The Hitting Set Attack on Anonymity Protocols

Dogan Kesdogan and Lexi Pimenidis

Aachen University of Technology,
Computer Science Department Informatik IV,
Ahornstr. 55, D-52074 Aachen, Germany

{kesdogan,lexi}@i4.informatik.rwth-aachen.de

Abstract. A passive attacker can compromise a generic anonymity protocol by applying the so called disclosure attack, i.e. a special traffic analysis attack. In this work we present a more efficient way to accomplish this goal, i.e. we need less observations by looking for unique minimal hitting sets. We call this the *hitting set attack* or just HS-attack.
In general, solving the minimal hitting set problem is NP-hard. Therefore, we use frequency analysis to enhance the applicability of our attack. It is possible to apply highly efficient backtracking search algorithms. We call this approach the *statistical hitting set attack* or SHS-attack.
However, the statistical hitting set attack is prone to wrong solutions with a given small probability. We use here duality checking algorithms to resolve this problem. We call this final exact attack the *HS*-attack*.

1 Introduction

Although anonymity and privacy is only a small part of what today is called computer or network security, it plays a vital role in data protection. There are a couple of issues where data encryption in public networks is not enough. Amongst those are important fields like free speech, elections, health care and social guidance systems. The simple fact of exchanging data packets with some of those entities might already be interesting to third parties, i.e. people would like to use these systems anonymously.

On the contrary most network protocols, and especially TCP/IP, are not designed for anonymous data transfer. Anyone along the route of the packet through a network can observe the origin and the destination of this packet even if the payload is encrypted. Therefore, Chaum et al. have proposed anonymity protocols that grant protection against this sort of eavesdropping in closed environments[1] [Cha81,Cha88,CGKS95,CB95].

This paper focuses on an analysis of the strength of those protocols in an open system[2] like the Internet[GRS96,GT96,RR98,KEB98,KBS02]. In particular we

[1] i. e. the number of the users is some known and not too large number n (e. g. $n \leq 1000$).

[2] i. e. the number of potential users is more than one million and usually not known exactly.

investigate how long they can hide the user's communications from a passive attacker.

The first contribution of this paper is a new attack which also makes use of network traffic observations without interfering with the traffic of the network in any kind. By looking for hitting sets in the observations the new algorithm does the work faster, i.e. it needs less observations than any other attack known to us. We call the new attack the *hitting set attack* or **HS-attack**. However, it requires the solution of a NP-hard problem, i.e. the minimal hitting set problem.

We relax the strict approach of the minimal hitting set by using the most frequent candidates. We call this attack as the *statistical minimal hitting set attack* or **SHS-attack**. This attack does not rely upon solving NP-hard problems. It is also very easy scalable and can be applied in situations that were far beyond feasibility of the HS-attack.

But this advantage comes with a drawback: the risk of errors. The solutions found by the SHS-attack are approximations and thus object to possible errors. We present some error probability statistics of these and suggest strategies that can be applied to reduce the error to any arbitrarily measure.

Our third contribution is a method of refinement. We show how the approximation can be used to either show the correctness of itself or give a hint of how to look for more information in upcoming observations. Especially the work of Fredman and Khachiyan can be used to great advantage. We call this attack **HS*-attack**, since it is exact as the HS-attack and can use the statistical properties of the SHS-Attack, but requires more observations.

This paper is structured as follows: in the section 2 we discuss general information concerning anonymity techniques and related works. Thereafter we have laid down enough knowledge to start our work. After related works we derive and explain in section 4 the hitting-set attack in detail. Then, we present the statistical hitting set algorithm. There are details about it's implementation, optimizations and behavior. In section 6 we present the HS*-attack. Finally we conclude the paper in section 7.

2 Anonymity techniques

As already mentioned in the introduction there are a number of anonymity techniques to prevent eavesdroppers from gaining information about a user's traffic. Since the content can be encrypted, our focus is on the traffic layer. Anyone that can read a packet can see the origin and the destination. Anonymity techniques strive to prevent this.

As an example: Alice wants to post her political opinion to a web forum where oppositional members exchange information. Unfortunately she lives in a country where the government is suspected to track down oppositional members. If she would just send the encrypted message, e.g. using HTTPS, her Internet Service Provider (ISP) could notice this action and save this to a record. This could lead to a point where Alice herself could get suspected because she has exchanged data with some entity.

To avoid this, Alice could use some service like JAP [BFK01]. For this she installs a proxy on her computer that encrypts all of her traffic and sends it to a JAP proxy (i.e. Mixes [Cha81]). Along with her there are several other, maybe several thousand, users doing likewise. The server decrypts those packets and forwards them on behalf of the users. Any returned data will be send to the users on the same way.

Thus, any primary evidence has now gone. What remains is that Alice sends out data to an anonymity server (e.g. Mixes) which itself does not provide any other service than untraceable packet forwarding. Because of this functionality a potential attacker is not able to link an incoming packet to an outgoing packet. Using this service, Alice is beyond any suspicion to have send any packets to the oppositional forum because any of the other users could have been done it, i.e. Alice and the other persons builds the so called *anonymity set*.

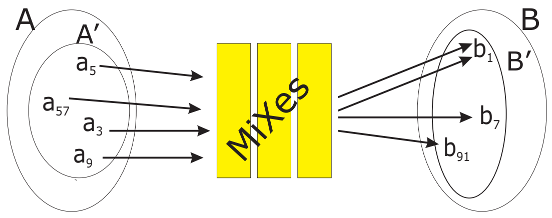

Fig. 1. Formal model of an anonymity set. In any anonymous communication (e.g. Mixes), a subset A' of all senders A sends a message to a subset B' of all recipients B.

2.1 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 [Cha81]. 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 [PP90]). A sequence of MIXes is used to increase the reliability of the system.

MIX Protocol: A MIX collects a 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, because only the first MIX can decide whether or not the packets are from distinct senders.

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 $(a-1)$ other senders do not all cooperate with the attacker.

[Pfi90] states that the MIX method provides information-theoretic deterministic anonymity based on complexity-theoretic secure cryptography.

2.2 Abstract Model

In this work we abstract from a specific type of anonymity service or implementation. Instead, we assume that a subset A' of all senders A sends a message to a subset B' of all recipients B, like shown in figure 1. Furthermore, in our model the adversary can easily determine anonymity sets, e.g. mixes assume that all network links are observable (see [Cha81]). However, this can be assumed also in a real world scenario if the attacker is able to observe messages to and from an anonymity service. We only assume the following properties of an anonymity system:

– In each anonymous communication, a subset A' of all senders A sends a message to a subset B' of all recipients B. That is, $A' \subseteq A$ and $B' \subseteq B$, as Figure 1 illustrates. In a particular system, the set of all senders A can be the same as the set of all recipients B.
– The size of the sender anonymity set[3] is $|A'| = a$, where $1 \leq a \ll |A|$. Note that a sender can even send multiple packets per batch.
– The size of the recipient anonymity set is $|B'| = b$, where $1 \leq b \ll |B|$ and $b \leq a$. That is, several senders can communicate with the same recipient.
– The anonymity system provides provides perfect untraceability between incoming and outgoing packets.

The typical values for $|A'|$, $|B'|$, $|A|$, and $|B|$ vary from implementation to implementation and with the environment in which they operate. In [BFK01] present an implementation in which $|A|$ is around $20,000$. They don't give typical values for $|A'|$, but we generally expect $|A'| < 100$.

To investigate the hitting set attack, we use the same formal model as suggested in [KAP02]. For the sake of simplicity we make certain assumptions. These assumptions are:

[3] Note that the above model can be easily adopted to other anonymity systems (e.g. pool-mixes) by determining the respective anonymity sets.

– In all observations $\mathcal{OBS} = \{B'_1, B'_2, \ldots\}$ Alice is a sender using the system to hide her m communication partners $\mathcal{B}_{Alice}$, i.e. $\forall B'_i \in \mathcal{OBS} : \mathcal{B}_{Alice} \cap B'_i \neq \emptyset$. This can be accomplished by restricting the attacker to observe only the anonymous communication of Alice. We will refer to the communication partners later on also as Alice's peers.
– Alice chooses a communication partner in each communication uniformly among her m partners $\mathcal{B}_{Alice}$, while the other senders choose their communication partners uniformly among all recipients B.

3 Related Works – Disclosure Attack

Disclosure attack is a traffic-analysis attack to identify all communication partners of a targeted user (Alice) [KAP02]. Since we follow here the same anonymity model disclosure attack has the same model properties, i.e. Alice uses the system to hide her communication partners $\mathcal{B}_{Alice}$ with $|\mathcal{B}_{Alice}| = m$ and the attacker knows the number of the peers, i.e. m.

A disclosure attack has a *learning phase* and an *excluding phase*.

Learning phase In this phase, the attacker's task is to find m mutually disjoint recipient sets – that is, each set has only one peer partner of Alice – by observing Alice's incoming and outgoing messages. We refer to these found sets as the *basis sets*.

Excluding phase The attacker's task in this phase is to observe new recipient sets until all Alice's nonpeer partners are excluded from the basis sets. Three possible outcomes exist:
  – *No intersection.* Contrary to our assumption, since none of the peer communication partners in the basis sets appear in the recipient set.
  – *Intersection only with one basis set.* The attacker knows that Alice's peer partner must be in the intersection (excluding act).
  – *Intersection with more than one basis set.* The attacker cannot tell which intersection contains Alice's peer partner.

The disclosure attack is an NP-complete problem. The proof, detailed elsewhere [KAP03], is technical and involves showing that the learning phase of the disclosure attack is equivalent to the well-known NP-complete Clique problem.

An enhancement of the disclosure attack is suggested in [Dan03], the *statistical disclosure attack.* The attack follows the general structure of the disclosure attack as suggested in [KAP02], but makes use of statistical properties in the observations and identify the victim's peer partners without solving the NP-complete problem. However, this is not for free. The solution is only correct with a given probability.

4 The Hitting-Set Attack

To investigate the hitting set attack we restrict ourselves again to observe Alice and to identify all her hidden peer partners $\mathcal{B}_{Alice}$. As before we continue observing only the anonymity sets where Alice is involved as a sender. We do not use any other properties of the anonymity system, thus the complete system can be described formally as follows:

$$\forall B'_i \in \mathcal{OBS} \text{ and } B'_i \subset B \; \exists b \in \mathcal{B}_{Alice} \subset B : b \in B'_i$$

Obviously, having only the above abstract system definition we can not identify (or exclude) any recipient $b \in B$ as the peer partner (or as not the peer partner) of Alice. All recipients are a priori equally likely.

Suppose $\beta = \{b_1, b_2, \dots b_m\} \subset B$ is the result of a successful attack, then informally the solution β has to be consistent with the system description, i.e. the following holds $\beta \cap B_i \neq \emptyset$ for all elements of $\mathcal{OBS}$. Suppose, two possible solutions $\beta = \mathcal{B}_{Alice}$ and $\beta' \neq \mathcal{B}_{Alice}$ are consistent with the anonymity system, i.e. $\forall B'_i : \beta \cap B'_i \neq \emptyset$ and $\beta' \cap B'_i \neq \emptyset$. There is no way for the attacker within the given system to pick the right solution. Clearly, it is not decidable wether β or β' is the true solution within the given model. Hence, to decide within the given system the following has to be met $\exists B'_j : \beta' \cap B'_j = \emptyset$. From this observations we suggest the following algorithm:

1. Since the attacker knows m, he can build all sets of cardinality m from the elements of B, we will call the collection of these sets of possible solutions as $\mathcal{S}_m$. Obviously, $\mathcal{B}_{Alice} \in \mathcal{S}_m$ because $\mathcal{S}_m$ contains all sets of size m. Note that $|\mathcal{S}_m| = \binom{n}{m}$.
2. The attacker excludes all sets in $\mathcal{S}_m$ that are not consistent with the observation, i.e. all sets in $\mathcal{S}_m$ that are disjunct with the observation.
3. If $|\mathcal{S}_m|$ becomes one, the remaining element has to be equivalent to $\mathcal{B}_{Alice}$ because all other sets of cardinality m proofed to be wrong.

Of course it is impossible for an attacker with space limitation to maintain a list of size $\binom{n}{m}$. For realistic numbers this would be about $\binom{20{,}000}{20} \approx 10^{68}$ elements. Although this number would decrease fast in the progress of the algorithm.

However, from the above problem description we can derive that the solution has to be a *unique minimum hitting set* of all observations in $\mathcal{OBS}$. First of all, the solution is not disjunct with any $B'_i \in \mathcal{OBS}$. So it is a hitting set. It is also unique, otherwise the algorithm would not have been stopped. And it is also minimal: if there would be some hitting set of a size smaller than m, all superior sets with cardinality m would be hitting sets, thus violating uniqueness. Consequently we define:

Definition 1 ***(Hitting Set)*** *Suppose $\mathcal{OBS}$ is a collection of sets. A hitting set for $\mathcal{OBS}$ is a set $H \subseteq \cup_{B'_i \in \mathcal{OBS}} B'_i$ such that $H \cap B'_i \neq \emptyset$ for each $B'_i \in \mathcal{OBS}$. A hitting set is minimal iff no proper subset is a hitting set.*

Garey and Johnson report that the hitting set problem is NP-complete. Notice, that the hitting set problem can be transformed to the vertex cover problem [GJ79].

5 Statistical Hitting Set Attack

Finding a unique minimal hitting set is NP-complete, but there is a helpful property that can be used, frequency analysis. Using frequencies we can apply some efficient though easy restrictions to the searching space. In fact Alice's peers appear more frequently in the recipient sets than the other hosts [Pim03]. Furthermore if an element is more frequent than others, it is more likely to be included in a hitting set. This fact can be exploited by a simple backtracking algorithm.

We restrict the search to those sets that are most likely to be a hitting set. This is done by counting the frequency of all elements in the observations. Those statistics are provided to backtracking algorithm. This algorithm will build a fixed number of hitting sets of size m. These sets are combinations of the elements with the highest frequency.

After checking some number of sets, the algorithm comes to decide whether the attacker has enough information. If only one out of those sets is valid, it is likely that it is the only one and thus equal to $\mathcal{B}_{Alice}$. If more than one sets have been found, the algorithm needs more observations and returns to the waiting status again (the same if no hitting set is found).

Applying the above strategy the search can be restricted to an arbitrarily small part of the complete searching space. However, this strategy is then of course not exact and may return a wrong result. In section 5.3 an analysis of failure probability depending on the number of tested sets is given.

5.1 Simulation of the Attack

In order to determine the properties of the attack we have written a simulation. Each simulation run was performed by generating observations until the stop criterion – only one valid combination is found – was given. The result was then validated for the number of observations needed and whether it was correct.

In fact, the code of the simulation could even be used to apply the attack, if real data is used as input data[4].

Inside the simulation the observations were generated using pseudo random numbers. The hosts in B are depicted by the numbers from 1 to n, and without loss of generality let $\mathcal{B}_{Alice} = \{1, 2, \dots m\}$. Such any saved observation $B'_i \in \mathcal{OBS}$ consists of $a - 1$ numbers from the set B and one element from $\mathcal{B}_{Alice}$. The numbers are chosen from the given intervals uniformly with the help of a random number generator.

After a new recipient set has been build, the main algorithm is started that looks for the minimum hitting sets by backtracking. The searching space is restricted to some number of sets that can be chosen in accordance to available resources and desired error probability.

Since we are looking for hitting sets that are build out of the most frequent elements we need to compile a list of all hosts contained in the observations,

[4] We also wrote an implementation of the attack that can be deployed on real traffic.

sorted by frequency. We name this list $\mathcal{F} = \{F_1, F_2, \dots\}$, with F_i having at least as many occurrences in the observations as F_j for $i < j$ and $|F_i| = m$.

The backtracking looks like this:

```
FUNCTION Backtracking(Chosen, Observations, $m$, $\mathcal{F}$, counter)
# terminate after a given number of tests
if counter=0 then RETURN $\emptyset$
solutions = $\emptyset$
# backtracking recursion, if not chosen $m$ elements
if $|$Chosen$| < m$ then
   for each $F_i \in \mathcal{F}$ do
      # add next element to chosen set
      solutions = solutions $\cup$
         Backtracking(Chosen$\cup$Host, Observations, $m$, $\mathcal{F}-$Host, counter)
else
   # generated set of size $m$, check validity
   solutions = solutions $\cup$ Check_Validity(Chosen,Observations)
   counter = counter - 1
RETURN solutions
```

The function is called with

```
Backtracking($\emptyset$,Observations,$m$, $\mathcal{F}$, counter)
```

and returns a set of valid hitting sets of size m. To determine whether or not a set is a hitting set ("*valid*"), a very simple routine is used. In case the set is not a hitting set an empty set is returned, otherwise the set is returned.

Additionally the amount of checked sets can be limited with the variable `counter`.

```
FUNCTION Check_Validity(Combination,Observations)
for each recipient_set $\in$ Observations do
   if recipient_set $\cap$ Combination $= \emptyset$ then
      RETURN $\emptyset$
RETURN Combination
```

With this functionality the algorithm would already be ready for validation. Indeed there are still some possibilities to "adjust" the algorithm. Either the running time of the algorithm can be decreased or the probability of a wrong result. We discuss this in the next section.

5.2 Optimizations

There are a lot of parameters that can be used to improve the algorithm. The following list shows the most significant ones.

Interlace Since the backtracking in the basic form does not rely on any prior results, it may be considered unnecessary to start the backtracking algorithm on every new observation. Therefore, an attacker can collect a number of observations and then start the algorithm. Hence, the attack would be applied after a fixed number of observations, that can be specified by a step width.

This leads to a nearly linear improvement in speed. However, in average the number of observations would increase by the half of the step width.

Termination criterion To raise the probability of a correct result, the termination criterion can be changed. Instead of stopping at once if a unique set is found, the solution can be retained and further observations can be made. If no contradiction is found until then, the algorithm stops and returns the set.

Carry-over If there are more than one solution, it is quite likely that most of them will also be found at the next run of the backtracking. In this case following strategy can be applied: all found solutions are saved. The next time a backtracking should be started, all saved sets are checked for validity with the new observations. This can be done very fast. If at least two of those sets are still valid, the backtracking part is not needed to be started.

Our experience has shown that the gain is nearly independent of the number of saved sets. Therefore, it can be sufficient to save two sets.

Uniqueness The aim is to find a unique set. As soon as there is a second set (or even more) found during the course of the backtracking, it is clear that more observations will be needed and the backtracking can be stopped at once.

Since we have seen above that it is unnecessary to collect more than two solutions, we have another reason not to waste computing time by looking for more solutions per run.

As the success of the simulation depends heavily on the generated random numbers it had to be repeated several times until the mean values were acceptable.

Note that we run the simulation on a single personal computer with no special equipment. The speed of the computer is about 1 GHz and an average simulation run can be done in less than 10 seconds.

5.3 Validation

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

The average number of observations needed to compromise the system is shown in the next figures. To compare our result with the disclosure and the statistical disclosure attack we have added the respective results of the both attacks. The chosen error rate for both statistical attacks were 0.5%.

Figure 2 shows graphs of the number of required observations for all three algorithms, i.e. disclosure attack, statistical disclosure attack, and statistical

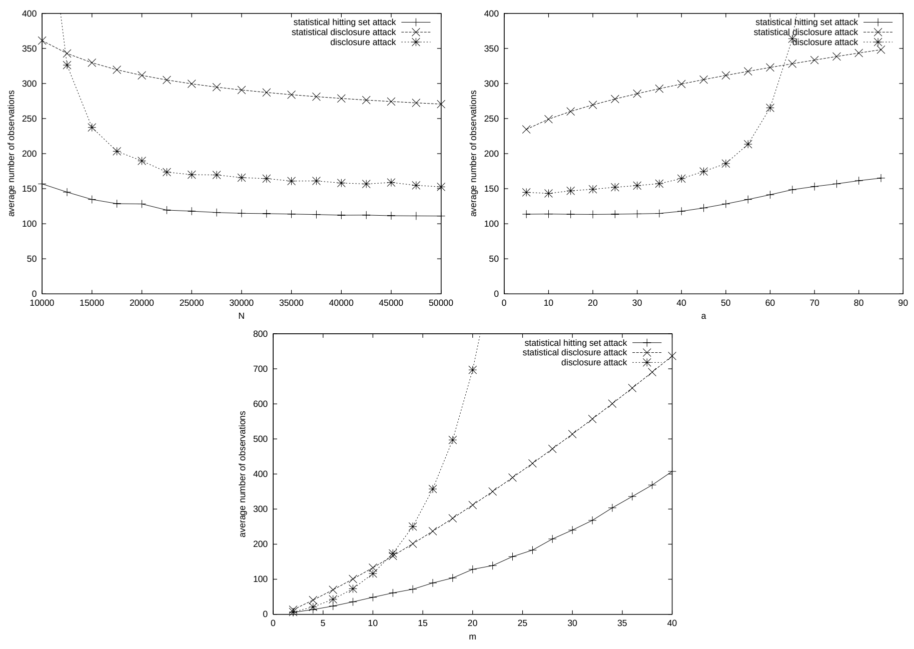

Fig. 2. Effect of n, a, and m on the number of observations

hitting set attack. Statistical disclosure attack and the statistical hitting set attack have nearly the same characteristic, i.e. for all values of n, b, and m the attack is still applicable (compare this with the exponential increase of the disclosure attack). The main difference between the statistical disclosure attack and the statistical hitting set attack is that the statistical hitting set attack needs less observations (see the above figure).

Figure 3 shows two graphs of the number of tested sets per backtracking, i.e. the number of tested possible solutions. The possible solutions are built using all combinations of the most frequent items. Intuitively, one expects a better error probability and a definite reduction of required number of observations if the number of possible solutions increases from one thousand to one million. However, as shown in figure 3 for all values after $10,000$ of possible solutions it turned out to be not so critical even negligible for systems with practical dimensions, e.g. with $n = 20000$, $a = 50$, and $m = 20$.

Changing the termination criterion can additionally reduce the error probability. In some series additionally 10% of observations reduces the rate of error of 80%, while 15% more observations could drop the rate down for 95%. That would be as large as 0.025% on a normal error rate of 0.5%.

We conclude that the statistical hitting set attack can break an anonymity protocol with a smaller amount of observation than it was possible before. Ad-

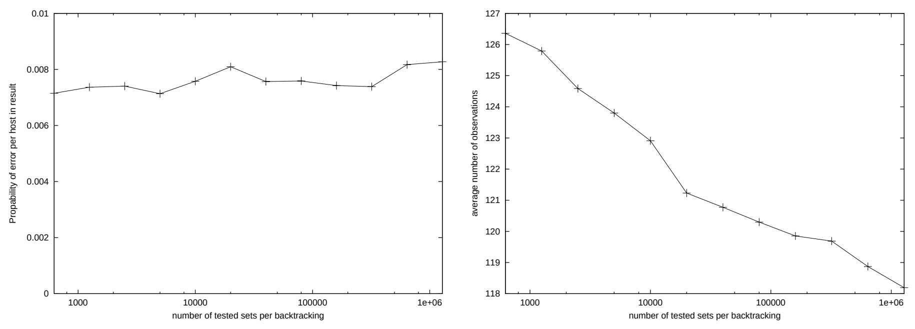

Fig. 3. Effect of tested sets on the correctness of the solution

ditionally the result is correct with a higher probability. The rate of error can also be reduced by adjusting certain parameters in the algorithm.

6 Refinement of the Hitting-Set Attack: the HS*-Attack

All of the known attacks, including the introduced hitting set attack, have certain drawbacks:

The disclosure attack relies on solving a NP-complete problem. Furthermore it needs in average more observations than the HS and SHS-attack. But it is a deterministic attack, i.e. the result is always correct.

The statistical disclosure attack needs less observations than the disclosure attack but relies on a unified probability distribution of the elements in the recipient sets. The result of it can possibly be wrong.

The hitting set attack is a deterministic attack but is NP-complete. It needs less observations than the prior known attacks.

The statistical hitting set attack is not an exact attack, but it needs less number of observations than the statistical disclosure attack. In exchange it needs more computational power and memory.

Next we are interested in an algorithm with the efficiency of the SHS-attack, but whose result is not subject to some possible errors. A suggestion for this follows: running the SHS-attack on some observations an attacker gains a good approximation to the result. Is there a possibility to check whether the result is correct? The answer is yes. By using the algorithm of *Fredman and Khachiyan* we are able to solve the problem in $\mathcal{O}(n^{\mathcal{O}(\log n)})$, i.e. superpolynomial complexity. This algorithm solves a dual algorithm to the minimum hitting set problem, the problem of boolean duality.

We do not describe the algorithm of Fredman and Khachiyan here due to space limitations. The authors themselves give a detailed view in [FK96].

6.1 The Algorithm

Given the input of two boolean formulas the algorithm of Fredman and Khachiyan decides if the formulas are dual or not. If not then the algorithm returns a counterexample. The counterexample is equivalent to a second valid set of peer partners.

The counterexample is especially convenient and useful in this application: it is equivalent to a second set of hosts which is a valid guess for Alice's peer partners. In that case one has to go back to the stage of the algorithm that uses the hitting set attack in order to find a unique minimum hitting set. But the counterexample can be very useful right now: if saved in the carry-over memory, there is no more need to modify the first part of the algorithm since it runs until again there is only one valid solution left.

There is one difference to the original algorithm of Fredman and Khachiyan that has to be made: we are not interested in any sort of counterexample with less than m or more than m elements in it. But, this results in not a big change of the basic algorithm. In fact, there are just some parts of the searching tree left out, i.e. those that would create counterexamples with more than m elements. Other parts are just stepped over, as there are those that would return counterexamples with less than m elements.

```
FUNCTION Hitting_Set_Star_Attack(m,counter)
observations = $\emptyset$
solutions = $\emptyset$
repeat
   # start main loop of hitting set attack
   repeat
      observations = observations $\cup$ New_Observation()
      # more than one solution as a carry-over? check those first
      if $|$ solutions $| > 1$ then
         solutions = $\{$ L $\in$ solutions $|$ Check_Validity(L) $\}$
      # less than 2 solutions in carry-over: start backtracking
      if $|$ solutions $| \leq 1$ then
         solutions = Backtracking($\emptyset$,observations,m,counter)
   # carry on, until only one solution remains
   until $|$ solutions $| = 1$
   # check result with algorithm of Fredman and Khachiyan
   solutions = Fredman_Khachiyan(observations,solutions,m)
# solution is correct if there was no counter example
until $|$ solutions $| = 1$
return solution
```

Most of the above code should be self-explanatory. The set `solutions` is not only used to return the result of the backtracking, but is also the carry-over memory.

The function `Fredman_Khachiyan` is called with both boolean formulas, the `observations` and the `solutions`, and the number of Alice's peer partners m.

It will return either just the unchanged set `solutions` in case it was verified, or it will add the counterexample into the set.

6.2 Validation

The new hitting set algorithm (i.e. HS*-attack) is **always correct**, like in the deterministic HS and disclosure attack. Hence, it avoids the most severe problem of the statistical attacks.

On the other hand the new attack is now not efficient any more in the means of a polynomial time algorithm. Testing series have shown that the running time of the algorithm is within reasonable bounds for $m \leq 20$ and $a \leq 50$, if $n = 20,000$. Note that this is enough for todays systems, i.e. those that are called to be of *practical size* (see section 2.2).

Interesting enough that in the intervals of the parameter in which this algorithm is feasible to compute, the average number of observations needed to detect Alice's peers is not much higher than those needed by the hitting set attack. A conservative guess is that it needs in systems of *practical size* about 5% to 10% more observations.

7 Conclusions

In this work we have suggested new attacks on anonymity systems using the hitting set algorithm. The pure HS-attack needs less observations than any other attack known to us. However, it requires the solution of a NP-complete problem, i.e. exponential run time in the worst case.

The SHS-attack finds the most frequent recipients of the observation and checks them if they fulfill the minimal hitting set property (instead of checking all possible sets). This attack solves the problem in polynomial time complexity. However, it is not exact, even if we can reduce the error to any arbitrarily measure.

Our third contribution is the combination of the SHS and the HS attack, the HS*-attack. We first search for a good candidate of the solution of the minimal hitting set problem and check this by using the work of Fredman and Khachiyan. HS*-attack has superpolynomial run time in the worst case.

References

[BFK01] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web Mixes: A System for Anonymous and Unobservable Internet Access. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies, (PET2001)*, pages 115–129. Springer-Verlag LNCS 2009, May 2001.

[CB95] D.A. Cooper and 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, 1995.

[CGKS95] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *36th IEEE Conference on the Foundations of Computer Science*, pages 41 – 50. IEEE Computer Society Press, 1995.

[Cha81] David L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84 – 88, Feb 1981.

[Cha88] David L. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, (1):65 – 75, 1988.

[Dan03] George Danezis. Statistical disclosure attacks: Traffic confirmation in open environments. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, *Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426, Athens, May 2003. IFIP TC11, Kluwer.

[FK96] Michael L. Fredman and Leonid Khachiyan. On the Complexity of Dualization of Monotone Disjunctive Normal Forms. *Journal of Algorithms*, (21):618 – 628, 1996. Article No. 0062.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[GRS96] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding Routing Information. *Information Hiding*, pages 137 – 150, 1996. Springer-Verlag LNCS 1174.

[GT96] Ceki Gülcü and Gene Tsudik. Mixing E-mail with Babel. In *Proceedings of the Network and Distributed Security Symposium - NDSS '96*, pages 2–16. IEEE, February 1996.

[KAP02] Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of Anonymity in Open Environments. In *Information Hiding, 5th International Workshop*. Springer Verlag, 2002.

[KAP03] Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Probabilistic Treatment of MIXes to Hamper Traffic Analysis. *IEEE Symposium on Security and Privacy*, 2003.

[KBS02] Dogan Kesdogan, Mark Borning, and Michael Schmeink. Unobservable Surfing on the World Wide Web: Is Private Information Retrieval an Alternative to the Mix Based Approach? *Privacy Enhancing Technologies (PET 2002)*, Springer-Verlag (LNCS 2482):224 – 238, 2002.

[KEB98] Dogan Kesdogan, J. Egner, and R. Büschkes. Stop-and-Go-Mixes Providing Anonymity in an Open System. In D. Aucsmith, editor, *Information Hiding 98 - Second International Workshop*, pages 83 – 98. Springer Verlag, 1998.

[Pfi90] A. Pfitzmann. Dienstintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz. IFB 234, Springer-Verlag, Heidelberg 1990, 1990. (in German).

[Pim03] Lexi Pimenidis. Structure and Analysis of Chaumian Mixes. Nov 2003. Master Thesis at the RWTH Aachen, Germany.

[PP90] B. Pfitzmann and A. Pfitzmann. How to break the direct rsa-implementation of mixes. pages 373 – 381. Eurocrypt '89, LNCS 434. Springer-Verlag, Berlin, 1990.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, pages 66 – 92, April 1998.