BLACR: TTP-Free Blacklistable Anonymous Credentials with Reputation

Man Ho Au
Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Australia

Patrick P. Tsang
Department of Computer Science
Dartmouth College, USA

Apu Kapadia
School of Informatics and Computing
Indiana University Bloomington, USA

Willy Susilo
Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Australia

Indiana University Technical Report TR695

May 2011

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 such misbehaving users. Recently, schemes such as BLAC, EPID, 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, BLAC, EPID, and PEREA allow only a basic form of revocation amounting to "revoke anybody on the blacklist". Recently BLAC was extended to support d-strikes-out policies that revokes anybody who has d or more entries on the blacklist. In this paper we significantly advance this concept and make the first attempt to generalize *reputation-based anonymous revocation* through our proposed scheme called BLACR. We show how various negative or positive scores can be assigned to anonymous sessions across various categories of misbehavior resulting in users being blocked based on their reputation scores. We show how various relevant policies can be instantiated in BLACR and the workload for authenticating users is reasonable for web services.

1 Introduction

Anonymity can give some users the license to misbehave. For example, using an anonymizing network like Tor [20], 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 Indiana University). 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, 7, 19, 26] and dynamic accumulators [2, 8, 14, 27, 29], and Nymble systems [25, 37, 24, 28, 31].

TTP-free, anonymous revocation 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 [34, 36], EPID [9], and PEREA [35, 4]. These schemes allow *anonymous revocation*, where misbehaving users can be revoked without the involvement of a TTP. Thus there is no entity who can deanonymize or link a user's anonymous authentications, but yet SPs can prevent such users from returning. Users are guaranteed strong privacy (they can never be deanonymized), and SPs are spared from future accesses by the user.

Related schemes that reduce or eliminate TTPs Recently Camenisch et al. [13] proposed a TTP-free scheme in the case where revocations are relatively rare and users can be revoked only at set epochs. While their scheme is useful when those tradeoffs are acceptable, we focus on scenarios where credential revocations are immediate and relatively more frequent. The only other TTP-free schemes are based on "double spending" of e-cash [18] (generalized to n-times spending [33]), where only if a user authenticates twice (or n times), the user's identity gets 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. [31] propose *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 it is necessary for a *human assessment* of whether a misbehavior occurred, and automated techniques such as contractual anonymity may not be feasible in these cases.

In our work, we continue with the philosophy behind BLAC, EPID and PEREA, to create schemes with *anonymous revocation* and *subjective blacklisting*, where users can never be deanonymized by the SP or some other TTP (even if it is "highly trusted"), and yet can be blocked from future accesses. We believe such schemes are well-suited to environments that support anonymous publishing, for example.

Recent anonymous revocation extensions 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. Recently, BLAC was extended [36] to incorporate d-strikes-out policies. For example, $d = 3$ enacts a "three strikes out policy" where three (or more) misbehaviors from a user results in revocation of that user. PEREA was then extended [4] to generalize d-strikes-out policies to a weighted version called *naughtiness* policies. Each misbehavior is assigned a *severity*, and users whose total severity (their *naughtiness*) exceeds a certain naughtiness threshold are denied authentication. In all cases, the server learns only whether an authentication for the anonymous user succeeds or not.

Our contributions In this paper we make the first significant effort to extend TTP-free anonymous revocation with subjective blacklisting to more general behavior-based policies. Our approach gives SPs a rich language to characterize acceptable and unacceptable uses of their services while supporting anonymous revocation. We make the following contributions:

- We generalize the concept of anonymous revocation and make the first attempt to formalize *reputation-based anonymous revocation* based on a rich policy language for anonymous revocation.

- We show that such a scheme can be realized by extending BLAC, which unlike PEREA does not require misbehaviors to be identified within a "revocation window." We name our construction BLACR (BLacklistable Anonymous Credentials with Reputation).

- 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.
- While improving on the linear time complexity (in the size of the blacklist) for authentications in BLAC and BLACR remains an open and important problem, through a quantitative analysis we show that BLACR can indeed be used in practical settings to support reputation-based anonymous revocation.

2 Overview of Approach

Intuition behind the construction of BLAC/EPID BLAC and EPID use the following idea: an anonymous authentication of a user with credential usk results in a *ticket* of the form b_i^{usk} that 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 usk of the ticket b_i^{usk}, 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 b_i^{usk} for that session into the blacklist. Users authenticating to the SP must prove their credential is not associated with any ticket on the blacklist. Such proofs are done in zero knowledge (usk is not revealed to the SP) and bound to their issued credentials (users cannot forge their own credential usk). In particular it is possible to prove in zero knowledge the "inequality of discrete log", that is, the user's credential usk' does not correspond to the discrete log usk of each entry b_i^{usk} on the blacklist.

BLACR: Reputation-based anonymous revocation In essence, BLACR adds a *score* parameter to each entry in the blacklist indicating the *severity* of the misbehavior, and SPs can require the overall score of an authenticating user satisfy a certain threshold. The challenge is in proving the user satisfies this requirement in zero knowledge, i.e., the server learns only whether the user satisfies that threshold and nothing else. BLACR actually features both positive and negative scores for good and bad behaviors (in a *meritlist* and *blacklist* respectively), resulting in an overall *reputation score* for each user, and thus the name BLACR (BLacklistable Anonymous Credentials with Reputation). Furthermore, servers can score reputation across different *categories*. 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. On the other hand, content rated highly by other users could result in a commensurate reward (positive score). Within the category of *comments*, inappropriate language, racist or intimidating comments could be considered to be much more severe than puerile posts containing offensive words. 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. 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 (NOT) 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. Again, we emphasize the SP learns only whether the entire policy was satisfied or not, and learns no other information (e.g., what subexpressions were satisfied, or what the user's specific reputation is in one or more categories) except what it can already infer from the result.

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 penalize 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, thus further incentivizing many more good behaviors.

Thus in "BLACR-Weighted", for each category the SP can specify a set of adjusting factors $\Delta_1, \Delta_2, \ldots, \Delta_L$ 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 $\{(1, \tau_1, s_1), (2, \tau_2, s_2), \ldots, (8, \tau_8, s_8)\}$. Each tuple represents the session i, ticket τ_i for the session, and the score s_i for the misbehavior in that session. 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 factor $\Delta_1, \Delta_2, \Delta_3, \Delta_4, \Delta_5$ for the list. In BLACR-Weighted, the 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 $(4, \tau_4, s_4)$.

Note that 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, without revealing to the server anything other than whether the entire policy was satisfied or not. Thus this extended construction of BLACR is also a significant contribution of our work.

Intuition behind the construction of BLACR BLACR follows the idea of BLAC and EPID in which each authentication session results in a *ticket* of the form b_i^{usk}. It supports a more sophisticated access policy as follows. When an SP wants to *score* a user associated with a session, it inserts the ticket b_i^{usk} for that session together with a score s_i into the list. 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) $t_i \neq b_i^{\mathsf{usk}}$ and C_i is a commitment of 0; or (2) $(t_i = b_i^{\mathsf{usk}})$ and C_i is a commitment of s_i. Finally, the user proves to the SP that the sum of the values committed in C_i is above the required reputation threshold. The idea can be further extended to multiple categories by having separate lists for individual category.

3 Security Goals and Syntax

3.1 Security goals

We now give informal definitions of the various security properties that a construction of the BLACR system must possess. Their formal definition will be given in Appendix A. It is similar to that of BLAC.

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.

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.

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

3.2 Notation

Tickets, lists and scores Define the ticket generation function which takes as input the user's secret key usk and randomness b, and output a value $t \leftarrow \mathsf{T}(b, \mathsf{usk})$. The tuple $\tau = (b, t)$ is called a ticket. As identified in BLAC, T needs to be one-way, injective and its outputs from the same input should be unlinkable. 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. $\mathcal{L}$ is a blacklist (resp. meritlist) if all score are negatives (resp. positive). For the ease of representation, define a *score* function S:

$$\mathsf{S} : (\mathcal{L}, \mathsf{usk}) \mapsto \sum_{i \in [|\mathcal{L}|], t_i = \mathsf{T}(b_i, \mathsf{usk})} s_i$$

where $[n]$ denotes the set $\{1, \ldots, n\}$ for any positive integer n.

The value $\mathsf{S}(\mathcal{L}, \mathsf{usk})$ is called the *list score* of a user with credential usk with respect to the list $\mathcal{L}$, and is the sum of all the scores on the list for tickets corresponding to that user. Looking ahead, the *reputation* of a user is his/her merit list score minus his/her blacklist score.

Categories and reputation The SP scores behaviors in various categories $\{c_1, c_2, \ldots, c_m\}$, where m is the number of categories. The SP maintains a blacklist $\mathcal{L}_i^-$ and meritlist $\mathcal{L}_i^+$ for each *category* c_i. The reputation of a user in a category c_i, denoted as R_i, is thus the difference of the list score of $\mathcal{L}_i^+$ and $\mathcal{L}_i^-$:

$$R_i = \mathsf{S}(\mathcal{L}_i^+, \mathsf{usk}) - \mathsf{S}(\mathcal{L}_i^-, \mathsf{usk})$$

Weighted scores In practice, *score* could be weighted based on the number of times the user has been put on the list to assign more or less penalty or reward for the repeated behaviors in a particular category. To generalize this idea, we define a set of values called 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 score multiplied by the appropriate adjusting factor. Specifically, for a list $\mathcal{L}$, a set of adjusting factors $\mathcal{D}$ and a credential usk, we define a function, called *weighted score*, as S':

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

where Δ_i is defined to be Δ_k for all $i > k$. This function states that the weights must be applied to the tickets corresponding to the user in the correct order, and then the weighted scores are added up to get the weighted list score for that list.

3.3 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 pair of 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 in the system.

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.

Authentication This protocol is executed between a user Alice with credential usk and an SP Bob. The input of Alice is her credential. The input to Bob is his set of meritlists/blacklists $\{\mathcal{L}_i^+, \mathcal{L}_i^-\}_{i=1}^{\ell}$ and a set of thresholds $n_1, \ldots, n_\ell$ and a policy `Pol`. Policy `Pol` is a combination of sub-policies in DNF form, where sub-policy $\mathcal{P}_i$ is a boolean function defined over two lists, a user credential and a threshold, whose truth value evaluates to 1 if

$R_i \geq n_i$ and 0 otherwise. `Pol` can contain negations $\neg P_i$ of sub-policies as well, in which case the truth value of $\neg \mathcal{P}_i$ evaluates to 1 if $R_i < n_i$ and 0 otherwise. Thus for each category, the corresponding sub-policy specifies the user's reputation should be above a certain threshold. The policy can then block users based on a boolean combination of behaviors across various categories. If c_1, c_2, c_3 represents respectively the categories for *video content*, *tagging* and *commenting*, the policy in Section 2 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 threshold for the respective categories. It thus consists of three sub-policies $\mathcal{P}_i = (c_i, n_i)$ arranged in two conjunctive clauses $\mathcal{P}_1$ and $(\mathcal{P}_2 \wedge \mathcal{P}_3)$. We do not require full disjunctive normal form, meaning that the same sub-policy may appear more than once in different conjunctive clauses. Note that negation of a policy, say $\neg\mathcal{P}_1$, is a boolean function which evaluates to 1 if and only if the reputation in c_1 strictly smaller than the required threshold n_1. As an example, the SP may enforce the following policy $(\mathcal{P}_1 \wedge \mathcal{P}_2) \vee (\neg\mathcal{P}_1 \wedge \mathcal{P}_3)$, meaning 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*.

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.

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 Alice's credential.

List management This is a suite of three algorithms: *Extract*, *Add* and *Remove*, which are executed by SPs for managing their lists. On input of an authentication protocol transcript, *Extract*(ϖ) extracts and 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 $\mathcal{L}$ while *Remove*$((\tau, s), \mathcal{L})$ deletes an existing entry (τ, s) from a list $\mathcal{L}$.

4 Our Construction

4.1 Building blocks

4.1.1 Proofs of knowledge

In a *Zero-Knowledge Proof of Knowledge (ZKPoK)* protocol [22], 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 [23], that are secure under the *Random Oracle (RO)* Model [5]. BLACR utilizes the ZKPoK that x does *not* equal $\log_b t$, denoted as $PK\{(x) : y = g^x \wedge t \neq b^x\}$ due to Camenisch and Shoup [16].

4.1.2 Commitment scheme

Our construction uses the well known non-interactive commitment scheme due to Pedersen [30], 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 $a + b$.

4.1.3 Credential signature scheme

We employ the signature scheme proposed by Au et al. [3], which is based on the schemes of Camenisch and Lysyanskaya [15] and of Boneh et al. [7], 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 $w = h^\gamma$. A signature over a tuple of values $(m_1, \ldots, m_k)$ is a tuple (A, e, y) such that $A = (g_0 g_1^{m_1} \cdots g_k^{m_k} g_{k+1}^y)^{\frac{1}{\gamma+e}}$. They also derive a Σ-protocol for the demonstration of a message-signature pair.

$$\mathrm{PK}\{(A, e, s) : \hat{e}(A, wh^e) = \hat{e}(g_0 g_1^{m_1} \cdots g_k^{m_k} g_{k+1}^y, h)\}$$

4.1.4 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. [11]. 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 specifically useful when the range is small.

4.2 Our construction of BLACR

4.2.1 Parameters

Let λ be a sufficiently large security parameter. Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair with computable isomorphism ψ as discussed such that $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for some prime p of λ bits. 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 $H_0 : \{0,1\}^* \rightarrow \mathbb{G}$ and $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ be collision-resistant hash functions.

4.2.2 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 $\mathsf{gpk} = (w)$.

4.2.3 SP Setup

Each SP publishes his unique identity string `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 $(\{0,1\}^\lambda, \mathbb{G}, [s_{\max}])$ were $s_{\max}$ is the maximum score associated with a misbehavior or good behavior.

4.2.4 Registration

Upon successful termination of this protocol between a user Alice and the GM, Alice obtains a credential signature (A, e, y) on Alice's secret value x. We note that x and y are known only to Alice (i.e., and not the GM). The private input to the GM is the group secret key gsk.

1. The GM sends m to Alice, where $m \in_R \{0,1\}^\lambda$ is a random challenge.

2. Alice sends a pair (C, Π_1) to the GM, where $C = g_1^x g_2^{y'} \in \mathbb{G}_1$ and Π_1 is a signature proof of knowledge of
$$SPK_1 \left\{ (x, y') : C = g_1^x g_2^{y'} \right\} (m) \tag{1}$$
on challenge m, which proves that C is correctly formed.

3. The GM returns `failure` if the verification of Π_1 returns `invalid`. Otherwise the GM sends Alice a tuple (A, e, y''), where $e, y'' \in_R \mathbb{Z}_p$ and $A = (g_0 C g_2^{y''})^{\frac{1}{e+\gamma}} \in \mathbb{G}_1$.

4. Alice computes $y = y' + y''$. She returns `failure` if $\hat{e}(A, w h_0^e) \neq \hat{e}(g_0 g_1^x g_2^y, h_0)$. Otherwise she outputs $\mathsf{usk} = (A, e, x, y)$ as her credential.

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

4.2.5 Authentication

During an execution of this protocol between a user Alice and the SP, Alice's private input is her credential $\mathsf{usk} = (A, e, x, y)$. In our construction, the ticket generation function T is defined as $t := H(b||\mathtt{sid})^x$.

Let $\mathtt{sid} \in \{0,1\}^*$ be the string that uniquely identifies the SP. When the protocol terminates, the SP outputs `success` or `failure`, indicating whether the SP should consider the authentication attempt successful.

1. *(Challenge.)* The SP sends to Alice the lists for each category as well as their corresponding thresholds and a random challenge $(\mathcal{L}_1^+, \mathcal{L}_1^-, n_1, \ldots, \mathcal{L}_\ell^+, \mathcal{L}_\ell^-, n_\ell, m)$ as well as the policy $\mathtt{Pol}$.

2. *(Inspection.)* Alice computes, for each category c_i, $S_i^+ = \mathsf{S}(\mathcal{L}_i^+, (A, e, x, y))$ and $S_i^- = \mathsf{S}(\mathcal{L}_i^-, (A, e, x, y))$. This involves checking if the entries on the list belongs to her. A direct approach would thus require $|\mathcal{L}_i^+| + |\mathcal{L}_i^-|$ exponentiations for category c_i since it requires Alice to test if $t_j \overset{?}{=} H(b_j||\mathtt{sid})^x$ for each entry (t_j, b_j) on the list. To speed things up, we assume Alice stores all her past tickets and simply checks, for each ticket, if it is present on the lists, and thus does not need to perform these exponentiations. She computes the truth value for each sub-policy $\mathcal{P}_i$ by testing if

$$S_i^+ - S_i^- \overset{?}{\geq} n_i$$

   For the negation of a sub-policy $\neg\mathcal{P}_i$, Alice tests if $S_i^+ - S_i^- < n_i$. She then checks if $\mathtt{Pol}$ evaluates to 1. If not, she returns as `failure`, indicating that she is revoked.

3. *(Proof Generation.)* If Alice is not revoked, she returns to the SP a pair (τ, Π_2), where $\tau = (b, t := H(b||\mathtt{sid})^x) \in \{0,1\}^\ell \times \mathbb{G}$ is the ticket associated with the current authentication, and Π_2 is a signature proof of knowledge that τ is correctly formed and that $\mathtt{Pol}$ evaluates to 1 with input usk. Here $b \in_R \{0,1\}^\lambda$ is a random value chosen by Alice.

We defer the details of Π_2 to the next subsection since it is quite involved. The SP stores ticket τ extracted from the transcript, along with information logging Alice's activity within the authenticated session.

4.2.6 List management

The three 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. *Remove*$(\mathcal{L}, (\tau, s))$ returns list $\mathcal{L}'$, which is the same as the input list $\mathcal{L}$, except with all entries equal to the input ticket (τ, s) dropped.

4.3 Details of the authentication protocol

Express the policy $\mathtt{Pol}$ in DNF form as

$$\mathtt{Pol} : \mathtt{Cl}_1 \vee \mathtt{Cl}_2 \ldots \vee \mathtt{Cl}_\ell$$

where each conjunctive clause $\texttt{Cl}_i$ is of the form $(\mathcal{P}_{i1} \wedge \mathcal{P}_{i2} \wedge \ldots)$ and each $\mathcal{P}_{ij}$ is a sub-policy requiring the authenticating user to have a reputation equal or higher than a threshold n_{ij} in category c_{ij}. The policy can contain negations too. The negation $\neg\mathcal{P}_{ij}$ of a sub-policy $\mathcal{P}_{ij}$ is defined as requiring the authenticating user to have a reputation lower than the threshold n_{ij}.

Recall that the goal of Π_2 is to demonstrate that $t = H(b||\texttt{sid})^x$ is correctly formed, and that Alice satisfies the policy $\texttt{Pol}$. For the ease of representation, we use $\hat{b}$ to represent the value $H(b||\texttt{sid})$.

Π_2 consists of several parts, and we describe each of them in detail. In the first part, Alice demonstrates that she is a certified user and that t is correctly formed. This is done by the generation of the following proof, Π_x.

$$SPK\left\{(A, e, x, y) : \hat{e}(A, wh_0^e) = \hat{e}(g_0 g_1^x g_2^y, h_0) \ \wedge\ t = \hat{b}^x\right\}(m) \tag{2}$$

Next, we describe the way Alice demonstrates, in zero-knowledge, that she satisfies an individual sub-policy $\mathcal{P}$ in Σ-protocol. To prevent colluding users from polling their credentials, each proof consists of the knowledge of the discrete logarithm of the ticket t which assures the verifier that the same x is being used for all sub-policy. Using standard techniques, one can easily combine those Σ-protocols and prove that the whole policy $\texttt{Pol}$ is satisfied without revealing what sub-policies were satisfied (beyond what can already be inferred by the final result).

For a sub-policy $\mathcal{P}_k$ which requires the authenticating user to have reputation in category c_k higher than or equal to a threshold n_k, define an index set $\mathcal{I} : \{\iota | (b_\iota, t_\iota, \cdot) \in \mathcal{L}_k^+, H(b_\iota||\texttt{sid})^x = t_\iota\}$. This set corresponds to the tickets associated with Alice in list $\mathcal{L}_k^+$. We use L_k^+ to denote the size of $\mathcal{L}_k^+$.

1. Produce auxiliary commitments for each score on the list $\mathcal{L}_k^+$

$$\mathsf{aux}_k^+ = (C_{k1}^+ \ldots, C_{kL_k^+}^+)$$

as follows. For all $\iota \in \mathcal{I}$, compute $C_{k\iota}^+$ as the commitment of s_ι. For $i \in [L_k^+]\backslash\mathcal{I}$, compute $C_{k\iota}^+$ as the commitment of 0.

2. Generate a proof Π_k^+ to demonstrate the correctness of aux_k^+.

$$SPK\left\{\begin{array}{l}(x) : t = \hat{b}^x \ \wedge \\ \left(\begin{array}{l}\left(t_\iota \neq \hat{b}_\iota^x \ \wedge\ C_{k\iota}^+ = \textsc{Cmt}(0)\right) \vee \\ \left(t_\iota = \hat{b}_\iota^x \ \wedge\ C_{k\iota}^+ = \textsc{Cmt}(s_\iota)\right)\end{array}\right)_{\iota=1}^{L_k^+}\end{array}\right\}(m) \tag{3}$$

3. Do the same for list $\mathcal{L}_k^-$, which results in the set of auxiliary commitments

$$\mathsf{aux}_k^- = (C_{k1}^-, \ldots, C_{kL_k^-}^-)$$

and a proof Π_k^- which demonstrate the correctness of aux_k^-.

4. Based on aux_k^+ and aux_k^-, Alice computes the commitment of her reputation in category c_k as

$$R_k = \prod_{\iota=1}^{L_k^+} C_{k\iota}^+ / \prod_{\iota=1}^{L_k^-} C_{k\iota}^-$$

5. Finally, Alice generates a proof Π_k^R which demonstrates she satisfies the sub-policy $\mathcal{P}_k$ regarding category c_k. Note that for negation of $\mathcal{P}_k$, Alice generates a proof that demonstrates $S < n_k$.

$$SPK\left\{(S) : R_k = \textsc{Cmt}(S) \ \wedge \ S \geq n_k\right\}(m) \tag{4}$$

6. Define aux_k as $\mathsf{aux}_k^+||\mathsf{aux}_k^-$, a proof that a sub-policy $\mathcal{P}_k$ is satisfied thus consists of

$$\Pi_k = (\mathsf{aux}_k, b, t, \Pi_k^+, \Pi_k^-, \Pi_k^R)$$

4.4 BLACR-Weighted extension

We highlight the way Alice demonstrates, in zero-knowledge, that she satisfies individual sub-policies $\mathcal{P}_k$ in Σ-protocols in BLACR-Weighted.

Firstly, for every category c_k, the SP has to publish the set of adjusting factors $\mathcal{D}_k^+$ and $\mathcal{D}_k^-$. 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^-)). It is required that the signatures are generated using a different key pairs for each $\mathcal{D}$.

As in BLACR-Unweighted, we use an index set $\mathcal{I}$ to index the tickets associated with Alice in list $\mathcal{L}_k^+$.

1. Produce auxiliary commitments

$$\mathsf{aux}_k^+ = (C_{k1}^+, \tilde{C}_{k1}^+, \ldots, C_{kL_k^+}^+, \tilde{C}_{kL_k^+}^+)$$

as follows. For all $\iota \in \mathcal{I}$, compute $C_{k\iota}^+$ as the commitment of $(\Delta_{\kappa_\iota}^+ * s_\iota)$, $\tilde{C}_{k\iota}^+$ as the commitment of 1, where $\Delta_{\kappa_\iota}^+$ is the appropriate adjusting factor of Alice for the ι-th entry. Thus, $C_{k\iota}^+$ is the commitment of the weighted score of Alice of the ι-th entry. For $i \in [L_k^+]\backslash\mathcal{I}$, compute $C_{k\iota}^+$, $\tilde{C}_{k\iota}^+$ as commitments of 0.

2. Generate a proof Π_k^+ to demonstrate the correctness of aux_k^+. We give the intuition of how Π_k^+ shows $C_{k\iota}^+$ is a commitment of the weighted score of the authenticating user. Firstly, $\tilde{C}_\iota^+$ acts as a boolean flag, hidden from the SP, indicating if τ_ι is a ticket from the authenticating user. Thus, for all ι such that $t_\iota \neq \hat{b}_\iota^x$, $C_{k\iota}^+$ as well as $C_{k\iota}^+$ should be a commitment of 0. On the other hand, when $t_\iota = \hat{b}_\iota^x$, $\tilde{C}_{k\iota}^+$ is a commitment of 1. Due to the homomorphic property of the commitment scheme, $\prod_{j=1}^{\iota} \tilde{C}_{k\iota}^+$ is a commitment of a value κ_ι where κ_ι is the number of times the authenticating user has been put on the list up to the ι-th entry. Thus, the correct adjusting factor for the weighted score of this entry is $\Delta_{\kappa_\iota}^+$. Recall that σ_i^+ is the signature from the SP on the tuple (Δ_i^+, i). Thus, the proof that Verify$(\sigma_{\kappa_\iota}^+, \Delta_{\kappa_\iota}^+, \kappa_\iota) = 1$ binds the value of κ_ι to the appropriate

adjusting factor $\Delta^+_{\kappa_\iota}$. Finally, the proof demonstrates that the correct weighted score of this entry, $\Delta^+_{\kappa_\iota} s_\iota$, is committed in $C^+_{k\iota}$.

$$SPK\left\{\begin{array}{l}(x,\{\sigma^+_{\kappa_\iota},\Delta^+_{\kappa_\iota},\kappa_\iota\}) : t=\hat{b}^x \wedge \\ \left(\begin{array}{l}\Big(t_\iota \neq \hat{b}^x_\iota \;\wedge\; C^+_{k\iota}=\textsc{Cmt}(0) \\ \tilde{C}^+_{k\iota}=\textsc{Cmt}(0)\Big)\vee \\ \Big(t_\iota=\hat{b}^x_\iota \;\wedge\; \tilde{C}^+_{k\iota}=\textsc{Cmt}(1)\wedge \\ \prod_{j=1}^{\iota}\tilde{C}^+_{kj}=\textsc{Cmt}(\kappa_\iota)\wedge \\ \text{Verify}(\sigma^+_{\kappa_\iota},\Delta^+_{\kappa_\iota},\kappa_\iota)=1\wedge \\ C^+_{k\iota}=\textsc{Cmt}(\Delta^+_{\kappa_\iota}s_\iota)\Big)\end{array}\right)_{\iota=1}^{L^+_k}\end{array}\right\}(m) \tag{5}$$

We provide full details on the implementation of this SPK in Appendix B.

3. Do the same for the list $\mathcal{L}^-_{k_{ij}}$. This results in another set of auxiliary commitments

$$\mathsf{aux}^-_k=(C^-_{k1},\tilde{C}^-_{k1},\ldots,C^-_{kL^-_k},\tilde{C}^-_{kL^-_k})$$

   and a proof Π^-_k which demonstrate the correctness of aux^-_k.

4. Based on aux^+_k and aux^-_k, Alice computes the commitment of her reputation in category c_k as

$$R_k=\prod_{\iota=1}^{L^+_k}C^+_{k\iota}/\prod_{\iota=1}^{L^-_k}C^-_{k\iota}$$

5. Finally, Alice generates a proof Π^R_k which demonstrates she satisfies the sub-policy $\mathcal{P}_k$. (For its negation $\neg\mathcal{P}_k$, Alice demonstrates $S<n_k$.)

$$SPK\{(S):R_k=\textsc{Cmt}(S)\;\wedge\;S\geq n_k\}(m) \tag{6}$$

6. Define aux_k as $\mathsf{aux}^+_k||\mathsf{aux}^-_k$, a proof that a sub-policy $\mathcal{P}_k$ is satisfied thus consists of

$$\Pi_k=(\mathsf{aux}_k,b,t,\Pi^+_k,\Pi^-_k,\Pi^R_k)$$

4.5 Security analysis

BLACR possesses mis-authentication resistance, authenticity, anonymity and non-frameability. Formal security analysis is presented in Appendix A.

Schemes	Authentication Efficiency				
	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	$O(L)$	$O(L)$	$O(L)$ +	$O(L)$	$O(L)$

Table 1: Asymptotic Complexities

5 Performance Evaluation

5.1 Complexity analysis

Table 1 summarizes the performance of BLACR in comparison with existing schemes. The asymptotic complexity of BLACR is the same as that to BLAC. Generating the proofs takes $O(L)$ time for the user in BLAC and BLACR, and $O(K\Delta_L)$ in PEREA as each witness must be updated Δ_L times. K is the size of the revocation window (the number of authentications before which a misbehavior must be caught to result in a revocation), and $\Delta(L)$ 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, whereas PEREA requires only $O(K)$ computation at the server.

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.

5.2 Quantitative analysis

5.2.1 Data transfer

Assume the score of each entry is within 5 bits and the threshold is within 10 bits, and further assume that the adjusting factors in BLACR-Weighted is relatively stable. That is, they can be treated as public parameter. The the total communication cost for BLACR with and without XDH assumption is given in Table 4 assuming a security parameter of 224. The constant ℓ is the number of sub-policies in BLACR. We assume $\ell = 10$ in our analysis, and the lists for each category are equal in size. For both BLACR-Unweighted and BLACR-Weighted, downloading a blacklist is under 110KB with 2,000 entries on the blacklist, and uploading the proof is similar in size to uploading a medium-size JPEG photo (714KB for BLACR-Unweighted and 1.5 MB for BLACR-Weighted). As a baseline, we note the costs for such data transfers at Amazon EC2 amounts to about 15–30 millicents per authentication and thus the transfer costs of authentications are negligible.

5.2.2 Computation

Table 3 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 preprocessing 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 1. Let L be the size of the blacklist. For our evaluation of BLACR, we assume there are ℓ sub-policies and each the meritlist and blacklist in each sub-policy are of size $L/(2\ell)$. Assume a total of B-out-of-L tickets belongs to the user and we further assume it is evenly distributed. Let $A = L - B$. Thus, for each meritlist or blacklist, there are $B/(2\ell)$ tickets belongs to the authenticating user. 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 following timings 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 $|p| = 224$, bundled with the PBC library. The following table summarizes our experimental results. Note that $\mathbb{G}$ could be an arbitrary group. We can choose $\mathbb{G}$ to be $\mathbb{G}_1$ or $\mathbb{G}_T$. The former gives much better performance. However, it requires an additional assumption that the DDH problem is hard in $\mathbb{G}_1$. This is formally known as the external Diffie-Hellman (XDH) assumption.

The figures for $EN1$ and $EN2$ are taken from [4], which runs on the same machine directly on the Windows platform. The test code is written in C based on the MIRACL library[3] (version 5.4.2). The modulus N is taken to be 2048 bits. The small exponent is taken to be 160 bits.

Operations	Legend	Mode	Time
$\mathbb{G}_1$-EXP	E1	Multi-based EXP	3.489ms
	EP1	With/Pre-Processing (fixed single base)	0.528ms
$\mathbb{G}_2$-EXP	E2	Multi-based EXP	28.281ms
	EP2	With/Pre-Processing (fixed single base)	4.134ms
$\mathbb{G}_T$-EXP	ET	Multi-based EXP	7.285ms
	EPT	With/Pre-Processing (fixed single base)	1.265ms
Pairing	P	Normal	24.568ms
	PP	With/Pre-Processing (one input fixed)	19.117ms
EXP modulo N	EN2	With/Pre-Processing (fixed base)	8.25ms
	EN1	Small Exponent (Without/Pre-Processing)	5.96ms
Size of element in $\mathbb{G}_1$	N/A	Normal	448 bits
	$[\mathbb{G}_1]$	Compressed	225 bits
Size of element in $\mathbb{G}_2$	$[\mathbb{G}_2]$	Normal	1344bits
Size of element in $\mathbb{G}_T$	$[\mathbb{G}_T]$	Normal	1344bits

Table 2: Benchmark of different operations

Because of the linear cost of authentication, it is currently infeasible for large websites to score every single session (potentially resulting in millions of entries in the meritlists and

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

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

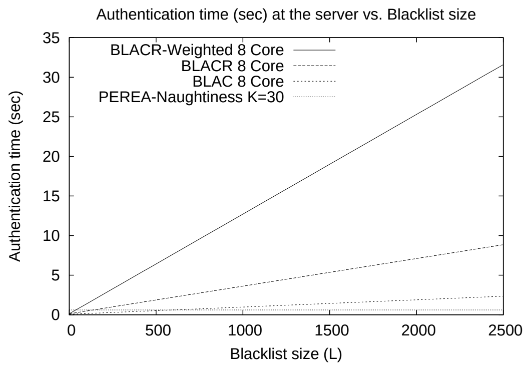
Schemes	Parties	Computation
BLAC	User	7E1+ $(2L+3)$ET + 1P
	User (w/pre-computation)	$2L$ET
	SP	4E1 + $(L+3)$ET + 2P
BLAC	User	$(2L+8)$E1+ 2ET + 1P
(XDH Assumption made)	User (w/pre-computation)	$2L$E1
	SP	$(L+5)$E1 + 2ET + 2P
BLACR-Unweighted	User	$(2\ell+1)$E1 + $(5\ell+5)$EP1 + $(7L+4)$ET + $(2L+4A+4\ell)$EPT + $(\ell+1)$PP
	User (w/pre-computation)	$(2L+A)$ET
	SP	$(8L+3\ell+3)$ET + 1EPT + $(2\ell+2)$E1 +$(2\ell+2)$PP
BLACR-Unweighted	User	$(7L+2\ell+3)$E1 + $(2L+4A+7\ell+5)$EP1 + 2ET + (2ℓ)EPT + $(\ell+1)$PP
(XDH Assumption made)	User (w/pre-computation)	$(2L+A)$E1
	SP	$(8L+3\ell+3)$E1 + 1EPT + $(2\ell+2)$ET +$(2\ell+2)$PP
BLACR-Weighted	User	$(L+A+2\ell+1)$E1 + $(2B+5\ell+5)$EP1 + $(10L+A+4)$ET + $(10L+4\ell)$EPT + $(2L+\ell+1)$PP
	User (w/pre-computation)	$(2L+A)$ET
	SP	$(13L+3\ell+3)$ET + $(2L+1)$EPT + $(2L+2\ell+2)$E1 +$(2L+2\ell+2)$PP
BLACR-Weighted	User	$(9L+2A+2\ell+3)$E1 + $(8L+4B+7\ell+5)$EP1 + $(2L+2)$ET + $(2A+2\ell)$EPT + $(2L+\ell+1)$PP
(XDH Assumption made)	User (w/pre-computation)	$(2L+A)$E1
	SP	$(13L+3\ell+3)$E1 + $(2L+1)$EPT + $(2L+2\ell+2)$ET +$(2L+2\ell+2)$PP
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 3: Complexities analysis for BLAC, PEREA, and BLACR

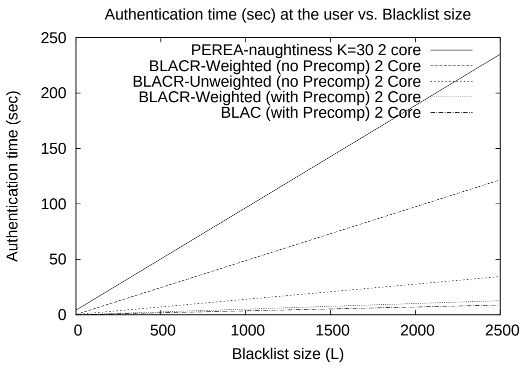
Schemes	Downlink	Uplink
BLACR-Unweighted	$(1573L+10\ell+224)$bits	$(6272L+2240\ell+3586)$bits
BLACR-Unweighted (XDH Assumption made)	$(454L+10\ell+224)$bits	$(2915L+2240\ell+2466)$bits
BLACR-Weighted	$(1573L+10\ell+224)$bits	$(12768L+2240\ell+3586)$bits
BLACR-Weighted (XDH Assumption made)	$(454L+10\ell+224)$bits	$(6048L+2240\ell+2466)$bits

Table 4: Space Complexities of BLACR

blacklists). Thus we expect servers to trim their blacklists and meritlists to "noteworthy" bad and good behaviors. For practical authentication rates we can see that BLACR can feasibly support meritlists and blacklists with 1,000–2,000 entries. 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 reflects this scalability.



(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 1: Estimated authentication times at the SP and User and the cost for the SP.

As we can see in Figure 1(a), for 1,000–2,000 entries on the blacklist/meritlist, BLACR supports authentication rates of 8–17 authentications/minute 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). BLACR-Weighted would support about 2–5 anonymous authentications/minute. BLAC is cheaper, of course, and can support about 32–61 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 $K = 30$, the authentication rate is about 60 authentications/minute.

For the computation at the user we show the costs are reasonable because users can precompute several values before authentication. In Figure 1(b) we can see that BLACR-Weighted with precomputation (both BLACR and BLACR-Weighted have the same performance with precomputation) is very close in performance to BLAC with precomputation—users would expect only a 5–10 second delay per authentication in BLACR, and a 3–7 second delay in BLAC with 1,000–2,000 blacklist/meritlist entries. In comparison, PEREA takes much longer at the user (PEREA trades off efficient verification at the SP for more work at the user). For $K = 30$ and with 1,000–2,000 blacklist/meritlist entries, authentication takes around 97–189 seconds. Thus BLACR performs well on the user side. We note that BLACR and BLACR-Weighted without precomputation take about 14–28 seconds and 49–97 seconds respectively for 1,000–2,000 blacklist/meritlist entries.

If server costs must be kept low, and if the SP desires to have much larger blacklists, 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 “a couple of thousand of 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. We also point out that BLACR-Weighted can be used for a handful of sub-policies, and BLACR-Unweighted for the rest for better performance. Finally, SPs willing to spend say \$10,000/year on four 8-core servers and data transfer costs would be able to support 10,000 blacklist/meritlist entries.

6 Discussion

Outsourced authentication For our cost analysis we used Amazon EC2 pricing as an example of baseline costs for the SP. While the SP may choose to house the authentication servers, cloud based authentication can be a compelling and low-cost choice. We note that in such a scenario the SP does not divulge any secrets to the cloud service provider, and it is only a matter of trusting the result of the computation at the cloud. We acknowledge it is possible for the cloud to lie about who is blacklisted. If the SP cannot trust the cloud to be honest about who is revoked, the SP can perform this computation itself.

Rate limiting As pointed out in previous work on anonymous revocation [25, 37, 34, 36, 35, 4], blocking anonymous users is ineffective if these users can misbehave several (e.g., hundreds) of times before their misbehaviors are discovered. Thus users’ anonymous authentications must be rate-limited in such schemes. For example, if users are limited to 5 anonymous authentications per day, and if it takes on average about a day to “catch” misbehaviors, users can at most misbehave in 5 sessions a day. Existing schemes such as n-times periodic anonymous authentication [12] provide this tool. The technique of unlinkable serial transactions [32] can be used to prevent multiple simultaneous authentications as well (although this is less of a concern if authentications are associated with actions as discussed in the context of concurrent sessions).

Sybil attacks Any anonymous authentication scheme will be vulnerable to Sybil attacks [21] if users can get new credentials after their issued credential is blacklisted. It is generally recognized that such schemes must be bootstrapped off “Sybil-free” credential schemes to ensure each user obtains only one credential. For example, a government issued passport can be viewed as Sybil-free (it is infeasible for most people to get passports with different identities), and then a BLACR credential issuing authority can make note of the passport number and that a credential has been issued to that passport. If the user tries to get a new credential the authority will realize that a credential has already been issued for that passport. Of course, another question that arises is what to do when a legitimate user loses his/her credential and needs a new credential. One possibility is to offer the user a credential with less privacy (pseudonymous credential) that can be easily blocked for a certain probation period before a new credential is issued again. Repeated requests for new credentials can result in permanent banning from the system or longer probation periods [34, 36].

Timing attacks We propose optimizations for the user based on precomputation. If certain users choose to not perform precomputation, or take different times to authenticate, the SP may be able to differentiate between users with less or more entries on the blacklist. Thus we advise that in real-world deployments, all client software must be required to perform precomputation before they request to authenticate. Furthermore, all users should pad the computation time to have some fixed overall delay to appear similar. Such precautions are needed in all anonymous authentication schemes, and we do not further discuss it here.

Blacklist gaming An attack not studied by anonymous revocation schemes thus far is what SPs can infer through gaming of blacklists or revocation lists. SPs can add and remove tickets in such a way that might reveal information about subsequent authentications from users. Studying such gaming attacks is a promising direction for future work. In the interim, users can recognize gaming attacks if a ticket disappears and then reappears on a blacklist. Refusing to authenticate in such circumstances prevents gaming attacks where the SP tries to produce different versions of blacklists to authenticating users.

Concurrent sessions We must avoid the case where a user can authenticate before being blacklisted and then remain logged in even after being blacklisted. We recommend that authentications are associated with short-lived *actions* as opposed to long-lived sessions—for example, in the Wikipedia setting, an authentication is performed while *submitting* an edit. Blacklisting can then be performed between actions, and even if such actions are preempted during a blacklisting operation, requiring a reauthentication (resubmission) is not inconvenient. If long-lived sessions must be supported, following a blacklisting, all logged in users should be required to prove their innocence by logging in again.

Efficient authentication We have already argued that BLACR is efficient enough for real-world deployments. Nevertheless, recent TTP-based schemes such as Nymble [25, 37], Nymbler [24] and Jack [28] 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

Patrick P. Tsang was a PhD student in the Computer Science program at Dartmouth College, and passed away on October 27, 2009 as a victim to cancer. He was 28 years old. Patrick was a disciplined and cheerful student, and he battled his illness with strength and courage. Even in those final moments, he was busy working on ideas presented in this paper. We dedicate this paper to his memory.

We thank Sean W. Smith for his helpful comments and his support for our line of work on Nymble, BLAC and PEREA. Patrick P. Tsang was partially supported in part by the Institute for Security Technology Studies, under Grant number 2005-DD-BX-1091 awarded by the Bureau of Justice Assistance, and the National Science Foundation, under grant CNS-0524695. The views and conclusions do not necessarily represent those of the sponsors.

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: Practical TTP-Free Revocation of Repeatedly Misbehaving Anonymous Users. Technical Report TR688, Indiana University Bloomington, Apr. 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 and X. Boyen. Short signatures without random oracles. In Cachin and Camenisch [10], pages 56–73.

[7] 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.

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

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

[10] C. Cachin and J. Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004.

[11] 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.

[12] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *ACM Conference on Computer and Communications Security*, pages 201–210. ACM, 2006.

[13] J. Camenisch, M. Kohlweiss, and C. Soriente. Solving revocation with efficient update of anonymous credentials. In *SCN*, volume 6280 of *Lecture Notes in Computer Science*, pages 454–471. Springer, 2010.

[14] 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.

[15] 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.

[16] 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.

[17] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

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

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

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

[21] J. R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, 2002. Springer-Verlag.

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

[23] 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.

[24] 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.

[25] 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.

[26] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In Cachin and Camenisch [10], pages 571–589.

[27] 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.

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

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

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

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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 for BLACR-Weighted, which also covers its special case BLACR-Unweighted. Below we refer to the system as BLACR. Firstly, we define an ideal world BLACR system assuming the existence of a trusted party $\mathcal{T}$. Any cryptographic construction of BLACR is secure if it is "equivalent" to the ideal world BLACR, which, is secure by default.[4]

We consider a static model in which the numbers of honest and dishonest users, as well as SPs, are fixed during system setup. We use $\mathcal{U}_{HU}$, $\mathcal{U}_{AU}$, $\mathcal{U}_{HS}$ and $\mathcal{U}_{AS}$ to denote the set of honest users, SPs and dishonest users and SPs respectively. The dishonest parties are under the control of a single PPT algorithm $\mathcal{A}$. We introduce another PPT algorithm $\mathcal{E}$, called environment, which provides the input to, and receives the outputs from, the honest players (users, SPs, GM). $\mathcal{E}$ can also interact with $\mathcal{A}$ freely. In the real world, the players (users, SPs and GM) communicate via cryptographic protocols while in the ideal world, the players do not communicate directly. Rather, they communicate via a trusted party $\mathcal{T}$, which is also responsible for handling all inputs and outputs on behalf of them. Next, we define the functionalities of BLACR in the real world as well as the ideal world. We use $\mathcal{U}_U$ to denote the set of all users, that is $\mathcal{U}_U = \mathcal{U}_{HU} \cup \mathcal{U}_{AU}$. Likewise, we use $\mathcal{U}_S = \mathcal{U}_{HS} \cup \mathcal{U}_{AS}$ to denote the set of all SPs. We use the word *event* to denote the execution of a functionality and the *events* are scheduled according to $\mathcal{E}$'s wishes, with the restriction that the first event must be INIT and that INIT can only be scheduled once. Each event is given a unique identifier `tid` generated by $\mathcal{E}$. We would like to remark that communications with $\mathcal{T}$ is not anonymous meaning that $\mathcal{T}$ knows the exact identity of the communicating party. Dishonest players under the control of $\mathcal{A}$ can deviate from the specification of the functionalities freely.

- INIT($\mathtt{tid}_{\text{INIT}}, \mathcal{U}_{HU}, \mathcal{U}_{AU}, \mathcal{U}_{HS}, \mathcal{U}_{AS}, \{\mathtt{SP}_k\}_{k\in\mathcal{U}_{AS}}$, $\{\mathtt{SP}_k\}_{k\in\mathcal{U}_{HS}}, b_{GM}$). The system begins when $\mathcal{E}$ fixes the honest and dishonest users and SPs, and assigns each SP indexed by $k \in \mathcal{U}_S$ with a unique name $\mathtt{SP}_k$. The bit b_{GM} indicates if the GM is honest or not.

  Real World. The GM generates a key pair ($\mathsf{gpk}, \mathsf{gsk}$), and gpk is sent to all players in the system.

  Ideal World. The trusted party $\mathcal{T}$ initializes an empty set $\mathcal{U}$, which will be used to record the registration and authentication events of the users in $\mathcal{U}_U$.

- REG($\mathtt{tid}_{\text{REG}}, i$). $\mathcal{E}$ instructs user $\mathtt{U}_i$ for $i \in \mathcal{U}_U$ to register with GM. Note that this protocol is not anonymous in the sense that GM knows the user index i.

[4]Note: The definition we give does not entail all formalities necessary to fit into the universal composability (UC) framework [17]; our goal here is to prove the security of our construction. The UC framework allows proving the security of schemes *that remain secure when composed with other schemes*, which we do not attempt to prove.

Real World. $\mathtt{U}_i$ sends a request for registration $(\mathtt{tid}_{\text{REG}}, i)$ to the GM. The user, as well as GM, outputs individually the outcome of this event ($\mathtt{tid}_{\text{REG}}$, `success/ failure`).

Ideal World. $\mathtt{U}_i$ sends a request to $\mathcal{T}$ which checks i has never registered before and informs the GM that user i would like to register. GM returns `accept/reject` to $\mathcal{T}$, which is forwarded to $\mathtt{U}_i$. $\mathtt{U}_i$ and GM also output ($\mathtt{tid}_{\text{REG}}$, `success/failure`) to $\mathcal{E}$. If $\mathtt{U}_i$ (resp. GM) is honest, its output to $\mathcal{E}$ is always the same as it receives from (resp. gives) $\mathcal{T}$. In case they are controlled by $\mathcal{A}$, they can deviate from the specification. If GM returns `accept`, $\mathcal{T}$ creates a column in $\mathcal{U}_U$ with index $(i, \mathtt{tid}_{\text{REG}})$.

- AUTH$(\mathtt{tid}_{\text{AUTH}}, i, j, \mathtt{Pol})$. $\mathcal{E}$ instructs user $\mathtt{U}_i$ to authenticate with $\mathtt{SP}_j$ and instructs $\mathtt{SP}_j$ to reply with policy `Pol`. A policy `Pol` is of the form $(\bigvee_{i=1}^{\ell} (\wedge_{j=1}^{\ell_i} \mathcal{P}_{k_{ij}}))$ and each $\mathcal{P}_{k_{ij}}$ is a list of 4-tuples $(c_{ij}, n_{ij}, \mathcal{D}_{ij}^{+}, \mathcal{D}_{ij}^{-})$, where c_{ij} represents a category, n_{ij} represents a threshold and the sets $\mathcal{D}_{ij}^{+}$, $\mathcal{D}_{ij}^{-}$ are the list of adjusting factors, which will be $\{1, 1, \ldots\}$ in case it is BLACR-Unweighted. The negation of the policy includes an extra bit to indicate it should be treated as negation.

Real World. $\mathtt{U}_i$ sends a request for authentication $(\mathtt{tid}_{\text{AUTH}})$ to $\mathtt{SP}_j$. The user, as well as $\mathtt{SP}_j$, outputs individually the outcome of this event ($\mathtt{tid}_{\text{AUTH}}$, `accept/ reject`).

Ideal World. $\mathtt{U}_i$ sends a request to $\mathcal{T}$, who forwards the request to $\mathtt{SP}_j$. $\mathtt{SP}_j$ replies with policy `Pol`, along with the relevant meritlists and blacklists. Each list is of the form $\{(\mathtt{tid}_{\text{AUTH}1}, s_1), (\mathtt{tid}_{\text{AUTH}2}, s_2), \ldots,\}$, where $\mathtt{tid}_{\text{AUTH}\cdot}$ represents an authentication event and s the corresponding score. $\mathcal{T}$ then checks, using $\mathcal{U}_U$, to see if $\mathtt{U}_i$ satisfies `Pol` and forwards `Pol` and the meritlists and blacklists to $\mathtt{U}_i$, along with a bit indicating if $\mathtt{U}_i$ satisfies the policy. $\mathtt{U}_i$ replies to $\mathcal{T}$ with a bit, indicating if he/she would like to proceed. If $\mathtt{U}_i$ chooses not to proceed, both players do not have to output anything. If $\mathtt{U}_i$ chooses to proceed, $\mathcal{T}$ sends `valid/invalid` to $\mathtt{SP}_j$ to indicating if the authenticating user satisfies `Pol` or not. $\mathtt{SP}_j$ replies with `accept/reject` and $\mathcal{T}$ forwards the results to $\mathtt{U}_i$. If $\mathtt{SP}_j$ replies with `accept`, $\mathcal{T}$ appends the entry $\mathtt{tid}_{\text{AUTH}}$ to the column in $\mathcal{U}_U$ indexed by $(i, \mathtt{tid}_{\text{REG}}, j)$. Both $\mathtt{U}_i$ and $\mathtt{SP}_j$ output ($\mathtt{tid}_{\text{AUTH}}$, `accept/reject`) to $\mathcal{E}$ to indicate the result of this event.

- ADD-TO-LIST($\mathtt{tid}_{\text{ATL}}$, j, c, $+/-$, $\mathtt{tid}_{\text{AUTH}}$, s). $\mathcal{E}$ instructs $\mathtt{SP}_j$ to add the authentication identified by $\mathtt{tid}_{\text{AUTH}}$ to its meritlist or blacklist for category c with score s.

Real World. If $\mathtt{tid}_{\text{AUTH}}$ corresponds to an authentication event such that $\mathtt{SP}_j$ outputs `accept`, $\mathtt{SP}_j$ adds the ticket of the corresponding authentication to its meritlist or blacklist for category c with score s. $\mathtt{SP}_j$ outputs ($\mathtt{tid}_{\text{ATL}}$, `success/failure`) to $\mathcal{E}$ to indicate if this event is executed successfully.

Ideal World. $\mathtt{SP}_j$ sends the request to $\mathcal{T}$, who checks if $\mathtt{tid}_{\text{AUTH}}$ corresponds to an authentication event in which $\mathtt{SP}_j$ outputs `accept`, and replies with a bit to $\mathtt{SP}_j$

indicating the result. $\mathtt{SP}_j$ adds $(\mathtt{tid}_{\text{AUTH}}, s)$ to its meritlist or blacklist for category c if the check is successful. $\mathtt{SP}_j$ outputs $(\mathtt{tid}_{\text{ATL}}, \mathtt{success/failure})$ to $\mathcal{E}$ to indicate if this event is executed successfully.

- REMOVE-FROM-LIST$(\mathtt{tid}_{\text{RML}}, j, \mathtt{tid}_{\text{ATL}})$. $\mathcal{E}$ instructs $\mathtt{SP}_j$ to remove the entry added to its list in the event $\mathtt{tid}_{\text{ATL}}$.

  Real World. If $\mathtt{tid}_{\text{ATL}}$ corresponds to a ADD-TO-LIST event such that $\mathtt{SP}_j$ outputs $\mathtt{accept}$, $\mathtt{SP}_j$ removes the ticket of the corresponding authentication from its meritlist or blacklist for category c. $\mathtt{SP}_j$ outputs $(\mathtt{tid}_{(\text{RML})}, \mathtt{success/failure})$ to $\mathcal{E}$ to indicate if this event is executed successfully.

  Ideal World. $\mathtt{SP}_j$ sends the request to $\mathcal{T}$, who checks if $\mathtt{tid}_{\text{ATL}}$ corresponds to a ADD-TO-LIST event in which $\mathtt{SP}_j$ outputs $\mathtt{success}$, and replies with a bit to $\mathtt{SP}_j$ indicating the result. $\mathtt{SP}_j$ removes the corresponding entry from its meritlist or blacklist if the check is successful. $\mathtt{SP}_j$ outputs $(\mathtt{tid}_{\text{RML}}$, $\mathtt{success/\ failure})$ to $\mathcal{E}$ to indicate if this event is executed successfully.

Ideal world BLACR provides all the desired security properties. Firstly, all the transactions, in the view of GM and SPs, are anonymous. $\mathcal{T}$ only informs GM and SP some anonymous user would like to register or authenticates in REG or AUTH and thus anonymity is guaranteed. Secondly, $\mathcal{T}$ verifies if the authenticating user has registered before and checks if the user satisfies the policy in AUTH and thus authenticity and mis-authentication resistance and authenticity are assured. Finally, $\mathcal{T}$ informs the SPs if an authenticating user satisfies the policy in AUTH and thus an honest user will always be accepted by an honest SP during authentication. Thus, the system provides non-frameability.

Finally, we are able to define the security of a cryptographic BLACR. A cryptographic BLACR is secure if for every real world adversary $\mathcal{A}$ and every environment $\mathcal{E}$, there exists an ideal world 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 whether it is running in the real world interacting with $\mathcal{A}$ or it is running in the ideal world interacting with $\mathcal{S}$, which has blackbox access to $\mathcal{A}$. Formally, we present Definition 1 below.

Definition 1 *Let* $\mathbf{Real}_{\mathcal{E},\mathcal{A}}(\lambda)$ *(resp.* $\mathbf{Ideal}_{\mathcal{E},\mathcal{S}_{\mathcal{A}}}(\lambda)$ *) 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 blackbox access to* $\mathcal{A}$*). A cryptographic BLACR is secure if for all PPT adversaries* $\mathcal{E}$ *and* $\mathcal{A}$*, the following expression holds:*

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

where we use $\mathsf{negl}(\lambda)$ to denote a negligible function in security parameter λ.

We state the following theorem regarding the security of BLACR.

Theorem 1 *BLACR satisfies Definition 1 under the q-SDH Assumption and the DDH Assumption in the random oracle model.*

We analyze the security of our BLACR construction by proving indistinguishability between an adversary's actions in the real world and the ideal world. Again, we focus on BLACR-Weighted since BLACR-Unweighted is just a special case of it. In the following we use BLACR to represent BLACR-Weighted.

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. The first case covers the security requirements of *anonymity* and *non-frameability* while the second case covers the property of *misauthentication resistance* and *authenticity*. For instance, *anonymity* and *non-frameability* should hold against a malicious GM.

We handle each case separately in the following two lemmas.

Lemma 1 *For all PPT environments $\mathcal{E}$ and all real world adversaries $\mathcal{A}$ controlling a subset of SPs and users, there exists an ideal world simulator $\mathcal{S}$ such that*

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

Proof We define a simulator $\mathcal{S}$ which interacts with the environment $\mathcal{E}$ as an ideal world adversary, and at the same time provides the view as the environment as well as the honest players to the real world adversary $\mathcal{A}$. $\mathcal{S}$ also acts as an ideal world adversary to the trusted party $\mathcal{T}$. We reiterate the goal of $\mathcal{S}$ here: for any PPT $\mathcal{A}$, $\mathcal{S}$'s goal is to provide the view to $\mathcal{E}$ so that $\mathcal{E}$ cannot distinguish if it is interact with $\mathcal{A}$ or $\mathcal{S}$ having blackbox access to $\mathcal{A}$. Firstly, $\mathcal{S}$ simply forwards any messages between $\mathcal{E}$ and $\mathcal{A}$. Next, we describe how $\mathcal{S}$ participates in each event.

- $\textsc{Init}(\mathtt{tid}_{\textsc{Init}}, \mathcal{U}_{HU}, \mathcal{U}_{AU}, \mathcal{U}_{HS}, \mathcal{U}_{AS}, \{\mathtt{SP}_k\}_{k\in\mathcal{U}_{AS}}, \{\mathtt{SP}_k\}_{k\in\mathcal{U}_{HS}}, 0)$. The system begins when $\mathcal{E}$ fixes the honest and dishonest users and SPs, and assigns each SP indexed by $k \in \mathcal{U}_S$ with a unique name $\mathtt{SP}_k$. The last bit 0 indicates GM is honest.

  Representing honest GM to $\mathcal{A}$. $\mathcal{S}$ generates the key pair $(\mathsf{gpk}, \mathsf{gsk})$ on behalf of the honest GM, and sends gpk to $\mathcal{A}$.

  Representing dishonest SP/users to $\mathcal{T}$. In this event, $\mathcal{S}$ has no interaction with $\mathcal{T}$.

- $\textsc{Reg}(\mathtt{tid}_{\textsc{Reg}}, i)$. $\mathcal{E}$ instructs user $\mathtt{U}_i$ to register with GM.

  Representing honest user to honest GM. If $i \in \mathcal{U}_{HU}$, $\mathcal{S}$ simulates the protocol itself on behalf of both sides.

  Representing honest GM to $\mathcal{A}$/ dishonest user to $\mathcal{T}$. If $i \in \mathcal{U}_{AU}$, $\mathcal{S}$ receives $(\mathtt{tid}_{\textsc{Reg}}, i)$ from $\mathcal{A}$. $\mathcal{S}$ extracts from $\mathcal{A}$ the values (x, y') in Eq. 1. If the extraction fails, $\mathcal{S}$ aborts. Otherwise $\mathcal{S}$ sends $(\mathtt{tid}_{\textsc{Reg}}, i)$ to $\mathcal{T}$. If $\mathcal{T}$ replies with $\mathtt{accept}$, $\mathcal{S}$ issues a credential to $\mathcal{A}$ following the protocol. Otherwise, it rejects the request from $\mathcal{A}$. $\mathcal{S}$ stores $\mathsf{usk}_i = (A, e, x, y)$ it issues to $\mathcal{A}$. Note that $\mathcal{S}$ is able to compute x and y since it has extracted the values (x, y') successfully in Eq. 1. The protocol outcome from $\mathcal{A}$ on behalf of user $\mathtt{U}_i$ is forwarded to $\mathcal{T}$.

- $\textsc{Auth}(\mathtt{tid}_{\textsc{Auth}}, i, j, \mathtt{Pol})$. $\mathcal{E}$ instructs user $\mathtt{U}_i$ to authenticate with $\mathtt{SP}_j$ and instructs $\mathtt{SP}_j$ to reply with policy $\mathtt{Pol}$. Note that $\mathtt{U}_i$ only receives $(\mathtt{tid}_{\textsc{Auth}}, j)$ and $\mathtt{SP}_j$ only receives $(\mathtt{tid}_{\text{AUTH}}, \mathtt{Pol})$ from this event. A policy $\mathtt{Pol}$ is of the form $(\bigvee_{i=1}^{\ell}(\wedge_{j=1}^{\ell_i}\mathcal{P}_{k_{ij}}))$ and each $\mathcal{P}_{k_{ij}}$ is a list of 4-tuples $(c_{ij}, n_{ij}, \mathcal{D}_{ij}^{+}, \mathcal{D}_{ij}^{-})$, where c_{ij} represents a category, n_{ij} represents a threshold and the sets $\mathcal{D}_{ij}^{+}$, $\mathcal{D}_{ij}^{-}$ are the list of adjusting factors, which will be $\{1, 1, \ldots\}$ in case it is BLACR-Unweighted.

Representing honest SP to $\mathcal{A}$/ dishonest user to $\mathcal{T}$. If $i \in \mathcal{U}_{AU}$ and $j \in_{\mathcal{HS}}$, $\mathcal{S}$ receives $(\mathtt{tid}_{\text{AUTH}})$ from $\mathcal{A}$. $\mathcal{S}$ does not know which user credential $\mathcal{A}$ is using. On the other hand, $\mathcal{S}$ receives, on behalf of $\mathtt{U}_i$, $(\mathtt{tid}_{\text{AUTH}}, j)$ from $\mathcal{E}$ as the event. $\mathcal{S}$ sends $(\mathtt{tid}_{\text{AUTH}})$ to $\mathcal{T}$ on behalf of a user i such that $i \in \mathcal{U}_{AU}$. $\mathcal{T}$ replies with $\mathtt{Pol}$, a set of meritlists and blacklists, together with a bit indicating if user i satisfies the policy. Based on the set of meritlists and blacklists, $\mathcal{S}$ reconstructs the meritlists and blacklists. Specifically, each entry in the list is of the form $(\mathtt{tid}_{\text{AUTH}}, s)$, $\mathcal{S}$ locates the corresponding authentication transactions and appends (t, s) to the corresponding list where t is the ticket associated with the authentication. $\mathcal{S}$ sends the meritlist and blacklist to $\mathcal{A}$, along with $\mathtt{Pol}$ to $\mathcal{A}$. If $\mathcal{A}$ chooses not to proceed, $\mathcal{S}$ replies to $\mathcal{T}$ that he would not proceed. Otherwise, $\mathcal{S}$ has to locate the actual user index $\hat{i}$ that $\mathcal{A}$ is using. Note that $\mathcal{A}$ may be using the credential of user $\hat{i} \in \mathcal{U}_{AU}$ instead of i to conduct the authentication. To do so, $\mathcal{S}$ tests if $t = \hat{b}^x$ for all x in its list of $\mathsf{usk}_{\hat{i}} = (A_{\hat{i}}, e_{\hat{i}}, x_{\hat{i}}, y_{\hat{i}})$. $\mathcal{S}$ aborts if it cannot locate an entry $\hat{i}$. Otherwise, $\mathcal{S}$ sends to $\mathcal{T}$ $(\mathtt{tid}_{\text{AUTH}})$ on behalf of user $\hat{i}$ and chooses to proceed regardless of whether $\mathcal{T}$ indicates if user $\hat{i}$ satisfies the policy. If $\mathcal{A}$ produces a valid authentication to $\mathcal{S}$ while $\mathcal{T}$ indicates user $\hat{i}$ does not satisfy the policy, $\mathcal{S}$ aborts.

Representing honest user to $\mathcal{A}$/ dishonest SP to $\mathcal{T}$. If $i \in \mathcal{U}_{HU}$ and $j \in_{\mathcal{AS}}$, $\mathcal{S}$ receives $(\mathtt{tid}_{\text{AUTH}})$ from $\mathcal{T}$ as $\mathtt{SP}_j$. It sends $(\mathtt{tid}_{\text{AUTH}})$ to $\mathcal{A}$. $\mathcal{A}$ replies with a set of meritlists and blacklists as well as $\mathtt{Pol}$. Based on the meritlists and blacklists and $\mathtt{Pol}$ from $\mathcal{A}$, $\mathcal{T}$ replies to $\mathcal{T}$ with $\mathtt{Pol}$ and the corresponding meritlists and blacklists. In case the tickets in the meritlists and blacklists provided by $\mathcal{A}$ does not correspond to any past authentications, $\mathcal{S}$ simply ignores them in the blacklists and meritlists sent to $\mathcal{T}$. If $\mathcal{T}$ replies with a bit indicates that the authenticating user satisfies the authentication policy, $\mathcal{S}$ conducts the authentication to $\mathcal{A}$ with a fake zero-knowledge proof-of-knowledge of Eq. 5. If $\mathcal{A}$ accepts the authentication, $\mathcal{S}$ replies to $\mathcal{T}$ with $\mathtt{accept}$.

- ADD-TO-LIST($\mathtt{tid}_{\text{ATL}}$, j, c, $+/-$, $\mathtt{tid}_{\text{AUTH}}$, s). $\mathcal{E}$ instructs $\mathtt{SP}_j$ to add the authentication identified by $\mathtt{tid}_{\text{AUTH}}$ to its meritlist or blacklist for category c with score s.

Representing honest SP. If $j \in \mathcal{U}_{HU}$, $\mathcal{S}$ checks if $\mathtt{tid}_{\text{AUTH}}$ corresponds to an authentication event such that $\mathtt{SP}_j$ accepts and adds the ticket of the corresponding authentication to its meritlist or blacklist for category c with score s.

Representing dishonest SP to $\mathcal{T}$. If $j \in \mathcal{U}_{AU}$, $\mathcal{S}$ sends the request to $\mathcal{T}$ and adds $(\mathtt{tid}_{\text{AUTH}}, s)$ to its meritlist or blacklist for category c if $\mathcal{T}$ replies that the check is successful.

- REMOVE-FROM-LIST($\mathtt{tid}_{\text{RML}}, j, \mathtt{tid}_{\text{ATL}}$). $\mathcal{E}$ instructs $\mathtt{SP}_j$ to remove the entry added to its list in the event $\mathtt{tid}_{\text{ATL}}$.

Representing honest SP. If $j \in \mathcal{U}_{HU}$, $\mathcal{S}$ checks if $\mathtt{tid}_{\text{ATL}}$ corresponds to a ADD-TO-LIST event such that SP_j outputs `accept` and removes the ticket of the corresponding authentication from its meritlist or blacklist for category c.

Representing dishonest SP to $\mathcal{T}$. If $j \in \mathcal{U}_{AU}$, $\mathcal{S}$ sends the request to $\mathcal{T}$ and removes the corresponding entry from its meritlist or blacklist if $\mathcal{T}$ replies that the check is successful.

If $\mathcal{S}$ does not abort, the output provided by all players, including the dishonest players represented by $\mathcal{S}$ in the ideal world is the same as that output by the all players, including the dishonest players represented by $\mathcal{A}$ to $\mathcal{E}$ and thus

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

It remains to show that $\mathcal{S}$ aborts with negligible probability.

1. In an REG event, $\mathcal{S}$ aborts if it cannot extract the values (x, y') from $\mathcal{A}$. This happens with negligible probability due to the soundness of the zero-knowledge protocol in the random oracle model.

2. In an AUTH event, $\mathcal{S}$ aborts if it cannot locate the user index $\hat{i}$. The corresponds to the case when $\mathcal{A}$ is able to produce an x such that it has a valid BBS+ signature (A, e, y) and this happens with negligible probability under the unforgeability of the BBS+ signature.

3. In an AUTH event, $\mathcal{S}$ aborts if $\mathcal{T}$ indicates user $\hat{i}$ does not satisfy the policy while $\mathcal{A}$ produces an authentication such that $\mathcal{S}$ has to accept.

□

Lemma 2 *For all PPT environments $\mathcal{E}$ and all real world adversaries $\mathcal{A}$ controlling the GM, a subset of SPs and users, there exists an ideal world simulator $\mathcal{S}$ such that*

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

Proof Similar to the proof of Lemma 1, we define a simulator $\mathcal{S}$ which interacts with the environment $\mathcal{E}$ as an ideal world adversary, and at the same time provides the view as the environment as well as the honest players to the real world adversary $\mathcal{A}$. $\mathcal{S}$ also acts as an ideal world adversary to the trusted party $\mathcal{T}$. Firstly, $\mathcal{S}$ simply forwards any messages between $\mathcal{E}$ and $\mathcal{A}$. Next, we describe how $\mathcal{S}$ participates in each event.

- INIT($\mathtt{tid}_{\text{INIT}}$, $\mathcal{U}_{HU}$, $\mathcal{U}_{AU}$, $\mathcal{U}_{HS}$, $\mathcal{U}_{AS}$, $\{\mathsf{SP}_k\}_{k\in\mathcal{U}_{AS}}$, $\{\mathsf{SP}_k\}_{k\in\mathcal{U}_{HS}}$, 1). The system begins when $\mathcal{E}$ fixes the honest and dishonest users and SPs, and assigns each SP indexed by $k \in \mathcal{U}_S$ with a unique name SP_k. The last bit 1 indicates GM is under the controlled of $\mathcal{A}$ as well.

  Representing honest users/SP to $\mathcal{A}$. $\mathcal{S}$ receives gpk from $\mathcal{A}$.

  Representing dishonest SP/users to $\mathcal{T}$. In this event, $\mathcal{S}$ has no interaction with $\mathcal{T}$.

- $\textsc{Reg}(\texttt{tid}_{\textsc{Reg}}, i)$. $\mathcal{E}$ instructs user $\texttt{U}_i$ to register with GM.

  Representing dishonest user to $\mathcal{A}$/ dishonest GM to $\mathcal{T}$. If $i \in \mathcal{U}_{HU}$, $\mathcal{S}$ receives $(\texttt{tid}_{\textsc{Reg}}, i)$ from $\mathcal{T}$. $\mathcal{S}$ initiates a registration request with $\mathcal{A}$ and uses the zero-knowledge simulator to simulate the zero-knowledge proof-of-knowledge in Eq. 1. If $\mathcal{S}$ obtains a credential from $\mathcal{A}$, it replies `accept` to $\mathcal{T}$.

- $\textsc{Auth}(\texttt{tid}_{\textsc{Auth}}, i, j, \texttt{Pol})$. $\mathcal{E}$ instructs user $\texttt{U}_i$ to authenticate with $\texttt{SP}_j$ and instructs $\texttt{SP}_j$ to reply with policy `Pol`. Note that $\texttt{U}_i$ only receives $(\texttt{tid}_{\textsc{Auth}}, j)$ and $\texttt{SP}_j$ only receives $(\texttt{tid}_{\textsc{Auth}}, \texttt{Pol})$ from this event. A policy `Pol` is of the form $(\bigvee_{i=1}^{\ell} (\wedge_{j=1}^{\ell_i} \mathcal{P}_{k_{ij}}))$ and each $\mathcal{P}_{k_{ij}}$ is a list of 4-tuples $(c_{ij}, n_{ij}, \mathcal{D}_{ij}^{+}, \mathcal{D}_{ij}^{-})$, where c_{ij} represents a category, n_{ij} represents a threshold and the sets $\mathcal{D}_{ij}^{+}$, $\mathcal{D}_{ij}^{-}$ are the list of adjusting factors, which will be $\{1, 1, \ldots\}$ in case it is BLACR-Unweighted.

  Representing honest SP to $\mathcal{A}$/ dishonest user to $\mathcal{T}$. If $i \in \mathcal{U}_{AU}$ and $j \in \mathcal{U}_{HS}$, $\mathcal{S}$ receives $(\texttt{tid}_{\textsc{Auth}}, j)$ as $\texttt{U}_i$ from $\mathcal{E}$. $\mathcal{S}$ sends $(\texttt{tid}_{\textsc{Auth}})$ to $\mathcal{T}$ on behalf of a user i. $\mathcal{T}$ replies with `Pol`, a set of meritlists and blacklists, together with a bit indicating if user i satisfies the policy. At the same time, $\mathcal{S}$ receives $(\texttt{tid}_{\textsc{Auth}})$ from $\mathcal{A}$. Based on the set of meritlists and blacklists, $\mathcal{S}$ reconstructs the meritlists and blacklists. Specifically, each entry in the list is of the form $(\texttt{tid}_{\textsc{Auth}}, s)$, $\mathcal{S}$ locates the corresponding authentication transactions and appends (t, s) to the corresponding list where t is the ticket associated with the authentication. $\mathcal{S}$ sends the meritlist and blacklist to $\mathcal{A}$, along with `Pol` to $\mathcal{A}$. If $\mathcal{A}$ chooses not to proceed, $\mathcal{S}$ replies to $\mathcal{T}$ that he would not proceed. Otherwise, if $\mathcal{A}$ produces a successful authentication, $\mathcal{S}$ checks if there exists $\hat{i} \in \mathcal{U}_{AU}$ such that $\texttt{U}_{\hat{i}}$ satisfy the policy. If yes, $\mathcal{S}$ sends $(\texttt{tid}_{\textsc{Auth}})$ to $\mathcal{T}$ on behalf of user $\hat{i}$. If none of the user in $\mathcal{U}_{AU}$ satisfies the policy, $\mathcal{S}$ has to create one since $\mathcal{A}$ is in possession of the GM's secret key and he can simply create one to pass the authentication. Specifically, $\mathcal{S}$ chooses an index $i \in \mathcal{U}_{AU}$ and submits $(\texttt{tid}_{\textsc{Reg}}, i)$ to $\mathcal{T}$ and subsequently reply to $\mathcal{T}$ with `accept` on behalf of the dishonest GM, and submits $\texttt{tid}_{\textsc{Auth}}$ to $\mathcal{T}$ on behalf of this newly created user i.

  Representing honest user to $\mathcal{A}$/ dishonest SP to $\mathcal{T}$. If $i \in \mathcal{U}_{HU}$ and $j \in_{\mathcal{AS}}$, $\mathcal{S}$ receives $(\texttt{tid}_{\textsc{Auth}})$ from $\mathcal{T}$ as $\texttt{SP}_j$. It sends $(\texttt{tid}_{\textsc{Auth}})$ to $\mathcal{A}$. $\mathcal{A}$ replies with a set of meritlists and blacklists as well as `Pol`. Based on the meritlists and blacklists and `Pol` from $\mathcal{A}$, $\mathcal{T}$ replies to $\mathcal{T}$ with `Pol` and the corresponding meritlists and blacklists. In case the tickets in the meritlists and blacklists provided by $\mathcal{A}$ does not correspond to any past authentications, $\mathcal{S}$ simply ignores them in the blacklists and meritlists sent to $\mathcal{T}$. If $\mathcal{T}$ replies with a bit indicates that the authenticating user satisfies the authentication policy, $\mathcal{S}$ conducts the authentication to $\mathcal{A}$ with a fake zero-knowledge proof-of-knowledge of Eq. 5. Note that the real value $\hat{b}^x$ is also replaced with a random value t. If $\mathcal{A}$ accepts the authentication, $\mathcal{S}$ replies to $\mathcal{T}$ with `accept`.

- $\textsc{Add-To-List}(\texttt{tid}_{\textsc{ATL}}, j, c, +/-, \texttt{tid}_{\textsc{Auth}}, s)$. $\mathcal{E}$ instructs $\texttt{SP}_j$ to add the authentication identified by $\texttt{tid}_{\textsc{Auth}}$ to its meritlist or blacklist for category c with score s.

Representing honest SP. If $j \in \mathcal{U}_{HU}$, $\mathcal{S}$ checks if $\texttt{tid}_{\text{AUTH}}$ corresponds to an authentication event such that $\texttt{SP}_j$ accepts and adds the ticket of the corresponding authentication to its meritlist or blacklist for category c with score s.

Representing dishonest SP to $\mathcal{T}$. If $j \in \mathcal{U}_{AU}$, $\mathcal{S}$ sends the request to $\mathcal{T}$ and adds $(\texttt{tid}_{\text{AUTH}}, s)$ to its meritlist or blacklist for category c if $\mathcal{T}$ replies that the check is successful.

- REMOVE-FROM-LIST$(\texttt{tid}_{\text{RML}}, j, \texttt{tid}_{\text{ATL}})$. $\mathcal{E}$ instructs $\texttt{SP}_j$ to remove the entry added to its list in the event $\texttt{tid}_{\text{ATL}}$.

Representing honest SP. If $j \in \mathcal{U}_{HU}$, $\mathcal{S}$ checks if $\texttt{tid}_{\text{ATL}}$ corresponds to a ADD-TO-LIST event such that $\texttt{SP}_j$ outputs `accept` and removes the ticket of the corresponding authentication from its meritlist or blacklist for category c.

Representing dishonest SP to $\mathcal{T}$. If $j \in \mathcal{U}_{AU}$, $\mathcal{S}$ sends the request to $\mathcal{T}$ and removes the corresponding entry from its meritlist or blacklist if $\mathcal{T}$ replies that the check is successful.

The simulation provided to $\mathcal{A}$ by $\mathcal{S}$ is perfect due to the zero-knowledgeness of the protocol as well as the DDH assumption. Thus, the output provided by all players, including the dishonest players represented by $\mathcal{S}$ in the ideal world is the same as that output by the all players, including the dishonest players represented by $\mathcal{A}$ to $\mathcal{E}$ and thus

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

□

B SPK Implementation Details

We detail the instantiation of an authentication protocol with policy $\texttt{Pol} = \mathcal{P}$ in BLACR-Weighted, a single-term policy which requires an authenticating user to have reputation in category c higher than or equal to a threshold n. It is easy to extend it to any $\texttt{Pol}$ in DNF form.

User Alice receives from SP the blacklist $\mathcal{L}^+ = \{(t_1, b_1, s_1), \ldots, (t_{L^+}, b_{L^+}, s_{L^+})\}$ and $\mathcal{L}^- = \{(t_1, b_1, s_1), \ldots, (t_{L^-}, b_{L^-}, s_{L^-})\}$, where L^+ and L^- are the size of $\mathcal{L}^+$ and $\mathcal{L}^-$ respectively. Alice also obtains the list of adjusting factors $\mathcal{D}^+ = \{(1, \Delta_1^+, \sigma_1^+), (2, \Delta_2^+, \sigma_2^+), \ldots\}$ and $\mathcal{D}^- = \{(1, \Delta_1^-, \sigma_1^-), (2, \Delta_2^-, \sigma_2^-) \ldots\}$.

In this particular instantiation, we use the credential signature for σ_i^+. Specifically, $\sigma_i^+ = (A_i^+, e_i^+, y_i^+)$ such that $\hat{e}(A_i^+, w_c^+ h_0^{e_i^+}) = \hat{e}(g_0 g_1^i g_2^{\Delta_i^+} g_3^{y_i^+}, h_0)$, with w_c^+ being the public key for the signature scheme for category c. Recall that each category may have its own adjusting factor and requires a different public key for the signature scheme.

We further assume $\mathfrak{u}_1, \mathfrak{u}_2, \mathfrak{u}_3$ are some independent generators of $\mathbb{G}_1$ and $\mathfrak{g}_1$, $\mathfrak{g}_2$, $\mathfrak{g}_3$ are some independent generators of $\mathbb{G}$. We use $\hat{E}_0$, $\hat{E}_1$, $\hat{E}_2$, $\hat{E}_3$, $\hat{E}_{\mathfrak{u}_1}$, $\hat{E}_{\mathfrak{u}_1 w}$ and $\hat{E}_{\mathfrak{u}_1 w^+}$ to denote $\hat{e}(g_0, h_0)$, $\hat{e}(g_1, h_0)$, $\hat{e}(g_2, h_0)$, $\hat{e}(g_3, h_0)$, $\hat{e}(\mathfrak{u}_1, h_0)$, $\hat{e}(\mathfrak{u}_1, w)$ and $\hat{e}(\mathfrak{u}_1, w^+)$. All these pairings can be pre-computed and is assume to be included as part of the public parameter.

Assume Alice's secret key is of the form (A, e, x, y). She first check if her reputation $\mathsf{R} = \mathsf{S}'(\mathcal{L}^+, x, \mathcal{D}^+) - \mathsf{S}'(\mathcal{L}^-, x, \mathcal{D}^-)$ is over the threshold n. Next, she computes the ticket for this authentication $t = H(b||\mathtt{sid})^x$ for a randomly generated value b. Next, she needs to compute the non-interactive proof $\Pi_2 = (\Pi_x, \Pi)$ which assures the SP she is a legitimate user satisfying policy $\mathtt{Pol}$.

Π_x is an non-interactive proof that assures t is correctly formed. Specifically, Alice computes $\mathfrak{A}_1 = \mathfrak{u}_1^{r_1}\mathfrak{u}_2^{r_2}$, $\mathfrak{A}_2 = A\mathfrak{u}_1^{r_2}$ for some randomly generated $r_1, r_2 \in_R \mathbb{Z}_p$. Next, Alice produces $\Pi_{x'}$ which is the following non-interactive proof:

$$SPK\left\{\begin{array}{l} (x, y, r_1, r_2, e, \beta_1, \beta_2): \\ \qquad t = \hat{b}^x \ \wedge \\ \qquad \mathfrak{A}_1 = \mathfrak{u}_1^{r_1}\mathfrak{u}_2^{r_2} \ \wedge \\ \qquad 1 = \mathfrak{A}_1^{-e}\mathfrak{u}_1^{\beta_1}\mathfrak{u}_2^{\beta_2} \ \wedge \\ \qquad \dfrac{\hat{e}(\mathfrak{A}_2, w)}{\hat{E}_0} = \hat{e}(\mathfrak{A}_2, h_0)^{-e}\hat{E}_{\mathfrak{u}_1 w}^{-r_2}\hat{E}_{\mathfrak{u}_1}^{\beta_2}\hat{E}_1^x\hat{E}_2^y \end{array}\right\}(M) \tag{7}$$

where $M = R||\mathfrak{A}_1||\mathfrak{A}_2||b||t$ with R being the random challenge given by the SP and $\beta_1 = r_1 e$, $\beta_2 = r_2 e$.

Π_x is then parsed as $(\Pi_{x'}, \mathfrak{A}_1, \mathfrak{A}_2)$.

The second part of the non-interactive proof, Π, is given below.

Firstly, define the index set $\mathcal{I}$ for list $\mathcal{L}^+$, that is, $\mathcal{I} : \{\iota|(b_\iota, t_\iota, \cdot) \in \mathcal{L}^+, H(b_\iota||\mathtt{sid})^x = t_\iota\}$. Intuitively, for any $i \in \mathcal{I}$, $(b_i, t_i, s_i) \in \mathcal{L}^+$ corresponds to a ticket together with its score of Alice on the list $\mathcal{L}^+$. For each $i \in \mathcal{I}$, $k_i := |\{\iota|\iota \leq i \wedge \iota \in \mathcal{I}\}|$ denotes the number of times Alice has been put on the list $\mathcal{L}^+$ and the correct weight is $\Delta_{k_i}^+$.

For all $i \in \mathcal{I}$, Alice computes $C_i = \mathfrak{g}_1^{\Delta_{k_i}^+ s_i}\mathfrak{g}_2^{r_i}$, $\tilde{C}_i = \mathfrak{g}_1\mathfrak{g}_2^{\tilde{r}_i}$ for some randomly generated $r_i, \tilde{r}_i \in_R \mathbb{Z}_p$. Likewise, for all $i \in [L^+]\backslash\mathcal{I}$, computes $C_i = \mathfrak{g}_2^{r_i}$, $\tilde{C}_i = \mathfrak{g}_2^{\tilde{r}_i}$ for some randomly generated $r_i, \tilde{r}_i \in_R \mathbb{Z}_p$. Parse aux^+ as $(C_1, \tilde{C}_1, \ldots, C_{L^+}, \tilde{C}_{L^+})$.

For $i \in [L^+]\backslash\mathcal{I}$, Alice computes $\mathfrak{A}_{i,1} = (\frac{t_i}{b_i^x})^{a_{i,1}}$, $\mathfrak{A}_{i,2} = \mathfrak{g}_1^{a_{i,1}}\mathfrak{g}_2^{a_{i,2}}$ for some randomly generated $a_{i,1}, a_{i,2} \in_R \mathbb{Z}_p$, and randomly generates $\mathfrak{A}_{i,3}, \mathfrak{A}_{i,4} \in_R \mathbb{G}$.

For $i \in \mathcal{I}$, Alice randomly generates $\mathfrak{A}_{i,1}, \mathfrak{A}_{i,2} \in_R \mathbb{G}$. She then computes $\mathfrak{A}_{i,3} = \mathfrak{u}_1^{a_{i,3}}\mathfrak{u}_2^{a_{i,4}}$ and $\mathfrak{A}_{i,4} = A_{k_i}^+\mathfrak{u}_1^{a_{i,4}}$ for some randomly generated $a_{i,3}, a_{i,4} \in_R \mathbb{Z}_p$.

For $i = 1$ to L^+, Alice computes a non-interactive proof Π_i:

$$SPK \left\{ \begin{array}{l} (x, a_{i,1}, a_{i,2}, \beta_{i,1}, \beta_{i,2}, \beta_{i,3}, \beta_{i,4}, \beta_{i,5}, \\ \; r_i, \tilde{r}_i, a_{i,3}, a_{i,4}, e^+_{k_i}, k_i, \Delta^+_{k_i}, y^+_{k_i}) : \\ \left(\begin{array}{l} t = \hat{b}^x \;\; \wedge \\ \mathfrak{A}_{i,2} = \mathfrak{g}_1^{a_{i,1}} \mathfrak{g}_2^{a_{i,2}} \;\; \wedge \\ 1 = \mathfrak{A}_{i,2}^{-x} \mathfrak{g}_1^{\beta_{i,1}} \mathfrak{g}_2^{\beta_{i,2}} \;\; \wedge \\ \mathfrak{A}_{i,1} = t_i^{a_{i,1}} b_i^{-\beta_{i,1}} \;\; \wedge \\ C_i = \mathfrak{g}_2^{r_i} \;\; \wedge \\ \tilde{C}_i = \mathfrak{g}_2^{\tilde{r}_i} \end{array} \right) \bigvee \\ \left(\begin{array}{l} t = \hat{b}^x \;\; \wedge \\ t_i = b_i^x \;\; \wedge \\ \dfrac{\tilde{C}_i}{\mathfrak{g}_1} = \mathfrak{g}_2^{\tilde{r}_i} \;\; \wedge \\ \displaystyle\prod_{j=1}^{i} \tilde{C}_j = \mathfrak{g}_1^{k_i} \mathfrak{g}_2^{\beta_{i,5}} \;\; \wedge \\ \mathfrak{A}_{i,3} = \mathfrak{u}_1^{a_{i,3}} \mathfrak{u}_2^{a_{i,4}} \;\; \wedge \\ 1 = \mathfrak{A}_{i,3}^{-e^+_{k_i}} \mathfrak{u}_1^{\beta_{i,3}} \mathfrak{u}_2^{\beta_{i,4}} \;\; \wedge \\ \dfrac{\hat{e}(\mathfrak{A}_{i,4}, w^+)}{\hat{E}_0} = \dfrac{\hat{E}_{\mathfrak{u}_1}^{\beta_{i,4}} \hat{E}_1^{k_i} \hat{E}_2^{\Delta^+_{k_i}} \hat{E}_3^{y^+_{k_i}}}{\hat{e}(\mathfrak{A}_{i,4}, h_0)^{e^+_{k_i}} \hat{E}_{\mathfrak{u}_1 w^+}^{a_{i,4}}} \;\; \wedge \\ C_i = (\mathfrak{g}_1^{s_i})^{\Delta^+_{k_i}} \mathfrak{g}_2^{r_i} \end{array} \right) \end{array} \right\} (M) \qquad (8)$$

where $M = R||\mathfrak{A}_{i,1}||\mathfrak{A}_{i,2}||\mathfrak{A}_{i,3}||\mathfrak{A}_{i,4}||\hat{b}||b_i||t||t_i||\mathsf{aux}^+$ with R being the random challenge given by the SP and $\beta_{i,1} = a_{i,1}x$, $\beta_{i,2} = a_{i,2}x$ for $i \in \mathcal{I}$ and $\beta_{i,3} = e^+_{k_i}a_{i,3}$, $\beta_{i,4} = e^+_{k_i}a_{i,4}$, $\beta_{i,5} = \sum_{j=1}^{i} \tilde{r}_i$.

Alice parses Π'_i as $(\Pi_i, \mathfrak{A}_{i,1}, \mathfrak{A}_{i,2}, \mathfrak{A}_{i,3}, \mathfrak{A}_{i,4})$. To verify Π', the SP also needs to check $\mathfrak{A}_{i,1} \neq 1$ in addition to checking Π_i. Alice parse Π^+ as $(\Pi_1, \dots, \Pi_L)$.

Similarly, Alice computes the proof for the list $\mathcal{L}^-$ and denotes the proof as Π^-. Denote aux^- the set of tuples $(C_i^-, \tilde{C}_i^-)$ for $i = 1$ to L^-. (The values in aux^+ will be denoted as $(C_i^+, \tilde{C}_i^+)$ hereafter.) Parse Π as $(\Pi^+, \Pi^-, \mathsf{aux}^+, \mathsf{aux}^-)$.

The last part of the proof, denote as Π^R, allows Alice to convince the SP that she satisfy the policy $\mathcal{P}$.

Alice first computes $C = \prod_{i=1}^{L^+} C_i^+ / \prod_{i=1}^{L^-} C_i^- = \mathfrak{g}_1^{\mathsf{R}} \mathfrak{g}_2^{\rho}$, where $\rho = \sum_{i=1}^{L^+} r_i + \sum_{i=1}^{L^-} r_i$ where r_i's are the randomness used in creating the values C_i and R is the reputation of Alice. Thus, this value C is a commitment of Alice's reputation! The goal of the proof Π^R is thus to show that the value committed in C, that is, R, is in the range above the threshold n.

In general, exact range proof can be used. However, the group order is known and for any reputation value R', one can always change it to $\mathsf{R}' + p$ so that it is above the required thresholds. To prevent this attack and to conduct an efficient proof, the SP first produce a rough estimation of the maximum possible reputation a user could have, which must be smaller than the group order. Let say the maximum reputation is T, the SP transmits $T-n+1$ signatures to Alice, who then proves to the SP that she has a signature the value committed in C.

This particular instantiation employs the weakly secure signature due to [6], which is sufficient for this purpose. A signature on a message i is a value B_i such that $\hat{e}(B_i, w_m h_0^i) = \hat{e}(g_0, h_0)$ for a public key w_m. In our protocol, the SP sends to Alice $(B_0, \ldots, B_{T-n+1})$. That is, signatures from 0 to $T-n$. Additionally, define $\hat{E}_{\mathfrak{u}_1 w_m}$ as $\hat{e}(\mathfrak{u}_1, w_m)$.

Alice computes $I = \mathsf{R} - n$, $\mathfrak{A}_1 = \mathfrak{u}_1^{r_1}\mathfrak{u}_2^{r_2}$, $\mathfrak{A}_2 = B_I \mathfrak{u}_1^{r_2}$ for some randomly generated $r_1, r_2 \in_R \mathbb{Z}_p$. Alice then computes the following non-interactive proof $\Pi^{R'}$:

$$SPK\left\{\begin{array}{l}(\mathsf{R},\rho, r_1, r_2, e, \beta_1, \beta_2): \\ \quad \prod_{i=1}^{L^+} C_i^+ / \prod_{i=1}^{L^-} C_i^- = \mathfrak{g}_1^{\mathsf{R}}\mathfrak{g}_2^{\rho} \wedge \\ \quad \mathfrak{A}_1 = \mathfrak{u}_1^{r_1}\mathfrak{u}_2^{r_2} \wedge \\ \quad \mathfrak{A}_1^{-n} = \mathfrak{A}_1^{-\mathsf{R}}\mathfrak{u}_1^{\beta_1}\mathfrak{u}_2^{\beta_2} \wedge \\ \quad \dfrac{\hat{e}(\mathfrak{A}_2, w_m)\hat{e}(\mathfrak{A}_2, h_0)^{-n}}{\hat{E}_0} = \hat{e}(\mathfrak{A}_2, h_0)^{-R}\hat{E}_{\mathfrak{u}_1 w_m}^{-r_2}\hat{E}_{\mathfrak{u}_1}^{\beta_2}\end{array}\right\}(M) \qquad (9)$$

where $M = R||\mathfrak{A}_1||\mathfrak{A}_2||\mathsf{aux}^+||\mathsf{aux}^-$ with R being the random challenge given by the SP and $\beta_1 = r_1 I$, $\beta_2 = r_2 I$. Alice parses Π^R as $(\Pi^{R'}, \mathfrak{A}_1, \mathfrak{A}_2)$.

If the threshold n does not change very often, the SP can regard these signatures $(B_0, B_1, \ldots)$ as public parameters and thus saves a lot of transmission bandwidth. Negation of the policy is treated in a similar manner in which the SP publishes the signatures from $n - T_{min}$ to $n-1$, where T_{min} is an estimation of the lowest possible reputation.

The whole proof from Alice thus consists of (Π_x, Π, Π^R).