Unlinkable Serial Transactions: Protocols and Applications

STUART G. STUBBLEBINE
CertCo
PAUL F. SYVERSON
Naval Research Laboratory
and
DAVID M. GOLDSCHLAG
USinternetworking, Inc.

We present a protocol for unlinkable serial transactions suitable for a variety of network-based subscription services. It is the first protocol to use cryptographic blinding to enable subscription services. The protocol prevents the service from tracking the behavior of its customers, while protecting the service vendor from abuse due to simultaneous or "cloned" use by a single subscriber. Our basic protocol structure and recovery protocol are robust against failure in protocol termination. We evaluate the security of the basic protocol and extend the basic protocol to include auditing, which further deters subscription sharing. We describe other applications of unlinkable serial transactions for pay-per-use transactions within a subscription, third-party subscription management, multivendor coupons, proof of group membership, and voting.

Categories and Subject Descriptors: J.1 [**Computer Applications**]: Administrative Data Processing; D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; *Cryptographic controls*; *Authentication*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*; *Unauthorized access* (e.g., hacking, phreaking); H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*User profiles and alert services*; H.2.4 [**Database Management**]: Systems—*Transaction processing*; H.3.7 [**Information Storage and Retrieval**]: Digital Libraries—*User issues*

General Terms: Design, Security, Verification

Additional Key Words and Phrases: Anonymity, blinding, cryptographic protocols, unlinkable serial transactions

A preliminary version of this paper appeared previously; see Syverson et al. [1997]. Work by the first author was primarily supported by AT&T Research, Florham Park, NJ. Work by the second and third authors was supported by ONR.
Authors' addresses: S. G. Stubblebine, CertCo, New York, NY 10004; email: stubblebine@cs.columbia.edu; P. F. Syverson, Center for High Assurance Computer Systems, Naval Research Laboratory, Washington, DC 20375; email: syverson@itd.nrl.navy.mil; D. M. Goldschlag, USinternetworking, Inc., One USi Plaza, Annapolis, MD 21401; email: david@goldschlag.com.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1999		**00-00-1999 to 00-00-1999**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Unlinkable Serial Transactions: Protocols and Applications	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Research Laboratory,Center for High Assurance Computer Systems,4555 Overlook Avenue, SW,Washington,DC,20375**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**36**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

1. INTRODUCTION

This paper is motivated by the conflict of interest inherent in maintaining privacy in an electronic exchange. Commercial service providers would like to make sure that they are paid for their services and protected from abuse due to simultaneous or "cloned" use by a single subscriber. To this end, they have an interest in keeping a close eye on customer behavior. On the other hand, customers have an interest in maintaining the privacy of their personal information, in particular, profiles of their commercial activity. One well-known approach is to allow customers to register with vendors under pseudonyms, typically one for each vendor [Chaum 1981]. Customers maintain anonymity by conducting transactions using anonymous electronic cash (e-cash). But vendors are able to protect their interests by maintaining a profile on each anonymous customer.

In this paper we present, effectively, the opposite solution to this problem. Customers may be known to vendors, but the customers' behavior is untraceable. This appears infeasible. If transactions cannot be linked to customers, what is to keep customers from abusing the service? For example, if someone fails to return a rented video, the video rental company would like, at minimum, to be sure that this person will not rent any more videos. But the company cannot do this if it cannot determine who the renter is.[1] We present a protocol that makes transactions unlinkable, but also protects vendors from abuse.

For the near future at least, a large part of the market on the Internet and in other electronic venues will rely on credit card-based models (such as simply sending credit card numbers over SSL or SET [SET 1999]). Applications of our protocol that require payment do not depend on the payment mechanisms. So our protocol can be easily applied now, but is equally amenable to e-cash. Even in an environment in which pseudonyms and anonymous e-cash are generally available, vendor profiles of customers (or their pseudonyms) might be undesirable because protecting the customer's anonymity can fail at a single point. If the vendor is ever able to link a pseudonym to a customer, the entire profile immediately becomes linked to that customer. In our solution, if a customer is ever linked to a transaction, only his or her link to the one transaction is disclosed. This is somewhat analogous to the property of perfect forward secrecy in key establishment protocols [Gunther 1990]. However, no underlying system appears immune to the service correlating application data. (In other words, the transaction payload might provide identifying information irrespective of the transaction protocol or channel anonymity; see the discussion following assumption 1 in Section 2.1.1.) Thus, security at the higher-level application should also be evaluated. At the same time, our protocol makes sharing account access more difficult than sharing ordinary account passwords. So

[1]In a pseudonym-based scheme, such a customer could try to open an account under a new pseudonym, but there are mechanisms that make this difficult [Chaum 1985]. Thus the interests of the vendor can be protected.

our approach affords better protection to both customers and vendors than existing schemes do.

In a nutshell, our approach works as follows: After registering with a vendor, a customer has a single-use token that authenticates the customer as an authorized user of the service. Using widely implemented cryptographic techniques, the vendor can activate the token without being able to recognize it when the customer uses it for authentication. On each use of the service, the customer receives a new token. The contribution of this work is not so much in setting out novel cryptographic mechanisms as it is the description of a new secure system problem area, a rigorous statement of the requirements to solve the problem, and the presentation of a system that meets the requirements, together with proof that it does so. Our basic protocol structure and recovery protocol have a novel design to protect against protocol termination attacks, which can, for instance, defraud the merchant by receiving goods without paying for them.

On what applications could our approach be used? Consider a subscription service for an online newspaper or encyclopedia. Customers have an interest in keeping their searches private. At the same time, vendors want to make it difficult for customers to transfer access to the vendor's service. This serves as our primary example.

We also consider other applications. One example is pay-per-use service within a subscription (e.g., Lexis-Nexis^{TM} or pay-per-view movies available to cable TV subscribers). Unlinkable serial transactions can also be used to provide multivendor packages (e.g., digital coupon books for services from various vendors) as well as ongoing discounts. They can also be used for anonymous proof of membership for applications with nothing directly to do with electronic commerce. Applications include proof of age and proof of residency. Unlinkable serial transactions can also be used to construct a simple but effective voter registration protocol.

The paper is organized as follows. In Section 2 we describe requirements and assumptions. We then present the basic protocol, including registration, certificate redemption, termination of a subscription, and recovery from broken connections. In Section 3 we prove that our basic protocol meets our requirements. In Section 4 we extend the protocol to include auditing, in order to make detection of unauthorized sharing of subscriptions possible. In Section 5 we describe various applications of unlinkable serial transactions and associated protocol variants. In Section 6 we describe related work. Most of the basic mechanisms on which we rely come from work on e-cash; but we were are able to simplify some of the mechanisms in this application. We describe them and their relation to our work. In Section 7 we present concluding remarks.

2. TRANSACTION UNLINKABILITY

In this section we describe protocols that prevent linking a customer's transactions to each other. Consequently, transactions also cannot be linked to the customer. We assume that the customer has subscribed to a

service with which the customer will conduct these transactions and has provided adequate identifying and billing information (e.g., credit card numbers).

The basic protocol allows a customer to sign up for unlimited use of some subscription service for a period of time, but prevents the service from determining when the customer has used the service or what he or she has accessed (subject to assumption 1). At the same time, mechanisms are provided that make it inconvenient for customers to share their subscriptions with others. Later, in Section 4.3, we describe an enhanced version of the basic protocol, which also leaves customers vulnerable to detection if they share their subscriptions despite the inconvenience.

2.1 Threats, Requirements, and Operating Assumptions

After stating our assumptions, we set out the requirements that such protocols should meet. We describe specific illegitimate use scenarios later, in Section 2.9. Here we express high-level requirements to protect against these threats and also express assumptions about the operating environment. Solutions for countering the threats are presented in subsequent sections of this paper.

We expect the protocols to be used predominately by vendors wishing to sell public network access to information (typically resident in databases). These vendors will need to offer customer anonymity on a transaction basis.

2.1.1 *Operating Environment Assumptions*. Customers of the service will access the servers through standard networking protocols. However, when customers desire anonymity they access the server using communications that protect their anonymity. We assume the following:

A1. Anonymity-protected network communications are unlinkable to prior communications, provided application content does not enable linkage.

Services that provide channel anonymity, i.e., disassociating the customer from the communication channel that carries data between the customer and server, are described in Section 6.3. No practical mechanisms for Internet communication provide information-theoretic security against an adversary able to observe the entire network. (The nature and degree of anonymity for various services are discussed in Section 6.3.) Also, a customer's channel anonymity is obviated if application data sent over the channel identifies the customer. While one may provide application-layer filtering of identifying information, if transactions involve running ever more functionality over connections, there can never be a comprehensive list of what must be stripped out to prevent identifying customers to the far end. There is a tradeoff to be made here between anonymity and functionality. What the above assumption says is that if there is nothing in the application data linking one communication session to a previous one, then, if a (channel) anonymity service is used, the channel anonymity provided is

assumed to be unbreakable. This is analogous to the usual protocol assumptions that encrypted messages can never be revealed by any direct cryptanalysis.

If all clients of a protected service collude by "exposing" their transactions, then it is obvious that the remaining transaction is "your" own. Thus we require the following assumption:

A2. Entities may collude. However, we assume that collusion among customers is insignificant, in the sense that there will always be a sufficient number of noncolluding customers and associated transactions to mask legitimate customer activity.

Keys and other numbers that must be unpredictable and unique are chosen from a large-enough space. Keyed cryptographic operations protect integrity. What sort of operations this includes is set out in Section 2.2.

A3. We assume that cryptographic keys, nonces, blinding factors, etc. are adequately chosen randomly from an adequately large space to prevent random collisions or the disclosure of secrets by cryptanalytic attacks.

A4. We assume that keyed cryptographic operations prevent any undetectable modification of fields to which those operations are applied. Furthermore, we assume the entity's inability to forge signatures without knowledge of the key.

Fresh, random, secret nonces are used to protect duplicate use of certificates. If accidental nonce collisions occur, off-line appeal may be necessary. Another assumption is that a message sent from a customer to the vendor or from the vendor to a customer cannot be indefinitely prevented from arriving uncorrupted.

A5. We assume that every message is received as sent after a finite number of attempts to send it.

Whatever the potential in attempting compromise of customer privacy, the vendor is assumed not to want to cheat paying customers.

A6. Vendors will provide services for which they accept payment.

2.1.2 *Fraud*. Protection against fraud is particularly important due to seemingly inherent conflicts with transaction anonymity. Despite transaction anonymity, vendors should be assured that they are not providing uncontracted services. We address two types of fraud. The first is sharing a subscription. The second is fraudulently obtaining new subscriptions or using subscriptions that were to be invalidated.

High-volume fraud, wherein an attacker sets up as an effective subscription subcontractor or fraudulently obtains large numbers of subscriptions in an organized fashion, is of primary concern. So our first requirement is

R1. Eliminate high-volume fraud.

Even if high-volume fraud is eliminated, there is still the possibility that an individual customer will share subscriptions or obtain additional subscriptions. To some extent, sharing is inevitable and permissible. For example, we expect an individual to share subscriptions among family members; but we would like to minimize its abuse. Also, individuals will in fact lose the certificates for their subscriptions from time to time, and may even pay without their subscription properly initializing. Subscriptions should be made robust against these possibilities, e.g., by issuing sufficient initial certificates with a subscription and/or by correcting subscriptions for such individuals. However, we would like to minimize abuse here as well. Thus, we require:

R2. Detect and reduce activity related to low-volume fraud.

The protocol should protect against unauthorized payment such as the theft of payment and refunds by third parties.

R3. Payments (including refunds if applicable) cannot be stolen.

2.1.3 *Customer Privacy*. There will be incentives for vendors to profile customers to obtain marketing information. Furthermore, outsiders and other customers may wish to obtain information for industrial espionage. Specifically, it should be difficult for a vendor or others to link the customer to any particular requested transaction. It should also be difficult for the vendor to link any one transaction request with any other (building a profile that might ultimately be tied to a customer is difficult.)

R4. Prevent the building of customer profiles (including pseudonymous profiles) by vendors, other customers, and outsiders.

R5. In a transaction, protect the identity of the customer from vendors, other customers, and outsiders.

2.1.4 *Service Guarantee*. Vendors should not be tricked into servicing invalid customers at a valid customer's expense. (This requirement depends on assumption 6, above.)

R6. Customers cannot be denied services they have contracted for.

As in most systems, we do not attempt to protect against denial of service attacks, such as flooding online services with service requests.

2.2 The Basic Unlinkable Serial Protocol

The basic protocol has two phases, registration and certificate redemption, optionally followed by a termination phase. The goal of registration is to issue credentials to a new customer. The new customer C presents sufficient identifying and payment information to the vendor, V. The vendor returns a single blinded certificate, which authorizes the customer to later execute a single transaction with that service.

In the certificate-redemption phase, the customer spends a certificate and executes a transaction. At the end of the certificate-redemption phase,

Table I. Notation

$[X_K]$:	Message integrity of X using K.
$\{X\}_K$:	Message integrity and confidentiality of X using K.
$\overline{X}$:	Blinding of X.
$h(X)$:	Hash of X.

the vendor issues the customer another blinded certificate. The vendor cannot link the new certificate to the spent one, so the vendor cannot use it to link transactions to one another. We assume that the customer has an associated identifier C for each account, whether or not the customer's identity is actually known by the vendor. (The customer may in fact have different identifiers for different accounts.)

We now discuss our notation, summarized in Table I. We use square braces to indicate message authentication. Thus, $[X]_K$ may refer to data X signed with key K or a keyed hash of X using K.

Curly braces represent both message confidentiality and integrity. $\{X\}_K$ refers to X encrypted with key K and, for our purposes, it refers to mechanisms that also provide integrity within the scope indicated. When there is no danger of confusion, we write $\{X\}_A$ to indicate that X is encrypted for A, typically using a public key. Similarly, we sometimes write $[X]_A$ to indicate that X is integrity-protected by A, typically using a signature key. It should be clear from the context whether or not A is the only identifier for a principal. For example, V and S below can both be associated with a single vendor in two different roles.

Blinding is a cryptographic technique used when we want entities to sign messages without seeing their contents [Chaum 1983]. The first step is for the originator to compute a blinded message using a blinding factor. We use over-lining to indicate the application of a blinding factor to a message, e.g., $\overline{X}$ refers to the result of blinding X. The next step involves signing the blinded document by the signing party, e.g., $[\overline{X}]_K$. Finally, the originator can remove the blinding factor, leaving the original document signed by the signer, e.g., $[X]_K$. Blinding involves the use of a secret such that anyone who does not know the secret blinding factor cannot associate $\overline{X}$ with either X or any $[X]_A$ [Menezes et al. 1997]. Note that our mechanism integrity assumption (4) intends for the blind signature to be secure from "one-more" forgery attacks [Pointcheval and Stern 1996]. Such forgery enables the originator to forgo more signed objects then it submitted for signing. Encryption, signing, and blinding are the "keyed cryptographic operations" referred to in Section 2.1.

Finally, $h(X)$ refers to a hash of X. This means that it is easy to compute $h(X)$ given X, but hard to compute X given $h(X)$. It is also hard to find any pair X and Y ($X \neq Y$) s.t. $h(X) = h(Y)$.

2.3 Registration

Message 1. $C \rightarrow V : \{Payment, D_{CV}\}_V, [Request\ for\ certificate\ of\ type\ S, C, \overline{H(N_1)}]_{K_{CV}}$.

Message 2. $V \rightarrow C : [\overline{h(N_1)}]_S$.

The encryption indicated by V uses a public key that is only used to encrypt for the vendor and only for the UST (Unlinkable Serial Transaction) protocols we describe. The signature indicated by S in message 2 uses the vendor's signature key for service S and is only used to sign blinded hashes. We call this signature key the service key. A service key is only used in transactions related to the service. A signed hash is a certificate. The service key is also subject to periodic renewal. The public components of service keys have published expiration times. All certificates should be used or exchanged by that time.[2] We see that there is no need to verify the structure of the blinded hashed nonce because the service key used to sign the message is the important factor, not the message content. If the customer substitutes something inappropriate, the result can only be an invalid certificate. If message 1 is not acceptable, e.g., if the payment in message 1 is inadequate for the requested service or is a credit card payment that is not authorized by the bank, then the protocol terminates, and the problem is communicated from the vendor to the prospective customer out-of-band.

The vendor must remember this sequence of messages in case message 2 is not received by the customer (see Section 2.7.) For this registration protocol, the service should consider message 2 to have been received after some period of time. For (space) efficiency, an acknowledgment message may be sent to the vendor:

Message 3. $C \rightarrow V : [Ack]_{K_{CV}}$

If message 3 is received, the vendor can be sure that the customer received message 2, and does not need to remember the sequence of messages for recovery purposes. However, the vendor may wish to keep a record of registration for other purposes.

A customer may wish to make use of his or her subscription from multiple machines, e.g., a base machine at home or office and a laptop machine when traveling. It may be considered too much of an inconvenience to require customers to transport the current unspent certificate for each subscription to the next likely platform. The vendor may therefore allow the customer to obtain a number of initial certificates, possibly at no additional fee, or for a nominal charge. Similarly, customers might be allowed to add an initial certificate during their subscription if beginning the use of a new machine. Another possibility is that customers might store

[2]An exchange consists of a certificate-redemption transaction wherein the vendor renews the customer's certificate under the new service key.

the certificate on their Web pages at an ISP (Internet service provider) and proxy requests through that page. This allows customers to share access to their accounts by sharing access to the Web pages, and also implies that customers trust the ISP with some profile information. The implications of such centralized proxy access are discussed in more detail in Section 2.9. Vendors will need to decide which policy best meets their needs.

K_{CV} is used to link protocol messages to one another. This becomes even more important when certificates are redeemed for transactions. We discuss further assumptions and requirements regarding this linking after presenting the certificate-redemption protocol.

2.4 Certificate Redemption

When customers want to make use of the service, they conduct a certificate-redemption protocol with V. Certificate redemption consists of certificate spending, transaction execution, and certificate renewal.

Message 1. $C \rightarrow V : \{[h(N_i)]_S, N_i, K_{CV}\}_V,$

$$[\textit{Request for transaction of type } S, \overline{h(N_{i+1})}]_{Kcv}$$

Message 2. $V \rightarrow C : [Approved]_{K_{CV}}$ (OR $[Not\ approved]_{K_{CV}}$)

Message 3. $C \leftrightarrow V : [Transaction]_{Kcv}$

Message 4. $V \rightarrow C : [\overline{h(N_{i+1})}]_S$

The transaction, message 3, is only done if message 2 is ($[Approved]_{K_{CV}}$). The other possibility is discussed in the next section. To prevent the customer from beginning a new certificate-redemption protocol before the current one completes, we delay the release of the new certificate $[\overline{h(N_{i+1})}]_S$ until the transaction ends. If the new certificate is released before the transaction, customers could run their own subscription servers that would proxy transactions for *their* customers. (This choice may force a certain structure on service implementations that allow concurrent transactions for legitimate customers; see Section 2.9.)

K_{CV} is a key used to protect the integrity of the session; it should be unique (chosen by C) for each session. If K_{CV} is compromised and then used in a later session, an attacker could create his or her own second field in the first message. By so doing, the attacker could hijack the subscription. Note that the confidentiality of these messages is not required to support this protocol; if confidentiality is required for other reasons, K_{CV} may be used for confidentiality as well.

Thus the uniqueness of K_{CV} is important to honest customers. But session integrity is important to vendors as well. Vendors would like to be sure that transaction queries are only processed in connection with a legitimate certificate renewal. Unfortunately, K_{CV} may not by itself be enough to guarantee integrity of a protocol session. One or more customers might intentionally reuse the same session key and share it with others.

Anyone who has this key could then submit queries. As long as such a query is submitted during an active legitimate session for which it is the session key, there is nothing in the protocol that distinguishes this query from legitimate ones. This allows for considerable sharing of subscriptions by effectively bypassing certificate spending. Other aspects of protocol implementation might prevent this. But, to be explicit, we assume that uses of K_{CV} are somehow rendered serial within a protocol run. For example, K_{CV} might be used in the protocol in a stream cipher. Alternatively, K_{CV} might be used as a secret nonce that is hashed with plaintext. The plaintext and hash are sent in each message. Each time a message is sent, the nonce could be incremented. (We make the same assumption for all protocols that use a session key, in order to protect the integrity of the session mentioned in this paper.) If something is done to make each use of K_{CV} in a protocol session unique and tied to previous uses within that run, then sharing subscriptions by this method becomes at least as inconvenient as sharing subscriptions by passing the unspent certificate around. (Furthermore, requiring at most a single TCP connection associated with the use of K_{CV} may also make sharing impractical and alert authorities to possible sharing. Sharing a single TCP connection among multiple entities is known to be very difficult, due to the need to maintain send and receive sequence numbers and to multiplex the different addresses for clients.)

As in the registration protocol, the vendor must remember the messages sent in this protocol (except for the transaction messages) in case the customer never received the new (blinded) certificate. For efficiency, an acknowledgment message may be added:

Message 5. $C \rightarrow V : [Ack]_{K_{CV}}$

2.5 Not Approved

If the response in message 2 is *Not approved*, then the protocol terminates. This response can only be authenticated with K_{CV} if the response is the same upon replay of message 1. So the response to a request for service is $[Not\ approved]_{K_{CV}}$ only if the nonce does not match the certificate. Other *Not approved* errors, including, e.g., double spending, are reported back unencrypted (out-of-band). This helps to ensure that the protocol is fail-stop (cf., Section 3.1).

If the customer is a valid subscriber who never received an initial certificate for the current key, it should be reflected in the vendor's records. The customer can then get an initial certificate in the usual manner. Offline appeal will be necessary for customers who feel they have been refused a legitimate transaction request. We have designed these protocols under the assumption that appeals will be automatically decided in favor of the customer as long as the customer does not appeal too many times.

2.6 Terminating a Subscription

Customer-initiated termination of a subscription is a variant of certificate redemption. While termination requires customers to turn in unspent

certificates, they do not get transactions or new certificates; however, customers may get refunds for any unused portions of their subscriptions.

Message 1.

$$C \rightarrow V : \{[h(N_i)]_S, N_i, K_{CV}\}_V,$$

$$[\textit{Request for transaction of type } (S \textit{ Termination}),\ C]_{K_{CV}}$$

.

Message 2. $V \rightarrow C : \{Refund\}_{K_{CV}}$ (OR $[Not\ approved]_{K_{CV}}$)

Whether or not refunds are available is a question of policy, decided by the vendor. They may be prorated on the basis of the vendor's policy for early termination. Should the subscription include multiple chains of certificates (e.g., for a workstation and a laptop), it may be required that all chains be returned to trigger a refund or the refund might be prorated accordingly. The refund may take any form, e.g., a credit to a credit-card account, e-cash, or even a mailed check.

In message 2, we encrypt using K_{CV}, since we do not require that customers possess private keys. As in registration, it may be required that customers authenticate themselves in some way in order to receive a refund. We do not consider this authentication part of our basic protocol structure. Although it is possible that such authentication could be integrated into our protocol, we do not discuss it.

As before, for efficiency, an acknowledgment message may be added:

Message 3. $C \rightarrow V : [Ack]_{K_{CV}}$

2.7 Recovering from Broken Connections

Protocols that break before the vendor receives an acknowledgment must be replayed in their entirety (except for the actual transaction, which is always skipped) with the same session key, nonce, and blinding factor. The protocols are designed not to release any new information when replayed.

Broken protocols are considered automatically acknowledged after some period of time has passed (i.e., the customer has that much time to recover from a broken connection). After that period of time, the protocols can no longer be replayed. This is not crucial for the redemption protocol, but is crucial for the registration protocol. After that period of time has passed, the subscription may be charged

We consider connection breaks that occur from the beginning to the end of the protocol. If a connection breaks after a new certificate has been acknowledged (message 5 in the Certificate Redemption protocol), the customer can simply initiate a new transaction with the new certificate. If a connection breaks after C receives message 4 but before V receives message 5, the customer can again simply initiate a new transaction.

Up to this point in the protocol, the customer will not yet have received a new certificate. So recovering from any connection breaks that occur prior

to this point in the protocol involves replaying the protocol. Vendors should keep a record of each protocol run until they receive the acknowledgment in message 5 (or until sufficient time has elapsed). Upon replay, the customer presents the same sequence of messages. The vendor will identify the presented certificate as spent and consult its recovery database. If the protocol is recoverable (i.e., has not yet been acknowledged), the vendor returns the stored response.

Notice that customers need never identify themselves when a broken connection occurs. Hence customers need not worry about being associated with any given transaction.

Disk crash and other media failures affect our system as well. It is unrealistic and unreasonable to expect customers to backup copies of subscription information every time they redeem a certificate. (It is often unrealistic to expect customers to make backups at all.) Therefore, customers must be allowed to reinitialize a subscription after a disk crash. How often individuals will be allowed to reinitialize over the course of a subscription is a policy decision for individual vendors. As in other appeals, customers would typically be expected to identify themselves. Another option is to provide customers with (distinct) backup initial certificates at registration, just as they may obtain initial certificates for multiple machines. This allows customers to recover from a disk crash without reregistering (assuming they have kept backups separately); however, it does provide additional subscription chains for the cost of one subscription. Finally, customers might proxy their requests through an online location that manages their certificates as well. As mentioned above, this allows customers to easily access their subscriptions from multiple platforms in multiple locations. It also makes it less likely that customers will lose certificates because one of their computers crashed. Unfortunately it also makes it easier, if no less risky, to share subscriptions. We discuss this further in Section 2.9.

2.8 Service Key Management

For unlinkable protocols to work, it is important that service keys not be "closely" associated with customers. For example, we do not want the vendor to be able to uniquely associate a service key with each customer, which would enable the vendor to associate transactions with customers.

2.8.1 *Committing to Service Keys*. A straightforward technique to overcome this potential vulnerability requires the vendor to publicly commit to all public authorization keys. This can be achieved by publishing information at regular intervals at a unique location well known to all potential customers of the service. An example publication format for each a service consists of the service type, expiration time, and signature confirmation key for signatures associated with the service.

2.8.2 *Subscription Termination*. Other than as a general security precaution, the primary reason to change service keys is to facilitate expira-

tion of subscriptions. When keys expire, customers obtain new certificates just as they did when signing up for a service initially. Service expiration can be structured in several different ways, each with advantages and disadvantages. We present some of these and briefly mention some of the tradeoffs. Which is most acceptable will depend on particular aspects of application and context. For the purposes of discussion, let us assume that the standard period of subscription is one year divided into months.

2.8.3 *Subscription Expiration*. One option is to have annualized keys that start each month; that is, twelve valid service keys for the same service at all times. This is convenient for the customer and similar to existing subscription mechanisms; however, it partitions those using a service into twelve groups, reducing the anonymity of customers accordingly. This may or may not be a problem. If subscriptions are annualized to quarters, it reduces the threat to anonymity; but this might still be unacceptable. It also reduces customer flexibility as to when subscriptions can begin.

An alternative is to have monthly keys that are good for all customers. Customers obtain twelve seed certificates when they subscribe, one for each month of the succeeding year. This does not reduce anonymity as the last option does. On the other hand, it requires that customers keep track of multiple certificates and requires issuing certificates well in advance of their period of eligibility. With monthly keys it is also that much easier to share a subscription, at least in monthly segments. Thus, the deterrent of inconvenience is somewhat reduced. Since a lost certificate now only means the loss of at most one month of service, the threat of subscription loss from sharing is also reduced. The deterrence against sharing provided by auditing (cf., Section 4) is similarly diminished.

Another option is to have all subscriptions end in the same month. Those subscribing at other than the beginning of the fiscal year will pay a prorated amount for their subscriptions. This avoids reductions in anonymity associated with monthly annualized keys. It also avoids the reduced deterrence to cheating associated with monthly keys. But it reduces customer flexibility in choosing the ending of the subscription. Another disadvantage is that subscription renewal is now all concentrated at one point in the year, creating an extremely unbalanced workload for the system that handles sign up and renewal. This would probably remain true even if customers were allowed to renew in advance. It could be diminished by splitting the year in half or even further. This creates a reduction in partitioning anonymity already mentioned.

2.8.4 *Early Termination of a Subscription*. Terminating a subscription early requires proving that the user is a particular customer and is spending a valid certificate. The customer will not get a new certificate, so there is no way for the customer to continue using the service. Note that early termination can even be customized, for example, so that it is available only to those customers who have already subscribed for at least a year. (Recall that customers reveal their identities or pseudonyms when

they terminate early.) Prorating refunds for terminated subscriptions removes one of the disadvantages of the third option for subscription expiration described above.

We have been describing customer termination of a subscription. Vendor termination of a particular customer or group is far more difficult (it may also be less important). In our current approach the only way to terminate a customer is to change the service key(s) for the remainder of the subscription and require everyone else to reinitialize their certificates with the new key. This creates tremendous expense and inconvenience, equivalent to what would be necessary if a service key were compromised.

2.9 Discussion: Subscription Sharing

In the protocols presented thus far, deterrence against sharing subscriptions has been based primarily on the inconvenience of sharing. This is akin to sharing a (hardcopy) newspaper or magazine subscription. There is nothing to prevent someone from sharing his or her newspaper subscription with neighbors. But the majority of subscribers prefer the convenience of having their own subscriptions to sharing with others, at least beyond their own households.

The analogy to the inconvenience of sharing a hardcopy subscription should not be taken too literally. Sharing a UST subscription is both more and less convenient than sharing a hardcopy newspaper. Unlike a hardcopy newspaper, the typical UST subscription need only be held long enough to perform the desired transaction, e.g., download a query response. The subscription can be passed on to another user, while the downloaded material is read or otherwise processed by the user. In this sense the inconvenience of sharing a UST subscription is not nearly as great as that of sharing a hardcopy newspaper. On the other hand, the inconvenience could potentially be higher: in UST, if the borrower fails to make the new certificate and nonce available to the subscriber, the subscriber has lost the subscription; but when a borrower fails to return a newspaper to a subscriber, the subscriber still gets the paper the next day. Of course, there is no official newspaper policy against sharing, and some sharing does occur. (Academic publishers typically differentiate between institutional and individual subscribers.) Unlike hardcopy newspapers, we assume that arbitrary sharing (e.g., beyond members of one's household) is usually against the stated policy.

It seems doubtful that a practical solution exists to fully protect vendors against sharing, given our goal of unlinkable transactions. (If transactions were linkable, one could detect widespread sharing through overuse of a subscription.) For example, subscriptions may be shared if a customer runs a subscription proxy server. This possibility is probably unavoidable and has many consequences. For example, moving the certificate-renewal phase of certificate redemption to before the transaction-execution phase has an advantage for potentially broken connections. First, it makes the need for recovery less likely because the customer will already have a valid certifi-

cate if the connection breaks during transaction execution. Thus the customer could simply initiate a new transaction. Second, the recovery itself would be simpler, since it need only be associated with certificate spending and renewal. There is no need to account for the transaction that occurred before the connection broke. Third, it allows legitimate customers to perform concurrent transactions. But there is an important disadvantage to such a protocol variant: it allows a cheating customer to run an even more fruitful subscription proxy server. Specifically, if the redemption protocol is altered, a customer running a proxy service could set up new transactions before active transactions had even completed. In the current protocol, only one transaction execution is possible at a time per subscription. In the variant, indefinitely many transactions can be run in parallel. If this is not considered a serious threat, then its advantages may make it an attractive variant. The basic protocol is designed to maximize inconvenience of subscription sharing by proxying transaction requests.

On the level of a small group of users, say ten or so, it would be easy enough to set up proxy-based subscription sharing; but it seems unlikely that the majority of subscribers have the technical expertise and motivation to do so. This is a large advance over simple password protection: virtually every subscriber is capable of sharing a password with arbitrarily many others, who can furthermore share that password with still others. Sharing via proxy service on a larger scale, i.e., running a pirate subscription service, has the usual attendant complexity of running a business. Such a business has the overhead and complexity of marketing, advertising, and maintaining service reliability. Perhaps more importantly, it has the potential disadvantage of being a focus for legal action. We will presently introduce further impediments to the proxy pirate.

We have already discussed the possibility of another form of proxying, where certificate management is proxied as well as transactions. We have already mentioned the advantages of mobility, platform-independence, and robustness against hardware failure that such proxying might enable. However, it also enables sharing of subscriptions in a much more convenient form. A lending customer need only share access to his proxy, presumably by means of a password. This effectively removes the inconvenience of sharing that was present for sharing transactions only. However, it is replaced with the risk of subscription loss. Borrowers of a subscription could steal it by stealing the current valid certificate. The lender could require a deposit to protect against such loss or charge enough to cover losses. Requiring a deposit or charging for fraudulent activity has historically been a key element in detecting and limiting fraud. More likely, the proxy could be so configured as to make theft of certificates difficult (e.g., not making the new certificate known to the customer). In Section 4.3 we consider an expanded protocol that increases the likelihood of a vendor detecting such sharing, should such measures be thought necessary.

3. EVALUATION

In this section we argue that our basic protocol meets all of the requirements mentioned in Section 2.1. First, we prove some of the protocol's functional properties. We then argue that these properties guarantee that the protocol satisfies the requirements. Unlike normal concurrent programs, we must show that these functional properties are satisfied in the face of a hostile environment, including both active and passive attacks. Therefore, to reduce the complexity of the correctness proof, we partition the proof into three stages:

—Prove that the protocols can be reasoned about as sequential programs by proving that they are fail-stop [Gong and Syverson 1998]. We justify this simplification in the next section.

—Prove that passive attacks do not reveal secrets. If secrets are revealed, subscriptions and money (credit) may be stolen, or services may be abused.

—Prove the functional safety properties as if the protocols were sequential programs [Chandy and Misra 1989].

3.1 Fail-Stop

Fail-stop protocols block when they are disrupted: they are immune to active attacks. So reasoning about fail-stop protocols is equivalent to reasoning about sequential programs.

To prove that our protocol is fail-stop, we must demonstrate that our protocols halt if there is an active attack. Our main observation is that the integrity of a protocol run is maintained for four reasons: (1) fields in a message are protected by the cryptographic operations (by assumption A4); (2) fields in a message are linked by a fresh shared key; (3) messages in a protocol are linked by the (changing) fresh shared key, K_{CV}; and (4) changes to fields in a protocol are immediately detectable, and changes cause the protocol to block. Thus, active attacks are not possible, and we need only consider passive attempts to compromise confidentiality.

What guarantees that the customer chooses K_{CV} fresh for each protocol run? For example, the customer may have an incentive to use the same K_{CV} for multiple runs to facilitate sharing. As discussed in Section 2.4, other aspects of the protocol implementation are likely to make sharing K_{CV} impractical. These aspects focus on requiring clients to maintain a local state associated with its use. However, as discussed below, the benefit is minimal and the risk of losing control of the subscription is severe.

So we assume that the customer chooses the key fresh for each protocol run, and that the protocol is therefore fail-stop.[3]

[3]If the customer reuses the same K_{CV} for several protocol runs, the protocol is not fail-stop. Based on this, one possible active attack is to substitute the corresponding field from a previous run for the second field in message 1 of the current certificate-redemption protocol

3.2 Passive Attacks

We distinguish passive attacks by outsiders from passive attacks by the vendor. All secrets in all basic UST protocols are hidden from outsiders by encryption and/or blinding and hashing. For example, K_{CV} is protected by the vendor's key. In the certificate-redemption protocol, the new nonce is both hashed and blinded. Secrets are protected from the vendor by blinding. For example, the hashed nonce that will be signed by the vendor is hidden from him by blinding. Since secrets are hidden by these mechanisms, passive attacks will not reveal any secret information.

3.3 Safety

Our safety properties are proved in the usual way, by observing that they are preserved across protocol steps. In all cases, we observed that we do not need to worry about interactions between two protocol executions because modifications to state variables in any given execution are independent of modifications to the variables in any other execution. Therefore, individual protocol executions can be treated as atomic. In fact, we see that the first two safety properties do not hold during a protocol execution, but only at boundaries (before or after an execution). In contrast, the third property is maintained during protocol execution. Indeed, it is only interesting if the protocol is not viewed as atomic because it considers state information that only makes sense during protocol execution.

3.3.1 *Definitions.* A *subscription chain* c is any sequence of certificates $\langle c_1, \ldots, c_n \rangle$. We often simply call a subscription chain a chain.

An *annotated chain* is a chain, together with an annotation for each element of the chain, $\langle (c_1, x_1), \ldots, (c_n, x_n) \rangle$.

The possible annotations are unspent u, active a, and spent s. The annotation $A(c)$ of chain c is the sequence of annotations of the elements of c, $\langle x_1, \ldots, x_n \rangle$, and $A(c)_i = x_i$.

A chain c is *well-formed* (*WFc*) if it is annotated, and all elements except the last are spent. That is, $A(c)_i = s$ for $1 \leq i < n$, and $A(c)_n \in \{u, a, s\}$ where n is the length of c.

---

run, using the same K_{CV}. At the end of the redemption protocol's run, the customer is issued an already spent certificate and must appeal to restart the subscription.

That the protocol can proceed in the face of this attack shows that it is not fail-stop if K_{CV} is used in more than one protocol run. Still, this attack does not result in the attacker seeing any messages that he could not have seen without the attack. Thus the protocol appears to be *fail-safe* [Gong and Syverson 1998]. For example, all the messages before message 4 are the same with or without the attack, and message 4 should have been sent in the previous protocol run, from which the attacker took the second field of message 1. Nonetheless, the protocol is not fail-safe. An attacker could substitute an *Approved* in message 2 from an earlier run that used K_{CV} for a *Not approved* message in a later run. The customer then sends a new transaction request under K_{CV} that would not have been sent without the attack. (That the customer must have intended to send the request when he began the protocol run does not change the fact that the protocol is not fail-safe.) We examine below the affect of these attacks with respect to functional invariants.

C is a bag (multiset) of chains.

$prefix(c)$ is the set of all prefixes of c, i.e., $prefix(c) = \{x : \exists t(x;t = c)\}$. (We denote concatenation by ";".) $Prefix(C)$ is the bag of all prefixes of all chains in C, i.e., $Prefix(C) = \uplus_{c \in C} prefix(c)$. $\uplus$ represents a union of bags, i.e, $\uplus_{i \in I} X_i$ represents the bag of all members of any X_i, where either I or any of the X_i's may be bags or merely sets. $\subseteq$ represents the subbag relation in an obviously analogous manner.

3.3.2 *Safety Properties*. Chains begin at protocol registration (except for replacement of lost certificates, etc.). They are extended during certificate redemption and reach a steady state upon termination (or possibly loss of certificate, etc.) The chain annotations map to the protocol steps as follows. A chain begins or grows when the vendor sends a certificate to the customer. If the custom has not spent a certificate in the chain previously, then this is the beginning. At this point the certificate is unspent. The certificate becomes active when the customer sends it to the vendor during the redemption or termination protocol. It becomes spent when the vendor sends the customer a new certificate (or a refund message in the termination protocol). One consequence of this mapping between the protocols and the above definitions is the following invariant.

Fact. I1. Subscription chains are persistent.
STABLE: $K \subseteq Prefix(C)$. This can also be written as
$\forall K[K \subseteq Prefix(C)\{Protocol\ Run\}K \subseteq Prefix(C)]$

The remainder of the invariants that we present are used to argue that the protocol meets one or another of the requirements set out in Section 2.1.

Fact. I2. Number of payments = Number of chains is invariant.
INV: $|P| = |C|$

By assumption A6, if vendors accept payment from customers for a service, then they will provide the customers with new certificates for that service. And, by assumption 5, all legitimate messages are eventually received. So there are at least as many chains as there are payments. From the protocol description and the fail-stop property, we cannot get more than one certificate from a run of a protocol session of the registration or redemption protocol. The recovery protocol of the redemption protocol's incomplete runs is fail-safe by definition: it only returns previously sent messages (cf., Section 2.7). Recovering from an incomplete registration is similarly fail-safe. So replays of the registration protocol are idempotent. Thus, a registration protocol run spawns exactly one chain, and a redemption protocol can only extend a chain. The termination protocol does not return a certificate at all. All chains require a certificate as an initial element. Since the termination protocol doesn't provide a certificate, it is not possible to spawn a new chain.

This invariant as stated is appropriate when one initial certificate is issued for each subscription. As we discussed, there are reasons why one might want to issue some number of initial certificates n at registration or allow issuing up to n new initial certificates during the course of a subscription. In these cases the invariant becomes $n|P| = |C|$ or $|P| \leq |C| \leq n|P|$, respectively. The proof, however, is roughly the same.

Fact. I3. Each subscription chain has at most one active certificate at any time.
INV: $\forall c \in C[|A(c)\restriction a| \leq 1]$ ($\restriction$ represents restriction).

Invariant I3 is a consequence of the following invariant:

Fact. I4. Well-formed chain invariant:
INV: $\forall c \in C.WFc$

The well-formed chain invariant is stronger than we need (or possibly even want). What we want is I3. The stronger well-formed chain invariant implies I3, but it precludes, for example, implementations that issue new certificates before the transaction, which would simplify recovery from broken connections. But this is a variant on the basic protocol. The well-formed chain invariant is a consequence of the basic protocol description.

Note the following property, which describes terminated subscriptions:

Fact. I5. Terminated subscription chains never grow.
STABLE: $\forall c \in C[A(c) = s^{|c|}]$

By property I4, all but the last element of any chain is spent. The termination protocol spends the last element in a chain without extending the chain (i.e., issuing a new certificate). Once termination has taken place, all the elements in the chain are spent. The only way for an existing chain to obtain a new element is through the redemption protocol (since the registration protocol can only start new chains). To initiate a session of the redemption protocol, an unspent certificate is needed. So a terminated subscription chain cannot grow and consists entirely of spent certificates.

The above arguments are made under the assumption that the customer chooses K_{CV} fresh for each protocol run (making the protocol fail-stop). If we do not assume that the customer chooses K_{CV} fresh for each protocol run, we can still demonstrate that the safety properties are maintained. There are two cases.

(1) The repeated use of K_{CV} could allow the integrity of message 1 to be violated, as in the first attack in Section 3.1. However, the attacked protocol run could also have been run under the assumption that the customer does choose K_{CV} fresh each time. For example, just as an attacker could substitute a second field of message 1 from a previous run for the second field in a current run, the customer could repeat that earlier blinded hash himself, within the second field of a current run.

Since this makes the protocol run identical to a run of the fail-stop protocol, the safety properties are maintained.

(2) Repeated use of K_{CV} could allow the integrity of some other message to be violated. But, except for message 1, all messages of all protocols consist of a single field. So our assumption about cryptographic field integrity amounts to an assumption about message integrity. We need therefore only consider the substitution of whole messages in active attacks. But these have no affect on the safety properties given above. For example, the attacker could substitute a previous *Approved* message for a *Not approved* message. In this case, the customer initiates the transaction even though the vendor expects no further messages. But this does not affect any of the invariant or stable properties that we show.

3.4 Fraud

Recall that there are two categories of fraud: subscription sharing and the use of illegitimate subscriptions.

Subscriptions can be shared in two ways: (1) sharing within a protocol session, and (2) sharing the certificates themselves. Illegitimate subscriptions are either (1) new, fraudulently obtained subscriptions or (2) subscriptions that were meant to be rendered invalid but are still usable. There are several ways to fraudulently obtain new subscriptions (a) claim you registered and never received a new certificate, (b) obtain more certificate chains than were paid for due to a protocol failure in the redemption or registration protocol, and (c) claim you have lost your chain and require a new seed certificate.

We wish to eliminate all activity related to high-volume fraud (i.e., R1). Sharing within a protocol session cannot be done at high volume because there are a fixed number of transaction requests allowed per redemption. Sharing certificates cannot be done at high volume because we show that there is at most one active certificate at any time per subscription chain (I3). It is still possible for an attacker to set up an unsanctioned subcontracted subscription service; however, the attacker is unable to do high-volume sharing within a protocol session, nor can he use a single subscription in a parallel fashion. Further, natural delay in processing transaction requests precludes obtaining large numbers of transactions in a serial fashion because the certificate sharer won't be able to obtain a new certificate to facilitate new transaction requests until previous transaction requests have been serviced. While by these arguments a customer cannot commit high-volume fraud, he can nonetheless still set himself up as a centralized service, albeit at a less than high volume. Should this threat be of concern, other countermeasures (e.g. an audit) can be added to the protocol (discussed in the next section}.

We now address obtaining new subscriptions illegitimately. By property I2, it is impossible to obtain a new subscription without payment. The only possible exception is to obtain a new subscription via off-line appeal (cases

(a) and (c) above). A customer who does this repeatedly risks detection via the high number of appeals associated with one account. But the protocol doesn't inherently authenticate customers. There is nothing to prevent an attacker from finding a list of customers and claiming to have registered under any one of them. In this way an attacker could obtain a large number of new illegitimate subscription chains without attracting attention. However, to prevent high-volume fraud, we could require off-line authentication or use of the audit mechanisms discussed in the next section.[4]

The only source for high-volume fraud yet to be discussed is the continued use of subscriptions intended for invalidation. By property I5, terminated subscriptions cannot be used. For subscriptions that are not explicitly terminated, ordinary properties of key expiration (i.e., the fact that a service doesn't work using expired keys) terminate expiring subscriptions.

We now turn our attention to low-volume fraud. We wish to detect and reduce activity related to low-volume fraud (i.e., R2). We do not concern ourselves with low-volume sharing within a subscription a sufficient threat, since the expected loss is minimal and prevention is virtually impossible. Direct sharing of certificates is minimized, since the inconvenience should outweigh the cost of subscribing (assuming this cost is not prohibitively expensive). Also, sharing a certificate requires the sharer to trust the recipient to return an unspent certificate to the sharer instead of keeping it, turning it in for a refund, or passing it on to someone else who may not return it. Determined customers could still set up a proxy service that would permit relatively low-volume subscription sharing with minimal risk of loss. There remains the risk of illegal activity. Since illegal activity cannot occur at high volume, it does not seem likely to be outweighed by monetary benefit. If this is nonetheless viewed as a threat, we can use the audit mechanisms discussed below (in Section 4) as a countermeasure.

Obtaining new subscriptions fraudulently can be detected and minimized, since each request for a new certificate by an individual customer is recorded. If this is a concern, then we can require authentication mechanisms during registration and off-line appeals or use the audit mechanisms in the next section. It is impossible to use terminated subscriptions for low volume fraud for the same reasons that they could not be used for the high-volume fraud.

Finally, we argue that theft of payment is not possible (R3), due to the confidentiality of the first registration message, the fact that K_V is only used in the UST protocols, and that the payment is integrity-protected with respect to the rest of the message, which has meaning only in the context of the current registration. Direct theft of a refund is similarly protected by the confidentiality and freshness of K_{CV}. Indirect theft of a refund, by direct theft of a subscription, is addressed in Section 3.6.

---

[4]For example, in the protocols of the next section, using I_{audit} ties the transaction requester back to a particular subscription.

3.5 Customer Privacy

We have two requirements for customer privacy: anonymity (5) and protection against profiling (4), both of which we address at once. The amount of privacy that customers have from each other and from the vendor depends on many factors, for example, the number of customers using a given service key and their patterns of usage. It is therefore infeasible to provide a rigorous characterization of the privacy provided by the basic protocol. We can, however, note the conditions that make it impossible for the vendor to link one spent certificate in a chain to the next. First, the customer's identity is not revealed by the connection between the vendor and the customer (or the communications medium in the connectionless case). Mechanisms for providing this assurance are discussed in Section 6.3. Second, the customer chooses nonces at random. Third, blinding messages works, i.e., in absence of the blinding factor, identities cannot be correlated to the messages themselves.

3.6 Guarantee of Contracted Service

We now argue that customers cannot be denied a service for which they contracted (i.e., 6), given that vendors are willing to provide the service (i.e., 6). This assumption covers both the competence and the intent of the vendor. For example, we assume that the vendor generates the right certificate: $\text{certificate}_C = \mathit{unblinded}([\overline{h(N_C)}]_S$.

By assumption A5, every message is eventually received if the sender keeps trying to send it. So we may proceed on the assumption that all sent messages are received. (If receipt of a message takes exceedingly long, offline appeal is available to rectify this.) Hence our analysis of 6 will proceed on the assumption that all sent messages are received.

Assuming the customer chooses a good K_{CV} and a fresh nonce, the redemption protocol guarantees the transaction request will be processed and a new certificate issued. (If a customer unluckily chooses a spent nonce, he should be able to appeal to the vendor offline. With an adequate nonce space, this should be very unlikely, certainly less likely than simply losing certificates. So vendors should be leery of more than a few total requests of this type.) The nonce in the certificate the customer spends is only known to him or her and the vendor (due to encryption). Thus it is not possible for others to steal the unspent certificate. Also, the spending of that certificate is tied to the new candidate certificate and transaction request by K_{CV}. So the vendor who follows the protocol will only service valid transaction requests and will only sign and send the legitimate candidate certificate to the customer. And only that customer knows the associated nonce. No one can have his request serviced at the expense of a valid customer. Similarly, no one can steal a subscription, either for their own use or to turn in for any potential refund.

4. UST PROTOCOL WITH AUDITING

As we have noted above, the primary deterrence against sharing of subscriptions is inconvenience and possible loss of the remainder of the subscription if the subscription borrower fails to return a token. But it may be useful to monitor subscription use, to see whether a subscription is being used more heavily than expected. This may indicate fraud.

We now describe a protocol that increases the vendor's ability to detect unauthorized sharing of subscriptions. To distinguish protocols discussed in the paper thus far from those we are about to describe, we will collectively refer to the protocols in this section by the term USTA.

4.1 Registration for UST with Audit

The USTA registration protocol is similar to the original UST registration:

Message 1. $C \rightarrow V : \{Payment, I_{audit}, K_{CV}\}_V$,

$$[Request\ for\ certificate\ of\ type\ S, C, \overline{h(N_i)}]_{K_{CV}}$$

Message 2. $V \rightarrow C : [\overline{h(N_1)}]_S \quad (\text{OR } [Not\ Approved]_{K_{CV}})$

We now describe the new fields in the protocol messages: in message 1, I_{audit} is some information that a customer is typically unwilling to share. For example, it may be the personal information the subscriber used when purchasing the subscription: name, address, and credit card number. The customer will be obliged to present this information when challenged in an audit, described in Section 4.3.

4.2 Certificate Redemption for UST with Audit

Message 1.

$$C \rightarrow V : \{[h(N_i)]_S, N_i, K_{CV}\}_V,$$

$$[Request\ for\ transaction\ of\ type\ S, h(N_i, I_{audit}, Salt), \overline{h(N_{i+1})}]_{K_{CV}}$$

Message 2. $V \rightarrow C : [Approved]_{K_{CV}}[\overline{h(N_1)}]_S \quad (\text{OR}[Not\ approved]_{K_{CV}}$ OR $[Audit]_{K_{CV}})$

Message 3. $C \leftrightarrow V : [\mathbf{Transaction}]_{K_{CV}} \ldots$

Message 4. $V \rightarrow C : [\overline{h(N_{i+1})}]S$

The protocol is largely the same as before, except that another field (which is only used if an audit occurs) is added to message 1; message 2 might now be *Audit* as well as *Approved* or *Not approved*. If the response is *Audit*, then a special audit occurs in which C must present some proof that he or she is a valid customer within a short period of time. This protocol is set out presently. In particular, C must prove knowledge of I_{audit}, which was sent to V during registration. If this is satisfactory, a new certificate is issued. If it is not satisfactory or if C does not comply, then the protocol

terminates, and the certificate is logged along with a note that it was used during a failed audit. In either case, no transaction takes place and so audited customers are not linked to specific transaction requests. The main purpose of audits here is to serve as a secondary deterrent to sharing a subscription with a noncustomer. (The primary deterrent is the inconvenience of passing the certificate back and forth between those sharing, as compared to the cost of obtaining another subscription.) During an audit, the customer must reveal I_{audit}. So the borrower of a subscription must either present the subscriber's personal information or the borrower will fail the audit. In the latter case, the subscriber looses the subscription (chain) because the subscriber knows that he or she is committing fraud and will not share the I_{audit} with the borrower. In the former case, the vendor will get statistical information about usage of the subscription. If a particular subscription is being overused, that subscription can be flagged, and the token is not replaced in a subsequent audit.

As always, we assume that something is done to make each use of K_{CV} in a protocol session unique and tied to previous uses within that run. For example, if a transaction itself has multiple steps, or multiple transactions are allowed for each spent certificate (e.g., message 3 repeats), K_{CV} should change for each use. Thus, sharing subscriptions by means of sharing within a protocol run remains at least as inconvenient as sharing them by passing the unspent certificate around. As we noted, although it is serial, it is not centralized. We further note that, in USTA, this sharing is not as risky as certificate sharing, since there is no possibility of audit once the transaction is submitted. Should this be considered an important threat, the suggestion above to allow only a single transaction per protocol run solves this problem as well.

4.3 Audit

The audit protocol follows:

Message 1.
$C \rightarrow V : \{[h(N_i)]_S, N_i, K_{CV}\}_V,$

$$[\textit{Request for transaction of type } S, h(N_i, I_{audit}, Salt), \overline{h(N_{i+1})}]_{K_{CV}}$$

Message 2. $V \rightarrow C : [Audit]_{K_{CV}}$

Message 3. $C \rightarrow V : \{C, N_i, I_{audit}, Salt\}_{K_{CV}}$

Message 4. $V \rightarrow C : [\overline{h(N_{i+1})}]_S \quad (\text{OR } [Not\ approved]_{K_{CV}})$

Message 1 in the certificate redemption and audit protocols is, of course, the same. The "audit" field in message 1, $h(N_i, I_{audit}, Salt)_{K_{CV}}$, is included for audit purposes only. The hash argument must be revealed during an audit. The only way a subscription borrower without I_{audit} can produce a proper audit field in message 1 is if he contacts someone who does possess I_{audit} before each execution of the redemption protocol. This adds greatly to

the inconvenience of such sharing. *Salt* is a random value to prevent the vendor from associating customers with transactions via a simple exhaustive search. (Even if no audit occurs, once the vendor knows the nonce value in the spent certificate he or she might try to associate the transaction request with the I_{audit} registered for that customer by searching through the list of known secrets. The random value *Salt* prevents the vendor from being able to confirm the "guessed-at" associations.) The nonce N_i is included in the audit field to further complicate sharing. A customer might be willing to lend the subscription to someone even if the customer does not trust that person with I_{audit}. If the audit field does not contain N_i, the borrower needs to contact the legitimate customer only during an audit to preserve the subscription. At this point, with N_i in the audit field, it is too late .

Similarly to the basic certificate-redemption protocol, if message 4 is

$$[Not\ approved]_{K_{CV}},$$

the protocol terminates. Unlike the basic certificate-redemption protocol, there is no transaction phase. So there is no direct link between any identifying information revealed in the audit and any particular transaction. However, by exercising the audit check frequently or at strategic times, the vendor can learn both the customer's usage frequency and patterns. This might allow the vendor to correlate later transactions (and possibly earlier transactions) with the particular customer. The customer might counter this limitation by employing a masking scheme on top of the basic protocol. However, this can increase the load on the subscription service considerably. Customers might also counter such vendor analysis by delaying ordinary transaction requests for a random amount of time following an audit. This places no extra burden on the subscription service, but may cause customers inconvenience substantially beyond that of the audits themselves. Since audits are a secondary deterrent to abuse, they might be conducted infrequently. The tradeoffs between threats to anonymity and the deterrence effect on subscription sharing are difficult to assess *a priori*. Hence, exactly how frequently audits should be made is currently difficult to say.

As in UST, the service in USTA must remember the sequence of messages in any run of this protocol in case of a broken connection. If the response in message 2 is *Audit*, V should keep a record of the protocol run even if C properly identifies himself upon reestablishing the connection. It may be that a cheater broke the connection and then quickly notified the legitimate customer of the audit. If some customer breaks an audit protocol repeatedly, a vendor may become suspicious and decide not to renew the customer's certificate.

For efficiency, an acknowledgment message may be added:

Message 5. $C \rightarrow V : [Ack]_{K_{CV}}$

In message 3 of the audit protocol we explicitly use K_{CV} as an encryption key. In other cases we encrypted it for the vendor using V. (In practice, the symmetric key K_{CV} is typically used in favor of the computationally expensive public key V.) However, it is essential that V *not* be used in message 3, since that would allow customers to share their subscriptions and produce responses to audit challenges without revealing their secret I_{audit} to those with whom they shared.

4.4 Terminating a USTA Subscription

Customer-initiated termination of a USTA subscription is a variant of USTA certificate redemption, it does not, however, trigger an audit. Termination requires the customer to prove she or he knows I_{audit} (unlike termination in the basic unlinkable serial protocol) and has an unspent certificate. If a subscription is terminated, the customer may be refunded a portion of his or her subscription cost. Since I_{audit} is assumed to be something the subscriber is unwilling to share, knowledge of I_{audit} demonstrates that the subscriber is terminating the subscription and is entitled to the refund.

Message 1.

$$C \rightarrow V : \{[h(N_i)]_S, N_i, I_{audit}, K_{CV}\}_V,$$

$$[\textit{Request for transaction of type } (S \textit{ Termination}), C]_{K_{CV}}$$

Message 2. $V \rightarrow C : (Refund)_{K_{CV}}$ (OR $[Not\ approved]_{K_{CV}}$)

Refunds may be prorated based on the vendor's policy for early termination. Should the subscription include multiple chains of certificates (e.g., for a workstation and a laptop), one termination transaction should be possible per chain. In message 2, we encrypt using K_{CV}, since we do not require that the customer possess a private key.

As before, an acknowledgment message may be added for efficiency:

Message 3. $C \rightarrow V : [Ack]_{K_{CV}}$

4.5 Reducing High-Volume Fraud by Auditing Subscription Use Levels

Given an appropriate sampling rate, the level of subscription use can be determined by the frequency at which a subscription appears in an audit. The vendor can make use of this fact by charging according to the use level. By doing so, the vendor can effectively reduce the profit margins (and hence incentive) associated with high volume, since these subscriptions appear more frequently in an audit and are charged at a higher subscription rate. Alternatively, someone intending to commit fraud could purchase additional subscriptions. However, this again has the desirable effect of increasing revenue and satisfying our goal of reducing high-volume fraud.

4.6 Audit with Client-Side Certificates

In USTA, I_{audit} is used as proof of the subscriber's identity. Client certificates could also be used to the same effect, by challenging the subscriber to use his or her private key, for example. The value of such certificates (in this application) depends upon how thoroughly the client was vetted before being granted a certificate, and on the consequences of private key compromise. For example, even if certificates must be obtained in person with government identification, if the certificates are not used anywhere but in UST, the consequences of compromise are not severe. Even if the certificates can be used in many applications, if they can be revoked easily without penalty to the user, the consequences of client certificate-sharing are not severe.

So a UST vendor cannot use client-side certificates in Audit until most customers have them and unless clients perceive that their private keys must indeed be kept private.

5. APPLICATIONS OF UNLINKABLE SERIAL TRANSACTIONS

Until now we have focussed on basic subscription services as the application of unlinkable serial transactions. We now explore both expansions of the basic subscription application and other applications as well. We simply describe these applications without giving full details on how to adapt the unlinkable serial transactions for them. Generally, it will be straightforward to see how to do so.

5.1 Pay-Per-Use Within a Subscription

Certain transactions may require extra payment by a customer. Next, we describe a means to allow pay-per-use within a subscription. The vendor becomes a mint for a simple, single denomination, digital tokens. Digital tokens are to digital cash roughly as tokens in a game arcade are to coins. The vendor may bill for these tokens by credit card, or some other mechanism.

During the transaction phase (message 3 in the certificate-redemption protocol), the customer spends previously purchased tokens. How do we guarantee that the customer pays the vendor for the pay-per-use transaction? Either the vendor never releases the new blinded certificate (message 4) unless he is paid or we assume some protocol for fair exchange [Franklin and Reiter 1997; Camp et al. 1996]. The latter choice properly partitions responsibility without complicating recovery.

There are alternatives to this protocol. For example, certificates could include a credit balance, which must be paid periodically. Payment is made as a transaction. There is no harm in this transaction identifying the customer because it is only for purposes of payment. The main limitation on this approach is that the credit balance increases monotonically. This may allow the vendor to link transactions, and even tie them to particular customers.

5.2 Third-Party Subscription Management

Vendors may be interested in making the anonymity afforded by our approach available, but may be less enthusiastic about the necessary overhead in maintaining a subscription, e.g., keeping track of spent certificates. Along with the ordinary overhead of maintaining subscriptions, handling billing, etc., vendors may choose to hire out the management of subscriptions. It is straightforward to have the vendor simply forward transaction requests to a subscription management service, which then negotiates the business (certificate management) phase of the protocol with the customer. Once this is completed, the transaction phase can proceed between the vendor and the customer as usual.

5.3 Multivendor Packages and Discount Services

For multivendor packages, we can purchase what is effectively a book of coupons good at a variety of individual vendors. A coupon book works by vendors authorizing the package vendor to issue certificates for their services. Customers then engage in a protocol to obtain the basic certificates.

If the coupons in the book are meant to be transferable, there is nothing more to the protocol. If, however, they are not, we must add a serial unlinkable feature to make sharing more cumbersome. In this case, when a customer submits a certificate for a service he or she must also submit a package certificate. The package certificate must be updated as in the basic protocol. Service certificates are not to be updated: they can only be redeemed once. Vendors could all be authorized with the necessary key to update the package certificate. Alternatively, the processing of the certificates could be handled by the package issuer, as in the third-party application of unlinkable serial transactions just given. Notice that individual vendors need not be capable themselves of producing coupons for their own services. It is enough that they can confirm the signatures associated with their services.

Package books such as those just described often offer discounts as a sales incentive, over the vendors' basic rates. Another form of discount is one that is made available to members of some group. Unlinkable serial transactions are useful for allowing someone to demonstrate such membership without revealing his or her identity. Depending on the application, the various vendors offering discounts can sign new certificates or signing can be reserved for some central membership service in association with any request for discount at a vendor. Again, the latter case is similar to the third-party application above.

5.4 Membership and Voting

The example just mentioned shows that the basic idea of unlinkable serial transactions can have applications outside of commerce. Specifically, it should be useful for any application in which membership in some group must be shown, and where the inconvenience of sharing a serial certificate

and the risk of audit outweighs the advantages of spoofing group membership, including some applications requiring proof of age or residency.

As another example, consider a voter registration certificate. At voting time, the voter spends his certificate, is issued a new certificate, and votes. The new certificate is signed by a key that becomes valid after the current voting period expires, so voters cannot vote twice. In this case, there is no possibility of sharing the certificate for a single election. If there is concern that formerly eligible voters continue to vote once their eligibility has expired, certificate keys could be subject to occasional expiration between elections. Ineligible voters would then be eliminated, since they would be unable to register for new seed certificates. This example is only meant to show the ease with which UST can be used for purposes of showing membership in a group. Voting has many properties, e.g., assuring that votes have not been altered or eliminated from the final tally, that we do not claim to have even approached. Voting protocols intended to provide many of the properties of a fair election are more complex. Some of these even make use of blind signatures; although, not surprisingly, their systems are rather different from the ones given here [Fujioka et al. 1993; Cranor 1996].

6. RELATED WORK

6.1 Digital Cash

Digital cash, especially anonymous e-cash as presented by Chaum et al. [1990], is characterized by several requirements [Okamoto and Ohta 1992], that is: being independent of physical requirements, being unforgeable and uncopyable, giving the capacity to make untraceable purchases, being offline, transferable, and subdividable. No known e-cash system has all of these properties, and certain properties, especially e-cash, which can be divided into unlinkable change, tend to be computationally expensive.

E-cash can either be online or offline. In an online scheme, before completing the transaction, the vendor can verify with a bank that the cash has not previously been spent. In an offline scheme, double-spending must be detectable later, and the identity of the double spender must then be revealed. Previously agreed-upon penalties can then be applied that make double spending not cost-effective.

Chaum's notion of *blinding* [Chaum 1985] is a fundamental technique used in anonymous e-cash and in assigning pseudonyms. A bank customer may want a certain amount of e-cash from the bank, but may not trust the bank not to mark (and record) the e-cash in some way. One solution is for the bank to sign something for the customer that the bank cannot read, while the customer presents the bank with evidence that the bank is signing something legitimate. Chaum's blinding depends on the commutativity of modular multiplication operations. Therefore, the customer can create an e-cash certificate and multiply it by a random number called a blinding factor. If the bank signs the blinded certificate, the customer can

then divide out the blinding factor. The result is the unblinded certificate signed by the bank. But the bank does not know what it signed.

How can the customer assure the bank that the blinded certificate is legitimate? In Chaum's scheme, the customer presents the bank with many blinded certificates that may differ in serial numbers, but not in denomination. The bank chooses the one it will sign and asks the customer for the blinding factors of the others. If the randomly chosen certificates turn out to be legitimate when unblinded, the bank can have confidence that the remaining blinded certificate is legitimate too.

One online e-cash scheme is presented in Simon [1997]. To obtain an e-cash certificate that only he or she can use, a customer presents the bank with a hash of a random number. The bank signs an e-cash certificate linking that hash with a denomination. To use the e-cash, the customer reveals the random number to a vendor, who in turn takes the e-cash to a bank. Since hashes are one-way functions, it would be very hard for someone other than the customer to guess the secret that allows spending the e-cash. After the money is spent, the bank must record the hash to prevent it from being spent again. This scheme can be combined with blinding to hide the actual e-cash certificate from the bank during withdrawal.

One offline e-cash scheme is presented in Franklin and Yung [1992]. There, the bank signs blinded certificates. To spend the e-cash, the customer must respond to a vendor's challenge. The response can be checked by inspecting the e-cash. Double spending is prevented because the challenge/response scheme is constructed so that the combination of responses to two different challenges reveals the identity of the customer. As long as the customer does not double spend, his identity is protected. Nobody but the customer can generate responses, so the customer cannot be framed for double spending.

It may be the case that truly anonymous unlinkable e-cash enables criminal activity. Several key escrow- or trustee-based systems [Brickell et al. 1995] have been developed that can reveal identities to authorities who obtain proper authorizations.

Our notion of unlinkable certificates came from asking the following question: What else shares some of the features of digital cash? Unlinkable certificates share many of these features: they must preserve the user's anonymity and not be traceable, and they must protect the issuer and not be forgeable or copyable. Unlike e-cash, however, transferability is not desirable. Also, unlike e-cash, UST is not a payment vehicle because spending, i.e., presenting a UST certificate, does not change the funds available to the customer. He starts with a certificate, gains access to a service in exchange, and ends up with both the service and a certificate. The UST certificate is not cash but an electronic membership pass. Like a membership pass, and unlike cash, its value is tied to the amount of time remaining in the membership.

We use hashing random numbers and blinding in our development of unlinkable certificates. Our unlinkable certificates differ from Chaum's

pseudonyms [Chaum 1985], which are an alternative to a universal identification system. Each pseudonym is supposed to identify its owner to some institution and not be linkable across different institutions. Unlinkable serial certificates are designed to be unlinkable both across institutions and across transactions within a single institution. In particular, we want the vendor to be unable to link transactions to a single customer, even if that customer had to identify himself initially (i.e., during the subscription process). At the same time, the vendor needs to be able to protect himself against customers that abuse his service.

Our blinding also differs from the usual approach. Typically, some mechanism is necessary to assure either the issuing bank or receiving vendor that the certificate blindly signed by the issuer has the right form, i.e., that the customer has not tricked the signer into signing something inappropriate. We described Chaum's basic approach above. By moving relevant assurances to other parts of the protocols, we are able to eliminate the need for such verification. The result is a simplification of the blinding scheme.

6.2 Authentication, Authorization, and Payment

We can obtain access to data or services online by a variety of means. Those that are not freely available require some form of authentication, authorization, payment, or a combination of the three. Some are available to anyone, e.g., most Web pages can be downloaded by anyone from any location. Some requests provide an identity (e.g., anonymous ftp does, although there is typically virtually no authentication). Some, such as the Kerberos-like authentication protocols, require authentication and authorization. Some require payment and authentication, e.g., subscription services. UST is a mechanism that can be used for a variety of authorization purposes, subscriptions, access control, etc. But it is not the same as any of the authentication, authorization, or payment mechanisms that have already been devised.

UST is not a simple password-based access scheme: it can be used anonymously, so that the authorization verifier cannot link the password to the authorizee. It is not a pseudonym scheme: when it is used to provide anonymity, it provides perfect forward and backward anonymity, i.e., linking one authorization attempt to the true individual does not reveal anything about any future or past authorization attempts by that individual.

UST is not like current software-based one-time password schemes (e.g., S/Key [Haller 1994]) because neither the authorizee nor the authorizor or authorization verifier can predict future passwords. It is not like time-based one-time passwords (e.g., SecureID token) in that we don't require clock synchronization between authorizee and authorization verifier.

An approach to anonymous membership authentication which has many goals in common with ours is presented in Schechter et al. [1999]. In their protocol, for each transaction, the vendor completes a key negotiation

protocol with an anonymous subscriber by (essentially) responding to all subscribers. Only the subscriber who initiated the protocol can interpret the response, but the vendor never learns which subscriber is negotiating, only that some subscriber is. The vendor's response requires a public key operation for each possible subscriber. So the cost of transactions increases linearly with the number of subscribers. However, it is easy for the vendor to remove a subscriber: the vendor simply no longer uses the subscriber's public key in the protocol.

As is often the case, there seems to be an unfortunate tradeoff between cheap transactions and cheap membership maintenance. UST has cheap transactions, but expensive subscriber removal; Schechter et al. [1999] has expensive transactions but cheap subscriber removal. Although it appears to us that the more frequent operations (transactions) should be made cheap, the more interesting observation is that both papers propose protocols that make tradeoffs at the opposite ends of the spectrum. Furthermore, both papers use partitioning to reduce the cost of the expensive operation.

6.3 Anonymity Services

How can customers keep their private information private if communication channels reveal identities? For example, vendors having toll-free numbers can subscribe to services that reveal callers' phone numbers to vendors, thereby obviating any pseudonym the customers may be using. A similar service in the form of caller-id is now available to many private customers. If a communication channel implicitly reveals identities, how can customers' private information be protected?

The solution lies in separating identification from connections. The connection should not reveal information. Identifying information should be carried over the connection. (Of course, vendors and private parties are welcome to close connections that do not immediately provide sufficient identifying information.) As discussed below, depending upon one's environment and threat model on the Internet, several solutions exist. However, understanding the type and degree of protection offered by the above anonymity services is important. Work specifying cryptographic protocols using CSP and analyzing some anonymity protocols using a model checker is found in Schneider and Sidiropoulos [1996]. More recent work formalizing a broad range of anonymity properties can be found in Syverson and Stubblebine [1999].

For e-mail, anonymous remailers can be used to forward mail through a service that promises not to reveal the sender's identity to the recipient. Users' worried about traffic analysis can use Babel [Gülcü, and Tsudik 1996] or other Mixmaster-based [Cottrell 1995] remailers that forward messages through a series of Chaum mixes [Chaum 1981]. Each mix can identify only the previous and next mix, and never (both) the sender and recipient.

For Web browsing, the Anonymizer [1997] provides a degree of protection. Web connections made through the Anonymizer are made anonymous.

By looking at connection information, packet headers, and so on, the destination Web server can only determine that the connection came from (through) the Anonymizer.

LPWA [Gabber et al. 1997; ProxyMate 1999] (formerly known as Janus, now known, commercially, as ProxyMate) is a "proxy server that generates consistent untraceable aliases for you that enable you to browse the Web, register at web sites, and open accounts, and be 'recognized' upon returning to your accounts, all while still *preserving your privacy*." Like the Anonymizer, the LPWA proxy is at a server that is remote from the user application. Hence it is subject to the same trust and vulnerability limitations.

It is possible, however, to shift trusted elements to the user's machine (or to a machine on the boundary between the trusted LAN and the Internet). Shifting trust in this way can improve the security of other privacy services such as the Anonymizer and LPWA, which are currently centralized to provide an intermediary that masks the true source of a connection. If, instead, anonymous connections are used to hide the source address, the other functions of these services may run as a local proxy on the user's desktop. Security is improved because privacy filtering and other services are done on a trusted machine and because communication is resistant to traffic analysis, and there is no central point of failure.

Crowds is one approach to decentralized anonymous Web connections [Reiter and Rubin 1998]. Essentially, it is a distributed and chained anonymizer with encrypted links between crowd members. Web traffic is forwarded to a crowd member, who flips a weighted coin and, depending on the result, forwards it either to some other crowd member or to the destination. This makes communication resistant to local observers. Since repeated transactions from the same crowd member emerges from the crowd at the same point, profile freedom is limited to at best the number of crowd members whose connections emerge from the crowd at that same member.

Onion routing also provides decentralized anonymizing services for a variety of Internet services over connections that are resistant to traffic analysis. Like Babel, onion routing can be used for e-mail. Onion routing can also be used to hide Web transactions, remote logins, and file transfers. If the communicating parties have secure connections to endpoint onion routers, communication can be anonymous to both the network and observers, but the parties may reveal identifying information to each other. The goal of onion routing is anonymous connections, not anonymous communication. The most up-to-date (albeit very terse) published description of onion routing is given in Goldschlag et al. [1999]. Other application-independent systems that complicate traffic analysis in networks have been designed or proposed. In Fasbender et al. [1996], a cryptographically layered structure similar to onions in onion routing is used to forward individual IP packets through a network, essentially building a connection for each packet in a connectionless service. In Pfitzmann et al. [1991],

mixes are used to make an ISDN system that hides the individual within a local switch originating or receiving a call.

7. CONCLUSION

In this paper we presented a protocol for unlinkable serial transactions suitable for a variety of network-based subscription services. It is the first protocol to use cryptographic blinding to enable subscription services. The protocol prevents the service from tracking the behavior of its customers, while protecting the service vendor from abuse due to simultaneous or "cloned" use by a single subscription. We evaluated the security of the basic protocol. The basic protocol is extended to include auditing to further deter subscription sharing. We described other applications of unlinkable serial transactions for pay-per-use transactions within a subscription, third-party subscription management, multivendor coupons, proof of group membership, and voting.

Our approach is based on primitives supporting e-cash, but is designed to function in a credit card commercial infrastructure as well. By manipulating what must be trusted and by whom, as compared with their more common applications, we are also able to simplify the use of such primitives in our protocols. Our approach relies on anonymous communication: there is no sense in using anonymous tokens, pseudonyms, etc., if identities are revealed by the communications channel. Our approach prevents profiling by vendors. However, the use of trusted intermediaries for profiling may be beneficial to both the customer and vendor, e.g., for marketing purposes. It might be complicated to incorporate such trusted intermediaries with the protocols we have presented. But decentralizing may ultimately provide better assurance to customers. For example, profiles can be collected locally at a user's workstation. This lets individuals control their own profiles. Individuals could a contact marketer through an anonymous connection (cf., Section 6.3) and request advertisements suited to their own profiles. Once individuals close the connection, the marketer can no longer contact them.

REFERENCES

1997. Anonymizer. www.anonymizer.com.

BRICKELL, E., GEMMELL, P., AND KRAVITZ, D. 1995. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms* (San Francisco, CA, Jan.) ACM Press, New York, NY, 457–466.

CAMP, L. J., HARKAVEY, M., YEE, B., AND TYGAR, J. D. 1996. Anonymous Atomic Transactions. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce* (Nov.), USENIX Assoc., Berkeley, CA.

CHANDY, K. M. AND MISRA, J. 1988. *Parallel Program Design: A Foundation*. Addison-Wesley Longman Publ. Co., Inc., Reading, MA.

CHAUM, D. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM 24*, 2 (Feb. 1981), 84–88.

CHAUM, D. 1983. Blind signatures for untraceable payments. In *Proceedings of the Conference on Advances in Cryptology* (CRYPTO '82, Santa Barbara, CA), D. Chaum, R. L. Rivest, and A. T. Sherman, Eds. Plenum Press, New York, NY, 199–203.

CHAUM, D. 1985. Security without identification: transaction systems to make big brother obsolete. *Commun. ACM 28*, 10 (Oct. 1985), 1030–1044.

CHAUM, D., FIAT, A., AND NAOR, M. 1990. Untraceable electronic cash. In *Proceedings of the Conference on Advances in Cryptology* (CRYPTO '88, Santa Barbara, CA, USA, Aug. 21–25, 1988), S. Goldwasser, Ed. Springer Lecture Notes in Computer Science Springer-Verlag, New York, NY, 319–327.

CORTTRELL, L. 1994. Mixmaster and remailer attacks. www.obscura.com/loki/remailer-essay.html.

CRANNOR, L. 1996. Electronic voting. *Crossroads 2*, 4 (Apr.).

FASBENDER, A., KESDOGAN, D., AND KUBITZ, O. 1996. Variable and scalable security: Protection of location information in mobile ip. In *Proceedings of the 46th IEEE Conference on Vehicular Technology* (Atlanta, GA, Mar.)

FRANKLIN, M. K. AND REITER, M. K. 1997. Fair exchange with a semi-trusted third party (extended abstract). In *Proceedings of the 4th ACM Conference on Computer and Communications Security* (CCS '97, Zurich, Switzerland, Apr. 1–4, 1997), R. Graveman, P. Janson, C. Neumann, and L. Gong, Eds. ACM Press, New York, NY, 1–5.

FRANKLIN, M. AND YUNG, M. 1992. Towards provably secure efficient electronic cash. Tech. Rep. CUCS-018092. Columbia Univ., New York, NY.

FUJIOKA, A., OKAMOTO, T., AND OHTA, K. 1993. A practical secret voting scheme for large scale elections. In *Proceedings of the Conference on Advances in Cryptology* (CRYPTO '92, Santa Barbara, CA), E. F. Brickell, Ed. Springer-Verlag, New York, 244–251.

GABBER, E., GIBBONS, P., KRISTOL, D., MATIAS, Y., AND MAYER, A. 1999. On secure and pseudonymous client-relationships with multiple servers. *ACM Trans. Inf. Syst. Secur. 2*, 4 (Nov.).

GOLDSCHLAG, D., REED, M., AND SYVERSON, P. 1999. Onion routing for anonymous and private internet connections. *Commun. ACM 42*, 2 (Feb.), 39–41.

GONG, L. AND SYVERSON, P. 1998. Fail-stop protocols: An approach to designing secure protocols. In *Proceedings of the 5th IFIP International Working Conference on Dependable Computing for Critical Applications* (Urbana-Champaign, IL, Sept. 1995), R. K. Iyer, M. Morganti, W. K. Fuchs, and V. Gligor, Eds. IEEE Computer Society Press, Los Alamitos, CA, 79–99.

GULCU, C. AND TSUDIK, G. 1996. Mixing email with Babel. In *Proceedings of the 1996 Internet Society Symposium on Network and Distributed System Security* (San Diego, CA, Feb.), 2–16.

GUNTHER, C. 1987. An identity-based key-exchange protocol. In *Proceedings of the Conference on Advances in Cryptology* (EUROCRYPT '89) Springer-Verlag, New York, 29–37.

HALLER, N. 1994. The s/key one-time password system. In *Proceedings of the ISOC Symposium on Network and Distributed System Security* (San Diego, CA, Feb. 1994).

MENEZES, A. J., VAN OORSCHOT, P. C., AND VANSTONE, S. A. 1997. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL.

OKAMOTO, T. AND OHTA, K. 1992. Universal electronic cash. In *Proceedings of the Conference on Advances in Cryptology* (CRYPTO '91) Springer-Verlag, New York, NY, 324–337.

PFITZMANN, A., PFITZMANN, B., AND WAIDNER, M. 1991. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems* (Feb., Mannheim, Germany) 451–463.

POINTCHEVAL, D. AND STERN, J. 1996. Provably secure blind signature schemes. In *Proceedings of the Conference on Advances in Cryptology* (CRYPTO '96, Santa Barbara, CA), N. Koblitz, Ed. Springer-Verlag, New York, 252–265.

1999. ProxyMate. www.proxymate.com.

REITER, M. K. AND RUBIN, A. D. 1998. Crowds: anonymity for Web transactions. *ACM Trans. Inf. Syst. Secur. 1*, 1, 66–92.

SCHECHTER, S., PARNELL, T., AND HARTEMINK, A. 1999. Anonymous authentication of membership in dynamic groups. In *Proceedings of the Conference on Financial Cryptogra-*

phy (Anguilla, British West Indies, Feb. 99), M. Franklin, Ed. Springer-Verlag, New York, 184–195.

SCHNEIDER, S. AND SIDIROPOULOS, A. 1996. Csp and anonymity. In *Proceedings of the Conference on Computer Security* (ESORICS 96, Rome, Italy), E. Bertino, H. Kurth, G. Martella, and E. Montolivo, Eds. Springer-Verlag, New York, 198–218.

SET, 1999. SET Secure Electronic Transaction LL. www.setco.org

SIMONE, D. 1997. Anonymous communication and anonymous cash. In *Proceedings of the Conference on Advances in Cryptology* (EUROCRYPT '97) Springer-Verlag, New York, 61–73.

SYVERSON, P. AND STUBBLEBINE, S. 1999. Group principals and the formalization of anonymity. In *Proceedings of the Conference on Formal Methods* (Toulouse, France, Sept.), J. Wing, J. Woodcock, and J. Davies, Eds. Springer-Verlag, New York, 814–833.

SYVERSON, P., STUBBLEBINE, S., AND GOLDSCHLAG, D. 1997. Unlinkable serial transactions. In *Proceedings of the Conference on Financial Cryptography* Springer-Verlag, New York, NY, 39–55.

Received: November 1997; revised: April 1999; accepted: April 1999