then, Luo et al. [21] extend [20] to support multi-hop profile matching protocol. In their scheme, users can customize their own matching preference, which makes the matching results more precise. Xing et al. [11] adopt homomorphic encryption to present a privacy-preserving symptoms matching scheme to thwart inflation attack. He et al. [12] present a new framework for the handshake scheme by hierarchical identity-based cryptography in MHSNs. Besides, they design an efficient Cross-Domain HandShake scheme to achieve the symptom matching.

To achieve privacy-preserving profile matching, a TTP is employed in existing schemes [22–25]. In their schemes, each user submits his/her symptoms to the TTP, which works as a matching center to compute the similarity among users and returns the best match result to each user. However, the TTP may have the problems of single point of failure and performance bottleneck. To remove the TTP, Private Set Intersection (PSI)-based approaches [13–15, 26, 27] are proposed. PSI allows two parties, each with its own set, to privately compute a set intersection. In other words, the two parties, each holding a private data set, can jointly compute the intersection or the intersection cardinality of the two sets without leaking any additional information of either party.

Apart from that, some researchers study the private discovery of common friends. Arb et al. [16] utilize the commutative encryption to study whether two users have the same contacts on their mobile phones or not, based on phone numbers. Cristofaro et al. [17] adopt index-based message encoding [28] to achieve privacy-preserving common contact discovery. Nagy et al. [18] employ bearer capabilities [29] to construct a Common Friends discovery scheme that allows two parties to privately learn whether they have common friends or not. Besides, a novel PSI scheme based on Bloom filter is constructed with lower communication overhead in their paper. To achieve privacy-preserving friendship establishment, Zhu et al. [30] propose their scheme based on bloom filter and Pairing-based blind signature. Li et al. [19] design a Bloom filter-based private matching scheme according to friends-of-friends’ recommendation. In addition, an improved similarity function is proposed in their paper.

To achieve fine-grained private matching in proximity-based mobile social networks, Zhang et al. [31] assign different weights to each interest according to the users’ preference and calculates the matching similarity value. Niu et al. [32] point out that Zhang’s scheme did not consider the priority of users’ attributes and present an improved protocol. Zhu et al. [33] propose a two-party and multi-party matching protocol to achieve a fine-grained and preferably private matching without TTP by using commutative encryption. Li et al. [34] propose a practical group matching scheme without TTP for privacy-aware users in mobile social networks. To realize this, they propose a fuzzy matrix algorithm based on PPSPC and construct a public set stored all the group members’s relevant profiles.

III. SYSTEM MODEL AND PRELIMINARIES

A. System Model

We give the system model in Fig. 1. The mobile patients who have the same symptoms and/or live in the neighborhood can constitute a social group according to [2]. Thus, we first divide the our system into different groups. Each of the group has an symptom authority (SA). SA is a powerful and trusted entity located at healthcare center. It is mainly responsible for managing the group. BSNs periodically collects PHI and reports it to SA through smartphone. Each SA and all the network nodes (e.g. APs) between this SA and the patients reports it to SA through smartphone. Each SA and all the network nodes (e.g. APs) between this SA and the patients managed by it can construct a SDN network. The controller of this SDN network can reside in SA or other designated server located in the same healthcare center. It is responsible for controlling data flows between SA and smartphones by downloading flow entries to the intermediate nodes to improve the flexibility and performance of data transferring. Therefore, patients belonging to the same SA can achieve securely...
medical experience or the PHI sharing according to the secure information.

As shown in Fig. 2, our APP enables users to make a social interaction. In our system model, we assume each user has a symptom profile \( P = \{I_{x_i}, v_{x_i}\} \), where \( I_{x_i} \) represents the \( i^{th} \) symptom of user \( x \) and \( v_{x_i} \in \{1, 2, \ldots, 9\} \) is the corresponding priority level of \( I_{x_i} \). For the coarse-grained symptom matching scenario, \( v_{x_i} = 0 \). Here, we assume the number of \( i^{th} \) symptom is more than one (Users can add several dummy symptoms when he/she only have one symptom.). Notice that both the symptom \( I_{x_i} \) and the corresponding levels \( v_{x_i} \) are defined by users. Moreover, they are stored and maintained locally by each user because of sensibility and privacy. Besides, when an initiator begins the social interaction, we always hope that the initiator and the responder have more common symptoms and closer priority levels.

B. Adversary Models and Design Objectives

Each user in our scheme is assumed to be honest-but-curious in this paper. It means that users will honestly follow the protocol but try their best to learn more information than permitted.

During the symptom matching, our proposed schemes should ensure the following privacy-preserving properties: (1) **Initiator’s Symptom Privacy**: The initiator does not reveal any information of his/her symptom to the responder; (2) **Responder’s Symptom Privacy**: The responder does not reveal any information of his/her symptom to the initiator. Thus, participants can only learn the number of common symptoms after the symptom matching. Moreover, the initiator can obtain the similarity degree about the common symptoms while the responder cannot.

C. Preliminary

1) **Blind Signature**: The blind signature scheme is a special type of signature scheme. The basic idea of most existing blind signature scheme is that the signature requestor randomly chooses some blind factors and embeds them into the message to be signed. Since the blind factors are kept secret, the signer cannot obtain the real message. After getting the blinded signature from the signer, the signature requestor can remove the blind factors and get the valid signature.

Fig. 3 presents a RSA-OPRF example. Here \( e \) is a fixed and public RSA exponent. Input \( e \), key generation algorithm can calculate \( (N, d) \) such that \( ed \equiv 1 \mod \phi(N) \), where modulus \( N \) is the product of two distinct primes of roughly equal length and \( N < e \). Finally, the algorithm returns the public key \( (N, e) \) and the corresponding private key \( (N, d) \). Then, for the message \( M \), the user can choose a random number \( r \in \mathbb{Z}_N \) and calculate the hash value \( h = H(M) \) where \( H : \{0, 1\}^* \rightarrow \mathbb{Z}_N \). Then, the user calculates \( x = (hr^e \mod N) \) and sends \( x \) to the server. The server signs it by calculates \( y = x^d \mod N \) and returns \( y \) back. Finally, the user can remove the blinding by calculating \( z = (y^{-1} \mod N) \) and then ensure \( (z \mod N) \) is indeed equal to \( h \).

2) **Bloom Filter**: The Bloom filter \( BF \) is a simple space-efficient randomized data structure for representing a set \( S \) in order to support membership queries [35]. From the point of data storage, it is a bit array with size \( m \). Initially, all bits in the array of \( BF \) is set to be 0. Bloom filters have two operations: \( BF.add(x) \) and \( BF.query(x) \), where \( x \) is an element. The \( BF.add(\cdot) \) operation consists of hashing an element with several hash functions \( h_1, \ldots, h_k \), which uniformly maps the element to a number such as \( h_i(x) = y_i \in [1 : m] (1 \leq i \leq k) \), and sets the \( y_i \)th bit in the array to 1 where \( BF(y_1) = 1, \ldots, BF(y_k) = 1 \). The \( BF.query(\cdot) \) operation repeats the same hashing procedure, and then checks if the appropriate bits are set as 1. If \( x \in BF \), then \( BF(y_1) = 1, \ldots, BF(y_k) = 1 \) is satisfied. Fig. 4 presents an example about Bloom filter where \( k = 3 \) and \( m = 12 \). As shown in this figure, \( x_1 \) and \( x_2 \) are added to Bloom filter by hash operations. During the query process, only \( x_1 \) can be ensured in the Bloom filter since all bits of its hash operations in the array is 1.

Notice that a false positive probability \( p \) exists when determining whether an element \( x \) belongs to a set or not because of the hash collision property. We can calculate \( p \) as follows [35]:

\[
p = (1 - (1 - \frac{1}{m})^{kn})^{k} \approx (1 - e^{-\frac{kn}{m}})^{k},
\]

where \( n \) is the number of elements in set \( S \). Obviously, when \( k = (ln2)^\frac{m}{n} \), the false positive probability \( p \) is minimal, i.e., \( 0.6185 \frac{m}{n} \).

IV. Our Proposed Scheme

In this section, we will present our blind signature-based privacy-preserving symptom matching schemes in detail. We divided it into two categories: coarse-grained symptom matching and fine-grained symptom matching.
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2799209, IEEE Internet of Things Journal

Finally, $N < e$ of roughly equal length and $N < e$.

The correctness of the blind signature process is given as formula (4).

**Fig. 5**: The coarse-grained symptom matching scheme.

**Parameters.** We assume $e$ is the public RSA exponent and $H(\cdot)$ is the hash function where $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N$.

**KenGen**($e$) For input $e$, each user calculates $(N, d)$ such that $ed \equiv 1 \mod \phi(N)$, where modulus $N$ is the product of two distinct primes of roughly equal length and $N < e$. Finally, the algorithm returns the public key $(N, e)$ and the corresponding private key $(N, d)$.

**BFGen**($N, d$) For Alice, who has the private key $(N, d)$ and $P_A = \{I_A_i\}$, launches a private matching as follows:

- Alice chooses a random number $R \in \mathbb{Z}_N$;
- Alice calculates $h_{A_i} = H(I_A_i || R)$ for each $I_A_i$;
- Alice computes $S_{A_i} = h_{A_i}^e \mod N$;
- Alice generates the Bloom filter $BF_A$ by $BF.add(S_{A_i})$.

Finally, Alice sends $\{BF_A, N, e, R\}$ to Bob. Notice that $BF_A$ can be computed offline.

**BSGen**($N, e, R$)

- **BSGen.BA**
  - Once Bob receives the matching request, Bob executes the blinding algorithm as follows:
    - Bob chooses a random number $r \in \mathbb{Z}_N$;
    - Bob calculates $h_{B_i} = H(I_B_i || R)$;
    - Bob computes $x_i = h_{B_i} \cdot r^e \mod N$;
    - Bob sends $\{x_i\}$ to Alice.

- **BSGen.SA**
  - After receiving $\{x_i\}$ from Bob, Alice executes the signing algorithm.
    - Alice calculates $y_i = x_i^d \mod N$;
    - Alice sends $\{y_i\}$ to Bob in a random permutation.

- **BSGen.UA**
  - After receiving the blind signatures $\{y_i\}$, Bob carries out unblinding algorithm.
    - Bob computes $z_i = y_i \cdot r^{-x} \mod N$ to get his symptom signatures;
    - Bob executes $BF.query(z_i)$ operation.

The correctness of the blind signature process is given as formula (2).

Finally, Alice sends $\{S_{A_i}, H(v_{A_i} || R_2), N, e\}$ to Bob. Notice that $S_{A_i}$ and $H(v_{A_i} || R_2)$ can be computed offline.

**FBSGen**($N, e, R_1, R_2$)

- **FBSGen.BA**
  - Once Bob receives the matching request, Bob executes the blinding algorithm as follows:
    - Bob chooses a random number $r \in \mathbb{Z}_N$;
    - Bob calculates $h_{B_i} = H(I_B_i)$;
    - Bob computes $x_i = h_{B_i} \cdot r^e \mod N$;
    - Bob sends $\{x_i, v_{B_i}\}$ to Alice.

- **FBSGen.SA**
  - After receiving $\{x_i, v_{B_i}\}$ from Bob, Alice executes the signing algorithm.
    - Alice calculates $y_i = (x_i \cdot R_1)^d \mod N$;
    - Alice computes $H(v_{B_i} || R_2)$ for each $v_{B_i}$;
    - Alice sends $\{y_i, H(v_{B_i} || R_2)\}$ to Bob in a random permutation.

- **FBSGen.UA**
  - After receiving the blind signatures $\{y_i, H(v_{B_i} || R_2)\}$, Bob carries out unblinding algorithm.
    - Bob computes $z_i = y_i \cdot r^{-x} \mod N$ to get his symptom signatures;
    - Bob creates two lists: $L_A = \{S_{A_i}, H(v_{A_i} || R_2)\}$, $L_B = \{z_i, H(v_{B_i} || R_2)\}$;
    - Bob sends $\{H(v_{A_i} || R_2), H(v_{B_i} || R_2)\}$ to Alice for all $S_{A_i} = z_j$ to Alice.

The correctness of the blind signature process is given as formula (4).

**Fig. 6**: The fine-grained symptom matching scheme.
A. Coarse-Grained Symptom Matching

In this situation, each user only has the corresponding symptoms without the degree for each symptom. An overview of our proposed scheme is given as follows.

1) Overview: In SDN-based MHSNs, the region is divided into several domains. Each domain has a public parameter $\text{params} = \{e, H(\cdot)\}$ and each user holds the symptom matching symptom $P_x = \{I_{x_i}, v_{x_i}, |I_{x_i}| \in I\}$. Users can achieve privacy-preserving symptom matching by the system parameter and blind signature scheme. We assume the initiator $Alice$ launches the coarse-grained symptom matching process, and $Bob$ is the responder. $Alice$ first signs symptoms in her domain and adds the signed symptom into Bloom filter by $BF.\text{add}(\cdot)$, then she sends the Bloom filter to $Bob$. To get the symptom signatures of himself, $Bob$ firstly sends his blinded symptoms to $Alice$. Then $Alice$ signs the blinded symptoms and returns to $Bob$ randomly. After getting blinded symptoms, $Bob$ carries out unblinding algorithm to get the symptom signatures. Then $Bob$ execute the Bloom filter query operation $BF.\text{query}(\cdot)$ by the symptom signatures of himself. Notice that, if symptoms are the same, signatures are the same, too. After that, $Alice$ can get the number of common symptoms and decide whether to make social interaction with $Bob$ or not.

2) Our scheme: We introduce our coarse-grained symptom matching scheme as shown in Fig. 5. Here $\text{KenGen}(e)$ is the key generation algorithm which is used to generate the public key $(N, e)$ and the corresponding private key $(N, d)$. $\text{BFGen}(N, d)$ is the Bloom filter constructing algorithm by signatures. $\text{BSGen}(N, e, R)$ is the blind signature algorithm which is used to generate the signature. It contains three algorithms which are $\text{BSGen.BA}$, $\text{BSGen.SA}$, and $\text{BSGen.UA}$.

\[
\begin{align*}
    z_i &= y_i \cdot r^{-1} \mod N = (x_{ii}^d \mod N) \cdot r^{-1} \mod N \\
        &= ((h_{Bi} \cdot r^d) \mod N) \cdot r^{-1} \mod N \\
        &= h_{Bi}^d \cdot r \cdot r^{-1} \mod N = h_{Bi}^d \mod N
\end{align*}
\]

After getting the number of common symptom from $Bob$, $Alice$ evaluates the similarity degree between them by the similarity metric based on Jaccard metric as formula (3).

\[
D(A, B) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|}.
\]

B. Fine-Grained Symptom Matching

In this situation, each symptom has a corresponding degree as $P_x = \{I_{x_i}, v_{x_i}, |I_{x_i}| \in I\}$. An overview of our proposed scheme is given as follows.

1) Overview: In SDN-based MHSNs, the region is divided into several domains. Each domain contains a parameter $\text{params} = \{e, H(\cdot)\}$ and each user holds the the symptom matching symptom $P_x = \{I_{x_i}, v_{x_i}, |I_{x_i}| \in I\}$. We assume the initiator $Alice$ launches the symptom matching process, and $Bob$ is the responder. As described in $\text{FSGen}(N, d)$ algorithm, $Alice$ signs her symptoms and hashes the corresponding levels. Then, she sends theses messages to $Bob$. $Bob$ uses the $\text{FBSGen}(N, e, R_1, R_2)$ algorithm to get the symptom signature of himself and protect the privacy. Then, $Bob$ can get hash values of level pairs about common symptoms. After getting these hash values of level pairs from $Bob$, $Alice$ can calculate the weighted Euclidean distance. After getting the result, $Alice$ can decide whether to make social interaction with $Bob$ or not.

2) our scheme: We introduce our fine-grained symptom matching scheme as shown in Fig. 6. Here $\text{FBSGen}(N, d)$ is the signature generation algorithm. $\text{FBSGen}(N, e, R_1, R_2)$ is the blind signature algorithm which is used to generate the signature. It contains three algorithms which are $\text{BSGen.BA}$, $\text{BSGen.SA}$, and $\text{BSGen.UA}$.

After receiving $\{H(v_{A_i} || R_2), H(v_{B_i} || R_2)\}$, $Alice$ executes a matching operation by $R_2$ to get $\{(v_{A_i}, v_{B_i})\}$. Then, she calculates $d_{ij} = (v_{A_i} - v_{B_i})^2$ and $w_{ij} = \frac{2}{\sqrt{\sum d_{ij} \cdot w_{ij}}}$, and gets $D(A, B) = \sqrt{\sum d_{ij} \cdot w_{ij}}$, where $m$ is the number of common symptoms. Finally, $Alice$ can decide whether to make social interaction with $Bob$ or not according to $D(A, B)$ and the pre-defined threshold.

\[
\begin{align*}
    z_i &= y_i \cdot r^{-1} \mod N = (x_{ii}^d \mod N) \cdot r^{-1} \mod N \\
        &= (((h_{Bi} \cdot r^d) \mod N) \cdot R_1^d) \cdot r^{-1} \mod N \\
        &= h_{Bi}^d \cdot r \cdot r^{-1} \cdot R_1 \mod N = h_{Bi}^d \cdot R_1^d \mod N \\
        &= (H(I_{Bi}) \cdot R_1) \mod N
\end{align*}
\]

C. Extend to Group Match

In this section, we extend our schemes to the group match process. Assume there is a formed group $G_i$ with $G_N$ member by using our schemes. When Alice wants to join the group, she executes the matching operation with each member of the group as follows:

1) Coarse-Grained Symptom Matching: For $Alice$, who has the private key $(N, d)$ and $P_A = \{I_{A_i}\}$, she executes $\text{BFGen}(N, d)$ algorithm to get $BF_A$. Then, she broadcasts the request message

\[
Alice \rightarrow * : BF_A, N, e, R.
\]

When the group member $M_i$ in the $G_i$ receives the matching request, $M_i$ executes the the blinding algorithm $\text{BSGen.BA}$ to calculate $\{x_{M_i}\}$. Then $M_i$ returns the following message:

\[
M_i \rightarrow Alice : x_{M_i}.
\]

After receiving $\{x_{M_i}\}$ from $M_i$, $Alice$ executes the signing algorithm $\text{BSGen.SA}$ to calculate $y_{M_i}$, and sends $\{y_{M_i}\}$ to $M_i$ in a random permutation.

\[
Alice \rightarrow M_i : y_{M_i}.
\]

After receiving the blind signatures $y_{M_i}$, $M_i$ carries out unblinding algorithm $\text{BSGen.UA}$ to get $z_{M_i}$. Then, $M_i$ executes the $BF.\text{query}(z_{M_i})$ operation to get the number of common symptom.
After getting the number of common symptoms from $M_i$, Alice calculates $D(A, M_i)$ with each group member as formula (3). Finally, Alice calculates an average match degree and compares it with the pre-defined threshold $\tau_g$ to decide whether to join the group or not.

2) Fine-Grained Symptom Matching: For Alice, who has the private key $(N, d)$ and $P_A = \{I_A, v_A\}$, she executes FBSGen$(N, d)$ algorithm to get $S_{A_i}$ and $H(v_{A_i} || R_2)$. Then, she broadcasts the request message

$$Alice \rightarrow * : \{S_{A_i}, H(v_{A_i} || R_2), N, e\}.$$  \hspace{1cm} (8)

When the group member $M_i$ in the $G_i$ receives the matching request, $M_i$ executes the blinding algorithm FBSGen.UA to calculate $\{x_{M_i}\}$. Then $M_i$ returns the following message:

$$M_i \rightarrow Alice : x_{M_i}, v_{M_i}.$$  \hspace{1cm} (9)

Once receiving $\{x_{M_i}\}$ from $M_i$, Alice executes the signing algorithm FBSGen.SA to calculate $y_{M_i}$, and sends $\{y_{M_i}, H(v_{M_i} || R_2)\}$ to $M_i$ in a random permutation.

$$Alice \rightarrow M_i : y_{M_i}, H(v_{M_i} || R_2).$$  \hspace{1cm} (10)

After receiving the blind signatures, $M_i$ carries out unblinding algorithm FBSGen.BA to get $z_{M_i}$. Then, $M_i$ sends $\{H(v_{A_i} || R_2), H(v_{M_i} || R_2)\}$ to Alice for all $S_{A_i} = z_{M_i}$ to Alice. Then Alice can calculate $D(A, M_i)$ with each group member. Finally, Alice calculates an average match degree and compares it with the pre-defined threshold $\tau_g$ to decide whether to join the group or not.

V. SECURITY ANALYSIS

In this section, we present the privacy-preserving analysis about our symptom matching schemes.

A. Coarse-Grained Symptom Matching

1) The initiator: According to our scheme, Alice only receives the message $\{x_i\}$ and the number of common symptoms from Bob. Before Bob sends $x_i$ to Alice, the BSGen.BA is used to hide Bob’s symptoms $I_B$. According to the nature of blind signature, Alice cannot learn anything about his symptom from $x_i$. Although Alice can get the number of common symptoms, she cannot distinguish them, or infer the accurate corresponding symptoms of Bob. It is obvious that no matter what Alice inputs, she only can learn the number of common symptoms and the weighted Euclidean distance between their symptoms. In this way, Bob’s privacy can be protected.

2) The responder: When the protocol runs, Bob can only get messages $\{BF_A, N, e, R\}$ and $\{y_i\}$ from Alice. BF$_A$ is the Bloom filter consisted by the signatures of Alice’s symptoms. According to the one-way property of hash functions in Bloom filter, Bob can learn nothing about Alice. After getting $y_i$ from Alice, Bob executes BSGen.UA to get the signatures of Bob’s symptoms. Notice that, to prevent Bob learns the detailed common symptoms of Alice, Alice returns $y_i$ in a random permutation. Thus, even knowing signatures of all his symptoms, Bob cannot learn the detailed names of common symptoms. As described above, our proposed scheme can protect Alice’s privacy. Thus, our proposed symptom matching scheme can protect the participants’ privacy.

Besides, for each matching process, the initiator chooses a random number $R \in \mathbb{Z}_N$ and calculates the signature as $S_{A_i} = H(I_{A_i} || R)^d \mod N$. Because of the addition information, the signatures for the same messages are different. Therefore, we can resist the side channel attack such as the trajectory tracking.

B. Fine-Grained Symptom Matching

1) The initiator: According to our scheme, Alice can only receive messages $\{x_i, v_{B_i}\}$ and $\{H(v_{A_i} || R_2), H(v_{B_i} || R_2)\}$ from Bob. Before Bob sends $\{x_i, v_{B_i}\}$ to Alice, the BSGen.BA has been adopted to hide Bob’s symptom $I_B$. According to the nature of blind signature, Alice cannot learn anything about his symptom from $\{x_i, v_{B_i}\}$. Although Alice can obtain the symptom level pair $\{(v_{A_i}, v_{B_i})\}$ according to $\{H(v_{A_i} || R_2), H(v_{B_i} || R_2)\}$, she cannot distinguish them, or infer the accurate corresponding symptoms of Bob. It is obvious that no matter what Alice inputs, she only can learn the number of common symptoms and the weighted Euclidean distance between their symptoms. In this way, Bob’s privacy can be protected.

2) The responder: When the protocol runs, Bob can only get messages $\{S_{A_i}, H(v_{A_i} || R_2), N, e\}$ and $\{y_i, H(v_{B_i} || R_2)\}$ from Alice. For $\{S_{A_i}, H(v_{A_i} || R_2), N, e\}$, $S_{A_i}$ is the signature of Alice’s symptoms. According to the RSA-based signature algorithm, Bob cannot get the plaintext of Alice’s symptoms without her private key. $H(v_{A_i} || R_2)$ is calculated by a hash function. The construction of the hash function is easy to compute but hard to invert. For $\{y_i, H(v_{B_i} || R_2)\}$, $y_i$ is the blind signature of Bob, which means he can learn nothing about Alice’s privacy. For the hash value $H(v_{B_i} || R_2)$, it is impossible to reconstruct $v_{B_i}$ according to the one-way property of hash algorithm. Given the above analysis, we can protect Alice’s privacy properly as Bob cannot infer any information, and our private matching scheme can ensure the privacy of the participants.

Moreover, for each matching process, the initiator chooses a random number $R_1 \in \mathbb{Z}_N$ and computes the signature as $S_{A_i} = (H(I_{A_i}) \cdot R_1)^d \mod N$. Because of the addition information, the signatures for the same messages are different. Thus, we can resist the side channel attack such as the trajectory tracking.

VI. PERFORMANCE ANALYSIS

We analyze the complexity of our schemes and some related work in this section. We adopt the offline/online computation overhead as well as the communication overhead to measure the complexity of our schemes. The computation overhead is measured by the number of the multiplication and exponentiation operations, which are always resource-consuming on mobile devices. The communication overhead is measured by the number of transmitted and received bits. Here, hash represents the hash function such as SHA-256. mul$_1$ and exp$_1$ denote one 160-bit multiplication and 160-bit exponentiation on $\mathbb{G}_1$, respectively. To provide the same security level, mul$_2$ and
exp_2 denote one 1024-bit modular multiplication and 1024-bit modular exponentiation on \( \mathbb{Z}_N \), respectively. Moreover, the public symptom set of symptoms is \( n \), each user has \( n_p \) symptoms and \( n_c \) (\( n_c \leq n_p \)) common symptoms. In Table I, we compare our coarse-grained symptom matching scheme with VENETA [16], PSI-CA [27] in [18] and Zhu et al. [30], where \( L_{BFA} \) is the length of the Bloom filter. Notice that we neglect the overhead of the Diffie-Hellman key generation operation and the message encryption operation during the comparing process for [18]. Moreover, we compare our fine-grained symptom matching scheme with Fine-grained [31], EWPMP Level-II [20], two-party scheme of [33] and Jiang et al [1] in TABLE II.

As shown in TABLE I and TABLE II, our schemes may have no advantage in complexity analysis. However, the Pairing-based operations are time-consuming on Android Studio. We execute the benchmark test based on JPBC library 2.0.0 on Xiaomi Mi 5 smartphone with 1.8GHz Snapdragon 820 CPU, 3GB RAM, Android 7.0 OS, API 24. The results are 0.24 ms for \( mul_1 \) and 70.46 ms for \( exp_1 \), respectively. While for RSA-based operations, they need 0.037 ms for \( mul_2 \) and 0.039 ms for \( exp_2 \), respectively. The detailed overhead will be presented in the following implementation section.

VII. IMPLEMENTATION

To demonstrate the efficiency of our schemes, we comparing our scheme with Zhu et al.'s [30] and Jiang et al.'s [1] schemes, both of which adopt the Pairing-based blind signature to achieve privacy-preserving matching. We design APPs with the Java SE Development Kit and Android Studio. For our scheme, we use 1024-bit RSA to construct the RSA-based blind signature and take SHA-256 as hash function. Bloom filter operation is executed by code available from https://github.com/MagnusS/Java-BloomFilter with SHA-256 as the underlying hash function. During the design process, the fixed false positive probability is \( p = 10^{-4} \). For Zhu et al.'s [30] and Jiang et al.'s [1] schemes, we used JPBC library 2.0.0 to design APPs. We use Type A pairing to construct the Pairing-based blind signature and take SHA-256 as hash function. Also we use the same parameter Bloom Filter.

To measure the execution time and bandwidth overhead, we implement our APP (over 30 trials) on Xiaomi Mi 5 smartphone with 1.8GHz Snapdragon 820 CPU, 3GB RAM, Android 7.0 OS, API 24, connected over WIFI. During the implementation, we assume that both patients have the same number of symptoms \( n_p \) which vary from 10 to 100. The common symptoms one at 10% of the symptom set size.

A. Computation Overhead and Communication Overhead

In this section, we use the running time and the message size of different schemes to evaluate the computation Overhead and communication overhead as follows:

Fig. 7 presents the time overhead for each algorithm in coarse-grained symptom matching schemes. As shown in Fig. 7(a), when \( n_p=100 \), the execution time is 83 ms for \( BFGen \), 19 ms for \( BSGen.BA \), 65 ms for \( BSGen.SA \), and 48 ms for \( BSGen.UA \). While for Zhu et al.'s scheme, the corresponding time is about 24819 ms, 31909 ms, 6953 ms, and 7018 ms, respectively. We acknowledge that the running time almost increases linearly with the increasing number of symptoms for both schemes, while the rates are different. Comparing with Fig. 7(b), it is obvious that our scheme is much more efficient than [30]. The main reason is that Pairing-based operations are time-consuming. This also makes [30] is not practical for mobile devices.

In Fig. 8, we presents the time overhead for each algorithm in fine-grained symptom matching schemes. As shown in Fig. 8(a), when \( n_p=100 \), the execution time is 80 ms for \( FSGen \),
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2799209, IEEE Internet of Things Journal

16 ms for FBSGen.BA, 75 ms for FBSGen.SA, 42 ms for FBSGen.UA, and 5 ms for matching operation. For Jiang et al.’s scheme, the corresponding time is about 25098 ms, 32381 ms, 7165 ms, 7036 ms and 5 ms, respectively. It is obvious that our scheme is much more efficient than [1]. Besides, the running time almost increases linearly with the increasing number of symptoms for both schemes. Furthermore, as can be seen from Fig. 7(a) and Fig. 8(a), our symptom matching schemes almost consume the same computation overhead for coarse-grained and fine-grained situations.

Table III lists the corresponding communication overhead for each algorithm in coarse-grained symptom matching schemes. As described in the table, the total number of bytes transmitted also increases linearly for both schemes. The communication overhead of both schemes is acceptable. Moreover, the communication overhead of two schemes are almost the same. Since during the application design, we serialize the element in $G_1$ into bytes to ensure it can be exchanged through socket operation.

Table IV lists the corresponding communication overhead for each algorithm in fine-grained symptom matching schemes. It is obvious that the number of bytes transmitted for each algorithm in fine-grained symptom matching schemes almost consume the same computation overhead for coarse-grained and fine-grained situations.
main reason is that we serialize the element in $G_1$ into bytes so that it can be exchanged through socket operation during the application design process.

**B. Power Analysis of Our Schemes**

To present the average power consumption of our APP, we use a power analysis tool for Android devices called WeTest-Assistant. We run our APP on Samsung Galaxy Note IV (N9100). During the implementation, we record CPU load and overall energy consumption which includes computation overhead and communication overhead. As shown in Fig. 9, we execute our coarse-grained symptom matching scheme 15 times (x-axis shows elapsed time and the peaks correspond to the 15 executions) with 50 symptoms and 100 symptoms matching, respectively. According to our measurements, the average CPU load is 16.4% and 17.3% for 50 symptoms and 100 symptoms, respectively. The corresponding average power consumption is about 2205.2 mw and 2502.1 mw. Due to the Wetest platform limitations, the sample rate of the CPU load is once every two seconds, and the power consumption is 3-4 times per second. Fig. 10 present the fine-grained symptom matching scheme results. The average CPU load is 15.7% and 17.6% for 50 symptoms and 100 symptoms, respectively. The corresponding average power consumption is about 2126.3 mw and 2457.7mw.

**VIII. CONCLUSIONS**

In this paper, we propose blind signature-based privacy-preserving symptom matching schemes without any TTP in SDN-based MHSNs. Specifically, we employ the RSA-based blind signature to achieve the coarse/fine-grained privacy-preserving symptom matching while using a random number to resist the side channel attack. Moreover, we adopt Bloom filter to reduce the communication overhead for the coarse-grained privacy-preserving symptom matching. The security analysis shows that our scheme can achieve private matching. The performance analysis indicates the practicality of our proposed schemes. The evaluation result shows that our scheme can achieve privacy-preserving matching for 100 symptoms in few seconds, while the corresponding communication overhead is about several dozens of KBytes. Finally, we demonstrate the efficiency of our solutions by a comprehensive evaluation according to developed APPs. In the future, we will try to improve the fine-grained symptom matching scheme without revealing responder’s priority level.

**ACKNOWLEDGEMENT**

This work was supported by the National Natural Science Foundation of China (No. 61702231, No. 61472001, and No. u1405255), the Natural Science Foundation of Jiangsu Province (BK20170556), Jiangsu Provincial Research Scheme of Natural Science for Higher Education institutions (No. 17KJB520005) and Key Laboratory for New Technology Application of Road Conveyance of Jiangsu Province (No.BM20082061707)

**REFERENCES**


Shunrong Jiang received the B.E. degree in information engineering from Chongqing University, Chongqing, China, in 2008. He received the Ph.D. degree in communication and information systems from the School of Telecommunications Engineering, Xidian University, Xi’an, China, 2016. He joined the Department of Internet of Things, Jiangsu University as an assistant professor in August 2016. His research interests include security and privacy for wireless networks, cloud computing, etc.

Mengjie Duan Menjie Duan received his B.S. degree from School of Internet of Things, Wenzheng College of Soochow University (2016). He is now a M.S. degree student at School of Computer Science and Communication Engineering, Jiangsu University. His research interests include cryptography, network and data security, and cloud computing security.

Liangmin Wang received the B.S. degree in computational mathematics from Jilin University, Changchun, China, in 1999 and the Ph.D. degree in cryptography from Xidian University, Xian, China, in 2007. He is currently a Full Professor with the School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China. His current research interests include security protocols and Internet of Things.