Limits of Anonymity in Open Environments

Dogan Kedogan[1], Dakshi Agrawal[2], and Stefan Penz[1]

[1] Aachen University of Technology,
Computer Science Department Informatik IV,
Ahornstr. 55, D-52074 Aachen, Germany
{kesdogan,penz}@i4.informatik.rwth-aachen.de
[2] IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA
agrawal@us.ibm.com

Abstract. A user is only anonymous within a set of other users. Hence, the core functionality of an anonymity providing technique is to establish an anonymity set. In open environments, such as the Internet, the established anonymity sets in the whole are observable and change with every anonymous communication. We use this fact of changing anonymity sets and present a model where we can determine the protection limit of an anonymity technique, i.e. the number of observations required for an attacker to "break" uniquely a given anonymity technique. In this paper, we use the popular MIX method to demonstrate our attack. The MIX method forms the basis of most of the today's deployments of anonymity services (e.g. Freedom, Onion Routing, Webmix). We note that our approach is general and can be applied equally well to other anonymity providing techniques.

1 Introduction

Anonymity and unobservability techniques to prevent network traffic analysis are not new. The basic techniques in this area date back to the 1970's and 1980's when David Chaum and others suggested several revolutionary techniques including broadcast and implicit addresses, MIXes and DC-Networks [1–5]. The goal of these techniques is to preserve the privacy of users by hiding the traffic information—who has communicated with whom, for how long and from which location. Pfitzmann and Waidner presented the basic techniques in their seminal paper [5]. Since then various enhancements and extensions in theory and practice have been proposed. It can be shown that their technique forms the basis of the most known works[3] and that they can provide perfect[4] protection if applied in

[3] In more recent times, a new technique providing perfect protection was discovered independently by two different groups [6, 7]. This technique, known as Private Information Retrieval (PIR), has similarities to the DC-Networks.

[4] MIX technique uses public key encryption, thus the technique provides perfect anonymity, iff encryption is not considered as a limiting factor [2, 8].

closed environments, e.g. the number of users is known and is not too large (say less than 1000).

Among these techniques, the MIX concept can be considered as the most popular and deployment friendly. As a result, it has been proposed for various networks like GSM, ISDN and the Internet [9–14, etc.]. Since these networks cannot be considered as closed environments, a natural question arises: what happens to the protection level of anonymity providing techniques when the application environment changes from closed to open? In this paper, we address this question.

2 Basic Notions: Open environments, Anonymity and Anonymity Set

The challenge for anonymity-providing techniques in an *open environment* is to accomplish their basic goal even if:

a) The underlying communication network is global and is not subject to any topology restrictions. And as a consequence of this, we assume:
   - The set of users of an anonymity technique is an undetermined subset of all subjects worldwide.
   - There are no general time agreements between these subjects, thus the participants of the anonymity technique vary from time to time.

b) The attacker[5] E is able to tap all transmission lines of the communication network and controls all but one intermediary switching node. The attacker E is not able to break the cryptographic techniques chosen in the communication network.

The question now is how to hide the existence of any communication relationship, i.e. that a message was sent (sender anonymity) or received (receiver anonymity) by a user. Although the content of a message can be effectively protected by cryptographic techniques, the use of cryptography alone cannot guarantee anonymity. The omnipresent attacker E can observe the sender of a message and follow the message up to the receiver, thereby detecting the communication relation without any need to read the content of the transmitted message.

Hence, the decisive goal of an anonymity technique is to organize additional traffic in order to confuse the adversary and conceal communication relationships. To achieve this goal, the sender and/or receiver of a message must be embedded in a so-called *anonymity set* [15].

Definition 1 *Given an attacker model E and a finite set of all users Ψ. Let R be a role for the user (sender or recipient) with respect to a message M. If, for an attacker according to model E, the a-posteriori probability p that a user $\mathfrak{u} \in \Psi$ has the role R with respect to M is non-zero ($p > 0$), then $\mathfrak{u}$ is an element of the*

[5] Note that the same attacker model in open environments leads to a stronger attacker than in closed environments.

anonymity set $\mathfrak{A} \subseteq \Psi$. *A technique (method) provides an anonymity set of size* n *if the cardinality of* $\mathfrak{A}$ *is* n ($n \in \mathbb{N}$).

Thus, the sender or the receiver is anonymous only within the anonymity set. In the proposed open environment scenarios, the anonymity set of a particular user would change with time, and indeed, the changing anonymity sets of a particular user may be disjoint, be the same or may overlap. In our work we analyze these changing anonymity sets and show that it is possible to discover all peer communication partners of a chosen subject (e.g. of Alice) without requiring a large number of anonymity sets of the subject.

In the next sections we will present the MIX technique and use it as the object of our investigation.

3 The MIX Concept

MIXes collect a number of packets from distinct users (anonymity set) and process them so that no participant, except the MIX itself and the sender of the packet, can link an input packet to an output packet [2]. Therefore, the *appearance* (i.e. the bit pattern) and the *order* of the incoming packets have to be changed within the MIX. The change of appearance is a cryptographic operation, which is combined with a management procedure and a universal agreement to achieve anonymity:

User Protocol: All generated data packets including address information are padded to equal length (agreement), combined with a secret random number RN, and encrypted with the public key of the MIX node (see also [16]). A sequence of MIXes is used to increase the reliability of the system.

MIX Protocol: A MIX collects b packets (called *batch*) from distinct users (identity verification), decrypts the packets with its private key, strips off the RNs, and outputs the packets in a different order (lexicographically sorted or randomly delayed). Furthermore, any incoming packet has to be compared with formerly received packets (management: store in a local database) in order to reject any duplicates. Every MIX (except the first) must include a functionality ensuring that each received packet is from a distinct user, e.g. apply an anonymous loop back[6], because only the first MIX can decide whether or not the packets are from distinct senders.

E.g. assume that **A**lice wants to send a packet M to **B**ob (Fig. 1). A must encrypt the packet two times with the public keys c_i of the respective MIXes and include the random numbers RN_i: $c_1(RN_1, c_2(RN_2, B, M))$

[6] Loop back: Every MIX knows the sender anonymity set. It signs the received packets and broadcasts them to the respective users. Each user inspects whether his own packet is included or not and transmits a yes or no. The MIX goes on if it receives yes from all members of the anonymity set.

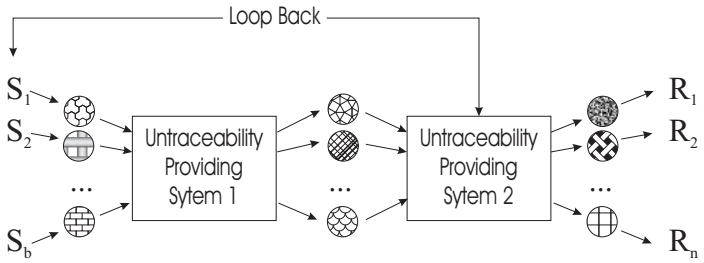

Fig. 1. Cascade of two mixes

Applying this protocol in closed environments where all subjects participate in all anonymity sets, the MIX method provides full security. The relation between the sender and the recipient is hidden from an omnipresent attacker as long as:

a) One honest MIX is in the line of the MIXes which the packet passes.
b) The $(b-1)$ other senders do not all cooperate with the attacker.

[8] states that the MIX method provides information-theoretic deterministic anonymity based on complexity-theoretic secure cryptography.

3.1 Related Works: Vulnerabilities of the MIXes

In the literature, several attacks have been proposed on anonymity techniques (see e.g. [17, 18, etc.]). Our work in this paper is most closely related to the *intersection attack*. The intersection attack gains information about a targeted user through repeated observations of the anonymity sets belonging to the targeted user. Since the intersection of two different anonymity sets is likely to be smaller than either of the anonymity sets (due to the assumed regularity in behavior), different intersections of anonymity sets could be used to gain information about the targeted user (see for such an analysis e.g. [18, 19]).

A more powerful attack is the $(n-1)$-*Attack* tackling directly the anonymity function of a MIX [8]. If a MIX cannot decide whether the packets are from different senders[7], the attacker can intercept the incoming packets, isolate each packet, and forward it together with $(n-1)$ of his own packets. This is also known as a *trickle attack* [10]. Note that also MIX variants like MIXmaster [20] are insecure against this attack [10, 21].

In [15] a MIX method (i.e. Stop-and-Go Mixes) an alternative approach is suggested with the goal of providing probabilistic security against the $(n-1)$-Attack with a security parameter μ. It was shown that a linear change of this

[7] Unfortunately, this is the case for all deployments in the Internet. Either the loop back functionality is not implemented or they assume a global Public Key Infrastructure (PKI), which is not existent yet.

parameter μ should have an exponential effect on the protection level of the method and on the achieved anonymity size n. However, recent research work [22, 23] limits the exponential effect on the size of the anonymity set by neglecting small probabilities, e.g. by neglecting all participants with a-*posteriori* probability less than $p \leq 0.0001$ (see Definition 1).

In our work we abstract from a special MIX realization and assume that a MIX can build a secure anonymity set. Furthermore, we assume that all known or unknown problems (i.e. vulnerabilities) attacking the anonymity function of a MIX are solved. These assumptions allow us to compute the fundamental weakness of anonymity techniques regardless of a particular implementation. To crystallize these notions, in the next section we will present a formal model of a MIX.

4 Formal Model: The Random Communication Model

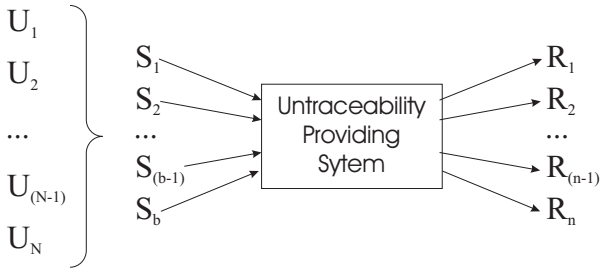

Fig. 2. Formal Model

In the literature, several extensions and modifications (see e.g. [17]) have been proposed to the original MIX technique. Since the focus of this paper is on analyzing the fundamental protection bounds of MIXes, we will assume a formal MIX model which is secure except for the fact that not all users are synchronized.

Our formal model is based on the following assumptions:

- There are N users $U_1, U_2, \ldots, U_N$ in the system.
- The untraceability providing system, i.e. a MIX, provides perfect untraceability between incoming and outgoing packets.
- The batch size of the system is b, where $1 < b < N$, and a batch may contain a receiver more than once. Thus, the size n of the anonymity set fulfills the following condition $n \leq b$ (see also Definition 1).
- The b packets in a batch are created by b different senders.
- Alice is one of the senders S_i, $1 \leqslant i \leqslant N$, and she uses the system to hide her m, $1 \leqslant m \leqslant N$, communication partners.

– Alice chooses her communication partner in each communication uniformly among her m partners.
– Other senders choose their communication partners uniformly among all N users.
– The attacker E takes notice of each untraceable communication act of Alice. This triggers the attacker to write down all recipients who are involved in this untraceable communication process, that is, the attacker simply records only those *recipient sets* which include a communication partner of Alice. For the sake of simplicity, we will enumerate the time t with increasing integer numbers whenever Alice sends a packet. Thus, when Alice communicates for the first time, $t = 1$, when she communicates for the second time, $t = 2$, and so on. We will denote the recipient set at time t by $R_t = \{R_t^1, \ldots, R_t^n\}$.

We now make several comments about these assumptions:

– Our formal model contains three essential parameters, namely user population N, batch size b, and the number of communication partners of the intended target m. These parameters can be easily identified in other anonymity providing techniques.
– The random communication behavior of the users determines the size of anonymity set n.
– Typically in open environments, the total number of users N is large. The anonymity size and the batch size ($n \leq b$) is likely to be small in comparison to N.
– Note that the number of communication partners of the intended target (Alice) m includes pseudo partners chosen by the target for sending dummy packets.

We note that our formal model contains three critical assumptions: Alice has m communication partners, she chooses her communication partners at random, and the recipient sets of all users are uncorrelated[8]. Later in this section, we will provide a justification for these assumptions. However, before we can justify these assumptions, we need to discuss our attack.

First, we make the following claim for our model:

Claim 1 *The MIX method is insecure if* $m \leqslant \lfloor N/n \rfloor$.

Proof A potential attacker can proceed in two stages: *the learning phase* and *the excluding phase*. In the learning phase the attacker waits until he observes m mutually disjoint recipient sets $(R_1, \ldots, R_m)$, i.e., for all $i \neq j$, $R_i \cap R_j = \varnothing$. After the learning phase, the attacker can be sure that in each set R_i, there is only one peer communication partner of Alice. In the excluding phase of the attack, the recipient sets $(R_1, \ldots, R_m)$ are refined using further observations. This can be done by using a new recipient set R which intersects with only one

[8] This assumption leads to a uniform distribution for the $(b-1)$ other recipients in a batch.

prior recipient set, that is, if $R \cap R_i \neq \varnothing$ and $R \cap R_j = \varnothing$ for all $j \neq i$. In that case, R_i can be refined to $R_i \cap R$. The refinement process is continued until each of the sets $R_1, \ldots, R_m$ contains only one user. It is clear that the remaining m users in $R_1, \ldots, R_m$ are the communication partners of Alice. ▪

We call the above attack the *disclosure attack*.

4.1 Success of the Disclosure Attack

Since the senders are not coordinated, the probability that one of the phases does not find the needed batches (i.e. m disjoint batches or batches that overlap with only one of these) converges to zero as the number of observations grows. Hence, we can deduce that with probability one the attack succeeds after a finite number of observations. This number forms an upper bound on the protection limit of the MIXes.

The number of observations that our attack needs to succeed clearly depends on the composition of the batches, which can be seen as a stochastic process as the senders are not coordinated. Thus the number of observations is a random variable $\mathbf{X}$. Our task is to calculate the statistical characteristics such as the mean $E(\mathbf{X})$ and the variance σ^2 of this number.

4.2 Explanation of the Critical Assumptions

Now we can justify the three critical assumptions made in our formal model. Clearly it is unrealistic to assume that Alice has a constant number of communication partners m for her whole life time. Thus, it is reasonable to restrict the time to a period of T, where Alice communicates frequently to m peer partners. Assuming this, following strategy can be applied to determine the number of communication partners of Alice:

> Assume, the real number of partners in a given time period is $\bar{m}$. If the attacker overestimates[9] m such as $m = \bar{m} - k$ where $k > 0$ and $k \in \mathbb{N}$, then the first phase of the disclosure attack can not be applied. Thus, the attacker adjusts m to $m := m - 1$ and applies the first phase again until the first phase is successful.
>
> We note that the above strategy to find m could be computationally intensive. However, in this paper we are interested in the fundamental limits of the MIXes regardless of the computation power of the adversary.

The second critical assumption of our formal model, that is, Alice chooses her communication partners randomly, is also unrealistic. However, here we are interested just in the upper bounds on the protection provided by a MIX.

[9] Of course, the attacker can only be sure that he has a false estimation for m, if he has already "enough" observations to apply the disclosing attack. To decide the point "when it is enough", depends on the stochastic structure of the random model. We will analyze this in the next sections.

Since some partners occur more frequently than others, the attacker would need more observations to conclude the learning phase and the excluding phase of the attack. Thus, the number computed by assuming equally frequently partners would be a lower limit on the number of observation required by an adversary. In other words, it would provide an upper limit[10] on the protection provided by the MIXes.

Our third assumption can also be justified on the similar grounds. If the recipient sets for different users were correlated, then it would take an adversary more time to conclude the learning and excluding phase. For example, this would be the case, if another user had exactly the same set of recipients as Alice and contributed frequently to the MIX. In this case any batch in which this user and Alice send a packet would have two communication partner of Alice and the observation would be useless for the learning phase. Thus our assumption leads to a lower bound on the number of observations needed by an adversary, or in other words, an upper limit on the protection level of the MIXes.

5 Simulation Results

In order to determine the statistical characteristics of the number of observation required by an adversary, we wrote a simulator, which performs the attack described above. In the following subsections, we will describe the four main parts of this simulator.

5.1 Generating Observations

The simulator identifies each of the N possible recipients by a unique number out of $\{1, \ldots, N\}$. Without any loss of generality, we assume that the numbers of the subset $\{1, \ldots, m\}$ represent the m communication partners of Alice. Hence, an observation is represented by a set of b numbers out of $\{1, \ldots, N\}$, where at least one of these numbers is from the range $\{1, \ldots, m\}$. To generate an observation, the simulator draws randomly a single number out of $\{1, \ldots, m\}$ and $(b-1)$ more numbers out of $\{1, \ldots, N\}$. Note that a recipient set may contain a number more than once.

```
FUNCTION generateObservation(N,b,m)
  r := random(1,m);
  R := {r};
  FOR i := 1 TO (b-1)
    r := random(1,N);
    R := R ∪ {r};
  RETURN R;
```

[10] In order to determine the protection bound of a security scheme two approaches are common. Either the *strongest* possible attack which *cannot break* the scheme has to be determined or the *weakest* attack which *can break* the scheme.

The function `random(x,y)` returns a random integer number out of the range $[x, \ldots, y]$. For simplicity, we use a standard random number generator that provides uniformly distributed numbers.

5.2 Learning Phase

The goal of the first phase is to find m mutually disjoint recipient sets. We are interested in the number of observations that an attack needs to reach this goal. The program iteratively generates new observations and checks if m recipient sets of these observations are mutually disjoint. When the check is successful, the number of observations and the found recipient sets are returned.

```
FUNCTION learning_phase(N,b,m)
  t := 0;
  O := ∅;
  WHILE O = ∅
    t := t + 1;
    Rt := generate_observation(N,b,m);
    O := find_disjoint_sets({R1,...,Rt},m);
  RETURN t,O;
```

Clearly, the function `find_disjoint_sets({R1,...,Rt},m)` is the most interesting part. It searches for m mutually disjoint sets within $\{R_1, \ldots, R_t\}$. Unfortunately, this problem turns out to be NP-complete. Note that solving this problem is a serious matter for us, but not for the attacker, if he is assumed to have unlimited computing power. In fact, the function `find_disjoint_sets()` forms the major performance bottleneck of the simulation. Our approach to solve this problem is to use enhanced backtracking algorithms. Therefore we transform the problem into a binary Constraint Satisfaction Problem (binary CSP, see [24]) and use well-known backtracking methods on the transformed problem. A CSP consists of a set of m variables $X = \{x_1, \ldots, x_m\}$, a value domain D_a for each variable x_a and a set of constraints. Each value domain is a finite set of values, one of which must be assigned to the corresponding variable. A constraint is a subset of the Cartesian product of the domains of some variables. This subset contains all allowed combinations of values for the corresponding variables. In a binary CSP all constraints are defined over pairs of variables. The goal of a CSP is to assign a value to each variable, so that all constraints are satisfied.

```
FUNCTION find_disjoint_sets({R1,...,Rt},m);
  csp := transform_to_csp({R1,...,Rt},m);
  solution := solve_csp(csp);
  O := retransform_solution(solution);
  RETURN O;
```

A straightforward transformation is to take each observation R_i as a value i $(i = 1, \ldots, t)$. Hence, the CSP is to assign values out of $1, \ldots, t$ to the m variables $x_1, \ldots, x_m$, so that these values correspond to observations that are

mutually disjoint. Therefore, the binary constraints consist of all pairs of values, whose corresponding observations are disjoint, i.e. the pair $(x_a = i, x_b = j)$ is allowed for all $a, b \in \{1, \ldots, m\}$, if and only if $R_i \cap R_j = \varnothing$.

```
FUNCTION transform_to_csp({R1,...,Rt},m);
  constraints := ∅;
  FOR a := 1 TO m
    Da := 1,...,m;            (domain of variable xa)
  FOR a := 1 TO m
    FOR b := 1 TO m
      FOR i := 1 TO t
        FOR j := 1 TO t
          IF Ri ∩ Rj = ∅
            THEN constraints := constraints ∪ {(xa=i,xb=j)};
  RETURN ({x1,...,xm},{D1,...,Dm},constraints);
```

For common input sizes, this transformation results in a huge search space and the search is often computationally infeasible. For our simulation, we use supplementary knowledge about the observations to significantly reduce the search space.

Firstly, we delete all observations that contain more than one of the peer partners. Since every observation in the solution contains exactly one of the peer partners, this deletion does not affect the completeness of the search.

Secondly, we divide the observations into m classes $C_1, \ldots, C_m$. Each class C_a consists of all observations that contain the peer partner a. Clearly, a solution consists of m observations, one out of each class. Hence, we can take the classes as domains for the CSP, i.e. the domain D_a consists of all observations that also belong to the class C_a.

```
FUNCTION transform_to_csp({R1,...,Rt},m);
  constraints := ∅;
  FOR i := 1 TO t
    IF contains_two_partners(Ri) THEN delete(Ri);
  FOR a := 1 TO m
    Da = {i | a ∈ Ri}           (domain of variable xa)
  FOR a := 1 TO m
    FOR b := 1 TO m
      FOREACH i IN Da
        FOREACH j IN Db
          IF Ri ∩ Rj = ∅
            THEN constraints := constraints ∪ {(xa=i,xb=j)};
  RETURN ({x1,...,xm},{D1,...,Dm},constraints);
```

Note that, in general, the attacker does not have the supplementary knowledge used by us. However, the attacker would find the same constraints as the ones found above even if he does not use the supplementary knowledge. Thus, using the supplementary knowledge significantly speeds up our simulations, but

the results, i.e. the number of required observations for a successful attack, would be the same for both transformation procedures.

The function `solve_csp(csp)` searches for a solution of the transformed problem and returns the first solution found, if any exists. This may be done by traditional backtracking algorithms. Our simulator makes use of a C-library developed by van Beek [25] that provides more sophisticated procedures to solve CSPs. An overview of the implemented procedures can be found in [24] and [26]. We experienced significant performance improvements by using the procedures developed by van Beek.

When `solve_csp(csp)` has found a valid variable assignment for the CSP, the function `retransform_solution(solution)` converts the returned solution to a solution O of the original problem. The observation R_i is an element of O, if and only if the solution of the CSP contains i. If `solve_csp(csp)` has not found a solution, the empty set $\varnothing$ is returned.

5.3 Excluding Phase

The goal of the second phase is to exclude elements from the recipient sets returned by the learning phase until each recipient set contains exactly one element, representing the peer partner. Again, we are interested in the number of observations the attacker needs to succeed.

For simplicity, we denote the set of mutually disjoint recipient sets returned by the learning phase by $O = \{O_1, \ldots, O_m\}$ and the recipient set of the new observation by $R_i \quad (i = 1, 2, \ldots)$. The simulation iteratively generates new recipient sets R_i and checks, if it is disjoint to all but one recipient set $O_j \in O$. If this is the case, O_j is replaced by the intersection $O_j \cap R_i$.

Even if the recipient set of a new observation overlaps with more than one O_i, it may be applicable later, when some elements are excluded from the affected recipient sets in O. Therefore, every time a new recipient set is found to be intersecting with only one recipient set in O, we have to check if any of the previously generated recipients sets is now applicable. Likewise, every time an old recipient set is found to be intersecting with only one recipient set in O, we have to check all old observations again for their applicability.

```
FUNCTION excluding_phase(N,b,m,O={O1,...,Om})
  t := 0 ;
  REPEAT
    t := t + 1;
    Rt := generate_observation(N,b,m);
    IF ∃i (Oi∩Rt≠ ∅  ∧  ∀j≠i Oj∩Rt=∅)
      THEN
        Oi := Oi∩Rt;
        Rt := ∅;          (Rt should not be checked again)
        REPEAT
          changed := FALSE;
          FOR k := 1 TO (t-1)
```

```
            IF ∃i (Oi∩Rk≠∅  ∧  ∀j≠i Oj∩Rk=∅)
             THEN
               changed := TRUE;
               Oi := Oi∩Rk;
               Rk := ∅;   (Rk should not be checked again)
        UNTIL changed = FALSE
  UNTIL ∀i∈{1,...,m}  |Oi|= 1
  RETURN t,O;
```

When this function intersects an observation R_i with an observation out of O, R_i must not be checked again, because this would lead to an infinite loop. Hence, we set $R_i = \varnothing$, so that the `IF`-clauses return `FALSE` when R_i is checked.

Note that some of the observations generated during the learning phase may be usable for the excluding phase. Hence, if we are interested in the total number of observations needed for both phases, we have to also check the observation generated during the learning phase for their intersection with O.

Repeating the Simulation. It is clear that the number of observations depends on the randomly generated observations. Therefore we need to repeat a simulation several times with different streams of random numbers in order to compute a mean that satisfies some desired statistical requirements.

After each simulation, we compute a 95% confidence interval for the mean, that is an interval which contains the real mean with a probability of 0.95. In general, the size of the confidence interval decreases with an increasing number of repetitions. We stop repeating the simulation when the confidence interval is sufficiently small, i.e. the size of the interval does not exceed 5% of the computed mean.

5.4 First Results

In our first simulations we were interested in in the effect of N, b and m on the number of observations an attacker needs to succeed. To see the effect of changing, N, b, and m, we first chose typical values for these parameters, viz, $N = 20000$, $b = 50$ and $m = 20$. Then we ran simulations with different values for one of the parameters while keeping the other two parameters unchanged.

We note that our simulations in the excluding phase do not make use of observations generated during the learning phase. Therefore, the total number of observations needed to complete the whole attack may be less than the sum of the numbers needed for each phase as shown here.

Number of Peer Partners (m). For a fixed number of total users N, and batch size b, as the number of peer partners m grows, it clearly becomes more difficult to find m mutually disjoint sets during the first phase. Similarly during the excluding phase, it gets harder to find sets that overlap with only one of

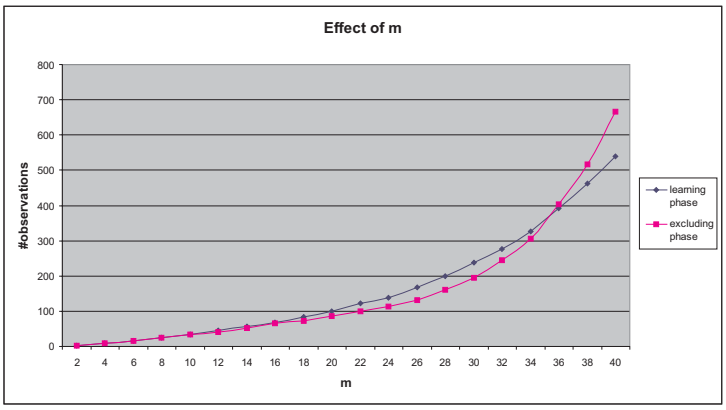

Fig. 3. Effect of m on the number of observations

the observations found in the learning phase since as m grows the number of elements in m mutually disjoint sets also grows.

Figure 3 shows a graph of the number of required observation for both phases of the attack. For high values of m, the excluding phase turned out to be more critical than the learning phase. For low numbers the difference between the number of observation required for two stages is insignificant. Further analysis showed that for large values of m ($m > 30$), the number of observation needed during the excluding phase grows exponentially in m.

Size of the Batch (b). One way for the anonymity provider to influence the security of the system is to take care of a suitably large batch size. Obviously, as b grows the attacks becomes very difficult, because if the total number of users N stays the same, it is much harder to find m large sets that are mutually disjoint than to find m small ones. Similarly, the probability that a set overlaps with only one of m sets becomes very small for large values of b. Hence the number of observations needed is expected to grow in b. The graph of the simulation results in Figure 4 supports this hypothesis. Again, the excluding phase turned out to be the more critical one for large values of b.

For $b < 50$, the number of observations is nearly constant. We deduce that for small batches other parameters are more important. On the other hand, for $70 < b \leq 85$ the number of required observations made out a fast exponential growth in b, approximately doubling the number of observations needed for the excluding phase every time the batch size grows by five elements.

Total Number of Users (N). In contrast to m and b, the attack becomes harder as N decreases, because the probability to generate a set that is disjoint to $(m-1)$ mutually disjoint sets or to generate a set that overlaps with only

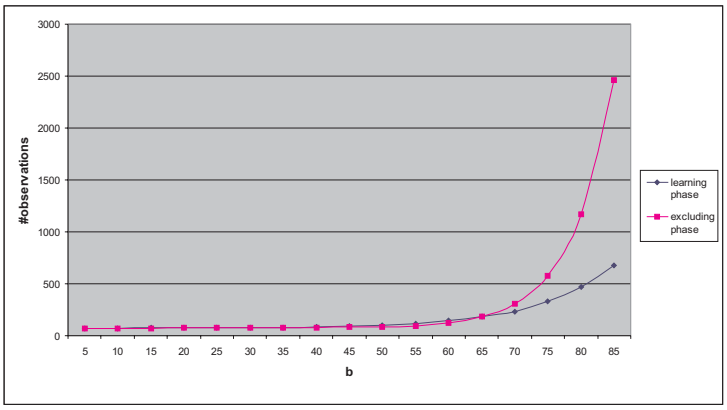

Fig. 4. Effect of b on the number of observations

one of these sets, is very small if you can choose the recipients only from a small domain. Figure 5 shows a graph of our results.

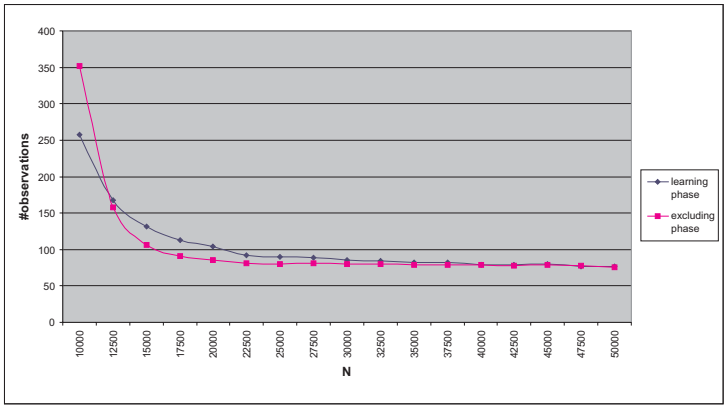

Fig. 5. Effect of N on the number of observations

Similar to the graph for the effect of b, we can see that for $N > 20000$, the observations needed remain nearly constant. On the other hand, for $N < 10000$ the graph for the excluding phase shows an exponential behavior. Again, when the total number of users becomes less, the number of observations needed for the excluding phase increase faster than for the learning phase.

Summary. The first experiments showed that it is essential for the security of an anonymity system to choose the three critical parameters, b, m and N, carefully. To achieve a sufficient anonymity, b and m should be as big as possible, and the total number of users must not exceed a certain limit.

One quite surprising result is that, when the problems get harder, that is for large b and m, and for small N, the excluding phase requires significantly more observations than the learning phase. On the other hand, the learning phase requires a lot more of computing power.

6 Conclusions

This paper presents a formal model of an anonymity service and identifies three critical parameters of such a service: the total number of users, batch size, and the number of peer communication partners. These parameters determine the fundamental protection provided by an anonymity service independent of the computing power of a potential adversary. The paper also provides a methodology to determine an upper limit on the fundamental protection provided by an anonymity service.

Specifically, the paper applies this methodology to the MIXes, and shows that the protection limit of a MIX increases exponentially in the three critical parameters once these parameters cross a certain threshold. The paper provides these thresholds for typical values of the parameters for a MIX.

In the hindsight provided by this paper, it is clear that the optimum values for these parameters are dependent on each other. Just choosing these values independently of each other does not necessarily result in a more secure system. In order to compute the optimum batch size it is essential to consider the total number of users in the system and the average number of communication partners of a user and compute the minimum threshold value for the size of the batch.

In our future work, we would explore the stochastic nature of our formal model. It turns out that the learning phase and the excluding phase of the attack can be modelled as a Markov process. Our goal is to give analytical formulas to compute the threshold limits and verify them using the simulations. Furthermore, we would examine the effect of three critical assumptions made in this paper on the tightness of our limit on the protection level of an anonymity service.

References

1. D. Chaum: The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology, 1:65-75, 1988.
2. D. Chaum: Untraceable electronic mail, return addresses and digital pseudonyms. Communications of the A.C.M., 24(2):84-88, February 1981.
3. D. J. Farber, K. C. Larson: Network Security Via Dynamic Process Renaming. Fourth Data Communication Symposium, 7-9 October 1975, Quebec City, Canada.

4. P. A. Karger: Non-Discretionary Access Control for Decentralized Computing Systems. Master Thesis, Massachusetts Institute of Technology. Laboratory for Computer Science, 545 Technology Square, Camebridge, Massachusetts 02139, Mai 1977, Report MIT/LCS/TR-179.
5. A. Pfitzmann, M. Waidner, Networks without user observability, design options. In: Advances in Cryptology. Eurocrypt '85, volume 219 of Lecture Notes in Computer Science. Spinger-Verlag, 1985.
6. B. Chor, O. Goldreich, E. Kushilevitz, M. Sudan: Private information retrieval. In: 36th IEEE Conference on the Foundations of Computer Science, pages 41-50. IEEE Computer Society Press, 1995.
7. D.A. Cooper, K.P. Birman: Preserving privacy in a network of mobile computers. In: 1995 IEEE Symposium on Research in Security and Privacy, pages 26-38. IEEE Computer Society Press, 1995.
8. A. Pfitzmann: Dienstintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz. IFB 234, Springer-Verlag, Heidelberg 1990 (in German).
9. H. Federrath, A. Jerichow, A. Pfitzmann: MIXes in Mobile Communication Systems: Location Management with Privacy. Information Hiding, LNCS 1174. Springer-Verlag, Berlin 1996, 121-135.
10. C. Gülcü, G. Tsudik: Mixing E-mail with BABEL. In: Symposium on Network and Distributed Systems Security (NDSS '96), San Diego, California, February 1996.
11. A. Jerichow, J. Müller, A. Pfitzmann, B. Pfitzmann, M. Waidner: Real-Time Mixes: A Bandwidth-Efficient Anonymity Protocol. IEEE Journal on Selected Areas in Communications, 1998.
12. A. Pfitzmann, B. Pfitzmann, M. Waidner: ISDN-mixes: Untraceable communication with very small bandwidth overhead. In: GI/ITG Conference: Communication in Distributed Systems, pages 451-463. Springer-Verlag, Heidelberg, February 1991.
13. M. G. Reed, P. F. Syverson, D. M. Goldschlag: Anonymous connections and onion routing. IEEE Journal on Special Areas in Communications, 16(4):482-494, May 1998.
14. M. G. Reed, P. F. Syverson, D. M. Goldschlag: Protocols using Anonymous Connections: Mobile Applications, Security Protocols. 5th International Workshop Proceedings. B. Christianson, B. Crispo, M. Lomas, and M. Roe (eds.). Springer-Verlag LNCS 1361, 1998, pp. 13-23.
15. D. Kesdogan, J. Egner, R. Büschkes: Stop-and-go mixes providing probabilistic security in an open system. In: David Aucsmith (ed.): Information Hiding: Second International Workshop, volume 1525 of Lecture Notes in Computer Science, pages 83-98. Springer-Verlag, Berlin, Germany, 1998.
16. B. Pfitzmann, A. Pfitzmann: How to Break the Direct RSA-Implementation of MIXes. Eurocrypt '89, LNCS 434. Springer-Verlag, Berlin 1990, pp. 373-381.
17. J. F. Raymond: Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. International Workshop on Design Issues in Anonymity and Unobservability, Berkley, LNCS 2009. Springer-Verlag, 2001.
18. O. Berthold, H. Langos: Dummy Traffic Against Long Term Intersection Attacks. Workshop on Privacy Enhancing Technologies, San Francisco, CA, USA, April 14-15, 2002.
19. M. Wright, M. Adler, B.N. Levine, C. Shields: An Analysis of the Degradation of Anonymous Protocols. Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS 2002), February 2002.
20. L. Cottrell: Mixmaster, http://www.obscura.com/ loki/.

21. D. Kesdogan: Evaluation of Anonymity Providing Techniques using Queueing Theory. The 26th Annual IEEE Conference on Local Computer Networks (LCN 2001), November 15-16, 2001, Tampa, Florida
22. A. Serjantov, G. Danezis: Towards an Information Theoretic Metric for Anonymity. Workshop on Privacy Enhancing Technologies, San Francisco, CA, USA, April 14-15, 2002.
23. C. Diaz, S. Seys, J. Claessens, B. Preneel: Towards Measuring Anonymity, Workshop on Privacy Enhancing Technologies, San Francisco, CA, USA, April 14-15, 2002.
24. R. Dechter, D. Frost: Backtracking algorithms for constraint satisfaction problems. An ICS technical report, September 1999.
25. P. v. Beek: A C-library of routines for solving binary constraint satisfaction problems. http://ai.uwaterloo.ca/ vanbeek/software/software.html.
26. V. Kumar: Algorithms for Constraint Satisfaction Problems, A Survey. AI magazine, 13(1):32-44, 1992.
27. O. Berthold, H. Federrath, S. Köpsell: Web MIXes: A System for Anonymous and Unobservable Internet Access. International Workshop on Design Issues in Anonymity and Unobservability, Berkley, 2009 LNCS. Springer-Verlag, 2001.
28. I. Goldberg, A. Shostack: Freedom network whitepapers.
29. C. Rackoff, D. R. Simon: Cryptographic defence against traffic analysis. In: Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing, pages 672-681, San Diego, California, 16-18 May 1993.