PE(AR)^2: Privacy-Enhanced Anonymous Authentication with Reputation and Revocation

Kin Ying Yu[1], Tsz Hon Yuen[1], Sherman S.M. Chow[2], Siu Ming Yiu[1], and Lucas C.K. Hui[1]

[1] Department of Computer Science, The University of Hong Kong
{kyyu,thyuen,smyiu,hui}@cs.hku.hk
[2] Department of Combinatorics and Optimization
University of Waterloo, Ontario, Canada N2L 3G1
smchow@math.uwaterloo.ca

Abstract. Anonymous authentication schemes allow users to act freely without being tracked. The users may not want to trust a third party in ensuring their privacy, yet a service provider (SP) should have the authority to blacklist a misbehaving user. They are seemingly contradicting requirements. PEREA was the most efficient solution to this problem. However, there are a few drawbacks which make it vulnerable and not practical enough. In this paper, we propose PE(AR)^2, which not only fixes PEREA's vulnerability, but also significantly improves its computation efficiency. Apart from revoking repeated misbehaving users, our system also rewards anonymous users via a built-in reputation system. Our scheme does not require the SP to timely review all previously authenticated sessions, and does not have the dependency on the blacklist size for user-side computation (c.f. EPID/BLAC(R)). Our benchmark on PE(AR)^2 shows that an SP can handle over 160 requests/second – a 460-fold efficiency improvement over PEREA, when the credentials store 1000 single-use tickets.

1 Introduction

It is common nowadays that user identification and authentication are required to perform actions on web applications in the Internet. Some of these applications depend much on users contributing contents or forming a user community to encourage interactions from each other. While most of these web applications require users to register with a self generated login identifier, the true identities of users could actually be revealed by skilled data miners from user-provided information, behavior of consecutive sessions, etc. Privacy concern on the use of these web applications is becoming a more prevailing issue.

Take Wikipedia, a collaboration based encyclopedia website, as an example; anyone could become an editor by contributing contents, or moderating any existing contents. This editing model removes the entry barrier for using the service, thus encourages a lot of content submissions. However, the identities of users who submit contents are traceable, which creates a lot of privacy concerns.

S. Foresti, M. Yung, and F. Martinelli (Eds.): ESORICS 2012, LNCS 7459, pp. 679–696, 2012.

When a user identity is linked to specific document content, reviewers or readers of the document could become biased because they may think that this particular user could be representing a particular organization or community, especially for government organizations or political parties. Another example is Internet discussion forums. Users of common interests on certain topics often express their views on these forums, and other users could further comment on the same threads. In contrast to any verbal conversation which is more transient in nature, users could become much more liable to what they put on the forum. In some cases, lawsuits are resulted. Traceable identities would make users be less liberal in expressing ideas or viewpoints in the worry of censorship.

One trivial solution to the above problem is to allow anonymous users to perform actions without authentications. However this solution creates another problem: misbehaving users could not be stopped. Referencing the Internet discussion forum example again, a misbehaving user could damage the forum by spamming, advertising, or reprimanding threads created by other users. Blacklisting users would not be possible if all users are perfectly anonymous. A forum moderator could only keep removing destructive contents manually.

Anonymous Authentication without Trusted Third Party. Anonymous credential schemes enable anonymous authentications among a set of registered users. These schemes may reduce the number of misbehaving users because the system owner can control who can be registered, and possibly revoke a user's membership when misbehavior is detected. To enable revocation, an intuitive idea is to introduce a trusted third party (TTP) who owns the trapdoor that can reveal the true identity of any user. There are systems based on group signatures (e.g., [1]) or accumulators (e.g., [2]) which use TTP-based revocation.

To enhance users' privacy, or to reduce the trust they need to put, BLAC [3] and EPID [4] eliminate the use of TTP in revocation. A user is required to prove in zero-knowledge that the credential is not listed in the blacklist, and the blacklist can also be constructed from the past authentication transcript without any trapdoor information. However, as the zero-knowledge proof needs to be performed on each of the L entries in the blacklist, the computation complexity for authentication is $O(L)$. These schemes become impractical in practice since the blacklist will keep growing and there is no way to reduce its size, except resetting the system and updating all users' credentials.

To make the authentication process more efficient, PEREA [5] is proposed. In PEREA, a user's credential stores K single-use tickets using a dynamical universal accumulator. After each authentication, the service provider (SP) will certify a new queue of tickets (the used ticket is dequeued and a new one is enqueued in a sliding window manner). Tickets are randomly generated by users which also serve as the identifiers of authenticated sessions. Each authentication spends one ticket, and will then be evaluated (according to certain criteria external to PEREA). If the need of blacklisting arises, the SP will put the corresponding ticket on the blacklist. Revocation check is enforced by a zero-knowledge proof-of-knowledge (ZKPoK) about the possession of a signature on the queued element, and a non-membership proof for each of the

queued tickets with respect to the SP side accumulator (storing blacklisted sessions' tickets). While the blacklist size is still $O(L)$, the computation overhead on an authentication step is $O(K)$ since it only needs $O(1)$ time to check each of the K tickets is not in the blacklist using the proof systems provided with the accumulator. Higher efficiency may be obtained by optimizing the parameter K which also governs the time window that the SP must catch any user misbehavior.

Problems with PEREA. The above design appeared to make a significant step towards practical TTP-free revocation in anonymous authentication. However, we observe that this design also has its problems for a practical deployment.

Recall that the SP needs to review authenticated sessions in a timely manner based on the choice of K. Otherwise, a user can quickly get authenticated for K times, get an entirely new queue of tickets, and will not be denied from the service even if all those K sessions are later reviewed as problematic, i.e., the users would be able to "clean" all unreviewed sessions from their queues before they have been published by the SP as revoked sessions. This could only be done by limiting the rate a user can use the service (i.e., a user could only be authenticated for K requests in a predefined period, within that period the blacklisted tickets must be determined). Even if the time window for rate-limiting is synchronized across all users, this still creates a few undesirable properties.

First, if there is no misbehavior whatsoever, the SP might not care much if a user "overuses" the service. Indeed, in applications such as Internet forums, high activity is what an SP wants to see. On the other hand, the SP is now in a stressed situation that all sessions conducted in the predefined period must be reviewed properly, or otherwise a session with misbehavior could not be revoked. Moreover, as blacklisted tickets are required to be published timely, the blacklist accumulator becomes volatile. Consequently, the witness for the non-membership proof for the users' tickets queue also requires updates (even if the tickets queue remains unchanged), which causes user-side overhead. Most importantly, observe that the requirement of *rate-limiting* means we need another kind of protocol (e.g., [6]) which is able to profile the behavior of the *same* user over a period of time. It is not clear how rate-limiting can be tightly coupled with PEREA, *without reducing anonymity*, not to say the additional overhead for such feature.

In terms of performance of deployment, recall that the computational overhead for authentication is dependent on the ticket queue size K. For the SP to achieve practical performance in authentication phase, it was suggested to use $K = 10$ for completing computation under 0.1 seconds using typical hardware. This imposes a very strict requirement for the SP to evaluate each session. Another issue related to realization of an accumulator-based design is that, the proof of non-membership protocol associated with the accumulator is not as "secure" as assumed in PEREA. A recent research [7] has broken the security of the proof of non-membership protocol used [8], which in turns breaks the security of PEREA.

Reputation and Naughtiness. Reputation system is widely used in many online services. (A survey of trust and reputation systems can be found in [9].)

The user reputation is an aggregate (e.g., sum) of the rating of all previously conducted sessions. It can be used as a way to profile a user, and then used for rewarding or punishment upon user's future authentications. For example, in Yahoo! Knowledge^{+}[1], each user is associated with a score, which is an aggregate sum of all individual score gained from the contributed answers. An answer submitted by a user with high score may implicate that the user has been successfully helping many others and hence is considered to be more trustworthy.

PEREA requires the SP to reject authentication once a user has been put on the blacklist. This may be reasonable for applications that only requires an "all or nothing" authentication, but real world applications usually require login session to be associated with different privilege levels such that the SP can provide different access rights to the user. An example could be an abusive user who keeps posting defamation messages to a forum should have the message posting suspended, but still has access right to the forum. Such operation is not possible in the basic setting of PEREA.

In view of this shortcoming, the basic PEREA is extended [5] to revoke users based on "naughtiness", a severity measure of misconducts for a user's K most recently conducted sessions. However, it is not useful in a reputation system since it can only capture the most recent K sessions.

Recently, Au, Kapadia and Susilo [10] proposed BLACR that extended BLAC to support reputation. The basic BLACR subjects to the same inefficiency as BLAC. Their BLACR-Express tries to decouple the blacklist in the past from the performance of authentication, by asking the SP to issue some "express passes" to privileged users. However, the management of these passes costs extra burden on the SP, especially when applied to numerous users in the system.

Our Contribution. We propose PE(AR)^2, which preserves the same anonymous authentication functionalities as PEREA, but with a few improvements. Our PE(AR)^2 also has a built-in reputation system. Each user can obtain "scores" from the past sessions. Users can also prove (in zero-knowledge) that their scores are higher than a certain threshold. The SP can then provide some privilege services accordingly.

PE(AR)^2 has a few advantages. Firstly, it does not require rate-limiting as in PEREA for avoiding any malicious user to shift tickets which will be potentially blacklisted out of the queue. Another advantage is that we no longer require the SP to publish the blacklist in a timely manner for the accuracy of revocation. The SP can then have higher flexibility and better resource scheduling on managing blacklist. Otherwise, the SP might potentially blacklist more sessions than needed, and unblacklist some of them afterwards. say when an "innocent" (and unhappy) user filed a dispute case. As the changes in the blacklist is less volatile, this also helps to achieve better computation efficiency in the user side.

We also remodel the structure of how authentication and blacklisting are done in PEREA to have further improvement in terms of efficiency. In particular, the complexity of SP-side for verifying an authentication in PE(AR)^2 does not depend on the ticket queue size K anymore; and for user side, the dependency

[1] `http://hk.knowledge.yahoo.com`

on K is changed from $O(K)$ exponentiations to $O(K)$ divisions, which are much cheaper. We further benchmark the performance of our implementation and show our design is indeed practical and feasible in real world use case.

Regarding reputation, we introduce a score redemption algorithm, such that users are allowed to remove tickets and reclaim the scores associated with the reviewed sessions, thus keeping the ticket queue in a practically small size and making the scores reclaimed aggregatable.

2 Formal Definitions

An anonymous authentication scheme with reputation and revocation, executed between a user $\mathcal{U}$ and a service provider $\mathcal{S}$, consists of the following algorithms:

- $\mathsf{KeyGen}(1^\ell) \to (\mathsf{mpk}, \mathsf{msk})$: KeyGen is the key generation algorithm for the service provider $\mathcal{S}$ that outputs the key pair.
- $\mathsf{Reg}(\mathcal{U}(\mathsf{mpk}), \mathcal{S}(\mathsf{msk})) \to \{\mathcal{U}(\mathsf{cred}), \mathcal{S}()\}$: Reg is the registration protocol that outputs a credential cred to the user, which includes a list of tickets $\mathbb{T}$ not known by $\mathcal{S}$.
- $\mathsf{Auth}(\mathcal{U}(\mathsf{mpk}, \mathsf{cred}, s_{\mathsf{base}}), \mathcal{S}(\mathsf{msk})) \to \{\mathcal{U}(\mathsf{cred}), \mathcal{S}(t, s_{\mathsf{base}})\}$: Auth is the authentication protocol that gives a ticket (session identifier) $t \in \mathbb{T}$ to the SP, where $\mathbb{T}$ is in cred. If the ticket t is not used before, the used tickets in $\mathbb{T}$ are not in the blacklist $\mathbb{B}$, and the score s in cred is larger than s_{base}, then the user is authenticated. The SP stores $(t, 0, \perp)$ in the ticket score list $\mathbb{L}$, and the user refreshes his credential.
- $\mathsf{Revoke}(t) \to \{\mathcal{U}(t), \mathcal{S}(t)\}$: Revoke is the procedure for the SP to put a ticket t on the blacklist, which revokes the credential where t is originated.
- $\mathsf{Rate}(t, s)$: $\mathcal{S}$ receives score s from the reviewers for the ticket t, updates (t, s, Pf) to the list $\mathbb{L}$, where Pf is a proof of validity of the rating.
- $\mathsf{Redeem}(\mathcal{U}(\mathsf{mpk}, \mathsf{cred}, \mathbb{L}_s), \mathcal{S}(\mathsf{msk})) \to \{\mathcal{U}(\mathsf{cred}), \mathcal{S}()\}$: Redeem is the algorithm that allows $\mathcal{U}$ to update the score in the credential according to the ticket list $\mathbb{L}_s$. $\mathcal{U}$ proves that some of his tickets in cred are in $\mathbb{L}_s \subseteq \mathbb{L} \setminus \mathbb{B}$. Finally $\mathcal{U}$ obtains an updated credential with new score s_{new}, the summation of the scores in $\mathbb{L}$ for all tickets in $\mathbb{L}_s$, with new tickets refilled, and with the tickets in $\mathbb{L}_s$ removed from the credential.

An anonymous authentication scheme with reputation and revocation should provide the properties *Misauthentication Resistance*, *Revocability*, *Anonymity*, *Unlinkability*, *Backward Untraceability* and *Identity-Escrow Freeness* as defined in PEREA [5], as well as the following properties:

- **Reputation**. (Completeness:) An honest user who has not been revoked should be able to be authenticated by an honest SP if his score is larger than s_{base}; and (Soundness:) No registered user can authenticate with an honest SP for a score s_{base} if his score is less than or equal to s_{base}.
- **Rating Unforgeability**. The rating is only performed by the honest SP.

Their formal definitions will be given via two main notions to be defined later.

In a reputation system, the unlinkability is guaranteed when the ticket is not redeemed. Some anonymity may be lost during the redeem protocol under some extreme cases. For example, if tickets A and B both have score 100 while the other 5 tickets in the system have score 1. If one redeems his tickets for a total score 200, then we conclude that tickets A and B are used by the same user. This situation would be more noticeable when the user community is small or the distribution of scores are uneven. This kind of unlinkability lost cannot be completely eliminated from reputation system. However, as the number of users and scored tickets gets large, then such issue becomes less likely to occur.

Security Models. We formally define the security notions as games played between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. $\mathcal{A}$ can arbitrarily and adaptively query various oracles, which together share a private state st_n that contains counters n, and sets U_P, U_A, U_B, which are initialized to 0 and $\varnothing$, respectively. Here we define a number of oracles, modeling an adversary's attacking power.

Our P-Reg, A-Reg, B-Reg, CorruptU, P-Auth, A-Auth, B-Auth, Add-to-BL, Remove-From-BL Oracles are almost the same as the oracles in PEREA [5]. The meanings of P, A and B in various Reg and Auth oracles are similar to those to be described in three different kinds of Redeem oracles. Below we will highlight the differences between our model and the existing model [5].

- Our private state st_n stores three tuples, including the user counter n and the credential as in [5], and also the refresh counter a which is used to differentiate past and present credentials. We use $\mathsf{cred}_{n,a}$ to represent each credential.
- In A-Reg (resp. A-Auth) Oracle, $\mathcal{A}$ sends all the randomness he wants to use in the Reg (resp. Auth) protocol. The oracle runs it using $\mathcal{A}$'s randomness for the corrupt user. It is because the challenger has to know the credentials of the corrupt users to prevent $\mathcal{A}$ from winning trivially using them.
- In CorruptU Oracle, it also takes the refresh counter a as an extra input, in order to simulate the corruption of past credentials for backward security.

The following oracles are newly introduced in our model:

- Rate Oracle: allows $\mathcal{A}$ to assign scores for tickets. On input a ticket t and a score s, the oracles updates (t, s, Pf) to the list $\mathbb{L}$.
- P-Redeem Oracle (resp. B-Redeem Oracle; A-Redeem Oracle): allows $\mathcal{A}$ to eavesdrop a redemption run between an honest user and an honest SP (resp. an honest user and a corrupt SP; or a corrupt user and an honest SP). The oracle description is similar to the P-Auth Oracle (resp. B-Auth Oracle; A-Auth Oracle), except that we replace the Auth protocol with the Redeem protocol; and the state st_n finally stores the refreshed credential with the new score s_{score} (the summation of scores of tickets in $\mathbb{L}_s$).
- A-Redeem Oracle: allows a corrupt user to redeem with an honest SP. On input i such that $i \in U_A$, the oracle searches for $\langle i, \mathsf{cred}_{i,a}, a\rangle$ from st_n with the largest a, plays the role of the SP and interacts with $\mathcal{A}$ in the Redeem

protocol. $\mathcal{A}$ sends all the randomness he wants to use and the oracles runs the Redeem protocol using them for the corrupt user i. If the redeem is successful, the oracles appends $\langle i, \mathsf{cred}_{i,a+1}, a+1\rangle$ to st_n, where $\mathsf{cred}_{i,a+1}$ is the refreshed credential with score s_{score} (the summation of scores of tickets in $\mathbb{L}_s$). The oracle returns $(\pi_a, a, \mathsf{cred}_{i,a+1})$ to $\mathcal{A}$, where π_a is the resulting protocol transcript.

We are now ready to introduce two security games to capture the properties of anonymous authentication [2].

Accountability. We capture the misauthentication resistance, revocability and reputation properties. In this game, the adversary is allowed to act as unregistered users, revoked users, and registered users without sufficient score. No coalition of these users can authenticate with the honest SP. The security game between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ is defined as follows.

1. (Setup.) $\mathcal{C}$ runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\ell)$ and gives mpk to $\mathcal{A}$.
2. (Query.) $\mathcal{A}$ can issue queries to all the oracles except those start with B-.
3. (End game.) $\mathcal{A}$ runs Auth with $\mathcal{C}$ and $\mathcal{C}$ obtains a ticket t^* and s^*_{base}.

Denote a^* as the largest number such that the ticket t^* is in cred_{n^*,a^*}, for $\langle n^*, \mathsf{cred}_{n^*,a^*}, a^*\rangle$ stored in st_n. $\mathcal{A}$ wins the game if one of the following holds:

- Case 1: (unregistered user) a^* cannot be found since t^* is not in any $\mathsf{cred}_{n,a}$.
- Case 2: (honest-looking user) $n^* \notin U_A$; and t^* is not stored in $\mathbb{L}$.
- Case 3: (malicious registered user): One of the following holds:
  1. ($\mathcal{A}$ reuses old tickets) t^* is stored in $\mathbb{L}$, or
  2. ($\mathcal{A}$ does not have enough score) s_{score} (the score of cred_{n^*,a^*}) $\leq s^*_{\mathsf{base}}$, or
  3. ($\mathcal{A}$ is blacklisted) $\exists \hat{t} \in \mathsf{cred}_{n^*,a^*}$ such that ticket $\hat{t}$ is in the blacklist.

The advantage of $\mathcal{A}$ is the probability that $\mathcal{A}$ wins the game.

Definition 1. *An anonymous authentication scheme is accountable if there is no PPT adversary $\mathcal{A}$ has a non-negligible advantage in the above game.*

The accountability model captures the misauthentication resistance since the adversary can pretend to be an honest user (by setting $n^* \in U_p \cup U_B$ in case 2) or try to authenticate as an unregistered user (by case 1). The model also captures the revocability and reputation, since in case 3 the challenge user is either blacklisted, is reusing old tickets or trying to authenticate with low score.

Privacy. We capture the anonymity, unlinkability and backward untraceability properties in the game below, where the adversary could act as a corrupted SP.

1. (Setup.) $\mathcal{C}$ runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\ell)$ and gives $(\mathsf{mpk}, \mathsf{msk})$ to $\mathcal{A}$.
2. (Query 1.) $\mathcal{A}$ is allowed to issue queries to all the oracles except those start with P- and A-.

[2] Completeness is easy to define. Rating unforgeability can be easily captured by the standard unforgeability of the Rate protocol. We omit them due to space limit.

3. (Challenge.) $\mathcal{A}$ picks i_0 and i_1 in U_B and sends them to $\mathcal{C}$. Denote the score of i_0 (resp. i_1) in st_n as s_0 (resp. s_1). $\mathcal{C}$ randomly picks a bit $b \in \{0,1\}$. After that, $\mathcal{A}$ can ask $\mathcal{C}$ to run B-Auth Oracle twice with $s_{\mathsf{base}} > s_0, s_1$, and without specifying the input. $\mathcal{C}$ answers the query assuming the input is i_b and i_{1-b} respectively. Denote the ticket used as t_0^* and t_1^* respectively.
4. (Query 2.) $\mathcal{A}$ is allowed to issue queries to all the oracles except CorruptU Oracle with input i_0 or i_1 and oracles start with P- and A-. $\mathcal{A}$ is allowed to query the B-Redeem Oracle if t_0^* and t_1^* are not involved. Redeeming t_0^* and t_1^* by the B-Redeem Oracle is allowed if and only if all tickets to be redeemed by i_b and i_{1-b} have the same total scores and the same number of tickets.
5. (End game.) $\mathcal{A}$ outputs a guess bit b'. $\mathcal{A}$ wins if $b = b'$.

The advantage of $\mathcal{A}$ is the probability that $\mathcal{A}$ wins the game minus 1/2.

Definition 2. *An anonymous authentication scheme is private if there is no PPT adversary $\mathcal{A}$ has a non-negligible advantage in the above game.*

The privacy model captures anonymity since the adversary tries to distinguish between two honest users from the challenge authentication instance. It captures unlinkability since the adversary can ask the challenge users to run many instances of Auth with him before and after the challenge phase. It captures backward untraceability since the adversary is allowed to put the tickets of the challenge users on the blacklist, without affecting the other two properties.

3 Building Blocks

We state the notations and constructions of some fundamental cryptographic building blocks used.

Zero-Knowledge Proof-of-Knowledge (ZKPoK). For a language L with witness relation R_L, a proof system is a triplet of algorithms (K, P, V):

- K: on input 1^ℓ, outputs a common reference string crs.
- P: on input crs, x and its witness w, outputs a proof π if $(x, w) \in R_L$.
- V: on input crs, x and its proof π, outputs 1 for accept and 0 for reject.

We use the standard notation (due to Camenisch and Stadler) $PK\{(\alpha, \rho) \colon z = g^\alpha h^\rho\}$, to denote a proof of knowledge of (α, ρ) where $z = g^\alpha h^\rho$ is satisfied.

CL Signature. CL signature [11] is a secure signature scheme which allows signing on (commitments of) a vector of message and proving the possession of valid signatures in zero-knowledge. The algorithm $\mathsf{KeyGen}_{\mathsf{CL}}(1^\ell)$ outputs a public key $\mathsf{pk}_{\mathsf{CL}} = (N, g, h)$ where N is a safe prime product, $g, h \in \mathbb{QR}_N$; and a secret key $\mathsf{sk}_{\mathsf{CL}} = \phi(N)$. For a vector of messages $(\alpha_0, \ldots, \alpha_k)$, the signature on the commitment of $(\alpha_0, \ldots, \alpha_k)$ is denoted as $\sigma \leftarrow \mathsf{Sign}_{\mathsf{CL}}(\{\alpha_i\}_{0\ldots k}, \mathsf{sk}_{\mathsf{CL}})$. The verification algorithm is denoted as $1/0 \leftarrow \mathsf{Verify}_{\mathsf{CL}}(\{\alpha_i\}_{0\ldots k}, \sigma, \mathsf{pk}_{\mathsf{CL}})$ for valid/invalid signature. More details likes its construction can be found in [11].

Dynamically Universal Accumulator (DUA). DUA [8] is a suite of algorithms/protocols which allows an accumulation of a set of values into a single accumulator, and the proof of knowledge of a non-membership or membership witness of any value, with respect to an accumulator. We summarize the scheme in [8] as follows. In PE(AR)^2, the tickets are the values to be accumulated.

DUA.Setup. On input the security parameter 1^ℓ, it picks a safe prime product N and $g_{\mathtt{Acc}} \in_R \mathbb{QR}_N$. It compute the public/private key pair $\mathsf{pk}_{\mathtt{Acc}} = (N, g_{\mathtt{Acc}})$, $\mathsf{sk}_{\mathtt{Acc}} = \phi(N)$, and sets the initial accumulator value $c = g_{\mathtt{Acc}}$.

DUA.Accumulating tickets. Denote $\mathsf{Accumulate}(c, \{t_i\}_{0\ldots\mathsf{L}})$ as the algorithm to accumulate a list of tickets $\{t_i\}_{0\ldots\mathsf{L}}$ to the accumulator value c. It computes $\hat{c} = c^{\Pi_{0\leqslant i\leqslant\mathsf{L}} t_i}$. $\mathcal{S}$ updates $c = \hat{c}$. Note that decumlating tickets can be done similarly by computing $\hat{c} = c^{\Pi_{0\leqslant i\leqslant\mathsf{L}} t_i^{-1} \bmod \phi(N)}$ using $\mathsf{sk}_{\mathtt{Acc}}$.

DUA.Non-membership witness generation.[3] Denote $\mathsf{Compute}_{\mathtt{nmw}}(x, \{t_i\}_{0\ldots\mathsf{L}}, \mathsf{sk}_{\mathtt{Acc}})$ as the following algorithm to generate a non-membership witness w, where x is the ticket to be witnessed, $\{t_i\}_{0\ldots\mathsf{L}}$ is the list of accumulated tickets.

1. Compute the accumulated ticket product $u = \prod_{i=0}^{\mathsf{L}} t_i$.
2. Compute $u' = u \bmod \phi(N)$.
3. Since x is not a factor of u, $\mathsf{gcd}(x, u') = 1$. By Euclidean algorithm, find a and b such that $au' + bx = 1 \bmod \phi(N)$.
4. Output the witness w in the form of $(a, d) = (a, g_{\mathtt{Acc}}^{-b})$.

The accumulator $c = g_{\mathtt{Acc}}^{\Pi_{0\leqslant i\leqslant\mathsf{L}} t_i}$ has the witness $w = (a, d)$ which can be validated by checking if $c^a = d^x g_{\mathtt{Acc}} \bmod N$. Denote this check by $\mathsf{Verify}_{\mathtt{nmw}}(c, x, w, \mathsf{pk}_{\mathtt{Acc}})$. It outputs 1 if the witness is valid, or 0 otherwise.

DUA.Non-membership witness update. Denote $\mathsf{Update}_{\mathtt{nmw}}(w, c, x, \{t_i\}_{\mathsf{L}+1\ldots\mathsf{M}})$ as the following algorithm to update the non-membership witness $w = (a, d)$, where c is the original accumulator value, x is the ticket to be witnessed, and tickets $t_{\mathsf{L}} + 1, \ldots, t_{\mathsf{M}}$ are the newly accumulated tickets which are absent in c.

1. Compute the new accumulated ticket product $\hat{u} = \prod_{i=\mathsf{L}+1}^{\mathsf{M}} t_i$.
2. Compute the new accumulator value $\hat{c} = c^{\hat{u}} \bmod N$.
3. Since x is not a factor of $\hat{u}$, by Euclidean algorithm, find a_0 and r_0 such that $\hat{a}_0\hat{u} + r_0 x = 1$. Compute $\hat{a} = \hat{a}_0 a \bmod x$.
4. Find r such that $\hat{a}\hat{u} = a + rx$. Output the updated witness $\hat{w}$ in the form of $(\hat{a}, \hat{d}) = (\hat{a}, dc^r \bmod N)$.

The updated witness $\hat{w} = (\hat{a}, \hat{d})$ is valid if $\hat{c}^{\hat{a}} = \hat{d}^x g_{\mathtt{Acc}} \bmod N$ holds.

DUA.Non-membership witness proof-of-knowledge. Setting $\mathsf{pk}_{\mathtt{Acc}} = (N, g_{\mathtt{Acc}}, g, h)$, on a hidden ticket x, a random value r and witness (a, d), a prover runs the ZKPoK with $\mathcal{S}$: $PK_{\mathtt{nmw}}\{(x, r, a, d) : \mathsf{comm}_x = g^x h^r \wedge c^a = d^x g_{\mathtt{Acc}}\}$ to prove the

[3] Note that this version of DUA is vulnerable to an attack described in [7]. Our instantiation is described in Section 4.1.

condition $\mathsf{gcd}(x, u) = 1$, where u is the product of all ticket accumulated, is satisfied. Instantiation of such could be found in [8, § 5].

Intractability Assumption. Both CL signature and DUA rely on the *strong RSA assumption*, which is, on input of an RSA modulus N and an element $u \in \mathbb{Z}_N^*$, it is computationally hard to find a pair (v, e) such that $v^e = u \bmod N$ where $e > 1$.

4 A New Scheme: PE(AR)²

In this section, we propose a new anonymous authentication scheme that settles the pitfall and vulnerability of PEREA. We name it as PE(AR)², denoting Privacy-Enhanced Anonymous Authentication with Reputation and Revocation.

4.1 Our Scheme as Improvement to PEREA

Fixing Vulnerability of Dynamic Universal Accumulator (DUA). As stated in [7], the non-membership witness generation algorithm executed by an SP [8] suffers from an attack of extraction of multiple of $\phi(N)$, which is the secret key. The attack is successful because the original scheme [8] makes use of $u' = u \bmod \phi(N)$ in order to efficiently generate a witness $w = (a, d)$ that satisfies $au' + bx = 1 \bmod \phi(N)$ and $d = g_{\mathsf{Acc}}^{-b} \bmod N$.

Yet, the use of $\phi(N)$ to generate witness is not necessary. As stated in the original DUA scheme, users (including SP) can use a less efficient method, without using of $\phi(N)$, to obtain the witness. It can still be generated from $au + bx = 1$ and $d = g_{\mathsf{Acc}}^{-b} \bmod N$ directly. As the computation complexity for the non-membership witness generation is $O(l^2)$, where l is the length of the larger number in u and x, the introduced overhead is negligible as compared with other heavy operations such as large number exponentiations.

Here is the fixed version of the DUA non-membership witness generation, which takes the public key $\mathsf{pk}_{\mathsf{Acc}}$ instead of the private key $\mathsf{sk}_{\mathsf{Acc}}$.

$\mathsf{Compute}_{\mathtt{nmw}}(x, \{t_i\}_{0\ldots\mathsf{L}}, \mathsf{pk}_{\mathsf{Acc}})$:

1. Compute the accumulated ticket product $u = \prod_{i=0}^{\mathsf{L}} t_i$.
2. Find a and b such that $au + bx = 1$ by Euclidean algorithm.
   (Since x is not a factor of u, $\mathsf{gcd}(x, u) = 1$.)
3. Output the witness w in the form of $(a, d) = (a, g_{\mathsf{Acc}}^{-b})$.

Improving Efficiency and Practicability. PEREA relies on a queue based structure of size K in the user side, which mandates the user to generate (or update) K witnesses in every authentication request. Thus the computation overhead is proportional to K times the overhead of DUA witness generation.

PE(AR)² removes the need of the queue, and use the product value of all tickets in all witness generations. Since the accumulator is in the form of $V = g_{\mathsf{Acc}}^X$ where $X = \prod x_i$ and x_i's are prime numbers, we can combine the non-membership witness verification of K tickets $\{t_1, \ldots, t_K\}$ to one single non-membership witness verification of a combined value $T = \prod t_i$.

Notice that now the non-membership witness verification will fail if *any* of the tickets among these K tickets are accumulated in the accumulator, since the underlying accumulator's non-membership proof [8] ensures the fact that $\mathsf{gcd}(X, T) \neq 1$ for the case. This gives another efficiency improvement to PE(AR)^2, as blacklist verification could be done in one operation only, except the cost of multiplying a maximum of K tickets together which can be pre-computed. As a result, the computation complexity shifts from K rounds of non-membership proof to a zero-knowledge proof requiring exponentiation of size $O(K \cdot \ell_t)$, where ℓ_t is the size of one ticket.

We further remove the requirement that tickets are added and deleted from the user side storage for *every* authenticated session. Tickets can only be added in the score redemption protocol after the scores associated with some tickets are redeemed (and of course, some new tickets are initially added in the registration protocol). Users can only delete tickets that have been reviewed in the score redemption protocol. Thus the choice of K no longer has impact to the ticket review time of the SP in our new scheme.

Reputation System. Our reputation mechanism associates a score with the credential. For each past session, there is an external mechanism which assigns score to it. The SP can then authenticate this score. The user can redeem scores from multiple sessions together at any desired time. After validity check, the SP refreshes the user's credential with the new score. The user can later authenticate to the SP and possibly access some special services from the SP when the score is above a certain threshold.

Recall that in PEREA, the SP needs to review each authentication in a timely-manner, and the naughtiness assigned to a credential can only be reflected from its K-most recent authentications. It seems that PE(AR)^2 may run into the same problem of requiring the SP to assign score to each session as quickly as possible, since the user can only delete tickets from the credential for those having the associated past sessions reviewed. It might be possible for a user to have all K tickets in the credential pending to be reviewed, and thus can no longer be authenticated for one more session. However, an importance difference between PE(AR)^2 and PEREA is that not only K affordable by our system is much higher than that of PEREA (which will be demonstrated experimentally in Section 5), but it is just a recommended size instead of a hard limit. A very active user can always store more than K pending tickets in the credential. True, this credential may stand out in the system since the size of ZKPoK must be larger, but the compromise in privacy is minimal since every other attributes (like the linkage of this credential with the past sessions) are hidden. On the other hand, one may consider it as a feature providing some sort of soft rate-limiting.

4.2 Construction

First we establish some notations and convention.

Given a security parameter ℓ, let $\ell_N, \ell_t, \ell_s, \ell_e, K$ be the system parameters. The former four are security parameters.

1. The first one is for the RSA modulus. A typical setting could be $(\ell_N, \ell) = (1024, 160)$.
2. The parameter ℓ_t determines the ticket length, which is also the length of a session identifier. User can pick tickets randomly from a set of ℓ_t-bits prime numbers, denoted as Π_{ℓ_t}. When $\ell_t = 166$, there are at least 2^{160} tickets in this set[4]. For a reasonably large number of tickets randomly picked from Π_{ℓ_t}, probability of having two of them collide is approximately 2^{-80}, due to the birthday paradox.
3. K represents the number of tickets that should be generated during registration, and a typical value could be $K = 1000$.
4. Finally, ℓ_s and ℓ_e determine the domain sizes for the components in the CL signature scheme. For its security we require $\ell_s = \ell_N + \ell_t + \ell$, $\ell_e > \ell_t + 2$.

We use the notation PK with different subscripts to refer to different ZKPoK. Their instantiations are standard and will be described in the full version of this paper. In particular, we use the ZK proof of CL signatures [11] during authentication, and the reputation system is realized by the sum of committed values [5].

KeyGen: The service provider $\mathcal{S}$ generates the keys as follows:

1. $\mathcal{S}$ chooses an ℓ_N-bit safe-prime product $N = pq$ as a special RSA modulus, where p and q are random safe primes.
2. $\mathcal{S}$ chooses $g, h, g_{\mathtt{Acc}}, g_{\mathtt{CL}} \in \mathbb{QR}_N$, the set of quadratic residue modulo N.
3. $\mathcal{S}$ sets $\mathsf{pk}_{\mathtt{CL}} = (N, g, h, g_{\mathtt{CL}})$ and $\mathsf{sk}_{\mathtt{CL}} = \phi(N)$, $\mathsf{pk}_{\mathtt{Acc}} = (N, g, h, g_{\mathtt{Acc}})$ and $\mathsf{sk}_{\mathtt{Acc}} = \phi(N)$, i.e., $g_{\mathtt{Acc}}$ and $g_{\mathtt{CL}}$ are the exponentiation base used in accumulator and CL-signature respectively.
4. $\mathcal{S}$ maintains and publishes a list $\mathbb{L}$ containing pairs of ticket (or session identifier) and score.
5. $\mathcal{S}$ maintains a public blacklist $\mathbb{B}$, and the corresponding accumulator value c, initialized to $g_{\mathtt{Acc}}$.
6. $\mathcal{S}$ runs $\mathsf{K}(1^\ell)$ for the ZKPoK protocols PK_1, PK_2 and PK_3 (which will be defined below). Denote all the common reference strings generated as crs.
7. Finally $\mathcal{S}$ sets $\mathsf{msk} = \phi(N)$ and $\mathsf{mpk} = (\mathsf{crs}, N, g, h, g_{\mathtt{Acc}}, g_{\mathtt{CL}}, \hat{t}, c)$.

In practice, $\mathcal{S}$ may split the public lists $\mathbb{L}$ and $\mathbb{B}$ into smaller lists for different time periods so users could store and download the smaller set of lists. We omit such construction for simplicity.

Reg: Assuming a user $\mathcal{U}$ and a service provider $\mathcal{S}$ has established a pre-authenticated channel via other means, $\mathcal{U}$ obtains a credential from $\mathcal{S}$ as follows:

1. $\mathcal{U}$ sets ticket $t_0 = \hat{t}$ and picks K tickets $t_i \in_R \Pi_{\ell_t} \setminus \mathbb{B}$ where $0 < i \leq K$.
2. $\mathcal{U}$ sets $T = \prod_{i=0}^{K} t_i$, and prepares its commitment.

[4] This follows from a result of Dusart [12]: the number of distinct primes less than x is larger than $\frac{x}{\ln x}(1 + \frac{0.992}{\ln x})$ for all $x > 598$.

3. $\mathcal{U}$ prepares t_1's commitment.
4. $\mathcal{U}$ computes $w = \mathsf{Compute}_{\mathtt{nmw}}(T, \mathbb{B}, \mathsf{pk}_{\mathtt{Acc}})$ and its commitment.
5. $\mathcal{U}$ sets the initial score $s = 0$, and prepares its commitment.
6. $\mathcal{U}$ sends $\mathcal{S}$ the ZKPoK:

$$PK_1\{(t_0, t_1, T, w, s) : t_0 t_1 | T \wedge 1 = \mathsf{Verify}_{\mathtt{nmw}}(c, T, w, \mathsf{pk}_{\mathtt{Acc}}) \wedge t_0 = \hat{t} \wedge s = 0\}.$$

   It proves that 1) t_0 and t_1 are stored in T, 2) w is a non-membership witness that no ticket in T is in the blacklist accumulator value c, and 3) the initial score s is 0.
7. If $\mathcal{S}$ accepts the ZKPoK, he runs $\sigma \leftarrow \mathsf{Sign}_{\mathtt{CL}}((T, t_1, s), \mathsf{sk}_{\mathtt{CL}})$ with $\mathcal{U}$, by signing on the commitments of T, t_1 and s. As a result, $\mathcal{U}$ obtains σ from $\mathcal{S}$.
8. $\mathcal{U}$ sets $\mathbb{T} = \{t_0, \ldots, t_K\}$, the auxiliary information $\mathbb{J} = (\{g^{T/t_i} : t_i \in \mathbb{T}\}, T, g^T, g^T_{\mathtt{Acc}})$, and stores the credential $\mathsf{cred}_0 = (0, \sigma, w, c, s, \mathbb{T}, \mathbb{B}, \mathbb{J})$.

We will then use the notation c_{cred} and $\mathbb{B}_{\mathsf{cred}}$ to denote the accumulator values stored by $\mathcal{U}$ with the credential, which are the current accumulator value and the current blacklist at the time of ticket generation, and may not be as up-to-date as the public values maintained by $\mathcal{S}$.

Auth: On the i-th round of authentication, where $1 \leq i < K$, the user $\mathcal{U}$ is in possession of $\mathsf{cred}_{i-1} = (i-1, \sigma, w, c_{\mathsf{cred}}, s, \mathbb{T}, \mathbb{B}_{\mathsf{cred}}, \mathbb{J})$. $\mathcal{U}$ authenticates anonymously with $\mathcal{S}$ and obtains a new credential as follows:

1. $\mathcal{U}$ obtains the current blacklist $\mathbb{B}$ and the corresponding accumulator c via a public channel.
2. $\mathcal{U}$ updates cred_{i-1} with an updated witness $w' = \mathsf{Update}_{\mathtt{nmw}}(w, c_{\mathsf{cred}}, T, \{\hat{t}_j\})$, where $\hat{t}_j \in \mathbb{B} \setminus \mathbb{B}_{\mathsf{cred}}$.
3. $\mathcal{U}$ sends $\mathcal{S}$ the ZKPoK:

$$PK_2\{(t_i, t_{i+1}, T, \sigma, w', s) : \quad s > s_{\mathsf{base}} \quad \wedge \quad t_{i+1} | T \\ \wedge \quad 1 = \mathsf{Verify}_{\mathtt{nmw}}(c, T, w', \mathsf{pk}_{\mathtt{Acc}}) \quad \wedge \quad 1 = \mathsf{Verify}_{\mathtt{CL}}((T, t_i, s), \sigma, \mathsf{pk}_{\mathtt{CL}})\}.$$

   It proves that 1) $\mathcal{U}$'s score s is high enough when $\mathcal{S}$ expects a base score s_{base}, 2) t_{i+1} is stored in T, 3) w' is a non-membership witness that none of the (previous) sessions in T is in the blacklist accumulator value c, and 4) the tuple (T, t_i, s) is signed by $\mathcal{S}$ in the previous session.
4. If $\mathcal{S}$ accepts the ZKPoK, $\mathcal{U}$ opens the commitment of t_i to $\mathcal{S}$.
5. $\mathcal{S}$ aborts if t_i has been stored on $\mathbb{L}$, the list of used tickets, as an old ticket cannot be reused again.
6. Otherwise, $\mathcal{S}$ authenticates $\mathcal{U}$, appends $(t_i, 0, \perp)$ to $\mathbb{L}$, and executes $\sigma' \leftarrow \mathsf{Sign}_{\mathtt{CL}}((T, t_{i+1}, s), \mathsf{sk}_{\mathtt{CL}})$ with $\mathcal{U}$.
7. $\mathcal{U}$ obtains σ' and stores credentials $\mathsf{cred}_i = (i, \sigma', w', c, s, \mathbb{T}, \mathbb{B}, \mathbb{J})$.

Revoke: $\mathcal{S}$ can revoke a previously authenticated session based on the behavior observed for that session. To revoke a session $t \in \mathbb{L}$, $\mathcal{S}$ adds t to $\mathbb{B}$ and accumulates it by setting $c \leftarrow c^t$. $\mathcal{S}$ publishes $\mathbb{B}$ and c via a public channel[5]. The corresponding $(t, \cdot, \cdot)$ entry should then be removed from $\mathbb{L}$.

[5] Multiple tickets can be revoked at once by setting $c \leftarrow c^{\hat{T}}$, where $\hat{T}$ is the product of the tickets to be revoked.

Rate: $\mathcal{S}$ can rate a previously authenticated session based on the behavior observed. To rate a session $t \in \mathbb{L}$ with a score s, $\mathcal{S}$ computes $\sigma_t \leftarrow \mathsf{Sign}_{\mathtt{CL}}((t, s), \mathsf{sk}_{\mathtt{CL}})$ and updates $(t, 0, \perp)$ to (t, s, σ_t) in the list $\mathbb{L}$.

Redeem: After the i-th round of authentication, $\mathcal{U}$ attempts to use $\mathsf{cred}_i = (i, \sigma, w, c_{\mathsf{cred}}, s, \mathbb{T}, \mathbb{B}_{\mathsf{cred}}, \mathbb{J})$ to exchange for a new credential with the updated score s' due to a list of redeemable tickets $\mathbb{T}_{\mathsf{old}}$. The protocol runs as follows:

1. $\mathcal{U}$ retrieves the scores of the tickets that he wants to redeem and the associated signatures: $\mathbb{S} = \{(t_j, s_j, \sigma_j) \in \mathbb{L} : t_j \in \mathbb{T}_{\mathsf{old}} \wedge \sigma_j \neq \perp\}$.
2. $\mathcal{U}$ removes t_j from $\mathbb{T}_{\mathsf{old}}$ for every j such that $(t_j, s_j, \perp) \in \mathbb{L}$.
3. $\mathcal{U}$ forms $\mathbb{T}_{\mathsf{new}}$ by picking n random new tickets from Π_{ℓ_t}, where n is the size of $\mathbb{T}_{\mathsf{old}}$.
4. $\mathcal{U}$ sets $T_{\mathsf{old}} = \prod t'_j$ where $t'_j \in \mathbb{T}_{\mathsf{old}}$ and $T_{\mathsf{new}} = \prod \hat{t}_j$ where $\hat{t}_j \in \mathbb{T}_{\mathsf{new}}$.
5. $\mathcal{U}$ sends $\mathcal{S}$ the ZKPoK: (where $t_{i+1} \in \mathbb{T}$ and $T \in \mathbb{J}$)

$$PK_3\{(t_{i+1}, T, T_{\mathsf{old}}, T_{\mathsf{new}}, \sigma, s, s', \mathbb{S}) : \quad T_{\mathsf{old}} | T \quad \wedge \quad s' = s + \sum_{(\cdot, s_j, \cdot) \in \mathbb{S}} s_j$$

$$\bigwedge_{(t_j, s_j, \sigma_j) \in \mathbb{S}} 1 = \mathsf{Verify}_{\mathtt{CL}}((t_j, s_j), \sigma_j, \mathsf{pk}_{\mathtt{CL}}) \wedge 1 = \mathsf{Verify}_{\mathtt{CL}}((T, t_{i+1}, s), \sigma, \mathsf{pk}_{\mathtt{Acc}})\}.$$

   It proves that 1) the tickets to be redeemed T_{old} are stored in T, 2) s' is the summation of the old score and the new scores to be redeemed, 3) the score s_j of each ticket t_j is signed by $\mathcal{S}$, and 4) the tuple (T, t_{i+1}, s) is signed by $\mathcal{S}$ in the previous session.
6. If $\mathcal{S}$ accepts the ZKPoK, $\mathcal{U}$ opens t_i from its commitment.
7. $\mathcal{U}$ computes $w' = \mathsf{Compute}_{\mathtt{nmw}}(T \cdot T_{\mathsf{new}}/T_{\mathsf{old}}, \mathbb{B}, \mathsf{pk}_{\mathtt{Acc}})$.
8. $\mathcal{S}$ runs $\sigma' \leftarrow \mathsf{Sign}_{\mathtt{CL}}((T \cdot T_{\mathsf{new}}/T_{\mathsf{old}}, t_i, s'), \mathsf{sk}_{\mathtt{CL}})$ with $\mathcal{U}$.
9. $\mathcal{U}$ obtains σ' from $\mathcal{S}$, recalculates auxiliary information $\mathbb{J}'$ using the new tickets, and stores credential $\mathsf{cred}'_{i+1} = (i+1, \sigma', w', c, s', \{\mathbb{T} \backslash \mathbb{T}_{\mathsf{old}} \cup \mathbb{T}_{\mathsf{new}}\}, \mathbb{B}, \mathbb{J}')$.

Theorem 1. *Our scheme is accountable and private if PK_1, PK_2, PK_3 are ZKPoK, the underlying CL signature scheme and the accumulator system are secure.*

The proof is given in the full version of the paper.

5 Discussions

5.1 Complexity Analysis

Here we present a computation analysis on PE(AR)^2, PEREA [5] and BLACR-Express [10] based on the number of expensive operations, in terms of K (size of user ticket queue), δ_L (number of tickets added to blacklist after the last time when a user retrieved the updated credential) and δ_R (number of tickets to redeem). We also included the Redeem protocol for PE(AR)^2 in the comparison

Table 1. Performance analysis in authentication phase

Scheme	Communication		Computation	
	Downlink	Uplink♠	User	Service Provider
PE(AR)^2				
Auth	$O(\delta_L)$	$O(K)$	1Eu♡ + 14$\mathrm{E_m}$	14$\mathrm{E_m}$
Redeem	$O(\delta_L + \delta_R)$	$O(K + \delta_R)$	1Eu♣ + $(6 + 6\delta_R)\mathrm{E_m}$	$(6 + 6\delta_R)\mathrm{E_m}$
PEREA [5]	$O(\delta_L)$	$O(K)$	$[(K+1)\delta_L + 5K + 2\left\lceil \frac{K+1}{3} \right\rceil + \left\lceil \frac{K-1}{3} \right\rceil + 3]\mathrm{E_m}$	$(4K + 2\left\lfloor \frac{K+1}{3} \right\rfloor + 3)\mathrm{E_m}$
BLACR [10] (Express)	$O(\delta_L)$	$O(\delta_L)$	$(30\delta_L + 81)\mathrm{E_1}$◇ + $(5\delta_L + 7)\mathrm{E_T}$ + $(\delta_L + 4)\mathrm{P}$	$(12\delta_L + 26)\mathrm{E_1} + (2\delta_L + 6)\mathrm{E_2}$ + $(5\delta_L + 18)\mathrm{E_T}$ + $(\delta_L + 3)\mathrm{P}$

♠ For PE(AR)^2, $O(K)$-size is due to the product of K tickets, which should be smaller than the $O(K)$ non-membership proof required in PEREA.

♡ Let $h_1 = \min(\delta_L, K)$. The Euclidean algorithm here require $O(h_1)$ division operations. The performance can be further improved as shown in [13].

♣ Let $h_2 = \min(L, K)$. The Euclidean algorithm here is running $O(h_2)$ division operations.

◇ For simplicity, we assume the fraction of tickets that belong to the user is close to zero when there are a lots of tickets in the system, and $\ell = m = 1$ for BLACR-Express.

since it is expected that a user would execute Redeem for once after K invocations of Auth. In the worst case, users could run Redeem after every single instance of Auth.

Table 1 outlines the analysis on computation and communication. Denote $\mathrm{E_m}$ as the multi-based modular exponentiation that 3 exponentiations could be done simultaneously, Eu as the extended Euclidean algorithm, $\mathrm{E_1}$, $\mathrm{E_2}$, $\mathrm{E_T}$ as the exponentiation of the pairing group $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ respectively, and P as the pairing operation. Only BLACR uses $\mathrm{E_1}$, $\mathrm{E_2}$, $\mathrm{E_T}$ and P.

Notice that the main bottleneck of these schemes are on the SP side, PE(AR)^2 has time complexities of $O(1)$ and $O(\delta_R)$ in Auth and Redeem phase respectively. For PEREA and BLACR-Express, authentication requires $O(K)$ and $O(\delta_L)$ witness verification on SP side respectively. On user side, PE(AR)^2 has time complexities of $O(1)$ and $O(\delta_R)$ multi-base exponentiation $\mathrm{E_m}$ in Auth and Redeem phase respectively. PEREA and BLACR-Express run in $O(K\delta_L)$ and $O(\delta_L)$ respectively on the user side[6]. We remark that PE(AR)^2 uses Eu, with time complexity dominated by the division operation, and the number of division involved is related to L, K, and δ_L. However, the division operation is much more efficient than exponentiation ($\mathrm{E_m}$, $\mathrm{E_1}$, $\mathrm{E_2}$, $\mathrm{E_T}$) and pairing P.

5.2 Empirical Result

We benchmarked the time required for authentication on PEREA and BLACR-Express, and both authentication and redemption for PE(AR)^2, for different K, on both SP and user side. We obtained the benchmark from a 2.4GHz Intel Core i5 box with 4GB memory. The result is visualized in Figure 1. We marked the y-axis in *logarithmic scale* so as to capture the benchmark of different magnitude.

[6] Here we use the BLACR-Express data without pre-computation. It was claimed that the dependence of δ_L for $\mathrm{E_m}$ and P can be removed without details [10], hence we cannot perform benchmark on them here.

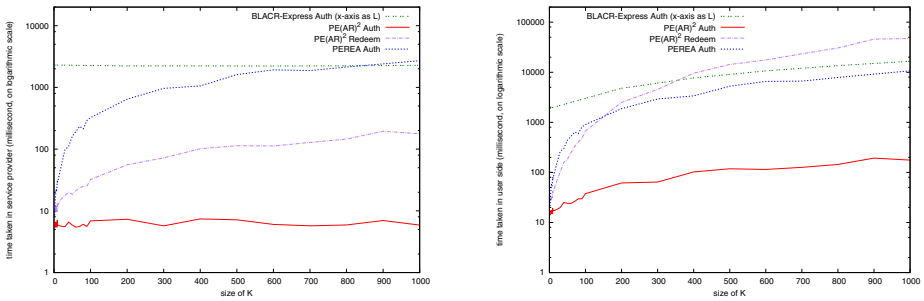

Fig. 1. Benchmark for time taken in service provider and user side

The benchmarks are obtained from averaging the result with repetitive experiments. For PEREA, our benchmark aligns with the result in [5]. The SP takes around 20ms for authentication when $K = 5$, and client takes around 50ms. As K increases to 1000, the SP takes 2758ms and client takes around 10672ms. For BLACR-Express with $\delta_L = 2\%$ of L, SP authentication takes around 2228 ms when $L = 200$ and 2281ms when $L = 1000$. On the other hand, client authentication takes around 4775ms when $L = 200$ and 16539ms when $L = 1000$. For PE(AR)^2, the SP takes 6ms and 10ms for Auth and Redeem respectively when $K = 5$, 6ms and 177ms when $K = 1000$. Client takes 15ms and 28ms for Auth and Redeem respectively when $K = 5$, 176ms and 47236ms when $K = 1000$.

Recall the fact that the bottleneck of the scheme should be on the SP side, in particular on authentication part. We look at the traffic statistics of 2channel, one of the most popular internet forum in Japan. They have an average of 2.5 million posts made every day[7], which is about 1736 posts per minute. Our PE(AR)^2 on the SP side can handle 10000 posts per minute[8], and hence can handle real traffic for popular internet forums.

5.3 Practical Considerations

Reducing the computation of the Service Provider. Authentication should be made efficient for the SP, since the SP may need to handle multiple user authentication requests simultaneously in real time. PE(AR)^2 relaxed the SP computation time requirement as compared with PEREA, such that only one signature verification and witness verification is required. This improvement is done by condensing the witnesses of multiple tickets into one single witness, and the actual computation is shifted from Auth phase to Redeem phase.

[7] `http://stats.2ch.net/suzume.cgi?yes`, in Japanese

[8] BLACR-Express (with pre-computation) is setup to support authentications at the rate of about 25 authentications/minute for active users and about 1 authentication/minute for inactive users [10], based on the traffic of Wikipedia. It is much less than the requirements for popular internet forums.

Pre-computation of Auxiliary Information. As the value to be proved changes from a single ticket to a product of ticket, the proof size can be larger. We thus suggest the use of auxiliary information gathered from Reg and Redeem phase. With terms like g^T, $g^T_{\mathtt{Acc}}$ and g^{T/t_i} precomputed and stored as auxiliary information in $\mathbb{J}$, the zero-knowledge proof (PK_2) in the Auth phase can be computed efficiently. For example, CL signature verification requires the computation of g^T, non-membership verification requires the computation of $(z_d)^T$ where $z_d = dg^\rho$, with a random ρ and $d = g^{-b}_{\mathtt{Acc}}$. Thus $(z_d)^T = (g^T)^\rho (g^T_{\mathtt{Acc}})^{-b}$ and can be computed efficiently given the auxiliary information $\mathbb{J}$.

Relaxing Mandatory Timely Blacklist Publishing. PE(AR)^2 does not require the SP to publish timely on the tickets that should be blacklisted, as a user could only shift a ticket out of the queue after it is rated. Thus an SP have a higher flexibility in resource management regarding when to review the authenticated sessions, mark them as blacklisted, or give a score rating.

Rate-Limiting Considerations. Our scheme does not require rate-limiting facility. The term rate-limiting refers to the ability to stop abuser from overusing the service over a period of time. In PEREA, rate-limiting is mandatory as an adversary could generate a lot of authentication sessions to shift any potential blacklist session out of the ticket queue before the SP would blacklist them. However, rate-limiting is not provided in the scope of PEREA and could only be integrated using other cryptographic techniques (e.g., see [14]). Our scheme does not have such weakness as the removal of any ticket is checked by the SP in Redeem phase. Our scheme could actually be modified to provide a less-efficient construction for rate-limiting by default as follows. Recall that in Reg and Redeem protocols, a user may attempt to generate more than K number of tickets in the ticket queue. We could modify our construction to have the user to provide zero knowledge proof on the number of tickets generated. Then, a user could no longer authenticate once all the tickets in the queue are used up, and must wait until the SP to participate in Redeem phase.

Types of Scores. Our current construction only considers positive scores to provide rewards for good behavior. It is suitable for a lot of popular Internet sites that use positive scores only, like Facebook (Like) and Google Plus (+1).

Nevertheless, we can extend it to support negative scores like BLACR [10]. For the security model, we need to give a new model which prevents an adversary from not redeeming the negative scores. For the construction, we have to remove the unlinkability of the ticket redeemed in the Redeem protocol, by removing $\mathbb{S}$ from the PK_3. Therefore, the SP can check whether a ticket is redeemed or not. If a ticket with a negative score is not redeemed after a long period of time, the SP can simply add it to the blacklist. Hence, its owner is forced to ask the SP to un-blacklist it by redeeming it, before any further authentication.

6 Conclusion

We presented PE(AR)^2, which preserves the same anonymous authentication functionalities as PEREA, but fixes its vulnerability and incorporates reputation

system. PE(AR)^2 also eliminated the complexity dependence on the ticket queue size for the SP. The improvement is significant for an active system with many sessions and gives more flexibility to the SP for tickets review. Furthermore, PE(AR)^2 avoids the reliance of additional anonymous rate-limiting protocol, which was critical to the operation of PEREA for prevent user from bypassing the misbehavior detection. We believe that PE(AR)^2 is more practical for real world use. We presented the construction of our design. We give a brief analysis of the complexity and benchmarks, justifying that PE(AR)^2 is efficient for real-world deployment.

References

1. Chow, S.S.M., He, Y.-J., Hui, L.C.K., Yiu, S.M.: SPICE – Simple Privacy-Preserving Identity-Management for Cloud Environment. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 526–543. Springer, Heidelberg (2012)
2. Camenisch, J., Kohlweiss, M., Soriente, C.: An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
3. Tsang, P.P., Au, M.H., Kapadia, A., Smith, S.W.: BLAC: Revoking repeatedly misbehaving anonymous users without relying on TTPs. ACM Trans. Inf. Syst. Secur. 13(4), 39 (2010)
4. Brickell, E., Li, J.: Enhanced Privacy ID: A Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities. In: WPES, pp. 21–30. ACM (2007)
5. Au, M.H., Tsang, P.P., Kapadia, A.: PEREA: Practical TTP-free revocation of repeatedly misbehaving anonymous users. ACM Trans. Inf. Syst. Secur. 14(4), 29 (2011)
6. Au, M.H., Susilo, W., Mu, Y., Chow, S.S.M.: Constant-size dynamic k-times anonymous authentication. IEEE Systems Journal (to appear)
7. Peng, K., Bao, F.: Vulnerability of a Non-membership Proof Scheme. In: SECRYPT, pp. 419–422. SciTePress (2010)
8. Li, J., Li, N., Xue, R.: Universal Accumulators with Efficient Nonmembership Proofs. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 253–269. Springer, Heidelberg (2007)
9. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decis. Support Syst. 43(2), 618–644 (2007)
10. Au, M.H., Kapadia, A., Susilo, W.: BLACR: TTP-Free Blacklistable Anonymous Credentials with Reputation. In: NDSS. The Internet Society (2012)
11. Camenisch, J.L., Lysyanskaya, A.: A Signature Scheme with Efficient Protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
12. Dusart, P.: The k^{th} prime is greater than $k(\ln k + \ln \ln k - 1)$ for $k \geq 2$. Math. Comput. 68(225), 411–415 (1999)
13. Möller, N.: On schönhage's algorithm and subquadratic integer gcd computation. Math. Comput. 77(261), 589–607 (2008)
14. Chow, S.S.M.: Real Traceable Signatures. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 92–107. Springer, Heidelberg (2009)