A Privacy-Preserving Networked Hospitality Service with the Bitcoin Blockchain

Hengyu Zhou[1,2], Yukun Niu[1], Jianqing Liu[3],
Chi Zhang[1], Lingbo Wei[1,2], and Yuguang Fang[3]

[1] School of Information Science and Technology
University of Science and Technology of China, Hefei 230027, P. R. China
[2] State Key Laboratory of Information Security, Institute of Information Engineering
Chinese Academy of Sciences, Beijing 100093, P. R. China
[3] Department of Electrical and Computer Engineering
University of Florida, Gainesville, Florida 32611, USA

Abstract. In recent years, we have witnessed a rise in the popularity of networked hospitality services (NHSs), an online marketplace for short-term peer-to-peer accommodations. Such systems, however, raise significant privacy concerns, because service providers such as Airbnb and 9flats can easily collect the precise and personal information of millions of participating hosts and guests through their centralized online platforms. In this paper, we propose PrivateNH, a privacy-enhancing and practical solution that offers anonymity and accountability for NHS users without relying on any trusted third party. PrivateNH leverages the recent progress of Bitcoin techniques such as Colored Coins and CoinShuffle to generate and maintain anonymous credentials for NHS participants. The credential holders (NHS hosts or guests) can then lease or rent short-term lodging and interact with the service provider in an anonymous and accountable manner. An anonymous and secure reputation system is also introduced to establish the trust between unfamiliar hosts and guests in a peer-to-peer fashion. The proposed scheme is compatible with the current Bitcoin blockchain system, and its effectiveness and feasibility in NHS scenario are also demonstrated by security analysis and performance evaluation.

Keywords: Networked hospitality services · Bitcoin blockchain · Anonymity and accountability.

1 INTRODUCTION

Over the last few years, the popularity of networked hospitality services (NHSs), such as Airbnb and 9flats, has significantly increased, serving millions of users in hundreds of cities [1]. These services provide an efficient online marketplace where users can register themselves as hosts (to lease short-term lodging) and/or guests (to rent lodging); the service provider (SP) matches guest requests with available accommodations. In general, NHS can provide more diversified and personalized choices in accommodations at lower costs or with lower transactional overhead, and shows great advantages over traditional hotel industry. Moreover, the accountability provided by NHSs (e.g.,

identity verification mechanism and reputation system adopted by Airbnb) is a key feature that contributes to NHSs widespread acceptance, as it makes hosts and guests feel safer.

Despite the popularity, NHSs come with significant privacy concerns. To offer such services, SPs in NHSs collect the details of each lodging, together with real identities of the host and the guest. Note that other forms of accommodations, such as traditional hotels, also leak private information. However, with the help of centralized online platforms, data collection in NHSs is more efficient, aggressive and large-scale [2]. As a result, the SP, or any entity with access to this data, can infer privacy-sensitive information about hosts or guests, such as where they live, work, and socialize.

In this paper, we analyze the privacy threats in the current form of NHSs and propose PrivateNH, a practical solution that enhances privacy for the guests w.r.t. the SP and privacy for the hosts w.r.t. malicious outsiders, while preserving the convenience and functionality offered by the current system. PrivateNH relies on the recent progress of Bitcoin techniques such as Colored Coins [7] and CoinShuffle [8] and well-known cryptographic primitives like blind signatures [5] and private information retrieval [6]. We utilize the unmodified Bitcoin blockchain as the powerful platform to create and manage anonymous credentials for NHS participants without relying on any trusted third party. The credential holders (NHS hosts or guests) can then lease or rent short-term lodging and interact with the SP in an anonymous and accountable manner. An anonymous and secure reputation system is also introduced to establish the trust between unfamiliar hosts and guests in a peer-to-peer fashion.

In summary, our main contributions are:

- We present the first general privacy analysis of NHSs. By analyzing currently deployed NHSs, we formalize the security and privacy objectives of the next-generation NHSs.
- We propose PrivateNH, a practical system that offers enhanced privacy for hosts and guests, without affecting the convenience of these services. To facilitate adaption, PrivateNH relies exclusively on the unmodified Bitcoin blockchain system and some well-established cryptographic primitives.
- We analyze and evaluate PrivateNH, showing its effectiveness and feasibility in practical NHS scenario.

2 Preliminaries

2.1 Background on the Bitcoin Blockchain

Bitcoin is a peer-to-peer digital cash system that allows miners to mint coins called bitcoins and exchange them without authorized parties. Bitcoin uses a novel permissionless consensus protocol known as proof-of-work [4] to make all nodes agree on a log of transactions and to prevent attacks such as double-spending. This log is the Bitcoin blockchain and is managed by all nodes in the network [4, 9].

The Bitcoin blockchain is an append-only public ledger which tracks all transactions in the system. A special set of participants, called miners, runs the proof-of-work protocol to extend the blockchain by appending newly generated block to the existing

blockchain. A block consists of a block header and a set of transactions. The block header in a block contains a hash pointer to the previous block. The transactions in a block are hashed in a Merkle tree [4, 9], and the tree's root hash is stored in the block header. Bitcoin Simplified Payment Verification (SPV) [4] is a method for verifying if particular transactions are included in a block without downloading the entire block. This method is used by some lightweight Bitcoin clients. The *blockchain* mentioned in this paper all refers to the Bitcoin blockchain.

A transaction consists of inputs and outputs. An output contains two fields: a value field which indicates the number of transferred bitcoins and a locking script that specifies what conditions must be fulfilled for those number of bitcoins to be further spent. An input contains two fields: an outpoint that references the previous output and an unlocking script to spend the bitcoins locked in the previous output. All unspent transaction outputs are called UTXO. For a valid transaction, the sum of the spent values in inputs should be greater than or equal to the sum of the values in outputs. The difference between these two sums is the mining fee for miners. The mining fee is optional, and a transaction creator can specify the amount of fee on their will.

Bitcoin allows embedding data in transactions through a particular kind of transaction output called OP_RETURN. One can specify up to 83 bytes of arbitrary data in an OP_RETURN output [3]. PrivateNH uses it to store application-specific data in the blockchain.

Due to the inherently public nature of the blockchain, users' privacy is severely restricted to linkable anonymity. Various mixing protocols have been introduced to mitigate this drawback. Tim, et al. [8] proposed *CoinShuffle*, a fully decentralized Bitcoin mixing protocol that allows users to utilize Bitcoin in a truly anonymous manner. PrivetNH builds a credential-mixing method based on this protocol.

2.2 Cryptographic Primitives

A blind signature as introduced by David Chaum [5] is a form of digital signature in which the signature requester blinds their message before sending it to the signer. The blinded signature can, in turn, be "unblinded", to obtain a valid signature for the original message. PrivateNH uses the key property, which a signer who is asked to verify the signature of an unblinded message cannot relate this message back to the blinded version they signed.

A private information retrieval (PIR) as introduced by Benny Chor, et al. [6] is a protocol that allows a user to retrieve an item from a server in possession of a database without revealing which item is retrieved. PrivateNH takes advantage of the PIR protocol to make the SP cannot link the reputation to a user when they retrieve their reputation.

3 Models, Assumptions, and Design Goals

3.1 System Model

An NHS includes three parties: hosts, guests, and the SP. The SP handles incoming querying requests from guests and matches guests with available accommodations

based primarily on their locations and dates. The SP also offers essential functionalities such as accountability and reputation ratings.

3.2 Adversarial Assumptions

We assume the SP is honest-but-curious that strives to protect its business and maximize its interests. It has incentives to mining sensitive information about its guests, to either improve its quality of service or to monetize harvested data. We assume hosts are honest and will provide accurate accommodation information.

We assume hosts will not collude with the SP after the guest checks in. But some hosts may want to infer guests' identities during the online booking process.

We assume most guests want to protect their privacy. But some guests may collude with the SP to infer other guests' identities. We also assume that guests are rational, and do not misbehave if the cost of misbehaviors is significant.

We assume that the network and upper-layer protocols do not leak users identifiable information to the SP. In practice, users can use anonymous network systems (e.g., Tor) to conceal their IP addresses. We also assume that users can generate secure asymmetric key pairs and maintain the confidentiality of their secret keys.

We assume that cryptographic building blocks used in the underlying blockchain, the blind signature scheme, and the private information retrieval protocol are secure. We also assume the Bitcoin network is secure and robust.

The outsiders are active adversaries who try to collect hosts' and guests' private information and infer their identities.

3.3 Design Goals

This section describes the design goals of PrivateNH. That is, if PrivateNH satisfies these goals, it is robust against the adversarial assumptions described in Section 3.2.

1) Authentication: Hosts and guests should be mutually authenticated to prevent criminals from participating in online booking. Together with the reputation mechanism, both the host and the guest can ensure that they are authenticated and trustful.

2) Guest anonymity: The SP cannot infer guests' identities. During the online booking process, the host cannot infer their guest's identity. Moreover, Even the SP and some guests collude, they still cannot learn any knowledge about the identity of a particular guest.

3) Guest unlinkability: The SP cannot tell whether two accommodations were booked by the same guest. This means the unlinkability of guests have to be preserved throughout the operations provided by the SP, including bookings and reputation ratings.

4) Accountability: The SP can blacklist misbehaving users (e.g., a guest who damages a host's house). Blacklisted users are no longer able to join future online bookings.

5) Anonymous reputation: It is computationally difficult for hosts and guests to misbehave during reputation ratings. It is computationally difficult for hosts and guests to show a tampered reputation without been discovered. It is computationally difficult for the SP to know whether two ratings are reviewed by the host and the guest in the same booking.

6) Efficiency: All the above properties consume low communication and storage overhead.

4 PrivateNH Overview

As shown in Fig. 1, there are three types of participants in the system: the SP, hosts, and guests. We use *users* to indicate both hosts and guests in this paper.

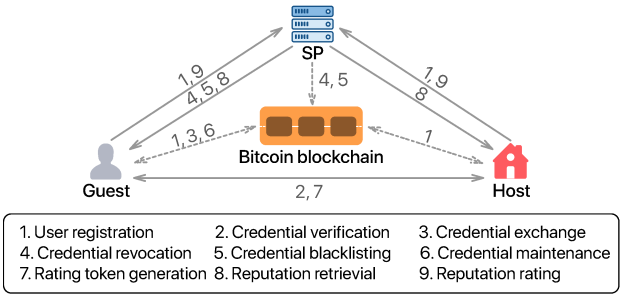


Fig. 1. System overview

A user registers themselves in the system and owns an anonymous credential which can be used to prove the validity of their identity. A credential is the hash of an ECDSA public key which can be deemed as a Bitcoin address. It can be verified using the signature signed by the corresponding secret key.

Hosts and guests can verify each other's credential without the SP. Due to the publicity of the blockchain, a host can verify whether a guest owns a valid credential by checking the blockchain, and vice versa for the guest.

Guests can exchange their credentials with the others using CoinShuffle protocol. Since CoinShuffle is a peer-to-peer protocol, there is no need for the SP to involve.

A guest can fetch their reputation from the SP using the PIR protocol and shows it to the host before the booking. Also, they can generate reputation token using the blind signature scheme to give a review for the host after the booking. These two operations are vice versa for the host.

5 PrivateNH Design and Implementation

5.1 User Registration

An individual can become a user and holds a credential by creating a registration transaction. As soon as the transaction is on the blockchain, the newly created credential becomes valid and everyone can verify it. To generate this transaction,

– First, an individual U sends a signing request $Sign(Addr_U)$ to the SP. In this step, the real identity of U is sending along with the request to the SP to resist the abuse of registrations;
– Next, after making sure the identity is valid and not been registered before, the SP signs $Addr_U$ and sends the result $Sig_{sk_{SP}}\{Addr_U\}$ back to U;
– After verifying the correctness of the signature, U generates a valid transaction and embeds the signature in it;
– Last, U broadcasts the transaction and waits for the Bitcoin network accept it.

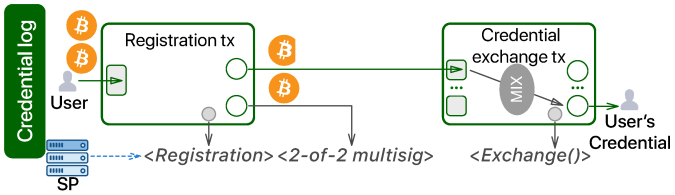


Fig. 2. Registration & credential log

The registration transaction contains three outputs. As shown in Fig. 2, they are represented in circles. We define outputs in order: the first is a genesis credential, the second is a deposit output, and the last is a registration output.

The genesis credential can be later transformed into a new and anonymous credential through *credential exchange protocol*. We discuss this in detail in Section 5.2.

The deposit output is a 2-of-2 multi-signature output where two signatures are required to unlock bitcoins in this output. One signature is signed by the SP, and the other is signed by the creator of the registration transaction. During the registration phase, a user transfers a fixed number of bitcoins to the output. This increases the cost for a user to misbehave. We discuss this in more detail in Section 5.4.

The registration output is an OP_RETURN output which contains a valid signature from the SP. The signature proves the user is authenticated.

5.2 Credential Exchange

Guests follow *credential exchange protocol* to get new credentials and invalidate their old ones. Our protocol works as follows. First, all participants need to verify each other. We discuss this in detail in Section 5.3. Then, each member provides one input address and one output address. The input is equivalent to their old credential, and the output will be their new credential. Next, all members use the decentralized mixing protocol *CoinShuffle* to construct a new transaction which randomized the mapping pattern between inputs and outputs provided by all members. Finally, all members broadcast the new transaction to the Bitcoin network.

When the transaction is added to the blockchain, every participant's new credential becomes valid, and the original one turns invalid automatically. Thanks to the mixing technique, nobody can tell which one is the corresponding old credential with a given new credential.

The values of all outputs must conform to a uniform distribution to make all outputs identical to observers. Suppose m guests engage in the protocol. The input value of the guest G_i is v_{in_i}, and the output value of G_i is v_{out_i}. The mining fee is $f(>=0)$. Then we have the following equations.

$$v_{out_1} = v_{out_2} = ... = v_{out_m} = k, \tag{1}$$

$$k = \frac{\sum_{i=1}^{m} v_{in_i} - f}{m}. \tag{2}$$

5.3 Credential Verification

Whether a guest requests an accommodation booking or participates in a credential exchange transaction, they are required to prove the validity of their credential. First, they use the private key to create a signature to show they own the credential. Then, they construct a proof to demonstrate the credential they hold is valid.

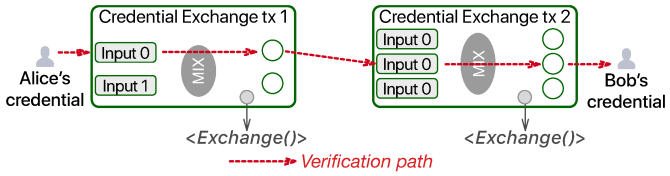


Fig. 3. Credential verification

We introduce *verification path* to construct such a proof. As noted in Section 2, an input is always linked to one specic output of a preceding transaction. Through this, transactions become linked. By defining a mapping between the inputs and the outputs in the same transaction, a path is formed among the transaction graph. In our scheme, we define the mapping with respect to the ordering of inputs and outputs. To clarify, $input_i$ is mapped to the $output_i$ within the same transaction. We specify the resulting path as the *verification path*. As depicted in Fig. 3, Bob's credential is validated using a path containing two credential exchange transactions. And the initial credential may be held by another guest, e.g., Alice.

Given a credential, a verifier can check whether there is a verification path within the blockchain by verifying:

- The output containing the credential is a UTXO;
- The credential is not on the blacklist (we discuss this in Section 5.4);
- There is a verification path between the credential and the checkpoint.

5.4 Credential Revocation

We introduce *credential revocation* to invalidate credentials. As shown in Fig. 4, the SP creates a blacklist transaction to invalidate credentials by embedding these credentials into the transaction's OP_RETURN output. A credential is revoked for two reasons: (1) Active leaving: A user leaves the system (e.g. delete the account), and (2) Passive leaving: The SP punishes a misbehaving user (e.g. blacklist the user).

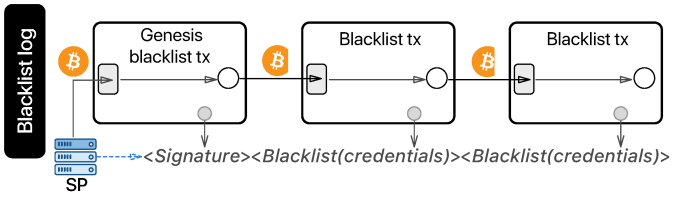


Fig. 4. Blacklist log

If a user leaves the system, they can reclaim the deposit that is locked in the registration transaction. Since the registration transaction is related to the user's identity, the SP can infer who is requesting the refunds. Thus for a passive leaving user (a.k.a. a misbehaving user), they can choose to either reclaim their deposit while revealing their identity to the SP or lost their deposit.

5.5 Credential Maintenance

If the value of a user's credential becomes too small for the user to involve in the next credential exchange transaction, the user can: (1) request the SP to revoke the credential and generate a new one, or (2) charge the credential.

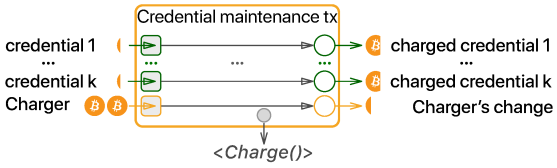


Fig. 5. Credential maintenance transaction

A user creates a credential maintenance transaction to charge their credential. As shown in Fig. 5, some users may create one credential maintenance transaction together to share the expense of the transaction fee. We define the same mapping pattern as in

Section 5.3 in compliance with the verification path rule. The charger can be the user or a charge service provider.

5.6 Reputation Rating

We show the online booking workflow of PrivateNH in Fig. 6. We detail the steps which are the main concerns of the reputation rating.

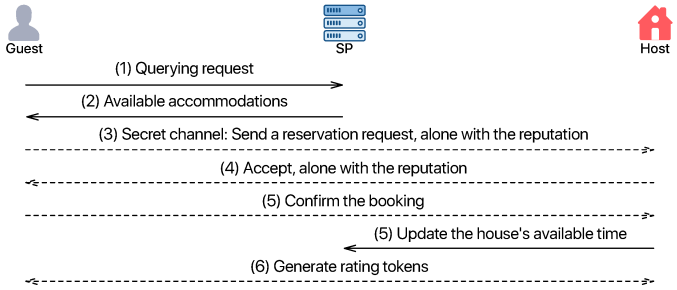


Fig. 6. PrivateNH's online booking workflow

In step 3 and 4, the guest and the host retrieves reputation from the SP using the PIR protocol without revealing which reputation they retrieve. The SP returns the reputation along with a signature $Sig_{SP}\{reputation\}$ which can be used to prove the completeness and correctness of the reputation. The SP can also return the signature with a timestamp, e.g., $Sig_{SP}\{reputation, timestamp\}$, to bring freshness to the reputation.

In step 6, the host H and the guest G use the blind signature scheme to generate rating tokens. The operations are as follows: (1) H and G generate a random value r_h and r_g respectively; (2) H sends the hash $c_h = hash(r_h)$ to G, and G sends the hash $c_g = hash(r_g)$ to H; (3) H requests a blind signature from the SP: $BlindSig_{SP}\{cert_h, sig_h\{c_g\}\}$, and G requests a blind signature from the SP: $BlindSig_{SP}\{cert_g, sig_g\{c_h\}\}$; (4) Both of them unblinds the signature to reveal the rating tokens: $RT_g = Sig_{SP}\{cert_g, sig_g\{c_h\}\}$ and $RT_h = Sig_{SP}\{cert_h, sig_h\{c_g\}\}$; and (5) G sends RT_g to H so that H can review G after the checking out, and vice versa.

The guest sends to the SP the rating token RT_h, the random value r_g used to generate the hash c_g, and a review of the host. When the SP receives this token, it checks the correctness of the signature, the correctness of the certificate $cert_h$, whether c_g has not been used before and whether c_g is the hash of r_g. And vice versa for the host. Note that, to avoid time-correlation attacks by the SP, the rating should not take place right after the checking out. This can be implemented by imposing some random delay before the rating.

6 Analysis and Evaluation

6.1 Authentication, anonymity, unlinkability, and accountability

Authentication During the authentication process, both the host and the guest have credentials, which can be validated by each other as described in Section 5.3. Specifically, the prover shows their credential has a verification path which originates from the credential in a registration transaction. The verification path cannot be tampered and is publicly verifiable. Since the underlying blockchain is a global append-only ledger and is assumed secure and robust, the verifier can believe the authenticity of the prover.

Anonymity In this scheme, anonymity is enabled through credential exchange protocol. We suppose a genesis credential has a backward anonymity of 1. It is usually increased after every exchange since the backward anonymity set after an exchange is the union of all anonymity sets of the participating credentials. Given a credential exchange transaction with the set of participating credentials G and the anonymity sets A_g for all $g \in G$, the anonymity increases Δa_g for a $g \in G$ can be written as:

$$\Delta a_g = |\bigcup_{p \in G} A_p \setminus A_g|. \tag{3}$$

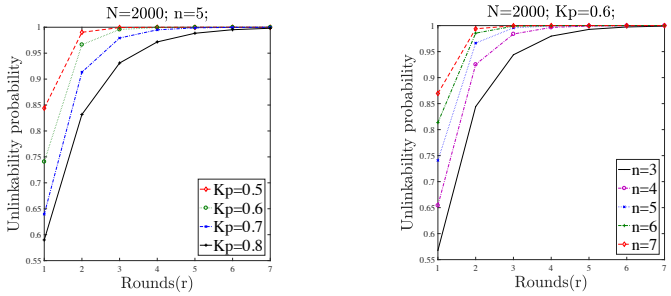
We have: (1) a user change their credential more frequently comes to a bigger anonymity set for themselves, and (2) more user participant in one exchange brings larger anonymity set.

Unlinkability PrivateNH uses credential exchange protocol to make credentials unlinkable. Credential exchange protocol takes advantage of CoinShuffle. We define the *unlinkability probability* that quantifies the probability where the credential exchange protocol has at least two honest participants. It means that honest guests can get an unlinkable credential in such a probability after participating in credential exchange protocols. We assume that colluded guests are randomly selected to participate in the protocol. Let N denotes the number of all guests, m denotes the number of colluded guests, K_p denotes colluded guest rate. Then, $K_p = \frac{m}{N}$. Let n denotes the number of guests in every credential exchange protocol, r denotes the credential exchange rounds for a credential, P_r denotes the unlinkability probability after r exchanges. Then we have:

$$P_r = 1 - \frac{C_m^{n-1} C_{N-m}^1}{C_N^n} \times (\frac{C_m^{n-1} C_1^1}{C_{N-1}^{n-1} C_1^1})^{r-1}. \tag{4}$$

Fig. 7a shows honest guests get high unlinkability through several credential exchanges even though most guests collude. Fig. 7b shows the more guests in every round, the higher unlinkability probability for honest guests.

PrivateNH uses the blind signature schemes to achieve unlinkability for reputation ratings. In a reputation rating, the host and the guest *do not* provide their identifying information to the SP during the reputation rating. That is, the IP addresses and real credentials of the hosts and guests are invisible to the SP. This, together with the fact that there is no identity information included in the token, guarantees unlinkability between the host and the guest.



(a) Colluded guests rate K_p (b) Size n

Fig. 7. Parameter impact on the unlinkability probability

Accountability PrivateNH achieves accountability without any trusted third party. First, the SP can revoke misbehaving credentials while it cannot link the credentials to users' identities. Second, users will refuse blacklisted credential holders participating in the credential exchange protocol since their new credentials may be the blacklisted ones after the protocol. Finally, the deposit raises the expense of misbehaviors. When users withdraw their deposits, their identities are disclosed and misbehaving credential holders will be identified.

6.2 Performance Evaluation

Table 1. Size evaluation of different system defined transactions

	registration	deposit	credential exchange	blacklist update	credential maintenance
size(bytes)	351	333	224	255	224

Communication Overhead A user needs to download some system transactions to verify the others' credentials. Given n users, all users change their credentials after every booking (for example every one week), the maximum length of validation path is l, and users need to charge credentials after l changes. For the worst case, a user needs to download $n * (registration + deposit + l * maintenance)$ data in bytes. The size of different types of transactions is given in Table 1. If the system has 10000 users, and a credential exchanges 20 times before its charging. The maximum communication overhead for a user in 20 weeks is: $10000 * (351 + 333 + 224 * 20) = 51.64Mb = 6.455MB$. This is only $369bytes$ on average per day. And if there are too many users in the system, they do not need to download all the system created transactions. They can either download the transactions on demand or use a trust-but-verifiable server who only pushes data they need.

Storage Overhead A user needs to store some public and private key pairs. Our system uses ECDSA with the secp256k1 curve. A pair of public and private key has $65 + 32 = 97bytes$. A user stores 2 pairs: one for withdrawing the deposit; another is for proving they own their latest credential. Also, a user stores all users' latest credentials, which is $(n-1) * output = (n-1) * 56bytes$. Given n = 10000, the storage overhead in total is $297 + 9999 * 56 = 0.56Mb = 0.07MB$. Like the discussion before, it is not necessary for a user to store all credentials of the others in the system. A user can store the other users' credentials on demand.

7 Conclusion

In this paper, we analyzed the privacy threats in the current form of NHSs. We also proposed PrivateNH, a practical solution that enhances privacy for the guests w.r.t. the SP and privacy for the hosts w.r.t. malicious outsiders, while preserving the convenience and functionality offered by the current system. The proposed PrivateNH is compatible with the current Bitcoin blockchain system, and its effectiveness and feasibility in NHS scenario are also demonstrated by the security analysis and performance evaluation.

Acknowledgments. This work was supported by the National Key Research and Development Program of China under grant 2017YFB0802202 and by the Natural Science Foundation of China (NSFC) under grant 61702474. The work of Y. Fang was partially supported by US National Science Foundation under grant IIS-1722791.

References

1. Airbnb about us. https://www.airbnb.com/about/about-us/, accessed: March 2018
2. Airbnb engineering & data science. https://medium.com/airbnb-engineering/tagged/data-science/, accessed: March 2018
3. Bartoletti, M., Pompianu, L.: An analysis of bitcoin op_return metadata. In: Financial Cryptography and Data Security. pp. 218–230. Springer International Publishing (2017)
4. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In: 2015 IEEE Symposium on Security and Privacy. pp. 104–121 (2015)
5. Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptology. pp. 199–203. Springer US (1983)
6. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: Proceedings of IEEE 36th Annual Foundations of Computer Science. pp. 41–50 (1995)
7. Rosenfeld, M.: Overview of colored coins. https://bitcoil.co.il/BitcoinX.pdf/ (2012), accessed: March 2018
8. Ruffing, T., Moreno-Sanchez, P., Kate, A.: Coinshuffle: Practical decentralized coin mixing for bitcoin. In: Computer Security - ESORICS 2014. pp. 345–364. Springer International Publishing (2014)
9. Tschorsch, F., Scheuermann, B.: Bitcoin and beyond: A technical survey on decentralized digital currencies. IEEE Communications Surveys Tutorials **18**(3), 2084–2123 (2016)