PERM: Practical Reputation-Based Blacklisting without TTPs

Man Ho Au
School of Computer Science and Software Engineering
University of Wollongong
Wollongong, NSW, Australia
aau@uow.edu.au

Apu Kapadia
School of Informatics and Computing
Indiana University
Bloomington, IN, USA
kapadia@indiana.edu

ABSTRACT

Some users may misbehave under the cover of anonymity by, e.g., defacing webpages on Wikipedia or posting vulgar comments on YouTube. To prevent such abuse, a few anonymous credential schemes have been proposed that revoke access for misbehaving users while maintaining their anonymity such that no trusted third party (TTP) is involved in the revocation process. Recently we proposed BLACR, a TTP-free scheme that supports 'reputation-based blacklisting' — the service provider can score users' anonymous sessions (e.g., good vs. inappropriate comments) and users with insufficient reputation are denied access.

The major drawback of BLACR is the linear computational overhead in the size of the reputation list, which allows it to support reputation for only a few thousand user sessions in practical settings. We propose PERM, a revocation-window-based scheme (misbehaviors must be caught within a window of time), which makes computation independent of the size of the reputation list. PERM thus supports *millions of user sessions* and makes reputation-based blacklisting practical for large-scale deployments.

Categories and Subject Descriptors

K.6.5 [**Operating Systems**]: Security and Protection—*Authentication*; E.3 [**Data Encryption**]: Public key cryptosystems

Keywords

accountable anonymity, anonymous blacklisting, revocation

1. INTRODUCTION

Anonymous access to services can be of great value in many circumstances. For example, journalists and activists can avoid censorship and persecution while posting content to Wikipedia and YouTube anonymously. Nevertheless, users can and do abuse their anonymity by defacing webpages and posting inappropriate material. Repeated abuse has led service providers (SPs) like Wikipedia to ban access though anonymizing networks such as Tor [15].

Anonymous blacklisting and subjective judging. To enable a less drastic reaction than banning anonymous access, several credential schemes for *accountable anonymity* have been proposed recently. These schemes support the *subjective judging* of misbehaviors [19, 27], allowing SPs to arbitrarily flag behaviors as inappropriate. Subjective judging is useful in applications in which a mathematical or algorithmic formulation of misbehaviors such as 'inappropriate edits' is unlikely.[1] It has been recognized that since the subjective judging of users' behaviors is arbitrary, it is desirable for such schemes to support *anonymous blacklisting* [19, 27] such that users can be blocked from returning while maintaining their anonymity.[2] Thus users are held accountable, but they are not worried about arbitrary, subjective deanonymization.

TTP vs. TTP-Free schemes. Several approaches to providing anonymous blacklisting with subjective judging include some kind of trusted third party (TTP). Group signature-based schemes feature a *group manager* who can revoke access for users [1, 8, 13, 20]. 'Nymble' schemes make authentication at the SP efficient, but they also feature some kind of TTP [19, 27, 21, 18]. Since users must still rely on the judgment of the TTP, users can never be certain of their anonymity.

Thus, several TTP-free schemes have been proposed recently to eliminate this point of trust. BLAC was the first such scheme [24, 26]. In BLAC users must prove in zero knowledge that each entry on the blacklist does not correspond to an authentication made earlier using their credential, resulting in authentication times linear in the size of the blacklist. PEREA removed this linear dependence on the size of the blacklist by requiring misbehaviors to be 'caught', i.e., identified, within a revocation window of the past K authentications [25, 4]. Authentication times are now linear in the size of K, and thus K cannot be too large; typically $K = 10$ provides much better performance than BLAC. When combined with rate limiting, PEREA would

[1]In contrast, schemes supporting digital cash can easily characterize misbehavior such as the "double spending" of a coin.

[2]In contrast many existing schemes for subjective judging deanonymize or reduce the privacy of users.

allow enough (i.e., K) authentications per day and be able to block users who were blacklisted within a day of their misbehavior. We refer to the rate limiting time period as the "revocation period". Under this revocation-window model FAUST significantly improves authentication times using a novel whitelisting approach [22].

TTP-free reputation-based blacklisting. While previous schemes revoke access to users who are on the blacklist, the most recent advance was made by BLACR, i.e., BLAC with Reputation [2]. Hereafter we shall use the terminology "reputation list" instead of "blacklist" in order to reflect the list's nature in reputation-based blacklisting systems. In BLACR the SP can score each entry with a positive or negative score on the reputation list along with a category identifier. For example, the SP can use different scoring schemes in the categories of *comments* (e.g., foul language: -2, racist comments: -10, and helpful comments: $+5$) and *content* (e.g., popular videos: $+5$, copyright violations: -2 or -5 depending on how egregious the violations are). SPs can then require an authenticating user's reputation (total of scores in a category) to be above a particular threshold. Furthermore, SPs can specify policies that are boolean combinations of such category-threshold statements (e.g., reputation in the *comments* category should be above -5 AND reputation in the *content* category should be above -15).[3] BLACR guarantees that users who do not satisfy the policy are denied access without revealing any other information beyond whether the policy was satisfied or not.

While BLACR offers a good solution for *reputation-based blacklisting*, it is slow because it has the same linear dependence of BLAC. BLACR improves performance by implementing the concept of an 'express lane token', which amounts to incremental proofs over the previous authentication. Nevertheless, BLACR supports reputation lists with only a few thousand entries. While useful, BLACR's scalability is limited, and a more efficient solution is needed for SPs like Wikipedia or YouTube that can have *millions* of sessions where anonymous users edit or upload content.

The recent PEREA scheme does offer a form of reputation-based blacklisting (albeit for only a single category) that is efficient at the SP, but it suffers from two major drawbacks. First, the reputation of an authenticating user is calculated over the current revocation window, e.g., a day. Thus, *the user's reputation in only the last day would be taken into account*, which severely limits the scheme's applicability to systems in which reputation is built over much longer time periods than a few days. Second, the time at the user is linear in the size of the reputation list and it would take hours of computation at the user to generate authentication proofs. FAUST alleviates the computation problem [22], but offers an even more limited form of reputation than PEREA. As pointed out earlier [2, §1], FAUST cannot support boolean combinations of reputation policies across categories and is not collusion resistant to attacks in which users can pool their credentials to gain unauthorized access. A scheme that can fix these two problems of functionality and performance would offer a major improvement in reputation-based blacklisting by offering the first scalable alternative to BLACR. We propose such a scheme, which we call "PERM".

Our contributions.

- **Persistence of reputation scores.** We propose a 'reconstruction' of PEREA called "PERM". Our first contribution is to add 'memory' to PEREA, hence the name "PERM" (viz. PEREA with Memory). As long as a connection is scored within its revocation window, the score persists over time. For example, if a user's anonymous comment a year ago was scored within a day of making that comment, then that score will always factor into the user's reputation.
- **Free reputation upgrades.** To ensure that users *necessarily* acquire negative reputation for misbehaviors, PEREA-like schemes require that such behaviors be scored within a revocation window. But the concept of a memory in PERM also allows for *voluntary* updates to reputation outside of this window. While users have no incentive to voluntarily reduce their reputation, they will certainly be motivated to apply higher reputation scores if the score of one of their sessions is later revised. Consider a video upload that receives high user ratings after several months. At this point a user may want to acquire a higher reputation for that video. PERM provides such a voluntary reputation upgrade mechanism through a memory update protocol and thus positive scores can be applied at any time.
- **Multiple scores per session.** In PERM a single session can be scored under multiple categories. For example, a webpage edit may contain both a copyright violation as well as objectionable content. BLACR, however, requires the SP to pick a single category when scoring a session.
- **Much more efficient at the user.** Authentications in BLACR and PEREA are slow at the user and require computation linear in the size of the blacklist. PERM requires only a constant amount computation, regardless of the blacklist size, and is thus much more efficient at the user end. What would take PEREA half an hour of computation at the user (over 200,000 reputation list entries) takes PERM only 0.3 seconds (Section 4.2).
- **Efficient at the server.** The SP's computational overhead is still $O(K)$, as it is for PEREA, and is independent of the reputation list size. The constant factor is a little higher than for PEREA, but remains much more efficient than BLACR.

Paper outline. In Section 2 we provide an overview of the system goals and our approach with PERM. We provide the full cryptographic construction of PERM in Section 3 followed by a quantitative analysis in Section 4. We discuss various issues in Section 5 and conclude in Section 6. The Appendix provides the security model and sketches the proof of security for PERM.

2. OVERVIEW

Before we provide the details of our construction, we will give an overview of our approach. *Service Providers (SPs)* issue credentials to *users*, who can then authenticate anonymously to the SP. SPs record *transaction identifiers* with each *authenticated session*. Later, if the SP wants to reward or penalize users for their behavior in that session, then

[3]We used examples with negative thresholds to indicate some leeway with misbehavior. Indeed reputation-based blacklisting schemes are useful when varying degrees of misbehavior may be tolerated, but "up to a point."

the SP uses the transaction identifier to record a reputation *score* for that transaction under a particular *category* (or categories). These scores are recorded in a *reputation list*. SPs can then specify *authentication policies*, which specify boolean combinations of reputation thresholds across various categories. Users authenticate by proving they satisfy the policy with respect to the reputation list.

2.1 Authentication policies

Similar to BLACR [2], policies are of the form $\bigvee_{l=1}^{\ell}(\bigwedge_{j=1}^{J} \mathcal{P}_{lj})$, i.e., a disjunction of conjunctive clauses. Each conjunctive clause can include J 'sub policies' $\mathcal{P}_{lj}$ that correspond to categories. $\mathcal{P}_{lj}$ is of the form (ζ_{lj}, η_{lj}), which states that the authenticating user's reputation in category j must be within the range of $[\zeta_{lj}, \eta_{lj}]$. Not all J categories need to be covered in each clause, and $\mathcal{P}_{lj}$ is set to $\perp$ if a sub policy for that category is skipped. Each transaction identifier is associated with J scores, representing the scores of that authentication with respect to each category. The SP maintains a reputation list $\mathcal{L}$ of all transaction identifiers and their corresponding scores. $\mathcal{P}_{lj}$ evaluates to 1 if the reputation of the user based on this reputation list is within the range $[\zeta_{lj}, \eta_{lj}]$. We assume there exists an integer R_{max}, which is the maximum value of reputation for any user in the system. In our analysis we assume $R_{max} = 1024$. A subpolicy of $(-10, R_{max})$ requires the user to have a reputation higher than the threshold of -10.

Scores assigned within the revocation window will necessarily be factored into the reputation calculation of an authenticating user. Reputation scores can be credited to users beyond the revocation window through a voluntary 'reputation upgrade' protocol.

2.2 Security goals

We present informal definitions of the desired security properties. They are similar to previous schemes in this space (BLAC, PEREA, and BLACR); formalized versions are in the Appendix.

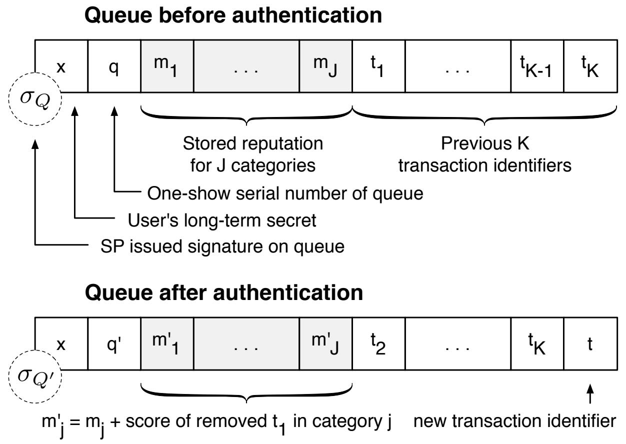
Authenticity: In a system with *authenticity*, SPs accept authentications only from users who satisfy the authentication policy. This property also implies *collusion resistance*, i.e., a group of revoked users should not be able to collude and somehow authenticate.

Anonymity: In a system with *anonymity*, all that SPs can infer about the identity of an authenticating user is whether the user satisfies the authentication policy at the time of protocol execution, regardless of future actions by the SP.

Non-frameability: A user is framed if she satisfies the authentication policy, but is unable to successfully authenticate to an honest SP. In a system with *non-frameability*, users satisfying the authentication policy can always successfully authenticate to honest SPs.

2.3 PERM: Conceptual approach

We sketch the central ideas of our PERM scheme with the aid of Figure 1. Like in PEREA, the main idea behind our construction is for the user to maintain a *queue* of its K most recent *transaction identifiers* $t_1, \ldots, t_K$. These transaction identifiers are assigned sequentially by the SP. The user will use this queue to prove in zero knowledge that his/her reputation scores with respect to those transaction identifiers, and the reputation list satisfy the authentication policy. The authentication time at the server is linear in



Queue before authentication
x
q
m1
...
mJ
t1
...
tK-1
tK
σQ
Stored reputation for J categories
Previous K transaction identifiers
One-show serial number of queue
User's long-term secret
SP issued signature on queue
Queue after authentication
x
q'
m'1
...
m'J
t2
...
tK
t
σQ'
m'j = mj + score of removed t1 in category j
new transaction identifier

Figure 1: Structure of a user's queue before and after authentication. During authentication the user 1) sends serial number q to the SP, which checks the freshness of the queue by verifying q has not been reused, 2) proves the integrity of the queue using σ_Q, 3) proves the user satisfies the authentication policy using stored values of each m_j and scores corresponding to $t_1, \ldots, t_K$ in category j, 4) obtains a new signature $\sigma_{Q'}$ for the updated queue, where s' is a new random number generated by the user.

K, and so $K = 10$ to keep authentication practical. Similar to PEREA, after each authentication the oldest transaction identifier t_1 is dropped from the queue, and the new transaction identifier t is added to the queue, adjacent to the most recent identifier t_K.

Adding memory. Our first 'trick' is to keep state of reputation scores associated with dropped identifiers; the queue stores the user's cumulative reputation for each of the J categories as $m_1, \ldots, m_J$. The user updates the stored reputation scores based on the scores on the reputation list associated with the dropped identifier t_1 for each category. These stored reputation values are combined with the reputation values corresponding to identifiers in the queue during authentication. The challenge is: 1) to ensure all these operations take place in zero knowledge, i.e., the SP should not learn the previous transaction IDs of an authenticating user and the reputation scores must be updated without revealing the scores to the SP; and 2) the user should be able to prove the integrity of the queue, i.e., the reputation scores were constructed correctly.

Improving efficiency. Our second trick to improve over PEREA involves eliminating its use of accumulators [7].[4] In contrast with PEREA's random transaction identifiers chosen by the user, transaction identifiers are now assigned

[4]An accumulator [7], allows the representation of a set of elements by a constant value. Elements may be added to (i.e., accumulated), or removed from, the accumulator. For each accumulated element one can compute a short witness that demonstrates this fact (and, in some cases, in zero knowledge). Thus, an accumulator is an efficient cryptographic construct for proving set membership, e.g., proving whether your transaction identifier is on a blacklist (or whitelist) without revealing the identifier.

serially by the SP, and if the SP judges the misbehaviors sequentially, the SP can maintain a single value, which we call the "judgment pointer", indicating the transaction identifier of the last judged authentication. Transaction identifiers larger than the pointer are 'unjudged'. Note that this requirement of judging transitions serially is reasonable for revocation-window-based schemes because regardless of what sessions were judged, the judgment pointer must advance as the revocation window expires for a transaction identifier or a set of identifiers. For example, if the revocation period is one day, the SP can advance the judgment pointer one day at a time. We refer to the maximum number of unjudged transactions at any given time as the "judgment window" N. N will depend on how many anonymous transactions typically occur within a revocation window and can be set high enough by the server so that it is unlikely for more transactions to occur within any revocation window. N affects the size of the public parameter, thus N should represent a pragmatic choice (20,000 is reasonable as discussed in Section 4.1).

For the judged transactions, the SP publishes signatures that bind each score to the corresponding transaction identifier. These signatures are commonly known as "CL-signatures" [10], and come with two efficient protocols. The first one is an efficient protocol for a user to obtain a signature on a committed message (we discuss commitments in Section 2.4) without revealing the message, while the second protocol allows a user to demonstrate in zero knowledge the possession of a signature on a committed message. These signatures can later be used by authenticating users to prove their reputation in zero knowledge. During authentication, for each identifier in the user's queue, the user proves that either the identifier is greater than the judgment pointer or, using the appropriate signature, that the identifier is associated with a particular score and that score is factored into the (more complex) reputation calculation. This entire authentication takes place in zero knowledge, so the SP learns only whether the policy was satisfied or not and learns no other information about the individual scores for example. Next we describe each step at a high level.

2.4 PERM: Construction overview

We present a high-level description of PERM with a single category. This section is intended to give the reader an idea of how we achieve constant computational cost while abstracting the construction details.

Building blocks. Our construction makes use of 'commitments' and 'interval proofs', which we now describe.

Commitments: $\mathbb{C}(x)$ denotes a 'commitment' of x. Given $\mathbb{C}(x)$, it is computationally infeasible for the SP to determine x, but the user can prove various properties about x without revealing x. We make use of a commitment scheme that is homomorphic. That is, given $\mathbb{C}(x)$, $\mathbb{C}(y)$, their product $\mathbb{C}(x)\mathbb{C}(y)$ is a commitment of the value $x + y$. An example of such a commitment scheme is provided by Pedersen [23].

Interval proofs: Given a commitment $\mathbb{C}(x)$, there exists an efficient interval proof that proves in zero knowledge the value of x is within a given interval $\{1, \ldots, N\}$. An example of such a construction is provided by Camenisch et al. [9].

SP Setup. The SP initializes various public parameters, including the revocation window K and the judgment window N. The SP maintains a list $\mathcal{L}$, which is empty initially and two counters tc and jp. tc is the current transaction identifier and jp is the judgment pointer (the identifier of the last-judged behavior).

User Registration. The user randomly picks two values $x, q \in_R \mathbb{Z}_p$ and prepares a queue of size $K + 3$ as $\mathsf{Q} = (x, q, 0, 0, \ldots, 0)$. x is the user's long-term secret, and q is a value unique to the queue. The next value represents the current reputation in the memory while the last K entries store the past K transaction identifiers. Upon completion of the registration protocol, the user obtains a signature σ_{Q} on the queue Q from the SP. The first diagram in Figure 1 illustrates this queue before authentication for the general case with J categories.

Authentication. Given a queue $\mathsf{Q} = (x, q, m, t_1, \ldots, t_K)$ and a signature σ_{Q}, the user obtains the current list $\mathcal{L}$. $\mathcal{L}$ is a list of the following form: $(i, s_i, \sigma_{i,s_i})_{i=1}^{\mathsf{jp}}$. i is the transaction identifier, s_i is the score associated with that authentication, and σ_{i,s_i} is the SP's signature on the tuple (i, s_i). We assume the public parameters include $(0, 0, \sigma_{0,0})$, which certifies that the score of the initial values of a queue is zero.

Part I (Queue Validation): Commit every entry of Q individually as:

$$\mathbb{C}(x), \mathbb{C}(q), \mathbb{C}(m), \mathbb{C}(t_1), \ldots, \mathbb{C}(t_K).$$

The user then proves that he/she has a signature σ_{Q} on the tuple $(x, q, m, t_1, \ldots, t_K)$ (thus proving the integrity of the committed queue). The user also commits the score related to each of his past K transaction identifiers:

$$\mathbb{C}(s_{t_1}), \mathbb{C}(s_{t_2}), \ldots, \mathbb{C}(s_{t_K}).$$

For each value a for $a = 1$ to K, the user proves to the SP one of the following is true:

- *Transaction with identifier t_a is unjudged:* $\mathbb{C}(s_{t_a})$ is a commitment of score 0 and $\mathbb{C}(t_a)\mathbb{C}(-\mathsf{jp})$ is a commitment of a value within the range 1 to N. In other words, $1 \leq t_a - \mathsf{jp} \leq N$;
- *Transaction with identifier t_a is judged with score s_{t_a}:* the user has a signature $\sigma_{t_a,s_{t_a}}$ on a tuple (t_a, s_{t_a}), and $\mathbb{C}(t_a)$ and $\mathbb{C}(s_{t_a})$ are commitments of t_a and s_{t_a}.

The proof that $t_a - \mathsf{jp}$ is within the range 1 to N is done via the interval proof discussed before. Nonetheless, the whole proof can be computed in time linear to K. This proof assures the SP that the score is committed in each $\mathbb{C}(s_{t_a})$ properly. The user further reveals the serial number q to show that he has not used this queue before (thus proving the freshness of the queue). This prevents the user from repeatedly using the same queue.

Part II (Subpolicy Satisfaction): Using the homomorphic property of the commitment scheme, the user then proves to the SP that the value $\mathbb{C}(R) = \mathbb{C}(m)\mathbb{C}(s_{t_1}) \cdots \mathbb{C}(s_{t_1})$ is a commitment of a value R and that R is above the reputation threshold specified in the policy. For multiple categories this approach is used for each individual category, and the boolean policy across categories can be proved in zero knowledge as described in Section 3.3.

Part III (Queue Update): The user sends a commitment $\mathbb{C}(q')$, which is a commitment of a random number q' that represents the new serial number for the queue. Using all the commitments, the SP issues a new signature $\sigma_{\mathsf{Q}'}$ on a new queue $\mathsf{Q}' = (x, q', m + s_{t_1}, t_2, \ldots, t_K, t')$, where $t' = \mathsf{tc}$

is the current transaction identifier. After that, the SP increases the value of tc by 1. The second diagram in Figure 1 illustrates this queue after the update for the general case with J categories. For the purpose of a possible score update in the future for transaction t_1, which was moved to the memory, the SP also issues a signature σ_{x,t_1} on the tuple (x, t_1). The signature σ_{x,t_1} is a 'receipt' for transaction t_1.

Scoring an entry. The SP judges entry $\mathsf{jp}+1$ and gives it a score $s_{\mathsf{jp}+1}$. It then creates a signature $\sigma_{\mathsf{jp}+1,s_{\mathsf{jp}+1}}$ and puts the tuple $(\mathsf{jp}+1, s_{\mathsf{jp}+1}, \sigma_{\mathsf{jp}+1,s_{\mathsf{jp}+1}})$ on the reputation list $\mathcal{L}$. Note that $s_{\mathsf{jp}+1}$ can be positive or negative. After putting $(\mathsf{jp}+1, s_{\mathsf{jp}+1}, \sigma_{\mathsf{jp}+1,s_{\mathsf{jp}+1}})$ on the list, the value of jp is increased by 1. These scores will be reflected in the user's reputation only if the transaction was judged within the revocation window for that transaction.

Upgrading a score. Suppose later (outside of the revocation window) the score of an entry t on the reputation list $\mathcal{L}$ is changed from s_t to s'_t. Let d be the difference. A user with queue $\mathsf{Q} = (x, q, m, t_1, \ldots, t_K)$ and a receipt $\sigma_{x,t}$ approaches the SP, reveals the value q and t, and demonstrates that he has a signature on σ_{Q} and $\sigma_{x,t}$. At the end of the protocol, he receives a new signature $\sigma_{\mathsf{Q}'}$ with the updated score in his memory as $\mathsf{Q}' = (x, q', m+d, t_1, \ldots, t_K)$. The SP marks the entry t as having been upgraded to prevent attacks where users try to apply the same credit multiple times. If the SP decides to further change the score in the future, the user can again apply the new balance, and as before this balance can be credited only once.

3. PERM CONSTRUCTION

In our construction we require digital schemes that support various zero-knowledge, proof-of-knowledge protocols. For clarity we employ the pairing-based variant (BBS+) [3] of the well-known CL signature [10, 11]. It should be noted that specific signature schemes can be employed in different protocols of our system for efficiency considerations.

3.1 Parameters

Let λ be a sufficiently large security parameter. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing such that $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$ for a λ-bit prime p. Let $g_0, g_1, g_2 \in \mathbb{G}_1$ (resp. $h_0 \in \mathbb{G}_2$) be generators of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) such that the relative discrete logarithm of the generators is unknown.[5] Let $H : \{0,1\}^* \to \mathbb{Z}_p$ be a collision-resistant hash function. Throughout this section these parameters will be available to all parties.

3.2 Building blocks

3.2.1 Zero-Knowledge Proofs of knowledge

In a *Zero-Knowledge Proof of Knowledge (ZKPoK)* protocol [16], a prover convinces a verifier that some statement is true while the verifier learns nothing except the validity of the statement. Σ-protocols are a type of ZKPoK protocol, which can be converted into non-interactive *Signature Proof of Knowledge (SPK)* schemes, or simply signature schemes [17], that are secure under the *Random Oracle (RO)* Model [6]. We follow the notation introduced by Camenisch and Stadler [12]. For instance, $\mathsf{PK}\{(x) : y = g^x\}$ denotes a ZKPoK protocol in which a prover proves the knowledge of an integer x such that the relationship $y = g^x$ holds. Symbols appearing on the left of the colon denote values whose knowledge is being proved while symbols appearing on the right, but not the left, of the colon denote public values. We use $\mathsf{SPK}\{(x) : y = g^x\}(M)$ to denote the corresponding signature proof of knowledge.

[5]This can be done by setting the generators to the output of a cryptographic hash function of some publicly known seeds.

3.2.2 Commitment scheme

Our construction uses the well-known, non-interactive commitment scheme due to Pedersen [23], which is briefly reviewed below. Let $\mathbb{G}$ be a cyclic group of prime order p and g, h be independent generators of $\mathbb{G}$. On input of value $x \in \mathbb{Z}_p$, the committer randomly chooses $r \in \mathbb{Z}_p$, computes and outputs $C = g^x h^r$ as a commitment of value x. To reveal the value committed in C, the committer outputs (x, r). Everyone can test if $C = g^x h^r$. C is a commitment of a value x with opening r.

Pedersen Commitment is perfect hiding and computationally binding. That is, even a computationally unbounded receiver cannot learn anything about the value committed from the commitment. On the other hand, a probabilistic polynomial time (PPT) sender can only reveal the commitment with one value under the discrete log assumption.

We use "$\mathbb{C}(x)$" to denote a Pedersen Commitment of a value x. Note that Pedersen Commitment is homomorphic in the sense that on input $\mathbb{C}(a)$ and $\mathbb{C}(b)$, $\mathbb{C}(a) * \mathbb{C}(b)$ gives a commitment of $a + b$.

3.2.3 Credential signature scheme

We briefly review the signature scheme proposed by Au et al. [3], which is called BBS+. Their scheme is based on the schemes of Camenisch and Lysyanskaya [11] and of Boneh et al. [8]. BBS+ is employed to certify queues as well as binding scores to transaction identifiers in PERM. Let $g, g_0, g_1, g_2, \ldots, g_k \in \mathbb{G}_1$ be generators of $\mathbb{G}_1$ and h be a generator of $\mathbb{G}_2$. The signer's secret is a value $\gamma \in \mathbb{Z}_p$ and the public key is $w = h^\gamma$. BBS+ supports a block of messages as follows. To sign a message block $(x_0, x_1, \ldots, x_k) \in \mathbb{Z}_p^{k+1}$, the signer randomly picks $e, y \in_R \mathbb{Z}_p$, and computes $A = (g g_0^{x_0} g_1^{x_1} \cdots g_k^{x_k} g_{k+1}^{y})^{\frac{1}{\gamma+e}}$. The signer outputs (A, e, y) as the signature on the block of messages $(x_0, \ldots, x_k)$. To verify a BBS+ signature, one can test if the following equation holds:

$$\hat{e}(A, wh^e) = \hat{e}(g g_0^{x_0} g_1^{x_1} \cdots g_{k+1}^{y}, h).$$

They also derive the two useful protocols below.

Protocol $\mathfrak{S}_{\mathsf{Iss}}$. $\mathfrak{S}_{\mathsf{Iss}}$ allows a user to obtain a credential signature from the signer on a block of values $(x_0, \ldots, x_k)$ committed in $C_0, \ldots, C_m$. Let the user's additional input be $x_0, r_0, \ldots, x_k, r_k$ such that $C_i = g_1^{x_i} g_2^{r_i}$ for $i = 0$ to k.

1. The user computes $C_M = g_0^{x_0} g_1^{x_1} \cdots g_k^{x_k} g_{k+1}^{y'}$ for some randomly generated $y' \in_R \mathbb{Z}_p$, sends C_M to the signer along with the following proof:

$$\mathsf{PK}\left\{\begin{array}{l} (\{x_i, r_i\}, y') : \\ C_M = g_0^{x_0} g_1^{x_1} \cdots g_k^{x_k} g_{k+1}^{y'} \bigwedge_{i=0}^{k} (C_i = g_1^{x_i} g_2^{r_i}) \end{array}\right\}$$

2. The signer returns 0 if verification of the proof fails. Otherwise the signer randomly generates $e, y'' \in_R \mathbb{Z}_p$,

computes $A = (gC_M g_{k+1}^{y''})^{\frac{1}{e+\gamma}}$, and returns (A, e, y'') to the user.

3. The user computes $y = y' + y''$. She returns 0 if $\hat{e}(A, wh^e) \neq \hat{e}(gg_0^{x_0} g_1^{x_1} \cdots g_{k+1}^{y}, h)$. Otherwise the user outputs σ as (A, e, y).

Protocol $\mathfrak{S}_{\mathsf{Sig}}$. In protocol $\mathfrak{S}_{\mathsf{Sig}}$ a prover convinces a verifier he/she knows a signature $\sigma = (A, e, s)$ on a block of messages $(x_0, \ldots, x_k)$ committed in $C_0, \ldots, C_k$. Additionally, the prover knows the values and openings of the commitments $x_0, r_0, \ldots, x_k, r_k$ such that $C_i = g_1^{x_i} g_2^{r_i}$ for $i = 0$ to k.

1. Let M be the random challenge. The prover randomly generates $k_1, k_2 \in_R \mathbb{Z}_p$, computes $A_1 = g_1^{k_1} g_2^{k_2}$, $A_2 = A g_2^{k_1}$, and the following SPK Π.

$$\mathsf{SPK}\left\{\begin{array}{c} (\{x_i, r_i\}, e, y, k_1, k_2, \beta_1, \beta_2) : \\ \bigwedge_{i=0}^{k} (C_i = g_1^{x_i} g_2^{r_i}) \wedge \\ A_1 = g_1^{k_1} g_2^{k_2} \wedge \\ 1 = A_1^{-e} g_1^{\beta_1} g_2^{\beta_2} \wedge \\ \frac{\hat{e}(A_2, w)}{\hat{e}(g, h)} = \hat{e}(g_0, h)^{x_0} \cdots \hat{e}(g_k, h)^{x_k} \\ \hat{e}(g_{k+1}, h)^{y} \hat{e}(g_2, w)^{k_1} \\ \hat{e}(g_2, h)^{\beta_1} / \hat{e}(A_2, h)^{e} \end{array}\right\}(M')$$

where $M' = M || A_1 || A_2$ and $\beta_1 = k_1 e$, $\beta_2 = k_2 e$.

2. The prover outputs $\mathsf{p}_{\mathsf{Sig}}$ as (Π, A_1, A_2).

3. Upon receiving $(\mathsf{p}_{\mathsf{Sig}}, M)$, the verifier parses $\mathsf{p}_{\mathsf{Sig}}$ as (Π, A_1, A_2) and outputs `accept` if Π is a valid proof.

3.2.4 *Signature-based interval proof*

To demonstrate that the reputation of a user is within a certain range, we employ the signature-based interval proof due to Camenisch et al. [9]. In a nutshell the verifier provides a set of 'digital signatures' on the elements of the required range under a verification key. We consider this set of digital signatures to be the public parameter. In order for the prover to demonstrate that a certain value committed in a commitment is within the range, the prover proves in zero-knowledge that he/she knows a signature under the verification key for the element committed. This proof is of constant size and is useful when the range is small.

3.3 Actual construction

SP Setup. The SP selects the revocation window K, judgment window N, and the total number of categories J. The SP further creates two instances of the BBS+ signatures (PK_1, SK_1) and (PK_2, SK_2). The first BBS+ signature instance supports a message block of size $K + J + 2$ and the second instance supports a message block of size $J + 2$.

The public parameter is (PK_1, PK_2, K, N, J) together with the judgment pointer jp and the current transaction counter tc which are both initialized to zero.

The secret key is (SK_1, SK_2).

Registration. The user randomly picks two values $x', q \in_R \mathbb{Z}_p$ and prepares a queue of size $K + J + 2$ as $\mathsf{Q} = (x', q, 0, \ldots, 0)$.

The user computes $\mathbb{C}(x')$, $\mathbb{C}(q)$ and sends them to the SP. The SP randomly picks another random number $x'' \in_R \mathbb{Z}_p$ and computes $\mathbb{C}(x' + x'')$. The value x'' is sent to the user. The user computes $x = x' + x''$ and sets the first element of his queue Q to x. The value x is used as his long-term secret key. Note that this is possible due to the homomorphic property of the commitment scheme.

The user and SP engage in the protocol $\mathfrak{S}_{\mathsf{Iss}}$ of the BBS+ signature scheme with public key PK_1. Upon successful completion of the protocol, the user obtains a signature σ_{Q} on his queues Q. The user stores $(\sigma_{\mathsf{Q}}, x, q)$.

Authentication. The user is in possession of σ_{Q} on his queue $\mathsf{Q} = (x, q, m_1, \ldots, m_J, t_1, \ldots, t_K)$. The user downloads the latest list from the SP as well as the authentication policy $\mathcal{P} = \bigvee_{l=1}^{\ell} (\bigwedge_{j=1}^{J} \mathcal{P}_{lj})$. Each $\mathcal{P}_{lj}$ is of the form (ζ_{lj}, η_{lj}) and requires the user's reputation in category j to be within the range (ζ_{lj}, η_{lj}). The list $\mathcal{L}$ is a list of jp entries $(i, s_{1,i}, \ldots, s_{J,i}, \sigma_i)_{i=1}^{\mathsf{jp}}$, where σ_i is a BBS+ signature on the tuple $(i, s_{1,i}, \ldots, s_{J,i})$ under PK_2. $s_{j,i}$ is the score of transaction with identifier i in category j.

The user computes his reputation in category j as $R_j = m_j + \sum_{a=1, t_a \leq \mathsf{jp}}^{K} s_{j,t_a}$ and checks that there exists an index $l \in \{1, \ldots, \ell\}$ such that $\zeta_{lj} \leq R_j \leq \eta_{lj}$. If yes, the user satisfies the policy and computes the following commitments.

$$\mathbb{C}(x), \mathbb{C}(m_1), \ldots, \mathbb{C}(m_J), \mathbb{C}(t_1), \ldots, \mathbb{C}(t_K)$$

Next, the user send q to the SP and employs $\mathfrak{S}_{\mathsf{Sig}}$ to show that he has a signature σ_{Q}. This assures the SP the set of commitments is computed correctly. The SP also checks the freshness of q and stores it in the database.

Next the user sends another set of commitments $\{\mathbb{C}(s_{1,t_a}), \ldots, \mathbb{C}(s_{J,t_a})\}_{a=1}^{K}$ to the SP, and proves that one of the following two statements is true for $a = 1$ to K:

1. $1 \leq (t_a - \mathsf{jp}) \leq N$ and $s_{j,t_a} = 0$ for $j = 1$ to J;

2. The user has a signature σ_i on message block $(t_a, s_{1,t_a}, \ldots, s_{J,t_a})$ and $s_{j,t_a} = s_{j,i}$ for all $j = 1$ to J.

The above proof assures the SP that $\mathbb{C}(s_{j,t_a})$ is the score of the user's authentication t_a in category j.

Next, both parties compute locally $\mathbb{C}(R_j) = \mathbb{C}(m_j) \prod_{a=1}^{K} \mathbb{C}(s_{j,t_a})$. The value $\mathbb{C}(R_j)$ is a commitment of the user's reputation in category j. In order to show that the user satisfies the boolean policy, the user proves in zero-knowledge that the set $\{R_j\}_{j=1}^{J}$ satisfies one sub-clause of $(\bigwedge_{j=1}^{J} \mathcal{P}_{lj})$ of the policy $\mathcal{P}$, which states that: $\exists l \in \{1, \ldots \ell\} : \zeta_{lj} \leq R_j \leq \eta_{lj}$ for $i = 1$ to J. Proving in zero-knowledge that $\zeta_{lj} \leq R_j \leq \eta_{lj}$ is accomplished by the interval proof. We assume the scores are small so that the interval proof discussed in Section 3 can be employed. Further, since we are working in a cyclic group of prime order p, we assume $\sum_{i=0}^{\mathsf{tc}} s_i < p$ to prevent the sum of positive reputation scores from wrapping around and being interpreted as negative. Combining the zero-knowledge proofs in a 1-out-of-ℓ manner can be accomplished using the technique described by Cramer et al. [14]. Thus, the user can prove in zero-knowledge that the boolean policy is satisfied.

Finally, the user randomly generates $q' \in_R \mathbb{Z}_p$ and computes $\mathbb{C}(q')$. Using protocol $\mathfrak{S}_{\mathsf{Iss}}$, the SP issues two signatures to the user. The first one is $\sigma_{\mathsf{Q}'}$ which is a BBS+ signature on the queue $\mathsf{Q}' = (x, q', m_1 + s_{1,t_1}, \ldots, m_J + s_{J,t_1}, t_2, \ldots, t_K, t)$. Note that $t = \mathsf{tc}$ is the value of the current transaction counter.

The SP also issues another BBS+ signature σ_{x,t_1}, which is a signature on message (x, t_1). The user stored this should he want to use the free upgrade in the future. The SP increases the value of tc by 1.

Scoring a transaction. The SP adds the entry $(\mathsf{jp}, s_{1,\mathsf{jp}}, \ldots, s_{J,\mathsf{jp}}, \sigma_{\mathsf{jp}})$ after giving the appropriate score to authentication j. Then the SP increases jp by 1.

Upgrading a score. After some time, the SP could increase the scores for transaction identifier t. Let's say the original score is $s_{1,t}, \ldots, s_{J,t}$ and the updated score is $s'_{1,t}, \ldots, s'_{J,t}$. To conduct an upgrade, the user uses his current secret σ_{Q}, which is a signature on his current queue $\mathsf{Q} = (x, q, m_1, \ldots, m_J, t_1, \ldots, t_K)$ as well as a signature $\sigma_{x,t}$, which he obtains when the transaction identifier t shifts out from his queue.

Similar to an authentication, the user releases q and produces the following commitments:

$$\mathbb{C}(x), \mathbb{C}(m_1), \ldots, \mathbb{C}(m_J), \mathbb{C}(t_1), \ldots, \mathbb{C}(t_K).$$

The SP checks the freshness of the value q and stores it in its database.

The user proves to the SP in zero-knowledge that he has $\sigma_{x,t}$ and σ_{Q} using the protocol $\mathfrak{S}_{\mathsf{Sig}}$. If the SP accepts the proof, the user further randomly generates $q' \in_R \mathbb{Z}_p$. They then engage in the protocol $\mathfrak{S}_{\mathsf{Iss}}$. Upon termination of the protocol, the user obtains a signature $\sigma_{\mathsf{Q}'}$ on messages $(x, q', m_1 + d_1, \ldots, m_J + d_J, t_1, \ldots, t_K)$, where $d_j = s'_{j,t} - s_{j,t}$. The SP marks the update as complete and no score update request on this entry will be entertained.

3.4 Security of our construction

The security model for PERM, and the proof of security for our construction is sketched in the Appendix.

4. QUANTITATIVE ANALYSIS

We now analyze the communication and computational performance of PERM and show that PERM is the first practical and scalable scheme to support reputation-based blacklisting. Note that for PEREA we assume the 'PEREA with Naughtiness' extension [4], which adds some form of reputation (albeit only negative reputation and the score computed is only over the most recent revocation window).

4.1 Data transfer

Assume each score is of 5 bits and the maximum reputation of the user in any category is in a 10-bit range. Further, assume the total number of authentications is less than 2^{50} (an entirely reasonable assumption considering that the total number of edits made to all pages in Wikimedia is about 2^{31}.[6]). Following the parameters in BLACR [2], we have $\mathbb{Z}_p = 249$ bits and elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ are of size 320 and 944 bits. Each entry in the reputation list is of $868 + 5J$ bits, where J is the number of categories. Together with the random challenge, the total downlink complexity is $((868 + 5J)L + 249)$ bits, where L is the size of the reputation list, which is the value of jp in PERM. Further, assume we have 5 categories. Each entry in the list is less than 1Kb. Suppose the number of anonymous authentications per day is 20,000 (this would correspond to about 1 in 5 edits on Wikipedia being anonymous, which is what Au et al. assume [2, §5.2]), a user would need to download 2.2MB of data a day to keep its list up-to-date. The overhead of each authentication is 0.2KB, which corresponds to the receipt plus a signature on the new queue. Here we assume the policy is not changed frequently and is thus treated as part of the public parameters.

The size of the uploaded proof by the user (with $J = 5$) is $(2K + 8 + 10\ell)$\320 + $(78K + 5\ell + 27)$249 bits. Further, assume $\ell = 5$, $K = 10$, a proof is of constant size at 28KB. Note that this is a conservative estimate, since further optimizations are possible. For instance, one could replace the commitment of an individual transaction identifier with a commitment of all in a single group element. We leave such optimizations for future work, although, given that the size is only 28KB, these may not be necessary.

The public parameter size is linear to the judgment window. Again, assume the total number of authentications per day is 20,000 and that each authentication is to be judged within one day and the public parameter size will be around 2MB. Even if the judgment window is computed over a week, a one-time download of about 14MB is reasonable.

4.2 Computation

We now compare the performance of PERM at the SP and the user. We compare PERM with BLACR and PEREA, as those are the two closest schemes that support reputation-based blacklisting with strong security properties.

We have identified the major operations for each of the schemes as shown in Table 1. The symbols EN1 and EN2 represent the time cost of an exponentiation of a small exponent to a random base without pre-processing and a range exponent to a fixed base with pre-processing modulo a large RSA modulus, respectively. The symbols "E1", "E2", and "ET" represent the time cost of a multi-based exponentiation without pre-processing in the groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, respectively. Likewise, "EP1", "EP2", and "EPT" represent the time cost of an exponentiation with respect to a known single base with pre-processing in the groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, respectively. Finally, P represents the time cost of a bilinear pairing operation. Since these parameters are the same as those in BLACR, we reuse the benchmarks from BLACR [2, Table 3], which were obtained on a Lenovo X200s with an Intel Core 2 Duo CPU L9400 and 4GB RAM running Windows Vista as the host.

For PERM the amount of computation at the user is drastically reduced as compared to other schemes. In fact, nearly all exponentiation operations at the user side are precomputable. Since the score of each transaction identifier is from a limited range (e.g. 5 bits), it is possible to precompute commitments of all possible scores. At the same time, the reputation is also from a limited range (e.g. 10 bits) and thus the range proof is pre-computable. The first move of the three-move, zero-knowledge proof can always be pre-computed, and the last move of the proof only consists of additions and subtractions in $\mathbb{Z}_p$. The only step that cannot be pre-computed relates to the digital signatures of the entries on the list, where the commitments of such signatures

[6] http://toolserver.org/~emijrp/wikimediacounter/

Schemes	**Parties**	**Computation**
BLACR Normal	User (w/pre-computation)	$\big((3-\zeta)L\big)$E1 + 1EPT + 1P
	SP	$\big(12L+3\ell J+8J+5\big)$E1 + $4J$ EP1 + $\big(5L+5\ell J+5\big)$EPT +
		$\big(L+\ell J+1)$P + $\big(2L+2\ell J+2\big)$EP2
BLACR Express	User (w/pre-computation)	$\big((3-\zeta)L\big)$E1 + 1EPT + 1P
	SP	$\big(12\Delta_L+3\ell J+12J+7\big)$E1 + $4J$EP1 + $\big(5\Delta_L+5\ell J+4J+9\big)$EPT +
		$\big(\Delta_L+\ell J+2)$P + $\big(2\Delta_L+2\ell J+4\big)$EP2
PEREA (w/ Naughtiness)	User	$[(A+1)\Delta_L]$EN1 + $[16K+\lceil\frac{K-1}{3}\rceil+12]$EN2
	SP	$[15K+\lceil\frac{K}{3}\rceil+8]$EN2
PERM	User	$[15KJ+15K+3J+6\ell J+16]$E1 +
		$[2K\lceil\frac{J+6}{3}\rceil+4KJ+\lceil\frac{J+K+6}{3}\rceil+2\ell J]$ET + [2KJ + 2K + J]P
	User (w/pre-computation)	$[K]$ E1 + $(2K+2)$P
	SP	$[5KJ+6K+2J+4\ell J+16]$E1 +
		$[3KJ+3\ell J+\lceil\frac{J+K+7}{3}\rceil+K\lceil\frac{J+7}{3}\rceil]$ET + [2KJ + 2K + 2J + 2]P

Table 1: Complexity analysis for BLAC, PEREA, and PERM. J is the total number of categories ℓ is the number of sub-policies in the boolean policy, where each sub-policy is assumed to involve all J categories. For BLACR and PEREA Δ_L is the number of new entries on the reputation list since the previous time period. Specific to BLACR, ζ is the fraction of entries belonging to the user in the reputation list.

are needed during the zero-knowledge proof. Each authentication identifier is associated with one signature, and thus the user needs to compute K commitments online. Specifically, the value A_2 is not known off-line in protocol $\mathfrak{S}_{\mathsf{Sig}}$. Note that computing A_2 does not require any exponentiations (since the value $g_2^{k_1}$ does not depend on the value A and can be pre-computed). However, in the proof two values related to A_2 are required: $\hat{e}(A_2, g)$ and $\hat{e}(A_2, w)$. Another exponentiation to base A_2 is needed in the production of the 'OR' part of the proof.

Performance at the SP. Figure 2(a) shows the performance of authentication of PERM compared to the other schemes at the SP. Au et al. argue that a server throughput of 25 authentications/minute would be practical for SPs such as Wikipedia [2, §5.2], and we use that as our baseline. The amount of computation at the SP in both PEREA and PERM are independent of the reputation list size and correspond to the two horizontal lines. For PERM with $K = 10$, the SP can support 10 authentications/minute as compared to 23 for PEREA (PEREA is faster, but recall the severe limitations to its functionality, i.e., reputation is calculated over only the short revocation window). Since authentications are easily parallelized, additional servers can be used to bring authentication rates higher. With 2–3 servers (costing about \$5–7.5K/year on Amazon EC2) these rates are more than adequate for anonymous authentications on a large SP such as Wikipedia.

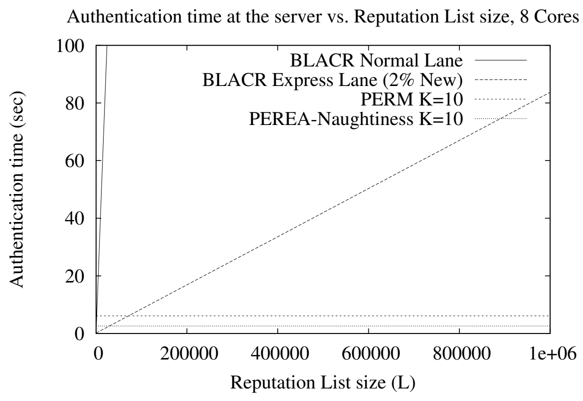
For BLACR the worst case performance is too slow for reputation lists with 1 million entries. Assuming three servers, BLACR would be able to support only 3 authentications every 1–2 *hours*. BLACR uses an 'express lane' technique to allow users who authenticated in the previous time period to prove their reputation incrementally from the last authentication. If we assume all users regularly authenticate in the express lane (with 2% new entries since the previous authentication), the SP can support 2 authentications per minute, which is still almost 10 times slower than PERM. Since the performance of BLACR in the normal lane is so slow, we do not attempt to fit it into the graph (in which case the curves for PEREA and PERM would not be visible). Now consider the case of Wikimedia where the running count of edits exceeds 1.5 billion.[7] In this case, PERM would be many thousands of times faster than BLACR for those users who authenticate in the normal lane.

[7] `http://toolserver.org/~emijrp/wikimediacounter/`

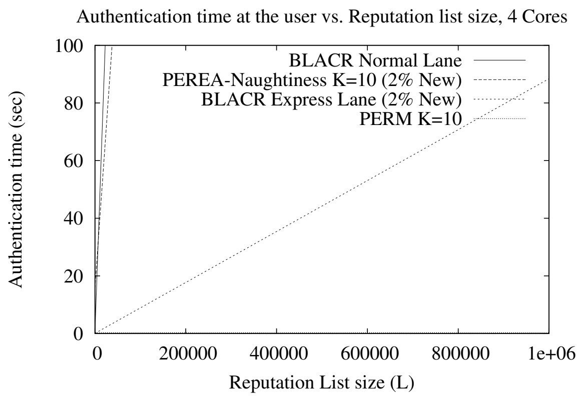
Performance at the user. Figure 2(b) shows the performance of authentication of PERM compared to other schemes at the user. PEREA's main drawback was the linear increase in performance in both K and the reputation list size at the user. Both BLACR in the normal lane and PEREA (assuming only 2% new entries since the last authentication) are unacceptably slow and we do not attempt to fit them within the graph. PEREA would take 38 minutes of computation, and BLACR would take about 74 minutes for a reputation list with 1 million entries. Assuming express lane authentication, BLACR would take 88 seconds for the user, which is much better (but recall the performance for BLACR is unacceptable at the SP at this scale). In contrast, PERM requires only 0.3 seconds of computation, and is close to the X-axis in the graph.

Thus, considering both the performance at the SP and the user for millions of entries on the reputation list, it is clear that only PERM offers a viable solution for reputation-based blacklisting. As one can note, even for tens of millions (or billions!) of entries, the performance at the SP and the user would remain unchanged.

A note on BLACR-Unweighted. Since we compare PERM to BLACR, we now describe our basis for this comparison. The authors of BLACR propose a weighted extension in which reputation scores can be increased for repeated misbehaviors by the same user, but we currently do not attempt to replicate this ramp-up functionality. Referring to their technical report [5], the computational complexity of BLACR-Unweighted at the user side is the same as that of BLACR-Weighted. At the SP, BLACR-Unweighted is roughly 3 to 4 times faster than BLACR-Weighted. While the express-lane authentication trick is not applied to BLACR-Unweighted, their analysis shows that it is also applicable. Since we are concerned with reputation list sizes of around a million, the overhead applicable to the express-lane trick becomes negligible, and the rate-determining step is the number of exponentiations and pairing operations per entry of the list. In BLACR-weighted the value is 19 exponentiations plus 1 pairing while the figure is 8 exponentiations for BLACR-unweighted. Since 1 pairing operation takes roughly the same time as 5 exponentiations, one can safely deduce that BLACR-unweighted is at most 4 times faster than BLACR-



(a) Authentication time at the SP. The horizontal curves correspond to PEREA and PERM. BLACR with Normal Lane is too slow and we do not attempt to fit it into the graph. Instead, one could assume BLACR with Express Lane is always used. As we can see, the authentication rates at the SP would be too low for BLACR with 1 million reputation list entries, whereas PERM supports practical authentication rates.



(b) Authentication time at User (includes precomputation). As we can see authentication rates for BLACR with Normal Lane and PEREA are unacceptably slow at the user. BLACR with Express Lane is somewhat acceptable, and in comparison the time taken by PERM is negligible at the user (and is close to the X-axis).

Figure 2: Estimated authentication times at the SP and User and the cost for the SP. For the SP we assume a server with 8 CPU cores, since such configurations are standard for servers. Likewise, we assume 4 CPU cores for the user, since such configurations are more standard for consumer desktops and laptops.

weighted with the express lane trick applied. We make this (conservative) assumption in our performance analysis.

5. DISCUSSION

Recently proposed alternative. FAUST [22] is the most recent scheme in the revocation window paradigm and aims to improve performance over PEREA. While FAUST is not designed specifically for reputation, it can simulate at least part of the functionality by making users maintain a collection of tokens. As individual tokens get revoked (for a misbehavior with score 5, for example, 5 tokens can be revoked), users may eventually not have the threshold number of tokens left to authenticate. The downside with this approach is that it is not collusion resistant and individual revoked users can pool their tokens together to gain access. Existing schemes such as BLACR, PEREA, and our PERM do not suffer from this drawback. Furthermore, FAUST cannot support general reputation-based policies (across categories for example) and thresholds cannot be changed after tokens have been issued (the number of authenticator tokens needs to be determined after the policy has been set). Because of these limitations we do not consider FAUST to be a viable solution for reputation-based blacklisting. Nevertheless, FAUST has novel features that can be leveraged by PERM. For example, we assume that users download all the signatures for entries on the reputation lists (so as not to reveal which transaction identifiers are in their queue). FAUST shows how users can selectively download different subsets while maintaining statistical privacy. We leave such combinations to future work, although we believe that the downlink complexity of PERM is already reasonable.

Revocation window. While PERM greatly improves performance over BLACR, BLACR can score sessions at any time, even long after the session (e.g., months later). PERM on the other hand requires that sessions be scored within the revocation window, which is on the order of hours to a couple of days in practical settings. The authors of FAUST argue, "89% of vandalism instances were repaired within 100 views [on Wikipedia]...[revocation windows] as low as 30 minutes seem feasible" [22]. Thus, while some flexibility is traded off for performance, the revocation-window-based setting is practical for SPs such as Wikipedia. Nevertheless, we point out that *positive* scores may materialize over longer time periods, where, for example, an uploaded video becomes popular after several months. In those cases we show how reputation can be upgraded in PERM, thus offering a novel improvement to revocation-window-based schemes.

Unblacklisting. Related to the previous discussion point, SPs can unblacklist (forgive) misbehaviors by simply upgrading negative scores to zero.

Sybil attacks. Any credential scheme (including BLACR and PEREA) is vulnerable to the Sybil attack, where if one credential is revoked by PERM, the user may attempt to obtain another credential. We assume some Sybil-resistant mechanism while issuing credentials, as must be assumed with any such scheme. For example, an organization (e.g. a company or a university) already has the means to issue one credential per member in the organization. Online services could require a credit card and make note of which identities (using the identity on the credit card) have been issued credentials. This identity-revealing step is reasonable because future authentications are anonymous and the identity is used only during the step of issuing credentials.

Side channels. In PERM the amount of time taken to verify an authentication at the SP and the user is independent of the list size and is the same for all authenticating users. Therefore, PERM is resistant to side-channel leaks related to the time it takes to authenticate. Of course, users on different platforms would exhibit different latencies, and the client software can delay each authentication to some small constant (e.g. 2 seconds) so that all users exhibit uniform behavior. Other side channels are out of scope for this work.

6. CONCLUSION

Since TTP-free anonymous blacklisting was first introduced, several advances have been made in an attempt to make such schemes more practical and useful to service providers. Recent work on BLACR attempted to generalize the concept of anonymous blacklisting to allow *reputation-based blacklisting*, but did not offer a scalable solution. We believe PERM is the first scheme to support scalable and practical reputation-based blacklisting and thus presents a major advance in the progression of such schemes.

7. ACKNOWLEDGMENTS

We thank the late Patrick P. Tsang for his work on BLAC and PEREA, which provided inspiration for the ideas in this paper. We also thank the anonymous reviewers for their comments and John McCurley for his editorial help.

8. REFERENCES

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer, 2000.

[2] M. H. Au, A. Kapadia, and W. Susilo. BLACR: TTP-Free Blacklistable Anonymous Credentials with Reputation. In *Proceedings of The 19th Annual Network and Distributed System Security Symposium (NDSS)*, Feb. 2012.

[3] M. H. Au, W. Susilo, and Y. Mu. Constant-Size Dynamic k-TAA. In *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2006.

[4] M. H. Au, P. P. Tsang, and A. Kapadia. PEREA: Practical TTP-Free Revocation of Repeatedly Misbehaving Anonymous Users. *ACM Transactions on Information and System Security*, 14:29:1–29:34, Dec. 2011.

[5] M. H. Au, P. P. Tsang, A. Kapadia, and W. Susilo. BLACR: TTP-Free Blacklistable Anonymous Credentials with Reputation. Technical Report TR695, Indiana University Bloomington, May 2011.

[6] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[7] J. C. Benaloh and M. de Mare. One-Way Accumulators: A Decentralized Alternative to Digital Sinatures (Extended Abstract). In *EUROCRYPT*, pages 274–285, 1993.

[8] D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, 2004.

[9] J. Camenisch, R. Chaabouni, and A. Shelat. Efficient Protocols for Set Membership and Range Proofs. In *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 234–252. Springer, 2008.

[10] J. Camenisch and A. Lysyanskaya. A Signature Scheme with Efficient Protocols. In *SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer, 2002.

[11] J. Camenisch and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72, 2004.

[12] J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In B. S. K. Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997.

[13] D. Chaum and E. van Heyst. Group Signatures. In *EUROCRYPT*, pages 257–265, 1991.

[14] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In Y. Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, 1994.

[15] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Usenix Security Symposium*, pages 303–320, Aug. 2004.

[16] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[17] S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[18] R. Henry, K. Henry, and I. Goldberg. Making a Nymbler Nymble Using VERBS. In *Privacy Enhancing Technologies*, volume 6205 of *Lecture Notes in Computer Science*, pages 111–129, 2010.

[19] P. C. Johnson, A. Kapadia, P. P. Tsang, and S. W. Smith. Nymble: Anonymous IP-Address Blocking. In *Privacy Enhancing Technologies*, volume 4776 of *Lecture Notes in Computer Science*, pages 113–133. Springer, 2007.

[20] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable Signatures. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589. Springer, 2004.

[21] Z. Lin and N. Hopper. Jack: Scalable accumulator-based Nymble system. In *WPES*, pages 53–62, 2010.

[22] P. Lofgren and N. Hopper. FAUST: Efficient, TTP-Free Abuse Prevention by Anonymous Whitelisting. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES)*, Oct. 2011.

[23] T. P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140, 1992.

[24] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. Blacklistable Anonymous Credentials: Blocking Misbehaving Users without TTPs. In *ACM Conference on Computer and Communications Security*, pages 72–81. ACM, 2007.

[25] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. PEREA: Towards Practical TTP-Free Revocation in Anonymous Authentication. In *ACM Conference on*

Computer and Communications Security, pages 333–344, 2008.

[26] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. BLAC: Revoking Repeatedly Misbehaving Anonymous Users without Relying on TTPs. *ACM Trans. Inf. Syst. Secur.*, 13(4):39, 2010.

[27] P. P. Tsang, A. Kapadia, C. Cornelius, and S. W. Smith. Nymble: Blocking Misbehaving Users in Anonymizing Networks. *IEEE Trans. Dependable Sec. Comput.*, 8(2):256–269, 2011.

APPENDIX

A. SECURITY ANALYSIS

We adopt the simulation-based security definition as in PEREA [4]. In the *real world* there are a number of players who communicate via cryptographic protocols while in the *ideal world* the same players communicate via a trusted party $\mathcal{T}$ who is responsible for handling all the inputs and outputs for the players. The adversary $\mathcal{A}$ controls the same players in the *real world* and the *ideal world*. All the inputs and the scheduling of the players' interactions are decided by another probabilistic polynomial time (PPT) algorithm, and the environment $\mathcal{E}$. $\mathcal{A}$ can communicate arbitrarily with $\mathcal{E}$. Informally speaking, PERM is secure if for any PPT algorithms $\mathcal{A}$ and $\mathcal{E}$, there exists another algorithm $\mathcal{S}$ controlling the same players in the ideal world as $\mathcal{A}$ does in the real world such that $\mathcal{E}$ cannot tell if it is interacting with $\mathcal{A}$ or $\mathcal{S}$. $\mathcal{S}$ has black-box access to $\mathcal{A}$.

PERM supports a set of functionalities. An invocation of a functionality is an event. We assume all events are scheduled according to $\mathcal{E}$'s wishes. We use a static model and assume the number of players, and whether they are honest or not is fixed before the system starts. All communications with $\mathcal{T}$ are *not* anonymous, meaning that $\mathcal{T}$ knows the identity of the communicating parties. It is also assumed that communication between honest parties is not observed by $\mathcal{A}$ and when $\mathcal{A}$ receives a message, it does not learn its origin. PERM supports the following functionalities:

1. SP SETUP. The system begins when $\mathcal{E}$ specifies the number of honest and dishonest users and SPs.
   - *Real World.* The SP generates a key pair (PK, SK). PK is made available to all players in the system.
   - *Ideal World.* The trusted party $\mathcal{T}$ initializes a database, which stores the registration status and authentication history of all the users. To capture the functional requirement of PERM, $\mathcal{T}$ keeps track of the score of every user with respect to each category as well as all authentications that the user has participated in.
2. REGISTRATION. $\mathcal{E}$ instructs user i to register with the SP. Note that this procedure is not anonymous in the view of the SP. For all i, an honest SP would allow user i to register only once.
   - *Real World.* User i sends a request for registration to the SP. The user, as well as the SP, outputs individually the outcome of this transaction to $\mathcal{E}$. If user i has obtained a credential in a previous registration event, then an honest SP would reject the request. Likewise, an honest user would discard the second credential it obtains from the SP if it has successfully registered in a previous registration event.
   - *Ideal World.* User i sends a registration request to $\mathcal{T}$, who informs the SP that user i would like to register and whether user i has obtained a credential before. The SP returns its decision to $\mathcal{T}$, who forwards it back to the user. If the SP accepts the request and that user i has not registered before, $\mathcal{T}$ stores the registration status of user i in its database. The user, as well as the SP, individually output the outcome of this transaction to $\mathcal{E}$.
3. AUTHENTICATION. $\mathcal{E}$ instructs user i to authenticate with the SP and instructs the SP to impose an access policy $\mathcal{P}$.
   - *Real World.* User i conducts the authentication protocol with the SP imposing the access policy $\mathcal{P}$. The user, as well as the SP, output individually the outcome of this transaction as well as the transaction identifier t to $\mathcal{E}$.
   - *Ideal World.* User i sends a request to $\mathcal{T}$, who informs the SP some anonymous user requests an authentication. The SP replies with the list $\mathcal{L}$, the current transaction identifier t and the policy $\mathcal{P}$. $\mathcal{T}$ forwards the reputation lists, the value t and $\mathcal{P}$ back to user i and whether i satisfies the authentication policy or not. User i then decides if he/she would continue. If yes, $\mathcal{T}$ informs the SP whether the anonymous user satisfies the authentication policy or not. The SP replies with `accept` or `reject` to $\mathcal{T}$, who forwards the reply to user i. If the authentication is successful, $\mathcal{T}$ stores t as one of the user's transaction identifiers. The user, as well as the SP, output individually the outcome of this transaction as well as the transaction identifier t to $\mathcal{E}$.
4. SCORING A TRANSACTION. $\mathcal{E}$ instructs the SP to give a score of $(s_1, \ldots, s_J)$ to transaction identifier t. If t is not a valid authentication or has already been put on reputation list $\mathcal{L}$, an honest SP ignores this request.
5. UPDATING A SCORE. $\mathcal{E}$ instructs the SP to update a score of $(s_1, \ldots, s_J)$ for transaction identifier t. If t is not a valid transaction identifier on $\mathcal{L}$, then an honest SP would ignore this request.
6. SCORE UPDATE. $\mathcal{E}$ instructs user i to update his score on transaction identifier t. If t is not a past identifier of the user, an honest user ignores the request.
   - *Real World.* User i conducts the score update protocol with the SP. The user, as well as the SP, output individually the outcome of this transaction as well as the transaction identifier t to $\mathcal{E}$.
   - *Ideal World.* User i sends a request to $\mathcal{T}$, who informs the SP some anonymous user requests a score update on transaction identifier t. The SP replies with `accept` or `reject`. If the SP replies `accept`, $\mathcal{T}$ updates the stored reputation of the user. The user, as well as the SP, output individually the outcome of this transaction as well as the transaction identifier t to $\mathcal{E}$.

The ideal-world PERM provides all the desired security properties and functionalities of PERM. Firstly, all the transactions, in the view of the SP, are anonymous. $\mathcal{T}$ only informs the SP that some anonymous user would like to authenticate and thus anonymity is guaranteed. Secondly, $\mathcal{T}$ verifies whether the authenticating user satisfies the access policy and thus the system functions correctly. The real-world PERM is secure if its behavior is the same as the ideal-world PERM. Thus, assuming $\mathsf{negl}(\lambda)$ is a negligible function in security parameter λ, we have the following definition of security for any construction of PERM.

Definition 1. Security. Let $\mathbf{Real}_{\mathcal{E},\mathcal{A}}(\lambda)$ (resp. $\mathbf{Ideal}_{\mathcal{E},\mathcal{S}}(\lambda)$) be the probability that $\mathcal{E}$ outputs 1 when run in the real world (resp. ideal world) with adversary $\mathcal{A}$ (resp. $\mathcal{S}$ having black-box access to $\mathcal{A}$). PERM is secure if for all PPT algorithms $\mathcal{E}$, $\mathcal{A}$, the following expression holds:

$$|\mathbf{Real}_{\mathcal{E},\mathcal{A}}(\lambda) - \mathbf{Ideal}_{\mathcal{E},\mathcal{S}}(\lambda)| = \mathsf{negl}(\lambda)$$

To prove that PERM is secure, we have to construct an ideal-world adversary $\mathcal{S}$ given any real-world adversary $\mathcal{A}$ in such a way that no PPT environment $\mathcal{E}$ can distinguish whether it is interacting with $\mathcal{S}$ or $\mathcal{A}$.

The proof is divided into two cases according to the subset of players controlled by $\mathcal{A}$. In the first case, $\mathcal{A}$ controls the SP and a subset of users while, in the second case, only a subset of users is dishonest. Note, the latter is not a special case of the former. An adversary controlling the SP covers the security requirements of anonymity, while an adversary controlling a subset of users covers the security requirements of authenticity and non-frameability.

Adding players controlled by $\mathcal{A}$ does not necessarily make the construction of $\mathcal{S}$ more difficult. On one hand, the construction of $\mathcal{S}$ is trivial if $\mathcal{A}$ controls all players in the system, for $\mathcal{S}$ can simply forward all messages exchanged between $\mathcal{E}$ and $\mathcal{A}$. On the other hand, the construction of $\mathcal{S}$ is also trivial when all players in the system are honest. In that case, $\mathcal{A}$ did not participate in the system and the indistinguishability depends only on the correctness of the system.

We sketch the proof strategy of how $\mathcal{S}$ can be constructed in these two cases. Firstly, $\mathcal{S}$ maintains a list of 'current' credentials issued to $\mathcal{A}$ during the lifespan of the system. At the same time, $\mathcal{S}$ acts as an ideal-world adversary to the trusted party $\mathcal{T}$. $\mathcal{S}$ simply forwards any messages between $\mathcal{E}$ and $\mathcal{A}$. Next, we specify how $\mathcal{S}$ responds to each possible event in the two different cases.

Case 1: The SP is honest

- SP Setup.
  - *Representing an Honest SP to $\mathcal{A}$.* $\mathcal{S}$ generates the key pair (PK, SK) and gives PK to $\mathcal{A}$.
- Registration.
  - *Representing a dishonest user i to $\mathcal{T}$ / an honest SP to $\mathcal{A}$.* Using the zero-knowledge extractor, $\mathcal{S}$ extracts from $\mathcal{A}$ the value x. x will be used to identify the dishonest user i. $\mathcal{S}$ sends the request to $\mathcal{T}$ on behalf of user i. If $\mathcal{T}$ replies `accept`, $\mathcal{S}$ issues the credential to $\mathcal{A}$ and also stores that credential.
- Authentication. Note that $\mathcal{S}$ does not receive $\mathcal{P}$ from $\mathcal{E}$ directly since $\mathcal{P}$ is sent to the honest SP in the ideal world. However, $\mathcal{S}$ learns about $\mathcal{P}$ from $\mathcal{T}$ on behalf of the dishonest user i in the ideal world.
  - *Representing a dishonest user i to $\mathcal{T}$/ an honest SP to $\mathcal{A}$.* The difficulty here is that $\mathcal{S}$ does not know which credential $\mathcal{A}$ is using for the authentication. For instance, while $\mathcal{E}$ specifies the user i should perform the authentication, it is entirely possible for $\mathcal{A}$ to use the credential from another dishonest user say, $\hat{i}$, to perform the authentication. To locate the actual user, $\mathcal{S}$ extracts and uses the value x during the authentication to locate the correct user.

The outputs of $\mathcal{S}$ and the honest users in the ideal world are always indistinguishable to $\mathcal{A}$ and the honest users in the real world unless the following happen. We also explain why such cases happen with negligible probability below.

1. During a Registration event, $\mathcal{S}$ fails to extract from $\mathcal{A}$ the value x. This happens with negligible probability under the soundness property of the protocol $\mathfrak{S}_{\mathsf{Iss}}$ of the BBS+ signature scheme.
2. During a successful Authentication event, $\mathcal{S}$ fails to extract from $\mathcal{A}$ the values x. This happens with negligible probability under the soundness property of the protocol $\mathfrak{S}_{\mathsf{Sig}}$ of the BBS+ signature scheme.
3. There exists a successful Authentication event from $\mathcal{A}$ such that $\mathcal{S}$ on behalf of an honest SP outputs `accept`, but $\mathcal{T}$ indicates the authenticating user does not satisfy the policy. This represents either that $\mathcal{A}$ has been able to fake one of the proofs in the authentication or $\mathcal{A}$ can forge a signature on a new queue that has never been signed. All these happen with negligible probability under the assumption that BBS+ signatures are existentially unforgeable and that the interval proof is sound.

Note that in the security proof, we require $\mathcal{S}$ to run the zero-knowledge extractor on $\mathcal{A}$ for each registration and authentication event. To keep $\mathcal{S}$ in polynomial-time, we have to require authentication and registration events are to be executed sequentially (security proofs of BLACR and PEREA also impose this restriction) or to employ the stronger universally composable proofs of knowledge.

Case 2: The SP is dishonest

- SP Setup.
  - *Representing Honest users to $\mathcal{A}$.* $\mathcal{S}$ receives PK from $\mathcal{A}$.
- Registration.
  - *Representing a dishonest user to $\mathcal{T}$ / an honest user i to $\mathcal{A}$.* Upon receiving a registration request from $\mathcal{T}$ on behalf of user i, $\mathcal{S}$ engages $\mathcal{A}$ in the registration protocol, using the zero-knowledge simulator to simulate the ZKPoK in $\mathfrak{S}_{\mathsf{Iss}}$. If $\mathcal{S}$ fails to obtain a valid credential from $\mathcal{A}$, then $\mathcal{S}$ replies `reject` to $\mathcal{T}$.
- Authentication.
  - *Representing a dishonest SP to $\mathcal{T}$/ an honest user to $\mathcal{A}$.* Upon receiving an authentication request from $\mathcal{T}$ on behalf of an anonymous user, $\mathcal{S}$ engages $\mathcal{A}$ in the authentication protocol. If $\mathcal{T}$ replies with a bit indicating that the underlying user would proceed and satisfies the authentication policy, $\mathcal{S}$ uses the zero-knowledge simulator to simulate the ZKPoK proofs in the authentication protocol using a random value q. If $\mathcal{A}$ rejects the authentication, $\mathcal{S}$ replies `reject` to $\mathcal{T}$.

The simulation provided to $\mathcal{A}$ is perfect due to the zero-knowledgeness of the ZKPoK protocols and the perfect hiding property of the commitment scheme. At the same time, the behavior of $\mathcal{S}$ in the ideal world is the same as that of $\mathcal{A}$ in the real world. Thus, the output of $\mathcal{S}$ to the environment $\mathcal{E}$ is indistinguishable from that of $\mathcal{A}$.

Based on this dual strategy in the construction of $\mathcal{S}$, our construction of PERM is secure according to Definition 1.