Secret Sets and Applications

R. Molva*　　G. Tsudik†

August 14, 1997

Abstract

This paper introduces the notion of a *secret set* – a basic construct for communication with groups of mutually suspicious entities. A set is secret if any entity can test its membership in the set but can determine neither the other set members nor the cardinality of the set. A number of possible secret set constructions are presented, analyzed and contrasted according to criteria such as: security (strength) as well as bandwidth and processing overheads. Example applications of secret sets are discussed.

Keywords: secret sets, secret multicast, location secrecy, mobile networks.

1 Introduction

Under certain conditions, electronic communication must be conducted in a manner that preserves the secrecy of either (or both) senders or receivers. While this is particularly applicable to military environments, similar requirements can also arise in civilian communications. For example, since mobile (especially, cellular) networks are often vulnerable to hostile eavesdropping, there is a need to provide location secrecy for network users.

In this paper we introduce a basic mechanism for building a secret set of identifiers that addresses the problem of receiver secrecy in computer communication. We first devise and discuss several solutions for constructing secret sets and then demonstrate two applications of the secret sets: secret multicasting and location secrecy in mobile networks.

The crux of the problem considered in this paper is how to construct a *secret set.* We call a set *secret* if and only if the following conditions hold:

C1 Any party (whether a set member or not) can test its membership in the set.

C2 No one, with the exception of the originator of the set, can test another party's membership in the set.

C3 No one, with the exception of the originator of the set, can determine with certainty the number of members in the set. (However, the upper bound on the number of members may be known.)

Our design goals are twofold: 1) **Security:** adherence to secret set conditions C1-C3, and 2) **Efficiency:** minimization of both time and space complexity for secret set construction and membership testing.

However, certain issues are considered to be **out of scope** of this paper, i.e., they constitute explicit non-goals:

- Data secrecy/privacy
  It is assumed that, if needed, bulk data privacy is provided by a parallel, independent mechanism. We note that, in some environments, data privacy might not be important or desired at all.

*Institut Eurécom, Sophia Antipolis, FRANCE, *molva@eurecom.fr*
†USC Information Sciences Institute, Marina del Rey, CA, USA, *gts@isi.edu*

- Data integrity, origin authentication and non-repudiation of all kinds
  Similar to data privacy, these services are considered to be independent of (parallel to) the issue of secret set construction. General-purpose security tools can be used to obtain all of these services.
- Denial of Service (DoS) Attacks
  Our initial model of an adversary is relatively weak (see below) and does not include DoS attacks.

The rest of this paper is organized as follows. We begin with some preliminaries in the next section. Sections 3–5 discuss several approaches to secret set construction; they are summarized in Section 6. The paper concludes with some application examples (Sections 7 and 7.2) and directions for future work (Section 8.)

2 Preliminaries

In the present context, an adversary is an entity whose goal is to attack the system in one the following ways:

A1 Determine whether a certain party is a member of a secret set (without collusion with or, subversion of, that party but with potential collusion with any number of other secret set members.)

A2 Determine the number of parties included in a secret set. We will also consider the special case of all set members colluding (without knowing that they constitute the entire set) in order to determine whether they constitute the entire set or a proper subset thereof.

The following notation is used throughout the rest of this paper:

- Let $\mathcal{U} = \{ST_1, ..., ST_n\}$ be the "universe", i.e., the publicly-known set of all potential secret set members. (We use a special symbol ST_{src} to denote the originator of the secret set.)
- For any subset $\mathcal{S} = \{ST_{i_1}, ..., ST_{i_m}\} \subset \mathcal{U}$, let $IND(\mathcal{S}) = \{i_1, ..., i_m\}$ denote a set of indices of set members (each ranging in $[1, n]$)
- For each $ST_j \in \mathcal{U}$ ($j \in [1, n]$), let $[PK_j, SK_j]$ denote the public and secret key components (respectively) of ST_j's public key-pair.
- $PK_j(data)$ and $SK_j(data)$ denote encryption of *data* under the public and secret public key components, respectively. Throughout the paper, we assume that encryption is always performed in tandem with data integrity, i.e., encrypted data is *non-malleable*. Mechanisms for doing this are described in [2].

3 A Trivial Solution

The most straightforward representation of an m-member secret set $\mathcal{S}$ is:[1]

$$PK_{i_1}(txt_{i_1}), ..., PK_{i_m}(txt_{i_m})$$

where txt_{i_j} (for $j \in [1, m]$) denotes some unambiguous indication that ST_{i_j} is a member of $\mathcal{S}$. To protect against verifiable-plaintext attacks, we assume that each encryption operation is randomized (salted) and integrity-protected as described in [2].[2]

The present method has two important drawbacks:

- The efficiency is poor since a member has no way of determining which encrypted block corresponds to it.
- Attack A2 is possible because encryption blocks are listed sequentially. This makes it trivial for anyone to determine the cardinality of $\mathcal{S}$.)

[1] Encryption with integrity is assumed. See [2].
[2] If encryption is not randomized and txt_{i_j} is predictable, the adversary could guess txt_{i_j} and verify it by encrypting it with PK_{i_j}.

The first issue can be helped if the members of $\mathcal{U}$ were somehow numbered or ordered. However, even if all members of $\mathcal{U}$ form an ordered set, and, moreover, all members of $\mathcal{S}$ are ordered, a member $ST_j \in \mathcal{S}$ has to try decrypting at least one and, at most $min(m, j, (n-j+1))$ blocks before finding $PK_{i_p}(txt_{i_p})$ where $i_p = j$.

On the other hand, if $ST_j \notin \mathcal{S}$ it has to always perform $min(m, j, (n-j+1))$ decryptions just to discover that it is not in the set.

(In either case, ST_j has to make a choice of starting from the first encrypted block or from the last one and working backwards. This decision depends on the value of $min(m, j, (n-j+1))$. In case $min(m, j, (n-j+1)) = m$, ST_j can start with either the first or the last block. If $min(m, j, (n-j+1)) = j$, ST_j starts from the first block, otherwise it starts from the last.)

A2-type attacks can be made harder if ST_{src} introduces *noise* or *decoys* into the representation of $\mathcal{S}$. This can be done with random "garbage" blocks sprinkled amongst real encrypted blocks or with actual *bona fide* encryption blocks where txt_{i_j} is amended to take on a binary meaning, i.e., it can be used to indicate non-membership as well as membership in $\mathcal{S}$. Once again, we note that because our encryption is done with integrity, a "garbage" block can not be misinterpreted as a *bona fide* one. At the same time, a "garbage" block and an encrypted block are indistinguishable.

An unpleasant side-effect of using decoys and noise is the corresponding increase in processing costs. Every noise or decoy block must be processed (via attempted decryption) by a multitude of members.

3.1 Variation on the Theme

In order to address the drawbacks discussed above, we now consider a straight-forward extension. It involves representing any secret set S as a sequence of n blocks, regardless of the real number of members in $\mathcal{S}$. The length of a block is determined by the (globally set) public key encryption block size, e.g., modulus length in case of RSA [5]. If $ST_j \in \mathcal{S}$, the j-th block is $PK_j(txt_j)$, as before. Otherwise, the j-th block is a sequence of random bits the length of the standard encryption block. This has two attractive features:

1. Increased efficiency: since blocks are ordered and there are as many blocks as members, only one decryption attempt needs to be made by each member.
2. A2 resistance: representations of all secret sets are of the same length – n blocks. Hence, it is impossible to determine $|\mathcal{S}|$, the cardinality of the secret set.

Despite the above, this method is impractical chiefly because of the communication overhead. (Consider, for example, the population size of 1024 and the use of RSA encryption with 1024-bit modulus. Irrespective of the cardinality of $\mathcal{S}$, a staggering 1 Mbit would be needed to represent $\mathcal{S}$!)

3.2 Using Pairwise Shared Keys

We now make an assumption that ST_{src} shares a conventional, secret key with every potential member, i.e., the system either operates on conventional cryptography alone or uses Diffie-Hellman key agreement method to compute pairwise keys. (Let K_j^{src} denote a secret key shared between ST_{src} and ST_j.)

The secret set $\mathcal{S}$ can now be constructed as:

$$K_{i_1}^{src}(txt_{i_1}), ..., K_{i_m}^{src}(txt_{i_m})$$

This is very similar to the public key-based version. The only important difference is that there is no requirement for each encryption operation to be randomized. Since all pairwise keys are secret, verifiable-plaintext attacks (of the sort mentioned in Section 3) do not apply.

For example, txt_{i_j} can be set to R, a random quantity generated by ST_{src}. In this case, R must be also included in cleartext as part of $\mathcal{S}$' representation; known-plaintext attacks notwithstanding.

Finally, since encryption block size for a typical conventional cryptosystem is smaller than that for a typical public key cryptosystem, spatial complexity can be reduced. For example, an eightfold decrease in length can be obtained if IDEA [4] (128-bit keys) is used instead of RSA [5] with 1024-bit moduli.

4 CRT Construction

It is evident from the preceding discussion that the simple-minded solutions are inefficient owing to both computational and spatial complexity. (This is besides lacking any notion of elegance.)

In order to remedy the situation, we now base the construction of $\mathcal{S}$ on the well-known Chinese Remainder Theorem (CRT).

In brief, *Chinese Remainder Theorem (CRT)* states that:

> Given a sequence of primes: $p_1, p_2, ..., p_n$ and n values:
>
> $$y_i = x_i \ mod \ p_i \ \ 0 < i \leq \ n$$
>
> there exist (and can be computed) a single value X such that $0 < X \leq \prod_{i=1}^{n} p_i$. Using n prime moduli $p_1, ..., p_n$, each x_i can be subsequently recovered from X by computing $x_i = X \ mod \ p_i$

A secret set is constructed as follows:

Step S0: We require each member ST_q to have a unique public prime p_q. (This is in addition to a public key-pair $[PK_q, SK_q]$ that ST_q is already assumed to possess.)

Step S1: As in Section 3, for each $ST_{i_j} \in \mathcal{S}$ $(0 < j \leq \ m)$ ST_{src} generates a random quantity R_j and computes $x_j = PK_{i_j}(R_j)$.[3]

Step S2: Given all x_j values and the corresponding primes p_{i_j}, ST_{src} computes (in $m-1$ steps) the value X that satisfies the CRT. X itself becomes the representation of the secret set $\mathcal{S}$.

A receiver of a secret set processes it as follows:

Step R1: In order to find out whether it is part of the secret set ST_q first computes $\hat{x_q} = X \ mod \ p_q$

Step R2: Next, ST_q decrypts $\hat{x_q}$ with SK_q. If the decryption (specifically, the integrity check) then $ST_q \notin \ \mathcal{S}$.

4.1 Evaluation

It is evident that, with respect to A1-type attacks, the security of the CRT-based secret set construction is equivalent to that of the trivial PK-based approaches of Sections 3 and 3.1.

A2-type attacks are more subtle. According to CRT, X is bounded by $\prod_{i=1}^{n}(p_i)$ which can allow an adversary to guess the number of members in $\mathcal{S}$. However, the situation is different from the construction in Section 3 where encrypted blocks are listed sequentially and $|\mathcal{S}|$ is evident to any observer. The CRT construction does not allow a deterministic and verifiable guess of $|\mathcal{S}|$; only its upper and lower bounds.

As far as bandwidth overhead, in the worst case the present construction is similar to that in Section 3 (once again, because $X < \prod_{i=1}^{n}(p_i)$.) The processing overhead, however, is greatly reduced. Recall that, in PK-based construction, a potential member has to attempt decryption of a number of blocks before either finding the correct block or giving up after processing all (or most, depending on the numbering/ordering scheme) blocks. CRT construction allows for uniform and efficient processing of $\mathcal{S}$ by all members (whether members of $\mathcal{S}$ or not):

> Every ST_q performs the same long integer division operation to obtain $(X \ mod \ p_q)$ and attempts to decrypt the result.

The only time-consuming task in CRT construction is the process of set construction by ST_{src}. In addition to m public key encryption operations to compute all x_j values, ST_{src} needs to go through $m-1$ invocations of the Euclidean algorithm to compute X.

4.2 Decoys

One way to make the CRT construction less susceptible to A2-type attacks, is by introducing *decoys*. A *decoy* is essentially a random value factored into the computation of X. Suppose that we pick a particular $ST_q \notin \mathcal{S}$ to be a

[3] Once again, we require encryption with integrity as described in [2].

decoy. The originator, ST_{src}, picks a random value $R_q < p_q$ and sets $x_q = R_q$. Then, X is computed to include x_q so that, in the end, $X \ mod \ p_q = x_q$.

Of course, when ST_q receives X and obtains x_q, the decryption will fail. In fact, ST_q can not tell the difference between being a decoy and not being considered at all. Any number of decoys can be factored into X in the same manner. The result is that the A2 attack can be made more difficult by obscuring $\mathcal{S}$.

It should be noted that the use of decoys comes at the expense of not only additional computation steps but also the increased length of X, i.e., additional bandwidth.

4.3 Comparison with Related Work

The CRT construction discussed above is very similar to that used by Chiou and Chen in their *Secure Lock* construct [3]. The main difference is in the goals: [3] concentrated on **secure broadcasting** which entails encrypting a message under a one-time key and then encrypting the key individually for each group member. We, on the other hand, consider a somewhat different problem; secret set construction requires only the addressing information to be kept secret. This lends itself to simpler constructs.

In addition, the *Secure Lock* construct does not take into account condition C2 as defined in Section 1. In other words, it allows any set member to test any other entity's membership in the set. This is possible because each set member's "share" is defined as: $x_j = PK_{i_j}(\hat{d})$ where $\hat{d}$ is the secret key that can be used to decrypt the data message. All PK_{i_j} values are publicly-known and $\hat{d}$ is the same for all set members. A set member ST_q first obtains $\hat{d}$ by extracting its share and decrypting it. Then, ST_q can test whether another entity ST_r is in the set by computing $PK_r(\hat{d})$ and comparing it to $X mod p_r$.

5 Bit Vectors

Having examined and discussed the issues surrounding secret set construction we can observe that very little information needs to be communicated to each potential member of the secret set. Conceptually, this information can be thought of as the value of a binary function $MEMBER(ST_q, \mathcal{S})$ which can be captured in a single bit. This can be interpreted to mean that the total amount of information needed to represent a secret set equals the cardinality of that set, i.e., the number of members therein. However, the difficulty in actually reducing the representation of a secret set $\mathcal{S}$ to $m = |\mathcal{S}|$ bits is the need to label them somehow. It is easy to see that simply generating a bit vector of length m ($m < n = |\mathcal{U}|$) will result in confusion since a potential member has no means to determine the correct bit position (i.e., the bit it should process.)

Based on the above, we can conjecture that the minimum length for a bit vector representation of a secret set is the number of all possible members – the cardinality of $\mathcal{U}$.

The only remaining question is exactly how to specify the function $MEMBER(ST_q, \mathcal{S})$ so as to resist A1- and A2-type attacks.

One possibility is to use Diffie-Hellman key agreement as follows:

> **Assumption:** Each receiver $ST_q \in \mathcal{U}$ pre-distributes its Diffie-Hellman public exponent: $g^{a_q} mod P$ (where g is the base, a_q is ST_q's private key and P is a large prime, $P - 1$ being the group order.)
>
> 1. ST_{src} generates a random quantity b and computes $g^b(modP)$.
> 2. $\forall\ ST_q \in \mathcal{U}$, ST_{src} computes $K_q^{src} = g^{ba_q}$.
> 3. Set the q-th bit of the bit vector to:
>
> $$MEMBER(ST_q, \mathcal{S}) = \begin{cases} MSB(K_q^{src}) & \text{if } ST_q \in \mathcal{S} \\ MSB(K_q^{src}) + 1 \ (mod 2) & \text{otherwise} \end{cases}$$
>
> where $MSB(y)$ denotes the leftmost (most significant) bit of y.
>
> The length of the resulting bit vector is n; however, the total length of the secret set is $(n + log_2\ P)$. The difference is in the size of the one-time residue $g^b(modP)$ that needs to accompany every bit vector.

This method has an important advantage in that its security is based directly on the well-established Diffie-Hellman key agreement protocol. This is further reinforced by a recent result by Boneh and Venkatesan [9] that evaluates the hardness of (the adversary) computing the leftmost 32 bits of a Diffie-Hellman key.

Another benefit is that, even if all secret set members collude (without knowing that they constitute the entire set) they are unable to determine the cardinality of the set.

As an aside, there are alternative ways to construct a bit vector. Assuming pairwise shared keys (as in Section 3.2) the individual bits can be computed as:

$$MEMBER(ST_q, \mathcal{S}) = \begin{cases} MSB(K_q^{src}(b)) & \text{if } ST_q \in \mathcal{S} \\ MSB(K_q^{src}(b)) + 1 (mod 2) & \text{otherwise} \end{cases}$$

where b and MSB are as before.

6 Summary

Of all secret set constructs presented above only the CRT-based and the bit-vector are *interesting*. The bit vector construct is advantageous chiefly because its space complexity is minimal (in most cases; see Section 8.) On the other hand, its main drawback is the need to perform a costly exponentiation for each entity; whether secret set member or not.

In contrast, the CRT construct requires much more space (not always; see Section 8.) but offers some important operational advantages. Most importantly, the number of encryption operations is proportional to the cardinality of the secret set. Furthermore, since a full encrypted block is conveyed to each member of the secret set, the integrity of the membership information can be assured on an individual basis; for example, by signing the contents of the block before encryption and enclosing the signature within.[4] This does not, however, protect the integrity of the set as a whole.

7 Application Examples

7.1 Secret Multicast

One fitting application for secret sets is *secret multicast*, a special case of multicast that maintains the secrecy of the set of receivers. The most basic setting where secret multicast may be applied is a broadcast local area network (LAN) such as an Ethernet. (See Figure 1.) We assume that there are n stations connected to the LAN; equivalently, the superset $\mathcal{U}$ is composed of at most n potential receivers. The problem at hand is how to multicast a message to m ($m \leq n$) receivers while satisfying the aforementioned conditions C1-C3.

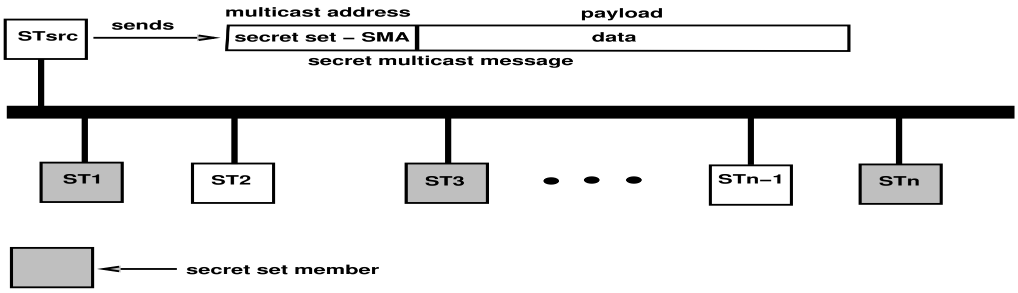

Figure 1: Secret Multicast on a LAN

We note that a simpler problem of unicasting to a secret receiver (i.e., multicasting to a secret set of size one) on a broadcast LAN has been solved by Pfitzmann and Waidner [6]. Basically, the solution involves encrypting

[4]The integrity of the secret set information can also be protected in the bit vector construct by signing the entire bit vector.

the message or portion thereof (including, at least, the sender and receiver addresses) under the public key of the intended receiver. While all stations receive the message, only the intended receiver can successfully decrypt it. (The solution presented in section 3 above can be considered a trivial extension of secret unicasting).

Secret multicasting, as defined in this paper, is also very similar to the well-known problem of *broadcast encryption* [10], [8]. It has been the subject of more than a few publication in the cryptographic community. However, one of the central goals of broadcast encryption is the secrecy (and, optionally, integrity) of the messages, i.e., bulk data. Secret multicasting is a *narrower* problem which is only concerned with the secrecy of the receivers' identities. This important difference allows us to employ the secret set mechanism that is much simpler than full-blown encryption.

A secret multicast scheme can be obtained by a straightforward assignment of secret set roles depicted in section 2 to the principal entities of a multicast communication. In the secret multicast scheme thus obtained:

- $\mathcal{U} = \{ST_1...ST_n\}$ denotes all possible receivers;
- ST_{src} denotes the sender of the secret multicast;
- $\mathcal{S}$ denotes the receivers of the secret multicast or the secret multicast group.
- $\mathcal{SMA}$, Secret Multicast Address, is the representation of the secret set obtained through one of the aforementioned secret set computation techniques.

$\mathcal{SMA}$ is appended to the multicast message and allows the receivers to find out whether they are part of the secret multicast group.

7.2 Location Privacy in Mobile Networks

Another application of secret sets is as a solution to the problem of location privacy in a mobile network. This essentially amounts to preventing an observer from tracking the whereabouts of a mobile user.

In a typical implementation of a mobile network, each mobile user is registered with a home domain and each domain maintains a Home Domain Agent (HDA). HDA's task is to keep a record of the current location of each mobile user registered in the domain. When user A calls another mobile user B, the call from A first reaches the HDA of B's home domain. Then, the call from A reaches B based on the location information provided by the HDA. In order for the HDA to keep accurate information on B's location, B has to keep HDA informed of all location changes.

This mobility management technique inherently requires a central entity that maintains timely location records for each constituent user. Its utility as a network mechanism notwithstanding, it constitutes a potential threat to privacy. A complete history of each user's mobile behavior and network utilization can be drawn from the location records kept by the HDA. Furthermore, this kind of exposure is an inherent weakness of all existing mobile network architectures (e.g., GSM, CDPD, DECT, Mobile IP) since they all require a central entity similar to the HDA to keep record of the mobile user's location.

We note that a related, albeit more general, anonymity problem has been already addressed in [1, 7]. These solutions involve hiding the user's true identity from intruders and from remote entities. Since the underlying mobile networks require an HDA for mobility management, these solutions suffer from the location privacy problem and can not hide the mobile user's location from the HDA.

The second application of the secret set mechanism allows for user mobility management based on central HDA's like in the current mobile network architectures while preserving the privacy of the user's whereabouts. In the suggested scheme, the HDA keeps a timely record of some secret routing information instead of the location record. Like the location record, the secret routing information is used by a calling party to reach the mobile user. The main difference between the secret routing information and the location record of the existing architectures is that the secret routing information does not give away any information on the user's location.

For each new location visited by the mobile user, the secret routing information is computed using the secret set construct, where the roles of the entities are instantiated as follows:

- $\mathcal{U} = \{ST_1...ST_n\}$ denotes the set of all possible routing nodes;

- ST_{src} denotes the originator of the secret routing information; this can either be the mobile user itself or a trusted routing node that is closest to the new location;
- $\mathcal{S} = \{ST_{i_1}, ..., ST_{i_m}\} \subset \mathcal{U}$, denotes the secret routing information;
- $\mathcal{SR}$ is the representation of the secret routing information obtained through one of the aforementioned secret set techniques.

For each mobile user B and each new location of B, a new value of $\mathcal{SR}$ is computed and placed in the HDA along with B's identification. Whenever another user A wants to communicate with B, A first contacts B's HDA. A's message is then prepended with the current value of $\mathcal{SR}$ and the resulting message is broadcasted to all routing nodes adjacent to HDA. Using the secret set mechanism, each routing node decides whether it is a member of the secret set. If a node determines that it is a member, it forwards the message on to all of its neighbors. Conversely, non-members do not forward.

The transmission of the message from HDA to B initially resembles a flooding mechanism but, soon thereafter, the flooding effect is reduced by having only the nodes on the actual route (the members of the secret set) forwarding the message. In order to create confusion and prevent the entire actual route from being discovered, some nodes that are not part of the actual route can be introduced to create so-called decoy routes. These generate message threads that vanish after a few hops, whereas only the thread corresponding to the actual route eventually reaches B. The example in Figure 2 illustrates a secret routing scenario where the actual route is: $\mathcal{SR} = \{ST_2, ST_4, ST_5, ST_7, ST_3\}$, and ST_3, ST_7 are decoys.

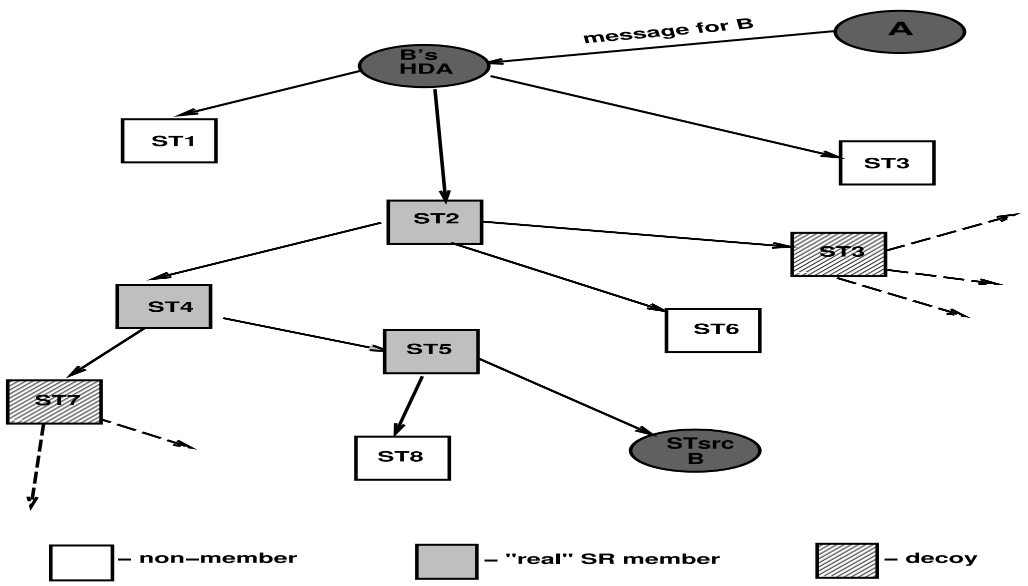

Figure 2: Location secrecy with secret sets

Variations on the basic secret routing scheme can be devised where the need for an HDA can be eliminated by placing the secret routing information in a public directory. Similarly, the routing mechanism can be refined by the introduction of several routing levels in order to keep only a portion of the routing information secret.

8 Future Work

This paper represents only an initial effort in the area of secret sets and their applications. There remain a number of items for follow-up work:

- The CRT and bit vector constructs need to be compared more systematically. Intuitively, the relationship between the total member population (n) and the cardinality of the secret set (m) can be used to determine which of the two methods is more efficient. However, other considerations (e.g., susceptability to A2 attacks) need to be reckoned with.

- Methods presented in this paper are all relatively simple. There needs to be further investigation of more sophisticated secret set constructs.
- Parallel security services such as secret set data integrity, and originator (source) anonymity need to be integrated with the secret set constructs.
- There are, undoubtedly, application examples other than the two presented above.
- A reference implementation of the secret set functions needs to be built. This can serve as a basis for the actual performance evaluation.

References

[1] D. Chaum, *Security Without Identification: Transactions Systems to Make Big Brother Obsolete,* CACM Vol. 28, No. 10, October 1985.

[2] M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption,* EUROCRYPT'94.

[3] G. Chiou and W. Chen, *Secure Broadcasting Using the Secure Lock*, IEEE Transactions on Software Engineering, Vol. 15 No. 8, August 1989.

[4] X. Lai and J. Massey, *A proposal for a New Block Encryption Standard,* EUROCRYPT'90.

[5] R. Rivest, A. Shamir, and L. M. Adleman, *Cryptographic Communications System and Method,* U.S Patent 4,405,829, September 1983.

[6] A. Pfitzmann and M. Waidner, *Networks Without User Observability – Design Options,* EUROCRYPT'85, Also in Computers & Security, Vol. 6 No. 2, 1987.

[7] D. Chaum, *The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability,* Journal of Cryptology, Vol. 1 No. 1, 1988.

[8] A. Fiat and M. Naor, *Broadcast Encryption,* CRYPTO'93.

[9] D. Boneh and R. Venkatesan, *Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes,* CRYPTO'96.

[10] C. Blundo and A. Cresti, *Space Requirements for Broadcast Encryption,* EUROCRYPT'94.