Jack: Scalable Accumulator-based Nymble System

Zi Lin
Computer Science & Engineering
University of Minnesota
Minneapolis, MN 55455
lin@cs.umn.edu

Nicholas Hopper
Computer Science & Engineering
University of Minnesota
Minneapolis, MN 55455
hopper@cs.umn.edu

ABSTRACT

Anonymous blacklisting schemes enable online service providers to block future accesses from abusive users behind anonymizing networks, such as Tor, while preserving the privacy of all users, both abusive and non-abusive. Several such schemes exist in the literature, but all suffer from one of several faults: they rely on trusted parties that can collude to de-anonymize users, they scale poorly with the number of blacklisted users, or they place a very high computational load on the trusted parties.

We introduce Jack, an efficient, scalable anonymous blacklisting scheme based on cryptographic accumulators. Compared to the previous efficient schemes, Jack significantly reduces the communication and computation costs required of trusted parties while also weakening the trust placed in these parties. Compared with schemes with no trusted parties, Jack enjoys constant scaling with respect to the number of blacklisted users, imposing dramatically reduced computation and communication costs for service providers. Jack is provably secure in the random oracle model, and we demonstrate its efficiency both analytically and experimentally.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection; E.3 [**Data Encryption**]: Public key cryptosystems

General Terms

Algorithms, Security

Keywords

Anonymous Authentication, Anonymous Blacklisting, Privacy-Enhancing Revocation

1. INTRODUCTION

Network anonymity providers enable users to connect to online services while concealing which users communicate with which servers. These providers typically achieve this goal by having one or more hosts act as a proxy that relays traffic between clients and servers, using layered encryption and multiple hops to conceal the correlation between incoming and outgoing traffic. For example, the most popular anonymity provider, Tor [16], is comprised of over 1700 volunteer "onion routers" from around the world, and provides service to roughly 200,000 users at any time [1].

From the point of view of service providers such as website operators – especially those that rely on contributions from users – anonymity services pose an unfortunate dilemma: anonymity can extend the range of users who are able to access and willing to contribute to the service, since they can do so without fear of monitoring, isolation or punishment. However, the ability to access a service anonymously can also be abused by those who wish to vandalize or otherwise behave inappropriately when accessing the service. In several cases, notably Wikipedia and Slashdot, service providers have opted in favor of abuse prevention and proactively block contributions from known anonymity providers, at the expense of potentially valuable contributions.

To address this problem, Johnson *et al.* [21] proposed the notion of an *anonymous blacklisting scheme*, which allows service providers (SPs) to maintain a "blacklist" that allows non-abusive users to access the service anonymously; users on the blacklist are unable to access the service, but remain anonymous. If deployed, an anonymous blacklisting scheme would allow the SP to enjoy the benefits of anonymity – increased participation – while limiting abuse. Widespread adoption of the schemes would also benefit anonymity providers by reducing the "collateral damage" these users incur from the actions of others. (For example, Tor exit node operators are currently banned from contributing to Wikipedia as a result of the behavior of abusive Tor users) Thus deployment of anonymous blacklisting schemes would be beneficial to SPs, anonymity providers, and users alike.

Several anonymous blacklisting schemes have appeared in the literature [21, 31, 30, 27, 28, 29, 9, 20]. All of them are based on the notion of an anonymous authentication token. Nymble [21, 31, 30] constructs a hash-chain based unlinkable token scheme to maintain user anonymity while a pair of Trusted Third Parties (TTPs) – the Nymble Manager (NM) and the Pseudonym Manager (PM) – help the SP to link future tokens of abusive users to block them. However, the TTPs in Nymble can easily collude together to de-anonymize all users, and the SP and NM can collude to link the actions of a single user. Such a strong trust assumption is undesirable in an anonymity scheme.

Following the proposal of Nymble, several schemes that trade efficiency and scalability for a weaker trust model have been introduced, including BLAC [27, 28], PEREA [29], and Nymbler [20]. At one extreme, BLAC [27, 28] and EPID [9] support anonymous blacklisting of misbehaved users with no TTP at all, by directly adding authentication tokens to a blacklist. The trade-off here is that when a user produces a new authentication token t, she must prove, for each token b in the blacklist, that t is not linked to b. As a result, the communication and computation cost of authentication scale linearly with the size of the blacklist. PEREA [29] improved the scalability of BLAC's authentication by introducing a universal dynamic accumulator for the blacklist, and having each user maintain an unforgeable queue of its last k tokens; upon authentication, the user proves that none of its k tokens are in the accumulator and then updates the queue with the aid of the SP. This scheme introduces a potential "race condition" in which the SP must catch an abusive user "in time" before the token is flushed out of the queue. Larger values of k reduce the likelihood of the the condition but increase the cost of the authentication. Nymbler [20] uses a pair of TTPs like Nymble, while preventing colluding TTPs from de-anonymizing users. However the blacklisting procedure in Nymbler requires the NM to compute discrete logarithms in a trapdoor group, making real-time blacklisting unscalable.

1.1 Our Contribution

Jack be nimble, Jack be quick...

In this paper, we introduce Jack, an anonymous blacklisting scheme that is secure against TTP collusion and provides improved scalability compared to previous schemes. Similar to the original Nymble scheme, the architecture of Jack separates user identities from online activities by introducing two TTPs. However, the TTPs in Jack are engaged in fewer interactions with the user, are unable to de-anonymize users, and cannot link user activities across SPs or "linkability windows." All interactions in our scheme are communication- and computation- efficient for all parties, and scale independently of the size of the blacklist, number of users, and number of authentication periods. Jack also supports several useful extensions, including objective blacklisting, anonymous rate-limiting, and fine-grained blacklist duration management.

1.2 Paper Outline

We briefly discuss related work in section 2. We give a high-level overview on our system in section 3. We further describe the cryptographic building blocks in section 4, followed by a detailed description of our protocols in section 5. In section 6 we present both analytical and experimental evaluations of our system, followed by a discussion of several practical extensions to our system in section 7.

2. RELATED WORK

Group Signatures Group signature schemes allow any member of a group to sign a message on behalf of the group, while concealing the identity of the signer [14, 3, 7]. By design, group signature schemes have the ability to revoke the anonymity of misbehaved users. They feature a trusted third party, called the Group Manager (GM), with the ability to trace the identity of a signer from a transcript and revoke the signer from the group. This revocation can be very expensive because the updated group public key must be broadcast to all SPs and all other members of the group. Instead of global membership revocation, many systems suggest the SP keep a local blacklist [11, 8, 22, 24]. In such schemes, the GM gives an SP a user pseudonym or linking token extracted from a transcript. The offending user is then revoked only at that SP, making revocation more efficient.

Because the GM assigns each user a pseudonym when the user registers, the GM in the above schemes has all the information required to link users with their provider accesses. Further, SPs with a local blacklist can link users with their misdoings, breaking user anonymity. Such powerful TTPs represent a significant security threat to anonymity schemes.

Anonymous Blacklist Systems Nymble, along with its enhanced versions [21, 31, 30], was proposed to prevent GMs from linking tokens to a pseudonym and SPs from linking user accesses. Nymble introduces the concept of linkability window. SPs no longer keep linking information about user pseudonyms. Instead they are given a linking token that is only valid within the current linkability window. A TTP called the Nymble Manager (NM) is in charge of issuing unlinkable access tokens, called "nymbles," to users. The NM assists SPs in discovering the linking token encrypted in each nymble to unmask misbehaving users. However, the NM retains the ability to link user activities.

Being aware of this, the authors of Nymble proposed BLAC [27, 28] to eliminate the NM. BLAC enables SPs to blacklist a user without assistance from a TTP, but the communication complexity grows linearly with the size of the blacklist. EPID [9] is similar to BLAC, except EPID originated from the perspective of TPM authentication and blacklisting. In an effort to reduce the complexity of BLAC and EPID, PEREA [29] adopted an accumulator to represent the blacklist. In PEREA, users produce one-time random authentication tokens and must prove that none of the last K authentication tokens is revoked. The communication and computational complexity of PEREA is linear in a fixed constant K. However, larger values of K, while favorable, make PEREA considerably slower [29].

Recently, Nymbler was proposed by Henry *et al.* [20] to improve Nymble using Nymble's existing framework. In Nymbler, users construct nymbles based on user credentials. Nymbler makes use of Verifier-Efficient Restricted Blind Signatures (VERBS) which allow the NM to sign a nymble without really seeing it. The NM is able to recover the linking token embedded in each nymble, but linking tokens are linkable neither to user credentials nor to linking tokens from past linkability windows. Therefore, security is strengthened since the NM has no way to keep track of user activity. However, Nymbler still requires heavy computation from the NM which leads to system scalability concerns.

3. OVERVIEW OF OUR APPROACH

In this section, we briefly give a high-level overview of Jack. Further details are discussed in section 4 and section 5.

We model Jack after the original Nymble system. Jack's fundamental architecture is shown in Figure 1. Jack builds entirely upon a public key infrastructure that binds each user to a public/secret key pair. In our scheme, there are two TTPs, the Credential Manager (CM) and the Nymble Manager (NM), which run independently.

Our CM replaces the Pseudonym Manager (PM) from pre-

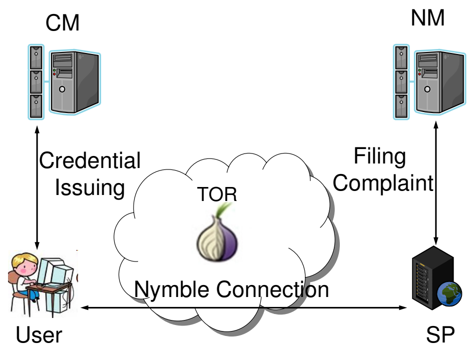

Figure 1: The architecture of Jack

vious schemes. In Jack, a user Alice requests a credential derived from her public key from the CM. The CM acts like a group manager and only enrolls legitimate users by issuing them credentials, ensuring each user is only issued a single credential.

The NM in Jack has a different role than in the original Nymble system. Instead of issuing nymble tickets to Alice, the NM is only concerned with the nymble opening procedure, allowing discovery of the underlying user pseudonym.

Upon receiving a credential, Alice constructs her nymbles based on her pseudonym with respect to SP Bob. Alice's pseudonym is determined by her own secret key and Bob's public pseudonym generator. In each nymble, Alice encloses a blinded version of her pseudonym and encrypts the blinding trapdoor by the public encryption key of the NM. Before Alice initiates a connection to Bob, Alice obtains the latest version of Bob's blacklist from a trusted source (e.g. the NM), and make sure her pseudonym is not in the blacklist. When Alice connects to Bob via an anonymizing network like Tor, Alice presents a nymble to Bob and proves its validity using zero-knowledge proofs of knowledge of discrete logarithm equalities and verifiable encryption of discrete logarithms. Bob will grant service after verifying a valid proof and will keep the nymble for later reference.

In case Alice abused Bob's service, Bob files a complaint against the related nymble to the NM. The NM will open the nymble and decrypt the underlying pseudonym. The NM returns the pseudonym to Bob, who then blacklists Alice by adding the pseudonym to a dynamic universal accumulator for Decisional Diffie-Hellman (DDH) groups.

Unlike the original Nymble, Jack's NM does not have the ability to link Alice with all of her activities. Alice is able to compute a new nymble, which is unlinkable to old ones, without registration. When Bob's pseudonym generator periodically changes, Alice's pseudonym changes unlinkably as well. Therefore, the NM and Bob together can only link Alice's online activity within one time window, preserving her forward anonymity (i.e. blacklisting Alice will not reveal Alice's past activities). Likewise, collusion between the NM and the CM does not reduce user anonymity.

Jack is also highly scalable. Since none of the protocols involved depend on any parameters that grow with the number of time periods or users, Jack's protocols enjoy constant computation and communication costs.

4. BUILDING BLOCKS

Below, we give an overview of the cryptographic primitives we use in our scheme.

4.1 Preliminary

4.1.1 Bilinear Pairing

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic multiplicative groups of prime order p. Let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Let a function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear pairing with the following properties:

- Bilinearity: $\hat{e}(P^a, Q^b) = \hat{e}(P, Q)^{ab}$ for all $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$
- Non-degeneracy: $\hat{e}(g_1, g_2) \neq e$ where e is the identity element of $\mathbb{G}_T$
- Computability: $\hat{e}(P, Q)$ can be computed in polynomial time for all $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.

4.1.2 Hardness Assumptions

The security of our scheme relies on the following two assumptions.

Strong RSA Assumption. Suppose p, q are λ-bit safe primes, that is, $(\mathsf{p} - 1)/2, (\mathsf{q} - 1)/2$ are also primes. Let $N = \mathsf{pq}$. The Strong RSA assumption states that there does not exist a probabilistic polynomial time algorithm that, given N and $v \in_R Z_N^*$, computes u and e such that $u^e \equiv v \mod N$ with non-negligible probability in λ.

DDH Assumption over a subgroup modulo p. Suppose $G_q \subset Z_p^*$ is the subgroup of kth residues modulo the ℓ-bit prime p, and $q = |G_q| = (p-1)/k$ is also a large prime. Let $g \in G_q$ be a generator of G_q. The DDH assumption over G_q states that there does not exist a probabilistic polynomial time algorithm that distinguishes between instances of the form (g, g^a, g^b, g^{ab}) from instances of the form (g, g^a, g^b, g^c), where $a, b, c \in_R Z_q$ are independently and randomly chosen, with non-negligible advantage in ℓ.

q**-Strong DH assumption** Suppose $\mathbb{G} = \langle g \rangle$ has order a large prime p. The q-strong DH assumption states that it is infeasible, given as input the $(q+1)$-tuple $(g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^q})$, to find a pair $(w, y) \in \mathbb{G} \times Z_p^*$ such that $w^{\alpha+y} = g$.

4.1.3 Proof of Knowledge

A Zero-Knowledge Proof of Knowledge (ZKPoK) [18] is an interactive protocol via which a prover convinces a verifier that the prover knows a value w satisfying some predicate, without giving the verifier any information about w. More formally, the protocol is zero-knowledge if there exists a simulator that allows the verifier to forge transcripts (without access to w) that are indistinguishable from the ZKPoK protocol's transcripts. A statistical ZKPoK is a variant of ZKPoK, in which the distribution of forged transcripts is statistically close to the distribution of actual transcripts.

Σ-protocols are a type of statistical ZKPoK protocol which can be converted into Signature Proof of Knowledge (SPK) schemes [19] by the Fiat-Shamir heuristic [17], that are secure under the Random Oracle Model [6].

Like many anonymous credential schemes, we follow the notation introduced by Camenisch and Stadler [13]. We use

$PK\{(x) : y = g^x\}$ to denote a Σ-protocol proving knowledge of $x \in Z_p$ such that $y = g^x$ for some $g, y \in \mathbb{G}$. The corresponding SPK is denoted as $SPK\{(x) : y = g^x\}(m)$ where m is an initial challenge message sent by the verifier.

4.2 Accumulator-based Blacklist

Our scheme employs a specific *dynamic universal accumulator* (DUA) scheme, first introduced by Nguyen [24], and later enhanced by Au *et al.* [4] and Damgård and Triandopoulos [15]. An accumulator scheme allows a *manager* to accumulate a set of elements into a *value* and compute *witnesses* that demonstrate that an element has not been accumulated. An accumulator is called "universal" if it supports both efficient zero-knowledge membership proofs and efficient zero-knowledge non-membership proofs. An accumulator is called "dynamic universal" if the accumulator value, membership witnesses and non-membership witnesses can be updated efficiently with newly added elements. A membership proof (or a non-membership proof) can be carried out in constant time independent of how many elements have been accumulated.

Au *et al.* [4] describe a DUA scheme for DDH groups, which has the additional property that one user, with a single secret key x, can create many unlinkable (under the DDH assumption) pseudonyms (ElGamal public keys) of the form (h, h^x) with different generators h. We can further extend the use of this property: if a DUA manager changes its own generator h to a new h', its accumulator is essentially emptied: Now any user with an accumulated pseudonyms can compute a new non-membership proof under h', and has a new pseudonym (h', h'^x), which is unlinkable to (h, h^x) as long as the discrete logarithm $\log_h(h')$ is unknown. In our scheme, SPs uses DUAs for DDH groups in a novel way to implement blacklists, accumulating user pseudonyms. Since we only require non-membership proofs for a blacklist, we will only refer to non-membership witnesses.

We present our construction of blacklists based on the DUA construction by Au *et al.* [4]. Our construction is a slight simplification of theirs due to the fact that we only require non-membership proofs. We omit the algorithms for witness computation and simplify the public key of the accumulator so that witness computations can be reduced to a series of witness updates.

Key Generation. Let λ be security parameter. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map such that $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for some λ-bit prime p. Let $g_0 \in \mathbb{G}_1$ be a generator of $\mathbb{G}_1$ and $h_0 \in \mathbb{G}_2$ be a generator of $\mathbb{G}_2$. Let $\mathbb{G}_q \subset \mathbb{Z}_p^*$ be a cyclic multiplicative group of prime order q, such that $q|(p-1)$, in which the DDH problem is intractable. All the above are inputs to the initialization procedure shown in Algorithm 1, which generates a public key and private key, (pk_{acc}, sk_{acc}) for the accumulator manager (the SP), as well as the initial value v for an empty accumulator.

Accumulating pseudonyms. Let $t \in \mathbb{G}_q$ be a pseudonym to be accumulated, and $v \in \mathbb{G}_1$ be the current accumulator value. The new accumulator value $\tilde{v}$ can be computed by the SP as $\tilde{v} = v^{s+t}$. The SP publishes $u = (t, \tilde{v})$ as the *accumulator update*, and the blacklist is updated as well: $BL := BL \cup \{t\}$.

In case there are n pseudonyms to be accumulated, the above algorithm is performed n times. Suppose $BL = \{t_1, \ldots, t_n\}$, and let $P(s, BL) = \prod_{t_i \in BL}(s + t_i)$. Then the new accumulator value will be $v = g_0^{P(s,BL)}$.

Non-membership witnesses For every pseudonym t we call the pair $w = (f, d) \in \mathbb{G}_1 \times \mathbb{Z}_p^*$ a witness for accumulator v if it satisfies the nonmembership relation

$$NonMembership(v, w, t) = \begin{cases} 1, \text{if } v = f^{(s+t)} \cdot g_0^d, \\ 0, \text{otherwise.} \end{cases}$$

The intuition behind the witness is that if $t \notin BL$, then if we interpret s as a formal variable, $(s + t) \not| P(s, BL)$. Thus when $t \notin BL$, there exists polynomial $q(s)$ and constant rem satisfying $q(s)(s + t) + rem = P(s, BL)$. It is easy to see that we can use q and rem to construct the witness (f, d), where$f = g_0^{q(s)}$ and $d = rem$. When the blacklist is initially empty, $w = (0, 1)$ is the non-membership witness for any t.

With the aid of the bilinear mapping $\hat{e}$, a user can check the validity of her witness (f, d) with the pseudonym t by verifying that $\hat{e}(v, h_0) = \hat{e}(f, \theta \cdot h_0^t)\hat{e}(g_0, h_0^d)$. Similarly, using bilinear mapping, we will see later that the witness can be used to produce a zero-knowledge proof of non-blacklisting, without revealing either t or w. The proof convinces the SP that the user is in possession of a pseudonym that is not blacklisted.

Non-membership Update. Let v and $\tilde{v}$ denote the old and the updated accumulator value respectively. Let $u = (\tilde{t}, \tilde{v})$ be an accumulator update published by the accumulator. For a non-accumulated pseudonym $t \neq \tilde{t}$, its witness $w = (f, d)$ can be updated to $\tilde{w} = (\tilde{f}, \tilde{d})$ with $\tilde{f} = vf^{\tilde{t}-t}$ and $\tilde{d} = d(\tilde{t} - t)$. It is straightforward to extend this procedure to handle multiple updates.

Zero-knowledge Proof of Non-blacklisting. User Alice with secret key $x \in \mathbb{Z}_q$ can prove her pseudonym $t = h^x$ is not accumulated in the accumulator v with the following zero-knowledge signature proof,

$$SPK\{(f, d, t, x) : f^{(s+t)} g_0^d = v \wedge d \neq 0 \wedge t = h^x\}$$

with knowledge of $w = (f, d)$. An efficient protocol for this relation is given by Au *et al.* [5].

4.3 Nymble Construction

A nymble in Jack is essentially a pseudonym encrypted under an ephemeral public key together with the verifiable encryption of the secret key under the NM's public key. The user Alice needs to prove to the SP Bob that this encryption is properly formed so that Bob can later ask the NM to decrypt Alice's pseudonym if needed.

We employ Camenisch and Shoup's Verifiable Encryption of Discrete Logarithms (VEDL) scheme [12] to construct the nymble. The construction is similar to the key escrow scenario mentioned in [12]. The only difference is here the encryption doesn't need to include a label $\mathcal{L}$ specifying the condition under which decryption can be carried out.

VEDL Key Generation. Let p, q be safe primes of ℓ bits, such that $\mathsf{p} = 2\mathsf{p}' + 1, \mathsf{q} = 2\mathsf{q}' + 1$ for some primes p' and q', and let $n = \mathsf{p} \cdot \mathsf{q}$ and $n' = \mathsf{p}' \cdot \mathsf{q}'$. Let $\mathbb{B}_{n'}$ be a subgroup of Z_n^* of order n', and $\mathcal{H} : \{0,1\}^m \times \{0,1\}^* \to \{0,1\}^\ell$ be a secure cryptographic keyed hash function where $m = m(\ell)$ is an auxiliary security parameter. These parameters are used to generate a key pair (pk_{NM}, sk_{NM}) as shown in Algorithm 2. Let $abs : Z_{n^2}^* \to Z_{n^2}^*$ map $a \in (n^2/2, n^2)$ to $n^2 - a$ and map $a \in (0, n^2/2]$ to a itself.

In addition to Bob's pseudonym generator h, Bob publishes a generator h_b of $\mathbb{G}_q$. Both h and h_b are public information to both the NM and user Alice.

Algorithm 1 *ACC-KeyGen*

```
Input: $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T, g_0 \in \mathbb{G}_1, h_0 \in \mathbb{G}_2,$
  $p, q, \mathbb{G}_q \subset Z_p^*$
Execution:
  $s \leftarrow_R \mathbb{G}_q$
  Choose a random generator $h \in_R \mathbb{G}_q$.
  $\theta = h_0^s$
  $pk_{acc} \leftarrow (h, g_0, h_0, \theta)$
  $sk_{acc} \leftarrow (s)$
  $v \leftarrow g_0$
  $BL \leftarrow \emptyset$
```

Nymble Production. A nymble consists of three components, B, C, and E, which will be computed as follows. Alice chooses randomly $r \in \mathbb{Z}_q$. Alice then computes $B = h^r$ and $C = h^x h_b^r$. Furthermore, Alice chooses a random $\mu \in_R [n/4]$, and computes $E = (u, e, v)$ as the encryption of r with the following algorithm:

$$u = g_1^{\mu},\ e = y_1^{\mu}(n+1)^r,\ \text{and}\ v = abs((y_2 y_3^{\mathcal{H}_{hk}(u,e)})^{\mu})\ .$$

The computation of E is denoted as $VerEnc(r)$. Together (B, C, E) form a nymble.

Nymble Opening. Bob sends the NM the tuple of (h, B, C, E). The NM parses (u, e, v) from E. The opening algorithm is performed as follows:

1. Check $abs(v) = v$. Otherwise return with "failure"
2. Check $u^{2(x_2+\mathcal{H}_{hk}(u,e)x_3)} = v^2$. Otherwise return with "failure"
3. Let $t = 2^{-1} \mod n$ and compute $\tilde{m} = (e/u^{x_1})^{2t}$
4. Check $n|(\tilde{m}-1)$. If so, compute $r = (\tilde{m}-1)/n$. Otherwise, return with "failure"
5. Check $(n+1)^r = \tilde{m}$. If so, return r. Otherwise, return with "failure".

Upon getting r, the NM computes $i = (h_b^r)^{-1} \mod p$. The NM computes pseudonym $h^x = C \cdot i$ if $h^r = B$. Otherwise, the NM returns "failure".

Proof of Verifiable Encryption. Alice should prove that E is an encryption of r. We rely on the following protocol described in [12]:

1. Alice chooses randomly $s \in_R [n/4]$ and computes $\mathbf{l} = \mathbf{g}^r \mathbf{h}^s$. Alice sends $\mathbf{l}$ to Bob.
2. Alice and Bob engage in a ZKPoK protocol.

$$PK\{(\rho, r, s) : u^2 = g^{2\rho} \wedge e^2 = y_1^{2\rho}(n+1)^{2r} \wedge$$
$$v^2 = (y_2 y_3^{\mathcal{H}_{hk}(u,e)})^{2r} \wedge \mathbf{l} = \mathbf{g}^r\mathbf{h}^s \wedge B = h^r\}$$

Alice must also include a proof that the nymble indeed contains a blinded copy of her pseudonym.

$$PK\{(x, r) : B = h^r \wedge C = h^x h_b^r\}$$

We can thus combine the above protocol with proof of equality of discrete logarithms to produce a protocol that proves that the nymble is correctly constructed based on a valid pseudonym.

Algorithm 2 *VEDL-KeyGen*

```
Input: $\mathsf{p} = 2\mathsf{p}'+1, \mathsf{q} = 2\mathsf{q}'+1, n = \mathsf{pq}, n' = \mathsf{p}'\mathsf{q}', \mathbb{B}_{n'},$
  $\mathcal{H} : \{0,1\}^m \times \{0,1\}^* \rightarrow \{0,1\}^{\ell}$
Execution:
  $\mathbf{hk} \leftarrow_R \{0,1\}^m$
  $x_1, x_2, x_3 \leftarrow_R [n^2/4]$
  $g' \leftarrow_R Z_{n^2}^*$
  $g = (g')^{2n}$
  $y_1 = g^{x_1}, y_2 = g^{x_2}, y_3 = g^{x_3}$
  Choose random generators $\mathbf{g}, \mathbf{h} \in_R \mathbb{B}_{n'}$
  $pk_{NM} = (\mathbf{hk}, n, g, y_1, y_2, y_3, \mathbf{g}, \mathbf{h})$.
  $sk_{NM} = (\mathbf{hk}, n, g, x_1, x_2, x_3)$.
```

5. OUR CONSTRUCTION

In this section, we describe our construction in detail and analyze its security and efficiency. Our construction achieves all security goals of Nymble under a weaker trust model with more scalable performance.

There are two trusted third parties called the Credential Manager (CM) and the Nymble Manager (NM). The CM is in charge of issuing an anonymous credential which endorses the validity of the user's public key/private key. The NM is in charge of converting a nymble into a pseudonym, which is in turn blacklistable w.r.t. one service provider (SP). Users connecting to an SP via an anonymizing network must present a valid nymble, proving that the nymble is convertible to an authentic pseudonym and that the pseudonym is not blacklisted. The nymble is recorded with the associated session. In the case of a misbehaving user, the nymble will be sent to the NM to be de-anonymized.

5.1 Parameters

Let λ, ℓ, k, k' be security parameters, and let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map such that $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for some λ-bit prime p. Also let $g_0, g_1, g_2 \in \mathbb{G}_1$ be generators of $\mathbb{G}_1$ and $h_0 \in \mathbb{G}_2$ be a generator of $\mathbb{G}_2$. Let $\mathbb{G}_q \subset \mathbb{Z}_p^*$ be a cyclic group of prime order q in which the DDH problem is intractable. Let $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_q$, $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p$, and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^k$ be secure cryptographic hash functions.

Let p, q be safe primes of ℓ bits, such that $p = 2p'+1, q = 2q'+1$ for prime p' and q', and let $n = p \cdot q$. Let $\mathcal{H} : \{0,1\}^m \times \{0,1\}^* \rightarrow \{0,1\}^{\ell}$ be a secure cryptographic keyed hash function where $m = m(\ell)$ is an auxiliary security parameter.

Keys. The CM randomly chooses $\gamma \in_R \mathbb{Z}_p$ as secret key for credential generation and computes the public key $\rho = h_0^{\gamma}$ for pseudonym validation. The NM generates a set of public/private keys by Algorithm 2, as mentioned previously. Each user is tied to a private key $x \in \mathbb{Z}_q$ and a public key $PK = g_1^x$.

Our construction makes use of the concept of a linkability window, just as the original Nymble does. A linkability window prevents the SP and NM from colluding to link all of a user's sessions. The length of linkability window, w, can be set to a value that is reasonable for an individual SP, such as one day, one week or one month. Each linkability window has an index j which should be easily derived from the calendar, such as "the j-th month in the year 2010".

Each SP maintains a blacklist of user pseudonyms, an accumulator and an update log of the accumulator. The public and private keys of the accumulator, along with the initial value v_0, are generated by Algorithm 1. The update log consists of an ordered list of tuples (t_i, v_i), where t_i is the ith element added to the accumulator and $v_i = Update(t_i, v_{i-1})$. As aforementioned, each SP is tied to a unique pseudonym generator h, contained in the public key.

We note that each SP's pseudonym generator h changes for each linkability window. As a consequence, the blacklist will be emptied as well. It is required that the relative discrete logarithm between h values across different SPs and different linkability windows is unknown. To implement this, we take $h = H_0(id, j)$ where id is the canonical name of the SP and j is the linkability window index.

5.2 Registration

This protocol enables a user Alice to obtain a CL-credential from the CM upon successful completion. The credential is of the form (A, e, x, y) such that $A^{e+\gamma} = g_0 g_1^x g_2^y$. The CM makes sure that only one valid credential is issued corresponding to any given private key x. An alternative approach is let the CM issue only one CL-credential to any IP address, similar to the treatment in Nymbler [20].

This protocol can be seen as a simplified version of the registration protocol in BLAC [27, 28] except that the user is not anonymous here.

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

2. Alice sends to the CM a tuple (pk_A, Π_1), where $pk_A = g_1^x \in \mathbb{G}_1$, and Π_1 is a signature proof of knowledge of: $SPK_1\{(x) : pk_A = g_1^x\}(m)$.

3. Alice and the CM engage in some protocol to ensure that each user is issued only a single credential. For example, Alice could present a certificate for pk_A from a strong CA, or the CM could locally bind pk_A to Alice's IP address and refuse to issue new credentials to a bound IP address.

4. The CM verifies Π_1. It returns "failure" if the validation failed. Otherwise, the CM sends a tuple (A, e, y) to Alice such that $A = (g_0 g_1^x g_2^y)^{\frac{1}{e+\gamma}}$

5. Alice returns "failure" if $\hat{e}(A, \rho h_0^e) \neq \hat{e}(g_0 g_1^x g_2^y, h0)$. Otherwise, she stores (A, e, x, y) as her credential.

5.3 Authentication

This protocol lets SP Bob grant access privileges to a user Alice and obtain a nymble token from Alice upon successful completion. The private input to Alice is her credential $cred = (A, e, x, y)$ and her non-membership witness $nwit = (w, d, pnym)$ for Bob's blacklist accumulator such that $\hat{e}(v, h_0) = \hat{e}(w, h_0^{pnym}\theta)\hat{e}(g_0, h_0)^d$ and $pnym = h^x \in Z_p^*$.

1. Alice obtains a list of recent blacklist updates, ΔL from Bob through an anonymizing network, or from other reliable sources.

2. Alice parses ΔL as $\{(p_1, v_1), \ldots, (p_n, v_n)\}$ where each p_i is a blacklisted pseudonym and v_i is the new accumulator value after adding p_i. If $\exists i, pnym = p_i$, Alice knows she is blacklisted, she returns "failure" and aborts the protocol. Otherwise, Alice updates her (w, d) with Δ_L, and initiates a fresh anonymized session with Bob.

3. Bob sends a random challenge m to Alice, where $m \in_R \{0,1\}^l$.

4. Alice randomly selects $r \in_R \mathbb{Z}_q$, computes $B = h^r$ and commits to her private key x with $C = h^x h_b^r$. Alice commits her $pnym$ to a commitment $D = g_0^{pnym} g_1^t$ where $t \in_R \mathbb{Z}_p$. Alice produces a verifiable encryption of r under the NM's public key, $E = VerEnc(r)$.

5. Alice sends to Bob a tuple (B, C, D, E, Π_2), where Π_2 is a signature proof of knowledge:

$$SPK_2\{(r, x, A, e, y, pnym, w, d) :$$
$$B = h^r \wedge C = h^x h_b^r \wedge A^{e+\gamma} = g_0 g_1^x g_2^y \wedge pnym = h^x \wedge$$
$$v = w^{pnym+s} g_0^d \wedge E = VerEnc(r) \wedge x \in [0, p]\}(m)$$

6. Bob returns "failure" if SPK_2 doesn't verify. Otherwise, Bob grants Alice access and retains (B, C, E) in the log.

The instantiation of SPK_1 and SPK_2 is omitted due to space restrictions and is specified in the full version of this paper.

5.4 Filing Complaint and Blacklist Update

If Bob finds Alice has misbehaved, Bob executes this protocol to obtain Alice's $pnym$ from the NM and update the blacklist. The private input to the NM is SK_{NM}.

1. Bob sends (h, B, C, E) to the NM.

2. The NM extracts r from the decryption of E. It checks if $B = h^r$. If so, $pnym = C(h_b^r)^{-1}$ is computed. Otherwise the NM returns "failure". Further, the NM checks if $pnym$ has been computed before within the same linkability window. If so, the NM returns a random pseudonym with respect to h. Otherwise, $pnym$ is returned to Bob.

3. Bob updates the accumulator value with $pnym$. The new value $\tilde{v}$ and $pnym$ together will be added to the blacklist update log.

The reason for checking whether $pnym$ has already been computed is to ensure that any SP cannot use the NM as an oracle to link a user's authentications. Note that different nymbles from the same user for the same SP and linkability window decrypt to the same pseudonym.

Also, because of the involvement of the current pseudonym generator h in the transcript, Bob cannot abuse the NM to do meaningless jobs such as decrypting transcripts for past linkability windows.

5.5 Security Analysis

Jack achieves the four security goals of the original Nymble: **Blacklistability**, **Non-frameability**, **Anonymity**, and, through extensions, **Rate-limiting**. In the full version of this paper, we prove the following

THEOREM 5.1. *Jack is secure under the strong RSA assumption, the q-SDH assumption, and the DDH assumption over the subgroup of k-th residues modulo a prime p.*

Schemes	Authentication					Nymble Issuing			
	Communication		Computation			Communication		Computation	
	Downlink	Uplink	User Check	User Prove	SP	Downlink	Uplink	User	NM
Nymble	$O(L)$	$O(1)$	$O(\Delta_L)$	$O(1)$	$O(1)^*$	$O(W)$	$O(1)$	$O(1)$	$O(W)$
BLAC/EPID	$O(L)$	$O(L)$	$O(\Delta_L)$	$O(L)$	$O(L)$	-	-	-	-
PEREA	$O(L)$	$O(K)$	$O(K\Delta_L)$	$O(K)$	$O(K)$	-	-	-	-
Nymbler	$O(L)$	$O(1)$	$O(\Delta_L)$	$O(1)$	$O(1)^*$	$O(W)$	$O(W)$	$O(W)$	$O(W)$
Jack	$\boldsymbol{O(L)}$	$\boldsymbol{O(1)}$	$\boldsymbol{O(\Delta_L)}$	$\boldsymbol{O(1)}$	$\boldsymbol{O(1)}$	-	-	-	-

*Requires $O(L)$ preprocessing per time period

Table 1: Jack only requires constant computation cost for SP and constant size of signature for the user in authentication while it doesn't require the NM to issue nymbles. Without help from the NM, Jack achieves the best asymptotic complexities in all aspects, compared to previous schemes.

6. EVALUATION

We asymptotically analyze the computation and communication complexity of Jack, and in addition we experimentally evaluate the efficiency of a prototype implementation of Jack.

6.1 Complexity Analysis

The core component of Jack, which has the most impact on performance, is the user authentication protocol. We summarize the complexity of the authentication protocol of Jack in Table 1 and show comparisons with existing schemes. In all schemes, the authentication has five steps:

1. (Downlink) The user downloads the blacklist from the SP.
2. (User Check) The user inspects the new entries to the blacklist, optionally updating its non-membership witness.
3. (User Prove) The user prepares a proof showing that she's not blacklisted.
4. (Uplink) Alice sends the proof to the SP.
5. (SP Verify) The SP verifies the proof the user sent.

In Nymble and Nymbler the authentication is very efficient while the actual computation is shifted to the NM in the Nymble Issuing protocol. To be fair, we list the cost of issuing Nymbles as well.

The analysis is based on the following variables: L is the total size of the blacklist while Δ_L is the number of newly added entries in the latest blacklist with respect to the user's local copy of the blacklist. K is the length of the user queue in PEREA and W is the number of time periods within a linkability window in Nymble and Nymbler. We note that K is in some sense a SP security parameter for PEREA (longer K decreases the likelihood of the race condition mentioned previously), while W is in some sense a user security parameter for Nymble/Nymbler (larger W means that linkable time periods are shorter, given a fixed linkability window).

As in previous schemes, Jack requires Alice to download the whole blacklist so the communication cost of Downlink is of course $O(L)$. Similarly, in all schemes, the complexity of User Check is at least $O(\Delta_L)$, with a simple linear search. The only exception is the PEREA scheme where the user has K witness to update with. In Jack, Alice only needs to update one witness, making computation cost grow linearly with Δ_L.

In Jack, Alice only proves to the SP her knowledge of a single non-membership witness against the blacklist, with $O(1)$ computation. Therefore, the uplink bandwidth cost and the verification cost are $O(1)$ as well. In contrast, in PEREA, Alice proves the validity of K witnesses, while in BLAC Alice produces a proof of L inequalities. The costs of producing, transferring and verifying such proofs are $O(K)$ and $O(L)$ in BLAC and in PEREA respectively. In the case of Nymble and Nymbler, Alice only presents a valid nymble token with negligible computation and constant uplink cost. The SP in these schemes checks the nymble against the "linking tokens" in the blacklist; this can be done in $O(1)$ time, with a pre-processing cost of $O(L)$ per time period.

It is easy to see that the computational complexity and the communication cost of Registration is constant for all schemes. It is also easy to see that access revocation in all schemes only requires the NM computation of $O(1)$ to open a nymble and return the pseudonym/linking token to the complaining SP.

6.2 Prototype Implementation

We implement a prototype of Jack in C to evaluate the performance. In our prototype, the client produces SPKs for nymble connections, which are verified by the server. We make use of the Pairing-Based Cryptography (PBC) Library (version 0.5.7) [1] for the elliptic-curve and pairing operations. The GNU MP Bignum (GMP) Library (version 5.0.1) [2], from which PBC Library is built, is also used directly in our implementation. We also rely on OpenSSL for cryptographic functions, such as its SHA-1 hash function, and HMAC function family for realizing the cryptographic hash functions and HMACs in our system.

6.3 Parameter Choice

The choice of curve parameters is worth elaboration. We use two types of pairings (instead of one) in our prototype. First, we choose a pairing over Type-A curves, defined in the PBC library, to implement the user credential system. Type-A curves have the form $E : y^2 = x^3 + x$ over the field $\mathbb{F}_t$ for some prime t and have embedding degree 2. So $\mathbb{G}_T$ is a subgroup of F_{t^2}. $\mathbb{G}_1$ and $\mathbb{G}_2$ in this type-A pairing have order of the form $p = 2^a + 2^b + 1$, which is a prime factor of $t + 1$. Currently t and p in Type-A pairings are 512-bits and 160 bits respectively. The user's secret key x is a 160-bit random number.

[1]http://crypto.stanford.edu/pbc/

[2]http://gmplib.org/

Operation	Host	Mean Time (ms)	Trials
Witness Update	User	12.5	100
SPK_2 Signing w/o prep.	User	263.8	100
SPK_2 Signing w/ prep.	User	2.0	100
SPK_2 Verification	SP	207.7	100
Blacklist Update	SP	2.0	100
Nymble Opening	NM	26.1	100

Table 2: The timing for critical computations in Jack

Type-A pairings have the fastest pairing speed among PBC-defined pairings. However, since $p-1$ has prime factors of size up to $(a-b)$ bits (56 in this case), it is not safe to to implement DUAs for DDH groups on a type-A pairing.

Instead, we choose a pairing over Type-D curves [23], also defined in the PBC library, for the implementation of our accumulator. Such curves have the form $E: y^2 = x^3+cx+d$ and have embedding degree 6. $\mathbb{G}_1$ are defined over $E(\mathbb{F}_s)$ and have order p where s and p are 248-bits and 224-bits respectively. Furthermore, $p-1 = 16 \cdot q$ where q is a 220-bit prime.

The security parameter ℓ for verifiable encryption is set to 512 and $m(\ell) = 160$. Thus the safe primes p and q are 512-bits. The modulus n of the underlying Paillier encryption scheme is 1024-bits. Security parameters k and k' are both set to 160. For the proof of knowledge of double discrete logarithm, the number of rounds is set to 80.

6.4 Experimental Performance

The prototype is tested on a Dell Precision T3500 machine with a quad-core 2.67 GHz Intel Xeon W3550 CPU and 12 GB RAM, running Ubuntu 9.04.

We record the average timing of major operations in Jack, illustrated in table 2, taken over 100 trials. The SPK signing procedure allows the user to pre-process expensive commitments in advance, thus optimizing the performance as shown in "SPK_2 Signing with prep." The data shows that the bottleneck of the performance is as expected the verification of SPK_2 on the SP side, which is around 208 milliseconds. However, there are ways to speed up this operation, e.g. parallel processing with multi-core, or nymble pre-verification by a TTP (discussed in §7.4).

First, we compare the performance of Jack with BLAC in terms of communication cost. As expected, the communication cost for Jack stays constant at around 80KB while that for BLAC grows linearly. Wikipedia currently lists between 20,000 and 25,000 IPs that are permanently blocked [3], which equivalently translates to about 6 to 7.5 MB of uplink cost.

Next, we compare the authentication time at the SP side for Jack, BLAC, Nymble and Nymbler for various blacklist sizes L. Although the SP in Nymbler has very efficient signature verification on nymbles, the fact is the NM takes on the computational load of nymble verification for each SP [20] in the Nymble Issuing Protocol. It is reasonable to compare the total computational resources consumed per authentication across different schemes, assuming a constant rate of 1 anonymous authentication per second. In Nymbler and Nymble, the NM has to compute a series of linking tokens for each blacklisted nymble, in addition to issuing nymbles. Therefore, the total computation for the NM per authentication is $t_a + t_l L/\mathsf{A}$ where t_a is the time needed for nymble issuing, t_l is the time to compute one linking token and A is the number of authentications in one time period. We choose a time window of 1 day with 288 time periods as the original Nymble does.

[3]http://en.wikipedia.org/wiki/Special:BlockList, accessed at Apr. 15th 2010

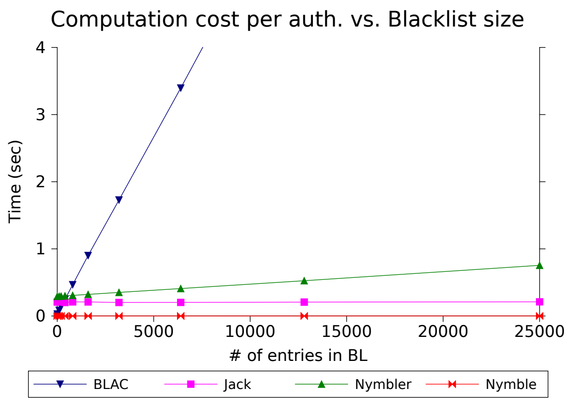

Figure 2: We compare the average computation load per authentication at the server side across the four schemes. Jack has the best scalability while the CPU consumption of the other three schemes scales linearly with the size of the blacklist. (Note the growth of Nymble is very small and thus invisible in the graph.)

In Figure 2 we show the computation load on the SP and NM per authentication. As expected, BLAC, while requiring the least trust in outside parties, doesn't scale. On the other end, Nymble offers superb efficiency, however, with the strongest trust model. Jack consumes less CPU than Nymbler, and both Jack and Nymbler work under a common moderate trust model. As the size of the blacklist grows, the performance gap between Jack and Nymbler widens linearly.

7. DISCUSSION

In this section, we briefly discuss several easy extensions to Jack.

7.1 Objective Blacklisting

As mentioned in [25], in certain circumstances where objective blacklisting is preferred, users may call for a third-party arbitrator other than SPs for the judgement of misbehavior and anonymity revocation. In that case, the NM in our scheme can play the role of arbitrator. The only change is that a policy needs to be incorporated into each nymble, specifying under what conditions the nymble can be opened. When a SP complains about a nymble, she will provide the policy as well. The NM will judge if the policy is satisfied and react accordingly.

Luckily, our construction of nymbles provides this for us by design. Users can specify the policy in the currently unused tag $\mathcal{L}$ in the verifiable encryption of $E(r)$. When the NM makes the judgement, it can look up the policy $\mathcal{L}$ with $E(r)$ and decide whether the user has violated the policy.

7.2 Rate Limiting

There are many existing protocols that enforce limits on the total number of anonymous authentications within a time period. For example, users can obtain at most k e-cash coins from the SP and spend one coin every time the user wants to authenticate. In k-times anonymous authentication (k-TAA) [26], users will keep their anonymity unless they authenticate more than k times. In periodic n-times anonymous authentication [10], a token dispenser will give out at most n one-time tokens in one time period and the dispenser automatically renews itself every time period.

Our scheme is compatible with any of the above rate-limiting protocols. To realize a rate-limiting Nymble connection, a user first authenticates with the SP with one rate-limiting protocol. Once that authenticates, the user can further execute the authentication protocol in our scheme to obtain the service.

7.3 Blacklist Duration Management

For some SPs, it is desirable to have flexibility in the duration of a user's blacklisting. It is reasonable and common in Wikipedia to block users for different durations [2]. The current design of our scheme only allows SPs to blacklist a user for up to the fixed duration w, the length of the linkability window. However, we can introduce an easy extension to solve this problem.

To support bans of different durations, SPs in our scheme could maintain multiple blacklists with different scales of linkability window. As an illustration, a SP can have three blacklists, one for the current month, one for the current week and one for the current day. To authenticate, the user must present a non-membership proof for all three blacklists. And the SP can choose to block a misbehaving user for the rest of the month, or until the end of the week, or for the day, according to the severity of the damage the user caused. In other words, with the extension of multi-blacklists, SPs have freedom to block a die-hard vandal for as long as they want, and to cool down an edit war by temporarily blocking the editors involved for a short time period.

7.4 Speeding-up Nymble Verification

The procedure of verifying a nymble and the attached signature proof is so expensive that deploying Jack on a popular SP seems difficult. To address this problem, we propose the following extension. Similar to the treatment in Nymbler, the computational expense of nymble verification at the SP side could be relieved by a public TTP [20]. Let's call this TTP the Authentication Manager (AM). In this case, the authentication breaks down into three steps:

1. Instead of sending the entire nymble (B, C, D, E, Π_2) to the SP, user Alice sends (B', C', D', Π_2') to the AM where (B', C', D') is a blinded version of (B, C, D) and Π_2' proves the integrity of (B', C', D'). Once AM verifies Π_2', AM gives Alice a (blinded) RSA signature $\sigma' = sig_{AM}(B', C', D')$.

2. Alice unblinds σ' to yield $\sigma = sig_{AM}(B, C, D)$.

3. Alice sends $(B, C, D, E, \sigma, \Pi_3)$ to the SP where Π_3 is a signature proof of knowledge:

$$SPK_3\{(r) : B = h^r \wedge E = VerEnc(r)\}(m)$$

   The SP authenticates Alice if σ verifies against (B, C, D) and Π_3 verifies with (B, E).

Because of the efficiency of RSA signature verification and VEDL, the computational load at the SP side is greatly reduced, to several milliseconds. The AM could even be implemented by nodes within the anonymization service; for instance, Tor nodes already have RSA public keys. We note that all computation by the AM can be verified by any other node, minimizing the trust placed in this party: a cheating AM can be caught by a single SP and distrusted by all SPs.

8. CONCLUSION

We have presented Jack, a new anonymous blacklisting scheme which provides scalable performance with an improved TTP trust model, wile providing identical functionality and privacy security guarantees to other TTP-based schemes. The system is constructed from dynamic universal accumulators for DDH groups, verifiable encryption of discrete logarithms and CL-signatures. Compared to previous work such as BLAC, where the time and bandwidth required for signature generation and verification grows linearly with the size of the blacklist, our scheme stands out with a constant cost in both time and bandwidth.

9. ACKNOWLEDGMENTS

We thank Jan Camenisch for helpful explanation of the VEDL scheme, Apu Kapadia for a helpful discussion about Nymble, and Ian Goldberg for several helpful clarifications of Nymbler. We also thank several anonymous reviewers for their constructive feedback. This work was supported by the NSF under grant 0546162.

10. REFERENCES

[1] The Tor Project. `https://www.torproject.org/`.

[2] Wikipedia:Blocking policy. `http://en.wikipedia.org/wiki/Wikipedia:Blocking_policy`.

[3] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Proceedings on Advances in Cryptology - CRYPTO '00*, pages 255–270, London, UK, 2000. Springer-Verlag.

[4] M. H. Au, P. P. Tsang, W. Susilo, and Y. Mu. Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In *Topics in Cryptology CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 295–308. Springer Berlin / Heidelberg, 2009.

[5] M. H. Au, P. P. Tsang, W. Susilo, and Y. Mu. Dynamic Universal Accumulators for DDH Groups and Their Application to Attribute-Based Anonymous Credential Systems. Technical report, Dartmouth Computer Science TR2009-643, 2009.

[6] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73, New York, NY, USA, 1993. ACM.

[7] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology - CRYPTO '04*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer Berlin / Heidelberg, 2004.

[8] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *CCS '04: Proceedings of*

the 11th ACM conference on Computer and communications security, pages 168–177, New York, NY, USA, 2004. ACM.

[9] E. Brickell and J. Li. Enhanced privacy id: a direct anonymous attestation scheme with enhanced revocation capabilities. In *WPES '07: Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 21–30, New York, NY, USA, 2007. ACM.

[10] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 201–210, New York, NY, USA, 2006. ACM.

[11] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Proceedings on Advances in Cryptology - CRYPTO '02*, pages 61–76, London, UK, 2002. Springer-Verlag.

[12] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Proceedings on Advances in Cryptology - CRYPTO '03*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer Berlin / Heidelberg, 2003.

[13] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In *Proceedings on Advances in Cryptology - CRYPTO '97*, pages 410–424, London, UK, 1997. Springer-Verlag.

[14] D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology - EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer Berlin / Heidelberg, 1991.

[15] I. Damgård and N. Triandopoulos. Supporting non-membership proofs with bilinear-map accumulators. Cryptology ePrint Archive, Report 2008/538, 2008. `http://eprint.iacr.org/`.

[16] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[17] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology - CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.

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

[19] 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.

[20] R. Henry, K. Henry, and I. Goldberg. Making a nymbler nymble using verbs. Technical report, University of Waterloo Technical Report CACR 2010-05.

[21] P. C. Johnson, A. Kapadia, P. P. Tsang, and S. W. Smith. Nymble: Anonymous IP-address blocking. In *Proceedings of The Seventh International Symposium on Privacy Enhancing Technologies (PET), Ottawa, Canada*, volume 4776 of *LNCS*, pages 113–133. Springer-Verlag, June 2007.

[22] J. Li, N. Li, and R. Xue. Universal accumulators with efficient nonmembership proofs. In *ACNS '07*, pages 253–269, Berlin, Heidelberg, 2007. Springer-Verlag.

[23] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for fr-reduction (special section on discrete mathematics and its applications). *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 84(5):1234–1243, 20010501.

[24] L. Nguyen. Accumulators from bilinear pairings and applications. In *Topics in Cryptology âĂŞ CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292. Springer Berlin / Heidelberg, 2005.

[25] E. J. Schwartz, D. Brumley, and J. M. McCune. Contractual anonymity. In *Proceedings of the 17th Annual Network and Distributed System Security Symposium*, San Diego, CA, February 2010.

[26] I. Teranishi, J. Furukawa, and K. Sako. k-times anonymous authentication (extended abstract). In *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 81–95. Springer Berlin / Heidelberg, 2004.

[27] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 72–81, New York, NY, USA, 2007. ACM.

[28] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. BLAC: Revoking Repeatedly Misbehaving Anonymous Users Without Relying on TTPs. Technical report, Dartmouth Computer Science TR2008-635, 2008.

[29] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. Perea: Towards practical ttp-free revocation in anonymous authentication. In *CCS '08: Proceedings of the 14th ACM conference on Computer and communications security*, pages 333–344. ACM, 2008.

[30] P. P. Tsang, A. Kapadia, C. Cornelius, , and S. W. Smith. Nymble: Blocking Misbehaving Users in Anonymizing Networks. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, Sept. 2009.

[31] P. P. Tsang, A. Kapadia, C. Cornelius, and S. W. Smith. Nymble: Blocking misbehaving users in anonymizing networks. Technical report, Dartmouth Computer Science TR2008-637, 2008.