CERIAS Tech Report 2007-29

ONLINE SUBSCRIPTIONS WITH ANONYMOUS ACCESS

by Marina Blanton

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Online Subscriptions with Anonymous Access

Marina Blanton
Department of Computer Science
Purdue University
mbykova@cs.purdue.edu

Abstract

Online privacy is an increasingly important problem, as many services are now offered in a digital form. Privacy (or the lack thereof) is of a special concern in subscriptions to large data repositories with heterogeneous information, where the service provider can easily profile its users and sell that information to third parties. In this work we present the design and implementation of a system that closely resembles the current practice of subscriptions to many services such as newspapers, digital libraries, music collections, etc., but at the same time offers anonymous access to the service. As with current practice, in our solution a user subscribes to the service obtaining access to it for a certain period of time, at the end of which the subscription expires.

In our system user access is always anonymous and no two transactions by the same user can be linked together. Moreover, the system assures a high level of protection to the service provider, as a user cannot share her subscription credentials with others without denying herself access to the service. We present experimental results showing that the design of our system results in only small computation overheads, in addition to having very low communication requirements. The main objective of this work is thus to illustrate the practically of integrating anonymity into today's subscription-based services.

1 Introduction

This work focuses on practical aspects of realizing pre-paid subscriptions to digital services in a setting where legitimate users can access such services anonymously. The services we consider include online newspapers, databases, digital libraries, music collections, etc. Two important considerations that governed the design of the system are: (i) anonymous access to the documents, objects, and files included in the subscription; and (ii) the flexibility in subscription periods and types of access.

The model of operation considered here is rather simple and naturally resembles currently existing services (not necessarily web-based or digital): a customer subscribes to an online or digital service for a certain period of time by making a payment for the duration of the subscription. Her subscription takes effect at end of the payment protocol and expires once the subscription interval is over, i.e., similar to how most (non-anonymous) services work today. There is no limit on how many times the service can be accessed during that time interval. Subscription lengths might vary depending on the application (e.g., subscription for a month, a year, promotional free subscription for a few days, etc.), and we would like to support services where the time granularity for the system is small (e.g., a day or less) to allow for flexible start and end times. If the service provider offers a number of subscription types (e.g., basic and premium packages), each customer will be free to choose her subscription type.

In the world where a user's online presence can be easily monitored, we seek user privacy, which is particularly important in case of services that provide access to information on a broad range of topics. If access is *anonymous*, the service provider has no information about the (legitimate) items a customer is accessing and is unable to take advantage of customer access history (by building profiles, selling this information to third parties, etc.). This also corresponds to the currently existing (non-digital) practices, as no one monitors what articles an individual reads in a newspaper or what books he browses at a library. It is obvious that an anonymous setting is not appropriate for access to restricted-use or potentially dangerous material access to which might require auditing by law, but rather can be used with commercially valuable but innocuous content such as databases of past events, historical trading transactions, music, etc.

There are a number of existing anonymity tools in the literature, but the straightforward use of any of them has certain drawbacks. For example, the use of digital credentials requires tying user credentials to some other information or infrastructure such as PKI to prevent system abuse, which might be difficult to do for a stand-alone service. As another example, the use of group signatures also permits realization of anonymous access, but requires user re-keying during each time slot and becomes unacceptable when such time slots are short (e.g., a day or less).

In this work we adopt existing anonymity techniques to design a solution that is both convenient to the user and safe for the service provider (even a stand-along service). We achieve a high level of security for the service provider by ensuring that a user cannot misuse her access rights. Similar to digital credentials, each user is issued an anonymous token that permits access to the service. The key difference in our mode of operation, however, is that if a user duplicates her token and gives a copy to a friend, she will deny herself access to the service.[1] We achieve this by issuing single-use tokens to each user: on each access a user spends her current (anonymous) token and is issued a new token.

Another important aspect in showing the practicality of our system is performance of the solution. Low computation overheads will foster adoption of such tools and one of our goals here is to evaluate such overheads. To implement anonymous tokens, we use an efficient signature scheme over elliptic curves of compact size. According to our experiments, user computation per access can normally be under 1.5 second on a commodity workstation with a server computation being under 1 second (using the same computational power). To the best of our knowledge, this is the first work that reports on empirical performance results of an anonymous authentication scheme.

To summarize, we present the design and performance analysis of the first system that simultaneously achieves the following properties:

- access by a user to the data repository is provably anonymous;
- the size of authentication token is constant and does not depend on the number of users in the system;
- subscription dates are flexible, and each subscription is terminated at its expiration date;
- termination of a subscription does not affect other users in the system;
- strong protection for the service provider is guaranteed, as no user can share or duplicate her credentials.

The rest of this paper is organized as follows: in Section 2 we give an overview of related work. The problem setting and security requirements are presented in Section 3, then Section 4 provides a high-level description of our solution. Section 5 describes building blocks, followed by a description of user-server interaction in Section 6. Section 7 describes our experiments and reports on performance results. Finally, Section 8 comments on extensions and concludes this work.

2 Related Work

This work is related to a number of research directions, and we discuss each such line of research separately.

Subscription-based services. Persiano and Visconti [26] address the problem of remote subscription-based services, and it is the closest to our work. The authors modify the SSL/TLS protocol (used with X.509 certificates) to achieve privacy, but the solution requires authentication linear in the number of users in

[1]In some other solutions, duplicating the token and its usage by several people undetected is feasible.

the system (both computation and communication). Their solution can be integrated into existing infrastructures, but for popular services with a large number of users it becomes infeasible (e.g, for a system of 10,000 users, the authors report authentication of 40-50 seconds). Another work on anonymous authentication [28] has similar properties, but its performance is also linear in the number of users in the system.

Another work on subscription-based services in the anonymous setting is due to Ramzan and Ruhl [27]. The model, however, is such that a user obtains a fixed number of accesses to the service with no expiration date. This does not solve our problem where we permit an unlimited number of accesses, but enforce expiration of user access rights.

Group signatures. Group signatures allow a member of a group to sign messages anonymously on behalf of that group. A large drawback of using them to realize anonymous subscriptions is that there is no convenient way to terminate group membership. That is, in group signature schemes that support revocation (e.g., [1, 3, 8, 12, 10, 30]), a member is revoked by (i) using a list of revoked users (every time a signature is verified, it is checked against the list of revoked users); or (ii) updating the secret key every time a user is revoked. Since in our model revocation is not a rear event, requiring each user to perform work linear in the number of revocations is infeasible. Another work [21] allows for user revocation without affecting other users in the system, but requires all users to authenticate by going through a third party (not affiliated with the server provider), causing additional overhead and infrastructure costs.

Digital credentials. Digital credentials (see, e.g., [5, 13]) allow someone to authenticate or obtain access to certain resources by providing credentials that are certified by a trusted authority and are tied to the user's public key. In the original scheme [5], multiple showings of the same digital credential were linkable, but recent work [2, 15] allows digital credentials to be randomized and achieves unlinkability. The biggest weakness with using them, however, is that such certificates can be shared or duplicated by dishonest users (and even simultaneous uses of the same credential cannot be detected in an anonymous setting). To overcome this limitation, [5] suggests encoding confidential data of the applicant into the digital credential. To make users liable, however, a certificate that, for instance, encodes a user's key to access a bank account as a protective measure must be accepted at the bank, which makes it hard to bootstrap certificate applications in practice [5]. [2] suggests binding a credential to the user by requiring the user to use an important secret key she owns every time she uses the certificate. Similarly, tying a credential to the user's important secret implies existence of an underlying infrastructure, which might not be readily available and too costly for the service provider to build and/or maintain.

Our solution, on the other hand, provides a stronger security guarantee: it is simply not possible to create multiple valid copies of a user's credential since every credential consists of a chain of single-use tokens. In addition, any measures for protecting digital credentials from being shown to other individuals can be used with our system as well.

Digital cash and k-times anonymous authentication. Digital cash could potentially be used for anonymous subscriptions if we use a coin to permit access to the subscription service during a single time interval. There are schemes that permit a coin to be spend up to k times (such as in [6, 7] and k-times anonymous authentication [32, 25, 11]), which would allow a user to access the service multiple times during each time interval. The limitations of using these solutions in our setting, however, outweigh its advantages. That is, we want to permit a user to access the service an unlimited number of times during each time interval, which means that the threshold k will have to be set to a high value. This unnecessarily increases a coin size and practically disables benefits of such schemes, since their goal is to prevent over-spending. Additionally, it is unclear how to enforce a coin to be spent during the pre-defined time interval (or "expire" unspent coins) and how to ensure that the number of issued coins is not linear in the subscription length.

Unlinkable serial transactions. Stubblebine, Syverson, and Goldschlag [31] gave an interesting approach based on a single-use token: when a user subscribes, she receives an (anonymous) token. At the time of access to the service, the token is spent and the user receives a different token (the server stores all tokens used). By deviating from the protocol, not only a customer cannot gain anything, but also might lose his access privileges. This scheme is very efficient and is well suited for, for instance, pay-per-view service, but does not work in our setting because there is no way in this scheme to stop the chain and force subscription termination. That is, in order for a single subscription to be terminated, the key must be changed and all remaining users must obtain new certificates, which clearly is not suitable.

3 The Model

In this section we define the model of secure anonymous subscriptions. In what follows, we let the service provider to specify different types of subscriptions for its users. Note that usage of subscription types greatly increases the expressive power of the scheme, as it allows the service provider to specify what objects each category of users is allowed to access and at what times (e.g., discontinuous time intervals such as evenings-and-weekends subscriptions are now supported, as well as traditional varying grades of service packages).

Let us first define some notation. Let time be partitioned into time intervals t_i (of possibly varying lengths), with t_{i+1} following t_i and and t_{curr} denoting the current time interval. Then an access function f, given a subscription type stype and a time interval t_x, returns the set of objects (or data items) accessible by a holder of stype at time t_x. Note that f does not take into account subscription expiration or the current time interval, but still might return an empty set depending on the subscription type (e.g., if the access is based on the time of the day or day of the week). We assume that payment information (if applicable) is available for all subscription types and durations and is implicit in the foregoing description. An anonymous subscription scheme is then comprised of the following procedures:

- Setup: The server generates a public key/secret key pair and publishes its public key.
- $\mathsf{Subscribe}$: During this protocol, a user who requests a subscription type stype for a duration d becomes a valid subscriber from time t_{start} until t_{end}, where $d = t_{end} - t_{start}$, and obtains from the server her credential information cred.
- Access: It is a protocol between a subscriber and the server that, given customer credentials cred and a requested object o, provides the customer with the contents of the object o if both $t_{curr} \le t_{end}$ and $o \in f(\mathsf{stype}, t_{curr})$ and then updates cred.

We assume that, given a subscription type stype, function f can be successfully computed and enforced and concentrate on proper authentication and termination of customer access rights at the appropriate time. The properties we seek from a secure anonymous subscription scheme are:

- *Correctness:* Any subscriber who joined and received credentials using $\mathsf{Subscribe}$ must be able to access the prescribed resources using Access during her subscription period.
- *Soundness:* Only legitimate subscribers are able to authenticate and obtain access to the prescribed resources during their subscription period. This applies to a coalition of users as well: any subset of colluding users cannot obtain access to more resources than what they can already legitimately access.
- *Anonymity:* No one is able to identify an authenticated subscriber within her category or to decide whether two different executions of the Access protocol were performed by the same subscriber.

Besides these properties, for efficiency and usability reasons we require the (non-security related) property of *Taking effect fast*. This means that the time interval when the service was requested and the time interval when the subscription takes effect are sought to be very close in time.

4 Overview of the Solution

Here we give an overview of interaction between a user $\mathcal{U}$ and the server $\mathcal{S}$ in our solution at two stages: *subscribing* to the service and *accessing* the service.

At the subscription time: The protocol consists of the following steps:

1. A user $\mathcal{U}$ and the service provider $\mathcal{S}$ agree on the subscription type (call it t) and the subscription duration (represented as the expiration date and time d). The user pays for her subscription.
2. To generate its first access token, $\mathcal{U}$ picks a random string m, sends a commitment $com(m)$ to $\mathcal{S}$, and proves that she knows m.
3. $\mathcal{S}$ creates an authentication token σ (which is a signature on m, t, and d) by using t, d, and $com(m)$ and sends it to $\mathcal{U}$.
4. $\mathcal{U}$ verifies the validity of the token, which will allow future access to the service.

At the time of access: Every time the user $\mathcal{U}$ would like to access the service, she will need to reveal the access type t to the server, which will allow $\mathcal{S}$ to enforce proper access control. Note that revealing the user's subscription type does not compromise the anonymity of the access, as $\mathcal{U}$ remains anonymous within her subscription type. The exact interaction between $\mathcal{U}$ and $\mathcal{S}$ is as follows:

1. $\mathcal{U}$ randomizes her token σ so that it cannot be recognized, and sends σ, t, and information about m to $\mathcal{S}$. $\mathcal{U}$ also proves to $\mathcal{S}$ that σ is a valid signature on m, t, and some date in the future (i.e., d is kept secret).
2. $\mathcal{S}$ checks to make sure that m has not been used before and grants access to the service.
3. To generate a new authentication token, $\mathcal{U}$ picks a new random $\hat{m}$, sends a commitment to it $com(\hat{m})$ to $\mathcal{S}$, and proves the knowledge of $\hat{m}$.
4. $\mathcal{S}$ creates a new token $\hat{\sigma}$ using commitment $com(\hat{m})$ and the previous token that contains information about t and d and sends it to $\mathcal{U}$. This new token will correspond to a signature on $m + \hat{m}$, t, and d.
5. $\mathcal{U}$ verifies the validity of $\hat{\sigma}$.

As can be seen from the above, the service provider must maintain a database of the previously used tokens (i.e., store each used m). Here, however, we would like to point out that, unlike digital cash systems, the size of the database will be limited because the authentication tokens have limited validity. If we let T_{max} denote the longest possible subscription, then $\mathcal{S}$ needs to store used tokens for at most $2T_{\mathsf{max}}$ time intervals in the past, and all older tokens can be safely discarded. This prevents the database from growing indefinitely.

5 Building Blocks

5.1 Preliminaries

The signature scheme which the server uses to construct authentication tokens uses groups with bilinear maps. Thus, we review the definition of groups with pairings next. In what follows, we use $a \overset{R}{\leftarrow} G$ to mean that a is chosen at random from all of the possible values that G can take.

Definition 1 (Bilinear map) *A one-way function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map if the following conditions hold:*

- *(Efficient) $\mathbb{G}$ and $\mathbb{G}_T$ are groups of the same prime order q, and there exists an efficient algorithm for computing e.*
- *(Bilinear) For all $g \in \mathbb{G}$ and $a, b \in \mathbb{Z}_q$, $e(g^a, g^b) = e(g, g)^{ab}$.*
- *(Non-degenerate) If g generates $\mathbb{G}$, then $e(g, g)$ generates $\mathbb{G}_T$.*

5.2 The signature scheme

In this work we make use of Camenisch-Lysyanskaya (CL) signature scheme [13]. Thus, here we briefly review this scheme that consists of the key generation, signing, and verification algorithms.

KeyGen: Generate $(q, \mathbb{G}, \mathbb{G}_T, g, e)$. Choose $s \overset{R}{\leftarrow} \mathbb{Z}_q$ and $u \overset{R}{\leftarrow} \mathbb{Z}_q$. Set the secret key $sk = (s, u)$ and the public key $pk = (q, \mathbb{G}, \mathbb{G}_T, g, e, g^s, g^u)$.

Sign: To sign a message $m \in \mathbb{Z}_q$, first choose $\alpha \overset{R}{\leftarrow} \mathbb{Z}_q$ and set $a = g^{\alpha}$. The signature σ is computed as $\sigma = (a_1, a_2, a_3) = (a, a^u, a^{s+sum})$.

Verify: To verify a signature $\sigma = (a_1, a_2, a_3)$ on message m, check its every field by performing:

$$(1)\ e(a_1, g^u) = e(a_2, g) \qquad (2)\ e(a_1, g^s)e(a_2, g^s)^m = e(a_3, g).$$

The security of the signature scheme relies on the LRSW assumption (see, e.g., [23, 13]).

The above scheme does not unconditionally hide the message m, which is undesirable in cases when m must be kept secret from the verifier (i.e., in the above, given a signature on an unknown message, one can try different values for m in the second step of signature verification until a match is found; this attack can be successful with high probability if the message domain is small). To address this, the above scheme can be extended to support signatures on two messages (m, r) where r is chosen at random and unconditionally hides m in the signature. This extension will allow the signer to produce a signature on committed value without learning the value being signed and also prove the knowledge of a signature without disclosing the message. Furthermore, if the message is completely hidden, one will be able to randomize the signature, so that it is infeasible to tell whether the original signature and its randomized version correspond to the same message or not. The next section shows how signatures on blocks of messages (two or more messages) can be created and randomized.

5.3 Signatures on multiple messages

As given in [13], to extend the above scheme to support multiple messages $(m_0, \ldots, m_n)$, the secret and public keys are expanded and the signature now has additional fields. For instance, to sign two messages (m_0, m_1), the secret key will consists of (s, u, z_1), and the public key will be $pk = (q, \mathbb{G}, \mathbb{G}_T, g, e, g^s, g^u, g^{z_1})$. To sign (m_0, m_1, m_2), set $sk = (s, u, z_1, z_2)$ and $pk = (q, \mathbb{G}, \mathbb{G}_T, g, e, g^s, g^u, g^{z_1}, g^{z_2})$, etc.

Our subscription scheme (given in Section 6) uses signatures on messages (m, r, t, d), where r is a random number used to hide other values. A signature σ on (m, r, t, d) will then look like $\sigma = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9) = (a, a^{z_1}, a^{z_2}, a^{z_3}, a^u, a^{uz_1}, a^{uz_2}, a^{uz_3}, a^{s+su(m+z_1r+z_2t+z_3d)})$. To verify such a signature, we need to verify all of the fields in it. For instance, for message (m, r, t, d), the verification procedure consists of the following steps, in addition to (1) $e(a_1, g^u) = e(a_5, g)$:

$$\begin{aligned}
\text{for } z_1:&\ (2)\ e(a_1, g^{z_1}) = e(a_2, g); \qquad (3)\ e(a_2, g^u) = e(a_6, g);\\
\text{for } z_2:&\ (4)\ e(a_1, g^{z_2}) = e(a_3, g); \qquad (5)\ e(a_3, g^u) = e(a_7, g);\\
\text{for } z_3:&\ (6)\ e(a_1, g^{z_3}) = e(a_4, g); \qquad (7)\ e(a_4, g^u) = e(a_8, g);\\
\text{for the message}:&\ (8)\ e(a_9, g) = e(a_1, g^s)e(a_5, g^s)^m e(a_6, g^s)^r e(a_7, g^s)^t e(a_8, g^s)^d.
\end{aligned}$$

To randomize signature σ (issued on any number of messages), we need to randomly choose r_1 and r_2 from $\mathbb{Z}_q$, raise all fields a_i of σ to r_1, and additionally raise the last field of the signature to r_2.

To have the signer produce a signature on an unknown message, it is sufficient to send a commitment to that message to the signer, from which it will be able to form the signature. The commitment scheme used is Pedersen commitment, and a commitment to m will look like $com(m) = g^m(g^{z_1})^r$, from which the signer can form a signature on (m, r). Similarly, given a commitment to several messages, a signature on all of them can be formed.

5.4 Zero-knowledge proofs of knowledge

Prior literature provides efficient zero-knowledge proofs of knowledge (ZKPK) for a variety of statements, with many efficient proofs being based on the discrete logarithm problem (see, e.g., [18, 17, 14, 4, 9]). Camenisch and Stadler [16] introduced notation for various proof of knowledge and we follow their notation here. For example,

$$PK\{(\alpha, \beta, \gamma) : A = g^{\alpha} h^{\beta} \wedge B = g^{\alpha} h^{\gamma} \wedge (\alpha \geq a)\}$$

denotes a ZKPK of integers α, β, and γ, where $A = g^{\alpha} h^{\beta}$, $B = g^{\alpha} h^{\gamma}$, and $\alpha \geq a$.

One of the ZKPKs used in our protocols is the proof of knowledge of the discrete logarithm representation, a solution to which is well known. For the purpose of completeness of this work and to facilitate deployment of these techniques, we describe its non-interactive version in Appendix A. More information on other proofs of knowledge used in our scheme is given in Section 6.

6 The Scheme

6.1 The protocols

Setup: $\mathcal{S}$ generates $(q, \mathbb{G}, \mathbb{G}_T, g, e)$ and chooses $s, u, z_1, z_2, z_3 \overset{R}{\leftarrow} \mathbb{Z}_q$. The secret key is $sk = (s, u, z_1, z_2, z_3)$ and the public key is $pk = (q, \mathbb{G}, \mathbb{G}_T, g, e, g^s, g^u, g^{z_1}, g^{z_2}, g^{z_3})$.

Subscribe:

1. $\mathcal{U}$ and $\mathcal{S}$ negotiate $\mathcal{U}$'s subscription type $t \in \mathbb{Z}_q$ and expiration date/time $d \in \mathbb{Z}_q$.
2. To create the first token, $\mathcal{U}$ picks at random $m, r \overset{R}{\leftarrow} \mathbb{Z}_q$, sends the commitment $M = g^m (g^{z_1})^r$ to $\mathcal{S}$, and gives a non-interactive ZKPK (NIZKPK) of the opening of the commitment:

$$PK\{(\alpha, \beta) : M = g^{\alpha} (g^{z_1})^{\beta}\}$$

3. $\mathcal{S}$ produces a signature σ on m, t, and d using M such that $\sigma = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)$ (as in Section 5.3). Since the only field that depends on the messages is a_9, $\mathcal{S}$ computes it as $a_9 = a^{s+su(z_2 t + z_3 d)} M^{su\alpha}$, where $a = g^{\alpha} = a_1$ for a randomly chosen α.
4. $\mathcal{U}$ checks the signature σ on (m, r, t, d) by checking all fields (see Section 5.3).

Access:

1. $\mathcal{U}$ chooses $r_1, r_2 \overset{R}{\leftarrow} \mathbb{Z}_q$ and randomizes σ (as in Section 5.3). $\mathcal{U}$ sends m, t, and (randomized) σ to $\mathcal{S}$. (Through the rest of this protocol we assume that σ's fields a_1 through a_9 have been modified according to the randomization procedure.)
2. $\mathcal{U}$ proves in zero knowledge that σ is a valid signature on m, t and some unknown d where $d \geq t_{curr}$ using the following proof of knowledge:
   (a) Both $\mathcal{U}$ and $\mathcal{S}$ locally compute $v_r = e(a_6, g^s)$, $v_d = e(a_8, g^s)$, $v_{sig} = e(a_9, g)$ and $V = e(a_1, g^s) \cdot e(a_5, g^s)^m \cdot e(a_7, g^s)^t$.
   (b) $\mathcal{U}$ and $\mathcal{S}$ execute a NIZKPK protocol for:

$$PK\{(\alpha, \beta, \gamma) : v_{sig}^{\gamma} = V \cdot v_r^{\alpha} \cdot v_d^{\beta} \wedge \beta \geq t_{curr}\}$$

3. $\mathcal{S}$ checks to ensure that m has not been used before and, if true, access to the service can be granted.
4. To generate a new token, $\mathcal{U}$ picks new $\hat{m}, \hat{r} \overset{R}{\leftarrow} \mathbb{Z}_q$, computes commitment $\hat{M} = (a_1^{\hat{m}} a_2^{\hat{r}})^{r_2} = (a_1^{\hat{m}} a_1^{z_1 \hat{r}})^{r_2}$, and sends $\hat{M}$ to $\mathcal{S}$. $\mathcal{U}$ and $\mathcal{S}$ execute a NIZKPK

$$PK\{(\alpha, \beta) : \hat{M} = a_1^{\alpha} a_2^{\beta}\}$$

5. $\mathcal{S}$ produces a signature on $(m+\hat{m}, r+\hat{r}, t, d)$ as follows: Choose $\beta \overset{R}{\leftarrow} \mathbb{Z}_q$ and randomize all signature fields that do not depend on $\hat{m}$ or $\hat{r}$ (i.e., all fields except a_9) using β. $\mathcal{S}$ updates a_9 using $\hat{M}$ as in $\hat{\sigma} = (a_1^\beta, a_2^\beta, a_3^\beta, a_4^\beta, a_5^\beta, a_6^\beta, a_7^\beta, a_8^\beta, (a_9\hat{M}^{su})^\beta)$ and sends $\hat{\sigma}$ to $\mathcal{U}$.
6. $\mathcal{U}$ sets $m = m + \hat{m} \pmod q$, $r = r + \hat{r} \pmod q$, and $a_9 = a_9^{r_2^{-1}}$ and verifies that $\hat{\sigma}$ is a valid signature on (m, r, t, d).

6.2 Proving the validity of a subscription

The proof of knowledge given in step 2 of the Access protocol can be re-written as:

$$PK\{(\alpha, \beta, \gamma) : V = (v_r^{-1})^\alpha \cdot (v_d^{-1})^\beta \cdot v_{sig}^\gamma \wedge (\beta \geq t_{curr})\}$$

Its first part can be executed using a proof of knowledge of the discrete logarithm representation (see Appendix A), and the second part requires a range proof for a hidden exponent β. The latter can normally be accomplished by sending a commitment to the exponent and showing that the committed value lies within a specific interval $[a, b]$.

The most efficient range proof to date is due to Boudot [4], but it must be carried out using groups of unknown order (i.e., arithmetic is modulo $n = pq$, where p and q are not known to either the prover or verifier). Unfortunately, creating such a group during protocol execution and integrating the proof into our setting makes this approach impractical. Thus, we employ a classical bitwise proof [24] showing that a committed number is in the range $[0, 2^k - 1]$, which may be impractical in general, but works well in our case. To show that $d \geq t_{curr}$, it is sufficient to prove that $d - t_{curr}$ is a positive number of at most k bits long for some k of the service provider's choice. Then if d is not in the near future, the user will not be able to construct such a proof (i.e., $d - t_{curr} \mod q$ is then a large $|q|$-bit number). For this application, it is sufficient to set k to a small value, as long as 2^k exceeds the number of time slots in the largest subscription. For instance, if each slot t_i corresponds to a day, setting k to 9 will support yearly subscriptions.

Let $x = x_0 2^0 + x_1 2^1 + \ldots + x_{k-1} 2^{k-1}$ for $x_i \in \{0, 1\}$ and $i = 0, \ldots, k-1$ be the binary representation of x. Then the range proof for x is conducted by sending to the server $com(x) = g^x h^y$, and $com(x_i) = g^{x_i} h^{y_i}$ for $i = 0, \ldots, k-1$, where $com(x) = \prod_{i=0}^{k-1} com(x_i)$ and each y_i is chosen randomly from $\mathbb{Z}_q$, and showing that each x_i hidden in $com(x_i)$ is either 0 or 1. In other words, the exact statement that the user proves is of the form:

$$\begin{aligned} &PK\{(\alpha, \beta, \gamma, \delta, \delta_0, \ldots, \delta_{k-1}, \tau, \tau_0, \ldots, \tau_{k-1}) : \\ &\quad V = v_r^{-\alpha} \cdot v_d^{-\beta} \cdot v_{sig}^\gamma \wedge A = g^\delta h^\tau \wedge A = \prod_{i=0}^{k-1} g^{\delta_i} h^{\tau_i} \text{ s.t. } \delta_i \in \{0, 1\} \wedge \beta - \delta = t_{curr}\} \end{aligned}$$

We give more details on how exactly this proof is constructed in Appendix A.

6.3 Security analysis

The security of our solution heavily relies on the security of CL signature scheme and proofs of knowledge used. In particular, unforgeability and anonymity requirements are fulfilled by the properties of the signature scheme. Let us examine each of the required security properties in more detail.

The correctness property is straightforward and can be shown by examination. The Subscribe protocol consists of two main elements: issuing a signature on a committed message and a ZKPK, correctness of both of which has been previously demonstrated. Correctness of the Access protocol also mostly due to the properties of the signature scheme, and the only part that we need to show is that a newly formed token which the user obtains at the end of step 6 corresponds to a valid signature on $(m + \hat{m}, r + \hat{r}, t, d)$. The token $\mathcal{U}$ has at the end is: $(a_1^\beta, a_2^\beta, a_3^\beta, a_4^\beta, a_5^\beta, a_6^\beta, a_7^\beta, a_8^\beta, (a_9\hat{M}^{su})^{\beta r_2^{-1}}) =$

$(a^{r_1\beta}, a^{z_1r_1\beta}, a^{z_2r_1\beta}, a^{z_3r_1\beta}, a^{ur_1\beta}, a^{uz_1r_1\beta}, a^{uz_2r_1\beta}, a^{uz_3r_1\beta}, (a^{(s+su(m+z_1r+z_2t+z_3d))r_1r_2}a^{su(\hat{m}+z_1\hat{r})r_1r_2})^{r_2^{-1}}) = (b, b^{z_1}, b^{z_2}, b^{z_3}, b^u, b^{uz_1}, b^{uz_2}, b^{uz_3}, b^{s+su(m+\hat{m}+z_1(r+\hat{r})+z_2t+z_3d)})$, where a is the base of $\mathcal{U}$'s previous token and $b = a^{r_1\beta}$ is the base of the newly formed signature.

Next, we proceed with the soundness and anonymity properties of the solution. The Subscribe protocol involves issuing a CL signature on a hidden value, the security of which has been shown in [13]. The random value r perfectly hides the message m, and the server does not learn any information about the value the user encodes in the signature. Then in the Access protocol, the user is required to prove the validity of its token which is not possible without having a token issued by the server. Also, the user will not be able to reuse one of the old token since the server records all previously used tokens. The anonymity property in the Access protocol is fulfilled by (i) making sure that the previous token is randomized (i.e., the server cannot link a token to one of its previously issued tokens) and (ii) the server does not learn any information about the messages encoded in the new token it is creating.

7 Implementation and Performance Evaluation

The client and server side of the protocols were implemented as C++ programs, the source code of which is available at `http://anonymized`. The server listens to incoming connections and accepts subscription requests from new clients, as well as access requests from existing clients. A client can execute either the subscribe protocol to obtain its first token or the access protocol using its current token. The experiments were performed on a 2GHz Power Mac G5 running Mac OS X 10.4.9.

We use the Miracl [29] library to perform big number operations. Miracl provides efficient mechanisms for computing elliptic curve and pairing operations required by our protocols. The protocols were implemented using subgroups of elliptic curves with pairings where the decisional Diffie-Hellman (DDH) problem is hard. According to [22], this is the most efficient and versatile type of pairings. Such groups are asymmetric meaning that the pairing function is now of the form $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_2$ is an extension field of $\mathbb{G}_1$. For our protocols this implies that certain parameters will be elements of $\mathbb{G}_1$ and others will be elements of $\mathbb{G}_2$. For efficiency reasons, we chose the signature elements to lie in $\mathbb{G}_1$ and most values in the server's public key (g, g^s, g^u, etc.) to be in $\mathbb{G}_2$.

The programs were built using DDH-hard subgroups of an MNT elliptic curve with pairings, and for simplicity we used a pre-generated curve provided with the Miracl library. The groups have prime order q, where q is 157 bits long, and the curve has an embedding degree of 6.

Basic protocol measurements. All of our protocols use a small number of point multiplications on elliptic curves as well as pairing operations, with the latter being the most expensive computation. Miracl permits us to use two types of coordinates: Affine and Projective. We first ran the protocols using both of these coordinates to determine which type will result in faster performance. The computation times corresponding to different parts of the Subscribe and Access protocols are shown in Table 1. The parts of the protocols that we measured are:

Client subscribe: (i) construction of a commitment to a message and a ZKPK that the commitment is well formed and (ii) verification of the validity of the authentication token received from the server.

Server subscribe: (i) verification of the client's ZKPK and (ii) construction of an authentication token for the client.

Client access: (i) construction of a complex ZKPK for the signature and the expiration date, (ii) construction of a commitment to a new message and a ZKPK that the commitment is well formed, and (iii) verification of the validity of the new authentication token received from the server.

Server access: (i) verification of the client's complex ZKPK, (ii) verification of the client's ZKPK of the commitment, and (iii) construction of a new authentication token.

The Access protocol was implemented using the range proof of 15 bits for the expiration date. These

	New subscription			Existing subscription			
Client	construction of ZKPK	signature verification	total	construction of complex ZKPK	construction of ZKPK	signature verification	total
affine	11.43 ms	889.27 ms	900.70 ms	720.86 ms	28.24 ms	931.12 ms	1680.22 ms
projective	4.75 ms	897.24 ms	901.99 ms	505.48 ms	10.57 ms	912.91 ms	1428.96 ms
Server	verification of ZKPK	signature construction	total	verification of complex ZKPK	verification of ZKPK	signature construction	total
affine	16.33 ms	152.85 ms	169.18 ms	672.73 ms	15.99 ms	56.78 ms	745.50 ms
projective	7.02 ms	117.85 ms	124.87 ms	486.24 ms	6.89 ms	23.30 ms	516.43 ms

Table 1: Performance of cryptographic elements of the Subscribe and Access protocols using Affine and Projective coordinates.

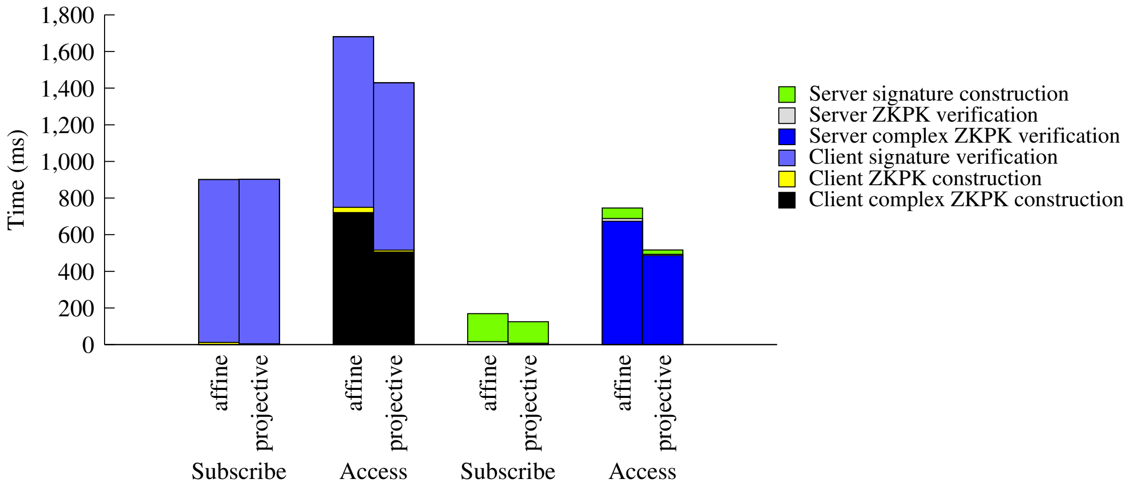

Figure 1: Client (left) and server (right) performance using Affine and Projective coordinates.

performance results are also plotted in Figure 1 and demonstrate that the largest overhead is due to signature verification (used in both Subscribe and Access) and complex ZKPK construction and verification (used in Access) because of expensive pairing operations.

All numbers shown in this section correspond to the average time computed over 100 executions of the protocols with varying values for subscription types and expiration dates. The times do not include delays due to communication, as our goal is to show performance overheads associated with using cryptographic anonymity techniques.

As Figure 1 illustrates, the use of Projective coordinates generally resulted in superior performance. It appears that performance of signature verification that mostly consists of pairing computations was insignificantly affected by the change, and the biggest difference is seen in performance of point multiplication operations on the curve. Because of faster performance of Projective coordinates, all other experiments were run using only this type.

Varying bitlengths in the range proof. The next experiment we performed was to see how changing the bit length of the range proof in the Access protocol affects the performance of the protocol. Figure 2 shows the computation required to construct and verify the complex ZKPK in the protocol as a function of the number of bits in the range proof (left) and the overall computation of the protocol for the client and the server as a function of the number of bits (right). Since the computation involved in a subscribe request is not affected, its performance is not included in the figures and can be found in Table 1.

An interesting observation here is that increasing the bitlength of the range proof does not significantly

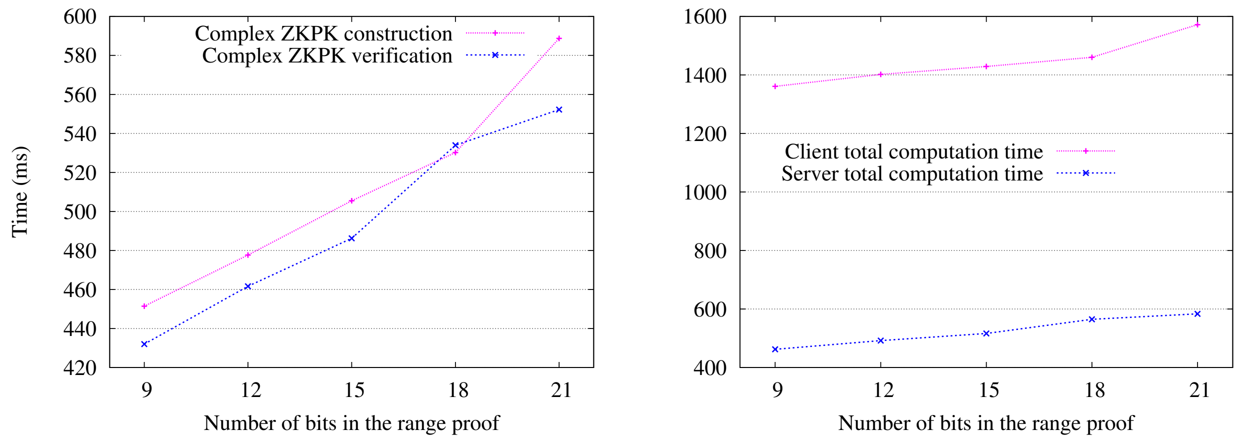

Figure 2: Computation time for the complex ZKPK and the total computation time per access.

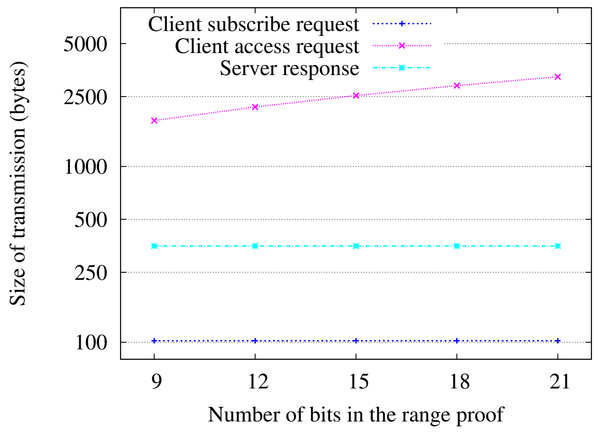

Figure 3: Size of transmission by client and server for new and existing subscriptions.

affect the performance of the protocol, and the pairing evaluations used in verifying the validity of a signature still outweigh the other parts of the protocol. Zero-knowledge proofs are traditionally considered to be computationally expensive, but in our case they do not constitute the majority of the protocols' overhead.

Communication requirements. Even though the signature scheme we utilize in the protocol results in compact authentication tokens, we were interested in measuring the total communication requirements of the protocols, especially with a varying number of elements in the range zero-knowledge proof. Thus, we next report on communication requirements of the protocols.

Points on an elliptic curve in $\mathbb{G}_1$ are represented as two $|q|$-bit coordinates x and y. It is possible to store just the first coordinate x and, given the curve parameters, compute the second one y. This, however, increases the computation which is undesirable. Since in our case the messages transmitted are short and we are interested in reducing computational load on both the client and the server, we store (and transmit) both coordinates. Thus, the numbers we present can as high as twice that of the theoretical bounds, but our goal here is to report the actual performance results. Similarly with the extension field $\mathbb{G}_2$, each point now is represented as 2 coordinates of $6|q|$ bits each.

Figure 3 shows the amount of data that needs to be transmitted by a client and the server during execution of the protocols. Since the bases in ZKPKs are known to both parties, they do not need to be transmitted and are not accounted for in the plots. As the figure shows, the server's response has identical size for new and

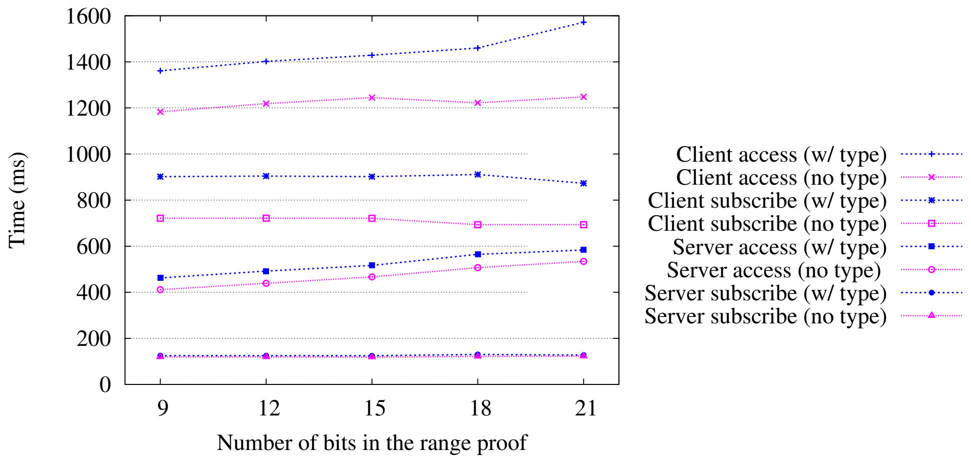

Figure 4: Performance comparison of the original protocols and the protocols without subscription type.

existing subscriptions (a single signature is sent). The size of the client access request only slightly increases with the increase in the number of bits in the range proof, and the largest transmission depicted on the graph does not exceed 3KB. This means that all transmissions are very short and there is no need to worry about reducing their size.

Protocols without specifying subscription type. Since some subscription services might provide only a single type of access to all of the customers, there will be no need in including the type t in the authentication protocols. This means that we will be able to shorten the signature and lower the computation associated with the protocols. When the type is not used, this change might be particularly desirable because signature verification constitutes a large part of the computation overhead.

To evaluate how significantly removal of the subscription type from the signature affects the performance of the protocols, we excluded the type and re-ran the experiments. Comparison of the performance of the protocols with and without subscription type is depicted in Figure 4. As expected, the change most significantly influenced the signature verification time (in client Subscribe and Access protocols), and resulted in lowering the computation overhead by 20–25%. It also had a noticeable impact on the server's time necessary to process client access requests. Processing of client subscribe requests, however, did not result in significant lowering of the server's computation time (the difference is about 6ms on average) because of the efficiency of point multiplications in $\mathbb{G}_1$.

Practical considerations. The proper functioning of the system relies on the clients choosing their secret values m at random. Should two clients use the same randomness while initializing the cryptographic software, access to the service might be denied to one of them if the same message m is used by both of them. Thus, in this application it is crucial for each client to use an adequate source of randomness to ensure correct operation of the system.

8 Extensions and Conclusions

In our scheme, there were two parameters to each subscription: type t and expiration date d. At the subscription time both of them were known to the service provider, while at the access time one was open while the other remained hidden. In general, subscriptions might depend on a different number of parameters, some of which are to be hidden from the service provider at subscription time, access time, or both. Thus, by varying the number of hidden and open parameters (while enforcing required constraints on them using

ZKPKs), the approach could be used with a wider range of applications. Chaining of user tokens in our case gives the service provider a higher level of protection than before since authentication tokens cannot be shared or duplicated by dishonest users.

Since an extension to a varying number of parameters is rather straightforward, we do not list full details of how this is accomplished, but only comment on certain aspects of it. As an example, consider a system where a user is permitted to select a certain number of categories from the list of topical categories available, without the server knowing which categories the user chose. Then at the time of subscribing, the user will send a commitment to a number of hidden attributes $p_1, \ldots, p_k$ (protected by a random value r) and execute a number of ZKPKs on these attributes. Similarly, at the time of access the user sends in clear attributes that are to be opened (these are incorporated into the signature verification proof) and possibly proves statements in ZK about other attributes that remain hidden. Thus, it is possible to accommodate a wide range of possible subscription types and policies associated with such services using the approach offered in this work.

To summarize, this work gives the design and implementation of a system that allows users to anonymously access services included in their subscription. One of its compelling features is that, despite being anonymous, users are unable to abuse the system. Additionally, our empirical results indicate that computational requirements due to the cryptographic protocols are low enough to be supported by today's services.

9 Acknowledgments

The author is grateful to Giuseppe Ateniese for useful discussions at early stages of this work. The work is supported by Intel Ph.D. fellowship.

References

[1] G. Ateniese, D. Song, and G. Tsudik. Quasi-efficient revocation of group signatures. In *Financial Cryptography (FC'02)*, pages 183–197, 2002.

[2] M. Backes, J. Camenisch, and D. Sommer. Anonymous yet accountable access control. In *ACM Workshop on Privacy in the Electronic Society (WPES'05)*, pages 40–46, 2005.

[3] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology – CRYPTO'04*, volume 3152 of *LNCS*, pages 41–55, 2004. Full version availalbe at `http://crypto.stanford.edu/ dabo/abstracts/groupsigs.pdf`.

[4] F. Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology – EUROCRYPT'00*, volume 1807 of *LNCS*, pages 431–444, 2000.

[5] S. Brands. A technical overview of digital credentials. Unpublished manuscript. Available from `http://www.credentica.com/whitepapers.php`.

[6] S. Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, Centre for Mathematics and Computer Science (CWI), 1993.

[7] S. Brands. Untraceable off-line cash in wallets with observers. In *Advances in Cryptology – CRYPTO'93*, volume 773 of *LNCS*, pages 344–359, 1993.

[8] E. Bresson and J. Stern. Efficient revocation in group signatures. In *International Workshop on Practice and Theory in Public Key Cryptography (PKC'01)*, volume 1992 of *LNCS*, pages 190–206, 2001.

[9] E. Bresson and J. Stern. Proofs of knowledge for non-monotone discrete-log formulae and applications. In *Information Security Conference (ISC'02)*, volume 2433 of *LNCS*, pages 272–288, 2002.

[10] J. Camenisch and J. Groth. Group signatures: Better efficiencey and new theoretical results. In *Conference on Security in Communication Networks (SCN'04)*, volume 3352 of *LNCS*, pages 120–133, 2005.

[11] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clone wars: Efficient periodic n-times anonymous authentication. In *ACM Conference on Computer and Communications Security (CCS'06)*, pages 201–210, 2006.

[12] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology – CRYPTO'02*, volume 2442 of *LNCS*, pages 61–76, 2002.

[13] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO'04*, pages 56–72, 2004.

[14] J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *LNCS*, pages 107–122, 1999.

[15] J. Camenisch, D. Sommer, and R. Zimmermann. A general certification framework with applications to privacy-enhancing certificate infrastructures. In *IFIP International Information Security Conference (SEC'06)*, May 2006.

[16] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology – CRYPTO'97*, volume 1296 of *LNCS*, pages 410–424, 1997.

[17] J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical Report No. 260, ETH Zurich, 1997.

[18] D. Chaum, J.-H. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *Advances in Cryptology – EUROCRYPT'87*, volume 304 of *LNCS*, pages 127–141, 1988.

[19] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology – CRYPTO'94*, volume 839 of *LNCS*, pages 174–187, 1994.

[20] A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung. On monotone formula closure in SZK. In *Symposium on Foundations of Computer Science (FOCS'94)*, pages 454–465, 1994.

[21] X. Ding, G. Tsudik, and S. Xu. Leak-free group signatures with immediate revocation. In *International Conference on Distributed Computing Systems (ICDCS'04)*, pages 608–615, 2004.

[22] S. Galbraith, K. Paterson, and N. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. `http://eprint.iacr.org/2006/165`.

[23] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography (SAC'99)*, pages 184–199, 1999.

[24] W. Mao. Guaranteed correct sharing of integer factorization with off-line shareholders. In *Public Key Cryptography*, pages 27–42, 1998.

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

[26] P. Persiano and I. Visconti. A secure and private system for subscription-based remote services. *ACM Transactions on Information and System Security (TISSEC)*, 6(4):472–500, November 2003.

[27] Z. Ramzan and M. Ruhl. Protocols for anonymous subscription services. Unpublished manuscript, 2000.

[28] S. Schechter, T. Pernell, and A. Hartemink. Anonymous authentication of membership in dynamic groups. In *International Conference on Financial Cryptography (FC'99)*, pages 184–195, 1999.

[29] M. Scott. MIRACL library. Indigo software, `http://indigo.ie/ mscott`.

[30] D. Song. Practical forward secure group signature schemes. In *ACM Conference on Computer and Communications Security (CCS'01)*, pages 225–234, November 2001.

[31] S. Stubblebine, P. Syverson, and D. Goldschlag. Unlinkable serial transactions: Protocols and applications. *ACM Transactions on Information and System Security (TISSEC)*, 2(4):354–389, November 1999.

[32] I. Teranishi, J. Furukawa, and K. Sako. k-times anonymous authentication. In *ASIACRYPT'04*, volume 3329 of *LNCS*, pages 308–322, 2004.

A Zero-Knowledge Proofs of Knowledge

In what follows, $H : \{0,1\}^* \to \{0,1\}^\kappa$ is a collision resistant hash function that maps the binary representation of its argument to a string of a fixed length κ, and $||$ denotes concatenation.

A NIZKPK of the discrete logarithm of $M = g^x$ to the base g is performed as follows:

1. $\mathcal{U}$ chooses $v \overset{R}{\leftarrow} \mathbb{Z}_q$ and computes commitment $t = g^v$.
2. $\mathcal{U}$ computes the challenge $c = H(g||M||t)$.
3. $\mathcal{U}$ computes the response $r = v - cx \pmod q$ and sends (c, r) to $\mathcal{S}$.
4. $\mathcal{S}$ computes $T = M^c g^r$ and checks $c \overset{?}{=} H(g||M||T)$.

This proof of knowledge (PK) above can be viewed as a special case of a PK of the discrete logarithm representation. The knowledge of the discrete logarithm representation of M to the base $(g_1, \ldots, g_n)$ is the knowledge of $(a_1, \ldots, a_n)$ such that $M = g_1^{a_1} g_2^{a_2} \ldots g_n^{a_n}$ holds. The NIZKPK of the discrete log representation then proceeds similar to the above proof with the following differences: $\mathcal{U}$ first chooses $v_1, \ldots, v_n \overset{R}{\leftarrow} \mathbb{Z}_q$. $\mathcal{U}$ computes the commitment $t = g_1^{v_1} \ldots g_n^{v_n}$, the challenge $c = H(g_1||\ldots||g_n||M||t)$, and the response $r_i = v_i - ca_i \pmod q$ for $1 \le i \le n$, and sends $(c, r_1, \ldots, r_n)$ to $\mathcal{S}$. To verify, $\mathcal{S}$ computes $T = M^c g_1^{r_1} \ldots g_n^{r_n}$ and checks whether $c = H(g_1||\ldots||g_n||M||T)$.

Next, we describe a proof that a committed value is either 0 or 1 [19, 20] (used in the Access protocol to show the validity of the expiration date). Recall that $M = com(x_i)$ is h^{y_i} if $x_i = 0$, and it is gh^{y_i} otherwise. The user proves that she knows either discrete log of M to base h or discrete log of M/g to the same base h. If for concreteness we let $x_i = 1$, then the proof proceeds as follows:

1. $\mathcal{U}$ chooses $v_1, v_2, w \overset{R}{\leftarrow} \mathbb{Z}_q$ and computes commitments $t_1 = M^w h^{v_1}$ and $t_2 = h^{v_2}$.
2. $\mathcal{U}$ computes the challenge $c = H(g||h||M||t_1||t_2)$.
3. $\mathcal{U}$ sets $c_1 = w$ and $c_2 = c - c_1 \pmod q$ and then computes the response $r_1 = v_1$, $r_2 = c_2 - v_2 y_i \pmod q$ and sends (c, r_1, r_2) to $\mathcal{S}$.

4. $\mathcal{S}$ computes $T_1 = M^{c_1}h^{r_1}, T_2 = (M/g)^{c_2}h^{r_2}$ and checks $c_1 + c_2 \overset{?}{=} H(g||h||M||T_1||T_2) \pmod q$.

Finally, the last condition that a user needs to prove is that the date in the bitwise range proof equals to the expiration date built into her token minus the current date. This is accomplished by using a proof for linear relationship between exponents in different commitments (see [17] for more information). More precisely, given $V = v_r^{-r} v_d^{-d} v_{sig}^{r_2^{-1}}$ and $A = g^x h^y$ where $x = d - t_{curr}$, it is conducted as follows:

1. $\mathcal{U}$ chooses $w_r, w_d, w_{sig}, w_h \overset{R}{\leftarrow} \mathbb{Z}_q$, sets $w_g = q - w_d$, and computes commitment $t_1 = v_r^{w_r} v_d^{w_d} v_{sig}^{w_{sig}}$ and $t_2 = g^{w_g}h^{w_h}$.
2. $\mathcal{U}$ computes the challenge $c = H(v_r||v_d||v_{sig}||V||t_1||g||h||A||t_2)$.
3. $\mathcal{U}$ computes the response $r_r = w_r - cr \pmod q$, $r_d = w_d - cd \pmod q$, $r_{sig} = w_{sig} - cr_2^{-1} \pmod q$, $r_g = w_g - cx \pmod q$, and $r_h = w_h - cy \pmod q$; then sends $(c, r_r, r_d, r_{sig}, r_g, r_h)$ to $\mathcal{S}$.
4. $\mathcal{S}$ computes $T_1 = V^c v_r^{r_r} v_d^{r_d} v_{sig}^{r_{sig}}, T_2 = A^c g^{r_g} h^{r_h}$ and checks $c \overset{?}{=} H(v_r||v_d||v_{sig}||V||T_1||g||h||A||T_2)$ and $r_d + r_g \overset{?}{=} ct_{curr} \pmod q$.