Thinking Inside the BLAC Box:*

Smarter Protocols for Faster Anonymous Blacklisting

Ryan Henry
Cheriton School of Computer Science
University of Waterloo
Waterloo ON, Canada N2L 3G1
rhenry@cs.uwaterloo.ca

Ian Goldberg
Cheriton School of Computer Science
University of Waterloo
Waterloo ON, Canada N2L 3G1
iang@cs.uwaterloo.ca

ABSTRACT

We present BLACRONYM, a suite of new communication- and computation-efficient protocols for anonymous blacklisting without trusted third parties. Our protocols improve on Tsang et al.'s Blacklistable Anonymous Credentials (BLAC) system and its variants by incorporating novel batch zero-knowledge proof and verification techniques. BLACRONYM provides comparable functionality and security guarantees to those of BLAC and its derivatives, but it is substantially faster and consumes much less bandwidth. At the heart of BLACRONYM is the first batch zero-knowledge protocol in the literature for proofs of partial knowledge over non-monotone access structures; we suspect that our new techniques will find applications in speeding up other cryptographic constructions that require proofs of similar statements.

Categories and Subject Descriptors

K.4.1 [**Public Policy Issues**]: Privacy; D.4.6 [**Security and Protection**]: Authentication, cryptographic controls, verification; K.4.2 [**Social Issues**]: Abuse and crime involving computers

General Terms

Algorithms, Security, Performance.

Keywords

Anonymous blacklisting; batch zero-knowledge proofs; efficiency.

1. INTRODUCTION

The Internet can be a dangerous place to visit. As the proportion of our daily activities that occur online continues to increase, so too does our exposure to online privacy risks imposed on us by fraudsters and identity thieves, by intrusive advertising companies, by oppressive governments, and by countless unknown others. Anonymous communications systems like Tor[1] mitigate some of these threats by helping users to access services over the public Internet while concealing their identities and usage patterns from prying eyes. A global user base leverages the anonymity afforded by Tor and its counterparts to circumvent online censorship, to research taboo and unpopular subjects, and to speak their minds without fear of retaliation. Not only is this a win for privacy and free speech online, but it is also a potential boon for many online communities that might benefit from added diversity in their respective user populations. Compelling examples of such online communities include collaborative encyclopedias like Wikipedia[2] and community-driven review sites like Yelp[3].

Yet reality is rarely so simple. The providers of such online services must ultimately weigh the expected benefits (both to themselves and to their user communities) of more inclusivity against the risks posed by abusive users, especially those who would hide behind the veil of anonymity to skirt accountability for their actions. A number of popular services—notably including Wikipedia, Yelp, Slashdot[4], Craigslist[5], and most major IRC networks [32]—presently block contributions from anonymous users, despite the implied loss of diversity and the broader implications for free speech and the open exchange of knowledge and ideas.

In response, the cryptographic and privacy research communities have proposed several *anonymous blacklisting* designs, which seek to provide mechanisms through which service providers (SPs) may hold anonymous users accountable for their individual actions *without threatening their anonymity*. SPs can thereby protect their user communities from abuse by the occasional "naughty" anonymous user without inflicting collateral damage on all the "nice" users. An early proposal called Nymble [23] solved the anonymous blacklisting problem both elegantly and efficiently; however, Nymble and its progeny [21, 26, 27] rely on powerful trusted third parties (TTPs) that can deanonymize (or link) users' connections undetectably and at will. Subsequent designs [4, 8, 34] have introduced clever cryptography to replace the TTPs, thus solving the trust problem at a cost of much computation and communication overhead for the users and for the SPs.

Blacklistable Anonymous Credentials.

One such TTP-free anonymous blacklisting design is Tsang et al.'s *Blacklistable Anonymous Credentials* (BLAC) [33]. In BLAC, a semi-trusted *group manager* (GM) registers each new user into the system by issuing it an anonymous credential $C(x)$ that encodes as an attribute a secret key x unique to that user. (The GM is semi-trusted in the sense that, although the users and the SPs must trust the GM for the system to provide availability and accountability, the users need not trust it to maintain their anonymity.) The user holding $C(x)$ authenticates to an SP by producing a *ticket* $\Gamma = (g, g^x)$

*An extended version of this paper is available [20].

[1] https://www.torproject.org/

[2] https://en.wikipedia.org/
[3] https://www.yelp.com/
[4] https://slashdot.org/
[5] http://www.craigslist.org/

WPES'13, November 4, 2013, Berlin, Germany.

ACM 978-1-4503-2485-4/13/11 ...$15.00.
http://dx.doi.org/10.1145/2517840.2517855.

together with zero-knowledge proofs that (i) the exponent x used to compute Γ is the same as the secret key x in $C(x)$, and (ii) no ticket on the SP's *blacklist* of tickets from past abusive sessions uses that same x. Both proofs are instantiable using standard techniques for proving statements about the equality [12, §3.2] and inequality [10, §6] of discrete logarithms (DLs). The SP grants the user access (and stores Γ for future reference) if and only if it accepts both proofs. If it later deems the user's actions during the session to have been abusive, the SP can add Γ to its blacklist to curtail further abuse by that user. The notion of "abuse" in this model is entirely subjective: each SP must define, identify, and penalize abusive behaviour in a way that is appropriate within the context of its user community and the services it provides. For example, an IRC network may define abuse to include hate speech, cyberbullying, evading a ban, spamming, and copyright infringement. Users that engage in abusive behaviour are blacklisted for a duration commensurate with the severity and perceived likelihood of recurrence of a particular abusive behaviour. Servers on the IRC network refuse connection requests from users on the blacklist.[6]

Fifty shades of BLAC. BLAC's all-or-nothing approach to revocation may be overly punitive in some settings. The anonymous blacklisting literature includes two variants of BLAC that seek to address this shortcoming. The first variant does so with a *d-strikes-out revocation policy* [34], wherein each anonymous user may authenticate until it has accumulated d or more tickets on the blacklist (after which future authentications will fail). The second variant supports *reputation-based blacklisting* [2], wherein SPs can assign scores (both positive and negative) to the anonymous actions of users, and each user may subsequently authenticate only if the aggregate score associated with all of its scored tickets exceeds some minimum threshold value. (More generally, SPs may categorize positive and negative scores according to the nature of the associated behaviours and require each authenticating user to prove a statement pertaining to its aggregate scores *across all categories*.) We herein refer to the first variant as 'd-BLAC' and to the second variant as 'BLACR'; we continue to refer to the original system simply as 'BLAC' or, occasionally, as 'vanilla BLAC' to emphasize when a remark applies to BLAC but not to d-BLAC or to BLACR.

Scalability of BLAC. Judged solely on the basis of privacy and functionality, the BLAC approach to anonymous blacklisting is very attractive indeed; judged also on the basis of scalability, however, it becomes much less so. In all three BLAC variants, the bottleneck operation is the second zero-knowledge proof (in which the user demonstrates that its own tickets on an SP's blacklist do not meet that SP's revocation criteria). The 'size' of this proof scales as the total number of tickets on the blacklist, which can introduce substantial delays and consume considerable bandwidth and computation capacity for large SPs that cater to millions of users. Prior work [2, 33, 34] essentially regards the zero-knowledge proofs as "black boxes", to be instantiated using the standard techniques from the literature. Unfortunately, those standard techniques become prohibitively expensive even for moderate-sized blacklists (say, those containing a few hundred tickets). This fact has contributed to the common conception [1, 4, 17] that—despite being both novel and elegant—BLAC's approach to anonymous blacklisting is impractical for large SPs.

[6]An alternative to such subjective revocation is *objective* (or *contract-based*) revocation [17, §IV.A], introduced by Schwartz, Brumley, and McCune and exemplified by their RECAP protocol [31]. In the objective revocation model, users and SPs enter into mutually binding *contracts* that stipulate unambiguously the SPs' terms of service. An SP can then revoke a given anonymous user's authentication privileges if and only if that user *provably* violates the terms set forth in its contract with the SP. Unfortunately, some perfectly reasonable terms of service are simply too nebulous to specify and enforce without relying on human subjectivity. For example, consider a contract that forbids vandalizing articles on Wikipedia or one that forbids posting disingenuous reviews on Yelp—in both examples, identifying contract violations seems to necessitate a human in the loop.

Our contributions.

In this work, we improve on BLAC and its derivatives by peering *inside* their zero-knowledge black boxes and optimizing the underlying protocols; that is, we innovate by *"thinking inside the BLAC box"*. We find, in particular, that existing *batch proof and verification* techniques can reduce substantially the communication and computation overhead in vanilla BLAC's bottleneck zero-knowledge proof. We then extend our optimized protocol in a novel way to deal also with the bottleneck proofs in d-BLAC and BLACR. At the heart of these latter constructions is a new system for batch zero-knowledge proofs of partial knowledge for DLs over *non-monotone* access structures. Our new protocols are the first in the literature for batch zero-knowledge proofs over non-monotone access structures and we suspect that our techniques will find applications in speeding up other cryptographic protocols that also require proofs of similar statements.

We refer to our new protocol suite—that is, to BLAC equipped with our new and improved black boxes—as BLACRONYM. Our BLACRONYM protocols offer similar functionality and superior performance when compared to the "default" protocol instantiations that are used in BLAC, d-BLAC, and BLACR.

Paper outline. The rest of the paper proceeds as follows. We begin with a brief discussion of our notation and cost model in §2. In §3, we introduce the formal model for BLACRONYM and describe each of our new zero-knowledge protocols in detail. We compare the communication and computation costs of our BLACRONYM protocols with those of BLAC, d-BLAC, and BLACR in §4 and we conclude in §5.

2. NOTATION AND PRELIMINARIES

Throughout, $\mathbb{G}$ denotes a finite multiplicative group with 2τ-bit prime order q and a fixed generator $g \in \mathbb{G}$, and $\mathbb{G}^* = \mathbb{G}\setminus\{1\}$ denotes the set of non-identity elements in $\mathbb{G}$. Similarly, $\mathbb{Z}_q$ denotes the field of integers modulo q, and $\mathbb{Z}_q^* = \mathbb{Z}_q \setminus \{0\}$ denotes its multiplicative group of units. For a given finite set S, $a \in_R S$ denotes uniform random selection of an element a from S and $A \subseteq_d S$ denotes that the set A is a size-d subset of S. If $S \subseteq \mathbb{N}$, then S_j is the j^{th} smallest element in S. We use $s\|t$ to denote the concatenation of binary strings s and t. A function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ is *negligible* if it vanishes faster than the inverse of every positive, real-valued polynomial. An event E occurs with *negligible probability* in κ if the probability that it occurs is a negligible function of κ, and E occurs with *overwhelming probability* in κ if the probability that $\neg E$ occurs is a negligible function of κ.

2.1 Computing powers & products of powers

As our focus in this work is on optimizing certain zero-knowledge proofs about DLs, we take this opportunity to introduce our cost model. In implementations of DL-based proofs, the CPU time required to compute powers (exponentiation) and products of powers (multi-exponentiation) dominates the running time. We measure the cost of such operations by the expected number of multiplications they require. (For simplicity, we ignore the cost of arithmetic required, for example, to determine which exponents to use in a given exponentiation, although we note that our own BLACRONYM protocols fare no worse than the original BLAC, d-BLAC, and BLACR protocols in this respect.) Following Bellare, Garay, and Rabin [5, §2.3], we write $\text{ExpCost}_{\mathbb{G}}^{m}(b)$ to denote the cost of raising a generator $g \in \mathbb{G}$ to m distinct, random b-bit powers. When $m = 1$, we omit it from

the notation. The classic "square-and-multiply" algorithm yields

$$\mathrm{ExpCost}_{\mathbb{G}}(b) \leq 1.5b$$

and more sophisticated windowing methods [7, 29] can reduce the coefficient in this bound to about 1.2. We also note the trivial bound

$$\mathrm{ExpCost}^{m}_{\mathbb{G}}(b) \leq m\,\mathrm{ExpCost}_{\mathbb{G}}(b)$$

and remark that well-known techniques from the literature [9, 24, 25] can make the inequality in this bound strict.

For products of powers, we replace the bit length b by ordered pairs, so that $\mathrm{ExpCost}^{m}_{\mathbb{G}}\big((n_1,b_1),\ldots,(n_k,b_k)\big)$ denotes the cost of computing m products of powers of a common set of $n = \sum_{i=1}^{k} n_i$ bases of which, in each product, exactly n_i are raised to random b_i-bit exponents for $i = 1,\ldots,k$. Bellare et al.'s fast multiexponentiation algorithm [5, §3.2] gives the bound

$$\mathrm{ExpCost}^{m}_{\mathbb{G}}\big((n_1,b_1),\ldots,(n_k,b_k)\big) \leq m\Big(\max_{1\leq i\leq k}\{b_i\} + \tfrac{1}{2}\sum_{i=1}^{k} n_i b_i\Big)$$

and we remark that one can lower the actual cost by using precomputation [9, 24, 25], at the expense of additional storage for the precomputed values.

3. FADE TO BLACRONYM

The BLACRONYM design is identical to that of BLAC and its derivatives; our own contributions are contained entirely within the black boxes implementing zero-knowledge proofs. Nonetheless, we shall find it useful to provide some additional detail on that basic design, if only for completeness. (Additionally, some aspects of the design are relevant in our security analyses.) We claim no originality in the first subsection; the BLAC design is due entirely to Tsang et al. [34] and any differences in our presentation of it are purely cosmetic.

3.1 Model

The basic setting is exactly as in the introduction: A population of anonymous users wish to use the services offered by one or more participating SPs who, in turn, will only service those users whom they can hold individually accountable for their respective actions. The SPs do this by each maintaining a *blacklist* of metadata about past abuses and requiring each authenticating user to prove that it is not responsible for "too many" of those abuses. (The precise definition of "too many" abuses, of course, varies from vanilla BLAC to d-BLAC to BLACR.) A semi-trusted (and possibly distributed) GM facilitates all of this.

Initialization. The GM runs the *initialization protocol* once to set up the system. The protocol takes as input 1^{τ} for security parameter τ and it outputs (i) a description $(\mathbb{G},q,g)$ of a *DDH-hard* group $\mathbb{G}$ [28, §3.7], (ii) a random oracle $\mathcal{H}$ mapping binary strings to elements of $\mathbb{G}$, and (iii) a public-private key pair (pk,sk). In an actual implementation, $\mathcal{H}$ would be a cryptographically secure hash function and $\mathbb{G}$ an elliptic curve group that can be efficiently "hashed into" [22]. Each of the remaining algorithms takes $(\mathbb{G},q,g,\mathcal{H},pk)$ as an *implicit* input.

Registration. Each user runs the interactive *registration protocol* once with the GM to enroll in the system. Upon successful completion of the protocol, the user obtains an anonymous credential $C(x)$ under the GM's public key, which encodes a secret key x unique to the user. We emphasize that the GM learns *zero information* about x during this interaction and it therefore has no computational advantage in, for example, linking authentications associated with a common real-world user. The particular choice of credential system is immaterial to the following protocols, provided credentials in it

1. are unconditionally hiding,
2. are fully anonymous (i.e., multiple showings of the same credential are mutually unlinkable), and
3. admit *efficient* honest-verifier perfect zero-knowledge proofs of knowledge about x.

(Note that *unconditional* hiding and *perfect* zero-knowledge are not strictly required, but they do help to simplify the security analysis.) The authors of BLAC and its derivatives suggest using *BBS+ signatures* [3, §4.2] and we do not object. BBS+ signatures satisfy each of the above criteria and are computationally binding under the n-SDH assumption [6, §3].

Despite its learning nothing about x during registration, the users and SPs must still trust the GM. For instance, SPs must trust the GM to issue at most one credential to any given user, lest some users obtain several credentials with which to launch *Sybil attacks* [15] against the SPs. A reliable GM must consequently collect (and retain, in some form) *personally identifiable information* (PII) about each enrolled user; those users, in turn, must trust the GM to act responsibly with their PII. (See Henry and Goldberg's survey of anonymous blacklisting systems [17, §V] for more on this point.) As our new protocols do not affect registration, we defer further discussion about it to Tsang et al. [34, §4.1.2 and §5.3].

Authentication. The *authentication protocol* is an interactive protocol that anonymous users run with SPs to initiate their anonymous sessions. The user's input is its credential $C(x)$ and a random string $z \in_{\mathrm{R}} \{0,1\}^{2\tau}$, and the SP's input is its current *blacklist* $\mathcal{B}$, *revocation policy* ρ_s, and a *soundness parameter* λ. (The revocation policy is a function that takes as input $\mathcal{B}$ and the user's secret key x, and outputs a boolean value that indicates whether the entries on $\mathcal{B}$ with tickets encoding x meet the SP's revocation criteria.) The SP will accept the authentication only if the user convinces it that $\rho_s(\mathcal{B},x) = 0$ with probability overwhelming in λ. The output of the authentication protocol is a return value $b \in \{0,1,\perp\}$ and an *authentication transcript* ϖ containing the *ticket* $\Gamma = \big(z,\mathcal{H}(z\|s)^x\big)$, where $s \in \{0,1\}^*$ is a fixed, publicly known *canonical name* for the SP. A return value of 0 indicates that the SP rejected the authentication and a return value of 1 indicates that the SP accepted the authentication. A return value of $\perp$ indicates that the user aborted the protocol prematurely (perhaps because it discovered that $\rho_s(\mathcal{B},x) = 1$). In practice, we assume that $\lambda \ll \tau$, say $\lambda = 40$ or 60. The SP should output $b = 1$ with probability at most $1 / 2^{\lambda}$ when $\rho_s(\mathcal{B},x) = 1$.

Blacklist management. Blacklist management involves three protocols that SPs use to manage their respective blacklists. The *extraction protocol* takes as input an authentication transcript ϖ and it outputs the associated *ticket* Γ. The *add protocol* takes as input a blacklist $\mathcal{B}$ and a ticket Γ (and, in the case of reputation-based blacklisting, an associated score ς), and it outputs a new blacklist $\mathcal{B}'$ that contains every entry from $\mathcal{B}$ plus a new entry for ticket Γ (with score ς). The *remove protocol* takes as input a blacklist $\mathcal{B}$ and a ticket Γ, and it outputs a new blacklist $\mathcal{B}'$ that contains every entry from $\mathcal{B}$ whose ticket is not equal to Γ.

Security definitions.

We provide (informal) definitions for the necessary security and privacy properties of a *secure* BLAC construction in Appendix A. (Note that such informal definitions suffice for our purposes: since we only modify the internals of black boxes, the existing system-level security proofs for BLAC and d-BLAC [34, §7.2] and for BLACR [2, Appendix A] also prove that our BLACRONYM protocol suite yields a secure BLAC construction.)

Federated identity systems.

Note that the GM and the SPs in a secure BLAC construction gain *no adversarial advantage* from colluding with one another (beyond, perhaps, some additional inference power they obtain by combining metadata about their respective transaction logs); in fact, in some settings a single entity may wish to operate simulatenously as a GM and as an SP in a single BLAC deployment. Alternatively, an SP may wish to outsource its (expensive) verification operations to some (computationally well-equipped, benevolent) third party. If the outsourced verifier were dishonest, then it could accept invalid proofs, thus helping repeatedly misbehaving users circumvent the revocation policy; however, the *SP must already trust the GM* to not help users circumvent revocation by giving them multiple credentials. Therefore, the SP can outsource verification to the GM *without introducing any new trust assumptions*. The latter observation implies that our BLACRONYM protocols could be used in the framework of a *federated identity system*, such as OpenID[7], to obtain strong anonymity guarantees. The *OpenID Provider* (OP) could be the GM and each *Relying Party* (RP) an SP. The GM would then check (on behalf of each SP) the zero-knowledge proofs in the authentication protocol and provide a signed token for the SP (including the associated ticket and blacklist version) provided the verification succeeds.

3.2 Vanilla BLACRONYM

The first black box that we redesign comes from the authentication protocol in vanilla BLAC; that is, we provide an alternative instantiation for the protocol that Tsang et al. label SPK_2 [34, §5.4 and §6.1]. That protocol is itself a composition of two subprotocols:

- SPK_4 proves knowledge of x such that (i) the user holds a valid credential $C(x)$ with secret key x and (ii) the second component in the user's ticket $\Gamma = (z_0, H_0)$ has the form $H_0 = \mathcal{H}(z_0 \| s)^x$ for that same x and the SP's canonical name s, and
- SPK_5 proves that $\log_{h_i} H_i \neq \log_{h_0} H_0$ for each $i \in [1,n]$, where $\mathcal{B} = \{(z_1, H_1), \ldots, (z_n, H_n)\}$ is the SP's blacklist and $h_i = \mathcal{H}(z_i \| s)$ for each $i = 0, \ldots, n$.

The cost of SPK_4 is independent of n and its implementation details depend on the particular choice of anonymous credential system; thus, we focus our attention on the more costly (and credential-agnostic) SPK_5 subprotocol.

An alternative instantiation for SPK_5 in vanilla BLAC

The user (playing the role of the *prover*) and the SP (playing the role of the *verifier*) in this subprotocol take as common input $n+1$ pairs of group elements $(h_0, H_0), \ldots, (h_n, H_n) \in \mathbb{G}^* \times \mathbb{G}^*$. The goal is for the user to prove that $\log_{h_i} H_i \neq \log_{h_0} H_0$ for every $i \in [1,n]$. For vanilla BLAC, Tsang et al. suggest instantiating SPK_5 with a textbook n-fold parallel composition of the following protocol for the special "$n = 1$" case of the above claim.

Special case: Inequality of two discrete logarithms. Suppose that prover P and verifier V take as common input two pairs of group elements $(h_0, H_0), (h_1, H_1) \in \mathbb{G}^* \times \mathbb{G}^*$. Protocol 1 implements a system for "special honest-verifier" perfect zero-knowledge proofs of knowledge[8] in which P demonstrates knowledge of an exponent $x \in \mathbb{Z}_q^*$ such that $\log_{h_0} H_0 = x$ and $\log_{h_1} H_1 \neq x$. We typically assume that $y = \log_{h_1} H_1$ is unknown to P, although the proof is still sound when this is not the case. The protocol is due to Camenisch and Shoup [10, §6] and we denote it in *Camenisch-Stadler notation* [11, §4] by $\mathrm{PK}\{\, x : H_0 = h_0^x \ \wedge\ H_1 \neq h_1^x \,\}$.

Protocol 1 (Inequality of two DLs).

Common input: $(h_0, H_0), (h_1, H_1) \in \mathbb{G}^* \times \mathbb{G}^*$

Prover's input: $x = \log_{h_0} H_0$

1. P chooses a blinding factor $r \in_{\mathrm{R}} \mathbb{Z}_q^*$, and then it computes the auxiliary commitment $C_1 = \left(h_1^x \,/\, H_1\right)^r$ and sends C_1 to V.
2. P engages V in $\mathrm{PK}\{(\alpha, \beta) : 1 = h_0^\alpha H_0^\beta \ \wedge\ C_1 = h_1^\alpha H_1^\beta\}$ using $\alpha = xr \bmod q$ and $\beta = -r \bmod q$:
   (a) P chooses blinding factors $s_1, s_2 \in_{\mathrm{R}} \mathbb{Z}_q^*$, and then it computes $R_0 = h_0^{s_1} H_0^{s_2}$ and $R_1 = h_1^{s_1} H_1^{s_2}$, and sends (R_0, R_1) to V.
   (b) V picks a challenge $c \in_{\mathrm{R}} \mathbb{Z}_q$ and sends it to P.
   (c) P computes the responses $u_1 = s_1 - cxr \bmod q$ and $u_2 = s_2 + cr \bmod q$, and sends (u_1, u_2) to V.
   (d) V accepts if $R_0 = h_0^{u_1} H_0^{u_2}$ and $R_1 = h_1^{u_1} H_1^{u_2} C_1^c$; otherwise, V rejects.
3. V accepts if $C_1 \neq 1$ and if it accepts in Step 2(d); otherwise, V rejects. ◇

The subprotocol in Step 2 assures honest V that, with a probability overwhelming in τ, P knows exponents (α, β) satisfying $1 = h_0^\alpha H_0^\beta$ and $C_1 = h_1^\alpha H_1^\beta$. From the first expression, we obtain $\alpha = -\beta \log_{h_0} H_0$. Substituting for α in the second expression yields $C_1 = \left(h_1^{-\log_{h_0} H_0} H_1\right)^\beta$, which, as V accepts in Step 3 only when $C_1 \neq 1$, implies that $\log_{h_0} H_0 \neq \log_{h_1} H_1$.

Note that P may send the auxiliary commitment C_1 as part of Step 2(a) and that V may verify that $C_1 \neq 1$ as part of Step 2(d); thus, Protocol 1 is in fact a three-message *sigma protocol* [14, Definition 1]. Regarding efficiency, we see by inspection that P sends three elements of $\mathbb{G}$ and two elements of $\mathbb{Z}_q$ to V, and that V sends just one element of $\mathbb{Z}_q$ to P. Likewise, the expected number of multiplications in $\mathbb{G}$ for P to compute is

$$\mathrm{ExpCost}_{\mathbb{G}}^2((2, 2\tau)) + \mathrm{ExpCost}_{\mathbb{G}}((2, 2\tau)) \leq 12\tau,$$

and for V it is

$$\mathrm{ExpCost}_{\mathbb{G}}((2, 2\tau)) + \mathrm{ExpCost}_{\mathbb{G}}((3, 2\tau)) \leq 9\tau.$$

General case: Inequality of one discrete logarithm with several others. In the textbook parallelization of Protocol 1 mentioned above, P chooses fresh blinding factors $r, s_1, s_2 \in_{\mathrm{R}} \mathbb{Z}_q^*$ for each of the n component instances, while V picks a *single* challenge $c \in_{\mathrm{R}} \mathbb{Z}_q$ to which P must issue an n-fold response. As an optimization, P may reuse a single set of blinding factors *across all instances*, thus eliminating the need to compute and send $n-1$ commitments from $\mathbb{G}$ (since R_0 will be identical across all instances) and $2n-2$ responses from $\mathbb{Z}_q$ (since u_1, u_2 will be identical across all instances). We note that such reuse of randomness in a zero-knowledge proof warrants extreme caution; indeed, the resulting protocol is emphatically *not* perfect zero-knowledge when considered in isolation. We believe that one could prove it is *computational* zero-knowledge under the DDH assumption, though we have not attempted to do so.

Instead, we recall that SPK_5 is just one of two subprotocols comprising SPK_2 and that, in the other subprotocol, SPK_4, the user (holding credential $C(x)$) computes $H_0 = h_0^x$ and then *proves in (perfect) zero-knowledge* that the secret exponent x is consistent with $C(x)$. Therefore, a simulator for the composed protocol simply (i) chooses a 'fake' exponent $y \in_{\mathrm{R}} \mathbb{Z}_q^*$ arbitrarily, (ii) outputs $H_0 = h_0^y$, (iii) perfectly *simulates* a proof of correctness for $H_0 = h_0^y$ with respect to

[7] https://openid.net/

[8] A proof is *special honest-verifier* zero-knowledge [14, Definition 1] if a PPT simulator can output accepting transcripts from a distribution indistinguishable from that arising from real interactions between the honest prover and the honest verifier with the same public inputs X and challenge c. The zero-knowledge property here is "special" because the simulator can produce such transcripts *given the challenge c as input*, which is a necessary condition for Cramer, Damgård, and Schoenmakers' *proofs of partial knowledge* framework [13].

$C(x)$, and then (iv) follows the above-optimized inequality of DLs proof *honestly* to complete the simulation. (If $\log_{h_i} H_i$ equals the chosen y for some $i \in [1,n]$, an event that only occurs with probability negligible in τ when $n \in poly(\tau)$, the simulator just selects a new $y \in_R \mathbb{Z}_q^*$ and restarts.) Since the first part of the simulation is perfect by assumption, and the second part is perfect by definition, it follows that the *entire* simulation is in fact perfect. In other words, the above simulation strategy establishes that, if SPK_4 is honest-verifier perfect zero-knowledge, then so is all of SPK_2, even when the user reuses its blinding factors across all parallel instances in SPK_5.

General case with batch verification. We can, in fact, do much better by incorporating ideas from *batch testing*. For example, V can employ Bellare et al.'s *small-exponent batch test* [5, §3.3] to check all verification equations in Step 2(d) of the composed protocol *simultaneously* using a single 3-base multiexponentiation in $\mathbb{G}$ with exponents from $\mathbb{Z}_q$, plus three $(n+1)$-base multiexponentiations and one n-base multiexponentiation in $\mathbb{G}$ with exponents from $[0, 2^\lambda - 1]$. (With small-exponents batching, the $n+1$ verification equations, $R_0 \stackrel{?}{=} h_0^{u_1} H_0^{u_2}$ and $R_i \stackrel{?}{=} h_i^{u_1} H_i^{u_2} C_i^c$ for $i = 1,\ldots,n$, reduce to a *single* equation $R_0 \prod_{i=i}^n R_i^{a_i} \stackrel{?}{=} \big(h_0 \prod_{i=1}^n h_i^{a_i}\big)^{u_1} \big(H_0 \prod_{i=1}^n H_i^{a_i}\big)^{u_2} \cdot \big(\prod_{i=1}^n C_i^{a_i}\big)^c$, where V selects the exponents $a_i \in_R [0, 2^\lambda - 1]$ for $i = 1,\ldots,n$.) The expected number of multiplications for V to compute in $\mathbb{G}$ therefore reduces from

$$\begin{aligned}&\mathrm{ExpCost}_{\mathbb{G}}\big((2,2\tau)\big)\\ &\quad + n\,\mathrm{ExpCost}_{\mathbb{G}}\big((3,2\tau)\big) \le (5n+4)\tau\end{aligned}$$

to just

$$\begin{aligned}&4\,\mathrm{ExpCost}_{\mathbb{G}}\big((n,\lambda)\big) + 3\\ &\quad + \mathrm{ExpCost}_{\mathbb{G}}\big((3,2\tau)\big) \le 5\tau + (2n+2)\lambda + 3.\end{aligned}$$

(Recall that λ is V's soundness parameter and that $\lambda \ll \tau$.) Note that V can use small-exponent batching to substantially reduce its verification cost, even *without P's knowledge or cooperation*. The resulting protocol is still special honest-verifier perfect zero-knowledge and a standard argument [5, §3.3] gives a (fairly loose) upper bound of $1 / 2^\lambda$ for its knowledge error.

Batch protocol for vanilla BLAC.

Protocol 2 extends the above idea by applying small-exponent batch testing *before parallelizing the subprotocol in Step 2 of Protocol 1*: upon receiving the auxiliary commitments $C_1,\ldots,C_n$ from P, V selects a list of n random scalars $a_1,\ldots,a_n \in_R [1, 2^\lambda - 1]$, and then *both parties* use the a_i to compute the two bases $h = h_0 \prod_{i=1}^n h_i^{a_i}$ and $H = H_0 \prod_{i=i}^n H_i^{a_i}$. The subprotocol in Step 2 becomes $\mathrm{PK}\{(\alpha,\beta) : C = h^\alpha H^\beta\}$, where $\alpha = rx \bmod q$, $\beta = -r \bmod q$, and $C = \prod_{i=1}^n C_i^{a_i}$. Note that, as seen in §2.1, h, H, and C each require about $(n/2+1)\lambda$ multiplications in $\mathbb{G}$ to compute, but that the resulting Step 2 subprotocol has cost *independent of* n. Thus, although the batch protocol requires an additional round of interaction (and, hence, P's cooperation), it extends V's computational savings from batch verification to P as well, and also significantly reduces the overall communication cost of the protocol.

Protocol 2 (Batched inequality of one DL with several others).

Common input: $(h_0,H_0),\ldots,(h_n,H_n) \in \mathbb{G}^* \times \mathbb{G}^*$
Prover's input: $x = \log_{h_0} H_0$

1. P chooses a blinding factor $r \in_R \mathbb{Z}_q^*$, and then it computes the auxiliary commitments $C_i = \big(h_i^x / H_i\big)^r$ for each $i = 1,\ldots,n$ and sends $(C_1,\ldots,C_n)$ to V.
2. V picks scalars $a_1,\ldots,a_n \in_R [0, 2^\lambda - 1]$ and sends them to P.
3. P and V each compute $h = h_0 \prod_{i=1}^n h_i^{a_i}$ and $H = H_0 \prod_{i=1}^n H_i^{a_i}$, and V computes $C = \prod_{i=1}^n C_i^{a_i}$.
4. P engages V in $\mathrm{PK}\{(\alpha,\beta) : C = h^\alpha H^\beta\}$, using $\alpha = xr \bmod q$, and $\beta = -r \bmod q$:
   (a) P chooses blinding factors $s_1, s_2 \in_R \mathbb{Z}_q^*$, and then it computes $R = h^{s_1} H^{s_2}$ and sends R to V.
   (b) V picks a challenge $c \in_R \mathbb{Z}_q$ and sends it to P.
   (c) P computes the responses $u_1 = s_1 - cxr \bmod q$ and $u_2 = s_2 + cr \bmod q$, and sends (u_1,u_2) to V.
   (d) V accepts if $R = h^{u_1} H^{u_2} C^c$; otherwise, V rejects.
5. V accepts if $C_i \neq 1$ for each $i = 1,\ldots,n$ and if it accepts in Step 2(d); otherwise, V rejects. ◇

We prove in the extended version of this paper [20] that Protocol 2 implements a system for special honest-verifier perfect zero-knowledge proofs of knowledge with knowledge error at most $1/2^\lambda$, provided the simulator is privy to $\log_{h_0} H_0$ (as we can assume it is in the BLACRONYM setting). We denote the new protocol by $\mathrm{BPK}\{x : H_0 = h_0^x \wedge \big(\bigwedge_{i=1}^n H_i \neq h_i^x\big)\}$, where BPK denotes a *batch* proof of knowledge in otherwise standard Camenisch-Stadler notation. The computation cost for V is little changed from that given in the above analysis with batch verification; however, the bidirectional communication cost and P's computation cost are both much improved. In particular, while V must now send $n\lambda$ extra bits to P (i.e., the small exponents $a_1,\ldots,a_n$), P only sends $n+1$ elements of $\mathbb{G}$ (i.e., the auxiliary commitments $C_1,\ldots,C_n$ and a single other commitment R) and two elements of $\mathbb{Z}_q^*$ (i.e., the responses u_1 and u_2) to V. (The naive instantiation requires P to send $3n$ elements of $\mathbb{G}$ and $2n$ elements of $\mathbb{Z}_q$.) Similarly, although P must still compute $n+1$ two-base multiexponentiations in $\mathbb{G}$ with exponents from $\mathbb{Z}_q^*$ (to produce $C_1,\ldots,C_n$ and R), the expected number of multiplications in $\mathbb{G}$ for P (to set up and then execute the Step 2 subprotocol) reduces from

$$(n+1)\,\mathrm{ExpCost}_{\mathbb{G}}\big((2,2\tau)\big) \le (4n+4)\tau$$

in the naive instantiation to just

$$\begin{aligned}&2 \cdot \mathrm{ExpCost}_{\mathbb{G}}\big((n,\lambda)\big) + 2\\ &\quad + \mathrm{ExpCost}_{\mathbb{G}}\big((2,2\tau)\big) \le 4\tau + (n+2)\lambda + 2.\end{aligned}$$

Likewise, the expected number of multiplications for V to compute in $\mathbb{G}$ (during the *entire* protocol) reduces from

$$\begin{aligned}&n \cdot \Big(\mathrm{ExpCost}_{\mathbb{G}}\big((2,2\tau)\big)\\ &\quad + \mathrm{ExpCost}_{\mathbb{G}}\big((3,2\tau)\big)\Big) \le 9n\tau\end{aligned}$$

in the "textbook" instantiation to just

$$\begin{aligned}&3 \cdot \mathrm{ExpCost}_{\mathbb{G}}\big((n,\lambda)\big) + 2\\ &\quad + \mathrm{ExpCost}_{\mathbb{G}}\big((3,2\tau)\big) \le 5\tau + \tfrac{1}{2}(3n+6)\lambda + 2.\end{aligned}$$

Comparison with vanilla BLAC.

In vanilla BLAC's default SPK_5 instantiation, the user sends $3n$ elements from $\mathbb{G}$ and $2n$ elements from $\mathbb{Z}_q$ to the SP; thus, our optimizations reduce the communication cost by about 75%.[9] (Using point compression for the elements of $\mathbb{G}$, the communication savings increase to about 80% at the cost of some additional computation overhead for point decompression.) The computational savings are similar: the user and the SP compute, respectively, about

[9] We arrive at this estimate by assuming (i) that $\mathbb{G}$ is an elliptic curve group whose elements are about twice the bit length of elements from $\mathbb{Z}_q$, and (ii) that implementations use Fiat and Shamir's heuristic [16] to make SPK_5 noninteractive in the random oracle model so that, rather than the SP sending $a_0,\ldots,a_n$ to the user, the user and the SP *each compute the* a_i *locally* as the output of some cryptographically secure hash function.

$(4\tau + \lambda) \,/\, 12\tau$ and $\lambda \,/\, 6\tau$ times as many multiplications in $\mathbb{G}$. For the (perfectly reasonable) parameter choices $\lambda = 40$ and $\tau = 128$, this is about a 64% cost reduction for the user and about a 95% cost reduction for the SP. Furthermore, most of the user's remaining computation cost arises from computing the auxiliary commitments $C_1, \ldots, C_n$, which are readily precomputable [34, §7.1]. (If the user precomputes the C_i, then its online computation cost is only about $\lambda \,/\, 12\tau$ times the cost of the naive protocol with precomputation.) Moreover, once it has computed $C_i = \left(h_i^x \,/\, H_i\right)^r$ to use in one protocol run, the user can choose a new blinding factor $r' \in_{\mathrm{R}} \mathbb{Z}_q^*$ and *reblind* C_i as $C_i^{r'} = \left(h_i^x \,/\, H_i\right)^{r \cdot r'}$ to use in a subsequent protocol run. Such reblinding requires just $\mathrm{ExpCost}_{\mathbb{G}}(2\tau) \leq 2.4\tau$ multiplications in $\mathbb{G}$, which is a little over half of what is required to compute a new C_i from scratch.

3.3 BLACRONYM with *d*-strikes-out

The second black box that we redesign comes from the authentication protocol in d-BLAC; that is, we provide an alternative instantiation for the protocol that Tsang et al. label SPK_3 [34, §5.5 and §6.1]. As with SPK_2, two subprotocols comprise SPK_3:

- SPK_4 again proves knowledge of x such that (i) the user holds a valid credential $C(x)$ encoding x and (ii) the second component in the user's ticket $\Gamma = (z_0, H_0)$ has the form $H_0 = \mathcal{H}(z_0 \| s)^x$ for that same x and the SP's canonical name s, and
- SPK_5 proves that there is a set $S' \subseteq [1,n]$ of size *at least* $n - d + 1$ such that $\log_{h_i} H_i \neq \log_{h_0} H_0$ for any $i \in S'$, where $\mathcal{B} = \{(z_1, H_1), \ldots (z_n, H_n)\}$ is the SP's blacklist and $h_i = \mathcal{H}(z_i \| s)$ for $i = 0, \ldots, n$.

We note that having the h_i be output by the random oracle $\mathcal{H}$ ensures, under the DDH assumption, that neither the user nor the SP knows $\log_g h_i$ for any $i \in [1,n]$ nor $\log_{h_i} h_j$ for any $z_i \neq z_j$, except perhaps with probability negligible in τ. (In what follows, we assume that the z_i are all pairwise distinct—which is trivial to check—so that neither party knows $\log_{h_i} h_j$ for any $i \neq j$.) Again, we focus our attention exclusively on improving SPK_5, the expensive (and credential-agnostic) protocol.

Building block: All-but-k mercurial commitments. Our protocol uses a recently proposed *all-but-k* variant of mercurial vector commitments [18]. In that scheme, a trusted initializer prepares a public reference string $\mathrm{ABK}(N)$ comprising $N + 1$ elements from a finite group $\tilde{\mathbb{G}}$ equipped with an admissible bilinear map $e\colon \tilde{\mathbb{G}} \times \tilde{\mathbb{G}} \to \mathbb{G}_{\mathrm{T}}$. (Note that $\tilde{\mathbb{G}}$ is *not* the same as the DDH-hard group $\mathbb{G}$; in fact, $\tilde{\mathbb{G}}$ need not have the same order as $\mathbb{G}$.) In the BLACRONYM setting, the trusted initializer would be the GM and the reference string would be part of the GM's public key.[10] Given $\mathrm{ABK}(N)$, a user can commit to an arbitrary subsequence of values $(c_i)_{i \in S}$ indexed by $S \subseteq [1, N]$ and later open that commitment to any *supersequence* $(c_i)_{i=1}^N$ consistent with the original commitment. The recipient learns nothing about S, except for a prover-specified upper bound $k \geq N - |S|$ on the number of terms in $(c_i)_{i=1}^N$ that were not specified in the initial subsequence. (Note that a user can also commit to subsequences of $(c_i)_{i=1}^n$ for $n < N$ by setting $c_i = 0$ for $i = n+1, \ldots, N$ and revealing the length n as part of the initial commitment.) We use the following abridged notation for the all-but-k protocols:

- $\text{ABK-Commit}\left(S, (c_i)_{i \in S}\right)$ outputs a commitment $\mathscr{C}$ to $(c_i)_{i \in S}$,
- $\text{ABK-Open}\left(\mathscr{C}, k, (c_i)_{i \in [1,N] \setminus S}\right)$ for any $k \geq N - |S|$ outputs a witness $\mathcal{W}$, and
- $\text{ABK-Verify}\left(\mathscr{C}, \mathcal{W}, k, (c_i)_{i=1}^N\right)$ verifies that $\mathcal{W}$ witnesses the *valid* opening $\left(k, (c_i)_{i=1}^N\right)$ of $\mathscr{C}$.

[10] Alternatively, each SP could generate its own all-but-k reference string—the *recipients* of a commitment (i.e., the SPs) must trust the initializer in order for commitments to be binding, but the *committers* (i.e., the users) need not trust the initializer in order for commitments to be hiding.

An alternative instantiation for SPK_5 in d-BLAC

The user and the SP in this subprotocol have as common input $n + 1$ pairs of group elements $(h_0, H_0), \ldots, (h_n, H_n) \in \mathbb{G}^* \times \mathbb{G}^*$ such that $\log_g h_i$ is unknown for all $i \in [1,n]$ and $\log_{h_i} h_j$ is unknown whenever $i \neq j$. Let $S \subseteq [1,n]$ be the set of indices for which $\log_{h_i} H_i = \log_{h_0} H_0$ and let $S' = [1,n] \setminus S$. The goal is for the user to prove that $|S| < d$ or, equivalently, that $|S'| > n - d$. The most standard instantiation for this proof follows by using Cramer et al.'s method [13] to parallelize Protocol 1 into a proof of partial knowledge for the latter, *monotone* statement; in our case, we opt to prove the former, *non-monotone* statement. We begin with a simpler protocol that proves $|S| = d - 1$ exactly (that is, the simpler protocol explicitly leaks $|S|$); then we extend it to handle the general $|S| < d$ case without leaking additional information about $|S|$. Our protocol uses the following batch proof of knowledge of a subset of DLs as a subroutine.

Building block: Proof of knowledge of a subset of discrete logarithms. Suppose prover P and verifier V take as common input a collection of n pairwise distinct group elements $C_1, \ldots, C_n \in \mathbb{G}^*$ (say, the auxiliary commitments in Protocol 2). Protocol 3 implements a system for special honest-verifier batch perfect zero-knowledge arguments of knowledge in which P demonstrates knowledge of an index set $S \subseteq_{d-1} [1,n]$ and corresponding exponents $\gamma_1, \ldots, \gamma_{d-1} \in \mathbb{Z}_q^*$ such that $C_{S_j} = g^{\gamma_j}$ for each $j = 1, \ldots, d-1$. Such a proof would typically be instantiated by applying Cramer et al.'s method for proofs of partial knowledge [13] to Schnorr's proof of knowledge of a DL [30], the latter protocol being perhaps the best-known example of a zero-knowledge proof in the literature. Our Protocol 3 is similar to that instantiation, but it is more efficient. Although technically new, the protocol is (essentially) just a simplified version of a recently proposed batch proof of knowledge *and equality* of $(d-1)$-out-of-n DL pairs [19, Protocol 2], which also relies on all-but-k mercurial commitments for its soundness. We denote Protocol 3 by $\mathrm{BPK}\left\{\left(S, \gamma_1, \ldots, \gamma_{d-1}\right) : S \subseteq_{d-1} [1,n] \wedge \left(\bigwedge_{j \in [1,d-1]} C_{S_j} = g^{\gamma_j}\right)\right\}$. Prior to executing Protocol 3, V must specify a public (and possibly long-term) all-but-k reference string $\mathrm{ABK}(N)$ for some $N \geq n$ and an arbitrary λ-bit prime p. (Recall again that λ is the soundness parameter and that $\lambda \ll \lg q$.) We assume the common inputs $C_1, \ldots, C_n$ are each valid group elements; in cases where P generates the C_i (such as in our use of the protocol below), V should check that indeed $C_i \in \mathbb{G}^*$ for $i = 1, \ldots, n$. Fortunately, such group membership tests are inexpensive when $\mathbb{G}$ is an elliptic curve group, such as would likely be the case in a real-world BLACRONYM deployment.

Protocol 3 (Batched knowledge of a size-$(d-1)$ subset of DLs).

Common input: $C_1, \ldots, C_n \in \mathbb{G}^*$, a λ-bit prime p, and an all-but-k public reference string $\mathrm{ABK}(N)$ for some $N \geq n$

Prover's input: $S \subseteq_{d-1} [1,n]$ and $r_j = \log_g C_{S_j}$ for each $j \in [1, d-1]$

1. Set $S' = [1,n] \setminus S$. P chooses a challenge $c_i \in_{\mathrm{R}} [0, p-1]$ for each $i \in S'$, and then it computes the all-but-k commitment $\mathscr{C} \leftarrow \text{ABK-Commit}\left(S', (c_i)_{i \in S'}\right)$ and sends $\mathscr{C}$ to V.
2. V picks scalars $b_1, \ldots, b_n \in_{\mathrm{R}} [0, 2^\lambda - 1]$ and sends them to P.
3. P chooses a blinding factor $r_0 \in_{\mathrm{R}} \mathbb{Z}_q^*$, and then it sets $a_i = (b_i + c_i) \bmod 2^\lambda$ for each $i \in S'$, computes $C' = g^{r_0}\left(\prod_{i \in S'} C_i^{a_i}\right)$, and P sends C' to V.
4. V picks a challenge $c \in_{\mathrm{R}} \mathbb{Z}_p$ and sends it to P.

5. P solves for the degree-$(n-d+1)$ polynomial $f \in \mathbb{Z}_p[x]$ satisfying $f(0) = c$ and $f(i) = c_i$ for each $i \in S'$, and then it sets $c_i = f(i) \bmod p$ and $a_i = (b_i + c_i) \bmod 2^\lambda$ for each $i \in S$. P computes the response $v = r_0 - \sum_{i \in S} a_i r_i \bmod q$ and the witness $w \leftarrow \text{ABK-Open}(\mathscr{C}, d, (c_i)_{i \in S})$, and sends $(f(x), v, w)$ to V.
6. V computes $c_i = f(i) \bmod p$ and $a_i = (b_i + c_i) \bmod 2^\lambda$ for each $i \in [1,n]$. V accepts if $\deg f \le n - d + 1$ with $f(0) = c$, if $C' = g^v \left(\prod_{i=1}^n C_i^{a_i}\right)$, and if $\text{ABK-Verify}(\mathscr{C}, w, d, (c_i)_{i=1}^n)$; otherwise, V rejects. ◇

In the extended version of this paper [20], we prove that Protocol 3 is a system for special honest-verifier perfect batch zero-knowledge arguments of knowledge with knowledge error at most $1/2^\lambda$. In the protocol, P sends just one element from $\mathbb{G}$ (i.e., the commitment C'), one element from $\mathbb{Z}_q$ (i.e., the response v), $n - d + 1$ coefficients[11] from $\mathbb{Z}_p$ (i.e., the nonconstant coefficients of f), and the all-but-k commitment-witness pair $(\mathscr{C}, w)$ to V; V sends n scalars from $[0, 2^\lambda - 1]$ (i.e., the short exponents b_i) and one challenge (i.e., c) from $\mathbb{Z}_p$ to P. (The standard instantiation requires P to send n elements of $\mathbb{G}$ and $2n - d + 1$ elements of $\mathbb{Z}_q$, though it only requires V to send just one element of $\mathbb{Z}_q$.) In addition to the cost of the all-but-k protocols (which we count in §4 and summarize in Table 1), P computes about

$$\text{ExpCost}_{\mathbb{G}}\big((1, 2\tau), (n - d + 1, \lambda)\big) \le 3\tau + \tfrac{1}{2}(n - d + 1)\lambda$$

multiplications in $\mathbb{G}$ and V computes about

$$\text{ExpCost}_{\mathbb{G}}\big((1, 2\tau), (n, \lambda)\big) \le 3\tau + \tfrac{1}{2} n\lambda$$

multiplications in $\mathbb{G}$. (In the standard instantiation, both P and V compute about $3n\tau$ multiplications in $\mathbb{G}$.) The following observation about Protocol 3 is crucial to our new SPK_5 construction.

Observation. *If C' is output by P in Step 3 of an accepting run of Protocol 3 with honest V, then, with probability overwhelming in λ, P knows $r_0 = v + \sum_{i \in S} a_i r_i \bmod q$ and $S' = [1,n] \setminus S$ such that $C' = g^{r_0}\left(\prod_{i \in S'} C_i^{a_i}\right)$.*

Special case: Exactly $d-1$ discrete logarithms are equal. We are now ready to present our new protocol for the special case in which the user has exactly $d - 1$ tickets on the blacklist (and is willing to reveal this fact). Let $S = \{i \in [1,n] \mid \log_{h_0} H_0 = \log_{h_i} H_i\}$ and let $S' = [1,n] \setminus S$. As in Protocol 2, the user chooses $r \in_R \mathbb{Z}_q^*$ and, for each $i \in S'$, computes the auxiliary commitment $C_i = (h_i^x / H_i)^r$; then, for each $j \in S$, the user chooses $r_j \in_R \mathbb{Z}_q^*$ and computes $C_j = g^{r_j}$. This ensures that $C_i \ne 1$ for any $i \in [1,n]$, notably for $i \in S$. The user employs Protocol 3 to prove knowledge of a size-$(d-1)$ subset S of DLs with respect to g among the C_i and a modified Protocol 2 to prove that $\log_{h_i} H_i = \log_{h_0} H_0$ only for the indices in $S' \in [1,n] \setminus S$. We leverage the observation following Protocol 3 to facilitate the latter proof; in particular, we treat the commitment $C' = g^{r_0}\left(\prod_{i \in S'} C_i^{a_i}\right)$ as a *blinded* alternative to $C = \prod_{i \in S'} C_i^{a_i}$ that hides which subset S' of indices the product of powers is taken over. Note that having $\log_{h_i} h_j$ unknown when $i \ne j$ prevents V from attempting to link an authenticating user to a ticket (h_i, H_i) from a prior session by, for example, also including $(h_j, H_j) = (h_i^s, H_i^s)$ on the blacklist for some $s \in \mathbb{Z}_q^*$ and noting that $C_i^s = C_j$ if and—with all but negligible probability in τ—only if the currently authenticating user has a secret key that is not equal to $\log_{h_i} H_i$.

[11] P sends just $n - d + 1$ coefficients from $\mathbb{Z}_p$ as the constant term $f(0) = c$ is already known to V. (Hence, having V check that $f(0) = c$ in Step 5 is redundant.)

Protocol 4 (Batched inequality of one DL with all-but-d others).

Common input: $(h_0, H_0), \ldots, (h_n, H_n) \in \mathbb{G}^* \times \mathbb{G}^*$, a λ-bit prime p, all-but-k public parameters ABK(N) for some $N \ge n$, and a d-strikes-out bound $d \in [1,n]$

Prover's input: $x = \log_{h_0} H_0$ such that $\left|\{i \in [1,n] \mid \log_{h_i} H_i = x\}\right| = d - 1$

1. Set $S = \{i \in [1,n] \mid \log_{h_i} H_i = x\}$ and $S' = [1,n] \setminus S$. P chooses blinding factors $r \in_R \mathbb{Z}_q^*$ and $r_i \in_R \mathbb{Z}_q^*$ for each $i \in S$, and then, for each $i \in [1,n]$, it computes the auxiliary commitment
$$C_i = \begin{cases} g^{r_i} & \text{if } i \in S, \text{ and} \\ (h_i^x / H_i)^r & \text{if } i \in S'. \end{cases}$$
2. P engages V in $\text{BPK}\{(S, \gamma_1, \ldots, \gamma_{d-1}) : S \subseteq_{d-1} [1,n] \wedge (\bigwedge_{j \in [1,d-1]} C_{S_j} = g^{\gamma_j})\}$, using $\gamma_i = r_{S_i}$. Referring to Protocol 3, let C' be P's output in Step 3, let $a_1, \ldots, a_n$ be as V computes them in Step 6, and let $r_0 = v + \sum_{i \in S} r_i a_i \bmod q$ using P's response v in Step 5.
3. P and V each compute $h = h_0 \prod_{i=1}^n h_i^{a_i}$ and $H = H_0 \prod_{i=1}^n H_i^{a_i}$.
4. P engages V in $\text{PK}\{(\alpha, \beta, \gamma) : C' = g^\gamma h^\alpha H^\beta\}$, using $\alpha = xr \bmod q$, $\beta = -r \bmod q$, and $\gamma = r_0 \bmod q$.
5. V accepts if $C_i \ne 1$ for each $i = 1, \ldots, n$ and if it accepts in Steps 2 and 5; otherwise, V rejects. ◇

Note that h/h_0 and H/H_0 are products of powers of the h_i and H_i, respectively, with indices ranging over $[1,n]$, whereas C' is a product of powers of g and the C_i with indices ranging only over the proper subset $S' \subset [1,n]$, which is *unknown to V*. (Since, for each $i \in S'$, C_i is also a product of powers of h_i and H_i, we have that C' is in fact a product of powers of g and the h_i and H_i with indices $i \in S'$.) When P is honest, the pairs (h_i, H_i) with indices in S are precisely those pairs for which $\log_{h_i} H_i = \log_{h_0} H_0$ so that $h_i^\alpha H_i^\beta = 1$; thus,

$$\begin{aligned} g^\gamma h^\alpha H^\beta &= g^\gamma \left(\textstyle\prod_{i=1}^n h_i^{a_i}\right)^\alpha \left(\textstyle\prod_{i=1}^n H_i^{a_i}\right)^\beta \\ &= g^\gamma \textstyle\prod_{i=1}^n \left(h_i^\alpha H_i^\beta\right)^{a_i} \\ &= g^\gamma \textstyle\prod_{i \in S} \left(h_i^\alpha H_i^\beta\right)^{a_i} \prod_{i \in S'} \left(h_i^\alpha H_i^\beta\right)^{a_i} \\ &= g^\gamma \textstyle\prod_{i \in S'} \left(h_i^\alpha H_i^\beta\right)^{a_i} \\ &= g^\gamma \textstyle\prod_{i \in S'} C_i^{a_i} \\ &= C'. \end{aligned}$$

But if P behaves dishonestly (by using a different S and S'), V will detect this: if P omits an index i from S for which $\log_{h_i} H_i = \log_{h_0} H_0$, then $C_i = 1$ and V will reject in Step 6; if it instead includes an extra index i in S for which $\log_{h_i} H_i \ne \log_{h_0} H_0$, then the above cancellation will fail and this time V will reject in Step 5. Now, since $\log_g h_i$ is unknown (by assumption) for each $i \in [1,n]$ and since Step 2 proves that, with overwhelming probability in λ, P knows at least $d - 1$ DLs among the C_i with respect to base g, V is convinced that there exists some size-$(d-1)$ subset S of indices i such that $C_i \ne h_i^\alpha H_i^\beta$ for any $i \in S$; furthermore, Step 5 convinces V that $1 = h_0^\alpha H_0^\beta$ and that $C_i = h_i^\alpha H_i^\beta$ for each index $i \in [1,n] \setminus S$. Therefore, V is convinced that *exactly* $d - 1$ pairs (h_i, H_i) have $\log_{h_i} H_i = \log_{h_0} H_0$. We prove in the extended version of this paper [20] that Protocol 4 implements a system for special honest-verifier computational zero-knowledge arguments of knowledge with knowledge error at most $1 / 2^\lambda$.

Regarding efficiency, we see by inspection that, in addition to what it sends in the subprotocol in Step 2 (that is, in addition to

executing Protocol 3), P sends just one element from $\mathbb{G}$ (i.e., a commitment in Step 5) and three responses from $\mathbb{Z}_q$ (i.e., the responses in Step 5); V, likewise, sends just one additional challenge from $\mathbb{Z}_q$ (i.e., the challenge in Step 5). Computationally, each of P and V in Protocol 4 must compute

$$2 \cdot \mathrm{ExpCost}_{\mathbb{G}}((n,\lambda)) + 2 + \mathrm{ExpCost}_{\mathbb{G}}(3,2\tau) \leq 5\tau + (n+2)\lambda + 2$$

multiplications in $\mathbb{G}$ beyond what they each compute in Protocol 3 (not including P's *precomputable* computation of $C_1, \ldots, C_n$).

Batch protocol for d-BLAC.

Extending Protocol 4 to the general $|S| < d$ case is simple: the user forms a "new" problem instance by sending $d-1$ "ephemeral" pairs $(h_{n+1}, H_{n+1}), \ldots, (h_{n+d-1}, H_{n+d-1})$ and corresponding C_{n+1}, $\ldots, C_{n+d-1}$ such that exactly $d - |S| - 1$ of the ephemeral pairs satisfy $\log_{h_i} H_i = \log_{h_0} H_0$. This ensures that *exactly* $d-1$ pairs in the "augmented" problem instance satisfy $\log_{h_i} H_i = \log_{h_0} H_0$, regardless of the value of $|S|$, so long as it is less than d. From here, the user and the SP follow the protocol exactly as above, thus proving that *at most* $d-1$ of the (h_i, H_i) pairs on the original list satisfy $\log_{h_i} H_i = \log_{h_0} H_0$. For soundness, we also need to ensure that the user knows $\log_{h_{n+i}} H_{n+i}$ for each $i = 1, \ldots, d-1$ so that the user knows the DL of $h_{n+i}^{\gamma_i} / H_i$ with respect to base h_{n+i} and, therefore, not with respect to base g for each $i = 1, \ldots, d-1$. To accomplish this efficiently, the user can compute each h_{n+i} as a random power of h_0 and then invoke the protocol denoted by $\mathrm{BPK}\{(\alpha_1, \beta_1, \ldots, \alpha_{d-1}, \beta_{d-1}) : \bigwedge_{i=1}^{d-1} \alpha_i = \log_{h_0} h_{n+i} \wedge \beta_i = \log_{h_0} H_{n+i}\}$. This increases the communication cost and computation cost of the protocol only slightly if we assume that d is constant (and small relative to n), as one would expect it to be in practice.

Comparison with d-BLAC.

In d-BLAC's default SPK_5 instantiation, the user sends about $3n$ elements from $\mathbb{G}$ and $2n$ elements from $\mathbb{Z}_q$ to the SP; thus, our optimizations again reduce the communication cost by about 75% without precomputation, assuming that the SP's challenges are output by a cryptographically secure hash function. The computational savings are also similar: the user and the SP compute, respectively, about $(4\tau + \lambda) \,/\, 12\tau$ and $\lambda \,/\, 6\tau$ times as many multiplications in $\mathbb{G}$, as was the case before for vanilla BLAC. Similar remarks as before apply with regards to point compression and with regards to precomputing and reblinding the auxiliary commitments C_i. When precomputation is used, the user's online computation cost is similar to the SP's computation cost.

3.4 BLACRONYM with reputation

The third and final black box that we redesign comes from the authentication protocol in BLACR; that is, we provide an alternative instantiation for a simplified version of the protocol that Au et al. label $\mathfrak{S}_{\text{WS-Adj}}$ [2, §4.3]. (Our protocol is "simplified" in that it only deals with the "unweighted" version of BLACR. In the weighted version, SPs can specify *adjusting factors* to weight scores differently depending on how many times the user has engaged in a particular kind of (either positive or negative) behaviour. It is not immediately clear that such functionality is even *possible* within the framework we use for our batch protocols; we therefore leave further investigation along those lines to future work.) The $\mathfrak{S}_{\text{WS-Adj}}$ protocol plays an analogous role to that of SPK_5 in vanilla BLAC and d-BLAC: after the user proves knowledge of x such that (i) the user holds a valid credential $C(x)$ encoding x and (ii) the second component in the user's ticket $\Gamma = (z_0, H_0)$ has the form $H_0 = \mathcal{H}(z_0 \| s)^x$ for that same x and the SP's canonical name s, it outputs a Pedersen commitment $D = \hat{g}^{\varsigma(x)} g^{\sigma}$, where $\hat{g}$ is a generator of $\mathbb{G}$ with $\log_g \hat{g}$ unknown and where $\varsigma(x)$ denotes the user's aggregate score on the SP's blacklist. The user then engages in $\mathfrak{S}_{\text{WS-Adj}}$ with the SP to prove that the aggregate score $\varsigma(x)$ committed to by D is the correct aggregate score for its secret key x. Note that the correct aggregate score is $\varsigma(x) = \sum_{i \in S} \varsigma_i$ for $S = \{i \in [1,n] \mid \log_{h_i} H_i = x\}$, where ς_i is the score associated with ticket $\Gamma_i = (z_i, H_i)$ in $\mathcal{B}$; thus, P can prove the correctness of D by (i) outputting a Pedersen commitment D_i for each $i = 1, \ldots, n$ that commits to ς_i if $i \in S$ and to 0 if $i \in [1,n] \setminus S$, (ii) proving that each such D_i commits to the correct value, and then (iii) using $D = \prod_{i=1}^{n} D_i$ as the commitment to $\varsigma(x)$.

An alternative instantiation for $\mathfrak{S}_{\text{WS-Adj}}$ in BLACR.

Building block: Proof of knowledge of one DL in each of several pairs. Suppose that prover P and verifier V take as common input a collection of n pairs of group elements $(C_1, D_1), \ldots, (C_n, D_n) \in \mathbb{G}^* \times \mathbb{G}^*$ with the goal being for P to prove knowledge of exponents $\gamma_1, \ldots, \gamma_n \in \mathbb{Z}_q^*$ such that, for each $i = 1, \ldots, n$, either $C_i = g^{\gamma_i}$ or $D_i = g^{\gamma_i}$. A direct analogue of Protocol 3 implements a system for special honest-verifier batch perfect zero-knowledge proofs of knowledge for this claim. In particular, P engages V in two parallel instances of a relaxed Protocol 3: in one instance, P proves knowledge of an (arbitrary sized) index set $S \subseteq [1,n]$ and to $r_i = \log_g C_i$ for each $i \in S$; in the other instance, P proves knowledge of an (also arbitrary sized) index sets $S' \subseteq [1,n]$ and to $r_i = \log_g D_i$ for each $i \in S'$. (Thus, V does not require the challenges within each parallel instance to satisfy any particular constraints; in fact, it must be possible for P to make V accept even if $S = \emptyset$ or $S' = \emptyset$.) Of course, to prove the desired claim P must convince V that $S' = [1,n] \setminus S$, which we accomplish by having P all-but-k commit to n components of a length-$2n$ vector of challenges $\langle c_1, \ldots, c_n, d_1, \ldots, d_n \rangle$, where c_i will be the challenge for the claim $C_i = g^{r_i}$ and d_i will be the challenge for the claim $D_i = g^{r_i}$. For each $i = 1, \ldots, n$, V will check if $c_i + d_i \equiv c \bmod 2^{\lambda}$, where c is the verifier-chosen challenge; if so, then V is assured (with overwhelming probability in λ) that, prior to receiving c from V, P chose *at most* one of c_i or d_i for each $i \in [1,n]$. To protect against algebraic attacks by dishonest P, V insists that P uses $k = n$ in the all-but-k opening, thus proving that P in fact chose *exactly* one of c_i or d_i for each $i \in [1,n]$ and, therefore, that the other was uniquely determined by the challenge c. We denote the resulting protocol by $\mathrm{BPK}\{(S, \gamma_1, \ldots, \gamma_n) : S \subseteq [1,n] \wedge (\bigwedge_{i \in S} C_i = g^{\gamma_i}) \wedge (\bigwedge_{i \in [1,n] \setminus S} D_i = g^{\gamma_i})\}$. Similar to with Protocol 3, if honest V accepts in the above proof, then, with probability overwhelming in λ, P must know $s_0 = v_0 + \sum_{i \in S} a_i \gamma_i \bmod q$ and $S' = [1,n] \setminus S$ such that $C' = g^{s_0} \prod_{i \in S'} C_i^{a_i}$, where C' is the commitment output by P to prove knowledge of $\gamma_i = \log_g C_i$ for each $i \in S$. (The only difference here from the observation following Protocol 3 is that the product is over a set S' of *unknown size*, which turns out to be inconsequential in our use of the observation.)

Batch protocol for BLACR.

Given the above-described variant of Protocol 3, we are now ready to present our protocol for proving the correctness of the aggregate score $\varsigma(x)$ committed to by $D = \hat{g}^{\varsigma(x)} g^{\sigma}$. The user and the SP in this subprotocol have common inputs two generators $g, \hat{g} \in \mathbb{G}^*$, a commitment $D \in \mathbb{G}^*$ and pair $(h_0, H_0) \in \mathbb{G}^* \times \mathbb{G}^*$, and a set of n triples $(h_1, H_1, \varsigma_1), \ldots, (h_n, H_n, \varsigma_n) \in \mathbb{G}^* \times \mathbb{G}^* \times \mathbb{Z}$ such that $\log_g \hat{g}$, $\log_g h_i$, and $\log_g h_i$ for $i \in [1,n]$, and $\log_{h_i} h_j$ for $i \neq j$ are all unknown. The goal is for the user to prove knowledge of $S \subseteq [1,n]$ and $\sigma \in \mathbb{Z}_q$ such that $S = \{i \in [1,n] \mid \log_{h_0} H_0 = \log_{h_i} H_i\}$ and $D = \hat{g}^{\sum_{i \in S} \varsigma_i} g^{\sigma}$.

As in Protocol 4, the user computes auxiliary commitments C_i as $C_i = (h_i^x \,/\, H_i)^r$ for $i \in S'$ and $C_i = g^{r_i}$ for $i \in S$, where

Legend n = blacklist size d = strikes-out bound m = number of user's tickets on blacklist τ = security parameter λ = soundness parameter	User computation [a] (multiplications in $\mathbb{G}$)		SP computation (multiplications in $\mathbb{G}$)		Proof size [b] (bits)	
	ORIGINAL	BLACRONYM	ORIGINAL	BLACRONYM	ORIGINAL	BLACRONYM
BLAC	$8n\tau$	$n\lambda$	$9n\tau$	$3n\lambda$ / 2	$3n\|\mathbb{G}\| + 2n\tau$	$n\|\mathbb{G}\|$
***d*-BLAC**	$(8n + d)\tau$	$d\tau + (n + d$ / $2)\lambda$	$9n\tau$	$4n\lambda$ / 2	$3n\|\mathbb{G}\| + 3n\tau$	$n\|\mathbb{G}\|$
BLACR	$(11n + 9m)\tau$	$n\tau + 2n\lambda$	$16n\tau$	$5n\lambda$ / 2	$5n\|\mathbb{G}\| + 8n\tau$	$2n\|\mathbb{G}\|$

Example: $\tau = 128$, $\lambda = 40$, and $d = m = 6$.			Savings			Savings			Savings
BLAC	$1024n$	$40n$	$25x$	$1152n$	$60n$	$19x$	$640n$	$128n$	$5x$
***d*-BLAC**	$1024n + 768$	$40n + 888$	$25x$	$1152n$	$80n$	$14x$	$768n$	$128n$	$6x$
BLACR	$1408n + 5760$	$208n$	$7x$	$2048n$	$100n$	$20x$	$1664n$	$256n$	$7x$

Table 1: Approximate computation and communication costs, including overhead from all-but-k commitments. Cost comparison between BLACRONYM protocols and their original BLAC, d-BLAC, and BLACR counterparts. The '**User computation**' column lists the approximate number of multiplications in $\mathbb{G}$ for the user to compute *not including the auxiliary commitments $C_1, \ldots, C_n$ or precomputable all-but-k commitment values*. The '**Proof size**' column lists the approximate transcript size (in bits) for the noninteractive form of the protocol, including $n|\mathbb{G}|$ bits for the auxiliary commitments $C_1, \ldots, C_n$. (In practice, $|\mathbb{G}| \approx 2\tau$ or 4τ, depending on whether or not point compression is used.) The second part of the table fixes $\tau = 128$, $\lambda = 40$, and $d = 6$ and then lists the costs for these parameters as a function of n. Many small additive terms have been omitted from the reported costs for clarity of presentation; their contribution to the actual cost is insignificant even for n as small as 10 or 20.

[a] User computation cost excludes costs for precomputable values such as $C_1, \ldots, C_n$ and the precomputable portions of the all-but-k commitments.
[b] Proof size includes $C_1, \ldots, C_n$ and assumes all challenge bits are output by a random oracle.

$r \in_R \mathbb{Z}_q^*$ and $r_i \in_R \mathbb{Z}_q^*$ for all $i \in S$. The user also computes Pedersen commitments $D_i = \hat{g}^{\varsigma_i} g^{t_i}$ for $i \in S$ and $D_i = g^{r_i}$ for $i \in S'$, where $t_i \in_R \mathbb{Z}_q^*$ for all $i \in [1, n]$. Let $D_i' = D_i \,/\, \hat{g}^{\varsigma_i}$ and note that if the user computes the C_i and D_i honestly, then (i) P knows $r_i = \log_g C_i$ and $t_i = \log_g D_i'$ (but not $\log_g D_i$) whenever $i \in S$, (ii) P knows $t_i = \log_g D_i$ (but not $\log_g C_i$ or $\log_g D_i'$) whenever $i \in S'$, and (iii) $\prod_{i=1}^n D_i = \hat{g}^{\varsigma(x)} g^{\sum_{i=1}^n t_i}$. In particular, if P proves that it indeed knows $\log_g D_i'$ if and only if it knows $\log_g C_i$, and that it knows $\log_g C_i$ if and only if $x = \log_{h_i} H_i$, then it follows that $D = \prod_{i=1}^n D_i$ is a Pedersen commitment to (honestly computed) $\varsigma(x)$. The proof of these claims is very similar to Protocol 4, but with Protocol 3 replaced by its above-described variant. The full details of this protocol are in the extended version of our paper [20].

4. COST COMPARISON

Table 1 compares the (online) computation cost and proof size of the original protocols for BLAC, d-BLAC, and BLACR with those of the corresponding BLACRONYM protocols, *including* the non-precomputable costs of the all-but-k protocols. The BLACRONYM protocols handily outperform the original non-batch protocol, usually by more than an order of magnitude for reasonable parameter choices. The computation costs listed in the table for d-BLAC and BLACR count the computation overhead from computing and verifying an all-but-k commitment. For d-BLAC, the commitment is to all-but-$(d-1)$ elements of a length-n sequence: the user computes an expected $4(d-1)\tau + \frac{n-d+1}{2}\lambda$ multiplications in $\tilde{\mathbb{G}}$ and the SP an computes expected $\frac{n}{2}\lambda$ multiplications in $\tilde{\mathbb{G}}$. (The user can precompute all but about $2(d-1)\tau$ of the multiplications.) For BLACR, the commitment is to all-but-n elements of a length-$2n$ sequence: the user computes an expected $4n\tau + \frac{n}{2}\lambda$ multiplications in $\tilde{\mathbb{G}}$ and the SP computes an expected $n\lambda$ multiplications in $\tilde{\mathbb{G}}$. (The user can precompute all but $n\tau + \frac{n}{2}\lambda$ multiplications.) The size of the proof is constant-size: $g|\tilde{\mathbb{G}}| + |\mathbb{G}_T| + 6\tau$, where $|\tilde{\mathbb{G}}|$ and $|\mathbb{G}_T|$ respectively denote the number of bits used to represent elements of the base group $\tilde{\mathbb{G}}$ and target group $\mathbb{G}_T$ of the bilinear pairing in ABK(N).

5. CONCLUSION

We presented the BLACRONYM suite of efficient protocols for anonymous blacklisting without trusted third parties. Our protocols improve on BLAC and its variants by providing comparable functionality and security guarantees with substantially lower communication and computation overhead. To accomplish this, we combine existing batch zero-knowledge techniques with a new technique for constructing batch proofs of partial knowledge about DLs over *non-monotone access structures*. We expect this latter technique will be useful in constructing zero-knowledge protocols for other statements of interest.

Acknowledgements. We thank the anonymous reviewers for their feedback. The first author is supported by a GO-Bell Graduate Scholarship and by the Natural Sciences and Engineering Research Council of Canada (NSERC) through a Vanier Canada Graduate Scholarship. We also thank the NSERC Discovery Grants program and The Tor Project, Inc. for funding this research.

References

[1] M. H. Au and A. Kapadia. PERM: Practical reputation-based blacklisting without TTPs. In *Proceedings of CCS 2012*, pages 929–940, Raleigh, NC, USA, October 2012.

[2] M. H. Au, A. Kapadia, and W. Susilo. BLACR: TTP-free blacklistable anonymous credentials with reputation. In *Proceedings of NDSS 2012*, San Diego, CA, USA, February 2012.

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

[4] M. H. Au, P. P. Tsang, and A. Kapadia. PEREA: Practical TTP-free revocation of repeatedly misbehaving anonymous users. *ACM Transactions on Information and System Security*, 14(4):29, December 2011.

[5] M. Bellare, J. A. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Proceedings of EUROCRYPT 1998*, volume 1403 of *LNCS*, pages 236–250, Espoo, Finland, June 1998.

[6] D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

[7] J. N. Bos and M. J. Coster. Addition chain heuristics. In *Proceedings of CRYPTO 1989*, volume 435 of *LNCS*, pages 400–407, Santa Barbara, CA, USA, August 1989.

[8] E. Brickell and J. Li. Enhanced Privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 9(3):345–360, May 2012.

[9] E. F. Brickell, D. M. Gordon, K. S. McCurley, and D. B. Wilson. Fast exponentiation with precomputation (extended abstract). In *Proceedings of EUROCRYPT 1992*, volume 658 of *LNCS*, pages 200–207, Balatonfüred, Hungary, May 1992.

[10] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Proceedings of CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144, Santa Barbara, CA, USA, August 2003.

[11] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In *Proceedings of CRYPTO 1997*, volume 1294 of *LNCS*, pages 410–424, Santa Barbara, CA, USA, August 1997.

[12] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Proceedings of CRYPTO 1992*, volume 740 of *LNCS*, pages 89–105, Santa Barbara, CA, USA, August 1992.

[13] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of CRYPTO 1994*, volume 839 of *LNCS*, pages 174–187, Santa Barbara, CA, USA, August 1994.

[14] I. Damgård. On Σ-protocols. Technical report, Aarhus Universitet, March 2011. CPT 2011; Available from `http://www.daimi.au.dk/~ivan/Sigma.pdf`.

[15] J. R. Douceur. The Sybil attack. In *Proceedings of IPTPS 2002*, volume 2429 of *LNCS*, pages 251–260, Cambridge, MA, USA, March 2002.

[16] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings of CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, CA, USA, August 1986.

[17] R. Henry and I. Goldberg. Formalizing anonymous blacklisting systems. In *Proceedings of IEEE S&P 2011*, pages 81–95, Berkeley, CA, USA, May 2011.

[18] R. Henry and I. Goldberg. All-but-k mercurial commitments and their applications. Technical Report CACR 2012-26, University of Waterloo, Waterloo, ON, Canada, December 2012. Available from `http://cacr.uwaterloo.ca/techreports/2012/cacr2012-26.pdf`.

[19] R. Henry and I. Goldberg. Batch proofs of partial knowledge. In *Proceedings of ACNS 2012*, volume 7954 of *LNCS*, pages 502–517, Banff, AB, Canada, June 2013.

[20] R. Henry and I. Goldberg. Thinking inside the BLAC box: Smarter protocols for faster anonymous blacklisting (extended version). Technical Report CACR 2013-26, University of Waterloo, Waterloo, ON, Canada, August 2013. Available from `http://cacr.uwaterloo.ca/techreports/2013/cacr2013-26.pdf`.

[21] R. Henry, K. Henry, and I. Goldberg. Making a Nymbler Nymble using VERBS. In *Proceedings of PETS 2010*, volume 6205 of *LNCS*, pages 111–129, Berlin, Germany, July 2010.

[22] T. Icart. How to hash into elliptic curves. In *Proceedings of CRYPTO 2009*, volume 5677 of *LNCS*, pages 303–316, Santa Barbara, CA, USA, August 2009.

[23] P. C. Johnson, A. Kapadia, P. P. Tsang, and S. W. Smith. Nymble: Anonymous IP-address blocking. In *Proceedings of PETS 2007*, volume 4776 of *LNCS*, pages 113–133, Ottawa, ON, Canada, June 2007.

[24] C. H. Lim. Efficient multi-exponentiation and application to batch verification of digital signatures. Technical report, Sejong University, August 2000. Available from `http://dasan.sejong.ac.kr/~chlim/pub/multi_exp.ps`.

[25] C. H. Lim and P. J. Lee. More flexible exponentiation with precomputation. In *Proceedings of CRYPTO 1994*, volume 839 of *LNCS*, pages 95–107, Santa Barbara, CA, USA, August 1994.

[26] Z. Lin and N. Hopper. Jack: Scalable accumulator-based nymble system. In *Proceedings of WPES 2010*, pages 53–62, Chicago, IL, USA, October 2010.

[27] P. Lofgren and N. Hopper. BNymble: More anonymous blacklisting at almost no cost (a short paper). In *Proceedings of FC 2011*, volume 7035 of *LNCS*, pages 268–275, Gros Islet, St. Lucia, February 2011.

[28] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, USA, October 1996. Fifth Printing, August 2001.

[29] J. Sauerbrey and A. Dietel. Resource requirements for the application of addition chains in modulo exponentiation. In *Proceedings of EUROCRYPT 1992*, volume 658 of *LNCS*, pages 174–182, Balatonfüred, Hungary, May 1992.

[30] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *Proceedings of CRYPTO 1989*, volume 435 of *LNCS*, pages 239–252, Santa Barbara, CA, USA, August 1989.

[31] E. J. Schwartz, D. Brumley, and J. M. McCune. Contractual anonymity. In *Proceedings of NDSS 2010*, San Diego, CA, USA, February 2010.

[32] The Tor Project. List of IRC/chat networks that block or support Tor. `https://trac.torproject.org/projects/tor/wiki/doc/BlockingIrc` (Retrieved: 2013-04-08).

[33] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. Blacklistable Anonymous Credentials: Blocking misbehaving users without TTPs. In *Proceedings of CCS 2007*, pages 72–81, Alexandria, VA, USA, October 2007.

[34] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. BLAC: Revoking repeatedly misbehaving anonymous users without relying on TTPs. *ACM Transactions on Information and System Security (TISSEC)*, 13(4):39:1–39:33, December 2010.

APPENDIX

A. SECURITY & PRIVACY PROPERTIES OF A SECURE BLAC CONSTRUCTION

We very briefly discuss the security and privacy properties that a *secure BLAC construction* must provide. The definitions we give here are informal; we refer the reader to Tsang et al. [34] for the formal versions. We note that such informal definitions suffice for our purposes, since we only modify the internals of black boxes. In particular, the security proofs provided by Tsang et al. [34, §7.2] for BLAC and d-BLAC and by Au, Kapadia, and Susilo [2, Appendix A] for BLACR will also prove that BLACRONYM is secure, provided we prove that our zero-knowledge protocols securely implement the black boxes we modify. (Note that we do make one minor change to the original definitions: we insist that the probability with which dishonest users can fool an honest SP in the authentication protocol is negligible *in the SP's soundness parameter* λ, rather than in the system-wide security parameter τ.)

1. *Correctness:* If the GM and a given SP are both honest, and if a given user's entries on that SP's blacklist do not meet its revocation criteria, then with probability overwhelming in the security parameter τ, the user can successfully authenticate to the SP.
2. *Misauthentication resistance:* If a user successfully authenticates to an honest SP, then with probability overwhelming in the SP's soundness parameter λ, that user possesses a valid credential $C(x)$ from the GM.
3. *Blacklistability:* A coalition of dishonest SPs and users holding secret keys $x_1, \ldots, x_k$ can successfully authenticate to an honest SP with blacklist $\mathcal{B}$ only if $\rho_s(\mathcal{B}, x_i) = 0$ for some $i \in [1, k]$, except with probability negligible in λ.
4. *Anonymity:* No coalition of dishonest SPs, users, and the GM has a computational advantage in distinguishing authentication transcripts associated with the same honest user from those associated with different honest users. Moreover, no such coalition has a computational advantage in linking any authentication transcript with the real-world identity of the honest user that produced it.[12]
5. *Non-frameability:* No coalition of dishonest SPs, users, and the GM can prevent an honest user from successfully authenticating with an honest SP, except with probability negligible in τ.

[12]We speak of *computational advantage* here to emphasize that coalitions may do better than random guessing if they utilize side channel information (e.g., timestamps, destination SPs, contents of posts, etc.). The usual formalization of this idea is to require that the coalition's guess distribution *after* seeing some authentication transcripts from honest users should be close to its guess distribution *before* seeing those transcripts.