Anonymous Authentication of Membership in Dynamic Groups

Conference Paper *in* Lecture Notes in Computer Science · February 2001

DOI: 10.1007/3-540-48390-X_14 · Source: CiteSeer

Anonymous Authentication of Membership in Dynamic Groups

Stuart E. Schechter
Harvard
stuart@post.harvard.edu

Todd C. Parnell
MIT
tparnell@mit.edu

Alexander J. Hartemink
MIT
amink@mit.edu

February 25, 1999

Abstract

We present a protocol for authenticating an individual's membership in a group without revealing that individual's identity and without restricting how the membership of the group may be changed. Existing protocols that authenticate membership by identifying individuals do not provide anonymity. Those in which members share a common key require a new key to be distributed whenever an individual leaves the group.

To overcome these limitations we introduce the *verifiably common secret encoding* to construct anonymous authentication protocols. These protocols both authenticate membership without identifying the member and enable a trusted third party to add and remove members of the group instantly, in a single message to the authenticator. Applications in electronic commerce and communication can now provide anonymous authentication while accommodating frequent changes in membership. Because a verifiably common secret encoding grows linearly with the size of the group, we describe techniques for partitioning groups to improve performance.

KEYWORDS: anonymity, authentication, key replacement, verifiably common secret encoding

1 Introduction

We present a protocol for authenticating an individual's membership in a group without revealing that individual's identity and without restricting the frequency with which the membership of the group may be changed.

Authenticating membership in a group is a common task because privileges, such as the right to read a document, are often assigned to many individuals. While permission to exercise a privilege requires that members of the group be distinguished from non-members, members need not be distinguished from one another. Indeed, privacy concerns may dictate that authentication be conducted anonymously.

For instance, subscription services such as *The Wall Street Journal Interactive Edition* [16] require subscribers to identify themselves in order to limit service to those who pay, but many subscribers would prefer to keep their reading habits to themselves. Employee feedback programs, which require authentication to ensure that employees can report only on their satisfaction with their own supervisor, also stand to benefit from enhanced privacy. Adding anonymity protects those employees who return negative feedback from being singled out for retaliation.

Most existing systems that authenticate membership in a group do so by identifying an individual, then verifying that the individual is a member. The requirement that an individual must identify herself to authenticate her membership can be eliminated by distributing a single group identity key to be used by all group members. However, this approach makes supporting dynamic groups unwieldy: whenever an individual is removed from the group, a new group identity key must be distributed to all remaining members. Not until every member receives this key can authentication be performed anonymously.

We achieve anonymous authentication using *verifiably common secret encodings*. This new primitive enables us to extend anonymous authentication to dynamic groups in which a trusted party may add and remove members of the group in a single message to the authenticator. We also enable members to replace their authentication keys if these keys should become compromised. Furthermore, our protocols ensure that even if a key does become compromised, all previous and future transactions remain anonymous and unlinkable.

Section 2 of this paper introduces some notation and conventions. Section 3 presents a set of requirements for anonymous authentication protocols. In Section 4, we define a verifiably common secret encoding and list the operations supported by this primitive. We use these encodings in Section 5 to create an elementary anonymous authentication protocol. In Section 6, we extend this elementary system to provide key replacement. In Section 7, we give a trusted third party the ability to add and remove group members by communicating only with the authenticator. In Section 8, we show how to encode, decode, and verify VCS vectors, an implementation of verifiably common secret encodings. Section 9 describes how to scale anonymous authentication for very large groups. We provide a context for our research by discussing related work in Section 10 and then conclude in Section 11.

2 Conventions

Throughout this paper, we refer to any individual requesting authentication as *Alice*. The authentication process exists to prove to the authenticator, *Bob*, that *Alice* is a member of a group, without revealing the *Alice's* name or any other aspect of her identity. When a trusted third party is needed, we call him *Trent*.

All parties are assumed to have a public-key pair used for identification. We represent public keys using the letter $\mathbf{p}$ and secret (or private) keys using the letter $\mathbf{s}$. For any message m and key $\mathbf{p}$, we define $\{m\}_{\mathbf{p}}$ to represent public-key encryption or the opening of a signature. For any message m and key $\mathbf{s}$, we define $\{m\}_{\mathbf{s}}$ to represent public-key decryption or signing. Symmetric encryption of message m with key k is represented as $E_k[m]$. When necessary, messages to be signed are appended with a known string to differentiate them from random strings. Messages sent by either *Bob* or *Trent* are also assumed to include a timestamp.

The set $\mathbf{P}$ is a set of public keys associated with a group. An individual whose public key is in $\mathbf{P}$ is called a *member* of $\mathbf{P}$. More precisely, a member of $\mathbf{P}$ is an individual possessing a secret key $\mathbf{s}$ corresponding to a public key $\mathbf{p} \in \mathbf{P}$, such that for the set M of messages that may be encoded using $\mathbf{p}$, $\forall m \in M$, $m = \{\{m\}_{\mathbf{p}}\}_{\mathbf{s}}$. To be authenticated anonymously is to reveal only that one is a member of $\mathbf{P}$. This definition of *anonymity* provides privacy only if there are other members of $\mathbf{P}$. We thus assume that the set $\mathbf{P}$ is public knowledge and that one can verify that the public keys in $\mathbf{P}$ are associated with real individuals.

Finally, we assume that all communication takes place over an anonymous communication channel [3, 7, 13, 14]. This prevents an individual's anonymity from being compromised by the channel itself.

3 Requirements for Anonymous Authentication Protocols

The following three requirements are essential to anonymously authenticate membership in $\mathbf{P}$.

> SECURITY: *Only members of* $\mathbf{P}$ *can be authenticated.*
>
> ANONYMITY: *If an individual is authenticated, she reveals only that she is a member of* $\mathbf{P}$*. If she is not authenticated, she reveals nothing.*
>
> UNLINKABILITY: *Separate authentication transactions cannot be shown to have been made by a single individual.*

Note that the above definition of *anonymity* is the broadest possible, since *security* requires that only members of $\mathbf{P}$ can be authenticated.

The authenticator may choose to compromise *security* by authenticating an individual who is not a member of $\mathbf{P}$. Similarly, an individual may choose to forfeit her *anonymity* by revealing her identity. Therefore, we must assume that authenticators act to maintain security and that individuals act to preserve their own anonymity.

The above requirements do not account for the fact that membership in $\mathbf{P}$ is likely to change. Moreover, people are prone to lose their keys or fail to keep them secret. For a system to be able to address these concerns, we add to the list of requirements the following:

> KEY REPLACEMENT: *A member of* $\mathbf{P}$ *may replace her authentication key with a new one and need only confer with the authenticator to do so.*
>
> DYNAMIC GROUP MEMBERSHIP: *A trusted third party may add and remove members of* $\mathbf{P}$ *and need only confer with the authenticator do so.*

To make membership in $\mathbf{P}$ dynamic, a third party is trusted to add and remove members. If this third party is not trustworthy, he can manipulate the set $\mathbf{P}$ to reduce *anonymity*. For instance, if he shrinks $\mathbf{P}$ so that the group contains only one member, that member's identity will be revealed during her next authentication transaction[1].

4 Verifiably Common Secret Encodings

We begin with a set of public keys, $\mathbf{P}$. Recall that we defined a *member* of $\mathbf{P}$ to be an individual possessing a secret key $\mathbf{s}$ corresponding to a public key $\mathbf{p} \in \mathbf{P}$. A *verifiably common secret encoding* e, of a value x, has the following properties:

[1] In the case that a trusted third party cannot be agreed upon, *anonymity* can still be protected by imposing rules governing the ways in which $\mathbf{P}$ can be modified. These rules should be designed to prevent any excessive modification of $\mathbf{P}$ that might compromise *anonymity*. Violations of the rules must be immediately detectable by an individual when she receives changes to $\mathbf{P}$ during authentication.

SECRECY: *Only members of* $\mathbf{P}$ *can decode* e *to learn* x.

COMMONALITY: *Any member of* $\mathbf{P}$ *can decode* e *and will learn the same value* x *that any other member of* $\mathbf{P}$ *would learn by decoding* e.

VERIFIABILITY: *Any member of* $\mathbf{P}$ *can determine whether commonality holds for a given value* e*, regardless of whether* e *is properly constructed.*

We manipulate this primitive using the following three operations:

$$e \leftarrow \text{ENCODE}(x, \mathbf{P})$$
$$x \leftarrow \text{DECODE}(e, \mathbf{s}, \mathbf{P})$$
$$isCommon \leftarrow \text{VERIFY}(e, \mathbf{s}, \mathbf{P})$$

In the next three sections, we use these three functions to build anonymous authentication protocols. In Section 8, we provide a concrete algorithmic implementation for the functions.

5 Anonymous Authentication

We start by presenting a simple anonymous authentication protocol that satisfies the requirements of *security*, *anonymity*, and *unlinkability*. It establishes a session key y between *Alice* and *Bob* if and only if *Alice* is a member of $\mathbf{P}$. The protocol will serve as a foundation for more powerful systems providing *key replacement* and *dynamic group membership* to be described in Sections 6 and 7.

This protocol requires that *Bob* be a member of $\mathbf{P}$. If he is not, both *Alice* and *Bob* add $\mathbf{p}_{bob}$ to $\mathbf{P}$ for the duration of the authentication transaction.

5.1 The Authentication Protocol

Before the authentication transaction in Figure 1 commences, *Alice* randomly selects a session key y. She then encrypts y with *Bob's* public key to form message (1). This message, which represents a request for authentication, may also be augmented to specify the group in which *Alice's* membership is to be authenticated.

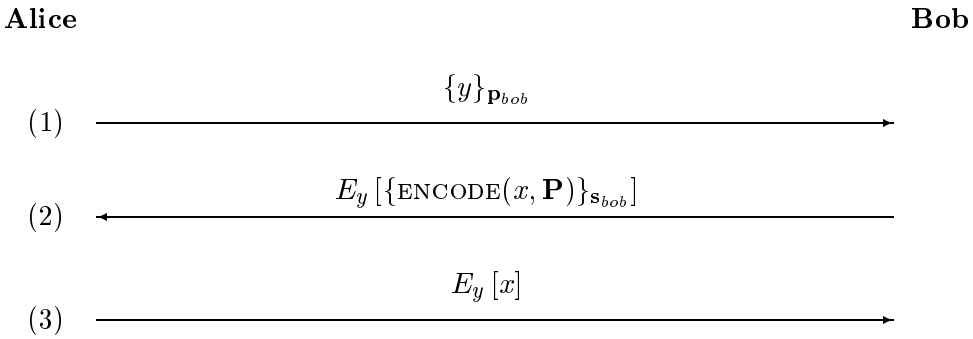


Figure 1: An Elementary Anonymous Authentication Transaction

In response, *Bob* randomly picks x. He creates a message containing a verifiably common secret encoding of x, signs it, and then encrypts with the session key y. He sends this to *Alice* as message (2).

Alice decrypts the message and verifies *Bob's* signature to reveal a value e. If VERIFY$(e, \mathbf{s}_{alice}, \mathbf{P})$ returns true, *Alice* is assured that e is an encoding that satisfies *commonality*. Only then does she use DECODE$(e, \mathbf{s}_{alice}, \mathbf{P})$ to learn x. If VERIFY$(e, \mathbf{s}_{alice}, \mathbf{P})$ returns false, *Alice* cannot be assured that e satisfies *commonality* and halts the transaction.

In message (3), *Alice* proves her membership in $\mathbf{P}$ by encrypting x with the session key y. Upon decrypting message (3) to reveal x, *Bob* concludes that *Alice* is a member of $\mathbf{P}$. Authenticated communication between *Alice* and *Bob* may now begin.

Alice may later wish to prove that it was she who was authenticated in this transaction. We show in Appendix A how *Alice* may request a receipt for this transaction. With such a receipt in hand, *Alice* may, at any point in the future, prove the transaction was hers.

5.2 Satisfying the Requirements

Secrecy ensures that only members of $\mathbf{P}$ can decode e to learn x. *Security* is therefore maintained because an individual is authenticated only when she can prove knowledge of x. By requiring that *Bob* be a member of $\mathbf{P}$ we prevent *Bob* from staging a man in the middle attack in which he uses *Alice* to decode a verifiably common secret encoding that he would not otherwise be able to decode.

Commonality guarantees that any member of $\mathbf{P}$ can decode e and will learn the same value x that any other member would learn by decoding e. If *Alice* is certain that e exhibits *commonality*, it follows that by using x to authenticate her membership, she reveals nothing more than that she is a member of $\mathbf{P}$.

Verifiability is required so that *Alice* may prove for herself that the encoding e exhibits *commonality*, even though she did not create this encoding. Thus, by sending message (3) only when VERIFY() returns true, *Alice* ensures that her authentication will be both anonymous and unlinkable. If *Bob* should be malicious and attempt to construct e in a way that would allow him to discover *Alice's* identity from her decoding of e, verification will fail. *Alice* will halt the transaction before she decodes e. Since message (2) must be signed by *Bob*, *Alice* can use the signed invalid encoding as proof of *Bob's* failure to follow the protocol.

The authentication transaction appears the same regardless of which member of $\mathbf{P}$ was authenticated. As a result, even an otherwise omniscient adversary cannot learn which member of $\mathbf{P}$ was authenticated by inspecting the transaction. Thus, even if *Alice's* key is compromised before authentication, the transaction remains anonymous and unlinkable.

6 Key Replacement

In the protocol above, *Alice* uses a single key pair $(\mathbf{p}, \mathbf{s})$ to represent both her identity and her membership in the group. Because she uses the same key pair for both functions, an adversary who compromises her secret key $\mathbf{s}$ can not only authenticate himself as a member of $\mathbf{P}$, but can also pose as *Alice* in any other protocol that uses $\mathbf{s}$. Ideally, compromising the key used in the authentication

process should not compromise *Alice's* identity. By using two key pairs, one to represent her identity and one for authentication, *Alice* significantly reduces the potential for damage should she lose her authentication key. Using two key pairs for the two separate functions also enables *Alice* to replace a lost authentication key.

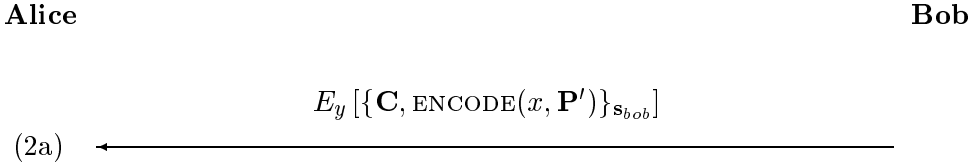
We continue to use the pair $(\mathbf{p}, \mathbf{s})$ to identify an individual. Each member of $\mathbf{P}$ now generates an authentication key pair $(\mathbf{p}', \mathbf{s}')$ for each group in which she is a member. Because of the severe consequences of losing $\mathbf{s}$, we assume that $\mathbf{s}$ is kept well guarded. Because only $\mathbf{s}'$ will be needed during the authentication transaction, we only consider the case where an authentication key $\mathbf{s}'$, not an identity key $\mathbf{s}$, is lost or compromised. When $\mathbf{s}'$ is lost or compromised, the individual can disable the key and obtain a replacement by conferring only with the authenticator.

In order to validate her public authentication key $\mathbf{p}'$, each member uses her secret identity key $\mathbf{s}$ to sign a certificate $c = \{\mathbf{p}'\}_{\mathbf{s}}$. This certificate can be opened to reveal the public authentication key as follows: $\{c\}_{\mathbf{p}} = \{\{\mathbf{p}'\}_{\mathbf{s}}\}_{\mathbf{p}} = \mathbf{p}'$.

To initialize the system, all members of $\mathbf{P}$ send their certificates to *Bob*. *Bob* collects all the certificates to form the set $\mathbf{C}$. The set of public authentication keys, $\mathbf{P}'$, can then be generated by opening each certificate in $\mathbf{C}$: $\mathbf{P}' = \{\{c_i\}_{\mathbf{p}_i} : c_i \in \mathbf{C}\}$.

6.1 Modifications to the Authentication Protocol

The only modification to the authentication protocol is to require *Bob* to add the set of certificates $\mathbf{C}$ to message (2). The augmented message will be labeled (2a):



¿From the set of certificates $\mathbf{C}$ and public identity keys $\mathbf{P}$, *Alice* computes $\mathbf{P}'$ using the technique shown above. She then verifies e using $\text{VERIFY}(e, \mathbf{s}'_{alice}, \mathbf{P}')$. If the encoding exhibits *commonality*, *Alice* learns x from $\text{DECODE}(e, \mathbf{s}'_{alice}, \mathbf{P}')$.

6.2 The Key Replacement Transaction

If *Alice* believes her secret authentication key has been compromised, she simply generates a new authentication key pair, creates a certificate for the new public authentication key, and sends that certificate to *Bob*. *Bob* returns a signed receipt to *Alice* acknowledging the new certificate. Since we assume that *Bob* acts to maintain security, we expect him to use *Alice's* new certificate and authentication key[2].

[2]Even if *Bob* fails to use the new certificate, *Alice* can either proceed using her old key (in the case that it was compromised and not lost) or can use the signed message (2a) as proof of *Bob's* failure to use the new certificate.

7 Dynamic Group Membership

We now describe how a trusted third party, *Trent*, may be given sole responsibility for maintaining the set of certificates $\mathbf{C}$. To this end, *Alice* requires that any $\mathbf{C}$ used by *Bob* be signed by *Trent*. During the authentication transaction, message (2a) is replaced by message (2b):

Alice **Bob**

$$E_y\left[\{\{\mathbf{C}\}_{\mathbf{s}_{trent}}, \text{ENCODE}(x, \mathbf{P}')\}_{\mathbf{s}_{bob}}\right]$$

(2b) ←

If *Alice* is to be granted membership in $\mathbf{P}$, she generates an authentication key pair, creates the certificate c_{alice}, and sends it to *Trent* who updates $\mathbf{C}$ and distributes a signed copy to *Bob*. To remove *Alice* from $\mathbf{P}$, and thereby prevent her from being authenticated, *Trent* simply removes *Alice's* certificate c_{alice} from $\mathbf{C}$ and distributes a signed copy to *Bob*. In both cases, *Bob* and other members of $\mathbf{P}$ can compute the new $\mathbf{P}'$ using $\mathbf{P}$ and the new set of certificates $\mathbf{C}$.

8 Constructing Verifiably Common Secret Encodings

We use public-key cryptography to construct verifiably common secret encodings that we call VCS vectors. Assuming that M_i represents the set of messages that may be encrypted by a public key $\mathbf{p}_i \in \mathbf{P}$, the set of messages that may be encoded as a VCS vector for group $\mathbf{P}$ is $\mathbf{M} = \bigcap M_i$.

A *VCS vector* encodes a value x as follows:

$$\vec{e} \leftarrow [\{x\}_{\mathbf{p}_1}, \{x\}_{\mathbf{p}_2}, \cdots, \{x\}_{\mathbf{p}_n}] \text{ where } n = |\mathbf{P}|$$

Encoding, decoding, and verifying VCS vectors can be performed by the following three functions:

$$\text{ENCODE}(x, \mathbf{P}): \quad \vec{e} \leftarrow \begin{cases} [\{x\}_{\mathbf{p}_1}, \{x\}_{\mathbf{p}_2}, \cdots, \{x\}_{\mathbf{p}_n}] & x \in \mathbf{M} \\ [] & x \notin \mathbf{M} \end{cases}$$

$$\text{DECODE}(\vec{e}, \mathbf{s}_i, \mathbf{P}): \quad x \leftarrow \{\vec{e}[i]\}_{\mathbf{s}_i}$$

$$\text{VERIFY}(\vec{e}, \mathbf{s}_i, \mathbf{P}): \quad isCommon \leftarrow \vec{e} = \text{ENCODE}(\text{DECODE}(\vec{e}, \mathbf{s}_i, \mathbf{P}), \mathbf{P})$$

When using VCS vectors, *secrecy* holds only if x is not revealed when encrypted multiple times with different public keys. This is not true of RSA with small exponents [10] or Rabin[3] [12]. For this reason, caution must be exercised when selecting a public-key encryption technique.

[3] Rabin encryption can be modified to permit secure multiple encryption of x. Given a hash function $h()$ we modify Rabin as follows:

Function	*Rabin*	*Modified Rabin*
Encryption	$E(x) = x^2 \bmod n$	$E(x) = (x + h(n))^2 \bmod n$
Decryption	$D(c) = \sqrt{c} \bmod n$	$D(c) = ((\sqrt{c} \bmod n) - h(n)) \bmod n$

Commonality holds because any secret key corresponding to a key in $\mathbf{P}$ can be used to decode $\vec{e}$ to learn x. Decrypting $\vec{e}[i]$ with $\mathbf{s}_i$ yields the same secret x for all i.

Any member of $\mathbf{P}$ can use DECODE() to learn x from $\vec{e}$ and then re-encode x using ENCODE() to obtain a valid encoding of x. Because ENCODE() generates a valid encoding, *commonality* will hold for this re-encoded vector. If the re-encoded vector equals the original vector $\vec{e}$, then $\vec{e}$ must also satisfy *commonality*. Hence, as long as ENCODE() is deterministic[4], we can verify the commonality of any encoding $\vec{e}$. Consequently, *verifiability* is satisfied.

That the VERIFY() operation can be expressed as a simple composition of the ENCODE() and DECODE() operations is a general statement, independent of how we construct our verifiably common secret encodings. For this reason, if we can construct ENCODE() and DECODE() operations for which *commonality* holds, *verifiability* becomes automatic. Thus, we can replace our implementation-specific definition of VERIFY() with a general definition:

$$\text{VERIFY}(e, \mathbf{s}, \mathbf{P}): \qquad isCommon \leftarrow e = \text{ENCODE}(\text{DECODE}(e, \mathbf{s}, \mathbf{P}), \mathbf{P})$$

9 Making Anonymous Authentication Scalable

The number of entries in a VCS vector grows linearly with the number of members of $\mathbf{P}$, as does the time required to generate, transmit, and verify the entries. The same is true of any verifiably common secret encoding[5]. This growth could make anonymous authentication impractical for very large dynamic groups.

We can address this issue by authenticating using subsets of $\mathbf{P}$. Individuals will now remain anonymous and unlinkable only among the members of their subset rather than among all members of $\mathbf{P}$. Because membership in a subset of $\mathbf{P}$ implies membership in $\mathbf{P}$, *security* is not affected. We propose two ways of assigning subsets: random generation of single-use subsets during each authentication transaction and the use of a static assignment algorithm.

9.1 Single-Use Subsets

During each authentication transaction, *Alice* selects a subset of $\mathbf{P}$ at random. To ensure her membership, *Alice* augments the subset to include herself. She sends this subset to *Bob* when requesting authentication. *Alice* and *Bob* then use this subset in place of $\mathbf{P}$ for the remainder of the protocol.

Alice picks her subset of $\mathbf{P}$ at the time she initiates the authentication transaction. If she has limited long-term storage, she can select the subset by picking keys in $\mathbf{P}$ by their indices. She then requests keys in $\mathbf{P}$ from *Bob* by index at the start of the authentication transaction. To prevent *Bob* from sending fraudulent identity keys, *Alice* maintains a hash tree of the keys or their fingerprints.

[4] Probabilistic encryption [9, 1] may still be used. We simply make the ENCODE() function deterministic by using its first input parameter, the secret x, to seed the pseudo-random number generator.

[5] The size of a verifiably common secret encoding must be linear in the size of $\mathbf{P}$. Suppose the size of an encoding e is sublinear ($|e| = o(n)$). Since $\mathbf{P}$ has 2^n possible subsets, $\exists P_1, P_2 \subset \mathbf{P}$ such that $P_1 \neq P_2$ and $\text{ENCODE}(x, P_1) = \text{ENCODE}(x, P_2) = e$. Without loss of generality, there is a member m of P_1 who is not a member of P_2. *Commonality* for P_1 tells us that m can decode e, but *secrecy* for P_2 tells us that m cannot decode e. This is a contradiction.

Alice must be cautious when using single-use subsets. If external circumstances link two or more transactions, *Alice* is anonymous only among the intersection of the subsets used for authentication.

9.2 Statically Assigned Subsets

Subsets may also be assigned by a static algorithm such that each member of $\mathbf{P}$ is always assigned to the same subset $\mathbf{P}_i \subseteq \mathbf{P}$ where $\bigcup \mathbf{P}_i = \mathbf{P}$. These subsets may change only when members are added or removed from $\mathbf{P}$. As above, *Alice* uses $\mathbf{P}_i$ wherever she previously would have used $\mathbf{P}$.

Even if *Trent* picks the subsets, he may do so in a way that unwittingly weakens anonymity or unlinkability. Using a one-way hash function, preferably generated randomly before the membership is known, ensures that no party can manipulate the assignment of individuals to subsets.

10 Related Work

Anonymity is an essential feature of digital cash schemes [5]. The requirements of these schemes differ markedly, however, from those of anonymous authentication systems. In particular, digital cash schemes do not allow for revocation of anonymous cash after it has been issued.

In addressing anonymity in transactions, Chaum [4] assumes that institutions collect information about individuals who use those institutions' systems. He therefore proposes that individuals use different pseudonyms when conducting transactions with different institutions to prevent those institutions from sharing information and linking user profiles together. This fails to protect those whose right to use a system comes from a pre-existing relationship in which their identity is already known. Moreover, Chaum's approach does not provide unlinkability, leaving open the possibility an individual might reveal her identity through behaviors that can be profiled.

Syverson, et al. [15] introduce a protocol for unlinkable serial transactions using Chaum's notion of blinding [5]. The protocol is designed for commercial pay-per-use services and relies upon the possibility that any particular service request may be forcibly audited. An audit requires the individual to reveal her identity or risk losing future service. After passing an audit, the individual must make another request before receiving the service originally requested. If requests are infrequent, she may have to wait a significant amount of time before making the second request lest the two requests become linked. This system does not provide adequate anonymity if the timing of any request indicates its nature, as audits can be made at any time. The system also cannot guarantee that a revoked individual does not receive service, as that individual may still make a request that is not audited.

Group signatures schemes [2, 6] give an individual the ability to anonymously sign messages on behalf of a group. Kilian and Petrank [11] exploit these signatures to create a scheme for identity escrow. Identity escrow provides anonymous authentication, though an individual's anonymity can be revoked by a trusted third party. While individuals may be added to the signature groups, no provision is made for removing members from these groups. Thus, group signatures in their current form are not a sufficient primitive for anonymously authenticating membership in dynamic groups.

11 Conclusion

In this paper we have shown, for the first time, that it is possible to anonymously authenticate membership in dynamic groups. We have also shown how to replace keys in these authentication systems. In order to make these advances possible, we introduced a new primitive: the verifiably common secret encoding. We presented VCS vectors as an example of how verifiably common secret encodings can be constructed. Because the size of a verifiably common secret encoding grows linearly with the size of the group **P**, we described how to authenticate membership using subsets of **P**.

Extending anonymous authentication to dynamic groups is a significant advance because it makes practical a new domain of services in the field of electronic commerce and communication. The addition of replaceable authentication keys further extends the set of applications that may now authenticate membership without identifying the member.

12 Acknowledgements

We would first like to thank the anonymous reviewers who provided invaluable suggestions and references. We are also indebted to Michael Bender, Yanzong Ding, Nailah Robinson, Jenn Vinson, and especially David Savitt for feedback on earlier drafts of this paper. Michael D. Smith helped us shape the presentation so as to maximize its accessibility. Michael Rabin's lively discussions served as excellent sanity checks. Ron Rivest, Silvio Micali, and David Gifford provided inspiration for our work.

References

[1] M. Blum and S. Goldwasser, "An Efficient Probabilistic Public-Key Encryption Scheme which Hides All Partial Information," *Advances of Cryptology — CRYPTO '84 Proceedings*, Springer-Verlag, pp. 289–299.

[2] J. Camenisch and M. Stadler, "Efficient Group Signature Schemes for Large Groups," *Advances in Cryptology — CRYPTO '97 Proceedings*, Springer-Verlag, v. 1294, pp. 410-424

[3] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Communications of the ACM*, v. 24, n. 2, Feb 1981, pp. 84–88.

[4] D. Chaum, "Security without Identification: Card Computers to make Big Brother Obsolete," *Communications of the ACM*, v. 28, n. 10, Oct 1985, pp. 1030–1044.

[5] D. Chaum, A. Fiat, and M. Naor, "Untraceable Electronic Cash," *Advances in Cryptology — CRYPTO '88 Proceedings*, Springer-Verlag, pp. 319–327.

[6] D. Chaum and E. van Heyst, "Group signatures," *Advances in Cryptology — EUROCRYPT '91 Proceedings*, Springer-Verlag, pp. 257-265.

[7] Anonymizer, Inc., *http://www.anonymizer.com.*

[8] D. Cooper and K. Birman, "Preserving Privacy in a Network of Mobile Computers," *1995 IEEE Symposium on Security and Privacy.*

[9] S. Goldwasser and S. Micali, "Probabilistic Encryption," *Journal of Computer and Systems Sciences*, v. 28 n. 2, Apr 1984, pp. 270–299.

[10] J. Hastad and A. Shamir, "On Using RSA with Low Exponent in a Public Key Network," *Advances in Cryptology — CRYPTO '85 Proceedings*, Springer-Verlag, pp. 403–408.

[11] J. Kilian and E. Petrank, "Identity Escrow," *Advances in Cryptology — CRYPTO '98 Proceedings*, Springer-Verlag, pp. 167–185.

[12] M. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization," *Technical Report MIT/LCS/TR-212*, MIT, 1979.

[13] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous Connections and Onion Routing," *IEEE Journal on Selected Areas in Communication Special Issue on Copyright and Privacy Protection*, 1998.

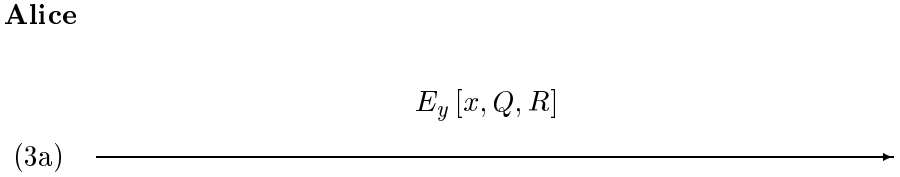
[14] M. Reiter and A. Rubin, "Crowds: Anonymity for Web Transactions" *DIMACS Technical Report 97-15*, Apr 1997.

[15] P. Syverson, S. Stubblebine, and D. Goldschlag. "Unlinkable Serial Transactions," *Financial Cryptography '97*, Feb 1997.

[16] The Wall Street Journal Online, *http://www.wsj.com.*

A Obtaining Proof of Authentication

Alice may obtain a receipt from *Bob* proving that she was authenticated at time t. To obtain such a receipt, *Alice* chooses a random z and uses a one-way hash function h to generate $Q \leftarrow h\left(\{z\}_{\mathbf{s}_{alice}}\right)$ and $R \leftarrow h(z)$. *Alice* includes Q and R in message (3a):



Bob can issue a receipt when he authenticates *Alice*. The receipt he sends is:

$$\{\text{“}Q \text{ and } R \text{ reveal whom I authenticated at time } t\text{”}\}_{\mathbf{s}_{bob}}$$

If she chooses, *Alice* can at any later time prove she was authenticated by *Bob* by revealing the receipt and the value $\{z\}_{\mathbf{s}_{alice}}$. Anyone can verify the receipt by checking that $Q = h(\{z\}_{\mathbf{s}_{alice}})$ and $R = h\left(\{\{z\}_{\mathbf{s}_{alice}}\}_{\mathbf{p}_{alice}}\right)$.