2012

BLACR: TTP-free blacklistable anonymous credentials with reputation

Man Ho Au
University of Wollongong, aau@uow.edu.au

Apu Kapadia
Dartmouth College, Hanover, NH

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Publication Details

Au, M., Kapadia, A. & Susilo, W. (2012). BLACR: TTP-free blacklistable anonymous credentials with reputation. NDSS Symposium 2012: 19th Network & Distributed System Security Symposium (pp. 1-17). USA: Internet Society.

BLACR: TTP-free blacklistable anonymous credentials with reputation

Abstract
Anonymous authentication can give users the license to misbehave since there is no fear of retribution. As a deterrent, or means to revocation, various schemes for accountable anonymity feature some kind of (possibly distributed) trusted third party (TTP) with the power to identify or link misbehaving users. Recently, schemes such as BLAC and PEREA showed how anonymous revocation can be achieved without such TTPs—anonymous users can be revoked if they misbehave, and yet nobody can identify or link such users cryptographically. Despite being the state of the art in anonymous revocation, these schemes allow only a basic form of revocation amounting to 'revoke anybody with d or more misbehaviors' or 'revoke anybody whose combined misbehavior score is too high' (where misbehaviors are assigned a 'severity' score). We present BLACR, which significantly advances anonymous revocation in three ways: 1) It constitutes a first attempt to generalize reputation-based anonymous revocation, where negative or positive scores can be assigned to anonymous sessions across multiple categories. Servers can block users based on policies, which specify a boolean combination of reputations in these categories; 2) We present a weighted extension, which allows the total severity score to ramp up for multiple misbehaviors by the same user; and, 3) We make a significant improvement in authentication times through a technique we call express lane authentication, which makes reputation-based anonymous revocation practical.

Keywords
blacklistable, blacr, ttp, credentials, free, anonymous, reputation

Disciplines
Physical Sciences and Mathematics

Publication Details
Au, M., Kapadia, A. & Susilo, W. (2012). BLACR: TTP-free blacklistable anonymous credentials with reputation. NDSS Symposium 2012: 19th Network & Distributed System Security Symposium (pp. 1-17). USA: Internet Society.

BLACR: TTP-Free Blacklistable Anonymous Credentials with Reputation

Man Ho Au,[†] Apu Kapadia,[‡] Willy Susilo[†]

[†]Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong
Wollongong, Australia
{aau, wsusilo}@uow.edu.au

[‡]School of Informatics and Computing
Indiana University Bloomington
Bloomington, IN, USA
kapadia@indiana.edu

Abstract

Anonymous authentication can give users the license to misbehave since there is no fear of retribution. As a deterrent, or means to revocation, various schemes for accountable anonymity feature some kind of (possibly distributed) trusted third party (TTP) with the power to identify or link misbehaving users. Recently, schemes such as BLAC and PEREA showed how anonymous revocation *can be achieved without such TTPs—anonymous users can be revoked if they misbehave, and yet nobody can identify or link such users cryptographically. Despite being the state of the art in anonymous revocation, these schemes allow only a basic form of revocation amounting to 'revoke anybody with d or more misbehaviors' or 'revoke anybody whose combined misbehavior score is too high' (where misbehaviors are assigned a 'severity' score).*

We present BLACR, which significantly advances anonymous revocation in three ways: 1) It constitutes a first attempt to generalize reputation-based anonymous revocation*, where negative or positive scores can be assigned to anonymous sessions across multiple categories. Servers can block users based on policies, which specify a boolean combination of reputations in these categories; 2) We present a weighted extension, which allows the total severity score to ramp up for multiple misbehaviors by the same user; and, 3) We make a significant improvement in authentication times through a technique we call* express lane authentication*, which makes reputation-based anonymous revocation practical.*

1. Introduction

Anonymity can give some users the license to misbehave. For example, by using an anonymizing network like Tor [16] a vandal may connect to Wikipedia and deface a webpage, or may post copyrighted material to Youtube. To tackle such misbehaving users several schemes have been proposed that strike different tradeoffs between privacy and accountability. For example, anonymous credential schemes allow users to authenticate to a service provider (SP) as 'some anonymous member in a group' to prove they belong to some class of users (e.g., students at University X). If a member within the group misbehaves, many schemes allow the SP to complain to a trusted third party (TTP) (or a distributed TTP) and either identify the user, link the user's accesses, or simply revoke the user's ability to authenticate in the future. Such schemes include those based on group signatures [1, 6, 15, 21], dynamic accumulators [2, 7, 10, 22, 25], and Nymble systems [20, 33, 19, 23, 27].

Having a TTP capable of deanonymizing or linking a user's accesses is dangerous. Such TTPs must be trusted to handle complaints by the SP fairly, and users can never be certain whether their accesses will remain private. Such TTPs will thus discourage several legitimate uses of anonymity such as activists posting material from countries with restricted freedoms, whistleblowers from posting material to sites such as Wikileaks, and so on. Recognizing the need to eliminate such TTPs, a few schemes have been proposed such as BLAC [30, 32], EPID [8], and

PEREA [31, 4]. These schemes allow *anonymous revocation*, where misbehaving users can be revoked without the involvement of a TTP. BLAC, EPID, and PEREA allow SPs to blacklist previous sessions so that offending users cannot authenticate in the future if *any* of their previous sessions has been blacklisted. Additionally, BLAC supports d-strikes-out policies [32], e.g., $d = 3$ enacts a 'three strikes out policy' where three (or more) misbehaviors from a user results in revocation of that user. PEREA generalizes d-strikes-out policies to a weighted version called *naughtiness* policies [4]. Each misbehavior is assigned a *severity*, and users whose total severity (*naughtiness*) exceeds a certain naughtiness threshold are denied authentication. In all cases, the SP learns only whether an authentication for the anonymous user succeeds or not. Thus there is no entity who can deanonymize or link a user's anonymous authentications, but yet SPs can prevent misbehaving users from returning. Users are guaranteed strong privacy (they can never be deanonymized), and SPs are spared from future accesses by such users.

A major drawback of PEREA is that naughtiness can be computed only over a short *revocation window* of the most recent K authentications of a user, where typically K ranges from 5–15. If a session is not blacklisted within this short window, then that misbehavior is automatically forgiven. In contrast, BLAC faces no such limitation and can incorporate misbehaviors performed arbitrarily in the past. In this paper we focus on improving BLAC given the stronger revocation semantics of BLAC. We note that BLAC lacks the naughtiness functionality of PEREA, and we extend BLAC to support such functionality and much more.

Our contributions We make the first significant effort to extend TTP-free anonymous revocation with general behavior-based policies, giving SPs a language to characterize (un)acceptable uses of their services while supporting anonymous revocation. We make the following contributions:

- We generalize the concept of anonymous revocation to *reputation-based anonymous revocation*. SPs can score positive and negative behaviors of users across various categories (resulting in reputation scores for each category), and then use a boolean policy language to express acceptable behaviors across categories. We name our construction BLACR (BLacklistable Anonymous Credentials with Reputation), which significantly extends the original construction of BLAC.
- We detail a novel weighted extension to BLACR, which allows SPs to penalize or reward *repeated* bad or good behaviors from the same anonymous user through a weighted function. For example, the severity of a misbehavior can be doubled if it is the third or more offense by a user. Providing this functionality while maintaining full anonymity during authentication is non trivial, because the new reputation must be computed based on the express lane token and the current blacklist with the weighted factors applied appropriately in zero knowledge. Thus this weighted extension is another major contribution of our work.
- Improving on the linear time complexity (in the size of the blacklist) for authentications in BLAC/EPID remains an open and important problem and currently limits the practicality of such approaches. We detail a novel approach for *express lane authentication*, where users obtain an express token to authenticate faster in the next time period. This technique allows for greatly reduced authentication times for active users, which for the first time makes anonymous blacklisting highly practical in systems where active users perform the majority of authentications. Through a quantitative analysis we show BLACR can indeed be used in practical settings to support reputation-based anonymous revocation.

Other schemes that reduce or eliminate TTPs One major class of TTP-free schemes is based on the 'double spending' of e-cash [14] (generalized to n-times spending [29]), where only if a user authenticates twice (or n times) does the user's identity get revealed or linked. Unfortunately, not all misbehaviors (such as defacing a webpage or subtle astroturfing campaigns for spreading disinformation) can be reduced to 'too many authentications.' BLAC, EPID and PEREA thus support *subjective blacklisting*, where SPs can revoke users based on human, subjective assessments of misbehavior by simply blacklisting a user's session.

Recently, Schwartz et al. [27] proposed *contractual anonymity*, where the TTP runs within trusted hardware. Nevertheless, one must trust the hardware and the code, and furthermore it is not feasible for the trusted program/hardware to automatically decide whether a prespecified contract is broken in the case of astroturfing attacks or cases of vandalism. In such cases, again it is necessary for a *human assessment* of whether a misbehavior occurred.

Finally, as also discussed by the authors of BLAC [32], techniques such as UST rely on whitelisting [28] and provide users with the ability to return if their behavior in the previous authentication was deemed good. These schemes require the SP to decide (subjectively) if each session was free of misbehavior while the user is online and are impractical in scenarios where the user has long since logged off. In a significant improvement, FAUST [24] removes the online requirement by allowing users to retrieve their tokens privately at a later time, but still does not provide an adequate solution for reputation-based anonymous revocation.

Their suggestion to use multiple tokens does not offer the important property of *collusion resistance*, and individual revoked users can 'pool in' their available tokens to gain authentication. Furthermore, FAUST does not implement boolean revocation policies like BLACR, and the scoring thresholds must be fixed before issuing credentials. Revocation policies can be updated easily in BLACR.

2. Overview of Approach

Adding "severity" to BLAC BLAC and EPID use the following idea: an anonymous authentication of a user with secret key x results in a *ticket* $\tau = (b, t)$, where $t \leftarrow H(b||\texttt{sid})^x$, for some collision-resistant hash function H and a unique identifier $\texttt{sid}$ for the service provider. The ticket τ is stored by the SP in association with the session (e.g., a video posted by the anonymous user). Since it is computationally hard for the SP to calculate the discrete log x of t, the user's identity remains anonymous to the SP. When an SP wants to *blacklist* a user associated with a session, it inserts the ticket τ for that session into the blacklist $\mathcal{L}$. Users authenticating to the SP must prove their credential is not associated with any ticket on the blacklist. Users prove this in zero knowledge (x is not revealed to the SP) and bound to their issued credentials (users cannot forge their own credential x). It is possible to prove in zero knowledge the "inequality of discrete log", that is, the user's credential x' does not correspond to the discrete log x of each entry t on the blacklist.

BLACR adds a *score* parameter s_i to each entry in the blacklist indicating the *severity* of the misbehavior. Let $\mathcal{L}$ be a list of pairs (τ_i, s_i) for $i = 1$ to $|\mathcal{L}|$. Here τ_i corresponds to a particular ticket, and s_i corresponds to the score associated with that ticket. SPs can require the overall *list score* of an authenticating user satisfy a certain threshold. The list score $\mathsf{S}(\mathcal{L}, x)$ of a user with secret key x with respect to the list $\mathcal{L}$ is the sum of all the scores on the list for tickets corresponding to that user: $\mathsf{S} : (\mathcal{L}, x) \mapsto \sum_{i \in [|\mathcal{L}|], t_i = \mathsf{T}(b_i, x)} s_i$ where $[n]$ denotes the set $\{1, \ldots, n\}$ for any positive integer n.

BLACR augments the proof technique used in BLAC in the following way: For each entry (t_i, b_i, s_i) in the list, the authenticating user creates a commitment C_i and proves to the SP that either (1) $t_i \neq H(b_i||\texttt{sid})^x$ and C_i is a commitment of 0; or (2) $(t_i = H(b_i||\texttt{sid})^x)$ and C_i is a commitment of s_i. Finally, the user proves in zero knowledge to the SP that the sum of the values committed in C_i is above the required threshold.

Generalizing severity to reputation BLACR actually features both positive and negative scores for good and bad behaviors (in a *meritlist* $\mathcal{L}^+$ and *blacklist* $\mathcal{L}^-$ respectively), resulting in an overall *reputation score* for each user $\mathsf{Rep}(\mathcal{L}^+, \mathcal{L}^-, x) \mapsto \mathsf{S}(\mathcal{L}^+, x) - \mathsf{S}(\mathcal{L}^-, x)$, i.e., his/her merit list score minus his/her blacklist score. Furthermore, servers can score reputation across different *categories* $\{c_1, c_2, \ldots, c_m\}$ where m is the number of categories. The reputation of a user with secret key x in category c_i, denoted as R_i, is thus $\mathsf{Rep}(\mathcal{L}_i^+, \mathcal{L}_i^-, x)$. For example, a server may maintain categories for *video content* and *comments*. Within the category of *video content*, egregious copyright violations such as reposting a television episode could be considered to be more severe (and scored appropriately) than a copyright violation by a home-made video with an unlicensed soundtrack. Within the category of *comments*, inappropriate language, racist or intimidating comments could be considered to be more severe than puerile posts containing offensive words. Likewise, content rated highly by other users could result in a commensurate reward (positive score), e.g., comments that have been rated as "helpful" by users could be rewarded. The user's reputation in each category could then be required to be above a certain threshold for authentication to succeed.

Policy based revocation We further allow SPs to specify arbitrary boolean combinations of policies across categories, e.g., users can be allowed access only if their (*video content* reputation is above a certain level) OR (*tagging* reputation is above a certain level AND *commenting* reputation is above a certain level). Here the *tagging* category refers to how well users tag their content with appropriate descriptors. Note that negations are easily supported, because the negation of an atom results in checking the reputation is *above* a certain threshold instead of *below* a certain threshold. In particular, policies are expressed as $\bigvee_{k=1}^{\ell}(\bigwedge_{i=1}^{m}(\neg)\mathcal{P}_{ki})$, which is a combination of conjunctive clause over m categories. Each $\mathcal{P}_{ki}$ have the form (c_i, n_{ki}) which requires the authenticating user to have a reputation equal or higher than a threshold n_{ki} in category c_i. The policy can contain negations too. The negation $\neg\mathcal{P}_{ki}$ requires the authenticating user to have a reputation lower than the threshold n_{ki} in category c_i. $\mathcal{P}_{ji}$ is $\perp$ if the j-th conjunctive clause of the policy does not involve category c_i. Each $\mathcal{P}_{ki}$ is called a sub-policy, and is a boolean function defined over a pair of lists $(\mathcal{L}_i^+, \mathcal{L}_i^-)$ regarding category c_i, a user secret x and a threshold n_{ki}. Its truth value evaluates to 1 if the user reputation $\mathsf{Rep}(\mathcal{L}_i^+, \mathcal{L}_i^-, x) \geq n_{ji}$ and 0 otherwise.

If c_1, c_2, c_3 represents respectively the categories for *video content*, *tagging* and *commenting*, the requirement above can be parsed as $(R_1 \geq n_1) \vee (R_2 \geq n_2 \wedge R_3 \geq n_3)$ where n_1, n_2, n_3 are the required thresholds. In this case $\mathcal{P}_{11} = (c_1, n_1)$, $\mathcal{P}_{22} = (c_2, n_2)$ and $\mathcal{P}_{23} = (c_3, n_3)$ and all others $\mathcal{P}_{ij} = \perp$. Note that we do not require full disjunctive normal form, meaning that the same sub-policy may appear more than once in different conjunctive clauses. As another example, we can set $\mathcal{P}_{12} = \mathcal{P}_{22}$ and the following policy

$(\mathcal{P}_{11} \wedge \mathcal{P}_{12}) \vee (\neg\mathcal{P}_{22} \wedge \mathcal{P}_{23})$ means that any user can enjoy the service if his/her reputation in both categories *video content* and *tagging* is high enough, or if his/her reputation in *video content* is below the threshold, he/she has to have a high reputation in *commenting*.

A "weighted" extension to BLACR Consider the case where a user's misbehaviors for a certain category have been scored as 2, 3, and 2 as ordered by the time of the user's session (note the SP doesn't know these correspond to the same user). In some cases an SP may want to ramp up the penalty for multiple misbehaviors with some multiplicative factor. For example, the SP may want to double the score of the second misbehavior to 6, and triple the score of the third misbehavior to 6, thus disincentivizing repeated misbehaviors. We also note the SP can do the same for the reputation lists, rewarding multiple good behaviors. The SP may even choose to reward multiple good behaviors *less* to provide users with diminishing returns, further incentivizing more good behaviors.

Thus in "BLACR-Weighted", for each category the SP can specify a set of adjusting factors $\mathcal{D} = \{\Delta_1, \Delta_2, \ldots, \Delta_k\}$ so that the *weighted list score* of a user with respect to a list is the sum of the scores where the score of the i-th instance when the authenticating user is put on the list is multiplied by Δ_i. For example, suppose a blacklist of a category is $\{(\tau_1, s_1), (\tau_2, s_2), \ldots, (\tau_8, s_8)\}$ corresponding to 8 sessions. For a certain authenticating user Alice, tickets τ_1 and τ_4 belong to her. Her score with respect to the list in BLACR-Unweighted is thus $s_1 + s_4$. Now suppose the SP publishes the adjusting factors $\{\Delta_1, \Delta_2, \Delta_3, \Delta_4, \Delta_5\}$ for the list. In BLACR-Weighted, the list score of Alice is $\Delta_1 s_1 + \Delta_2 s_4$ since Alice is put on the blacklist for a second time in the entry (τ_4, s_4).

Formally, for a list $\mathcal{L}$, a set of adjusting factors $\mathcal{D}$ and a secret key x, the *weighted score* function is:

$$\mathsf{S}' : (\mathcal{L}, x, \mathcal{D}) \mapsto \sum_{i \in [|\mathcal{L}|], t_i = \mathsf{T}(b_i, x)} \Delta_{|\{j : j \leq i \wedge \mathsf{T}(b_j, x) = t_j\}|} s_i$$

where Δ_i is defined to be Δ_k for all $i > k$. This function states the weights must be applied to the tickets corresponding to the user in correct order, and then the weighted scores are added up to get the weighted list score for that list. For notational convenience, we define the reputation function in this scenario as $\mathsf{Rep}'(\mathcal{L}_i^+, \mathcal{L}_i^-, x) \mapsto \mathsf{S}'(\mathcal{L}_i^+, x) - \mathsf{S}'(\mathcal{L}_i^-, x)$

This extension is non-trivial because a user must prove in zero knowledge that all the tickets corresponding to him/her have the correct factors applied in correct order. Thus this extended construction of BLACR-Weighted is another significant contribution of our work.

Express-lane authentication: a novel ∂-Approach The construction of the above proof requires computation and transmission linear in the size of the list. We outline a scheme to reward *active users*, i.e., those users who visit the site regularly (e.g., once a day, or once a week) with much faster authentication times at both the SP and the user's side. For services where the number of active users dominates the total authentications at the site, a large savings in computations costs is also seen at the SP, thus incentivizing SPs to offer express-lane authentication.

Consider Alice who authenticates at time T with respect to a particular list $\mathcal{L}_T$ and authenticates again at time $T + t$ with respect to the list $\mathcal{L}_{T+t} = \mathcal{L}_T \cup \partial_t$, where ∂_t is the set of newly blacklisted entries between time T and $T + t$. If ∂_t represents a small fraction of the list, a large amount of work in the second authentication is repeated. Specifically, the list score of Alice with respect to $\mathcal{L}_{T+t}$ is the sum of scores with respect to $\mathcal{L}_T$ *plus* that to ∂_t with appropriate weighting factors applied. Based on this observation, we can optimize the performance if a helper value tk_T, called *express pass*, is delivered to Alice after the first authentication. This express pass is a signature on the value s, the list score of an anonymous authenticating user Alice with respect to list $\mathcal{L}_T$. When Alice is going to authenticate in time $T + t$ based on list $\mathcal{L}_{T+t} = \mathcal{L}_T \cup \partial_t$, she produces a commitment of s and proves that she is in possession of the express pass on s. Additionally, she proves that her reputation with respect to list ∂_t is s_∂ and that her reputation is $s_{T_i+t} = s + s_\partial$. To speed up her authentication in the future, the SP issues a new express pass tk_{T+t} after Alice authentication. Note this approach would only work if the list presented at a later time can always be represented in the form of $\mathcal{L}_T \cup \partial_t$. This is true because we assume that entries are not removed from the blacklist. In Section 6 we discuss "unblacklisting" to forgive misbehaviors at larger system epochs.

If the express lane scheme is implemented as described, Alice would have to reveal the list $\mathcal{L}_T$ with respect to her express pass tk_T. This gives additional information to the SP about when Alice's last authentication was made. To address this issue, we assume the time is divided into time periods $T_1, T_2, \ldots$ corresponding to the authentication rates from active users. The length of this time period will depend on the particular system, but we expect this length to be a "few days," i.e., it is likely that most active users will authenticate every time period.

Let T_i be the current time period. At the start of period T_i, the SP announces the list $\mathcal{L}_i$, which is the list current up to the end of T_{i-1}. Recall that an express pass tk_{i-1} is the certification of a user's reputation with respect to list $\mathcal{L}_{i-1}$. It is required that $\mathcal{L}_i = \mathcal{L}_{i-1} \cup \partial_{i-1}$. At any time within time period T_i, an authenticating user will be given a list of the form $\mathcal{L} = \mathcal{L}_{i-1} \cup \partial_{i-1} \cup \partial_i^\star$, where $\partial_i^\star$ is the set of

new entries in T_i as of the time when the user is authenticating. Any user who has obtained an express pass tk_{i-1} in period T_{i-1} can choose to authenticate in the express lane or the normal lane. On the other hand, users who have not authenticated in time period T_{i-1} will not be in possession of tk_{i-1} and thus must authenticate in the normal lane. In case user Alice chooses to authenticate in the express lane, she computes the commitment of s_{i-1}, s'_{i-1} and s'_i which is her reputation with respect to list $\mathcal{L}_{i-1}$, ∂_{i-1} and $\partial_i^\star$ respectively. She can prove s_{i-1} is correctly formed using tk_{i-1}. If the authentication is successful, the SP issues a new express pass tk_i which certifies the value $s_{i-1} + s'_{i-1}$, Alice's reputation with respect to list $\mathcal{L}_i$. On the other hand, if Alice authenticates in the normal lane, she computes commitment of s_i and s'_i, which is her reputation with respect to list $\mathcal{L}_i$ and $\partial_i^\star$ respectively. Next, she proves that s_i and s'_i are correctly formed. A new express pass tk_i is issued with respect to s_i upon successful authentication. Figure 1 shows a concrete example of a user u authenticating first in the normal lane in time period T_9 with ZK-proofs for the entries L_9 and $\partial_9^\star$ and then in the express lane in time period T_{10} using tk_9 and ZK-proofs for the entries ∂_9 and $\partial_{10}^\star$.

3. Security Goals and Syntax

Security goals We give informal definitions of the security properties that a construction of the BLACR system must possess. It is similar to that of BLAC. The appendix formalizes these definitions.

Authenticity In a BLACR system with *authenticity*, SPs are assured to accept authentication only from users who satisfy the authentication policy.

Anonymity In a BLACR system with *anonymity*, all that SPs can infer about the identity of an authenticating user is whether the user satisfies the policy at the time of protocol execution, regardless of whatever the SPs do afterwards. With express lane authentication, the SP can infer additionally that an authenticating user has conducted an update protocol with the SP for the time period.

Note that express pass tk_i is issued whenever Alice authenticates in period T_i, regardless of the number of times Alice authenticates in the period. Note also tk_i is always with respect to list $\mathcal{L}_i$ in time period T_i. Thus, for anyone who chooses to authenticate in time period T_i using the express lane, all that the SP can infer is that the authenticating user has made at least one authentication in time period T_{i-1}.

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

Mis-authentication Resistance Mis-authentication occurs when an unregistered user successfully authenticates herself to an SP. In a BLACR system with *mis-authentication resistance*, SPs are assured to accept authentications only from registered users.

Syntax The entities in the BLACR system are the Group Manager (GM), a set of Service Providers (SPs) and a set of users. We note the GM cannot deanonymize or link users. The GM simply issues a credential to users and ensures each user gets exactly one credential (see Section 6 for a discussion on Sybil attacks). The GM is thus trusted to issue single credentials to users and is not trusted with the privacy of the users. The BLACR system consists of the following protocols:

Setup. This algorithm is executed by the GM to set up the system. On input of one or more security parameters, the algorithm outputs a group public key gpk and a group private key gsk. The GM keeps gsk private and publishes gpk to the public. gpk is an implicit input to all the algorithms described below. *SP Setup* This algorithm is executed by the SP to set up its public parameters. In particular, it initializes several lists $\mathcal{L}_1^+, \mathcal{L}_1^-, \ldots, \mathcal{L}_m^+, \mathcal{L}_m^-$, where $\mathcal{L}_i^+$, $\mathcal{L}_i^-$ are the meritlist and blacklist for category c_i respectively. The algorithm also outputs an identity string that uniquely identifies the SP.

Registration This protocol is executed between the GM and a legitimate user to register the user into the system. Upon successful completion of the protocol, the user obtains a credential usk, which she keeps private, and is thereby enrolled as a member in the group of registered users. We stress that this credential is known only to the user, i.e., the GM issues this credential in a *blind* way. In particular, a secret value x, which is part of the user's credential usk, is unknown to the GM.

Authentication This protocol is executed between a user Alice and an SP Bob. The input of Alice is her credential usk (with secret value x being part of usk). The input to Bob is a set of meritlists/blacklists $\{\mathcal{L}_i^+, \mathcal{L}_i^-\}_{i=1}^{\ell}$ and a policy $\bigvee_{k=1}^{\ell}(\bigwedge_{i=1}^{m}(\neg)\mathcal{P}_{ki})$.

When an execution of the protocol terminates, Bob outputs a binary value of `success` or `failure`. If the SP outputs `success` in an execution of the protocol, we call the execution a successful authentication and say that the authenticating user has succeeded in authenticating herself; otherwise the authentication is unsuccessful and the user has failed. Only upon a successful authentication does the SP establish an authenticated session with the authenticating user during which the user can access the service provided by the SP. Note that the *protocol transcript* of a successful authentication as seen by the SP contains $(b, \mathcal{T}(b, \mathsf{usk}))$ where b is some randomness specified by the user.

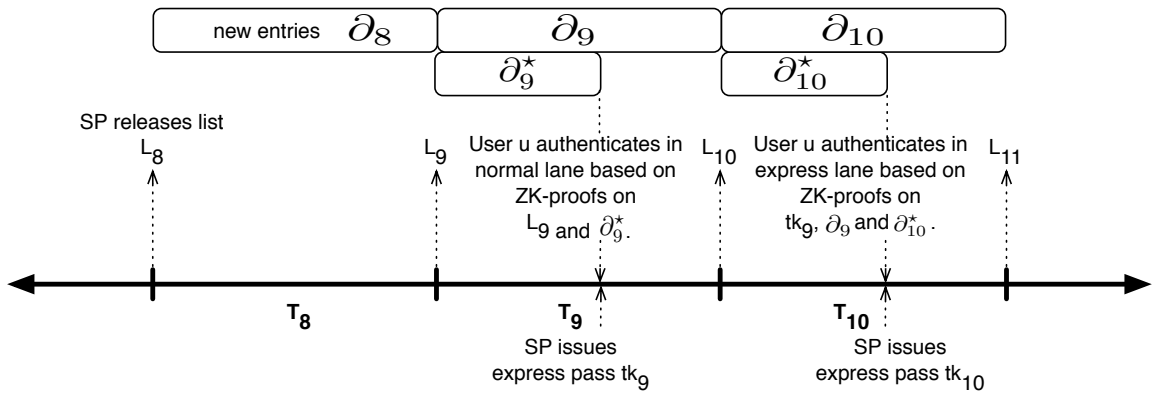


Figure 1. Timeline demonstrating the authentication process in normal and express lanes.

It is required that Alice is able to successfully authenticate herself to Bob with overwhelming probability if `Pol` evaluates to 1 on input of Alice's credential usk. When we say a user Alice with credential usk is revoked by an SP Bob with respect to policy `Pol`, we mean `Pol` evaluates to 0 on input of usk.

List management This is a suite of two algorithms: *Extract* and *Add*, which are executed by SPs for managing their lists. On input of an authentication protocol transcript, *Extract*(ϖ) returns a *ticket* $\tau = (b, t)$ from an authentication transcript ϖ. *Add*$((\tau, s), \mathcal{L})$ appends a new entry (τ, s) to the list.

4. Our Construction

4.1. 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 $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ be an efficiently computable isomorphism. Also let $\mathbb{G}$ be a group of order p where DDH is intractable. Let $g_0, g_1, g_2 \in \mathbb{G}_1$ and $h_0 \in \mathbb{G}_2$ be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively such that $g_0 = \psi(h_0)$ and the relative discrete logarithm of the generators are unknown.[1] Let $H_0 : \{0,1\}^* \to \mathbb{G}$ and $H : \{0,1\}^* \to \mathbb{Z}_p$ be collision-resistant hash functions. Throughout this section these parameters will be available to all parties.

4.2. Building blocks

Proofs of Knowledge In a *Zero-Knowledge Proof of Knowledge (ZKPoK)* protocol [17], 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 [18], that are secure under the *Random Oracle (RO)* Model [5]. We follow the notation introduced by Camenisch and Stadler [13]. For example, PK $\{(x) : y = g^x\}$ denotes a ZKPoK protocol that proves the knowledge of an integer x such that $y = g^x$ holds. Symbols appearing on the left of the colon denote values whose knowledge are being proved while symbols appearing on the right, but not the left, of the colon denote public values. The corresponding SPK on message M will be denoted as $SPK\{(x) : y = g^x\}(M)$. BLACR utilizes the ZKPoK that x does *not* equal $\log_b t$, denoted as $PK\{(x) : y = \textsc{Cmt}(x) \wedge t \neq b^x\}$ due to Camenisch and Shoup [12].

Commitment scheme Our construction uses the well known non-interactive commitment scheme due to Pedersen [26], 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 a 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 $C = g^x h^r$.

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 hand hand, a PPT sender can only reveal the commitment with one value under the discrete log assumption.

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

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

Credential signature scheme We employ the signature scheme proposed by Au et al. [3], which is based on the schemes of Camenisch and Lysyanskaya [11] and of Boneh et al. [6], to certify enrolled users in our system. Their scheme, called BBS+ signature, is briefly reviewed here. Let $g, g_0, g_1, g_2, \ldots, g_k \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$ be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively such that $g = \psi(h)$, where ψ is a computable isomorphism and $(\mathbb{G}_1, \mathbb{G}_2)$ is a pair of groups of prime order p. Let $\hat{e}$ be a pairing defined over the pair of groups.

The signer's secret is a value $\gamma \in \mathbb{Z}_p$ and the public key is $w = h^\gamma$. To create a signature over a tuple of messages $(m_0, m_1, \ldots, m_k)$, the signer randomly picks $e, y \in_R \mathbb{Z}_p$, computes $A = (g g_0^{m_0} g_1^{m_1} \cdots g_k^{m_k} g_{k+1}^{y})^{\frac{1}{\gamma+e}}$. The signer outputs (A, e, y) as the signature on message $(m_0, \ldots, m_k)$.

They also derive two useful protocols.

Protocol $\mathfrak{S}_{\mathsf{lss}}(C_0, C_1, \ldots, C_k)$: $\mathfrak{S}_{\mathsf{lss}}$ allows a user to obtains 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.

- 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:

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

- The signer returns `failure` if verification of the proof fails. Otherwise the signer randomly generates $e, y'' \in_R \mathbb{Z}_p$, computes $A = (g C_M g_{k+1}^{y''})^{\frac{1}{e+\gamma}}$ and returns (A, e, y'') to the user.
- The user computes $y = y' + y''$. She returns `failure` if $\hat{e}(A, w h^e) \neq \hat{e}(g g_0^{x_0} g_1^{x_1} \cdots g_{k+1}^{y}, h)$. Otherwise she outputs σ as (A, e, y).

Protocol $\mathfrak{S}_{\mathsf{Sig}}(C_0, C_1, \ldots, C_k)$: $\mathfrak{S}_{\mathsf{Sig}}$ allows a prover to convince a verifier he/she knows a credential signature σ on a block of messages $(x_0, \ldots, x_k)$ committed in $C_0, \ldots, C_k$. Let the prover'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 and the credential signature (A, e, y) such that the verification equation holds:

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

- 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.

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

  where $M' = M||A_1||A_2$ and $\beta_1 = k_1 e$, $\beta_2 = k_2 e$.
- The prover outputs $\mathfrak{p}_{\mathsf{Sig}}$ as (Π, A_1, A_2).
- Upon receiving $(\mathfrak{p}_{\mathsf{Sig}}, M)$, the verifier parses $\mathfrak{p}_{\mathsf{Sig}}$ as (Π, A_1, A_2) and outputs accept if Π is a valid proof.

Signature-based range proof To demonstrate the reputation of a user is greater than a certain value, we employ the signature-based range 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 as 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.

4.3. Useful Protocols

Based on the above building blocks, we construct the following zero-knowledge proof-of-knowledge protocols that are useful in BLACR.

Protocol $\mathfrak{S}_{\mathsf{x}}(C_x, t, g)$: $\mathfrak{S}_{\mathsf{x}}$ allows a prover to assure the verifier the value $\log_g t$ is committed in C_x. Verifier's input: C_x, t, g. Prover's input x, r such that $C_x = {g_1}^x {g_2}^r$ and $t = g^x$. Let M be the random challenge. The prover computes

$$\mathfrak{p}_{\mathsf{x}} := SPK\left\{(x, r) : C_x = g_1^x g_2^r \wedge t = g^x\right\}(M).$$

The output of the prover is $\mathfrak{p}_{\mathsf{x}}$.

Upon receiving $\mathfrak{p}_{\mathsf{x}}, M$, the verifier outputs accept if $\mathfrak{p}_{\mathsf{x}}$ is a valid proof.

Protocol $\mathfrak{S}_{\mathsf{WS\text{-}Adj}}(C_x, C_s, C_n, C_c/\perp, \mathcal{L}, \mathcal{D})$: Let $\mathcal{L}$ be a list which maybe the sub-sequence of any longer list $\mathcal{L}_i = \mathcal{L}_{i-1} \cup \mathcal{L}$. The weighted score of a user with secret value x

with respect to this sub-sequence $\mathcal{L}$ depends on both $\mathcal{L}$, the set of adjusting factors $\mathcal{D}$ and the value $n = \mathsf{Cnt}(\mathcal{L}_{i-1}, x)$, where

$$\mathsf{Cnt}(\mathcal{L}, x) \mapsto |\{b : \big((b, \mathsf{T}(b, x)), \cdot\big) \in \mathcal{L}\}|,$$

the number of times this user has been put on $\mathcal{L}_{i-1}$.

Protocol $\mathfrak{S}_{\mathsf{WS\text{-}Adj}}$ allows a user to convince any verifier that the value s committed in C_s is the weighted score of the user with secret value x committed in C_x with respect to the list $\mathcal{L}$, a set of adjusting factor $\mathcal{D}$, given a value $n = \mathsf{Cnt}(\mathcal{L}, x)$ committed in C_n. If $C_c \neq \perp$, the proof also convinces the verifier that C_c is a commitment of the value $\mathsf{Cnt}(\mathcal{L}, x)$.

The provers has additional inputs x, r_x, s, r_s, n, r_n such that $C_x = g_1^x g_2^{r_x}$, $C_s = g_1^s g_2^{r_s}$, $C_n = g_1^n g_2^{r_n}$ and in case $C_c \neq \perp$, the prover also knows c, r_c such that $C_c = g_1^c g_2^{r_c}$ respectively.

Parse $\mathcal{D}$ as $(i, \Delta_i, \sigma_i)_{i=1}^D$. Let $\mathcal{I} : \{\iota | (b_\iota, t_\iota, s_\iota) \in \mathcal{L}, H(b_\iota || \mathtt{sid})^x = t_\iota\}$ be an index set. For all $\iota \in \mathcal{I}$, let $k_\iota = |\{j : 1 \leq j \leq \iota \wedge t_\iota = \hat{b}_\iota^x\}| + n$. Note the role of n in the definition of k_ι. It can be seen that for any $\iota \in \mathcal{I}$, Δ_{k_ι} is the appropriate adjusting factor for the score s_ι. Let M be the random challenge.

- Produce auxiliary commitments for each score on the list $\mathcal{L}$

  $$\mathsf{aux} = (C_1^s, C_1^n, \ldots, C_L^s, C_L^n)$$

  as follows. Randomly generates $a_\iota, b_\iota \in_R \mathbb{Z}_p$ for $\iota = 1$ to L and computes:

  $$(C_\iota^s, C_\iota^n) = \begin{cases} (g_1^{\Delta_{k_\iota} s_\iota} g_2^{a_\iota}, g_1 g_2^{b_\iota}) & \text{for} \quad \iota \in \mathcal{I} \\ (g_2^{a_\iota}, g_2^{b_\iota}) & \text{for} \quad \iota \in [L] \backslash \mathcal{I} \end{cases}$$

  Note that C_ι^s is a commitment of the adjusted score on list $\mathcal{L}$ with respect to secret value x. C_ι^n is a commitment of 0 or 1 which indicates if the entry is on the list. The superscript s and n here does not represent exponentiation.

- Generate a proof Π_1 to demonstrate the correctness of aux. We give the intuition of how Π_1 shows C_ι^s is a commitment of the weighted score of the authenticating user. Firstly, C_ι^n acts as a boolean flag, hidden from the verifier, indicating if τ_ι is a ticket from the authenticating user. Thus, for all $\iota \in [L] \backslash \mathcal{I}$, C_ι^s, C_ι^n should be commitment of 0. On the other hand, when $\iota \in \mathcal{I}$, C_ι^n is a commitment of 1. Due to the homomorphic property of the commitment scheme, $C_n \prod_{j=1}^{\iota} C_\iota^n$ is a commitment of the value k_ι, which is the number of times the user with secret x has been put on the list up to the ι-th entry (note that C_n is applied to the product to increase the number of times by n). Thus, the correct adjusting factor for the of this entry is Δ_{k_ι}. Recall that σ_ι is the signature from the SP on the tuple (Δ_ι, ι). Thus, the proof that $\mathsf{Verify}(\sigma_{k_\iota}, k_\iota, \Delta_{k_\iota}) = 1$ binds the value of k_ι to the appropriate adjusting factor Δ_{k_ι}. Finally, the proof demonstrates that the correct weighted score of this entry, $\Delta_{k_\iota} s_\iota$ is committed in C_ι^s (here $\beta_\iota = \sum_{j=1}^{\iota} b_\iota + r_n$).

$$\Pi_1 = SPK \left\{ \begin{array}{l} (x, r_x, \{\sigma_{k_\iota}, \Delta_{k_\iota}, k_\iota, \beta_\iota, a_\iota, b_\iota\}_{\iota=1}^L) : \\ \left(\begin{array}{c} \left(\begin{array}{c} C_x = g_1{}^x g_2{}^{r_x} \wedge \\ t_\iota \neq \hat{b}_\iota^x \wedge \\ C_\iota^s = g_2^{a_\iota} \wedge \\ C_\iota^n = g_2^{b_\iota} \end{array} \right) \bigvee \\ \left(\begin{array}{c} C_x = g_1{}^x g_2{}^{r_x} \wedge \\ t_\iota = \hat{b}_\iota^x \wedge \\ C_\iota^n = g_1 g_2^{b_\iota} \wedge \\ C_n \cdot \prod_{j=1}^{\iota} C_\iota^n = g_1^{k_\iota} g_2^{\beta_\iota} \wedge \\ 1 = \mathsf{Verify}(\sigma_{k_\iota}, k_\iota, \Delta_{k_\iota}) \wedge \\ C_\iota^s = g_1^{\Delta_{k_\iota} s_\iota} g_2^{a_\iota} \end{array} \right) \end{array} \right)_{\iota=1}^{L} \end{array} \right\} (M)$$

- The prover computes $C_s' = \prod_{i=1}^L C_i^s$, $r_s' = \sum_{i=1}^L a_i$. If $C_c \neq \perp$, the prover also computes $C_c' = \prod_{i=1}^L C_i^n$, $r_c' = \sum_{i=1}^L b_i$ and produces the following proof.

$$\Pi_2 = \begin{cases} SPK \left\{ \begin{array}{l} (s, r_s, r_s', c, r_c, r_c') : \\ C_s = g_1^s g_2^{r_s} \wedge \\ C_s' = g_1^s g_2^{r_s'} \wedge \\ C_c = g_1^c g_2^{r_c} \wedge \\ C_c' = g_1^c g_2^{r_c'} \end{array} \right\} (M) \text{ if } C_c \neq \perp \\ SPK \left\{ \begin{array}{l} (s, r_s, r_s') : \\ C_s = g_1^s g_2^{r_s} \wedge \\ C_s' = g_1^s g_2^{r_s'} \end{array} \right\} (M) \text{ otherwise.} \end{cases}$$

- The prover outputs $\mathfrak{p}_{\mathsf{WS\text{-}Adj}}$ as $(\Pi_1, \Pi_2, \mathsf{aux})$.
- Upon receiving $(\mathfrak{p}_{\mathsf{WS\text{-}Adj}}, M)$, the verifier parses $\mathfrak{p}_{\mathsf{WS\text{-}Adj}}$ as $(\Pi_1, \Pi_2, \mathsf{aux})$, computes locally $C_s' = \prod_{i=1}^L C_i^s$, $C_c' = \prod_{i=1}^L C_i^n$ and outputs accept if Π_1 and Π_2 are both valid proofs.

Protocol $\mathfrak{S}_{\mathsf{Pol}}(\mathtt{Pol}, C_1, \ldots, C_m)$: $\mathfrak{S}_{\mathsf{Pol}}$ allows a prover to convince a verifier the set of values committed in C_i would satisfy the authentication policy $\mathtt{Pol}$ should they represent reputation of a user in the m categories. Let M be the random challenge. The prover has additional inputs R_i, r_i such that $C_i = g_1^{R_i} g_2^{r_i}$.

- The prover parse $\mathtt{Pol}$ as the statement $\bigvee_{k=1}^{\ell} (\bigwedge_{i=1}^{m} (\neg) \mathcal{P}_{ki})$.
- The prover produces the following proof Π:

$$SPK \left\{ \begin{array}{l} (\{R_i, r_i\}_{i=1}^m) : \\ \bigvee_{k=1}^{\ell} (\bigwedge_{i=1}^{m} C_i = g_1^{R_i} g_2^{r_i} \wedge R_i \, \square_{ki} \, n_{ki}) \end{array} \right\} (M)$$

where $\square_{ki} \in \{\geq, <\}$ depending on whether the policy is $\mathcal{P}_{ki}$ or $\neg\mathcal{P}_{ki}$.

- The prover outputs $\mathfrak{p}_{\mathsf{Pol}}$ as Π.
- Upon receiving $(\mathfrak{p}_{\mathsf{Score}}, M)$, the verifier outputs accept if Π is a valid proof.

4.4. Our construction of BLACR

4.4.1 Setup

The GM randomly chooses $\gamma \in_R \mathbb{Z}_p$ and computes $w = h_0^{\gamma}$. The group secret key is $\mathsf{gsk} = (\gamma)$ and the group public key is $\mathsf{gpk} = (w)$.

4.4.2 SP Setup

Let m be the number of categories. Each SP publishes a unique identity string $\mathtt{sid}$ and a set of generators $g_{\mathtt{sid},}, g_{\mathtt{sid},0}, \ldots, g_{\mathtt{sid},m} \in_R \mathbb{G}_1$, $h_{\mathtt{sid}} \in_R \mathbb{G}_2$ such that $\psi(h_{\mathtt{sid}}) = g_{\mathtt{sid},}$. It chooses $\gamma_{\mathtt{sid}} \in_R \mathbb{Z}_p$, computes $w_{\mathtt{sid}} = {h_{\mathtt{sid}}}^{\gamma_{\mathtt{sid}}}$ and publishes $w_{\mathtt{sid}}$ as well. Note that they are in fact the verification key of the credential signature scheme with signing key $\gamma_{\mathtt{sid}}$. SP also initializes meritlist and blacklist of each category. A meritlist $\mathcal{L}_i^+$ and blacklist $\mathcal{L}_i^-$ for category c_i is a list of tuples $(\{0,1\}^{\lambda}, \mathbb{G}, [s\mathrm{max}])$ were $s\mathrm{max}$ is the maximum score associated with a misbehavior or good behavior. The SP also publishes the set of adjusting factors $\mathcal{D}_i^+$ and $\mathcal{D}_i^-$ for each category c_i. For every $\Delta_i^+ \in \mathcal{D}_k^+$ (resp. $\Delta_i^- \in \mathcal{D}_k^-$), the SP further publishes a signature σ_i^+ (resp. σ_i^-) on the values (i, Δ_i^+) (resp. (i, Δ_i^-)). For each $\mathcal{D}$, the signatures should be generated using a different key pairs.

4.4.3 Registration

Alice randomly picks a secret number $x \in_R \mathbb{Z}_p$ and computes $C_x = \mathrm{CMT}(x)$. She engages with the GM in protocol $\mathfrak{S}_{\mathsf{iss}}(C_x)$ and obtains a tuple (A, e, y) such that (A, e, y) is a credential signature on x. She stores her credential $\mathsf{usk} = (A, e, y, x)$. We note that x is known only to Alice (i.e., and not the GM).

4.4.4 Normal Lane Authentication

Let $\mathtt{pd}$ be the current time period. Recalled that any list presented in this period by the SP will have the form: $\mathcal{L}_{i,\mathtt{pd}}^{\diamond} \cup \partial_{i,\mathtt{pd}}^{\diamond,\star}$ where $\diamond \in \{+,-\}$ for category c_i. The set of lists $\mathcal{L}_{i,\mathtt{pd}}$ for $i = 1$ to m at the beginning of period $\mathtt{pd}$ and the adjusting factors are assumed to be known. So the SP would simply transmit the set $\partial_{i,\mathtt{pd}}^{\diamond,\star}$ to the user.

During an execution of this protocol between a user Alice and the SP, Alice's private input is her credential $\mathsf{usk} = (A, e, x, y)$.

When the protocol terminates, the SP outputs `success` or `failure`, indicating whether the SP should consider the authentication attempt successful. If SP outputs `success`, Alice's output is $\mathsf{tk}_{\mathtt{pd}}$, which is the express pass token for her to authenticate in the express lane for period $\mathtt{pd}+1$.

1. *(Challenge.)* The SP sends to Alice the lists for each category, the adjusting factors and a random challenge $(\partial_{1,\mathtt{pd}}^{+,\star}, \partial_{1,\mathtt{pd}}^{-,\star}, \ldots, \partial_{m,\mathtt{pd}}^{+,\star}, \partial_{m,\mathtt{pd}}^{-,\star}, M)$ as well as the policy `Pol`.

2. *(Inspection.)* Alice parses each list as $\mathcal{L}_i^{\diamond} = \mathcal{L}_{i,\mathtt{pd}}^{\diamond} \cup \partial_{i,\mathtt{pd}}^{\diamond,\star}$ for $i = 1$ to m and $\diamond \in \{+,-\}$. She computes $s_{i,\mathtt{pd}}^{\diamond}$, $s_{i,\partial_{\mathtt{pd}}^{\star}}^{\diamond}$ which is her weighted-score with respect to list $\mathcal{L}_{i,\mathtt{pd}}^{\diamond}$ and $\partial_{i,\mathtt{pd}}^{\diamond,\star}$ respectively. She computes her reputation $R_i = s_{i,\mathtt{pd}}^{+} + s_{i,\partial_{\mathtt{pd}}^{\star}}^{+} - s_{i,\mathtt{pd}}^{-} - s_{i,\partial_{\mathtt{pd}}^{\star}}^{-}$ and checks if she satisfies the policy `Pol`. If not, she returns as `failure`, indicating she is revoked.

3. *(Proof Generation.)* If Alice is not revoked, she computes the following commitments, for $i = 1$ to m:

$$C_x = \mathrm{CMT}(x), \qquad C_{i,\mathtt{pd}}^{s^+} = \mathrm{CMT}(s_{i,\mathtt{pd}}^{+}),$$
$$C_{i,\partial_{\mathtt{pd}}^{\star}}^{s^+} = \mathrm{CMT}(s_{i,\partial_{\mathtt{pd}}^{\star}}^{+}), \quad C_{i,\mathtt{pd}}^{n^+} = \mathrm{CMT}(n_{i,\mathtt{pd}}^{+}),$$
$$C_{i,\mathtt{pd}}^{s^-} = \mathrm{CMT}(s_{i,\mathtt{pd}}^{-}), \quad C_{i,\mathtt{pd}}^{n^-} = \mathrm{CMT}(n_{i,\mathtt{pd}}^{-})$$
$$C_{i,\partial_{\mathtt{pd}}^{\star}}^{s^-} = \mathrm{CMT}(s_{i,\partial_{\mathtt{pd}}^{\star}}^{-})$$

where $n_{i,\mathtt{pd}}^{+} = \mathsf{Cnt}(x, \mathcal{L}_{i,\mathtt{pd}}^{+})$ and $n_{i,\mathtt{pd}}^{-} = \mathsf{Cnt}(x, \mathcal{L}_{i,\mathtt{pd}}^{-}$. She also computes a value $\tau = (b, t := H(b||\mathtt{sid})^x)$. Here $b \in_R \{0,1\}^{\lambda}$ is a random value chosen by Alice. She sends $(C_x, \{C_{i,\mathtt{pd}}^{s^+}, C_{i,\mathtt{pd}}^{n^+}, C_{i,\partial_{\mathtt{pd}}^{\star}}^{s^+}, C_{i,\mathtt{pd}}^{s^-}, C_{i,\mathtt{pd}}^{n^-}, C_{i,\partial_{\mathtt{pd}}^{\star}}^{s^-}\}_{i=1}^{m}, \tau)$, along with a proof Π such that these values are correctly formed and that `Pol` evaluates to 1. Define C_i for $i = 1$ to m as $\frac{C_{i,\mathtt{pd}}^{s^+} C_{i,\partial_{\mathtt{pd}}^{\star}}^{s^+}}{C_{i,\mathtt{pd}}^{s^-} C_{i,\partial_{\mathtt{pd}}^{\star}}^{s^-}}$. Note that both party can compute C_i locally. The proof Π is carried out through executions of the following protocols.

   - Execute protocol $\mathfrak{S}_{\mathsf{Sig}}(C_x)$ to assure the SP that Alice is in possession of a credential signature (A, e, y) on a secret value x, which is committed in C_x.
   - Execute $\mathfrak{S}_{\mathsf{x}}(C_x, t, H(b||\mathtt{sid}))$ to assure the SP $\log_{H(b||\mathtt{sid})} t = x$ and x is committed in C_x.
   - Execute $\mathfrak{S}_{\mathsf{WS\text{-}Adj}}(C_x, C_{i,\mathtt{pd}}^{s^\diamond}, C_{i,\mathtt{pd}}^{n^\diamond}, 1, \mathcal{L}_{i,\mathtt{pd}}^{\diamond}, \mathcal{D}_i^{\diamond})$ for $i = 1$ to m, $\diamond \in \{+,-\}$ to assure the SP all $C_{i,\mathtt{pd}}^{s^\diamond}$, $C_{i,\mathtt{pd}}^{n^\diamond}$ are correctly formed. Note that 1 represents a commitment of 0.
   - Execute $\mathfrak{S}_{\mathsf{WS\text{-}Adj}}(C_x, C_{i,\partial_{\mathtt{pd}}^{\star}}^{s^\diamond}, \perp, C_{i,\mathtt{pd}}^{n^\diamond}, \partial_{i,\mathtt{pd}}^{\diamond,\star}, \mathcal{D}_i^{\diamond})$ for $i = 1$ to m, $\diamond \in \{+,-\}$ to assure the SP all $C_{i,\partial_{\mathtt{pd}}^{\star}}^{s^\diamond}$ is correctly formed.

- Execute $\mathfrak{S}_{\mathsf{Pol}}(\mathtt{Pol}, C_1, \ldots, C_m)$ to ensure that the set of reputations of each category committed in C_i satisfies the policy $\mathtt{Pol}$.

The SP outputs `success` if and only if the proof Π is a valid proof. The SP stores ticket τ extracted from the transcript, along with information logging Alice's activity within the authenticated session. If the SP outputs `success`, the SP issues the express pass $\mathsf{tk}_{\mathtt{pd}}$ for the user by executing protocol $\mathfrak{S}_{\mathsf{lss}}(C_x, C^{s^+}_{1,\mathtt{pd}}, C^{n^+}_{1,\mathtt{pd}}, C^{s^-}_{1,\mathtt{pd}}, C^{n^-}_{1,\mathtt{pd}}, \ldots, C^{s^+}_{m,\mathtt{pd}}, C^{n^+}_{m,\mathtt{pd}}, C^{s^-}_{m,\mathtt{pd}}, C^{n^-}_{m,\mathtt{pd}})$ with Alice to issue a signature σ_x on values $(x, s^+_{1,\mathtt{pd}}, n^+_{1,\mathtt{pd}}, s^-_{1,\mathtt{pd}}, n^-_{1,\mathtt{pd}}, \ldots, s^+_{m,\mathtt{pd}}, n^+_{m,\mathtt{pd}}, s^-_{m,\mathtt{pd}}, n^-_{m,\mathtt{pd}})$. Alice stores $(\sigma_x, s^+_{1,\mathtt{pd}}, n^+_{1,\mathtt{pd}}, s^-_{1,\mathtt{pd}}, n^-_{1,\mathtt{pd}}, \ldots, s^+_{m,\mathtt{pd}}, n^+_{m,\mathtt{pd}}, s^-_{m,\mathtt{pd}}, n^-_{m,\mathtt{pd}})$ as her express pass $\mathsf{tk}_{\mathtt{pd}}$.

4.4.5 Express Lane Authentication

We describe the difference with normal lane authentication. Let `pd` be the current time period. Suppose Alice has an express pass $\mathsf{tk}_{\mathtt{pd}-1}$. Let the current list be $\mathcal{L}^{\diamond}_{i,\mathtt{pd}} = \mathcal{L}^{\diamond}_{i,\mathtt{pd}-1} \cup \partial^{\diamond}_{i,\mathtt{pd}-1} \cup \partial^{\diamond,\star}_{i,\mathtt{pd}}$.

1. *(Challenge.)* Same as normal lane.
2. *(Inspection.)* Alice parses each list as $\mathcal{L}^{\diamond}_i = \mathcal{L}^{\diamond}_{i,\mathtt{pd}-1} \cup \partial^{\diamond}_{i,\mathtt{pd}-1} \cup \partial^{\diamond,\star}_{i,\mathtt{pd}}$ for $i = 1$ to m and $\diamond \in \{+,-\}$. Again, she returns `failure` if she is revoked.
3. *(Proof Generation.)* If Alice is not revoked, she computes the following commitments, for $i = 1$ to m, $\diamond \in \{+,-\}$:

$$C_x = \textsc{Cmt}(x), \qquad C^{s^\diamond}_{i,\mathtt{pd}-1} = \textsc{Cmt}(s^{\diamond}_{i,\mathtt{pd}-1})$$
$$C^{s^\diamond}_{i,\partial_{\mathtt{pd}-1}} = \textsc{Cmt}(s^{\diamond}_{i,\partial_{\mathtt{pd}-1}}), \qquad C^{s^\diamond}_{i,\partial^{\star}_{\mathtt{pd}}} = \textsc{Cmt}(s^{\diamond}_{i,\partial^{\star}_{\mathtt{pd}}})$$
$$C^{n^\diamond}_{i,\mathtt{pd}-1} = \textsc{Cmt}(n^{\diamond}_{i,\mathtt{pd}-1}), \qquad C^{n^\diamond}_{i,\mathtt{pd}} = \textsc{Cmt}(n^{\diamond}_{i,\mathtt{pd}})$$

where $n^{\diamond}_{i,\cdot} = \mathsf{Cnt}(x, \mathcal{L}^{\diamond}_{i,\cdot})$. She also computes a value $\tau = (b, t := H(b||\mathtt{sid})^x)$. Here $b \in_R \{0,1\}^\lambda$ is a random value chosen by Alice. She sends all the commitments and τ, along with a proof Π such that these values are correctly formed and that `Pol` evaluates to 1. Define C_i for $i = 1$ to m as $\frac{C^{s^+}_{i,\mathtt{pd}-1} C^{s^+}_{i,\partial_{\mathtt{pd}-1}} C^{s^+}_{i,\partial^{\star}_{\mathtt{pd}}}}{C^{s^-}_{i,\mathtt{pd}} C^{s^-}_{i,\partial_{\mathtt{pd}-1}} C^{s^-}_{i,\partial^{\star}_{\mathtt{pd}}}}$. Note that both party can compute C_i locally.

The proof Π is carried out through executions of the following protocols.

- Execute protocol $\mathfrak{S}_{\mathsf{Sig}}(C_x)$ to assure the SP that Alice is in possession of a credential signature (A, e, y) on a secret value x, which is committed in C_x.
- Execute protocol $\mathfrak{S}_{\mathsf{Sig}}(C_x, C^{s^+}_{1,\mathtt{pd}-1}, C^{n^+}_{1,\mathtt{pd}-1}, C^{s^-}_{1,\mathtt{pd}-1}, C^{n^-}_{1,\mathtt{pd}-1}, \ldots, C^{s^+}_{m,\mathtt{pd}-1}, C^{n^+}_{m,\mathtt{pd}-1}, C^{s^-}_{m,\mathtt{pd}-1}, C^{n^-}_{m,\mathtt{pd}-1})$ to assure the SP that Alice is in possession of an express pas $\mathsf{tk}_{\mathtt{pd}-1}$.
- Execute $\mathfrak{S}_{\mathsf{x}}(C_x, t, H(b||\mathtt{sid}))$ to assure the SP $\log_{H(b||\mathtt{sid})} t = x$ and x is committed in C_x.
- Execute $\mathfrak{S}_{\mathsf{WS\text{-}Adj}}(C_x, C^{s^\diamond}_{i,\partial_{\mathtt{pd}-1}}, C^{n^\diamond}_{i,\partial_{\mathtt{pd}-1}}, 1, \partial^{\diamond}_{i,\mathtt{pd}-1}, \mathcal{D}^{\diamond}_i)$ for $i = 1$ to m, $\diamond \in \{+,-\}$ to assure the SP all $C^{s^\diamond}_{i,\partial_{\mathtt{pd}-1}}, C^{n^\diamond}_{i,\partial_{\mathtt{pd}-1}}$ are correctly formed. Note that 1 represents a commitment of 0.
- Execute $\mathfrak{S}_{\mathsf{WS\text{-}Adj}}(C_x, C^{s^\diamond}_{i,\partial^{\star}_{\mathtt{pd}}}, \perp, C^{n^\diamond}_{i,\mathtt{pd}}, \partial^{\diamond,\star}_{i,\mathtt{pd}}, \mathcal{D}^{\diamond}_i)$ for $i = 1$ to m, $\diamond \in \{+,-\}$ to assure the SP all $C^{s^\diamond}_{i,\partial^{\star}_{\mathtt{pd}}}$ is correctly formed.
- Execute $\mathfrak{S}_{\mathsf{Pol}}(\mathtt{Pol}, C_1, \ldots, C_m)$ to ensure that the set of reputations of each category committed in C_i satisfies the policy $\mathtt{Pol}$.

Again, the SP outputs `success` if and only if the proof Π is a valid proof. The express pass $\mathsf{tk}_{\mathtt{pd}}$ is issued for the user by executing protocol $\mathfrak{S}_{\mathsf{lss}}$. Both party can computes locally the commitment $C^{s^\diamond}_{i,\mathtt{pd}}$ as $C^{s^\diamond}_{i,\mathtt{pd}-1} C^{s^\diamond}_{i,\partial_{\mathtt{pd}-1}}$ and $C^{n^\diamond}_{i,\mathtt{pd}} = C^{n^\diamond}_{i,\mathtt{pd}-1} C^{n^\diamond}_{i,\partial_{\mathtt{pd}-1}}$.

4.4.6 List management

These algorithms are all very simple and efficient. *Extract*(ϖ) returns ticket τ in the input transcript ϖ. *Add*$(\mathcal{L}, (\tau, s))$ returns list $\mathcal{L}'$, which is the same as the input list $\mathcal{L}$, except with the input tuple (τ, s) appended to it.

4.5. Security analysis

BLACR possesses mis-authentication resistance, authenticity, anonymity and non-frameability. As mentioned earlier, we describe the formal security model and the proof of security of BLACR in the appendix.

5. Performance Evaluation

We demonstrate the practicality of BLACR and compare it to both BLAC and PEREA. First we compare the asymptotic complexities, and then provide a detailed quantitative analysis of the schemes.

5.1. Complexity analysis

As summarized in Table 1, the asymptotic complexity of BLACR-Normal is the same as BLAC. Generating the proofs takes $O(L)$ time for the user in BLAC and BLACR-Normal, and $O(K\Delta_L)$ in PEREA as each witness must be updated Δ_L times. K is the size of the revocation window

Schemes	Communication		Computation		
	Downlink	Uplink	User (Check+Prove)		Server
BLAC/EPID	$O(L)$	$O(L)$	$O(L)$ + $O(L)$		$O(L)$
PEREA	$O(L)$	$O(K)$	$O(L)$ + $O(K\Delta_L)$		$O(K)$
BLACR-Normal	$O(L)$	$O(L)$	$O(L)$ + $O(L)$		$O(L)$
BLACR-Express	$O(\lvert\partial^\star_{\mathtt{pd}}\rvert)$	$O(\lvert\partial_{\mathtt{pd}-1}\rvert + \lvert\partial^\star_{\mathtt{pd}}\rvert)$	$O(\lvert\partial_{\mathtt{pd}-1}\rvert + \lvert\partial^\star_{\mathtt{pd}}\rvert)$	+ $O(\lvert\partial_{\mathtt{pd}-1}\rvert + \lvert\partial^\star_{\mathtt{pd}}\rvert)$	$O(\lvert\partial_{\mathtt{pd}-1}\rvert + \lvert\partial^\star_{\mathtt{pd}}\rvert)$

Table 1. Asymptotic Complexities of Authentication

(the number of authentications before which a misbehavior must be caught to result in a revocation), and Δ_L is the number of new entries on the blacklist since the user's previous authentication. Verifying the proofs also takes $O(L)$ at the SP for BLAC/EPID and BLACR-Normal, whereas PEREA requires only $O(K)$ computation at the server. For BLACR-Express, the complexities depend on the number of new entries in the previous time period ($\partial_{\mathtt{pd}-1}$) and the current time period ($\partial^\star_{\mathtt{pd}}$).

The downlink communications complexity is linear in the size of the list in all schemes. The uplink communication complexities are the same as the computational complexities at the server: $O(K)$ for PEREA, $O(L)$ for BLAC/EPID/BLACR-Normal and $O(|\partial_{\mathtt{pd}-1} + \partial_{\mathtt{pd}}|)$ for BLACR-Express.

5.2. Quantitative analysis

Data transfer The various communication costs for BLACR are given in Table 2. For our analysis we make the following assumptions: the score of each entry is represented with 5 bits, the threshold with 10 bits, and we assume the adjusting factors are treated as a public parameter. Following the suggested parameters [4] to give a security level of 112 bits for BLAC, we set the security parameter to 249, i.e., we assume p is a 249-bit prime. Under optimal conditions, elements in $\mathbb{Z}_p$ and $|\mathbb{G}_1|$ would be 249 and 250 bits respectively. Currently we use 281 bits for $|\mathbb{G}_1|$ because that is the closest matching curve in the PBC library. The constant ℓ is the number of conjunctive clauses in the policy in BLACR. Further, we assume each of these clauses involves all m categories and thus each policy consists of ℓm sub-policies. We assume $\ell = 10$, $m = 10$ in our analysis, and that the lists for each category are equal in size. Downloading a list of 5,000 entries is under 325KB and a full normal lane authentication requires uploading a proof of size less than 4MB. Assuming that only 2% of the 5,000 entries are new, express lane authentication requires a download and upload of size less than 6.5KB and 123KB respectively, which is a considerable savings. Nevertheless, since the transfer sizes for normal lane authentication are similar to uploading standard JPEG images (and cost on the order of millicents based on current Amazon EC2 pricing), we deem the data transfer sizes to be acceptable.

Computation Table 4 outlines the number of multi-based exponentiations (EXPs) as a measure of time complexity of BLAC, PEREA and BLACR, with and without precomputation, i.e., the pre-processing that can be done by the user before seeing the lists. Fortunately, in BLACR a significant amount of work can be pre-computed by the user and that results in low latencies at the user as we show in Figure 2. Let L be the size of the blacklist in BLAC and PEREA. Assume A is the number of tickets that do not belong to the user. For our evaluation of BLACR, we assume there are ℓ sub-policies and that both the meritlist and blacklist in each sub-policy are of size $L/(2\ell)$. Assume ζ is the fraction of tickets that belong to the user and that they are distributed evenly across all the meritlist or blacklist. Δ_L represents the change of the list from the last time the user authenticates. Both BLAC and BLACR show significant improvement in performance if the XDH assumption is made, and thus we make this assumption in our performance analysis.

The benchmarks shown in Table 3 are obtained on a Lenovo X200s with an Intel Core 2 Duo CPU L9400 and 4GB RAM running Windows Vista as the host. We used Oracle VirtualBox 4.0.4 to emulate a guest machine of 512MB RAM running Ubuntu 10.10. Timings of $E1$, ET, EG, and P are obtained using test code based on the Pairing-Based Cryptography (PBC) library[2] (version 0.5.11) based on the type D pairing parameter, with $|p| = 249$, bundled with the PBC library. The following table summarizes our experimental results. For $EN1$ and $EN2$ the test code is written in C based on the MIRACL library[3] (version 5.4.2). The modulus N is taken to be 4593 bits. The small exponent is taken to be 224 bits. The computation for an authentication in BLACR is highly parallelizable. In fact, all the work with respect to each entry in the list can be done independently. Thus, the system scales well with the increase in the number of cores in the CPU and our subsequent analysis assumes this scalability. Since 8-core server configurations are standard today, we assume such servers at the SPs. Likewise, 4-core desktops and laptops are becoming more commonplace, and we assume such configurations at the user.

What is practical? For a heavily loaded SP such as Wikipedia, the English webpages are updated at the rate of about 120 edits/minute.[4] Absent any clear guideline, we assume 20% of these edits may come from users desiring

[2] `http://crypto.stanford.edu/pbc/`

[3] `http://www.shamus.ie/`

[4] http://stats.wikimedia.org/EN/PlotsPngDatabaseEdits.htm

Schemes	Protocol	Downlink	Uplink
BLACR-Weighted	Authentication-Normal	$(504(\|L\|) + 1994)$bits	$(6479(\|L\|) + 2741\ell m + 5484m + 3740)$bits
BLACR-Weighted	Authentication-Express	$(504(\|\partial_{\mathtt{pd}-1}\| + \|\partial^{\star}_{\mathtt{pd}}\|) + 1994)$bits	$(6479(\|\partial_{\mathtt{pd}-1}\| + \|\partial^{\star}_{\mathtt{pd}}\|) + 2741\ell m + 7476m + 5983)$bits

Table 2. Space Complexities of BLACR

Operations	Legend	Mode	Time
$\mathbb{G}_1$-EXP	E1	Multi-based EXP	5.891ms
	EP1	With/Pre-Processing (fixed single base)	0.784ms
$\mathbb{G}_2$-EXP	E2	Multi-based EXP	44.433ms
	EP2	With/Pre-Processing (fixed single base)	6.278ms
$\mathbb{G}_T$-EXP	ET	Multi-based EXP	12.035ms
	EPT	With/Pre-Processing (fixed single base)	2.015ms
Pairing	P	Normal	40.373ms
	PP	With/Pre-Processing (one input fixed)	32.892ms
EXP modulo N	EN1	Small Exponent (Without/Pre-Processing)	40.64ms
	EN2	With/Pre-Processing (fixed base)	89.63ms

Table 3. Benchmark of different operations

strong anonymity, and *our main goal is to support authentications at the rate of about 25 authentications/minute at the SP for active users and about 1 authentication/minute for inactive users.* A secondary goal is to keep authentication latencies low at the user. We deem tens of seconds of computation per authentication is reasonable for users.

In our analysis we show that around 5,000 entries on the blacklist is a reasonable tradeoff because normal-lane users can be authenticated in about a minute, and most authentications take place in the express lane in a couple of seconds (assuming only 2% of the entries have changed since the last authentication). Given this tradeoff, we expect servers to trim their blacklists and meritlists to noteworthy bad and good behaviors. Larger sites (e.g., Wikipedia) will need to focus on egregious violations or particularly rewarding behaviors, while smaller sites (e.g., organizations such as a large university) can penalize/reward many more behaviors. Scalability is maintained by tuning what degrees of good/bad behaviors make it on the list and how many servers the provider is willing to purchase (costs scale as O(L)). As discussed in Section 6, we can assume that most users are in the express lane, and that the SP can limit the number of authentications accepted in the normal lane to maintain a high throughput of authentications.

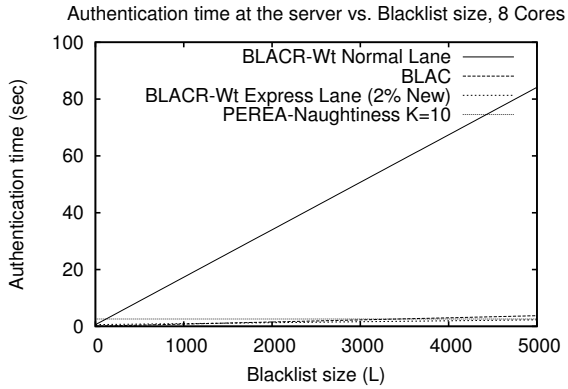
Performance at the SP. As we can see in Figure 2(a), for 3,000–5,000 entries on the blacklist/meritlist, authentication in BLACR-Normal takes 51–84 seconds using an 8-core server (to give a baseline idea of cost, a fully-reserved instance of such a server on Amazon EC2 would cost around $2,500/year including data transfer costs). Authentication in BLACR-Express would on the other hand support about 26–38 anonymous authentications/minute. For comparison, BLAC can support about 16–26 authentications/minute. We also see that PEREA-Naughtiness (the closest analog to BLACR) has authentication times independent of the size of the blacklist but linear in the revocation window K. For $K = 10$, the authentication rate is about 23 authentications/minute. Thus BLACR-Express actually outperforms all existing schemes, with the tradeoff that users in the normal lane take about a minute to authenticate.

Performance at the user. For the computation at the user we show the costs are reasonable because users can precompute several values before authentication. In Figure 2(b) we can see that in BLACR-Normal with precomputation users would expect only a 13.3–22.1 second delay per authentication, and a 4.5–7.4 second delay in BLAC with 3,000–5,000 blacklist/meritlist entries. BLACR-Express on the other hand would take only 0.3–0.48 seconds. In comparison, PEREA takes much longer at the user (PEREA trades off efficient verification at the SP for more work at the user). For $K = 10$, and with 3,000–5,000 blacklist/meritlist entries, authentication takes around 22.4–26.9 seconds. Thus BLACR performs well on the user side, and BLACR-Express vastly outperforms all other schemes.

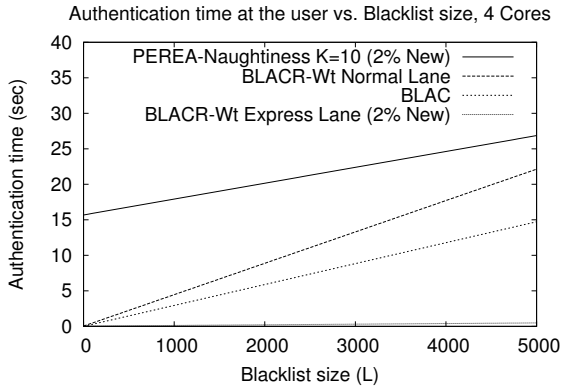
Practical considerations. If server costs must be kept low, and if the SP desires to have much larger blacklists, then PEREA with naughtiness offers a reasonable alternative at the cost of some accountability—misbehaving users must be identified within the revocation window, otherwise they get away with the misbehavior. Importantly, PEREA can apply reputation only to the last K authentications of a user. On the other hand, with BLACR we demonstrate that the costs are reasonable for several thousand entries in total on the meritlists and blacklists and the revocation guarantees are much stronger (a user's misbehavior can be identified at any time, days or months later, and the user can still be revoked). Also, the reputation-based anonymous revocation of BLACR offers a much richer language for revocation than PEREA with naughtiness. Finally, SPs willing to spend say $10,000/year on four 8-core servers and data transfer costs would be able to support 20,000 blacklist/meritlist entries, which we posit is large enough for most settings.

Schemes	Parties	Computation
BLAC	User	$(2L+8)$E1+ 2ET + 1P
(XDH Assumption made)	User (w/pre-computation)	$2L$E1
	SP	$(L+5)$E1 + 2ET + 2P
BLACR Normal	User	$\big((3-\zeta)L\big)$E1 + $\big((27-\zeta)L+11\ell m+22m+18\big)$EP1 + $\big((5-4\zeta)L+\ell m+4m+1\big)$EPT + 1P $\big(L+\ell m+1\big)$PP
	User (w/pre-computation)	$\big((3-\zeta)L\big)$E1 + 1EPT + 1P
	SP	$\big(12L+3\ell m+8m+5\big)$E1 + $4m$ EP1+ $\big(5L+5\ell m+5\big)$EPT + $\big(L+\ell m+1\big)$P +$\big(2L+2\ell m+2\big)$EP2
BLACR Express	User	$\big((3-\zeta)\Delta_L\big)$E1 + $\big((27-\zeta)\Delta_L+11\ell m+40m+30\big)$EP1 + $\big((5-4\zeta)\Delta_L+\ell m+4m+2\big)$EPT + 1P $\big(\Delta_L+\ell m+2\big)$PP
	User (w/pre-computation)	$\big((3-\zeta)L\big)$E1 + 1EPT + 1P
	SP	$\big(12\Delta_L+3\ell m+12m+7\big)$E1 + $4m$ EP1+ $\big(5\Delta_L+5\ell m+4m+9\big)$EPT + $\big(\Delta_L+\ell m+2\big)$P +$\big(2\Delta_L+2\ell m+4\big)$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

Table 4. Complexities analysis for BLAC, PEREA, and BLACR (ζ is the fraction of entries belongs to the user in the list) $\Delta_L = \partial_{\mathtt{pd}-1} \cup \partial^{\star}_{\mathtt{pd}}$



(a) Authentication time at the SP. The horizontal curves correspond to PEREA, which is linear in K and not L.



(b) Authentication time at User (includes precomputation)

Figure 2. Estimated authentication times at the SP and User and the cost for the SP.

6. Discussion

The authors of BLAC and PEREA already discuss the issues of rate-limiting authentications (all schemes need a standard rate-limiting scheme to limit the number of misbehaviors a user can perform before getting caught), concurrent sessions (standard techniques that can be used to limit users to one session at a time), timing attacks (as with other schemes, all users should wait for a standard amount of time to authenticate to avoid fingerprinting attacks), and Sybil attacks (all schemes must ensure users get only one credential). Here we discuss the following pertinent issues:

Express lane We assume there exists a subset of *active users* who can benefit from express lane authentication. In practical deployments, however, one could assume that *all users* are active by requiring them to refresh their credential for every time period. This refreshing can be accomplished by client software, much like an "auto update" check performed by popular software applications and operating systems. In this case the slow lane can still be used as a fallback mechanism, but the expected mode of operation is of much higher performance at the SP.

Unblacklisting Forgiving misbehaviors means that entries need to be removed from the blacklist. Removing entries invalidates the reputation proofs of express lane tokens, and thus all users must authenticate via the normal lane following unblacklisting. We assume such unblacklisting is performed at much larger epochs compared to the granularity of time periods. We also note that our policy-based approach allows users to perform other good actions at the

site to increase their reputation in order to gain access the SP again.

Efficient authentication We have already argued that BLACR is efficient enough for real-world deployments. Nevertheless, recent TTP-based schemes such as Nymble [20, 33], Nymbler [19], and Jack [23] aim to make the authentications as fast as possible (to the order of micro and milliseconds) at the SP. While BLACR cannot compete to be faster than such schemes, BLACR (and BLAC, EPID, and PEREA) offer the benefit of being TTP-free. Thus certain applications may require an extremely low impact on the SP, and the users may be willing to use TTP-based schemes given no other choice.

7. Conclusions

Several anonymous authentication schemes with varying degrees of accountable anonymity have been proposed in the past. In our work we focus on the paradigm of TTP-free schemes that offer both subjective blacklisting (where misbehaviors need to be flagged by humans) and anonymous revocation (where users can be blocked without knowing who they are). In this paradigm, more research is needed on the policy side, where service providers can articulate various forms of misbehaviors (or good behaviors) of anonymous users and thus deny access not on simple count-based policies but with a richer language. We make a step in this direction by generalizing TTP-free "reputation-based anonymous revocation", and open several possibilities for future improvements in this line of research.

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.

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] G. Ateniese, D. X. Song, and G. Tsudik. Quasi-efficient revocation in group signatures. In *Financial Cryptography*, volume 2357 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2002.

[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: Towards practical TTP-free revocation in anonymous authentication. *To appear in ACM Transactions on Information and System Security*. Also appears as Indiana University Bloomington Technical Report TR688, July 2011 `https://www.cs.indiana.edu/cgi-bin/techreports/TRNNN.cgi?trnum=TR688`.

[5] 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.

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

[7] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *ACM Conference on Computer and Communications Security*, pages 168–177. ACM, 2004.

[8] E. Brickell and J. Li. Enhanced privacy ID: a direct anonymous attestation scheme with enhanced revocation capabilities. In *WPES*, pages 21–30. ACM, 2007.

[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. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. 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. Springer, 2004.

[12] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003.

[13] 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.

[14] D. Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.

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

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

[17] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[18] 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.

[19] 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. Springer, 2010.

[20] 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.

[21] 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.

[22] J. Li, N. Li, and R. Xue. Universal accumulators with efficient nonmembership proofs. In *ACNS*, volume 4521 of *Lecture Notes in Computer Science*, pages 253–269. Springer, 2007.

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

[24] 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 2011)*. ACM, October 2011.

[25] L. Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292. Springer, 2005.

[26] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140, 1992.

[27] E. J. Schwartz, D. Brumley, and J. M. McCune. A contractual anonymity system. In *Proceedings of the Network and Distributed System Security Symposium*, 2010.

[28] P. F. Syverson, S. G. Stubblebine, and D. M. Goldschlag. Unlinkable serial transactions. In *Financial Cryptography*, volume 1318 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 1997.

[29] I. Teranishi, J. Furukawa, and K. Sako. *k*-times anonymous authentication (extended abstract). In *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*. Springer, 2004.

[30] 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.

[31] 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. ACM, 2008.

[32] 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.

[33] 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.

A. Formal Security Analysis

We use a simulation-based approach to define the security notions, in a similar sense as the model adopted by [4]. Due to page limitations, we present a simplified version of the analysis and full details will be made available in a separate technical report.

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' interaction are decided by another PPT algorithm, the environment, $\mathcal{E}$. $\mathcal{A}$ can communicate arbitrarily with $\mathcal{E}$. Informally speaking, BLACR 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 $\mathcal{A}$ in the real world or $\mathcal{S}$ in the ideal world. $\mathcal{S}$ has black-box access to $\mathcal{A}$.

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

- INIT. The system begins when $\mathcal{E}$ specifies the number of honest and dishonest users and SPs in the system.
  - *Real World.* The GM generates a key pair $(\mathsf{gpk}, \mathsf{gsk})$. All the SPs also generates their key pairs $(\mathsf{pk_{SP}}, \mathsf{sk_{SP}})$. gpk, $\mathsf{pk_{SP}}$ are made available to all players in the system.
  - *Ideal World.* The trusted party $\mathcal{T}$ initializes a database $\mathcal{U}$ which stores the registration status and authentication history of all the users.
- REG(i). $\mathcal{E}$ instructs user i to register with the GM. Note that this procedure is not anonymous in the view of the GM.
  - *Real World.* User i sends a request for registration to the GM. The user, as well as the GM, outputs individually the outcome of this transaction to $\mathcal{E}$. If user i has obtained a credential in previous registration event, an honest GM would reject the request. Likewise, an honest user would discard the second credential it obtains from the GM if it has successfully registered in a previous registration event.
  - *Ideal World.* User i sends a registration request to $\mathcal{T}$, who informs the GM user i would like to register and whether user i has obtained a credential before. GM returns its decision to $\mathcal{T}$, who forwards it back to the user. If the GM 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 GM, output individually the outcome of this transaction to $\mathcal{E}$.

- NP(j). $\mathcal{E}$ instructs an SP j to announce a new period.
  - *Real World.* SP j published the lists for period pd.
  - *Ideal World.* SP j sends a request to $\mathcal{T}$ to indicate the current period becomes pd, along with the set of lists for this period. If the list for period pd is not a superset of the list for pd $-$ 1, $\mathcal{T}$ returns reject. Otherwise, $\mathcal{T}$ stores this set of lists.
- AUTH(i, j)(Express/Normal). $\mathcal{E}$ instructs user i to authenticate with SP j in the express/normal lane.
  - *Real World.* User i conducts the authentication protocol with SP j using the express/normal lane.
  - *Ideal World.* User i sends a request to $\mathcal{T}$, who informs SP j some anonymous user requests an authentication using the express/normal lane. SP j replies with the lists $\partial_{\mathtt{pd}-1}, \partial^{\star}_{\mathtt{pd}}$. $\mathcal{T}$ forwards the lists back to user i and that whether i satisfy the authentication policy or not. User i then decide if he/she would continue. If yes, $\mathcal{T}$ informs SP j whether the anonymous user satisfies the authentication policy or not. SP j replies with accept or reject to $\mathcal{T}$, who forwards the reply to user i. Note that if user i has not authenticate in period pd $-$ 1, $\mathcal{T}$ would inform i that he does not satisfy the policy if $\mathcal{E}$'s instruction is to authenticate in the express lane. $\mathcal{T}$ stores all authentication history of all the users in his database.
- ADD($j, \mathcal{L}_i^{\diamond}$). $\mathcal{E}$ instructs the SP j to alter the list $\mathcal{L}_i^{\diamond}$ with an authentication event.
  - *Real World.* SP j adds the tickets corresponds to that authentication event if SP j outputs accept in that event.
  - *Ideal World.* $\mathcal{T}$ checks if SP j outputs accept in that authentication event and informs SP j the results of the check, who replies with accept or reject.

Ideal world BLACR provides all the desired security properties. Firstly, all the transactions, in the view of the SP, are anonymous. $\mathcal{T}$ only informs the SP some anonymous user would like to authenticate and thus anonymity is guaranteed. Secondly, $\mathcal{T}$ verifies if the authenticating user satisfies the authentication policy and thus authenticity, mis-authentication resistance and non-frameability are assured. Real world BLACR is secure if its behavior is the same as the ideal world BLACR. Thus, assuming $\mathsf{negl}(\lambda)$ is a negligible function in security parameter λ, we have the following definition of security for our system.

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* $\mathcal{A}$*). BLACR is secure if for all PPT algorithms* $\mathcal{E}$*,* $\mathcal{A}$*, the following expression holds:*

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

We analyze the security of our construction for BLACR by proving indistinguishability between an adversary's actions in the real world and the ideal world.

The idea of the proof is that given a real world adversary $\mathcal{A}$, we show how to construct an ideal world adversary $\mathcal{S}$ such that no PPT environment $\mathcal{E}$ can distinguish whether it is interacting with $\mathcal{A}$ or $\mathcal{S}$.

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 GM and a subset of SPs and users while in the second case, only a subset of SPs and users are dishonest. Note that the later is not a special case for the former as the capability of $\mathcal{A}$ does not necessarily aid or prevent $\mathcal{E}$ from distinguishing whether it is interacting with $\mathcal{A}$ or not. Indeed, the former case covers the security requirements of anonymity while the latter covers the requirement of authenticity, mis-authentication resistance and non-frameability. We sketch the strategy of how $\mathcal{S}$ can be constructed in this two cases. Firstly, $\mathcal{S}$ will be maintaining 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: GM is honest

- INIT.
  - *Representing Honest GM/SP to* $\mathcal{A}$*.*
    $\mathcal{S}$ generates the key pairs $(\mathsf{gpk}, \mathsf{gsk})$, $(\mathsf{pk}_{\mathsf{SP}}, \mathsf{sk}_{\mathsf{SP}})$ and gives gpk, $\mathsf{pk}_{\mathsf{SP}}$ to $\mathcal{A}$.
- REG(i).
  - *Representing dishonest user* i *to* $\mathcal{T}$ */ the honest GM to* $\mathcal{A}$*.*
    $\mathcal{S}$ extracts from $\mathcal{A}$ the value of x from C_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 and the credential to $\mathcal{A}$ and also stored that credential.
- AUTH(i, j).
  - *Representing dishonest user* i *to* $\mathcal{T}$*/ the honest SP* j *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 user i to 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 the value of x from C_x as well as the signature (A, e, y) and locate the value i such that x is used and that (A, e, y) is issued in the registration event.

Note that the output of $\mathcal{S}$ is always indistinguishable to $\mathcal{A}$ unless the following happen. We also briefly explain why such cases happens with negligible probability below.

1. During a REG event, $\mathcal{S}$ fails to extract from $\mathcal{A}$ the value x. This happens with negligible probability under the soundness property of $\mathfrak{S}_{\mathsf{iss}}(C_x)$.
2. During a successful AUTH, $\mathcal{S}$ fails to extract from $\mathcal{A}$ the values (A, e, x, y). This happens with negligible probability under the soundness property of $\mathfrak{S}_{\mathsf{Sig}}(C_x)$.
3. There exists a successful AUTH event from $\mathcal{A}$ such that $\mathcal{S}$ on behalf of an honest SP j outputs accept but $\mathcal{T}$ indicates the authenticating user does not satisfy the policy. This represents either $\mathcal{A}$ has been able to fake one of the proofs in the authentication, $\mathfrak{S}_{\mathsf{Sig}}$, $\mathfrak{S}_{\mathsf{WS\text{-}Adj}}$ or $\mathfrak{S}_{\mathsf{PoI}}$ or $\mathcal{A}$ can forge a signature (A, e, y) on a new value x which never appear before. All these happen with negligible probability under the soundness of the protocol.

Case 2: GM is dishonest

- INIT.
  - *Representing Honest SP to $\mathcal{A}$.*
    $\mathcal{S}$ receives gpk from $\mathcal{A}$. $\mathcal{S}$ generates the key pairs $(\mathsf{pk}_{\mathsf{SP}}, \mathsf{sk}_{\mathsf{SP}})$ and gives $\mathsf{pk}_{\mathsf{SP}}$ to $\mathcal{A}$.
- REG(i).
  - *Representing the dishonest GM to $\mathcal{T}$ / the 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 $\mathcal{A}$, $\mathcal{S}$ replies `reject` to $\mathcal{T}$ upon receiving the request from $\mathcal{T}$.
- AUTH(i, j).
  - *Representing the dishonest SP to $\mathcal{T}$/ the 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 $\mathfrak{S}_{\mathsf{Sig}}$, $\mathfrak{S}_{\mathsf{x}}$, $\mathfrak{S}_{\mathsf{WS\text{-}Adj}}$, $\mathfrak{S}_{\mathsf{PoI}}$ in the authentication protocol using a random value t. If $\mathcal{A}$ rejects the authentication, $\mathcal{S}$ replies `reject` to $\mathcal{T}$.

The only difference in the simulation provided by $\mathcal{S}$ to $\mathcal{A}$ and a real world $\mathcal{A}$ would face is the value of t. Under the DDH assumption, $\mathcal{A}$ would not notice such difference. Other than that, the simulation provided to $\mathcal{A}$ is perfect due to the zero-knowledgeness of the ZKPoK protocols. 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 to that of $\mathcal{A}$.

Based on this two strategy in the construction of $\mathcal{S}$, we have:

$$\mathbf{Real}_{\mathcal{E},\mathcal{A}}(\lambda) = \mathbf{Ideal}_{\mathcal{E},\mathcal{S}}(\lambda).$$

That is, our construction of BLACR is secure according to Definition 1. In the full version of our paper, detailed proof of security will be given. As a final remark, as the construction of such simulator $\mathcal{S}$ requires an extraction for the protocol $\mathfrak{S}_{\mathsf{Sig}}$ in each authentication, the scheme is secure only when the authentication is done in an sequential manner. Otherwise, the complexity of $\mathcal{S}$ would blow to exponentiating in the number of authentication event.