BLAC: Revoking Repeatedly Misbehaving Anonymous Users Without Relying on TTPs*

Patrick P. Tsang[†], Man Ho Au[‡], Apu Kapadia[§¶], and Sean W. Smith[†]

Dartmouth Computer Science
Technical Report TR2008-635

October 1st, 2008

Abstract

Several credential systems have been proposed in which users can authenticate to service providers anonymously. Since anonymity can give users the license to misbehave, some variants allow the selective deanonymization (or linking) of misbehaving users upon a complaint to a trusted third party (TTP). The ability of the TTP to revoke a user's privacy at any time, however, is too strong a punishment for misbehavior. To limit the scope of deanonymization, systems have been proposed in which users are deanonymized if they authenticate "too many times," such as "double spending" with electronic cash. While useful in some applications, it is not possible to generalize such techniques to more subjective definitions of misbehavior, e.g., it is not possible to block users who "deface too many webpages" on a website.

We present BLAC, the first anonymous credential system in which service providers can revoke the credentials of repeatedly misbehaving users without relying on a TTP. Since revoked users remain anonymous, misbehaviors can be judged subjectively without users fearing arbitrary deanonymization by a TTP. Finally, our construction supports a *d-strikes-out* revocation policy, whereby users who have been subjectively judged to have repeatedly misbehaved at least d times are revoked from the system.

[*] This work was 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.

[†] Department of Computer Science, Dartmouth College, USA

[‡] Centre for Computer and Information Security Research, School of Computer Science and Software Engineering, University of Wollongong, Australia

[§] MIT Lincoln Laboratory, USA

[¶] This research was conducted while Apu Kapadia was at Dartmouth College.

Contents

1 Introduction

While anonymous access to *service providers (SPs)* offers users a high degree of privacy, it can give users the license to misbehave without the fear of punishment. For example, Wikipedia[1] has allowed editors to modify content anonymously, and as a result several users have misbehaved by posting inappropriate content. SPs, therefore, desire some level of accountability against misbehaving users. Several anonymous credential systems have been proposed in which users can be selectively deanonymized or have their accesses linked (pseudonymized) under special circumstances. As we will discuss, for certain applications the existing schemes are either too punitive — deanonymization (or linking) is unreasonably harsh, and often relies on *trusted third parties (TTPs)* capable of revoking a user's privacy at any time — or too restrictive — allowing deanonymization under only certain narrowly defined types of misbehavior.

Deanonymizing a user is not always necessary to discourage misbehavior; in some cases it is sufficient to simply block misbehaving users from making future accesses (while maintaining their anonymity). We call this property *privacy-enhanced revocation*,[2] where revoked users remain anonymous. For example, anonymous access at SPs such as Wikipedia and YouTube[3] empowers users to disseminate content without the fear of persecution — a user may add political content on Wikipedia that is forbidden by his or her government, or post a video of police brutality to YouTube. In such cases, while SPs may want to penalize users who deface webpages or post copyrighted material, it is of paramount importance for SPs to preserve the anonymity of their well-behaving users. By guaranteeing anonymity to *all* users, SPs can penalize misbehavior without the risk of exposing legitimate users.

Anonymous credential systems that support accountability [15, 1, 11, 6, 14, 22] feature a TTP called the *Open Authority (OA)*. The OA is capable of deanonymizing (or linking) the user behind any anonymous authentication. Anonymous credential systems with dynamic membership revocation [2, 12, 7, 24], many of which are constructed from dynamic accumulators [12], also feature a TTP that is capable of deanonymizing (or linking) users. Recently, some of the authors of this paper proposed the Nymble system [21] to allow SPs to block misbehaving users hiding behind an anonymizing network such as Tor [17]. Nymble makes several practical considerations for anonymous IP-address blocking, but it does rely on multiple entities that can collude to deanonymize (or link) a misbehaving user. The existence of such TTPs, however, is undesirable — users can never be assured that their privacy will be maintained by the TTP. Defining the circumstances under which a TTP can expose a user, and ensuring its trustworthiness to judge fairly, is an undue burden on SPs. For such applications, therefore, *a system without TTPs is desirable.*

To eliminate the reliance on TTPs, certain "threshold-based" approaches such as e-cash [3, 9, 10] and *k-Times Anonymous Authentication (k-TAA)* [27, 25, 4, 28] have been proposed. In these schemes, users are guaranteed anonymity unless they authenticate more than a certain number of threshold times. For example, spending an e-coin twice ("double spending," an undesirable action) or authenticating $k + 1$ times in a k-TAA scheme provides the SP with enough information to compute the user's identity. Linkable ring signatures [23, 34, 33] and periodic n-times anonymous authentication [8] also fall into this category. Unfortunately, misbehavior cannot always be defined in terms of threshold values such as double spending. For example, "inappropriate" edits to a Wikipedia page, or "offensive" video uploads to YouTube are usually identified based on human subjectivity and cannot be reduced to "too many authentications". For such applications, therefore,

[1] `http://www.wikipedia.org`

[2] We originally called this concept *anonymous blacklisting* [30]. As will become clear, we differentiate between the action of blacklisting, which may or may not result in revocation.

[3] `http://www.youtube.com`

subjective judging is desirable. Taking the concept of threshold-based schemes one step further, ideally, one would be able to revoke users who have *misbehaved* more than a certain number of times (we call this a *d-strikes-out revocation policy*). k-TAA and related schemes cannot provide such functionality because $k+1$ authentications provide enough information to reduce the privacy of users, and there is no way to penalize $k+1$ misbehaviors instead.

To reiterate, it is important to have an anonymous credential system in which repeatedly misbehaving users can be revoked in a way that (1) preserves their anonymity, (2) is based on subjective definitions of misbehavior, and (3) does not rely on a TTP. Syverson et al. citeSyversonSG97fc present a scheme in which SPs issue users blind tokens, which are renewed at the end of a user's transaction for a subsequent authentication. An SP can block future connections from a user by simply not issuing a new token at the end of a transaction (e.g., if the user fails to pay for continued service). The major drawback to this approach is that misbehavior must be judged while the user is online. Indeed, their scheme was not designed for privacy-enhanced revocation since a user's misbehavior is usually identified long after the user has disconnected. Furthermore, d-strikes-out revocation policies are not possible in such a scheme.

1.1 Our Contributions

We present the *BLacklistable Anonymous Credential system* (BLAC), which was the first[4] construction of an anonymous credential system that supports privacy-enhanced revocation and subjective judging without relying on TTPs that are capable of revoking the privacy of users at will. We formalize the security model for such a system and prove that our construction is secure under this model. Furthermore, we implement our construction and evaluate its performance analytically and experimentally. These results were reported in a conference paper [30] and a technical report [31], which included more details.

In this paper, we make a significant additional contribution by extending our original construction of BLAC to provide more flexible revocation — SPs can specify a d-strikes-out revocation policy, so that users can authenticate anonymously only if they have not misbehaved d or more times. Such a policy forgives a few (i.e., up to $d-1$) misbehaviors, but then blocks users who misbehave repeatedly. Following authentication, users remain anonymous, and SPs learn only whether a user has crossed the threshold of d misbehaviors. The original construction of BLAC is a special case with $d=1$.

Defining the exact meaning of security (and privacy) of BLAC that supports a d-strikes-out revocation policy is a non-trivial task. We have spent considerable effort in formalizing a security model and proving the security of our construction under this model.

Our proposed concept of d-strikes-out is an important improvement on existing threshold schemes such as k-TAA, which deanonymize (or link) users who *authenticate* more than a certain number of times. k-TAA cannot be used to punish "too many *misbehaviors*" because users necessarily suffer degraded privacy after k *authentications*. Our scheme, for the first time, decouples the notion of misbehaviors from authentications — users can verify the SP's blacklist of identified misbehaviors and be assured that their authentication will be anonymous, irrespective of the number of past authentications.

[4] Concurrently and independently, Brickell and Li citeepid proposed a similar scheme called *Enhanced Privacy ID (EPID)*; more recently, the same authors of this paper presented *Privacy-Enhanced Revocation with Efficient Authentication (PEREA)* [32], which alters the semantics of revocation for more efficient authentication. We discuss them in Section 9.

1.2 Paper Outline

We provide a high-level overview of BLAC in Section 2. In Section 3 we present preliminary information on the various cryptographic tools and assumptions used in our construction. In Section 4, we formalize the syntax and security properties for BLAC. We present our construction at a high level in Section 5, and fill in the details of how the various zero-knowledge proofs can be instantiated in Section 6. We analyze the algorithmic complexity and security of our construction in Section 7, and present an experimental evaluation of it in Section 8. We discuss several issues in Section 9, and finally conclude in Section 10.

2 Solution Overview

We give a high-level overview of our *BLacklistable Anonymous Credential system* (BLAC) in this section, and defer the details of its construction to the subsequent sections.

In our system, *users* authenticate to *Service Providers (SPs)* anonymously using *credentials* issued by a *Group Manager (GM)*. The GM is responsible for *enrolling* legitimate users into the system by issuing credentials to them.[5] Each enrolled user privately owns a unique credential, which is not known even by the GM. We emphasize that the GM is *not* a TTP that can compromise the privacy of users, and is trusted only to enroll legitimate users into the system, and issue at most one credential per user. SPs are willing to serve enrolled anonymous users that have never misbehaved thus far, where misbehavior may be arbitrarily defined and subjectively judged by each individual SP. We describe this process next.

The novelty of our approach is that SPs maintain their own *blacklists* of misbehaving users without knowing the identity of the misbehaving users. Users anonymously authenticating to the SP must first prove that there are fewer than d entries on the blacklist corresponding to that user (otherwise authentication will fail). Following a user's authentication, SPs store a *ticket* extracted from the *protocol transcript* of the authentication and if the user is later deemed to have misbehaved during the *authenticated session*, possibly long after the user has disconnected, the SP can add the ticket as an entry into its blacklist.[6] If a user Alice detects that she is on the blacklist (d or more times), she terminates the authentication and disconnects immediately. The SP, therefore, learns only that some anonymous revoked user was refused a connection, and does not learn the identity of the revoked user. Users that are not revoked will be able to authenticate successfully, and the SPs learn only that the user is not on the blacklist d or more times. Furthermore, our system allows SPs to *remove* entries from the blacklist, thereby forgiving past misbehaviors. Depending on the severity of misbehavior, a user may be blacklisted for varying periods of time — using inappropriate language could correspond to being blacklisted for one week, whereas posting copyrighted material could correspond to blacklisting for one month. Users are always assured that if they successfully authenticate to an SP their access will always remain anonymous — all that an SP can do is block future accesses by a misbehaving user.

2.1 A Glimpse into Tickets

Tickets are a vital object in BLAC. A ticket is the only piece in the authentication protocol transcript that contains information about the identity of the authenticating user. Jumping ahead, tickets in BLAC have the form of (s, t), where the serial number s is a bit-string and the tag t is an element in

[5] Who is a legitimate user and how to verify such legitimacy are application-dependent.

[6] In practice, the SP may privately log arbitrary information about an authenticated session that is necessary for it to judge at a later time whether the anonymous user misbehaved during that session.

a DDH-hard group $\mathbb{G}$ (to be defined later). A user produces a new ticket during each authentication by randomly choosing s and computing t as $H(s||\texttt{sid})^x$, where sid is the target server's identity, x is from the user's credential and H is a secure cryptographic hash function.

Here we highlight three features of such a ticket construction. First, it allows every user to produce tickets that are different and more importantly unlinkable, for otherwise SPs would be able to tell if two authentications are from the same user. Second, users can prove and disprove to the SPs that a ticket belongs to them. This allows, on the one hand, for users to prove that they are not blacklisted, and, on the other hand, the prevention of users fabricating incorrect tickets to circumvent blacklisting and/or impersonating and hence framing other users. Finally, it provides the option to allow or disallow the sharing of blacklist entries (tickets) between SPs. Sharing a blacklist entry would allow multiple SPs to block a user who misbehaved at one of the SPs. We will first present the system where such sharing is *disallowed* and then point out how to allow sharing in Section 9.

2.2 Supporting a d-strikes-out Revocation Policy

We present a new extension to our original BLAC construction [30, 31] to provide more flexible blacklisting of users. In our previous construction, each blacklist entry represented an instance of a misbehavior, where each misbehavior was unlinkable to the other misbehaviors. As a result, SPs could not express policies spanning multiple misbehaviors, e.g., such as a "three-strikes-out" revocation policy where up to two misbehaviors by the same user are forgiven. Our extended construction now supports a "d-strikes-out" revocation policy, where users are allowed access as long as they have misbehaved fewer than d times. In this more general construction, our original construction is simply a special case that supports a "1-strike-out" revocation policy, where a single misbehavior would result in the user being blocked. As noted in Section 1, such functionality is not possible in schemes such as k-TAA, which necessarily reduces a user's privacy after k *authentications*.

3 Preliminaries

In this section we outline the assumptions and cryptographic tools that we use as building blocks in our BLAC construction.

3.1 Notation and terminology

$|S|$ represents the cardinality of a set S. If S is a non-empty set, $a \in_R S$ means that a is an element in S drawn uniformly at random from S. $A \subseteq_d S$ denotes that A is a subset of S of cardinality d. We denote by $\mathbb{N}$ the set of natural numbers $\{1, 2, \ldots\}$ and by $\mathbb{Z}^*$ the set of non-negative integers $\{0, 1, 2, \ldots\}$. If $n \in \mathbb{Z}^*$, we write $[n]$ to mean the set $\{1, 2, \ldots, n\}$; $[0]$ is the empty set $\emptyset$. If $s, t \in \{0, 1\}^*$, then $s||t \in \{0, 1\}^*$ is the concatenation of binary strings s and t.

A sequence $Q = (a_1, a_2, \ldots, a_\ell)$ is an ordered list of $\ell \in \mathbb{Z}^*$ (not necessarily unique) natural numbers $a_1, a_2, \ldots, a_\ell$. Q is *index-bounded-from-below*, or simply *bounded*, if $a_i \geq i$ for all $i \in [\ell]$. Q is empty if $\ell = 0$; an empty sequence is by definition always *bounded*. For any $k \in \mathbb{N}$, Q can be partitioned into k (possibly empty) subsequences $Q_1, Q_2, \ldots, Q_k$ in an order-preserving manner, i.e., an element a is before another element b in a subsequence only if a is also before b in the original sequence. We call the set $P = \{Q_1, Q_2, \ldots, Q_k\}$ a *k-partitioning* of Q. There is at least one *k-partitioning* of Q for all $k \in \mathbb{N}$. for all $Q_j \in P$. Finally, a sequence Q is *k-boundedly-partitionable*, or simply *k-partitionable*, if there exists a *bounded k-partitioning* of Q.

We note the following two facts. (1) If Q is *k-partitionable*, then Q is also *k'-partitionable*, for all $k' > k$. Thus, if Q is not *k-partitionable*, then Q is also not *k'-partitionable* for all $k' \in [k-1]$. (2) If Q is *k-partitionable*, then any subsequence Q' of Q is also k-partitionable. Thus, if Q is *not k-partitionable*, then any sequence Q' that contains Q as a subsequence is also *not k-partitionable*.

3.2 Pairings

A *pairing* is a bilinear mapping from a pair of group elements to a group element. Specifically, let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}$ be multiplicative cyclic groups of order p. Suppose P and Q are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. A function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}$ is said to be a pairing if it satisfies the following properties:

- *(Bilinearity.)* $\hat{e}(A^x, B^y) = \hat{e}(A, B)^{xy}$ for all $A \in \mathbb{G}_1$, $B \in \mathbb{G}_2$ and $x, y \in \mathbb{Z}_p$.
- *(Non-degeneracy.)* $\hat{e}(P, Q) \neq 1$, where 1 is the identity element in $\mathbb{G}$.
- *(Efficient Computability.)* $\hat{e}(A, B)$ can be computed efficiently (i.e., in polynomial time) for all $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$.

3.3 Mathematical Assumptions

The security of our BLAC construction requires the following two assumptions:

Assumption 1 (DDH) The *Decisional Diffie-Hellman (DDH)* problem in group $\mathbb{G}$ is defined as follows: On input of a quadruple $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, output 1 if $c = ab$ and 0 otherwise. We say that the DDH assumption holds in group $\mathbb{G}$ if no probabilistic polynomial time (PPT) algorithm has non-negligible advantage over random guessing in solving the DDH problem in $\mathbb{G}$.

Assumption 2 (q-SDH) The *q-Strong Diffie-Hellman (q-SDH)* problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follows: On input of a $(q+2)$-tuple $(g_0,\ h_0,\ h_0^x,\ h_0^{x^2},\ \ldots,\ h_0^{x^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, output a pair $(A, c) \in \mathbb{G}_1 \times \mathbb{Z}_p$ such that $A^{(x+c)} = g_0$ where $|\mathbb{G}_1| = p$. We say that the q-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no PPT algorithm has non-negligible advantage in solving the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$.

3.4 Proofs of Knowledge

In a *Zero-Knowledge Proof of Knowledge (ZKPoK)* protocol [19], a prover convinces a verifier that some statement is true without the verifier learning anything except the validity of the statement. Σ-protocols are a special type of three-move ZKPoK protocols, which can be converted into non-interactive *Signature Proof of Knowledge (SPK)* schemes, or simply signature schemes [20] that are secure under the *Random Oracle (RO)* Model [5].

In the following, we review several Σ-protocols that will be needed as building blocks in our construction. We follow the notation introduced by Camenisch and Stadler citeCamenischS97crypto. For instance, $PK\{(x) : y = g^x\}$ denotes a Σ-protocol that proves the knowledge of $x \in \mathbb{Z}_p$ such that $y = g^x$ for some $y \in \mathbb{G}$. The corresponding signature scheme resulting from the application of the Fiat-Shamir heuristic to the above Σ-protocol is denoted by $SPK\{(x) : y = g^x\}(M)$.

3.4.1 Knowledge and Inequalities of Discrete Logarithms

Let $g, b \in \mathbb{G}$ and $b_i \in \mathbb{G}$ for all i be generators of some group $\mathbb{G}$ of prime order p such that their relative discrete logarithms are unknown. One can prove in zero-knowledge the knowledge of the discrete logarithm $x \in \mathbb{Z}_p$ of $y \in \mathbb{G}$ in base g by using the Σ-protocol:

$$PK\left\{(x) : y = g^x\right\},$$

the construction of which first appeared in Schnorr identification [26]. As we shall see, our BLAC construction requires the SPK of this protocol to prove the correctness of tickets.

One can further prove in zero-knowledge that x does *not* equal $\log_b t$, the discrete log of $t \in \mathbb{G}$ in base b, using the Σ-protocol:

$$PK\left\{(x) : y = g^x \wedge t \neq b^x\right\},$$

the most efficient construction of which is due to Camenisch and Shoup cite[§5]CamenischS03crypto.

In our BLAC construction we will need a generalized version of the above Σ-protocol to prove that a user is not currently on the blacklist. In particular, we need a protocol that allows one to prove in zero-knowledge that, for some $n > 1$ and for all $i = 1$ to n, $x \neq \log_{b_i} t_i$, where $t_i \in \mathbb{G}$. That is,

$$PK\left\{(x) : y = g^x \wedge \left(\bigwedge_{i=1}^{n} t_i \neq b_i^x\right)\right\}.$$

Such a Σ-protocol can be constructed by applying a technique due to Cramer et al. cite-CramerDS94crypto to Camenisch and Shoup's construction mentioned above.[7]

3.4.2 Proving d out of n DL representations

Let n, d be positive integers such that $d \leq n$. Let $\tilde{A}_i, b_i, t_i$ be elements in some group $\mathbb{G}$ of prime order p such that there exist $\mathcal{I} \subseteq_d ([n])$ and $\beta, \rho \in \mathbb{Z}_p$ such that $\tilde{A}_i = b_i^\beta t_i^{-\rho}$ for all $i \in \mathcal{I}$. One can prove in zero-knowledge the knowledge of such (β, ρ) by using the Σ-protocol:

$$PK\left\{(\beta, \rho) : \bigvee_{\mathcal{I} \subseteq_d ([n])} \bigwedge_{i \in \mathcal{I}} \tilde{A}_i = b_i^\beta t_i^{-\rho}\right\},$$

the construction of which was first presented by Cramer et al. citeCramerDS94crypto with $O(n)$ complexity both during signing and verification.

3.4.3 BBS+ Signatures

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 their relative discrete logarithms are unknown, where ψ is a computable isomorphism and $(\mathbb{G}_1, \mathbb{G}_2)$ is a pair of groups of prime order p in which the q-SDH assumption holds. Let e be a pairing defined over the pair of groups. One can prove possession of a tuple $(A, e, x, y) \in \mathbb{G}_1 \times \mathbb{Z}_p^3$ such that $A^{e+\gamma} = g_0 g_1^x g_2^y$, or equivalently, $\hat{e}(A, wh_0^e) = \hat{e}(g_0 g_1^x g_2^y, h_0)$, where $w = h_0^\gamma$, by the Σ-protocol:

$$PK\left\{(A, e, x, y) : A^{e+\gamma} = g_0 g_1^x g_2^y\right\}.$$

[7] The technique describes a general method of constructing proofs of disjunction or conjunction of any of the two statements about knowledge of discrete logarithms.

Boneh et al. cite[§4]BonehBS04crypto present the construction of this protocol, which is secure under the Decision-linear Diffie-Hellman assumption. Au et al. citeAuSM06scn provide a modified construction that does not rely on this assumption. As first pointed out by Camenisch and Lysyanskaya citeCamenischL04crypto, the protocol's corresponding SPK is actually the SDH-variant of CL signatures [13], which Au et al. citeAuSM06scn refer to as BBS+ Signatures. Our BLAC construction will need this protocol as a building block for users to prove that they are enrolled in the system. We will employ Au et al.'s citeAuSM06scn construction to avoid the need of less standard assumptions.

4 Model

We present the syntax of BLAC, followed by the security properties that any BLAC construction must satisfy.

4.1 Syntax

The entities in BLAC are the *Group Manager (GM)*, a set of *Service Providers (SPs)* and a set of *users*. BLAC consists of the following protocols:

4.1.1 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 consisting 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.

4.1.2 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 `cred`, which she keeps private to herself, and is thereby enrolled as a member in the group of registered users.

4.1.3 Authentication

This protocol is executed between a user with credential `cred` and an SP. The initial input to the user is her credential. The initial input to the SP is its blacklist and a threshold value. When an execution of the protocol terminates, the SP 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 is useful for the SP to blacklist the authenticating user, as described next.

4.1.4 Blacklist Management

This is a suite of three algorithms: *Extract*, *Add* and *Remove*, which are executed by SPs for managing their blacklists. On input of an authentication protocol transcript, *Extract* extracts and returns a *ticket* from the transcript. A *blacklist* is a collection of tickets. On input of a blacklist and a ticket, *Add* returns a new blacklist that contains all the tickets in the input blacklist as well as the input ticket. On the other hand, on input of a blacklist and a ticket, *Remove* returns a new blacklist that contains all the tickets in the input blacklist, except the one(s) equivalent to the input ticket.[8]

When we say that a user Alice is *blacklisted* by an SP, we mean that there exists an authentication between Alice and the SP such that the SP has added the ticket extracted from the authentication transcript to its blacklist and has not removed it (yet). Otherwise Alice is *not blacklisted* by the SP. Also, we say that Alice is *misbehaving*, and thus *revoked*, with respect to the SP if she has been blacklisted by the SP at least a threshold number of times d. Otherwise, she is *well-behaving*.

Any BLAC construction must be correct:

Definition 1 (Correctness) A construction of the BLAC system is correct if all entities in the system being honest (i.e., they follow the system's specification) implies that for any enrolled user Alice and for any SP, Alice is able to successfully authenticate herself to the SP with overwhelming probability if Alice has been blacklisted by the SP fewer than a specified threshold number of times.

4.2 Security Notions

We now give informal definitions of the various security properties that a BLAC construction must possess. Their formal definitions will be given in the next subsection.

4.2.1 Mis-authentication Resistance

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

4.2.2 Blacklistability

Any SP may blacklist a user who has authenticated successfully at any later time. In a BLAC construction with *blacklistability*, SPs are assured to accept authentications from only well-behaving users, i.e., users who are blacklisted fewer than a threshold number of times. As a consequence, misbehaving users are revoked from the system, and they will no longer be able to successfully authenticate themselves to the SP until enough of their misbehaviors are unblacklisted by the SP.

4.2.3 Anonymity

In a system with *anonymity*, all that SPs can infer about the identity of an authenticating user is whether the user is or was blacklisted at the time of protocol execution, regardless of whatever the SPs do afterwards, such as arbitrarily manipulating their blacklists.

4.2.4 Non-frameability

A user Alice is framed if she is not currently blacklisted by an honest SP, but is unable to successfully authenticate herself to the SP. In a BLAC construction with *non-frameability*, well-behaving users can always successfully authenticate themselves to honest SPs.

Any BLAC construction must be secure:

[8] We don't define the equivalence of tickets here because it is construction-dependent.

Definition 2 (Security) A BLAC construction is secure if it has mis-authentication resistance, blacklistability, anonymity and non-frameability.

4.3 Formal Definitions

We use a game-based approach to define the notion of security formally. The adversary's capabilities are modeled by arbitrary and adaptive queries to oracles, which are stateful and together share a private state denoted as state, which contains three counters $\mathsf{i}, \mathsf{j}, \mathsf{k}$, and six sets $\mathcal{I}_P, \mathcal{I}_A, \mathcal{I}_B, \mathcal{J}_P, \mathcal{J}_A, \mathcal{K}_A$. Initially, the three counters are 0 and the six sets are $\emptyset$. We next describe the oracles, which are simulated by the simulator $\mathcal{S}$ during the games.

- P-Reg(). This oracle allows the adversary to register an honest user with the honest GM. Upon invocation, the oracle increments i by 1, simulates the *Registration* protocol between an honest user and the honest GM, sets $\mathsf{state} := \mathsf{state}||\langle \mathsf{i}, \varpi_{\mathsf{i}}, \mathtt{cred}_{\mathsf{i}} \rangle$, where ϖ_{i} is the resulting protocol transcript and $\mathtt{cred}_{\mathsf{i}}$ is the resulting user credential, adds i to $\mathcal{I}_P$ and finally outputs $(\varpi_{\mathsf{i}}, \mathsf{i})$ to the adversary. The newly registered user is indexed by i.

- A-Reg(). This oracle allows the adversary to register a corrupt user with the honest GM. Upon invocation, the oracle increments i by 1, plays the role of the GM and interacts with the adversary in the *Registration* protocol, sets $\mathsf{state} := \mathsf{state}||\langle \mathsf{i}, \varpi_{\mathsf{i}}, \perp \rangle$, where ϖ_{i} is the protocol transcript, adds i to $\mathcal{I}_A$ and finally outputs i to the adversary. The user is indexed by i.

- B-Reg(). This oracle allows the adversary to register an honest user with the corrupt GM. Upon invocation, the oracle increments i by 1, plays the role of a user and interacts with the adversary in the *Registration* protocol, sets $\mathsf{state} := \mathsf{state}||\langle \mathsf{i}, \perp, \mathtt{cred}_{\mathsf{i}} \rangle$, where $\mathtt{cred}_{\mathsf{i}}$ is the credential issued to the user by the adversary, adds i to $\mathcal{I}_B$ and finally outputs i to the adversary. The user is indexed by i.

- Corrupt-U(i). This oracle allows the adversary to corrupt an honest user. On input $i \in \mathcal{I}_B \cup \mathcal{I}_P$, the oracle removes i from $\mathcal{I}_B$ or $\mathcal{I}_P$, adds i to $\mathcal{I}_A$, and finally outputs $\mathtt{cred}_i$ to the adversary.

- Add-SP($\mathtt{sid}$). This oracle allows the adversary to introduce an SP with *fresh* identity $\mathtt{sid} \in \{0,1\}^*$ into the system. Upon invocation, the oracle increments j by 1, adds it to $\mathcal{J}_P$, and finally outputs it to the adversary. The SP is indexed by j.

- Corrupt-SP(j). This oracle allows the adversary to corrupt an honest SP. On input $j \in \mathcal{J}_P$, the oracle removes j from $\mathcal{J}_P$ and adds it to $\mathcal{J}_A$.

- P-Auth(i, j, d). This oracle allows the adversary to eavesdrop an authentication run between an honest user and an honest SP. On input (i, j, d) such that $i \in \mathcal{I}_P \cup \mathcal{I}_B$, $j \in \mathcal{J}_P$ and $d \in \mathbb{N}$, the oracle increments k by 1, simulates (using $\mathtt{cred}_i$) the *Authentication* protocol between honest user i and honest SP j assuming a threshold value of d, sets $\mathsf{state} := \mathsf{state}||\langle \mathsf{k}, \varpi_{\mathsf{k}} \rangle$, where ϖ_{k} is the resulting protocol transcript, and finally outputs $(\mathsf{k}, \varpi_{\mathsf{k}})$ to the adversary.

- A-Auth(j, d). This oracle allows a corrupt user to authenticate to an honest SP. On input $j \in \mathcal{J}_P$ and $d \in \mathbb{N}$, the oracle increments k by 1, plays the role of SP j assuming a threshold value of d and interacts with the adversary in the *Authentication* protocol, adds k to $\mathcal{K}_A$, sets $\mathsf{state} := \mathsf{state}||\langle \mathsf{k}, \varpi_{\mathsf{k}} \rangle$, where ϖ_{k} is the resulting protocol transcript, and finally outputs k to the adversary.

- B-Auth(i, j). This oracle allows the adversary to have an honest user authenticate to a corrupt SP. On input $i \in \mathcal{I}_B \cup \mathcal{I}_P$ and $j \in \mathcal{J}_A$, the oracle increments k by 1, plays the role of user i to authenticate to SP j and interacts with the advesary in the *Authentication* protocol, sets $\mathsf{state} := \mathsf{state}||\langle \mathsf{k}, \varpi_{\mathsf{k}} \rangle$, where ϖ_{k} is the resulting protocol transcript, and finally outputs k to the adversary.

- Add-To-BL(j, k). This oracle allows the adversary to influence an honest SP to judge a user as have misbehaved during an authenticated session. On input $j \in \mathcal{J}_P$ and $k \in [\mathsf{k}]$, the oracle adds the ticket $\tau_k = \mathit{Extract}(\varpi_k)$ to SP j's blacklist.

- Remove-From-BL(j, τ). This oracle allows the adversary to influence an honest SP to forgive a user for her misbehavior during an authenticated session. On input $j \in \mathcal{J}_P$ and τ such that τ is in SP j's blacklist, the oracle removes τ from that blacklist.

We remark that queries to P-Reg and A-Reg do not interleave because the honest GM registers users one at a time; queries to Add-To-BL$(j, \cdot)$ and Remove-From-BL$(j, \cdot)$ do not interleave with one another, or with queries to P-Auth or A-Auth because honest SPs update their blacklists only when no authentication is in progress. Queries to P-Auth is atomic, but we allow interleaving among queries to P-Auth, A-Auth and B-Auth.

4.3.1 Mis-authentication resistance and blacklistability

Mis-authentication resistance is in fact implied by Blacklistability: if someone can authenticate to an SP without having registered, she can authenticate after being blacklisted by mounting an attack against mis-authentication resistance. The following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$ formally defines Blacklistability.

Setup Phase $\mathcal{S}$ takes a sufficiently large security parameter and generates $\mathtt{gpk}$ and $\mathtt{gsk}$ according to *Setup*. $\mathtt{gpk}$ is given to $\mathcal{A}$.

Probing Phase $\mathcal{A}$ is allowed to issue queries to all the oracles except B-Reg. In other words, the GM is always honest.

End Game Phase $\mathcal{A}$ outputs $j, n \in \mathbb{N}$, $k_1, k_2, \ldots, k_n \in \mathcal{A}_A$.

Let $S_{\mathcal{O}}$ be the sequence of all oracle queries made throughout the game in chronological order. $\mathcal{A}$ wins the game if all of the following hold:

1. There exist $d_1, d_2, \ldots, d_n \in \mathbb{N}$ such that the sequence $S^*_{\mathcal{O}} =$

$$\left(\begin{array}{rlc} k_1 \leftarrow & \text{A-Auth}(j, d_1), & \text{Add-To-BL}(j, k_1), \\ k_2 \leftarrow & \text{A-Auth}(j, d_2), & \text{Add-To-BL}(j, k_2), \\ & \vdots & \\ k_{n-1} \leftarrow & \text{A-Auth}(j, d_{n-1}), & \text{Add-To-BL}(j, k_{n-1}), \\ k_n \leftarrow & \text{A-Auth}(j, d_n) & \end{array}\right)$$

   is a subsequence of $S_{\mathcal{O}}$.

2. In all n A-Auth queries in $S^*_{\mathcal{O}}$, the honest SP j as simulated by $\mathcal{S}$ terminated the *Registration* protocol successfully.

3. Without loss of generality, before the query $k_n \leftarrow$ A-AUTH(j, d_n), $\mathcal{A}$ never queried REMOVE-FROM-BL(j, *Extract*(ϖ_{k_i})), where $\langle k_i, \varpi_{k_i} \rangle \in \mathsf{state}$, for all $i \in [n-1]$.

4. Either $|\mathcal{I}_A| = 0$ or the sequence $D = (d_1, d_2, \ldots, d_n)$ is *not* $|\mathcal{I}_A|$*-partitionable.*

This completes the description of the game.

Throughout the game, adversary $\mathcal{A}$ has corrupted $|\mathcal{I}_A|$ registered users. If $|\mathcal{I}_A| = 0$, then the existence of even a single successful A-AUTH query implies that the adversary has broken mis-authentication resistance and thus blacklistability. Otherwise, i.e., $|\mathcal{I}_A| > 0$, the adversary wins only if D is not $|\mathcal{I}_A|$*-partitionable.* Below we provide more explanation for this latter case.

Consider the contrary that D is $|\mathcal{I}_A|$*-partitionable.* Let $P = (D_1, D_2, \ldots, D_{|\mathcal{I}_A|})$ be one such partitioning. Adversary $\mathcal{A}$ could have successfully made the n A-AUTH queries in D according to the following strategy: use the credential of the i-th corrupted user in the j-th A-AUTH query if $d_j \in D_i$. In fact, this strategy is always feasible for any BLAC construction with the *correctness* property; D could just have been a sequence of legitimate authentications in the system where there are only honest participants. Therefore, an $|\mathcal{I}_A|$*-partitionable* sequence D is *not* considered as a breach in blacklistability and thus a victory of the adversary in the game.

Now, consider the case when $D = (d_1, \ldots, d_n)$ is *indeed not* $|\mathcal{I}_A|$*-partitionable.* There is always an $n' \leq n$ such that $D' = (d_1, \ldots, d_{n'})$ is also *not* $|\mathcal{I}_A|$*-partitionable.* Let n^* be the smallest such n'. The n^*-th A-AUTH is considered to be a breach in blacklistability for the following reason: no matter in what order and who among any group of $|\mathcal{I}_A|$ honest registered users have authenticated in the first $n^* - 1$ successful authentications, each of them has already authenticated at least d_{n^*} times by the time the n^*-th authentication is about to be made. Since the n^*-th authentication has a threshold value of d_{n^*}, none of them should be able to successfully authenticate in the n^*-th authentication.

4.3.2 Anonymity

The following game between the simulator $\mathcal{S}$ and adversary $\mathcal{A}$ formally defines anonymity.

Setup Phase $\mathcal{S}$ takes a sufficiently large security parameter and generates gpk and gsk, which are given to $\mathcal{A}$.

Probing Phase $\mathcal{A}$ is allowed to issue queries to all the oracles except P-REG and A-REG. Oracle queries can be interleaved and/or span the Challenge Phase and Probing Phase 2.

Challenge Phase $\mathcal{A}$ outputs $i_0^*, i_1^* \in \mathcal{I}_B$, $j^* \in$ [j] and $d^* \in \mathbb{N}$. $\mathcal{S}$ then flips a fair coin $\hat{b} \in \{0, 1\}$. $\mathcal{A}$ queries P-AUTH$(\perp, j^*, d^*)$ if $j^* \in \mathcal{J}_P$, or B-AUTH$(\perp, j^*, d^*)$ otherwise. Notice that $\mathcal{A}$ leaves i unspecified in either query. $\mathcal{S}$ answers the query assuming $i = i_{\hat{b}}^*$. Let ϖ_k^* be the resulting transcript. Furthermore, let d_0^* (resp. d_1^*) be the current number of tickets on the blacklist sent from SP j^* to $\mathcal{S}$ during the P-AUTH or B-AUTH query that are extracted from the trancript of an authentication involving user i_0^* (resp. i_1^*).

Probing Phase 2 $\mathcal{A}$ is allowed to issue queries as in the Probing Phase, except that queries to CORRUPT-U(i_0^*) or CORRUPT-U(i_1^*) are not allowed.

End Game Phase $\mathcal{A}$ outputs a guess bit b'. $\mathcal{A}$ wins the game if $\hat{b} = b'$ and at least one of the following is true:

- *(Case I.)* Both d_0^* and d_1^* are smaller than d^* and $\mathcal{A}$ never queried ADD-TO-BL$(*, k^*)$.

- *(Case II.)* Both d_0^* and d_1^* are greater than or equal to d^*.

  The condition of Case I implies that $\mathcal{A}$ cannot blacklist the challenge authentication in an attempt to break anonymity. This prevents the trivial attack in which $\mathcal{A}$ simply blacklists the authentication and has the two users (i_0^* and i_1^*) attempt to authenticate. Whoever fails to authenticate will be the underlying user of the challenge authentication.

4.3.3 Non-frameability

The following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$ formally defines Non-frameability.

Setup Phase $\mathcal{S}$ takes a sufficiently large security parameter and generates `gpk` and `gsk`, which are given to $\mathcal{A}$.

Probing Phase $\mathcal{A}$ is allowed to issue queries to all the oracles except P-Reg and A-Reg. Oracle queries may span over the End Game Phase.

End Game Phase $\mathcal{A}$ outputs $i^* \in \mathcal{I}_B$, $j^* \in \mathcal{J}_P$ and $d^* \in \mathbb{N}$. Let d_i^* be the number of unique Add-To-BL(j^*, k), where k is such that $(\cdot, k)$ is the output of a P-Auth$(i^*, j^*, \cdot)$ query, minus the number of unique Remove-From-BL(j^*, τ), where τ is the ticket extracted from the transcript of an authentication involving user i^*. $\mathcal{S}$ then runs P-Auth(i^*, j^*, d^*). $\mathcal{A}$ wins the game if $d^* > d_i^*$ and SP terminates *unsuccessfully* in the P-Auth query.

5 System Construction

We now detail our cryptographic construction and assess its security and efficiency. For simplicity, we first present our construction without flexible blacklisting, and then show how the construction can be extended to support a d-strikes-out revocation policy.

5.1 Parameters

Let λ, ℓ be sufficiently large security parameters. 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.[9] Let $H_0 : \{0,1\}^* \to \mathbb{G}$ and $H : \{0,1\}^* \to \mathbb{Z}_p$ be secure cryptographic hash functions.

5.2 Setup

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

5.3 Registration

Upon successful termination of this protocol between a user Alice and the GM, Alice obtains a credential in the form of (A, e, x, y) such that $A^{e+\gamma} = g_0 g_1^x g_2^y$, and (A, e, x, y) is known only to Alice. The private input to the GM is the group secret key `gsk`.

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

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

2. Alice sends a pair (C, Π_1) to the GM, where $C = g_1^x g_2^{y'} \in \mathbb{G}_1$ is a commitment of $(x, y') \in_R \mathbb{Z}_p^2$ 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 $\hat{e}(A, wh_0^e) \neq \hat{e}(g_0 g_1^x g_2^y, h_0)$. Otherwise she outputs $\texttt{cred} = (A, e, x, y)$ as her credential.

To prevent the possibility of a concurrent attack [16], we require that users must be registered sequentially, as opposed to concurrently.

5.4 Authentication: The Special Case

For an easier understanding, we first describe the protocol construction assuming the special case when the SP enforces a 1-strike-out revocation policy, i.e., it uses a threshold value of 1. In the next subsection, we show how the construction can be extended to allow a d-strike-out revocation policy, with any $d \geq 1$.

During an execution of this protocol between a user Alice and the SP, Alice's private input is her credential $\texttt{cred} = (A, e, x, y)$. Let $\texttt{sid} \in \{0,1\}^*$ 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 a pair (BL, m), where $m \in_R \{0,1\}^\ell$ is a random challenge and $BL = \langle \tau_1, \ldots, \tau_n \rangle$ is its current blacklist and $\tau_i = (s_i, t_i) \in \{0,1\}^\ell \times \mathbb{G}$, for $i = 1$ to n, is the i-th ticket in the blacklist.

2. *(Blacklist Inspection.)* Alice computes, for $i = 1$ to n, the bases $b_i = H_0(s_i || \texttt{sid})$. She returns as `failure` if tag $t_i = b_i^x$ for some i (indicating that she is blacklisted). She proceeds otherwise.

3. *(Proof Generation.)* Alice returns to the SP a pair (τ, Π_2), where $\tau = (s, t) \in \{0,1\}^\ell \times \mathbb{G}$ is a ticket generated by randomly choosing a serial $s \in_R \{0,1\}^\ell$ and computing the base b as $H_0(s || \texttt{sid})$ and then the tag t as b^x, and Π_2 is a signature proof of knowledge of:

$$SPK_2 \left\{ (A, e, x, y) : A^{e+\gamma} = g_0 g_1^x g_2^y \ \wedge \ t = b^x \ \wedge \ \left(\bigwedge_{i \in [n]} t_i \neq b_i^x \right) \right\} (m) \tag{2}$$

on the challenge m, which proves:

   (a) $A^{e+\gamma} = g_0 g_1^x g_2^y$, i.e., Alice is a group member,

   (b) $t = b^x = H_0(s || \texttt{sid})^x$, i.e., the ticket τ is correctly formed.

   (c) $\bigwedge_{i=1}^n t_i \neq H_0(s_i || \texttt{sid})^x$, i.e., Alice is not currently on the SP's blacklist, and

4. *(Proof Verification.)* The SP returns `failure` if the verification of Π_2 returns `invalid`.[10] Otherwise it returns `success`.

The protocol transcript of a successful authentication at the SP is thus $\varpi = \langle \texttt{sid}, BL, m, \tau, \Pi_2 \rangle$. As discussed, the SP stores ticket τ extracted from the transcript, along with information logging Alice's activity within the authenticated session.

5.5 Authentication: The General Case

We now modify the protocol for *Authentication* presented in Section 5.4 to support a d-strikes-out revocation policy. Our extension does not alter the time and communication complexities of the authentication protocol, which remain $O(n)$, where n is the size of the blacklist.

The inputs to each of user Alice and the SP in the protocol in the general case are the same as those in the special case, except that the SP additionally gets a threshold value d. When the protocol terminates, the SP outputs `success` or `failure`, indicating whether the SP should consider the authentication attempt successful. Authentication succeeds only if there are less than d entries corresponding to Alice's credential in the blacklist.

1. *(Challenge.)* In addition to (BL, m), the SP sends to Alice the threshold value $d \in [n]$, where n is the size of BL.

2. *(Blacklist Inspection.)* Alice computes, for $i = 1$ to n, the bases $b_i = H_0(s_i || \texttt{sid})$. She returns `failure` if tag $t_i = b_i^x$ for d or more distinct i's (indicating that she has reached the blacklisting threshold). She proceeds otherwise.

3. *(Proof Generation.)* Π_2 is instead a signature proof of knowledge of:

$$SPK_3 \left\{ (A, e, x, y) : \begin{array}{c} A^{e+\gamma} = g_0 g_1^x g_2^y \ \wedge \ t = b^x \ \wedge \\ \left(\bigvee_{\mathcal{I} \subseteq_{(n-d+1)} [n]} \bigwedge_{i \in \mathcal{I}} t_i \neq b_i^x \right) \end{array} \right\} (m) \tag{3}$$

   on the challenge m, which proves:

   (a) $A^{e+\gamma} = g_0 g_1^x g_2^y$, i.e., Alice is a group member,

   (b) $t = H_0(s || \texttt{sid})^x$, i.e., the ticket τ is correctly formed.

   (c) $\bigwedge_{i=1}^{n} t_i \neq H_0(s_i || \texttt{sid})^x$, i.e., Alice is not currently on the SP's blacklist d times or more, and

4. *(Proof Verification.)* The verification of Π_2 changes accordingly.

The protocol transcript of a successful authentication at the SP thus becomes $\varpi = \langle \texttt{sid}, BL, m, d, \tau, \Pi_2 \rangle$.

5.6 Blacklist Management

The three algorithms are all very simple and efficient. *Extract*(ϖ) returns ticket τ in the input transcript $\varpi = \langle BL, m, \tau, \Pi_2 \rangle$. *Add*$(BL, \tau)$ returns blacklist BL', which is the same as the input blacklist BL, except with the input ticket τ appended to it. *Remove*(BL, τ) returns blacklist BL', which is the same as the input blacklist BL, except with all entries equal to the input ticket τ dropped.

[10] The SP also terminates with `failure` if the blacklist is being updated concurrently. This behavior ensures that if a user is blacklisted at time t, she cannot authenticate to the SP after t or until she is unblacklisted.

6 Instantiation of ZKPoK Protocols

The ZKPoK protocols SPK_1, SPK_2 and SPK_3 presented above require instantiation. We omit spelling out the relatively trivial instantiation of SPK_1. In the following, we instantiate SPK_3. Note that SPK_2 is a special case of SPK_3 at $d = 1$.

6.1 SPK_3

6.1.1 Signing

Let $(A, e, x, y) \in \mathbb{G}_1 \times \mathbb{Z}_p^3$ be a tuple such that $A^{e+\gamma} = g_0 g_1^x g_2^y$, $t = b^x$, and $\bigwedge_{i \in \mathcal{J}} (t_i \neq b_i^x)$ for some $\mathcal{J} \subseteq_{(n-d+1)} [n]$. To produce a proof Π_3 for SPK_3 on message $m \in \{0,1\}^*$, a prover with the knowledge of (A, e, x, y) does the following.

1. Produce auxiliary commitments

$$\mathsf{aux} = (C_1, C_2, C_3, \tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_n)$$

as follows. First pick $\rho_1, \rho_2, \rho_3, \rho_4 \in_R \mathbb{Z}_p$ and compute $C_1 = g_1^{\rho_1} g_2^{\rho_2}$, $C_2 = A g_2^{\rho_1}$, and $C_3 = g_1^{\rho_3} g_2^{\rho_4}$. Then pick $\tilde{C}_i \in_R \mathbb{G}$ for all $i \in [n] \backslash \mathcal{J}$, pick $r_i \in_R \mathbb{Z}_p$ for all $i \in \mathcal{J}$ and compute $\tilde{C}_i = (b_i^x / t_i)^{r_i}$ for all $i \in \mathcal{J}$.

2. Return Π_3 as $(\mathsf{aux}, \Pi_4, \Pi_5)$, where Π_4 and Π_5 are respectively signature proofs of knowledge of:

$$\mathit{SPK}_4 \left\{ \begin{pmatrix} e, x, y, \\ \rho_1, \rho_2, \rho_3, \rho_4, \\ \alpha_1, \alpha_2, \beta_3, \beta_4 \end{pmatrix} : \begin{array}{c} C_1 = g_1^{\rho_1} g_2^{\rho_2} \ \wedge \ 1 = C_1^{-e} g_1^{\alpha_1} g_2^{\alpha_2} \ \wedge \\ C_3 = g_1^{\rho_3} g_2^{\rho_4} \ \wedge \ 1 = C_3^{-x} g_1^{\beta_3} g_2^{\beta_4} \ \wedge \\ \frac{\hat{e}(C_2, w)}{\hat{e}_0} = \hat{e}(C_2, h_0)^{-e} \hat{e}_1^x \hat{e}_2^{y+\alpha_1} \hat{e}(g_2, w)^{\rho_1} \ \wedge \\ 1 = b^{\beta_3} t^{-\rho_3} \end{array} \right\} (\hat{m})$$

$$\mathit{SPK}_5 \left\{ (\mu_i, r_i)_{i \in [n]} : \bigvee_{\mathcal{I} \subseteq_{(n-d+1)} [n]} \bigwedge_{i \in \mathcal{I}} \left(1 = b^{\mu_i} t^{-r_i} \ \wedge \tilde{C}_i = b_i^{\mu_i} t_i^{-r_i} \right) \right\} (\hat{m})$$

on message $\hat{m} = \mathsf{aux} || m$. Π_4 can be computed using the knowledge of

$$(e, x, y, \rho_1, \rho_2, \rho_3, \rho_4, \alpha_1, \alpha_2, \beta_3, \beta_4),$$

where $\alpha_1 = \rho_1 e$, $\alpha_2 = \rho_2 e$, $\beta_3 = \rho_3 x$ and $\beta_4 = \rho_4 x$; Π_5 can be computed using the knowledge of $(\mu_i, r_i)_{i \in \mathcal{J}}$, where $\mu_i = r_i x$ for all $i \in \mathcal{J}$.

In the above, we denoted $\hat{e}(g_i, h_0)$ as $\hat{e}_i$ for $i = 0$ to 2.

6.1.2 Verification

To verify a proof $\Pi_3 = (\mathsf{aux}, \Pi_4, \Pi_5)$ for SPK_3 on message m where $\mathsf{aux} = (C_1, C_2, C_3, \tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_n)$, return `valid` if the verification of both Π_3 and Π_4 on $\hat{m} = \mathsf{aux} || m$ returns `valid`, and $\tilde{C}_i \neq 1$ for all $i = 1$ to n. Return `invalid` otherwise.

The instantiation of SPK_4 and SPK_5 is enumerated next.

6.2 SPK_4

6.2.1 Signing

To produce a proof Π_4 for SPK_4 on message $\hat{m} \in \{0,1\}^*$, do the following:

1. *(Commit.)* Pick $r_e, r_x, r_y, r_{\rho_1}, r_{\rho_2}, r_{\rho_3}, r_{\rho_4}, r_{\alpha_1}, r_{\alpha_2}, r_{\beta_3}, r_{\beta_4} \in_R \mathbb{Z}_p^*$ and compute
$$T_1 = g_1^{r_{\rho_1}} g_2^{r_{\rho_2}}, \quad T_2 = C_1^{-r_e} g_1^{r_{\alpha_1}} g_2^{r_{\alpha_2}},$$
$$T_3 = g_1^{r_{\rho_3}} g_2^{r_{\rho_4}}, \quad T_4 = C_3^{-r_x} g_1^{r_{\beta_3}} g_2^{r_{\beta_4}},$$
$$T_5 = \hat{e}(C_2, h_0)^{-r_e} \cdot \hat{e}_1^{r_x} \cdot \hat{e}_2^{r_y + r_{\alpha_1}} \cdot \hat{e}(g_2, w)^{r_{\rho_1}}, \quad T = b^{r_{\beta_3}} t^{-r_{\rho_3}}.$$

2. *(Challenge.)* Compute
$$c = H(T_1, \ldots, T_5, T, \hat{m}).$$

3. *(Respond.)* Compute
$$s_e = r_e - ce, \quad s_x = r_x - cx, \quad s_y = r_y - cy,$$
$$s_{\rho_i} = r_{\rho_i} - c\rho_i \ \text{ for } i = 1 \text{ to } 4,$$
$$s_{\alpha_i} = r_{\alpha_i} - c\rho_i e \ \text{ for } i = 1,2, \text{ and } s_{\beta_i} = r_{\beta_i} - c\rho_i x \ \text{ for } i = 3,4.$$

4. *(Output.)* The signature proof of knowledge Π_4 on $\hat{m}$ is
$$\Pi_4 = (c, s_e, s_x, s_y, s_{\rho_1}, s_{\rho_2}, s_{\rho_3}, s_{\rho_4}, s_{\alpha_1}, s_{\alpha_2}, s_{\beta_3}, s_{\beta_4}).$$

6.2.2 Verification

To verify a proof Π_4 for SPK_4 on message $\hat{m}$, do the following:

1. Compute
$$T_1' = g_1^{s_{\rho_1}} g_2^{s_{\rho_2}} C_1^c, \quad T_2' = C_1^{-s_e} g_1^{s_{\alpha_1}} g_2^{s_{\alpha_2}},$$
$$T_3' = g_1^{s_{\rho_3}} g_2^{s_{\rho_4}} C_3^c, \quad T_4' = C_3^{-s_x} g_1^{s_{\beta_3}} g_2^{s_{\beta_4}},$$
$$T_5' = \hat{e}(C_2, h_0)^{-s_e} \cdot \hat{e}_1^{s_x} \cdot \hat{e}_2^{s_y + s_{\alpha_1}} \cdot \hat{e}(g_2, w)^{s_{\rho_1}} \cdot \left(\frac{\hat{e}(C_2, w)}{\hat{e}_0}\right)^c, \quad T' = b^{s_{\beta_3}} t^{-s_{\rho_3}}.$$

2. Return `valid` if
$$c = H(T_1', \ldots, T_5', T', \hat{m}).$$

   Return `invalid` otherwise.

6.3 SPK_5

6.3.1 Signing

To produce a proof Π_5 for SPK_5 on message $\hat{m} \in \{0,1\}^*$, do the following:

1. *(Commit.)* Pick $r_{\mu_i}, r_{r_i} \in_R \mathbb{Z}_p$ for all $i \in \mathcal{J}$ and pick $c_i, s_{\mu_i}, s_{r_i} \in_R \mathbb{Z}_p^*$ for all $i \in [n] \backslash \mathcal{J}$. Then compute

$$T_i = \begin{cases} b^{r_{\mu_i}} t^{-r_{r_i}}, & i \in \mathcal{J}, \\ b^{s_{\mu_i}} t^{-s_{r_i}}, & i \in [n] \backslash \mathcal{J}, \end{cases} \quad \text{and} \quad \tilde{T}_i = \begin{cases} b_i^{r_{\mu_i}} t_i^{-r_{r_i}}, & i \in \mathcal{J}, \\ b_i^{s_{\mu_i}} t_i^{-s_{r_i}} \tilde{C}_i^{c_i}, & i \in [n] \backslash \mathcal{J}. \end{cases}$$

2. *(Challenge.)* Compute
$$c_0 = H((T_i, \tilde{T}_i)_{i=1}^n, \hat{m}).$$
Construct a polynomial f over $\mathbb{Z}_p^*$ of degree at most $(d-1)$ such that $c_i = f(i)$ for all $i \in \{0\} \cup [n] \backslash \mathcal{J}$. Compute $c_i = f(i)$ for all $i \in \mathcal{J}$.

3. *(Respond.)* Compute, for all $i \in \mathcal{J}$,
$$s_{\mu_i} = r_{\mu_i} - c_i \mu_i \quad \text{and} \quad s_{r_i} = r_{r_i} - c_i r_i.$$

4. *(Output.)* The signature proof of knowledge Π_4 on $\hat{m}$ is:
$$\Pi_4 = (f, (s_{\mu_i}, s_{r_i})_{i \in [n]}).$$

6.3.2 Verification

To verify a proof Π_5 for SPK_5 on message $\hat{m}$, do the following:

1. Compute, for all $i \in [n]$,
$$T_i' = b^{s_{\mu_i}} t^{-s_{r_i}} \quad \text{and} \quad \tilde{T}_i' = b_i^{s_{\mu_i}} t_i^{-s_{r_i}} \tilde{C}_i^{f(i)}.$$

2. Return `valid` if $deg(f) \leq d-1$ and
$$f(0) = H((T_i', \tilde{T}_i')_{i=1}^n, \hat{m}).$$
Return `invalid` otherwise.

6.4 Efficiency

Note that among the 5 pairings needed to compute T_5 above, 4 of them are constant and are assumed to be included in the system's parameters. The signer thus only needs to compute one pairing, namely $e(A_2, h_0)$. This pairing does not depend on the blacklist and the message and can thus be precomputed. Similarly, the SP needs to compute two pairings during verification, namely $e(A_2, h_0)$ and $e(A_2, w)$.

Table 1: Number of operations during an authentication with a blacklist of size n.

Operation	User		SP
	w/o Preproc.	w/ Preproc.	
$\mathbb{G}_1$ multi-EXP	7	0	4
$\mathbb{G}$ multi-EXP	$3n + 3$	$2n$	$2n + 4$
Pairing	1	0	2

7 Analysis

7.1 Complexities

We analyze the efficiency of our construction in terms of both time and space/communication complexities. First we emphasize that both complexities are independent of the number of users and SPs in the system. Thus our system scales well with respect to these two quantities. Both complexities, however, are dependent on the size of the blacklist. In particular, the communication overhead and the time it takes for both a user and an SP to execute the *Authentication* protocol grow linearly with the current size of the SP's blacklist.

More specifically, a blacklist of size n contains n tickets, each consisting of an ℓ-bit string and an element of $\mathbb{G}$. A proof Π_3 of SPK_3 consists of 3 $\mathbb{G}_1$ elements, n $\mathbb{G}$ elements and $3n + 12$ $\mathbb{Z}_p$ elements. The total communication complexity for an authentication is thus $n + 1$ ℓ-bit strings, 3 $\mathbb{G}_1$ elements, $(2n + 1)$ $\mathbb{G}$ elements and $3n + 12$ $\mathbb{Z}_p$ elements. SPs need to store a ticket for every successful authentication.

A breakdown of time complexity of the *Authentication* protocol into the number of pairing operations and *multi-exponentiations (multi-EXPs)*[11] in various groups is shown in Table 1. Operations such as $\mathbb{G}$ addition and hashing have been omitted as computing them takes relatively insignficant time. Some preprocessing is possible at the user before the knowledge of the challenge message and the blacklist. In fact, all but $2n$ multi-EXPs in $\mathbb{G}$ can be precomputed by the user.

7.2 Security

The correctness of our construction mostly stems from the correctness of the SPKs. Its proof is thus relatively straightforward. We claim that our construction has *correctness* without proof for the sake of conciseness.

We now state the following theorem about the *security* of our construction, and then sketch its proof.

Theorem 1 (Security) *Our construction of BLAC is secure if the q-SDH problem is hard in* $(\mathbb{G}_1, \mathbb{G}_2)$ *and the DDH problem is hard in* $\mathbb{G}$ *under the Random Oracle Model.*

7.2.1 Blacklistability

Suppose there exists a PPT adversary $\mathcal{A}$ who can win in game *Blacklistability* with non-negligible probability, we show how to construct a PPT simulator $\mathcal{S}$ that solves the q-SDH problem with non-negligible probability.

[11] A multi-EXP computes the product of exponentiations faster than performing the exponentiations separately. We assume that one multi-EXP operation multiplies up to 3 exponentiations.

On input of an instance of the q-SDH problem $(g_0', h_0', h_0'^{\gamma}, \dots, h_0'^{\gamma^q})$, $\mathcal{S}$'s task is to output a pair $(\bar{A}, \bar{e})$ such that $\hat{e}(\bar{A}, h_0'^{\gamma} h_0'^{\bar{e}}) = \hat{e}(g_0', h_0')$. We assume that $\mathcal{A}$ queries A-Reg or Corrupt-U at most q times. $\mathcal{S}$ uses the problem instance to generate the public parameters so that it can answer these queries. Specifically, $\mathcal{S}$ randomly generates a degree $(q-1)$ polynomial f such that $f(x) = \prod_{i=1}^{q-1}(x + e_i)$. It computes $h_0 = h_0'^{f(\gamma)}$ and $w = h_0^{\gamma} = h_0'^{\gamma f(\gamma)}$. It also computes $h_1 = [(w h_0^{e_*})^{\nu_1} h_0^{-1}]^{1/\nu_2}$ and $h_2 = h_1^{\mu}$ for some $e_*, \nu_1, \nu_2, \mu \in \mathbb{Z}_p^*$ generated uniformly at random. Next, it computes $g_i = \psi(h_i)$ for $i = 0$ to 2. Finally, $\mathcal{S}$ gives $(h_0, h_1, h_2, g_0, g_1, g_2, w)$ to $\mathcal{A}$ as the system parameters. Let $\mathcal{K}$ be the set $\{1, \dots, q-1\} \cup \{*\}$ and $\mathcal{K}'$ be an empty set $\emptyset$.

$\mathcal{S}$ keeps track of every user in the system. For a user $\iota \in \mathcal{I}_P$, $\mathcal{S}$ simulates the P-Reg oracle by first selecting $x_\iota \in \mathbb{Z}_p^*$ uniformly at random and then using it to simulate the *Registration* protocol. This is possible since the *Registration* protocol has Honest-Verifier Zero-Knowledgeness (HVZK). To simulate P-Auth and B-Auth for user $\iota \in \mathcal{I}_P$, $\mathcal{S}$ computes the tag as $t = b^{x_\iota}$ and simulates the other steps using the HVZK property of the *Authentication* protocol.

When user $\iota \in \mathcal{I}_P$ is to be corrupted, $\mathcal{S}$ chooses φ from set $\mathcal{K} \setminus \mathcal{K}'$ uniformly at random. If $\varphi = *$, $\mathcal{S}$ sets $y_\iota = (\nu_2 - x_\iota)/\mu$, $A_\iota = g^{\nu_1}$ and $e_\iota = e_*$, and returns $(A_\iota, e_\iota, x_\iota, y_\iota)$ as the credential of user ι. Otherwise, $\mathcal{S}$ chooses $y_\iota \in \mathbb{Z}_p^*$ uniformly at random, sets $e_\iota = e_\varphi$, computes A_ι as:

$$\begin{aligned} A_\iota = \left(g_0 g_1^{x_\iota + \mu y_\iota}\right)^{\frac{1}{e_\iota + \gamma}} &= g_0'^{\frac{f(\gamma)}{e_\iota+\gamma}} g_1^{\frac{x_\iota + \mu y_\iota}{\gamma + e_\varphi}} \\ &= g_0'^{\frac{f(\gamma)}{e_\iota+\gamma}} \left(g_0^{\frac{(x_\iota+\mu y_\iota)\nu_1 (e_* + \gamma) - (x_\iota + \mu y_\iota)}{(e_\iota+\gamma)\nu_2}} \right) \\ &= g_0'^{\frac{f(\gamma)}{e_\iota+\gamma}\left(1 - \frac{x_\iota+\mu y_\iota}{\nu_2}\right)} \left(g_0^{\frac{(x_\iota+\mu y_\iota)\nu_1}{\nu_2}} \right)^{\left(1 - \frac{e_\iota - e_*}{e_\iota+\gamma}\right)} \\ &= g_0'^{\frac{f(\gamma)}{e_\iota+\gamma}\left(1 - \frac{x_\iota+\mu y_\iota}{\nu_2} - \frac{(e_\iota - e_*)(x_\iota+\mu y_\iota)\nu_1}{\nu_2}\right)} g_0^{\frac{(x_\iota+\mu y_\iota)\nu_1}{\nu_2}}, \end{aligned}$$

and finally returns $(A_\iota, e_\iota, x_\iota, y_\iota)$ as the credential of user ι. In both cases, $\mathcal{S}$ adds φ to $\mathcal{K}'$.

The simulation of A-Reg is similar. Upon receiving C for user ι (to be added to $\mathcal{I}_A$), $\mathcal{S}$ first extracts the pair (x_ι, y_ι') by rewinding the adversary and selects φ from $\mathcal{K} \setminus \mathcal{K}'$ uniformly at random. If $\varphi = *$, $\mathcal{S}$ chooses y_ι'' such that $x_\iota + \mu(y_\iota' + y_\iota'') = \nu_2$, sets $A_\iota = g^{\nu_1}$, $e_\iota = e_*$ and finally returns $(A_\iota, e_\iota, y_\iota'')$. Otherwise, $\mathcal{S}$ chooses y_ι'' uniformly at random, sets $e_\iota = e_\varphi$ and $y_\iota = y_\iota' + y_\iota''$, computes A_ι as:

$$A_\iota = g_0'^{\frac{f(\gamma)}{e_\iota+\gamma}\left(1 - \frac{x_\iota+\mu y_\iota}{\nu_2} - \frac{(e_\iota - e_*)(x_\iota+\mu y_\iota)\nu_1}{\nu_2}\right)} g_0^{\frac{(x_\iota+\mu y_\iota)\nu_1}{\nu_2}},$$

and finally returns $(A_\iota, e_\iota, y_\iota'')$. $\mathcal{S}$ adds φ to $\mathcal{K}'$ in both cases.

$\mathcal{S}$ stores all the credentials issued to $\mathcal{A}$. For each SP j, $\mathcal{S}$ maintains a table $\mathcal{T}_j$ of q rows. The rows are indexed by elements of $\mathcal{K}$ and contains no elements initially.

For each $k :=$ A-Auth(j, d) query, $\mathcal{S}$ tests if the associated ticket (s, t) satisfies $t = H_0(s||SP_j)^{x_\varphi}$ for all $\varphi \in \mathcal{K}$. If such i exists, it appends $\varrho := (k, d)$ to row φ. Otherwise, it rewinds and extracts the underlying credential $c' := (A', e', x', y')$. We call c' a special tuple. For each Remove-From-BL(j, τ) query, $\mathcal{S}$ removes the corresponding element from the table $\mathcal{T}_j$.

If a special tuple c' exists, $\mathcal{S}$ solves the q-SDH problem as follows.

- Case I: $e' \neq e_\varphi$ for all $\varphi \in \mathcal{K}$.

Denote $z = x' + \mu y'$. We have:

$$\begin{aligned}
A'^{e'+\gamma} &= g_0 g_1^z \\
A'^{e'+\gamma} &= g_0^{\frac{\nu_1 z(e_*+\gamma)-z}{\nu_2}} \\
A' &= g_0^{\frac{\nu_2 - z}{\nu_2(e'+\gamma)}} \left(g_0^{\frac{\nu_1 z}{\nu_2}} \right)^{\left(1 - \frac{e'-e_*}{e'+\gamma}\right)} \\
g_0^{\frac{1}{e'+\gamma}} &= \left(A' g_0^{\frac{-\nu_1 z}{nu_2}} \right)^{\frac{\nu_2}{\nu_2 - z - \nu_1 z(e'-e_*)}} .
\end{aligned}$$

Denote $B' = g_0{}^{\frac{1}{e'+\gamma}} = g_0'{}^{\frac{f(\gamma)}{e'+\gamma}}$. Using long division, there exists a degree $(q-2)$ polynomial f_q such that $\frac{f(\gamma)}{(e'+\gamma)} = f_q(\gamma)(e'+\gamma) + f_1$ for some $f_1 \in \mathbb{Z}_p^* \backslash \{0\}$. Thus $B' = g_0'{}^{\frac{f_1}{e'+\gamma} + f_q(\gamma)}$. Finally, $\mathcal{A}$ computes $\bar{A} = \left(B' g_0'{}^{-f_q(\gamma)} \right)^{1/f_1}$ and sets $\bar{e} = e'$. $(\bar{A}, \bar{e})$ is a solution to the q-SDH problem.

- Case II: $(e' = e_\iota \wedge A' = A_\iota)$ for some $\iota \in \mathcal{I}_P \cup \mathcal{I}_A$.
  This case happens with negligible probability unless $\mathcal{A}$ can solve the discrete logarithm of h_2 to base h_1.

- Case III: $e' = e_\varphi$ for some $\varphi \in \mathcal{K}$.
  If $e' \neq e_*$, $\mathcal{S}$ aborts. Otherwise denote $z = x' + \mu y'$. We have:

$$\begin{aligned}
A'^{e_*+\gamma} &= g_0 g_1^z \\
A' &= g_0^{\frac{\nu_2 - z}{\nu_2(e_*+\gamma)}} g_0^{\frac{\nu_1 z}{\nu_2}} \\
g_0^{\frac{1}{e_*+\gamma}} &= \left(A' g_0^{\frac{-\nu_1 z}{\nu_2}} \right)^{\frac{\nu_2}{\nu_2 - z}} .
\end{aligned}$$

  Denote $B' = g_0{}^{\frac{1}{e_*+\gamma}} = g_0'{}^{\frac{f(\gamma)}{e'+\gamma}}$. $\mathcal{S}$ uses the same method as in Case I to solve the q-SDH problem.

It remains to argue that, if $\mathcal{A}$ can win the game, a special tuple exists with non-negligible probability.

Assume there is no special tuple and $|\mathcal{A}| \neq 0$. $\mathcal{A}$ wins the game if there exists a sequence of A-Auth oracle query (A-Auth$(j, d_1), \ldots,$ A-Auth(j, d_m)) such that the sequence $(d_1, \ldots, d_m)$ is not *q-partitionable*. We can assume m is equal to the number of A-Auth$(j, \cdot)$ query, for if the sequence formed by the whole series of A-Auth$(j, \cdot)$ query is q-partitionable, any sub-sequence of it must be q-partitionable. Let the sequence be Q^*

The set of the sequences formed by the second element of each row of table $\mathcal{T}_j$ is a q-partition of Q^*. Denote this particular q-partition as Q'. Since Q^* is not q-partitionable, it must be the case that Q' is not bounded. It implies that there exists an ι and d such that the table entry d^* in the ι-th row and the d-th column is less than d, i.e., $d^* < d$. Let the corresponding query be $k :=$ A-Auth(j, d^*), during which $\mathcal{A}$ has constructed a valid proof of knowledge of SPK Π_3 using a witness with x_ι in it.

Now, on the blacklist used in $k :=$ A-Auth(j, d^*), there are $(d-1)$ tickets generated using x_ι. (Otherwise, the adversary would have already violated the soundness of Π_3 in at least one of the authentications associated with those $(d-1)$ tickets.) The soundness of Π_3 thus implies that, with non-negligible probability, $d < d^*$, which contradicts to $d^* < d$ above.

Otherwise, $|\mathcal{I}_A| = 0$. In that case, there must be a special tuple as $\mathcal{K}$ is empty. Moreover, case I above happens with overhelming probability since the adversary does not have any information about the e_i's.

7.2.2 Anonymity

Suppose there exists a PPT adversary $\mathcal{A}$ who can win in game Anonymity with non-negligible probability (say, $\frac{1}{2}+\epsilon$), we show how to construct a PPT simulator $\mathcal{S}$ that solves the DDH problem with non-negligible probability. On input a DDH problem instance $(g', g'^{u'}, g'^{v'}, T')$, $\mathcal{S}$ is required to decide if $T' = g'^{u'v'}$. $\mathcal{S}$ sets $G = \langle g' \rangle$ and generates all other parameters honestly. The parameters and the master key of the GM are given to $\mathcal{A}$.

$\mathcal{S}$ keeps track of every user in the system. $\mathcal{S}$ chooses one user, denoted as user i^*. For all oracle queries (except the Hash oracle) not related to user i^*, $\mathcal{S}$ follows the protocol honestly.

Queries related to user i^* are handled as follows. For B-Reg, $\mathcal{S}$ simulates the protocol as if (u', y') is an opening of the commitment C. The distribution is perfect since for any u' there exists an y' such that $C = g_1^{u'} g_2^{y'}$. Upon receiving (A, e, y'') from $\mathcal{A}$, $\mathcal{S}$ records the credential for i^* as $(A, e, \perp, \perp)$. The credential for user i^* is (A, e, u', y) such that $y = y' + y''$. This credential, however, is unknown to $\mathcal{S}$. For P-Auth or B-Auth queries, $\mathcal{S}$ chooses s uniformly at random and sets $H_0(s||ID_j) = g'^R$, where ID_j is the ID of SP j, for some R generated uniformly at random. $\mathcal{S}$ then computes $t = g'^{u'R}$ and simulates the protocols with $\tau = (s, t)$.

In the challenge phase, $\mathcal{A}$ outputs two users i_0^* and i_1^* from $\mathcal{I}_B$. If $i^* \notin \{i_0^*, i_1^*\}$, $\mathcal{S}$ aborts. Else, $\mathcal{A}$ queries P-Auth$(\perp, j^*, d^*)$ if $j^* \in \mathcal{J}_P$, or B-Auth$(\perp, j^*, d^*)$ otherwise.

Now there are two cases to consider. In the case when d_0^* and hence d_1^* are greater than or equal to d^*, $\mathcal{S}$ simply return $\perp$ as the protocol transcript. It is straight-forward to see that the corresponding B-Auth or P-Auth query does not contains any information on i_0^* or i^* and probability of $\mathcal{A}$ winning cannot be greater than $\frac{1}{2}$.

In the case when d_0^* and d_1^* are both less than d^*, $\mathcal{S}$ flips a fair coin $\hat{b}$. If $i^* \neq i_{\hat{b}}^*$, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ chooses s_{i^*} uniformly at random and sets $H_0(s_{i^*}||SP_j) = g'^{v'}$. $\mathcal{S}$ computes the ticket $\tau = (H_0(s_{i^*}||SP_j), t_{i^*}) = (g'^{v'}, T')$ and simulates the corresponding *Authentication* protocol.

If T' is a DDH tuple, the simulation is perfect and $\mathcal{A}$ wins with probability $\frac{1}{2}+\epsilon$. On the other hand, if T' is a random element, the simulation is imperfect in the sense that the authentication transcript is not related to either of the challenge users. In that case probability of $\mathcal{A}$ winning cannot be greater than $\frac{1}{2}$.

Finally, if $\mathcal{A}$ guessed $\hat{b}$ correctly, $\mathcal{S}$ answers that $(g', g'^{u'}, g'^{v'}, T')$ is a DDH-tuple.

Now with probability $\frac{2}{|\mathcal{I}_B|}$, $i^* \in \{i_0^*, i_1^*\}$. With probability $\frac{1}{2}$, $i^* = i_{\hat{b}}^*$. Thus, the probability of not aborting is $\frac{1}{|\mathcal{I}_B|}$, which is non-negligible. We assume $\mathcal{S}$ answers "no" when it has to abort, in which case the probability of $\mathcal{S}$ winning is $\frac{1}{2}$.

If $\mathcal{S}$ does not abort and T' is a DDH tuple, $\mathcal{A}$ wins with probability $\frac{1}{2}+\epsilon$. If T' is a random element, $\mathcal{A}$ can only output the guess bit correctly with probability no more than $\frac{1}{2}$ since the transcript of the challenge authentication does not contain any information on $i_{\hat{b}}^*$. In fact, $\mathcal{A}$ could abort or behave randomly and for simplicity we let its winning probability be ϵ' such that $0 \leq \epsilon' \leq \frac{1}{2}$.

To summarize, there are 4 cases.

1. T' is a DDH tuple, $\mathcal{S}$ answers "yes". The corresponding probability is $\frac{1}{2}+\epsilon$.

2. T' is a DDH tuple, $\mathcal{S}$ answers "no". The corresponding probability is $\frac{1}{2} - \epsilon$.

3. T' is *not* a DDH tuple, $\mathcal{S}$ answers "yes". The corresponding probability is ϵ' for some ϵ' such that $0 \leq \epsilon' \leq \frac{1}{2}$.

4. T' is *not* a DDH tuple, $\mathcal{S}$ answers "no". The corresponding probability is $1 - \epsilon'$ for some ϵ' such that $0 \leq \epsilon' \leq \frac{1}{2}$.

The probability that $\mathcal{S}$ answers correctly (case 1 + case 4) is therefore

$$\frac{1}{2}(\frac{1}{2} + \epsilon + 1 - \epsilon') = \frac{1}{2} + \frac{\epsilon}{2} + (\frac{1}{2} - \epsilon'),$$

which is no less than $\frac{1}{2} + \frac{\epsilon}{2}$. Summing up the cases of aborting and not aborting, the probability of $\mathcal{S}$ winning is at least $\frac{1}{2} + \frac{\epsilon}{2|\mathcal{I}_B|}$.

7.2.3 Non-Frameability

Suppose there exists a PPT adversary $\mathcal{A}$ who can win in game *Non-Frameability* with non-negligible probability, we show how to construct a PPT simulator $\mathcal{S}$ that solves the discrete logarithm problem in $\mathbb{G}$.

On input of a DL problem instance (T', g'), $\mathcal{S}$ is required to compute u' such that $g'^{u'} = T'$. $\mathcal{S}$ sets $\mathbb{G} = \langle g' \rangle$ and all other parameters are generated honestly. The parameters and the master key of GM are given to $\mathcal{A}$.

$\mathcal{S}$ keeps track of every user present in the system. $\mathcal{S}$ chooses one user, denoted as user $\hat{i}$. For all oracle queries (except Hash oracle) not related to user $\hat{i}$, $\mathcal{S}$ follows the protocol honestly. Let $\mathcal{K}$ be the set of credentials $\mathcal{S}$ has obtained from $\mathcal{A}$ in the B-Reg query.

Queries related to user $\hat{i}$ are handled as follows. For B-Reg, $\mathcal{S}$ simulates the protocol as if u', y' is an opening of the commitment C. The distribution is perfect since for any u' there exists a y' such that $C = g_1^{u'} g_2^{y'}$. Upon receiving (A, e, y'') from $\mathcal{A}$, $\mathcal{S}$ adds $(A, e, \perp, \perp)$ to $\mathcal{K}$. Note that the credential for user $\hat{i}$ is (A, e, u', y) such that $y = y' + y''$ and is unknown to $\mathcal{S}$. For P-Auth or B-Auth, $\mathcal{S}$ chooses s and R uniformly at random and sets $H_0(s||ID_{j^*}) = g'^R$, where ID_{j^*} is the ID of SP j^*. $\mathcal{S}$ then computes $t = g'^{u'R}$ and simulates the protocols with $\tau = (s, t)$.

Finally, $\mathcal{S}$ aborts if $\hat{i} \neq i^*$. With probability $\frac{1}{|\mathcal{I}_B|}$, $\mathcal{S}$ does not abort. Since $d^* > d_i^*$ (refer to Section 4.3.3 for their meaning), the fact that the challenge P-Auth query terminated unsuccessfully implies that, with overwhelming probability, there exists an Add-To-BL(j^*, k') query for some k' such that k' is the output of an A-Auth query. (Observe that k' cannot be an output of P-Auth$(i^*, \cdot, \cdot)$ or B-Auth$(i^*, \cdot, \cdot)$.) Denote by $\varpi_{k'}$ the transcript of the A-Auth query. Assume $(s', t') = \mathit{Extract}(\varpi'_k)$, $\mathcal{S}$ rewinds the SPK in the A-Auth query and obtains $u' = \log_{H_0(s'||\cdot)}(t')$. It returns u' as the solution of the DL problem.

Informally speaking, the above means that, to prevent an honest user from authenticating himself successfully, the adversary must have conduct some kind of $k' :=$ A-Auth query such that the associate ticket is equal to $H_0(\cdot||\cdot)^x$ so that x is the secret of the target user i^*. Then the adversary queries Add-To-BL(j^*, k'), making the corresponding P-Auth to terminate unsuccessfully. $\mathcal{S}$, by rewinding the the A-Auth query k', thus gets to know the DL of u', which is x.

8 Performance Evaluation

We now present results from the experimental evaluation of our BLAC construction.

8.1 Prototype Implementation

We implemented our construction of BLAC in C and packaged the code into a software library to allow for easy adoption by application developers. We have not implemented the extension for supporting a d-strikes-out revocation policy and our current implementation therefore supports only a 1-strike-out revocation policy. We used the Pairing-Based Cryptography (PBC) Library[12] (version 0.4.11) for the underlying elliptic-curve and pairing operations, which is built on the GNU MP Bignum (GMP) Library.[13] We also made use of several routines in OpenSSL,[14] such as its SHA-1 hash function for instantiating the cryptographic hash functions needed by our construction.

The choice of curve parameters can have a significant effect on the performance of an implementation. We used pairings over Type-A curves as defined in the PBC library. A curve of this type has the form of $E : y^2 = x^3 + x$ over the field $\mathbb{F}_q$ for some prime q. Both $\mathbb{G}_1$ and $\mathbb{G}_2$ are the group of points $E(\mathbb{F}_q)$ of order p for some prime p such that p is a factor of $q + 1$. The pairing is symmetric and has an embedding degree k of 2. Thus $\mathbb{G}$ is a subgroup of $\mathbb{F}_{q^2}$. In our implementation, q and p are respectively 512-bit and 160-bit integers. We also used $\mathbb{G}$ for the group wherein the tickets reside.

The interface to the library we implemented is defined by a list of C functions. Some of the more important functions are as follows. `setup()` is a function that implements the *Setup* algorithm. The functions `register_gm()` and `register_user()`, executed by the GM and the user respectively, together implement the *Registration* protocol. Similarly `authen_sp()` and `authen_user()` together implement the *Authentication* protocol.

To test and evaluate our library implementation, we wrote a driver application that allows users to post text messages at a web forum. This can be thought of as users editing Wikipedia pages. We did not prototype the user registration part of the system because our major interest was to study the performance of the *Authentication* protocol.

In the application, authentication is carried out as follows. The SP first creates a listening socket. Upon the arrival of a connection request from a user, the SP sets up an SSL socket with the user using OpenSSL.[15] This means that a confidential and server-authenticated channel is set up between the user and the SP. From within this channel, the user and the server respectively execute `authen_user()` and `authen_sp()`. If `authen_sp` returns `failure`, the SP closes the SSL connection, thereby refusing to serve the user. Otherwise, SP serves the user using the same channel by recording the text message sent by the user, along with the ticket extracted from the authentication transcript. The SP may then manually inspect the text message and add the associated ticket to its blacklist.

Alternatively, by integrating it with SSL server-authentication, BLAC authentication can be turned into a mutual authentication, in which the user authenticates the server's identity but the server is ensured that and only that the user is some well-behaving user.

8.2 Experimental Results and Analysis

For our experiments, we used a Dell OptiPlex 745 desktop machine with an Intel dual-core (Core 2) 1.86GHz CPU and 1GB of RAM, running Linux/Ubuntu 7.10. All the timings reported below were averaged over 10 randomized runs.

We measured two time quantities related to the execution of the *Authentication* protocol: (1) the time it took for an SP to verify the authentication (i.e., step 4 of the protocol), and (2) the time

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

[13] `http://gmplib.org/`

[14] `http://www.openssl.org/`

[15] For the sake of simplicity, the SP uses a self-signed key-pair to authenticate itself.

it took for a user to inspect the blacklist and produce a proof (i.e., steps 2 and 3 of the protocol), with preprocessing enabled. The sum of these two quantities roughly represents the total latency incurred by the protocol as perceived by the user if we ignore the network I/O delay, which is network-dependent.

When the blacklist was empty, it took the SP 0.06s to verify the authentication. When the blacklist had 400 entries instead, it took the SP 0.46s to do the same. On the other hand, when the blacklist size was 0 and 400, the user spent 0.09ms and 0.73s respectively to inspect the blacklist and produce a proof. The estimated protocol latencies are thus 0.06s and 1.19s respectively. The total communication overhead due to the authentication protocol is roughly 0.27KB per blacklist entry. Figures 1(a) and 1(b) show experimental data collected with different blacklist sizes. Please see our discussion in Section 9 that elaborates on the feasibility of our construction in real applications.

Note that our authentication protocol scales well with the number of cores in CPUs because virtually all computation that grows linearly with the blacklist size is parallelizable.[16] As evidence, on our dual-core machine, all the timings we collected using a single-threaded implementation almost doubled the figures of our current multi-threaded implementation, whose figures are reported above.

9 Discussion

We now discuss various practical issues related to the deployment of BLAC in a real-world setting.

9.1 Efficiency

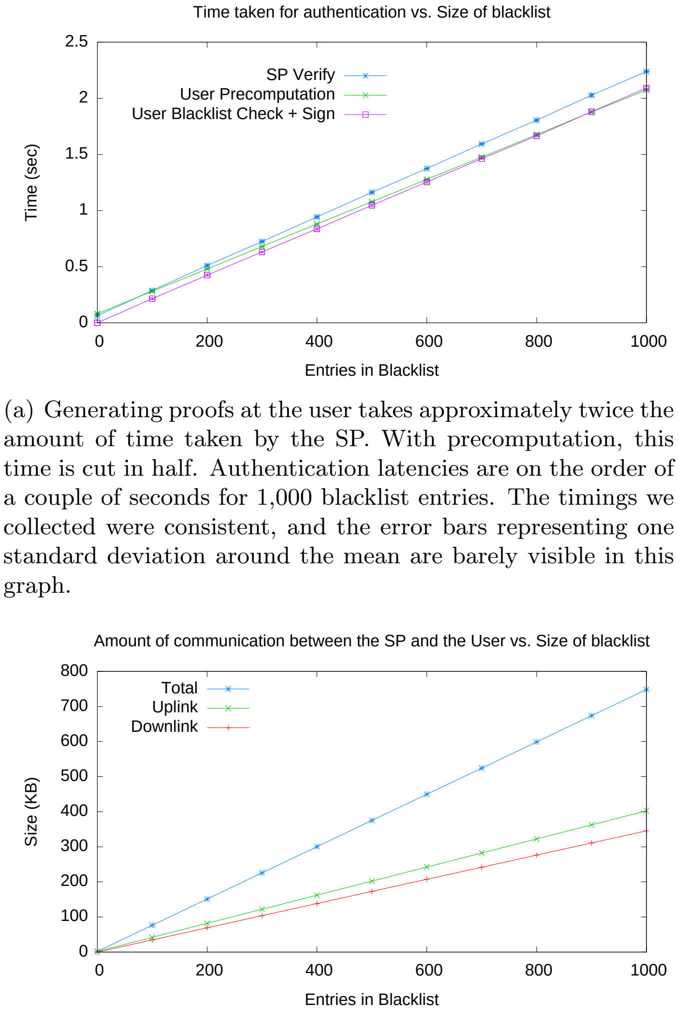
In our cryptographic construction, blacklist verification requires $O(n)$ computations, where n is the number of entries in the blacklist. As indicated by Section 8, our scheme would support 1,600 blacklist entries with 2 authentications per second on an 8-core machine.[17] Since anonymous authentication will be used at SPs such as Wikipedia only for certain operations such as editing webpages, we believe this performance is reasonable. Consider two extreme examples. In March 2007, Wikipedia averaged about two edits per second to its set of English webpages.[18] Likewise, YouTube reported less than one video upload per second on average in July 2006.[19] The communication complexity required to sustain one or two authentications per second with 1,600 blacklist entries would be about 3.5 to 7 Mbps for the SP. Such a data rate would be high for an individual server, but would be reasonable for large SPs such as YouTube and Wikipedia, which may have distributed servers across the nation for handling large bandwidth. Based on these calculations, SPs with much lower authentication rates than Wikipedia or YouTube (e.g., one authentication every few seconds) can easily be served on commodity hardware and T-1 lines. We reiterate that our construction is the first to allow anonymous blacklisting without TTPs, and more efficient blacklist checking, perhaps in $O(\log n)$ or $O(1)$ time, is an open problem that deserves further research. Faster verification will allow much higher rates of authentication while supporting extremely large blacklists, and this problem is, therefore, worthy of further study. As mentioned earlier, our follow-up work on PEREA [32] alters the semantics of revocation to provide more efficient authentication. PEREA introduces the concept of a *revocation window*, the number of subsequent authentications before which a misbehavior must be recognized and blacklisted for the user to be revoked. These

[16] The only exception is the two calls to SHA-1, but they take comparably negligible time.

[17] An 8-core Mac Pro with two 2.8GHz Quad-Core Intel Xeon processors was available for approximately $2,800 at the time of writing.

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

[19] `http://technology.guardian.co.uk/weekly/story/0,,1823959,00.html`



(a) Generating proofs at the user takes approximately twice the amount of time taken by the SP. With precomputation, this time is cut in half. Authentication latencies are on the order of a couple of seconds for 1,000 blacklist entries. The timings we collected were consistent, and the error bars representing one standard deviation around the mean are barely visible in this graph.

(b) Communication is on the order of 300–400 KB for 1,000 blacklist entries and is approximately the same for uplink and downlink communication.

Figure 1: The communication and execution times scale linearly with the size of the blacklist.

semantics allow for more efficient authentication at the server, but allows for the possibility of blacklisted users to remain unrevoked (if misbehaviors are not recognized within the revocation window).

9.2 Interleaving Authentications

One concern is that an individual user may attempt to interleave multiple authentications and take up several hundreds of entries in the blacklist by misbehaving several times in a short span of time. Such an attack is possible because users can parallelize several anonymous sessions with an SP. A promising approach would be to use a scheme such as Camenisch et al.'s citeCamenischEtAl06ccs periodic n-times anonymous authentication to rate-limit the number of anonymous accesses from users. In such a scheme, an anonymous user would be able to access the SP anonymously at most n times within a time period. For example, for $n = 10$ and a time period of 1 day, a single user would be able to contribute at most 10 entries to the blacklist in a given day.

Remark. Since concurrent sessions are preempted while an entry is added (atomically) to a blacklist, our system guarantees that once an entry is added to the blacklist at time t, the blacklisted user will not be able to access the service after time t (or until unblacklisted at a later time).

9.3 Enrollment Issues

We assume that the Group Manager issues only one credential per legitimate user and assume it is difficult to perform "Sybil" attacks [18], where users are able to obtain multiple credentials by posing as different identities. The Sybil attack, however, is a challenging problem that any credential system is vulnerable to, and we do not attempt to solve this problem here.

In a real deployment of BLAC, users may eventually misplace their credentials, or have them compromised. Since that credential may be blacklisted by an SP, issuing a new credential to a user can help that user circumvent blacklisting. As a trade-off, we suggest that if a user misplaces his or her credential, that user is issued a pseudonymous credential for a certain amount of time called the "linkability window" before a new anonymous credential is issued. If a user repeatedly attempts to acquire new credentials, the linkability window of that user can be increased to curb misbehavior.

9.4 Allowing the Sharing of (Entries in) Blacklists

We have presented our construction of BLAC in which an SP cannot use an entry from another SP's blacklist (corresponding to Alice) to prevent Alice from successfully authenticating to the SP. Nevertheless, in some applications, a group of SPs may desire to block users misbehaving at any one of the SPs.

Our system can be modified to allow such sharing: instead of computing the tag in a ticket as $t = H(s||\mathsf{SP})^x$, a user computes it as $t = H(s)^x$ regardless of the SP the user is connecting to. Tickets computed as such can be shared among SPs as adding a user's ticket borrowed from another SP is no different from the SP obtaining a ticket directly from the same user. Such a modified construction, however, has different security (and privacy) implications. For instance, Wikipedia may decide to add only YouTube's tickets to its blacklist. If a user's authentication fails, Wikipedia knows that the user has previously visited YouTube. Even though the user is anonymous, an SP can learn some information about the user's behavior at another SP.

9.5 Revoking Compromised TPMs

Concurrently and independently, Brickell and Li citeepid have proposed a method to unlinkably revoke compromised Trusted Platform Modules (TPMs) [29]. While they focus on revoking compromised hardware, as opposed to blacklisting misbehaving users, their construction is similar to ours. Both solutions use a protocol for proving the inequality of multiple discrete logarithms to prove that a user is not revoked/blacklisted. Nevertheless, signatures in their solution are not bound to the verifier's identity and authenticating even once could result in the global revocation of the prover. Our solution provides more privacy by allowing sharing and non-sharing of blacklist entries among verifiers. Their solution does not support a d-strikes-out revocation policy, and finally, their solution is RSA-based while ours is pairing-based.

10 Conclusions

We present BLAC, an credential system for anonymous authentication that for the first time simultaneously provides *privacy-enhanced revocation*, *subjective judging*, and *eliminates the reliance on trusted third parties* capable of revoking the privacy of users. We believe that the ability to revoke users while maintaining their anonymity is a worthwhile endeavor. While BLAC demonstrates the feasibility of such a goal, we encourage researchers to develop solutions that are more efficient — BLAC requires computation at the SP that is linear in the size of the blacklist. We make one such attempt with PEREA [32], albeit with different revocation semantics. We also believe that our contributions of supporting a d-strikes-out revocation policy is a novel analog to threshold-based approaches such as k-TAA. Future work could explore more complex policies such as boolean combinations of misbehaviors (such as "user has defaced a webpage AND user has posted copyrighted material more than 3 times").

11 Acknowledgments

The authors would like to thank Jiangtao Li and the anonymous CCS'07 reviewers for their valuable comments.

References

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO*, volume 1880 of *LNCS*, 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 *LNCS*, pages 183–197. Springer, 2002.

[3] M. H. Au, S. S. M. Chow, and W. Susilo. Short e-cash. In *INDOCRYPT*, volume 3797 of *LNCS*, pages 332–346. Springer, 2005.

[4] M. H. Au, W. Susilo, and Y. Mu. Constant-size dynamic k-TAA. In *SCN*, volume 4116 of *LNCS*, pages 111–125. Springer, 2006.

[5] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM Press, 1993.

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

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

[8] 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.

[9] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *EUROCRYPT*, volume 3494 of *LNCS*, pages 302–321. Springer, 2005.

[10] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Balancing accountability and privacy using e-cash (extended abstract). In *SCN*, volume 4116 of *LNCS*, pages 141–155. Springer, 2006.

[11] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, volume 2045 of *LNCS*, pages 93–118. Springer, 2001.

[12] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, volume 2442 of *LNCS*, pages 61–76. Springer, 2002.

[13] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN*, volume 2576 of *LNCS*, pages 268–289. Springer, 2002.

[14] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO*, volume 3152 of *LNCS*, pages 56–72. Springer, 2004.

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

[16] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT*, pages 418–430, 2000.

[17] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, pages 303–320. USENIX, 2004.

[18] J. R. Douceur. The sybil attack. In *IPTPS*, volume 2429 of *LNCS*, pages 251–260. Springer, 2002.

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

[20] 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.

[21] 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.

[22] A. Kiayias and M. Yung. Group signatures with efficient concurrent join. In *EUROCRYPT*, volume 3494 of *LNCS*, pages 198–214. Springer, 2005.

[23] J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *ACISP*, volume 3108 of *LNCS*, pages 325–335. Springer, 2004.

[24] L. Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA*, volume 3376 of *LNCS*, pages 275–292. Springer, 2005.

[25] L. Nguyen and R. Safavi-Naini. Dynamic k-times anonymous authentication. In *ACNS*, volume 3531 of *LNCS*, pages 318–333. Springer, 2005.

[26] C.-P. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[27] I. Teranishi, J. Furukawa, and K. Sako. k-times anonymous authentication (extended abstract). In *ASIACRYPT*, volume 3329 of *LNCS*, pages 308–322. Springer, 2004.

[28] I. Teranishi and K. Sako. k-times anonymous authentication with a constant proving cost. In *Public Key Cryptography*, volume 3958 of *LNCS*, pages 525–542. Springer, 2006.

[29] TPM Work Group. TCG TPM specification version 1.2 revision 94. Technical report, Trusted Computing Group, 2006.

[30] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. Blacklistable anonymous credentials: Blocking misbehaving users without TTPs. In *ACM Conference on Computer and Communications Security*. ACM, 2007. To Appear.

[31] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. Blacklistable anonymous credentials: Blocking misbehaving users without TTPs (full version). Technical Report TR2007-601, Dartmouth College, Aug 2007.

[32] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. PEREA: Towards practical TTP-free revocation in anonymous authentication. In *CCS '08: 15th ACM conference on Computer and communications security (To Appear)*. ACM, 2008.

[33] P. P. Tsang and V. K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *ISPEC*, volume 3439 of *LNCS*, pages 48–60. Springer, 2005.

[34] P. P. Tsang, V. K. Wei, T. K. Chan, M. H. Au, J. K. Liu, and D. S. Wong. Separable linkable threshold ring signatures. In *INDOCRYPT*, volume 3348 of *LNCS*, pages 384–398. Springer, 2004.