Protocols for Anonymous Subscription Services

Zulfikar Ramzan Matthias Ruhl

MIT Laboratory for Computer Science
Cambridge, MA 02139, USA
{zulfikar,ruhl}@theory.lcs.mit.edu

November 6, 2000

Abstract

In this paper we discuss protocols that allow a user to subscribe to an electronic service, and then anonymously access the service. That is, neither the service provider nor anyone else knows who accesses the service at any time, and moreover no one can link two accesses to the same person. On the other hand, the provider obtains proof that the user is authorized to use the service.

We formally define the problem and discuss the security features these protocols should have. An important property for a protocol is termination: the access privileges can be used only a fixed number of times. In this paper, we state and analyze two practical schemes which have this property while maintaining unconditional anonymity and unlinkability. The protocols also achieve lower storage and communication requirements than related schemes. In our first protocol, the vendor signs blinded access tokens, while in the second protocol, the client is given limited signing capabilities to create his own access tokens. The security analysis of the second protocol includes identifying a new equivalent variant of the Decisional Diffie-Hellman security assumption, which may be of independent interest.

1 Introduction

The Internet is becoming ubiquitous in our daily lives and with this trend more and more opportunities are becoming available for gathering essential information or obtaining goods and services online. In many cases, the premium information and services on a site are restricted to paid subscribers. Unfortunately, it is extremely easy for the provider to monitor the online activities of a user, and as a result, sensitive information about the subscriber's reading habits may be exposed; this extra knowledge can often be exploited to the severe detriment of the customer.

Consider, for example, the Forrester research or IBM patent server web sites. These sites are heavily used by many companies. Once you become a registered user, you gain access to a large volume of information. On the flip side, it is feasible for Forrester Research and IBM to determine who you are. And even worse, by examining your reading habits, they may be able to infer your company's corporate strategy and the new markets you are considering. Imagine if this information found its way to your competitors. Such user tracking can also occur when subscribing to newspapers, magazines, stock databases, pay per view movies, and many other resources. The behavior of users might be aggregated and used in a potentially malicious manner.

Motivated by these examples, we have undertaken a study of *anonymous subscription protocols* which allow a user to purchase a subscription to a service, and then use the service anonymously. That is, nobody, not even the provider, knows who accesses the service at any time, and moreover no one can link two accesses to the same person. On the other hand, the provider is guaranteed that only legitimate users access the service, and that a user cannot use the service more often than he paid for (for example if a subscription was good for 30 accesses, then the user cannot make more than 30 accesses to the system).

In addition, we provide a protocol for the more general problem where items can also be returned. This is called a "rental scheme", and an important example are site-licenses for software. Here, a company pays a fixed price and, in exchange, its employees can concurrently use up to a certain number of copies of a particular software product. Other rental examples are video rentals, or library checkouts.

Conceptual Contributions. We give a detailed definition and analysis of the anonymous subscription problem. We provide a list of the required sub-protocols and mention the desired security features associated with anonymous subscriptions. One important feature besides anonymity and unlinkability is *termination*. Subscription protocols given in the literature often do not have a mechanism to limit the number of accesses to a subscription [SSG97, SPH99]. This feature, however, is useful in many practical applications.

Technical Contributions. In this paper, we give two protocols for unconditionally anonymous subscription that include such an implicit termination mechanism, and are efficient in terms of storage and communication complexity.

The first protocol, called the *bit counting scheme*, uses blind signatures [Cha83] to achieve anonymity. In this scheme, the user keeps track of the remaining subscription length, and shows only part of that information to the supplier when accessing the service. We show that the supplier learns only an insignificant amount of information about the customer, and can link two transactions with only a negligible probability. In this scheme, the initial subscription length can be set to an arbitrary value, and we can modify the protocol to support rental schemes where items can be returned.

The second protocol is based on *client created tokens.* Here the user is given limited signing capabilities and can create a pre-defined number of access tokens. We instantiate the protocol by making several crucial modifications to a known group signature scheme [CS97]. Group signature schemes usually involve a trusted party who can revoke anonymity, and thus offer only conditional anonymity. The challenge lies in removing the revocation authority and in making the signatures deterministic to detect multiple spending without compromising the other security features. Since our subscription scheme has a different trust model than group signature schemes, we need an additional security analysis. To prove that our scheme remains secure we introduce the *Promised Decisional Diffie-Hellman* assumption, which may be of independent interest. We prove that it is computationally equivalent to the well-known *Decisional Diffie-Hellman* assumption. In this second protocol, the subscription length is the same for every subscriber, and it cannot easily be used for rental schemes. Its communication complexity and storage overhead are even better than for the bit counting scheme.

Outline of Paper. The remainder of this paper is structured as follows. In section 2, we formally define the anonymous subscription problem, and discuss in more detail the properties that we demand of these schemes. We discuss related work in section 3. In sections 4 and 5 we present and analyze our two new protocols for anonymous subscription services. Finally, we give a concluding discussion and state open research problems in section 6.

2 Problem Specification

2.1 Protocols

Three main parties are involved in an anonymous subscription scheme: the customer **C**, who wants to access the system; the vendor **V**, who sells the subscription, or in general grants access privileges; and the supplier **S**, who actually delivers the goods to the customer **C**. In many practical instances, the vendor and supplier might be the same person, though they need not be.

Apart from an initial setup mechanism, there are two main protocols in a subscription scheme:

Open Account / Subscribe: First, **C** and **V** agree on a duration ℓ of the subscription. Following this, the customer **C** sends the payment and a proof of identity to **V**. The vendor **V** then sends an authorization certificate to **C**. This certificate will allow **C** to access the system at most ℓ times.

Receive Item: The customer **C** computes an access token from his authorization certificate, and sends it to **S**. If the token is valid, and has not been used before, **S** sends the requested item and a new certificate to **C**. The customer then incorporates this new certificate into his current authorization certificate so it can be used for the next transaction.

In some circumstances it might be desirable to have additional protocols for the early termination of the service (by the vendor or the customer). In the case of rental scheme, we also need a protocol that allows the client to return an item.

2.2 Properties

There are clearly many ways to instantiate the above procedures to obtain a system for handling subscriptions. But it becomes considerably harder when we impose additional constraints on security, memory, computation and bandwidth. We now discuss the properties these schemes should have in order to be interesting or useful in practice. We assume that all schemes are correct; i.e. that if all parties follow the protocols, the user is able to obtain exactly the specified number of items.

Security properties

1. **Anonymity.** No one can determine a customer's identity when the customer accesses the system.
2. **Unlinkability.** No one can determine whether two accesses belong to the same customer.
3. **Unforgeability.** A customer cannot forge tokens to obtain more than the specified number of items.
4. **Coalition-Resistance.** Even if several customers collude (by sharing their tokens/secret keys etc.) they cannot obtain more items than they paid for as a group. I.e., it is infeasible to forge new tokens based on the combined information of several people.

System properties

1. **Termination / Limited Access.** The number of item transactions is limited. That is, the user is allowed to access the system only a specified number of times, or during specific time frames (or a combination of both). After that his access privileges (implicitly, i.e. without any communication between the parties) terminate.
2. **Unshareable.** It should be highly inconvenient to share a single subscription among two or more customers.
3. **Early Termination.** A user can terminate the service at any time, and obtain a refund for the unused part of the subscription.

Performance

1. **Storage Requirements.** How much information customer and supplier have to store to make the protocol possible.
2. **Communication complexity.** Refers to the amount of data exchanged during a transaction (ignoring the transmitted item itself). This is usually measured in terms of n (the number of users in the system), and k (the security parameter, i.e. public/private key size). We distinguish between the first transaction (to open an account), and later transactions to obtain items.
3. **Time complexity.** Time it takes to perform computations for customer and supplier.

3 Related Work

There are three main approaches to anonymous subscription (or similar) problems in the literature (see also Table 1 on page 18).

Unlinkable Tokens. The first approach is to use unlinkable tokens for accessing the service. For example, using a digital cash scheme [Cha83], the user can receive ℓ dollars from the vendor, and spend one dollar each time the service is accessed. The problem with this direct approach is that the storage requirements for the user are proportional to the subscription length, and the subscription is easily sharable among many users.

These problems were overcome by [SSG97]. Here a user receives only one blinded token when he opens an account. During the transaction phase, the user hands in his current token, and receives another blinded token for the next transaction. The supplier remembers all the tokens he has seen to prevent double-spending. In this scheme, however, the subscription length can not be bounded.

Brands [Bra00] proposed a scheme based on limited-show certificates. These certificates encode the owner's identity, and are designed so that the owner's identity is revealed if the certificates are used more than a certain number of times. The drawbacks of this scheme are the same as for e-cash, and additionally transactions are linkable.

Group signatures. Group signature schemes [CvH91, CS97, ACJT00] can be used for subscription services, where subscribing to a service corresponds to joining a group. The right to access a service is proven by signing a given message on behalf of the group. Such an application of group signatures to the subscription problem was recently studied in [NHS99].

The main problem with the application of group signatures in general (and also in [NHS99]) is the fact that the group manager (or a designated revocation authority) is able to revoke the anonymity of a client. Moreover, since the schemes are randomized and unlinkable, there is no effective way of limiting the length of a subscription without recovering the identity of users. It is non-trivial to derandomize these schemes and make them unconditionally anonymous, so that no one, not even the revocation authority, can link accesses.

Commonly Verifiable Secrets. The protocol described in [SPH99] uses a broadcast public-key encryption mechanism based on a *deterministic* encryption scheme such as RSA. The vendor encrypts a common secret value x under the public keys of all subscribers and gives the concatenated encryptions to the user. In order for a user to authenticate himself, he must tell the vendor what x is. This scheme has an unreasonable bandwidth requirement, as every authentication requires a transmission whose size is linear in the number of subscribers. Moreover, the subscription length is not bounded.

We also mention two other protocols, which are not subscription schemes per se, but can be used in conjunction with subscription schemes to obtain additional security properties.

Identity Escrow. Escrowed identity schemes, which were first formally studied in [KP98], allow a user to authenticate himself anonymously and unlinkably. There is, however, a trusted third party who, together with the vendor, can revoke the user's anonymity should that become necessary. We could incorporate identity escrow into our schemes to implement an anonymity revocation protocol.

Private Information Retrieval (PIR). PIR schemes, which were first formally studied in [CGKS95], consist of two parties: a user and a database. The database possesses a secret string $B = b_1 b_2 \ldots b_n$, and the user has a secret index i between 1 and n. The two engage in a protocol during which the user learns b_i in a communication-efficient way while the database learns nothing about i. This can be combined with anonymous subscription schemes so that the supplier knows neither who accessed the service nor what was accessed.

4 Bit counting scheme

Our first protocol for anonymous subscriptions is the *bit counting* scheme. In this scheme, the user stores a number of anonymous tokens which encode the binary representation of the remaining length of his subscription. When accessing the service, he does not transmit all tokens, but just a subset that allows the supplier to verify that the subscription has not ended yet, and to decrease the counter by one by sending new tokens to the customer. The supplier keeps track of all spent tokens to prohibit double spending. In this scheme, the initial subscription length can be set to an arbitrary number, and since it is easy to increase the counter in a similar manner, the scheme can also be used for rental schemes.

4.1 Description of Protocol

System Setup. When setting up the protocol, the vendor establishes the longest possible subscription duration. More precisely, he fixes a number m such that $T := 2^m - 1$ will be the maximum subscription length (or maximum number of items to rent, if it is a rental scheme). He then chooses $2m$ public/secret-key pairs. We name the keys pairs $t_1, t_2, \ldots, t_m$ and $f_1, f_2, \ldots, f_m$. Note that the signature scheme must support blinded signatures, as we will use these crucially in the following.

Open account / Subscribe. When a customer begins a subscription that lasts for ℓ items, he sends the vendor m blinded tokens. The vendor signs these tokens as follows: for all $i \in \{1, \ldots, m\}$, he signs the i-th token using key t_i if the i-th bit in the binary representation of ℓ is 1, and using key f_i if the bit is 0. The signed tokens are then returned to the customer.

During the execution of the protocol, the customer will always be in possession of m signed tokens. The number ℓ of items left in the subscription will be encoded by which keys have been used to sign the tokens. The customer will have one token signed with either t_1 or f_1, depending on whether the lowest order bit of ℓ is 1 or 0, respectively. He will have one token signed with t_2 or f_2 depending on whether the second lowest order bit is 1 or 0, and so on. This immediately implies that the customer has a token signed with some t_i if and only if the subscription has not yet terminated.

Receive item. When requesting an item, the customer first determines the lowest i such that he has a token signed with t_i. He then sends i tokens to the supplier: the ones signed with $f_1, f_2, \ldots, f_{i-1}, t_i$. In addition, he sends the supplier i new blinded tokens. The supplier checks that the signatures on the first i tokens are valid, and that these tokens have not been used before. If these requirements are met, he sends the item to the customer. Also, he signs the i new tokens using keys $t_1, t_2, \ldots, t_{i-1}, f_i$, and sends them to the customer. Note that after this round, the number of items left for the customer to receive, as encoded by the tokens he has, has decreased exactly by one.

Return item. In a rental scheme, if the customer wishes to return an item, he finds the smallest i for which he has a token signed with key f_i, and sends the i tokens signed with keys $t_1, t_2, \dots, t_{i-1}, f_i$ and i new blinded tokens to the supplier. The supplier checks validity as above, and signs the new tokens with keys $f_1, f_2, \dots, f_{i-1}, t_i$, and sends them to the customer. This corresponds to increasing the encoded subscription length ℓ by one.

4.2 Performance Analysis

On average, the customer transmits two tokens per item request ($1/2$ of the transactions involve one token, $1/4$ involve two, $1/8$ involve three, and so on; so the average number of transmitted tokens is at most $\sum_{i=1}^{\infty} i/2^i = 2$). Thus, the communication complexity is optimal: it is amortized $O(k)$, where k is the security parameter.

The storage complexity on the customer side is $O(k \log \ell)$, since we always have to keep $\log \ell$ signed tokens around. This is not quite optimal, but usually $\log \ell$ is small, and therefore this is almost $O(k)$ in practice.

The supplier's storage requirements are evidently greater, since he has to store all tokens that have been cashed in so far. To avoid that this number becomes too large, vendor and supplier should change their keys at regular intervals. There will obviously be a transition period in which tokens signed with the old keys are still accepted, but only tokens signed with the new keys will be returned.

If all $2m$ keys are expired at regular intervals, say after the average completion time of a subscription, then the server will have to store $O(n\ell)$ tokens. If key expiry should be kept at a minimum, for example because the associated broadcast of the new keys is very expensive, then the following observation is helpful. Since half of the stored tokens will be tokens signed with t_1 or f_1, $1/4$ are signed with t_2 or f_2, $1/8$ with t_3 or f_3, and so on, the storage per key-pair t_i, f_i can be kept constant as follows. We expire keys t_1 and f_1 twice as often as t_2 and f_2, and these in turn twice as often as t_3 and f_3, and so on. Then, on average, we expire four keys at a time.

4.3 Security Analysis

For the following analysis we make two assumptions, which can reasonably be expected to hold in practice. First, we assume that the number of subscribers n is much greater than the maximum subscription length $T = 2^m - 1$. Second, we assume that on average the users have the same total subscription length, and access the service with the same frequency. This implies that at every given point in time, there will be a similar number of users that have each possible remaining subscription length. In other words, the number of remaining issues for a user is equally likely to be any number between 1 and the maximal length.

Single Tokens. During a transaction, the supplier learns two things. First, he sees the tokens used in the transactions, and second, he knows how many tokens were transmitted. We claim that the supplier learns nothing from the tokens themselves, and only very little from the number of tokens.

Claim 4.1 *The supplier learns no information from seeing individual tokens.*

This is true because the tokens were chosen at random by the subscriber, so contain no information linking them to each other or to the user. Moreover, since the tokens were signed blindly, the supplier cannot link the signing process to seeing the unblinded tokens in a transaction. Thus, he does not learn any information from seeing the tokens.

Anonymity. If a transaction contains t tokens, the supplier learns that the number of issues left in that customers subscription is equal to 2^{t-1} modulo 2^t. This, however, is true for $n/2^t$ of all users, since the subscription lengths are uniformly distributed. This leads to the following fact.

Claim 4.2 *If a transaction contains t tokens, then the supplier on average learns t bits of information about the user; i.e., if there are n subscribers, the user is only known to belong to a certain subset of $n/2^t$ users.*

Note that on average, the user transmits only two tokens, so can be linked only to a group of size at least $n/4$, an acceptable security risk. Even in the worst case, being known to belong to a subset of n/T subscribers is not that problematic, since $n \gg T$. And in general, the supplier might not even be able to determine the sets of subscribers with similar numbers of remaining issues, since the accesses to the service are too random. This only adds to the security of the scheme.

Linkability. We now analyze the probability that the supplier is able to determine that two different transactions belong to the same person. We define this *linkability probability* as follows.

Definition 4.3 (Linkability Probability) *Let A be any family of non-deterministic circuits. We perform the following experiment:*

- *Choose at random a sufficiently long legal sequence of transactions* $\sigma = \langle T_1, T_2, \ldots, T_q \rangle$.
- *Pick a random* $i \leftarrow \{1, 2, \ldots, q\}$.
- *Let* $j \leftarrow A(\sigma, i)$.

The success probability of A is $p_A := \Pr[\, i \neq j \wedge$ *transactions i and j were done by the same user* $]$, *where the probability is taken over all random choices. The linkability probability of the scheme is* $\max_A p_A$. □

This intuitively corresponds to the following scenario: The supplier detects an interesting transaction (T_i), and wants to know more about the behavior of that particular user. What is the probability that he will find another transaction (T_j) of the same user?

If the access behaviors of all users are the same, then the linkability probability is $1/n$, achieved by randomly guessing the second transaction T_j. On the other hand, if the remaining number of subscriptions were transmitted every time, the linkability probability would increase to $(2^m - 1)/n$. We will now prove that for the bit counting scheme, the linkability probability is much smaller than that worst case.

Lemma 4.4 *The linkability probability for the bit counting scheme is at most m/n.*

Proof: Let transaction T_i consist of t tokens. Based on this information, the supplier only learns that the corresponding user belongs to a certain group of size $n/2^t$ (see Claim 4.2). Since, based on all available information, the supplier cannot distinguish between these users, the best he can hope for is to find a second transaction T_j for some user from this group. Assuming similar access patterns, any user of the group is responsible for that transaction with the same probability $1/(n/2^t)$. This is therefore the linkability probability if T_i consists of t tokens.

But transaction T_i is chosen at random. Since a fraction of about $1/2^t$ of all transactions involve t tokens, the total linkability probability for the bit counting scheme is at most

$$\sum_{t=1}^{m} \frac{1}{2^t} \frac{1}{n/2^t} = \frac{m}{n} = \frac{\log T}{n}. \quad \blacksquare$$

Other properties. The scheme also achieves coalition-resistance, since every token signed with key t_i is 'worth' exactly 2^{i-1} items: even if users share tokens, they do not increase the total number of items they will receive. Other properties are unforgeability and unshareability (the lowest order bit token essentially plays the role of the token in [SSG97]; there is no easy way to share this token, since it has to be handed in at every transaction). Finally, the scheme also supports early termination, since the user can transmit all remaining tokens, which proves the number of items that he has not received; he therefore can obtain a refund for exactly this number of items.

4.4 Improving the Security and Application to Rental Schemes

A simple enhancement can even further improve the unlinkability properties of the bit counting scheme. Suppose a user maintains not one, but two counters. When initially subscribing to the service their values are chosen at random, such that they add up to the total subscription length. Every time the service is accessed, the user randomly chooses one of the two counters and uses this counter for the transaction. Now if a transaction consists of t tokens, the supplier does *not* learn that the remaining subscription length is equal to 2^{t-1} modulo 2^t, since he has no information about the other counter. It is clear that this change improves the security of the scheme. We omit a full analysis due to space restrictions.

As mentioned earlier, the bit counting scheme can be extended to handle anonymous rental schemes. In these schemes, items can be returned as well as received by the client, using the protocols stated in section 4.1. To prevent a user from returning more items than he has actually rented (e.g. by returning another user's items), each user should keep two counters, one keeping track of the items that the user can still rent (initially ℓ), the other recording the number of items that the user currently has rented (initially 0). Renting and returning an item then corresponds to increasing one of the counters, and decreasing the other.

This rental protocol is most appropriate if there is no need for the user to return an item after a certain time, because due to the anonymity there is no way to tell who rented a particular over-due item. But for example with software licenses (which can be checked out or checked in by the users), this problem is not present, since there is no need to return a license as long as it is in use.

5 Client Created Token Scheme

In this section, we describe a scheme in which the clients have limited signing capabilities and can create their own tokens. The scheme still maintains the necessary security properties.

5.1 Description of Protocol

Informally, our scheme works as follows. The vendor sets up a "subscriber group," that the clients can join at any time. When a client joins, he is given the ability to sign messages on behalf of the group, but in such a way that the signatures are anonymous, unlinkable, and yet hard to forge by anyone else including the vendor and other members of the subscriber group.

To access the service, the client deterministically signs a message M from a set of ℓ messages published by the vendor. The supplier keeps track of all signed messages he has seen, and therefore the number of accesses by each user is limited to the number of messages ℓ. The supplier only needs a single verification key to check that any given client is indeed a subscriber. This key stays the same throughout the life of the protocol, and is independent of the group size.

We can impose additional restrictions on the subscription, such as limiting access to only once per day. For every time interval in which we want to allow at most ℓ' accesses, we provide a new set of ℓ' messages, and require that a customer signs one of these messages to access the service during this time interval. This time interval could be a single day (with, for example, $\ell' = 1$) or unlimited (to bound the total number of accesses to the service).

We can instantiate our scheme by making several crucial modifications to the group signature scheme of [CS97].[1] Group signatures, by definition, have a group manager who can revoke anonymity, which we do not want. Also, many group signature schemes are randomized. Thus the challenge lies in removing the revocation capabilities of the group manager and making the signatures deterministic so we can easily detect overspending. The security analysis is more complex than a simple reduction from the original scheme since we have to prove that even the group manager is unable to identify or link members.

5.2 High Level Description of the Modifications

We now discuss the modifications we make to [CS97]. First, we incorporate a blind RSA signature [Cha83] into the process by which the customer opens an account with the vendor. As a result, it is infeasible for the vendor to determine the customer's identity.

Second, we use an additional random oracle $\mathcal{H}_2$ to derandomize the signatures (see [BR93] for a discussion on the random oracle model). This oracle is applied to the message M to generate the random bits needed for signing that message. This oracle derandomizes the [CS97] scheme; thus if a customer signs the same message twice, then the resulting signatures will be identical. This prevents the user from using the same message for two different accesses. This change does not compromise the overall security analysis of [CS97], since it was already based on the random oracle model. Of course, whether a scheme in the random oracle model can be instantiated securely, with an actual polynomial time computable function instead of the random oracle, is uncertain [CGH98].

Our techniques are fairly general, and could possibly be modified to work with other group signature schemes as well. Before going into details, we discuss *signatures of knowledge*.

5.3 Signatures of Knowledge

The signature of knowledge was the basic building block of the Camenisch and Stadler [CS97] group signature scheme. It is a construct by which a signer can use knowledge of some secret information in order to digitally sign a message via a non-interactive zero-knowledge proof. It can also be used to prove knowledge of a particular secret.

Signatures of knowledge are based on three-move zero-knowledge protocols which are made non-interactive via the use of a random oracle $\mathcal{H}$, as in the well-known Fiat-Shamir heuristic [FS87]. All the signatures of knowledge proposed by [CS97] can be proved secure in the random oracle model [BR93] and their interactive versions are zero knowledge.

In the following, $G = \langle g \rangle$ is a group generated by g. We let $c[i]$ denote the i-th leftmost bit of a bit-string c, and we let $\mathcal{H}_k$ denote the k least significant bits of the output of $\mathcal{H}$. We first consider a signature of knowledge of the discrete logarithm of a given $y \in G$ to a given base g ($\langle g \rangle = G$). This signature of knowledge was originally derived from the Schnorr Signature scheme [Sch90].

Definition 5.1 *A $(k+1)$-tuple $(c, s_1, \ldots, s_k) \in \{0,1\}^k \times \mathbb{Z}_n^k$ satisfying $c = \mathcal{H}_k(m, y, g, g^{s_1} y^{c[1]}, \ldots, g^{s_k} y^{c[k]})$ is a signature of knowledge of the discrete logarithm of $y \in G$ to the base g on a message m, with respect to security parameter k, denoted $SKLOG_k[\alpha \mid y = g^{\alpha}](m)$.* $\square$

[1] Actually, we modify a more secure variant of [CS97] as suggested by [AT99].

In general we use Greek letters to represent values whose knowledge will be proven by the signer, and Roman letters to denote values that are known to both the signer and the user. If the signer knows the discrete logarithm of y to the base g, he can easily generate the signature. If he does not then it is infeasible for him to construct the $k+1$ tuple $(c, s_1, \ldots, s_k)$ satisfying the above equation. We can think of the above definition as an interactive protocol in which the $c[i]$'s represent challenges and the hash function $\mathcal{H}$ serves to remove the interaction. If the prover knows x such that $x = \log_g y$, he computes $r_1, r_2, \ldots, r_k \in_R \mathbb{Z}_n$, plugs $m, y, g, g^{r_1}, g^{r_2}, \ldots, g^{r_k}$ into hash function $\mathcal{H}$ to obtain the random challenge c, and obtains $s_1, s_2, \ldots, s_k$ by setting $s_i = r_i - c[i]x \pmod{n}$.

We now present two other signatures of knowledge. These signatures of knowledge are based on the double discrete logarithm problem, and the root of the discrete logarithm problem respectively. Details can be found in [CS97].

Definition 5.2 *A signature of knowledge of a double discrete logarithm of y to the bases g and a, on message m, with security parameter k, denoted $SKLOGLOG_k[\alpha \mid y = g^{(a^\alpha)}](m)$, is a $(k+1)$-tuple $(c, s_1, \ldots, s_k) \in \{0,1\}^k \times \mathbb{Z}^k$ satisfying the equation:*

$$c = \mathcal{H}_k(m, y, g, a, P_1, \ldots, P_k), \text{ where } P_i = \begin{cases} g^{(a^{s_i})} & \text{if } c[i] = 0 \\ y^{(a^{s_i})} & \text{otherwise} \end{cases} \quad \square$$

Definition 5.3 *A signature of knowledge of an e-th root of the discrete logarithm of y to the base g, on message m, denoted $SKROOTLOG_l[\alpha \mid y = g^{(\alpha^e)}](m)$, is a $(k+1)$-tuple $(c, s_1, \ldots, s_k) \in \{0,1\}^k \times \mathbb{Z}_n^{*k}$ satisfying the following equation:*

$$c = \mathcal{H}_k(m, y, g, e, P_1, \ldots, P_k), \text{ where } P_i = \begin{cases} g^{(s_i^e)} & \text{if } c[i] = 0 \\ y^{(s_i^e)} & \text{otherwise} \end{cases} \quad \square$$

Having discussed the necessary building blocks, we now give our construction.

5.4 The Protocol

System Setup. The Vendor generates an RSA modulus $n = pq$ and public key e, a cyclic group G generated by an element g of order n, and an element $a \in \mathbb{Z}_n^*$ of large multiplicative order modulo p and q. In addition, he selects a bound λ on the size of the customers' authentication keys. Finally, he selects a random value $\delta \in \mathbb{Z}_n^*$. His public key is (n, e, G, g, a, λ). Next, the vendor publishes a list of messages $\{M_1, \ldots, M_\ell\}$ where ℓ is the length of the subscription. We implicitly assume the existence of a public-key infrastructure, and a certificate authority so that the message list can be signed, and the group public key may be validated.

Open Account / Subscribe.

1. With the help of the vendor **V**, the customer **C** chooses a *random* secret key x in the following manner.
   (a) The customer chooses a random value $\hat{x}$ between 0 and $2^\lambda - 1$, and computes $\hat{z} = g^{a^{\hat{x}}}$. He sends this value to the vendor.
   (b) The vendor computes a random value γ between 0 and $2^\lambda - 1$ and sends this value to the customer.
   (c) If $\hat{x} + \gamma$ is between 0 and $2^\lambda - 1$, then the customer makes $x = \hat{x} + \gamma$ his secret key. Otherwise **C** and **V** repeat the process.

The customer **C** computes $y = a^x \pmod{n}$ and $z = g^{(a^x)}$. He sends z to **V**, and proves in zero knowledge that he knows x. In addition, he must prove that $0 \leq x \leq 2^\lambda - 1$. This task can be accomplished via techniques from [CM98] or [CFT98] (or one can execute a more expensive cut and choose protocol).

2. **V** checks that $z = \hat{z}^{a^\gamma}$ and that **C**'s zero-knowledge proofs are correct.
3. Then **C** obtains a *blind* RSA signature on $y + \delta$ as follows. First, **C** chooses a value r at random from $\mathbb{Z}_n^*$, and computes $m = (y+\delta) \cdot r^e \bmod n$, and he sends this value to **V**. Next, **V** computes $\hat{v} = m^{1/e}$, and send this back to **C**, who computes $v = \hat{v}/r$. Note that $v = (y+\delta)^{1/e} \pmod{n}$.

Receive Item.

1. The customer chooses an index i (that he has not used before), and computes $r = \mathcal{H}_2(M_i)$. Next, **C** computes $\tilde{g} = g^r$ and $\tilde{z} = \tilde{g}^y$. He then computes:
   - $V_1 = SKLOGLOG[\alpha | \tilde{z} = \tilde{g}^{a^\alpha}](M_i)$
   - $V_2 = SKROOTLOG[\beta | \tilde{z}\tilde{g}^\delta = \tilde{g}^{\beta^e}](M_i)$.

   Finally, he sends $\tilde{g} = g^r$, $\tilde{z}$, V_1, and V_2 to the supplier **S**.
2. If the signatures of knowledge are valid, and the signature has not been used before, then **S** sends the requested item to **C**.

5.5 Performance Analysis

This scheme performs very well with respect to communication complexity and storage requirements. In particular, the communication complexity (both for the account opening and for transactions) is linear in the security parameter; it is thus *independent* of the the number of subscribers and the subscription length ℓ. Similar to the bit counting scheme, the supplier has to store all previously used signatures, which can be up to $O(n\ell)$ signatures. This can be alleviated by keeping subscriptions short, and selling a customer several small subscriptions instead of one big subscription.

The user storage complexity is $O(k)$ for the key, and $O(\ell)$ to remember which of the ℓ messages were already signed. This could be reduced to $O(k + \log \ell)$ if the customer uses a pseudo-random permutation [GGM84, LR88] to determine the order in which he signs messages.

5.6 Security Analysis

We now analyze the security of the client created token scheme. Even though the protocol is constructed by making only a few changes to the original signature scheme of [CS97], the security analysis turns out to be significantly more complex. The original [CS97] scheme was based on the difficulty of the Schnorr [Sch90] and RSA [RSA78] signature schemes, the intractability of deciding whether two discrete logarithms are equal (which was used in the construction of undeniable signatures [CvA89]), and the additional assumption that computing membership certificates is hard (which was also used in [KP98, LR98]).

In addition to these previous assumptions, our new scheme also relies on an additional assumption which we call the *Promised Decisional Diffie-Hellman* assumption. It turns out that this assumption is computationally equivalent to the well-known *Decisional Diffie-Hellman* assumption. We discuss these assumptions in more detail.

5.6.1 Regular and Promised Decisional Diffie-Hellman Assumptions

This regular Decisional Diffie-Hellman (DDH) assumption has been used repeatedly in the literature (see [Bon98] for an overview).

Assumption 5.4 (Decisional Diffie-Hellman) *Let g be a generator for a cyclic group G of sufficiently large order n. Then consider a poly-time bounded adversary who takes as input a tuple* (g, g^x, g^y, g^z) *where either z is chosen randomly or* $z = xy$*. This adversary will be unable to distinguish between the two cases with probability non-negligibly better than* $1/2$*.*

In this paper we introduce the *Promised Decisional Diffie-Hellman* (PDDH) assumption and show that it is computationally equivalent to the original Decisional Diffie-Hellman assumption.

Assumption 5.5 (Promised Decisional Diffie-Hellman) *Let g be a generator for a cyclic group G of sufficiently large order n. Let* $P \geq 2$ *be a constant. Consider a poly-time bounded adversary who is given an input of the form* $(g^{y_1}, g^{y_2}, \ldots, g^{y_P}, g^r, g^{ry_j})$ *where j is a secret random number between 1 and P. This adversary will be unable to determine j with probability non-negligibly better than by randomly guessing.*

Consider the case of $P = 2$. The adversary is given an input of the form $(g^{y_1}, g^{y_2}, g^r, g^{ry_j})$. In DDH the adversary would be given an input of the form (g^{y_1}, g^r, g^s), and asked if s is a random value or if $s = ry_1$. In this sense PDDH appears easier since the adversary is *promised* that if $s \neq ry_1$ then it must be the case that $s = ry_2$. If he can determine whether or not y_j is equal to y_1 with probability non-negligibly better than $1/2$, then he will have succeeded in breaking this assumption. So, if can find an adversary capable of breaking DDH, then it can easily be converted to one that breaks PDDH. We show that the converse holds as well. That is if there is an adversary who can break PDDH, then we can construct an adversary who breaks DDH.

Theorem 5.6 *There exists a poly-time bounded adversary* $\mathcal{A}$ *that can break* PDDH *if and only if there exists a poly-time bounded adversary* $\mathcal{A}'$ *that breaks* DDH.

Proof: See Appendix A. ■

5.6.2 Security Claims and Proofs for the Client Created Token Scheme

We now state and prove various security claims for our scheme.

Claim 5.7 (Unforgeability) *It is computationally infeasible for non-subscribers to access the service.*

Sketch of Proof: If an illegitimate subscriber accessed the service, it would constitute an existential forgery on the [CS97] signature scheme, which is believed to be intractable. ■

We remark that although illegitimate use of the service would result in solving the seemingly intractable problem of forging signatures in the [CS97] scheme, the converse is not true. In particular, since our subscription scheme involves signing particular messages, and since the "random" bits used in our scheme are generated by applying a hash function to the message, an existential forgery on [CS97] may not break our scheme.

Claim 5.8 (Anonymity) *It is computationally infeasible for the vendor to determine the identity of subscribers when they access the service.*

Sketch of Proof: We sketch the case when only two clients subscribe to the service. The general case can be handled similarly. Suppose the vendor receives the values g^{y_1} and g^{y_2} from these two group member when they respectively join. He also receives various non-interactive zero-knowledge proofs from these members. However, since these proofs are zero knowledge, the vendor can generate them himself, so we can ignore them without loss of generality. Now, when a client accesses the service, he gives a signature consisting of g^r, g^{ry_i} and two non-interactive zero-knowledge proofs. Again, we can ignore these. Thus, the vendor only "sees" the four-tuple $(g^{y_1}, g^{y_2}, g^r, g^{ry_i})$, and he must determine whether $i = 1$ or 2. Moreover, under the random oracle assumption, the value r is chosen uniformly at random from $\mathbb{Z}_n$. This is an instance of PDDH which we have shown to be intractable assuming DDH is. ■

Claim 5.9 (Unlinkability) *It is computationally infeasible for the vendor to determine whether two distinct accesses were made by the same subscriber with probability non-negligibly better than 1/2.*

Sketch of Proof: For simplicity, we sketch the case when only two clients subscribe to the service. The vendor will receive the values g^{y_1} and g^{y_2} from these two clients. Suppose the vendor sees two signatures generated from accesses to the services. These signatures will contain the information $(g^{r_1}, g^{y_i r_1})$ and $(g^{r_2}, g^{y_j r_2})$, where $i, j \in \{1,2\}$. Under the random oracle assumption, the r_i are uniformly distributed over $\mathbb{Z}_n$. Again, we can ignore the various non-interactive zero-knowledge proofs that the vendor sees. As a result, the vendor has only seen a tuple of the form $(g^{y_1}, g^{y_2}, g^{r_1}, g^{r_2}, g^{r_1 y_i}, g^{r_2 y_j})$ and he must determine if the two accesses were made by the same subscriber (i.e. $i = j$). If he can determine the answer with probability $1/2 + \varepsilon$ then we claim he can break PDDH with the same probability.

Suppose that we have an instance of PDDH: $(g^{x_1}, g^{x_2}, g^r, g^{x_i r})$. We can reduce it to the linkability problem as follows. Pick a value $r' \in_R \mathbb{Z}_n$, and form the tuple $(g^{x_1}, g^{x_2}, g^r, g^{r'}, g^{x_i r}, g^{x_1 r'})$ and give these to the vendor. Now, if the vendor returns that these accesses are made the by same person, then $i = 1$ with probability $1/2 + \varepsilon$. Otherwise, $i = 2$ with the same probability. This gives us a method of breaking PDDH, which we showed to be intractable assuming DDH is. ■

Corollary 5.10 (Linkability Probability) *The linkability probability (definition 4.3) is at most negligibly greater than $1/n$ for poly-time bounded adversaries.*

Other Properties. We remark that if it were possible for a number of legitimate subscribers to collude and create signatures based on some combination of their secret signing keys in our scheme then the modification by [AT99] to the scheme of [CS97] would also be vulnerable to these same collusion attacks. Also, the scheme allows for early termination: the client simply signs any remaining messages and sends them to the vendor. This provides a proof of the number of unused items for which the client can get a partial refund.

6 Discussion

In this paper we defined two new schemes for anonymous subscription and rental services. These schemes are the first to achieve the property that the number of accesses is limited, while maintaining anonymity and unlinkability. The schemes are also very efficient with respect to storage and communication requirements.

The "bit counting" scheme limits the total number of accesses by having the remaining subscription length encoded in the user's access certificate. In the "client created token" scheme the access is limited by giving the client a (potentially time-dependent) fixed set of messages to sign.

The combination of the two schemes allows for even more complex access restrictions. Consider the following example. The Metropolitan Opera might want to sell subscriptions for their performances. A customer can pay for, say, 15 admissions in advance. This can be handled using the bit counting scheme. But there might be additional restrictions on which tickets can be obtained in the course of a subscription. For example, for regular performances, a customer might obtain up to 4 tickets on his subscription, but for premieres (which are more popular), he can get at most two. For these kinds of restrictions, group signatures are ideally suited. The customer receives a group key, and for premieres, he has to sign one of two available messages in the ticketing protocol, while for other shows there are four available messages to choose from.

By making group keys or bit counting keys time-dependent, i.e. they will only be accepted during certain specified intervals, accesses can be limited to certain times. By the combination of these methods virtually any imaginable set of restrictions on how a subscription may be used can be incorporated efficiently into the protocols.

An obvious direction for future research is to design even more efficient schemes than the ones given in this paper. In particular, it would be interesting to see whether the techniques that were used to convert the [CS97] group signature scheme into a subscription protocol can also be applied to other group signature schemes, such as the one in [ACJT00]. These other group signature schemes might be more efficient or rely on fewer security assumptions, translating into better properties for the subscription protocol.

Another open problem is to design a subscription protocol that allows for early termination of the subscription by the vendor, but still has reasonable storage and communication requirements. This problem is difficult since the scheme's anonymity and unlinkability appears to conflict with giving a supplier/vendor the power to revoke a user's subscription.

References

[ACJT00] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Proc. CRYPTO 2000*, pages 255–270. Springer-Verlag, 2000. Lecture Notes in Computer Science No. 1880.

[AT99] G. Ateniese and G. Tsudik. Some open issues and new directions in group signature. In M. Franklin, editor, *Proc. International Conference on Financial Cryptography 99*, pages 196–211. Springer-Verlag, 1999. Lecture Notes in Computer Science No. 1648.

[Bon98] Dan Boneh. The decision Diffie-Hellman problem. In *Algorithmic Number Theory (ANTS-III)*, pages 48–63. Springer-Verlag, 1998. Lecture Notes in Computer Science No. 1423.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[Bra00] Stefan Brands. *Rethinking Public Key Intrastructures and Digital Certificates: Building in Privacy*. The MIT Press, 2000.

[CFT98] Agnes Chan, Yair Frankel, and Yiannis Tsiounis. Easy come – easy go divisible cash. In *Proc. EUROCRYPT 98*. Springer-Verlag, 1998. Lecture Notes in Computer Science No. 1403.

[CGH98] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proc. 30th ACM Symp. on Theory of Computing*, 1998.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Proc. 36th IEEE Symp. on Foundations of Comp. Science*, 1995.

[Cha83] David Chaum. Blind signatures for untraceable payments. In R. L. Rivest, A. Sherman, and D. Chaum, editors, *Proc. CRYPTO 82*, pages 199–203. Plenum Press, 1983.

[CM98] Jan Camenisch and Markus Michels. A group signature scheme with improved efficiency. In *Advances in Cryptology–ASIACRYPT '98*, pages 160–174. Springer-Verlag, 1998. Lecture Notes in Computer Science No. 1514.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signatures for large groups. In *Proc. CRYPTO 97*, pages 410–424. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1294.

[CvA89] D. Chaum and H. van Antwerpen. Undeniable signatures. In *Proc. CRYPTO 89*, pages 212–216. Springer-Verlag, 1989. Lecture Notes in Computer Science No. 435.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In *Proc. EUROCRYPT 91*, pages 257–265. Springer-Verlag, 1991. Lecture Notes in Computer Science No. 547.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A.M. Odlyzko, editor, *Proc. CRYPTO 86*, pages 186–194. Springer-Verlag, 1987. Lecture Notes in Computer Science No. 263.

[GGM84] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1984.

[KP98] J. Killian and E. Petrank. Identity escrow. In H. Krawczyk, editor, *Proc. CRYPTO 98*, pages 169–185. Springer-Verlag, 1998. Lecture Notes in Computer Science No. 1462.

[LR88] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Computing*, 17(2):373–386, April 1988.

[LR98] Anna Lysyanskaya and Zulfikar Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In R. Hirschfeld, editor, *Proc. International Conference on Financial Cryptography 98*, pages 184–197. Springer-Verlag, 1998. Lecture Notes in Computer Science No. 1465.

[NHS99] Toru Nakanishi, Nobuaki Haruna, and Yuji Sugiyama. Unlinkable electronic coupon protcocol with anonymity control. In *Proceedings of 2nd International Information Security Workshop*, pages 37–46. Springer-Verlag, 1999. Lecture Notes in Computer Science No. 1729.

[NR97] Moni Naor and Omer Reingold. Number theoretic constructions of efficient pseudo-random functions. In *Proc. 38th IEEE Symp. on Foundations of Comp. Science*, pages 458–467, 1997.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sch90] C. P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Proc. CRYPTO 89*, pages 239–252. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.

[SPH99] Stuart Schechter, Todd Parnell, and Alexander Hartemink. Anonymous authentication of membership in dynamic groups. In M. Franklin, editor, *Proc. International Conference on Financial Cryptography 99*, pages 184–195. Springer-Verlag, 1999. Lecture Notes in Computer Science No. 1648.

[SSG97] Paul Syverson, Stuart Stubblebine, and David M. Goldshlag. Unlinkable serial transactions. In R. Hirschfeld, editor, *Proc. International Conference on Financial Cryptography 97*, pages 39–55. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1318.

[Sta96] Markus Stadler. Publicly verifiable secret sharing. In *Proc. EUROCRYPT 96*, pages 190–199. Springer-Verlag, 1996. Lecture Notes in Computer Science No. 1070.

A Appendix: Equivalence of DDH and PDDH

Proof of Theorem 5.6: We sketch the case of $P = 2$ – the general case can be treated similarly. Suppose we are given a tuple of the form $(a,b,c) = (g^x, g^y, g^z)$ where either $z = xy$ or z is chosen at random. Our goal is to construct an adversary $\mathcal{A}'$ that takes this as input and with advantage non-negligibly better than $1/2$ determines which is the case. In addition, this adversary will have oracle access to $\mathcal{A}$. The adversary $\mathcal{A}'$ first constructs a series of q random instances of DDH problem where the i-th tuple ($1 \leq i \leq q$) is:

$$(a_i, b_i, c_i) = (a^{v_i} g^{u_{i1}}, b g^{u_{i2}}, c^{v_i} b^{u_{i1}} a^{v_i u_{i2}} g^{u_{i1} u_{i2}})$$

where $v_i, u_{i1}, u_{i2} \in_R \mathbb{Z}_n$. This construction appeared in [Bon98], and is a slight generalization of a construction that appeared in [Sta96] and [NR97]. Now, if $z = xy$, then each of the triples (a_i, b_i, c_i) are also valid triples. In addition, it turns out that each (a_i, b_i, c_i) is independently chosen from a distribution that is *statistically* indistinguishable from the uniform distribution on DDH triples over the group G [Bon98]. If z was just a random value then each triple (a_i, b_i, c_i) is independently chosen from a distribution that is *statistically* indistinguishable from the uniform distribution on triples over the group G.

Now, the adversary $\mathcal{A}'$ converts each of these tuples into what may be an instance of the PDDH. It first chooses a series of random bits $\beta_1, \ldots, \beta_q$ ($\beta_i \in \{0,1\}$) and a series of random values in $\mathbb{Z}_n$: $r_1, \ldots r_q$. Then, $\mathcal{A}'$ constructs a series of four-tuples (a'_i, b'_i, c'_i, d'_i) as follows. If $\beta_i = 0$ then we set $a'_i = a_i$ and $b'_i = g^{r_i}$, otherwise $a'_i = g^{r_i}$ and $b'_i = a_i$. In both scenarios $c'_i = b_i$ and $d'_i = c_i$. There are two cases to consider:

1. If $z = xy$, then for all i, (a'_i, b'_i, c'_i, d'_i) is independently chosen from a distribution that is statistically to the uniform distribution on PDDH tuples.

2. If z were chosen at random, then for all i, (a'_i, b'_i, c'_i, d'_i) is independently chosen from a distribution that is statistically indistinguishable from the uniform distribution on four-tuples over the group G.

Suppose we have an adversary $\mathcal{A}$ that can break PDDH with probability $1/2 + \varepsilon$, where ε is a non-negligible quantity. If we are in the first case, then for every i this adversary will correctly determine with probability $1/2 + \varepsilon$ whether a'_i or b'_i was used in the formation of d'_i. Note that this corresponds to correctly guessing the series of random values $\beta_1, \ldots, \beta_q$. This adversary will be correct an expected $q/2 + q\varepsilon$ times. Now, if we are in the second case, then the adversary $\mathcal{A}$ will be given a random four-tuple over G, and it will have no information about the value of β_i. Thus, for each i, it will correctly predict the β_i with probability $1/2$. In this case, it will be correct an expected $q/2$ times. This gives us a sizable distinguishing gap of $q\varepsilon$. If $\mathcal{A}$ was correct on at least $1/2 + \varepsilon q/2$ fraction of the inputs, then $\mathcal{A}'$ should output that the triple is a valid DDH triple (that is $z = xy$). Otherwise, it should output that the triple was random. For sufficiently large q, one can use standard bounds to show that $\mathcal{A}'$ is correct with probability non-negligibly better than $1/2$. ■

Protocol	*Limited Access*	*Unshareable*	*Commun. Complexity (Set-up)*	*Commun. Complexity (Transaction)*	*Storage (Server)*	*Storage (User)*
Digital Cash	Yes	No	$O(k\ell)$	$O(k)$	$O(nk\ell)$	$O(k\ell)$
Unlinkable Serial Transactions [SSG97]	No	Yes	$O(k)$	$O(k)$	$O(nk\ell)$	$O(k)$
Commonly Verifiable Secrets [SPH99]	No	No	$O(nk)$	$O(nk)$	$O(nk)$	$O(nk)$
Bit Counting	Yes	Yes	$O(k\log\ell)$	$O(k)$	$O(nk\ell)$	$O(k\log\ell)$
Client Created Tokens	Yes	Yes	$O(k)$	$O(k)$	$O(nk\ell)$	$O(k+\log\ell)$

Table 1: Comparison of various schemes; k = security parameter (key size), n = number of users, ℓ = number of times service is used.