Decentralized Blacklistable Anonymous Credentials with Reputation

Rupeng Yang[1,2], Man Ho Au[2(✉)], Qiuliang Xu[1(✉)], and Zuoxia Yu[2]

[1] School of Computer Science and Technology, Shandong University, Jinan 250101, China
orbbyrp@gmail.com, xql@sdu.edu.cn

[2] Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong
csallen@comp.polyu.edu.hk, zuoxia.yu@gmail.com

Abstract. Blacklistable anonymous credential systems provide service providers with a way to authenticate users according to their historical behaviors, while guaranteeing that all users can access services in an anonymous and unlinkable manner, thus are potentially useful in practice. Traditionally, to protect services from illegal access, the credential issuer, which completes the registration with users, must be trusted by the service provider. However, in practice, this trust assumption is usually unsatisfied.

In this paper, we solve this problem and present the decentralized blacklistable anonymous credential system with reputation (DBLACR), which inherits nearly all features of the BLACR system presented in Au et.al. (NDSS'12) but does not need a trusted party to register users. The new system also has extra advantages. In particular, it enables blacklist (historical behaviors) sharing among different service providers and is partially resilient to the blacklist gaming attack, where dishonest service providers attempt to compromise the privacy of users via generating blacklist maliciously.

Technically, the main approach to achieve DBLACR system is a novel use of the blockchain technique, which serves as a public append-only ledger. The system can be instantiated from three different types of cryptographic systems, including the RSA system, the classical DL system, and the pairing based system. To demonstrate the practicability of our system, we also give a proof of concept implementation for the instantiation under the RSA system. The experiment results indicate that when authenticating with blacklists of reasonable size, our implementation can fulfill practical efficiency demands.

1 Introduction

There always exists a conflict between users and service providers (SP) on the Internet. On the one hand, the SPs need to protect their services from illegal

R. Yang—This work was mainly done when doing the internship at The Hong Kong Polytechnic University.

W. Susilo and G. Yang (Eds.): ACISP 2018, LNCS 10946, pp. 720–738, 2018.
https://doi.org/10.1007/978-3-319-93638-3_41

users and users with misbehaviors, thus hope to know the exact identity and historical behaviors of each user. On the other hand, the users would like to protect their privacy, and thus hope to access services in an anonymous and unlikable manner.

The blacklistable anonymous credential system [9,25] is a good attempt to address this conflict. In this system, each SP maintains a blacklist to record users with misbehaviors, and a user attempting to access services of a SP is required to prove that he is legitimately registered and that he is not in the blacklist of the SP. Both the authentications and the maintenance of the blacklists are conducted in an anonymous and unlinkable fashion, thus privacy of users are well protected. Compared to traditional anonymous credential systems [8,10–14,16], the blacklistable anonymous credential system supports revocation of users, thus can protect SPs from users with misbehaviors. Moreover, compared to some other revocable anonymous credential systems [10,11], this is achieved without relying on a trusted third party, so in practice the blacklistable anonymous credential system is preferable.

Subsequently, there are a series of works following this line of research. Some of them consider how to improve the efficiency [24,26,31], and some others consider how to utilize historical behaviors of users in a cleverer way [5,6,27,29]. In particular, in [6], an anonymous credential system supporting fine-grained "blacklist" is proposed. In this system, instead of merely putting misbehaved users into the blacklist, the SP will rate behaviors of users in using the services. The rated scores can be either positive or negative for good and bad behaviors respectively, and belong to different categories based on types of behaviors rated. When authenticating, SPs can set complex policies about these scores, and a user attempting to access services of a SP needs to prove that he is legitimately registered and that his scores satisfy the policy of the SP. Likewise, all those operations are conducted in an anonymous and unlinkable fashion. For simplicity of notation, in this section, we still use the word "blacklist" to denote this fine-grained type of "blacklists".

To better explain how these blacklistable anonymous credential systems work, we illustrate the workflow for them in Fig. 1a. Generally speaking, a user who wants to access services of a SP first registers himself to the credential issuer and gets a credential back. Then he requests a policy from the SP and proves to the SP that he has a valid credential and that he satisfies the policy of the SP each time he wants to access the services of a SP. Behaviors of the user will be rated by the SP after he finishes using the services.

Note that to protect services from illegal access, the credential issuer must be trusted by the SP. Therefore, it is usually suggested that the credential issuer should be acted by the SP itself. However, in practice, this suggestion is often contradicted. Considering a SP who runs a forum about alcohol abuse, anyone who registers for this service runs the risk of revealing his drinking problem to the SP. So, at worst, no one would register for using this forum. As a result, the SP faces the dilemma of either trusting a third party credential issuer and suffering potential attacks or insisting on issuing credentials all by itself and suffering a

loss of potential users. A similar dilemma occurs when we consider the blacklist management. More precisely, services will be better protected if the SP can refer to blacklists of other SPs and further evaluate a user according to his historical behaviors when using other services, but it may bring additional security issues if the shared blacklists are fake. Besides these two problems, current blacklistable anonymous credential systems are also vulnerable to the blacklist gaming attack, where a malicious SP attempts to learn the identity of the user via providing a maliciously generated blacklist during the authentication.

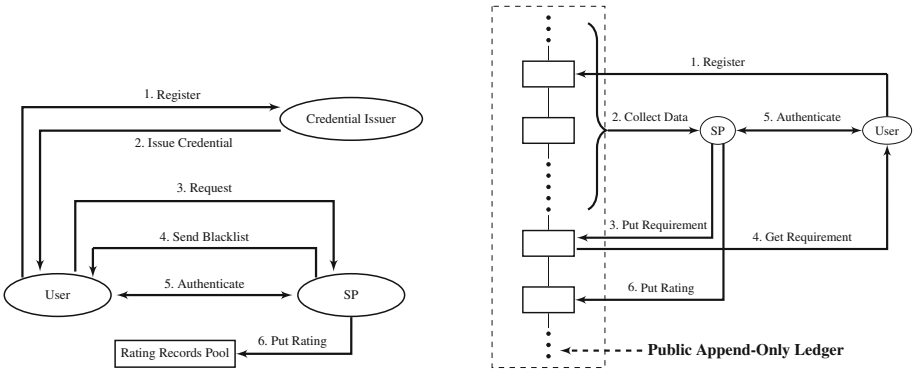


Fig. 1. Workflows of the traditional blacklistable anonymous credential systems (left) and our new decentralized blacklistable anonymous credential system with reputation (right).

The first problem, namely the requirement of a trusted credential issuer, is partially solved in [20], in which a decentralized anonymous credential system is constructed. In particular, in [20], a blockchain based public append-only ledger is employed to replace the credential issuer, and to register in the system, a user just needs to put his personal information attached with his credential to the ledger. When authenticating, a user proves to a SP that his credential belongs to a set, which is selected by the SP from credentials of all registered users. However, in [20], revocation of users is not considered, and it is unknown whether their techniques can be applied to decentralize current blacklistable anonymous credential systems. Besides, the other two problems, namely the blacklist management problem and the blacklist gaming attack, are still open.

1.1 Our Results

In this paper, we solve these open problems by presenting the decentralized blacklistable anonymous credential system with reputation (DBLACR), whose workflow is illustrated in Fig. 1b. More precisely, similar to that in [20], in our new

system, there is no central credential issuer, and a user registers via uploading his credential together with his personal information to the public append-only ledger, which can be instantiated with the blockchain technique. Each SP collects data from the ledger automatically and put its requirement, including the selected candidate users set and the blacklist, to the ledger regularly. When a user wants to access a service of a SP, he first gets the latest requirement of the SP from the ledger, then he checks its validity and whether he satisfies it. If both tests are passed, he then proves to the SP that he satisfies its requirement. The user can access the service if the proof is valid, and scores for his behavior in using the service will be rated and put on the ledger by the SP then.

The DBLACR system can achieve enhanced security guarantee in the following three aspects. We also give a comparison between our system and existing blacklistable (or decentralized) anonymous credential systems in Table 1.

- *The registration is decentralized.* In our new system, no trusted credential issuer is needed, and each SP can select candidate users by itself. Thus, security for the SPs is improved. Note that the user does not need to indicate which service he would like to access when registering and only the fact that he wants to access at least one service in the system is revealed. Thus, the real purpose of the user is well hidden if there are some common and insensitive services in the system. Therefore, our solution will not compromise the privacy of users.
- *There is a consistency between the used blacklist and the shared blacklist for any SP.* This is because a SP will put his own used blacklist in the public append-only ledger, thus cannot share a fake blacklist without being caught. The property implies that to refer to blacklists of other SPs, a SP only needs to trust that they will not *use* a fake blacklist when conducting their own authentication protocols instead of trusting that they will not *share* a fake blacklist. So, to a great extent, the SP can employ blacklists of other SPs safely and makes better evaluations for users.
- *The system is partially resilient to the blacklist gaming attacks, thus provides a better protection for the privacy of users during the authentication.* This is achieved in two aspects. First, as in our system SPs update their blacklists regularly, a malicious SP can only make a less powerful passive blacklist gaming attack in each time period, where it fixes a blacklist in the beginning. Besides, in our system, a user can learn whether he could pass the verification in advance and will not attempt to launch an authentication if he does not satisfy the requirement, thus less information is leaked from authentication results. We give a more detailed discussion on how these two modifications could boost the security in Sect. 3.

Our Techniques. We construct decentralized blacklistable anonymous credential system with reputation by introducing the blockchain technique to current blacklistable anonymous credential systems and employ it as a public append-only ledger to store credentials and blacklists. However, there exists issues when integrating the blockchain technique and current (blacklistable) anonymous credential systems. To see this, recall that in a blockchain-based (blacklistable)

Table 1. The comparison.

	Decentralized registration	Blacklist supporting	Blacklist sharing	Blacklist-Gaming resilience
BLAC[25]	✗	†	‡	✗
EPID[9]	✗	†	‡	✗
PEREA[26]	✗	†	✗	✗
PE(AR)^2[31]	✗	†	✗	✗
FAUST[24]	✗	†	✗	✗
BLACR[6]	✗	✓	‡	✗
EXBLACR[27]	✗	✓	‡	✗
PERM[5]	✗	✓	✗	✗
FARB[29]	✗	✓	✗	✗
DAC[20]	✓	✗	-	-
Ours	✓	✓	✓	✓ *

† : only a basic blacklist is supported.
‡ : blacklists can be shared if SPs trust each other.
✓ * : the system is partially resilient to the blacklist gaming attacks.

decentralized anonymous credential system, users registers by putting its credential to the ledger. Then, to argue that he is legitimately registered, a user just proves that he knows the secret key for a credential stored in the ledger. To make the proof size constant, cryptographic accumulator is desired to accumulate all credentials in the ledger. However, in most (if not all) current (blacklistable) anonymous credential systems, credentials are commitments of the users' secret keys, thus are either (1) points in an elliptical curve, which cannot be accumulated using existing number-theory-based accumulators or (2) exponential in the users' secret keys (i.e., $C = g^s h^r$ where s is a secret key, C is the corresponding credential, r is a random number, and g, h are group elements), which bring expensive double discrete logarithm proof[1]. In both cases, the practicability of the system are reduced.

In this work, we solve these issues by presenting a new method to construct credential systems. In particular, the secret key of a user is two large primes p, q and his credential is another prime $n = 2pq + 1$. The credential can be accumulated by a strong-RSA assumption based accumulator and one can efficiently prove that his secret key relates to a credential in an accumulator. As a result, the efficiency of the system is boosted. The experiment result in Sect. 6 demonstrates that our new system is quite practical. Especially, it implies a decentralized anonymous credential system that is as much as 30 times faster for a user to generate an authentication, when compared with the decentralized anonymous credential sytem in [20].

[1] The decentralized anonymous credential system in [20] also suffers from this problem.

2 Notation

For a finite set $\mathcal{S}$, we use $\|\mathcal{S}\|$ to denote the size of $\mathcal{S}$ and write $x \xleftarrow{\$} \mathcal{S}$ to indicate that x is sampled uniformly from $\mathcal{S}$. We write $negl(\cdot)$ to denote a negligible function. For two random variables $\mathcal{X}$ and $\mathcal{Y}$, we write $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$ to denote that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable. We will use a few cryptographic assumptions, including the strong RSA assumption, the LD-RSA assumption, the discrete logarithm assumption, the DDH assumption, and the DDH-II assumption. We will also use cryptographic primitives, such as zero-knowledge proof of knowledge, commitment scheme, dynamic accumulator, CL signature, and public append-only ledger. Note that all zero-knowledge proofs of knowledge used in this paper are non-interactive and admit an additional message as input, thus it is also called signature proof of knowledge (SPK), and is usually written as $SPK\{(w) : \mathcal{S}\}[m]$, for a statement $\mathcal{S}$ with witness w and additional message m. Due to lack of space, we do not provide detailed descriptions for the used assumptions and cryptographic primitives and refer the readers to the full version of this paper [30] for more details.

3 Syntax and Security Goals

3.1 The Syntax

There are two types of entities, namely the users and the service providers, and a public ledger in the DBLACR system, and the system consists of the following protocols:

- **Setup.** To setup the system, a trusted party is employed to generate the public parameter of the system. Note that this party is only used in the setup phase and we only need to trust that it will generate the public parameter honestly and will erase all the internal states of the generation process.
- **Registration.** In this protocol, a user registers himself to the system. To complete this task, a user just needs to put some information to a public ledger, which should include some auxiliary proof data and his attributes to aid the SPs in deciding whether to accept the user as a valid candidate user for accessing their services.
- **Authentication.** This protocol is executed between a user and a SP. The user attempts to access services of the SP in an anonymous and unlinkable fashion, and the SP will accept the user if and only if the user fulfills its requirement. Here, the requirement includes three parts, namely the candidate users set $\mathcal{C}$, the policy $\mathcal{P}_R$ and the rating records list $\mathcal{L}$. Our system can support a policy of any DNF formula, whose inputs are accumulated scores for a user's behaviors in different category. We refer the readers to the full version of this paper [30] for a more detailed explaination of the requirement.
- **Interaction with The Ledger.** The public ledger in this system is public and accessible to every participant, including the users and the SPs. In addition, the SPs can put data to the ledger. In particular, it can upload

its requirement to the ledger regularly. Besides, it can submit a rating for the anonymous user in an authentication event and submit a revocation of a rating record submitted by itself.

3.2 The Security

We refer the readers to the full version of this paper [30] for a formal security definition of our decentralized blacklistable anonymous credential system with reputation. Here, we only highlight a few security properties of the system that are most concerned in practice:

- **Authenticity.** The authenticity property guarantees that SPs are assured to accept authentication events only from users satisfying their requirements.
- **Anonymity.** The anonymity property guarantees that all a SP learns from an authentication is if the authenticating user satisfies its current requirement.
- **Non-frameability.** The non-frameability property guarantees that if a SP is honest, then users satisfying the current requirement of this SP can always successfully authenticate to it.
- **Sybil-Attack Resilience.** The Sybil attack [18] allows users to get new credentials after their current credentials are blacklisted, thus may expose services to users with misbehaviors. In our new system, since users register to the system via uploading their identities to the public ledger, the Sybil attack can be prevented if SPs only select users whose identities have not been uploaded previously as candidate users.
- **Authenticity of Registration.** This property guarantees that SPs can decide which users are legitimate directly and do not have to resort to a third party. The property can provide a better protection for SPs.
- **Privacy of Registration.** This property guarantees that only the fact that the registered user hopes to access at least one service supported by the system can be learned from a registration event. As personal information is usually required in registration, this property is significant in protecting the privacy of users.
- **Consistency of Blacklists.** This property guarantees that each rating record selected by a SP will be honestly assessed unless there exist SPs hoping to expose their services to possible malicious users. The property can greatly reduce the requirement of trust when using rating records from other SPs.
- **Blacklist-Gaming Attack Resilience.** The blacklist gaming attack [26] allows a SP to compromise the privacy of users via generating blacklists (requirements) maliciously. Our new system is partially resilient to the blacklist gaming attack and this is achieved in the following two aspects:
  - First, in our new system, the SPs can only update their requirements regularly, thus in each time period, the requirement used in authentication protocols is fixed. Compared to that in previous systems, where the malicious SP can use an adaptively chosen blacklist during each authentication event, the privacy of users is better protected now. To demonstrate this, we consider the following scenario. Via some auxiliary information,

a SP conjectures that the next authentication event is launched by the same user who launches a previous authentication event with identifier "t". In current blacklistable anonymous credential systems, the SP can definitely verify its conjecture via providing a blacklist with merely "t" in it. However, this attack is not applicable in our system since no SP is able to use a temporary blacklist in an authentication.

- Next, in our new system, as a user could obtain the latest requirement of a SP from the public ledger, he can check whether he is able to pass the verification in advance and will not attempt to launch the authentication protocol if he does not satisfy the requirement. To see why this can better protect the privacy of users, we consider the following scenario. Again, via some auxiliary information, a SP learns that the following authentication events will be launched by one of two lists of users. It also learns whether each user in these two lists satsifies a pre-defined requirement. Previously, even restricting the malicious SP to the pre-defined requirement, it can still determine the list of users in use if there exists an index i that the ith users in the two lists are different in satisfying the requirement. In contrast, in our new system, the malicious SP can learn nothing if the numbers of users satisfying the requirement in these two lists are identical.

We remark that the first four properties are already achieved in current blacklistable anonymous credential systems. The property "authenticity of registration" and the property "privacy of registration" have also been achieved previously, but no system has these two properties simultaneously, and our system is the first one that can protect both the security of the SPs and that of the users in the registration. The last two properties are new security properties that are only available in our new system.

4 General Construction

In this section, we provide a general framework for constructing the decentralized blacklistable anonymous credential system with reputation. We start by introducing a few algorithms and protocols used for building the system. Then we describe how to combine these components to complete the construction.

4.1 Building Blocks

Our DBLACR system can be instantiated from various public key systems, and for each public key system, we need the following sub-protocols to help build our system:

A Key Generation Algorithm. On input a security parameter 1^λ, the key generation algorithm returns a public key/secret key pair, namely, $(pk, sk) \leftarrow KeyGen(1^\lambda)$. In our system, the public key is the credential of a user, and the secret key is the witness for it. So we require that the key generation algorithm has the following properties:

- *Verifiability.* There exists a polynomial-time algorithm T s.t. $T(pk, sk) = 1$ iff the pair (pk, sk) is a legal key pair of the public key system.
- *Onewayness.* Given the public key pk, it is computationally hard to compute a secret key sk such that $T(pk, sk) = 1$.
- *Collision Resistance.* It is computationally hard to find a pair of different secret keys (sk_1, sk_2) and a public key pk such that $T(pk, sk_1) = T(pk, sk_2) = 1$.

A Ticket Generation Algorithm. On input a secret key sk, the ticket generation algorithm generates a ticket for sk, namely, $\tau \leftarrow TicketGen(sk)$. In our system, each ticket will be the representation of an authentication event, and an authentication event with a ticket τ will be regarded as launched by the owner of a secret key sk iff $S(sk, \tau) = 1$. So we require that the ticket generation algorithm has the following properties:

- *Verifiability.* There exists a polynomial-time algorithm S s.t. $S(sk, \tau) = 1$ iff τ is a valid ticket of sk.
- *Indistinguishability.* Let $(pk, sk) \leftarrow KeyGen(1^\lambda)$, then for any probabilistic polynomial time adversary $\mathcal{A}$, $\Pr[b \xleftarrow{\$} \{0,1\}; b \leftarrow \mathcal{A}^{\mathcal{O}_b}(pk)] \leq 1/2 + negl(\lambda)$, where $\mathcal{O}_0$ outputs a ticket of sk each time invoked, and $\mathcal{O}_1$ outputs a random element in the range of the ticket generation algorithm each time.
- *Verifying Consistency.* For any secret keys sk_1, sk_2, if there exists a τ s.t. $S(sk_1, \tau) = S(sk_2, \tau) = 1$, then for any τ' in the range of the ticket generation algorithm, we have $S(sk_1, \tau') = S(sk_2, \tau')$.
- *Connectivity.* Let $(pk, sk) \leftarrow KeyGen(1^\lambda)$, $\tau \leftarrow TicketGen(sk)$, and sk' be a secret key s.t. $S(sk', \tau) = 1$, then given (pk, sk'), one can efficiently compute sk.

An SPK System Proving the Possession of the Secret Key. We need a SPK system to prove that the prover possesses the secret key sk of a given public key pk. Formally, the prover needs to prove $SPK\{(sk) : T(pk, sk) = 1\}$.

An SPK System Proving the Validity of a Public Key and a Ticket. We need a SPK system proving that the prover possesses a secret key sk for a given ticket τ and the secret key is associated with a public key in a given set $\mathcal{C}$. Formally, the prover needs to prove $SPK\{(sk, pk) : S(sk, \tau) = 1 \wedge T(pk, sk) = 1 \wedge pk \in \mathcal{C}\}$.

An SPK System Proving the Fulfilment of a Policy. We also need a SPK system proving that the prover possesses a secret key sk for a given ticket τ and the secret key represents a user whose scores evaluated according to a policy $\mathcal{P}_R$ and a rating records list $\mathcal{L}$ satisfies $\mathcal{R}_R$. For simplicity of description, in this section, we define a boolean function $\mathcal{E}$ that outputs 1 iff the latter condition is satisfied. Then, the prover needs to prove $SPK\{(sk) : S(sk, \tau) = 1 \wedge \mathcal{E}(\mathcal{P}_R, \mathcal{L}, sk) = 1\}$.

4.2 The Construction

Now, we present the general construction of our DBLACR system, which is built on the sub-protocols shown in Sect. 4.1 and a public append-only ledger with ideal functionality $\mathcal{F}_{BB}^{\star}$, whose formal definition is given in the full version [30]. Formally, we have:

Setup. On input a security parameter 1^{λ}, a trusted party runs the setup algorithm for each sub-protocol of each public key system and outputs all those generated public parameters as the public parameter for the DBLACR system.

Registration. To register himself to the system, a user with auxiliary proof data aux and attributes $attr$ first generates his public key/secret key pair $(pk, sk) \leftarrow KeyGen(1^{\lambda})$ for one of the supported public key systems. Then he computes $\Pi_R \leftarrow SPK\{(sk) : T(pk, sk) = 1\}[aux \| attr]$. Finally, he stores the tuple $(Nym, pk, \Pi_R, attr, aux)$ to the public ledger via $\mathcal{F}_{BB}^{\star}$, where Nym is his pseudonym in the public ledger. We remark that here the user can use a temporary pseudonym and not a permanent one.

Authentication. In this protocol, a user uid attempts to authenticate with a service provider sid. Interactions between these two parties are summarized in Fig. 2. For the clarity of presentation, here we assume that there are k public key systems employed in our system, and denote them as $\Psi_1, \ldots, \Psi_k$ respectively. All algorithms in Ψ_i will be labeled with a superscript "(i)", and w.l.o.g. we assume that the user uid chooses the first public key system when registering.

In more detail, in this protocol, the user uid first downloads the requirement $(\mathcal{C}, \mathcal{P}_R, \mathcal{L})$ for accessing services of sid from the public ledger. Then he verifies the validity of this requirement. If the requirement is valid, the user then checks whether he satisfies the requirement. If not, he aborts the protocol even without communicating with sid. Otherwise, uid sends a request to sid and gets a challenge $m \| sid'$ back, where m is a randomly chosen bit string whose length is polynomial in the security parameter. Then, uid checks whether $sid = sid'$ and if so he generates a ticket $\mathcal{T}$ and a proof Π_A, and sends $(\mathcal{T}, \Pi_A)$ to sid. More precisely, to generate the ticket $\mathcal{T}$, the user computes $\tau_1 \leftarrow TicketGen^{(1)}(sk)$, randomly samples τ_i in the range of $TicketGen^{(i)}(\cdot)$ for $i \in [2, k]$, and sets $\mathcal{T} = \{\tau_1, \ldots, \tau_k\}$. To generate the proof Π_A, the user computes $\Pi_A = SPK\{(sk, pk) : \bigvee_{i=1}^{k} (T^{(i)}(pk, sk) = 1 \wedge pk \in \mathcal{C}_i \wedge S^{(i)}(sk, \tau_i) = 1 \wedge \mathcal{E}^{(i)}(\mathcal{P}_R, \mathcal{L}^{(i)}, sk) = 1)\}[m \| sid]$, which is constructed by employing the technique in [15] to combine the proof of "validity of a public key and a ticket" and the proof of "fulfillment of a policy" for each public key system, where $\mathcal{C}_i$ consists of all public keys of Ψ_i that are in $\mathcal{C}$, and $\mathcal{L}^{(i)}$ consists of all rating records in $\mathcal{L}$ but for each record the ticket $\mathcal{T}' = (\tau_1', \ldots, \tau_k')$ is replaced with τ_i'. Upon receiving the response $(\mathcal{T}, \Pi_A)$, sid verifies the proof and sends the result, which will be "accept" iff the proof is valid, back to uid.

Interaction with Ledger. To obtain data from the public ledger, a participant just needs to submit a "retrieve" request to $\mathcal{F}_{BB}^{\star}$. To put data to the public ledger, a SP just needs to submit a "store" request together with its permanent pseudonym and its data to $\mathcal{F}_{BB}^{\star}$. The submitted data vary depending on the

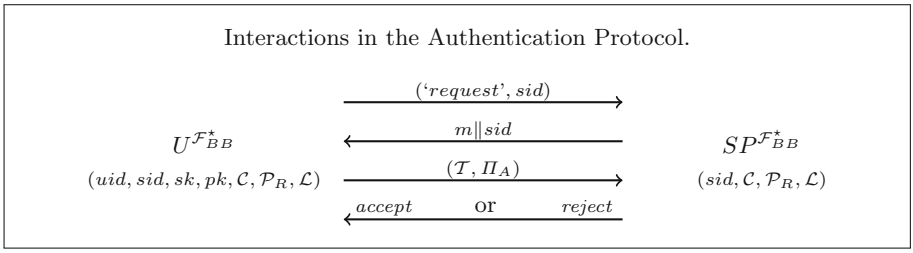


Fig. 2. Interactions in the authentication protocol. Here, we use "U" to denote the user, and "SP" to denote the service provider.

purpose of the SP. In particular, when a SP would like to submit a rating s, it needs to put a tuple $(rid, \mathcal{T}, s, \Gamma)$ to the public ledger, where rid is a unique string identifying this rating record, $\mathcal{T}$ is the ticket for the rated authentication event, and Γ is the transcript of this authentication event, which is used to prove that the rated authentication event can be accepted by this SP. When a SP would like to submit a revocation of a rating record rid, it needs to put a tuple $(\text{'}revoke\text{'}, rid)$ to the public ledger. When a SP would like to publish a new requirement, it first generates a valid requirement $(\mathcal{C}, \mathcal{P}_R, \mathcal{L})$, then puts it to the public ledger. To generate a valid requirement, apart from meeting those demands listed in Sect. 3.1, the SP should further ensure that each selected user in $\mathcal{C}$ is attached with a valid proof Π_R. We remark that all those data uploaded to the public ledger will not be verified in this phase, instead, the verification will be postponed until the data are used.

The Security. Security of our system is guaranteed by Theorem 4.1 stated as following. We refer the readers to the full version of this paper [30] for proof of Theorem 4.1.

Theorem 4.1. *The system presented in Sect. 4.2 is a secure DBLACR system if each sub-protocol has the properties demanded in Sect. 4.1.*

5 The Instantiations

To demonstrate the utility of our general framework, in this section, we instantiate sub-protocols defined in Sect. 4.1. The sub-protocols can be instantiated under three different types of public key systems, namely, the classical DL system, the pairing based system, and the RSA system. Here, we only present a high-level idea on how to instantiate the system from the RSA system and refer readers to the full version [30] for detailed instantiations from all three systems.

Our RSA based sub-protocols works in a quadratic residue group QR_N with a generator $\mathfrak{g}$, where N is the product of two big safe prime numbers. The secret key of the system is two safe primes p and q that $2pq + 1$ is also a prime and the public key is $n = 2pq + 1$. To generate a ticket $\tau = (b, t)$, one first samples $r \xleftarrow{\$} \mathbb{Z}_N$, then computes $b = \mathfrak{g}^r \mod N$ and $t = b^{p+q} \mod N$.

To prove the possession of a secret key $sk = (p, q)$ for a properly generated public key $pk = n$, the user works in two steps. First, the prover proves that $(n-1)/2$ is a product of two *primes*. This can be accomplished by employing a variant the proof system proposed in [21]. Then, the prover needs to prove that he knows two numbers p, q with identical lengths that satisfy $2pq + 1 = n$. To instantiate this proof system, we apply the framework presented in [23], which provides a simple method to prove knowledge of discrete logs that are in an interval and fulfil a set of equations over groups of unknown order.

Then, to construct the SPK system proving that a user possesses a secret key $sk = (p, q)$ associated with a public key $pk = n$ in a given set $\mathcal{C}$, we apply the approach presented in [17], which also builds on the framework of [23]. More precisely, the prover first accumulates public keys in $\mathcal{C}$ with a dynamic accumulator, then proves in zero-knowledge the possession of the secret key of a public key in the accumulator. To further prove that a given ticket $\tau = (b, t)$ is also generated from the same secret key, the prover just plugs the equation $t = b^p b^q$ into the above statement.

Finally, to prove the fulfilment of a policy, we exploit the idea in [6] to construct the proof system, but will employ RSA-based cryptographic primitives instead of those pairing-based ones. In particular, we will apply strong-RSA assumption based additive homomorphic commitment scheme [19] and CL signature scheme [12], and we will also apply the framework in [23] to construct our proof system.

6 The Implementation

To demonstrate the practicability of our system, in this section, we provide a proof of concept implementation for it. The implementation includes two relatively independent parts, namely, the public ledger part and the credential system part, and we describe the results for them in Sect. 6.1 and in Sect. 6.2 respectively.

6.1 The Public Ledger

First, we explore how the public ledger could be realized. Recall that the public ledger can be instantiated via the blockchain technique. So, we choose the Bitcoin and the Ethereum, which are the two most popular blockchain technique instantiations currently, as the test object. The test is conducted on a personal computer with a 3.16 GHz Intel(R) Core(TM)2 Duo Processor E8500, 8 GB RAM and 500 GB disk, running ARCHLinux version 4.10.6. The Bitcoin client run in the experiment is Bitcoin Core Version 0.14.0 and the Ethereum client is go-ethereum 1.5.9. The result is summarized in Table 2.

The row "Market Cap" indicates the market capitalizations of each instantiation, and the data come from [1]. This can reflect the robustness of the blockchain to some extent. The row "Initial Data Size" and the row "Initial Sync Time" indicates the disk space and time needed before one could employ

Table 2. Comparison of public ledger instantiations.

	Bitcoin	Ethereum
Market Cap	19257718797 USD	4376127411 USD
Initial Data Size	118 GB	15 GB
Initial Sync Time	9 h	5 h
Ease of Use	Difficult	Easy
Data Size Limit	80bytes	*
Cost	0.5342 USD	0.0225 USD
Confirmation Time	6 min/70 min	a few seconds/3 min

the public ledger. The row "Ease of Use" and the row "Data Size Limit" indicates the accessibility of using blockchain as a public ledger. For Bitcoin, in each transaction, there exists a field OP_RETURN allowing one to put up to 80 bytes arbitrary data [3] on it, but it seems that the Bitcoin community do not hope people to use this field, and the client Bitcoin Core also does not provide a convenient way to implement this functionality. Thus, we test this facility via a third party open source project on GitHub [22]. For Ethereum, putting data in a transaction is natively supported. There is also no explicit limits on the size of data put in a transaction, but for each block, there is a block gas limit, which is about 4 millions for current blocks. As it will consume gas to attach data to a transaction, one could only put dozens to hundreds kilobytes data in one transaction now according to the content of the data. The row "Cost" indicates the amount of money cost to put data on the blockchain. For Bitcoin, this is the transaction fee for rewarding the miners. According to statistics (data from [4]), to hope miners to deal with the transaction immediately, the transaction fee should be above 1.8×10^{-6} BTC per byte, and for our purpose, which will send a transaction of about 250 bytes (about 200 bytes for the basic transaction and about 50 bytes for the attached data), the transaction fee should be 0.00045 BTC, which is about 0.5342 USD according to the price of 1 BTC at April 14th, 2017. For Ethereum, the cost comes from the gas consumed. Currently, each gas is about 2×10^{-8} Ether, and according to the yellow paper of Ethereum [28], a transaction will cost 21000 gas for itself, and each non-zero byte put in the data field will cost 68 gas. In our experiment, we put 32 bytes in a transaction and this cost us 0.00047 Ether, or about 0.0225 USD according to the price of 1 Ether at April 14th, 2017. The row "Confirmation Time" indicates the time needed to wait for the transactions and the data to be confirmed. For Bitcoin, on average, it will take 10 min to generate a new block, so on average, it will take about 5 min to see the data appear on the blockchain, and about 1 h to confirm that the data are put in the blockchain (6 confirmation). For Ethereum, the new block appears every a few seconds, so the data will appear on the blockchain immediately. As claimed by the Ethereum Blog [2], 10 confirmation in Ethereum is enough to

achieve a similar degree of security as that of 6 confirmation in Bitcoin, so it may take about 3 min to wait for the confirmation of the transaction/data.

From the experiment result, we observe that neither the Bitcoin nor the Ethereum can support large data storage. So in practice, to use them as a public ledger, one should first upload the data to some public cloud, then put the link (40 to 60 bytes for a dropbox link and 10 to 20 bytes if google url shorten service is used) and hash value of the data (32 bytes if SHA-256 is used) to the blockchain. In this way, the functionality of the public ledger still reserves. Another problem is that while it is quite easy for a service provider to sync and maintain a Bitcoin blockchain or an Ethereum blockchain in its server, this is not the case for a normal user. To tackle this problem, we suggest users with constrained devices to use a lightweight client or refer to an online service to complete interactions with the public ledger (they could exploit multiple approaches to retrieve data to boost the security), and this will not harm the security as long as there exists services providing correct Bitcoin or Ethereum blockchain information. When comparing the Bitcoin and the Ehtereum, it seems that the Bitcoin blockchain is more robust, while the Ethereum is also very secure and is much more convenient to use. Thus, in practice, Ethereum seems a better choice and we prefer to employ Ethereum to realize our system.

6.2 The Credential System

Then we examine the practicality of the credential system part of our system. The implementation is for the RSA-based instantiation. To simplify the criterion for evaluating the experiment result, we only consider a simple policy with a single category, threshold 0, and no adjusting factor, and a rating records list with one blacklist. The experiment is conducted on a Macbook Pro with 8 GB of 1866 MHz LPDDR3 onboard memory and a 2.7 GHz dual-core Intel Core i5 processor, running OSX 10.12.4. The test code is written in C based on the OPENSSL library (version 1.0.2).

There are two main operations, namely the registration and the authentication, in the system, thus our experiment also focuses on the performance of these two protocols. First, we test the performance of the registration protocol, including the time for a user to generate a credential, the time for a service provider to verify a credential, and the credential size. As the user may already have a key pair when joining the system, the time consumption for generating a credential is tested in two modes, namely the normal mode, where the user needs to generate both the key pair and the proof, and the pre-computation mode, where credential is generated on a given public key/secret key pair. Then, we test the performance of the authentication protocol, including the time for generating a proof, the time for verifying a proof, and the size of the proof. Since the user can access the requirement in advance and precompute some parts, we will test the times for generating a proof both with and without pre-computation.

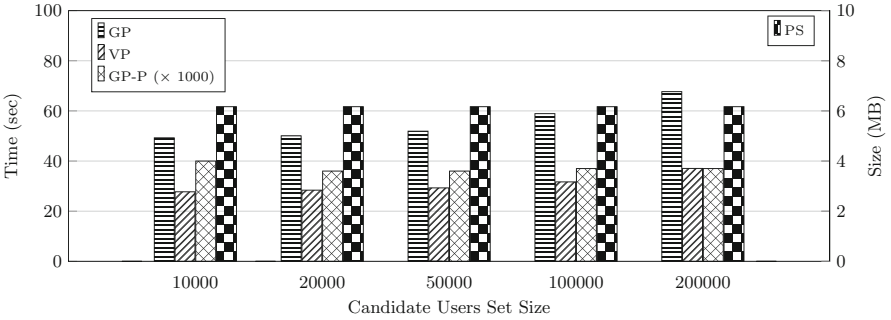
The experiment performance is measured under different parameters, including the security parameter, the candidate users set size, and the blacklist size. In more detail, we will consider security parameters of 1024 bits, 2048 bits, and 3072 bits, which can achieve a security strength of about 80 bits, 112 bits, and 128 bits respectively (according to [7]), and summarize the performance of our system under different security parameters in Table 3. we will consider candidate users set size of 10000, 20000, 50000 100000, and 200000, and blacklist size of 1000, 2000, 3000, 4000, and 5000, and summarize the performance of the authentication protocol under these parameters in Fig. 3. When analyzing the relation between the performance and one particular parameter, the other two parameters will be set as default, and the default values of the security parameter, the candidate users set size, and the blacklist size are 2048 bits, 50000, 3000 respectively. Besides, we also test the performance for the setting with an empty blacklist, which is exactly the scenario considered in [20], and compare our results with theirs in Fig. 4.

Table 3. The performance of the registration protocol and the authentication protocol under different security parameters with 50000 users and 3000 blacklist records.

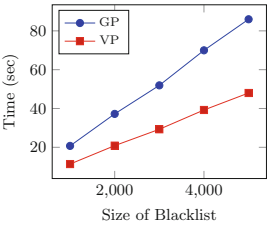
	GC	GC-P	VC	CS	GP	GP-P	VP	PS
1024 bits	1.316 s	0.153 s	0.047 s	70.1 KB	10.878 s	0.021 s	5.686 s	3.1 MB
2048 bits	19.296 s	0.932 s	0.295 s	139.9 KB	51.917 s	0.036 s	29.289	6.2 MB
3072 bits	69.578 s	2.959 s	0.910 s	209.8 KB	142.123 s	0.047 s	84.872 s	9.3 MB

Here, we use GC, GC-P, and VC to denote time consumed in generating a credential, generating a credential with pre-computation, and verifying the validity of a credential respectively; we use GP, GP-P, and VP to denote time consumed in generating a proof, generating a proof with pre-computation, and verifying a proof respectively; and we use CS and PS to denote the size of a credential and an authentication proof respectively.

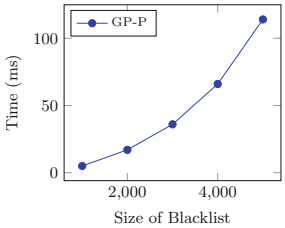
From the experiment results, we can conclude that our system is quite practical when deployed in practice. First, at the user side, the time consumption is extremely low if pre-computation is enabled. At the service provider side, it is also fairly fast to verify the validity of a credential, but it seems time-consuming to verify the validity of a proof. Nonetheless, the service provider often controls more computation resources, so it will take less time to wait for the verification in real world applications. Besides, the size of the credential and the proof is also not very large, thus the communication cost of our system is also quite low. One advantage of our system is that both the communication cost and the computation cost hardly increase with the increasing of the candidate users, i.e. it is scalable in the number of supported users. This is important for the usefulness of our system, since a large number of registered users is always desired to protect the privacy of particular users. However, this is not the case for the blacklist size, as both the communication cost and the computation cost grow linearly



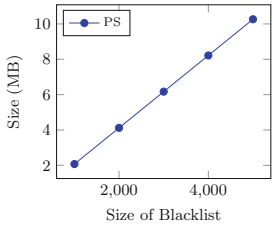
(a) Performance for the authentication protocol under different candidate users set size with security parameter 2048 bits and 3000 blacklist records. GP, GP-P and VP are times for generating a proof without pre-computation, generating *1000* proofs with pre-computation, and verifying a proof respectively, and PS is the size of the authentication proof.



(b) Time for generating and verifying a proof under different blacklist size with security parameter 2048 bits and 50000 users. GP and VP are times for generating (without pre-computation) and verifying a proof respectively.



(c) Time (in milliseconds) for generating a proof with pre-computation under different blacklist size with security parameter 2048 bits and 50000 users.



(d) Size (in Megabyte) of an authentication proof under different blacklist size with security parameter 2048 bits and 50000 users.

Fig. 3. Performance of our system under different candidate users set size and blacklist size.

with the size of the blacklist. So, it is better to employ our system in settings with a small blacklist. We leave how to upgrade the system to scalable in the size of the blacklist as an open problem.

When comparing the efficiency of our system with that in [20], we observe that our efficiency is much better than theirs. More precisely, our system can be as much as 30 times faster than theirs for a user to generate an authentication, and can be as much as 4 times faster for a service provider to verifiy. Also, the communication cost of our system is only about 15% to 45% of theirs. Thus our system is preferable even in scenarios that no revocation is needed.

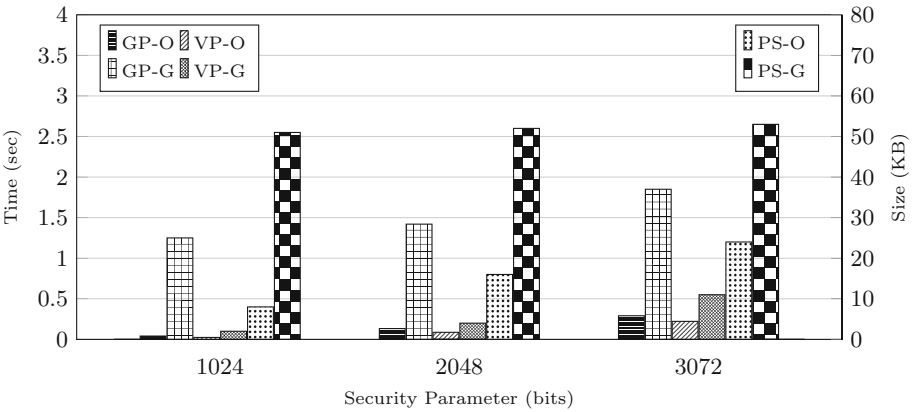


Fig. 4. Comparison between the performance of our authentication protocol with empty blacklist and the performance of the authentication protocol in [20]. Since in their experiment, accumulator is computed separately, we also do not count time consumed by this part in the test. Here, GP-O and VP-O are times for generating an authentication proof without pre-computation and verifying an authentication proof in our system respectively; GP-G and VP-G are respective times in [20]; and PS-O and PS-G are our authentication proof size and theirs respectively.

7 Conclusion

In this paper, we explore how to employ the blockchain technique to solve several open problems for previous anonymous credential systems, including trust of the credential issuer and the blacklist gaming attack. Note that, our system is only partially resilient to the blacklist gaming attack. Especially, a malicious verifier can still learn information such as the number of successfully authenticated users in a time period and may use this information to compromise the privacy of users. We leave how to construct a blacklistable anonymous credential system that is fully resilient to the blacklist gaming attack as an open problem.

Acknowledgement. We appreciate the anonymous reviewers for their valuable suggestions. Part of this work was supported by the National Natural Science Foundation of China (Grant No. 61602396, U1636205, 61572294, 61632020, 61602275), the MonashU-PolyU-Collinstar Capital Joint Lab on Blockchain and Cryptocurrency Technologies, and from the Research Grants Council of Hong Kong (Grant No. 25206317).

References

1. Cryptocurrency market capitalizations. https://coinmarketcap.com/. Accessed 15 Apr 2017
2. On slow and fast block times. https://blog.ethereum.org/2015/09/14/on-slow-and-fast-block-times/
3. Op_return. https://en.bitcoin.it/wiki/OP_RETURN. Accessed 15 Apr 2017

4. Predicting bitcoin fees for transactions. https://bitcoinfees.21.co/. Accessed 14 Apr 2017
5. Au, M.H., Kapadia, A.: PERM: practical reputation-based blacklisting without TTPs. In: CCS, pp. 929–940. ACM (2012)
6. Au, M.H., Kapadia, A., Susilo, W.: BLACR: TTP-free blacklistable anonymous credentials with reputation. In: NDSS (2012)
7. Barker, E.: Recommendation for key management-part 1: general (revision 4) (2015)
8. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and non-interactive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_20
9. Brickell, E., Li, J.: Enhanced privacy id: a direct anonymous attestation scheme with enhanced revocation capabilities. In: Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society, pp. 21–30. ACM (2007)
10. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_7
11. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_5
12. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Persiano, G., Galdi, C. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36413-7_20
13. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_4
14. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. Commun. ACM **28**(10), 1030–1044 (1985)
15. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_19
16. Damgård, I.B.: Payment systems and credential mechanisms with provable security against abuse by individuals. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 328–335. Springer, New York (1990). https://doi.org/10.1007/0-387-34799-2_26
17. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in *ad hoc* groups. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_36
18. Douceur, J.R.: The sybil attack. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45748-8_24
19. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052225
20. Garman, C., Green, M., Miers, I.: Decentralized anonymous credentials. In: NDSS (2014)

21. Gennaro, R., Micciancio, D., Rabin, T.: An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In: CCS, pp. 67–72. ACM (1998)
22. Greenspan, G.: Project php-op_return. https://github.com/coinspark/php-OP_RETURN. Accessed 15 Apr 2017
23. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 571–589. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_34
24. Lofgren, P., Hopper, N.: FAUST: efficient, TTP-free abuse prevention by anonymous whitelisting. In: Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society, pp. 125–130. ACM (2011)
25. Tsang, P.P., Au, M.H., Kapadia, A., Smith, S.W.: Blacklistable anonymous credentials: blocking misbehaving users without TTPs. In: CCS, pp. 72–81. ACM (2007)
26. Tsang, P.P., Au, M.H., Kapadia, A., Smith, S.W.: PEREA: towards practical TTP-free revocation in anonymous authentication. In: CCS, pp. 333–344. ACM (2008)
27. Wang, W., Feng, D., Qin, Y., Shao, J., Xi, L., Chu, X.: ExBLACR: extending BLACR system. In: Susilo, W., Mu, Y. (eds.) ACISP 2014. LNCS, vol. 8544, pp. 397–412. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08344-5_26
28. Wood, G.: Ethereum yellow paper (2014)
29. Xi, L., Feng, D.: FARB: fast anonymous reputation-based blacklisting without TTPs. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society, pp. 139–148. ACM (2014)
30. Yang, R., Au, M.H., Xu, Q., Yu, Z.: Decentralized blacklistable anonymous credentials with reputation. IACR Cryptology ePrint Archive, vol. 2017, p. 389 (2017)
31. Yu, K.Y., Yuen, T.H., Chow, S.S.M., Yiu, S.M., Hui, L.C.K.: PE(AR)^2: privacy-enhanced anonymous authentication with reputation and revocation. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 679–696. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33167-1_39