Using Linkability Information to Attack Mix-Based Anonymity Services *

Stefan Schiffner[1] and Sebastian Clauß[2]

[1] K.U.Leuven, ESAT/SCD/COSIC and IBBT
Kasteelpark Arenberg 10
B-3001 Leuven-Heverlee, Belgium
Stefan.Schiffner@esat.kuleuven.be
[2] Technische Universität Dresden
Institute of Systems Architecture
D-01062 Dresden, Germany
Sebastian.Clauss@tu-dresden.de

Abstract. There exist well established models for anonymity focusing on traffic analysis, i. e., analysing properties of single messages as, e. g., timing. However there is only little work done that use linkability information, that is information about the probability that two messages have been sent by the same sender.
In this paper we model information about linkability between messages as a weighted graph. We show lower and upper bounds with regards to the usefulness of linkability information for matching messages to senders. In addition to that we present simulation results, showing to which extent a matching of messages to senders is possible by using linkability information with different grades of noise.

1 Introduction

The number of applications and services on the Internet that enable or even require the user to create a user account increases rapidly. By offering user accounts, services try to achieve customer retention in a positive as well as in a negative sense. More precisely, providers are able to offer user-specific services, but they might also trace users, in order to place customized advertisements or even to deploy a discriminatory pricing model. Also, with regard to recent privacy scandals, service providers might aim for less personal data in their databases to avoid recourse receivables from customers in the case of data loss.

Privacy-enhancing identity management (see e. g. [1]) is being developed in order to protect users from overly greedy data collectors, but many services need

* This work was supported by the Integrated Projects IST-015964 AEOLUS on Algorithmic Principles for Building Efficient Overlay Computers and ICT-2007-216483 PrimeLife on Privacy and Identity Management in Europe for Life.The information in this document reflects only the authors' views, is provided as is, and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

a minimal amount of data to actually serve their customers. This leads to the problem that users actually do reveal personal data, which might be analyzed by the service provider. An identity management system needs to estimate how much a service provider can learn from a user's messages in order to assist the user in choosing the least privacy compromising data for a given purpose.

Recently several attempts have been made to define and formalize the notions of anonymity [2,3] and unlinkability [4,5]. Most of the models are only formulated for communication scenarios.

In this paper we show how information about linkability between messages (gathered, e. g., by a service provider from knowledge of the content of messages) can be used to reduce sender anonymity beyond what is possible by traffic analysis alone. We present a model where prior knowledge learned from network traffic can be integrated in a "layer-combining model". We simulate such a model and show analytical lower and upper bounds for the attacker's success rate, and we show under which conditions the attacker can breach the user's privacy.

In the next section we summarize related work on this topic. We describe our model in the section thereafter. In Sect. 4 we present our attack. Finally, in Sect. 5 we provide a conclusion on the results of this paper and briefly discuss issues open to further research.

2 Related Work

In this paper we discuss how noisy linkability information can be utilized to attack sender anonymity. Therefore, we focus on specifying a *connection* between information gathered by traffic analysis and linkability information gathered elsewhere.

First we need to model an anonymity system at the network layer, so that we can model the information an attacker obtains by observing this system. Over the past couple of years, much research has been done on aspects of anonymity with regards to network layer anonymity systems. Basic concepts of anonymity systems have been proposed [6,7] and enhanced in various ways. Systems, which proved to be practically usable on the Internet, e. g., Web mixes [8] or Tor [9] are based more or less on Chaum's Mixes [6]. Various attacks on such systems have been discussed, e. g. [10]. Since we focus on specifying a *connection* between anonymity properties on the network layer and linkability information gathered elsewhere, we do not emphasize a sophisticated traffic analysis model here. However, we want to keep close to well established models. Hence, the network-layer part of the model we describe in Sect. 3 is based on Chaum's Mixes [6].

Linkability aspects with regards to user profiles have been discussed not only in the course of privacy-enhancing identity management systems, e. g. [11,12], but as well with regards to statistical databases, e. g. [13,14]. In this paper we abstract from the derivation of linkability information. Similar to [4], we just assume that there *is* information about the fact whether pairs of messages have been sent by the same sender or not, which, e. g., might be derived from the contents of the messages sent over the network-layer anonymity system.

Recently, some aspects regarding the *connection* between information gathered from observing an anonymity system on the network layer and linkability information have been researched. In [11,12], Clauß and Schiffner focus on modelling knowledge gained from attributes of user profiles, but information gained by traffic analysis is not explicitly incorporated in this model. In [15], Díaz et al. calculate an example for combining information from network layer and application layer. Finally, in [16], Díaz et al. simulate a social network setting. In this setting, they calculate anonymity of users of the social network based on a combination of information gained by observations of the network layer and information gained from the known social network graph. From their simulations, they derive conclusions about relations between profiles, size of network etc. to anonymity of users. They especially focused on how much one can learn from these profiles depending on their quality, i. e., their expressiveness. In contrast to this work, we abstract from the source of profile information and model this information as a weighted graph where every node is a message and every edge is a score representing the probability that the two messages are from the same user or not.

Furthermore, attacks have been presented that gain from longterm traffic analysis, especially from evaluating the natural behavior of users with regards to leaving and joining the system. Such attacks are, e. g., intersection attacks, like the attack recently presented by Berthold et al. with regards to data retention [17], and the hitting-set attack [10] by Kesdogan and Pimenidis. In contrast to these we focus on a single round of a batch mix where we can assume that the user set stays the same during the whole attack.

3 Model Description

When Internet users communicate with service providers, they often reveal personal information. A service provider can use this information to build up user profiles, that is all kinds of data a service provider can collect about a user. Some of these profiles might be linkable with a certain probability, i. e., the service provider can guess that these profiles belong to the same user. In this section we first explain our model, which is later on formalized.

We assume a set of users who send their messages to a single service provider, while a batch mix is obfuscating the relation between senders and messages. The service provider is considered as the attacker, who wants to de-anonymize his users, i. e., he aims at a complete mapping of messages to users[3]. Naturally, he has access to the content of the messages. We further assume that he gains

[3] In our model, a user is an entity which can send messages. The attacker can distinguish users by observing senders on the network layer. With regards to information about linkability between messages on the application layer, we do not explicitly model users by their profiles. We just assume that there exists information about the fact whether pairs of messages have been sent by the same sender or not, which, e. g., might be derived from the contents of the messages sent over the network-layer anonymity system.

additional information by observing all links in the network, but he is not able to observe the mixing process of the messages. Furthermore the system is assumed to be *closed*, i. e., all messages are transmitted between nodes within the system and there are no messages sent/received to/from outside of the system. Fig. 1 illustrates our model.

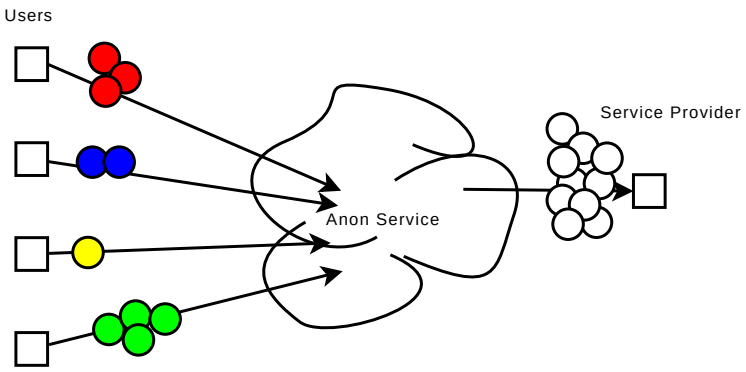

Fig. 1. The attacker's view on the system. Users (squares on the left) send messages (circles) to an anonymity service. The service provider on the right hand side receives these anonymized messages and may analyze the content of the messages.

In the following paragraphs we formalize our model. It contains a set of users $U = \{u_1, \ldots, u_m\}, m \in \mathbb{N}, m \geq 1$, and sets of messages $c_{u_1}, \ldots, c_{u_m}$, where c_u consists of the messages user u has sent. Furthermore, in our model we assume a perfect anonymizer that obfuscates the relation between senders and messages. Finally, we assume a service provider which receives all messages sent by the users.

With respect to the (anonymized) network layer, the service provider can observe the number of messages that a user u has sent, cardinality $|c_u|$. This is the type of information an attacker can learn from observing the network traffic of a perfect batch mix implementation, where all messages sent within the system form one single batch.

Clusterings and Number of Clusterings. With slight abuse of notation we will use clusterings of the set of messages to describe intermediate results of the attack, where a cluster c_i is not necessarily assigned to a user u_i, since the attacker can often assume that messages are from the same sender but not from which. A *clustering* is defined as follows:

Definition 1 (clustering). *A set of sets $C = \{c_1, \ldots, c_m\}, m \in \mathbb{N}$, is called* clustering *of a set S if and only if $S = \bigcup_{i=1}^{m} c_i$ and $\forall c_i, c_j$ with $i \neq j$: $c_i \cap c_j = \emptyset$.*

Informally speaking, that is, sorting all elements of a set in different classes, where every element can only be member of exactly one class.

For our aims, we are interested in the number of different clusterings of a set S, $n = |S| = \sum_{i=1}^{m} |c_i|$, under the condition that the clusters' cardinalities are given. This number can be calculated as follows:

$$\begin{aligned} & \binom{n}{|c_1|} \cdot \binom{n-|c_1|}{|c_2|} \cdot \ldots \cdot \binom{n-\sum_{i=1}^{n-1}|c_i|}{|c_m|} \\ = & \prod_{k=1}^{m} \binom{n-\sum_{i=1}^{k-1}|c_i|}{|c_k|} \\ = & \frac{n!}{\prod_{i=1}^{m}|c_i|!} \end{aligned} \tag{1}$$

Figuratively speaking, for the first (without loss of generality) partitioning we start by choosing the elements of the first cluster (c_1) from all n elements. For the second partitioning now only $n - |c_1|$ elements are left to choose from and so on.

Complexity. As one can see from (1), the number of possible clusterings with regard of the order of clusters becomes huge even for small examples. Without additional knowledge, each of these clusterings could represent the correct system state. Thus, even if we can in principle calculate the probability of a state, it is extremely time-consuming, and therefore practically not feasible, to iterate over all states to find the most likely. Therefore, in Sect. 4 we present a simulation that uses simulated annealing in order to find a good, i. e., a likely system state.

The Random Attacker (Lower Bound). The random attacker is an attacker that randomly maps messages to senders, but only takes the known cluster sizes into account. Given a set of messages $S = \{s_1, \ldots, s_n\}$, a clustering $C = \{c_1, \ldots, c_m\}$ of S, a hidden function $f : S \mapsto C$ that maps every message to its actual cluster, and $f' : S \mapsto C$ which describes the random guess of the attacker for f, we can calculate the expected number of messages the attacker guesses correctly. Given an urn filled with coloured balls where the number of balls of each colour is known, then the number of balls of a certain colour in a snap sample of a given size follows a multivariante hypergeometric distribution. If all messages of the same sender are seen as balls of the same colour, the number of messages allocated to a certain cluster c_i, that is messages that are actually sent by user u_i in a snap sample follows this distribution. The mean of the number of messages belonging to cluster c_i in a sample a of size $|a|$ is then $\frac{|a|*|c_i|}{n}$.

Without loss of generality, we assume that the attacker first draws a sample of size $|c_1|$, then $|c_2|$ and so on in order to construct f'. The expected number of correctly guessed mappings for c_1, i. e., the mean of the number of guessed messages which really belong to c_1, is thus $\frac{|c_1|*|c_1|}{n}$. For the second sample, the choice has narrowed down to $|c_2|$ out of $n-|c_1|$ and we have to take into account that on average $\frac{|c_1|*|c_2|}{n}$ messages of cluster c_2 are mapped to c_1 in f'. That is, the expected number of correctly mapped messages in f' for c_2 is $\frac{|c_2|*(|c_2|-\frac{|c_1|*|c_2|}{n})}{n-|c_1|}$. Analogous, the number of correctly guessed messages can be described for c_3 to c_n.

More generally, the expected number of correctly guessed messages in f' for c_i is $E(c_i) = \frac{|c_i| * \left(|c_i| - \frac{\left(\sum_{j=1}^{i-1} |c_j|\right) * c_i}{n} \right)}{n - \left(\sum_{j=1}^{i-1} |c_j|\right) * |c_i|}$. By factoring $|c_i|$ out of the numerator and n out of the denominator we derive $E(c_i) = \frac{|c_i|^2}{n}$.

The sum of all $E(c_i)$ is the expected number of correctly guessed messages for a random attacker:

$$E_{\text{Random}} = \sum_{(C)} \frac{|c_i|^2}{n}$$

Beyond these network layer observations, the service provider can analyze the message content in order to "link" messages, i. e., to estimate whether pairs of messages have been sent by the same user or not. Without such content analysis, any clustering with the correct cardinalities is possible and equally likely.

The Perfect Attacker (Upper Bound). A perfect attacker, that is an attacker that knows exactly which messages are from the *same* sender, might even not be able to map all messages to the *right* sender. Even though he has a perfect clustering, he can not distinguish between two clusters of the same size and thus he can only randomly map the clusters of the same size to the senders that sent the corresponding number of messages.

The multiplicity of a cluster size is the number of clusters of this size. Formally, given the multiplicity mult_i of the size of cluster $|c_i|$ and the cluster sizes $|c_i|$, we can calculate the expected number of correctly assigned messages of a perfect attacker E_{perf}.

$$E_{\text{perf}} = \sum_{(C)} \frac{|c_i|}{\text{mult}_i}$$

Soundness of Lower and Upper Bound. Since the random attacker should never be more successful than the perfect attacker, we need to show that the lower bound is always smaller or equal to the upper bound. The multiplicity mult_i of the cluster size c_i is always smaller or equal to $\frac{n}{|c_i|}$, since the sum of all cluster sizes is n. Thus, $E_{\text{Random}} = \sum_{(C)} \frac{|c_i|^2}{n} = \sum_{(C)} \frac{|c_i|}{n/|c_i|} \leq \sum_{(C)} \frac{|c_i|}{\text{mult}_i}$. Since equality holds only for $\text{mult}_i = \frac{n}{|c_i|}$, a random attacker can achieve as much as a perfect attacker iff all clusters are of equal size. Otherwise, he is less successful. Note that the perfect attacker indeed has the choice between less states than the random attacker, since he will never assign messages sent from one single sender to different senders. Nevertheless, in case of equal cluster sizes the perfect attacker either assigns all messages sent by a given user correctly to this user, or he assigns all messages of this user to another user. This leads to the same expected number of correctly assigned messages as for a random attacker, even though the number of possible states is much smaller for the perfect attacker.

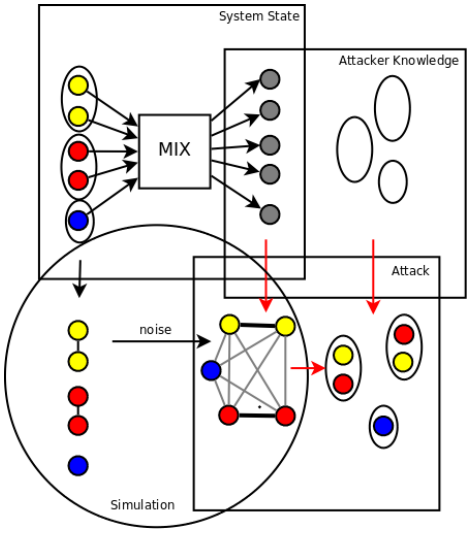

Fig. 2. Overview of the experimental setting (gray arrows: simulation, black arrows: attacker's behavior)

4 Simulation and Results

In this section we present our simulation method and how we model the behavior of the attacker. In Fig. 2 our experimental setting is sketched. From the knowledge gained from observation of the network layer, the attacker derives the cluster sizes. Furthermore, by analyzing the message content, he derives a weighted graph that represents the knowledge about which messages were probably sent by the same sender. Since we want to abstract from the concrete process of gaining this knowledge by content analysis, we run the attack with the original graph plus noise (circle). The following pseudocode shows the steps for one simulation round.

```
SystemState sys = generate(message number n, cluster sizes
min,max,dist)
SystemState noisy = addNoise(s,d)
Clusters c = cluster(noisy)
for SystemState i = allPossibleStates(c) do
  compare i and sys
end for
```

Initialization. The system state (see Fig. 2) is a random mapping of messages to senders with a given total number of messages n, given minimal and maximal number of messages per sender. Two senders might have sent either the same number of messages, or the number of messages differs in at least a given distance dist. Furthermore the messages are organized in a graph where a message s_i is

connected to another message s_j with a weight 1 iff the two messages have been sent by the same sender, otherwise with the weight 0.[4]

In the next step we add noise in order to model the uncertainty of the attacker. Therefore, for every edge of the graph a random number r is chosen from a zero-mean Gaussian distribution. If the edge's weight was zero, it is replaced by r. If the edge's weight was 1, is is replaced by $r + d$, where d is a *noise distance*[5]. Hence the resulting distribution for the former zero-weighted edges becomes a zero-mean Gaussian distribution. For the former one-weighted edges the resulting distribution becomes a Gaussian distribution with its mean at the noise distance.

Clustering. In this step an optimal clustering is needed, i. e., a clustering where messages that are strongly connected in the graph are assigned to the same clusters. As fitness function we use the average fitness of all clusters, where the fitness of a cluster is the sum of all edges within the cluster. In order to cluster the graph we use *simulated annealing* [18], since we can guess a good starting solution and the change of the fitness function is fast calculable. Furthermore the algorithm is easy to adapt to fixed cluster sizes. The following algorithm sketches simulated annealing. Note that $c_i[s_j/s_k]$ denotes that within cluster c_i message s_j is replaced by s_k.

```
Clustering c choseStartSolution(graph)
temp = startTemp
repeat
  time = maxTime
  repeat
    chose 2 different clusters c_1 and c_2
    chose s_1 from c_1, and s_2 from c_2
    if fit(c_1, c_2) < fit(c_1[s_1/s_2], c_2[s_2/s_1]) then
      c_1 ⇐ c_1[s_1/s_2]
      c_2 ⇐ c_2[s_2/s_1]
    else
      if temp < rnd(temp) then
        c_1 ⇐ c_1[s_1/s_2]
        c_2 ⇐ c_2[s_2/s_1]
      else
        s = s − 1
      end if
    end if
  until time == 0
  temp = temp − 1
until temp == 0
```

[4] Thereby, the weight can be interpreted as an (inverse) distance measure.

[5] The higher the value of the noise distance, the better it is possible to distinguish between former zero-weighted and one-weighted edges.

The general idea of simulated annealing is that it starts with a guessed solution, then randomly picks two elements and swaps these two. If the new solution is better than the old one, it repeats the loop with the newly found solution. Otherwise it continues with a certain probability with either the old or the new solution. The probability that it continues with a worse solution decreases over the running time and depends on how much the average fitness decreases by using the worse solution.

For our problem, we search for the clustering where all messages that are in the same cluster have been sent by the same sender. Edges between messages from the same sender have more likely a higher weight than others, thus the average of the sums of the edges' weights between messages in the same cluster should be maximal for the clustering where all messages from the same sender are in the same cluster.

In order to calculate the change of quality of the solution in one optimization step it is sufficient to calculate the fitness of each of the two clusters that are chosen in that step. Thereby, a cluster's fitness is the sum of all edges among the messages within the cluster. If after swapping the sum of these two fitnesses is higher than before, then the average fitness over all clusters increases as well, thus the new solution is better than the old one. Otherwise, the old solution was better.

In order to speed up the clustering we also take into account that the cluster size is proportional to the degree of the nodes, which should be in this cluster. Thereby, the degree of a node is the sum of the weights of all edges of this node. We deploy this in two ways. At the beginning of the simulation, we need to guess a first solution. This is done by putting higher degree messages in larger clusters. Furthermore the fitness function is adapted in a way that the quality of the solution is lower if messages are in clusters that are of a very different size than their degree would let expect. This prevents that messages that are actually members of small clusters are grouped in large ones, since this would lead to high local maxima, that is a solution to which the algorithm is likely to converge to, although it is not globally optimal.

However, if clusters are of similar size it still might happen that messages end up in cluster of the wrong size as Fig. 3 illustrates. Since with simulated annealing it is impossible to estimate the quality of a final solution, the algorithm is then very likely to end up in the wrong maxima. In the following paragraph we describe what the attacker does with this solution and which effect wrong clustered messages have on the result of the attack.

System States and Success Rate. The attacker uses this optimal clustering to enumerate all still possible system states. How many that are depends on the number of clusters of equal size, since two states where clusters of the same size are mapped to different senders are indistinguishable for the attacker (cp. Sect. 3). In order to determine the quality of the attack we compare every possible system state with the original state and count the number of correctly allocated messages. The average number of correctly allocated messages in relation to the

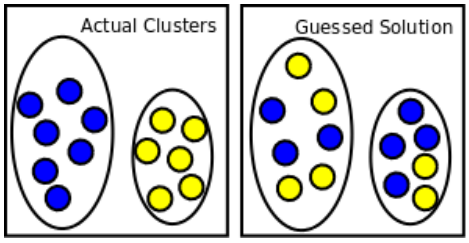

Fig. 3. Guessed solution with messages mapped to clusters of the wrong size

total number of messages in the system is the success rate of the attacker in the given experiment.

Since we average over all possible system states the success rate goes never over its upper bound. However it might fall below the expectation since if the clustering is in fact not the right clustering (because of noise) it might make the correct state impossible as illustrated in Fig. 3. Assume that most of the messages of cluster c_i are in cluster c_j with $|c_i| \neq |c_j|$. If now a message pair from these two clusters, which is actually correctly assigned to these clusters, is chosen to be switched, the resulting fitness of the solution will be considered better by the algorithm, since then both messages are among more messages from the same sender.

Simulation Results

In this section we present simulation results that illustrate how an attacker could use knowledge about the linkability of messages.

In Fig. 4, a typical result of our attack is shown. 100 messages were sent by 11 senders. On the x-axis the distance between the two Gaussian distributions which were used to add noise is displayed, while on the y-axis the (min, max and average) success rate is displayed. For this example, a random attacker would have a success rate of about 0.1. Note that our attack is already for very small noise distances, i. e., below 1, slightly better. However, for higher noise distances our simulation reaches the theoretical upper bound (cp. Sect. 3) of 0.81. Furthermore, one can see that the errors are quite large. This is because the noise affects also the local maxima, which might become global maxima by analogous reasons as shown in Fig. 3 in the section before.

In cases where every sender sent a different number of messages our attack can totally deanonymize the systems' users as shown in Fig. 5. Furthermore, one can see that with rising distance between the weights for messages belonging to the same cluster and messages not belonging into the same cluster also the errors start to diminish and therefore the average converges to the actual optimal solution.

In Fig. 6 one can see that the larger the distances between the number of messages different senders have sent are, the faster our attack converges to its individual maximum. Thereby the red pluses represent results from a system

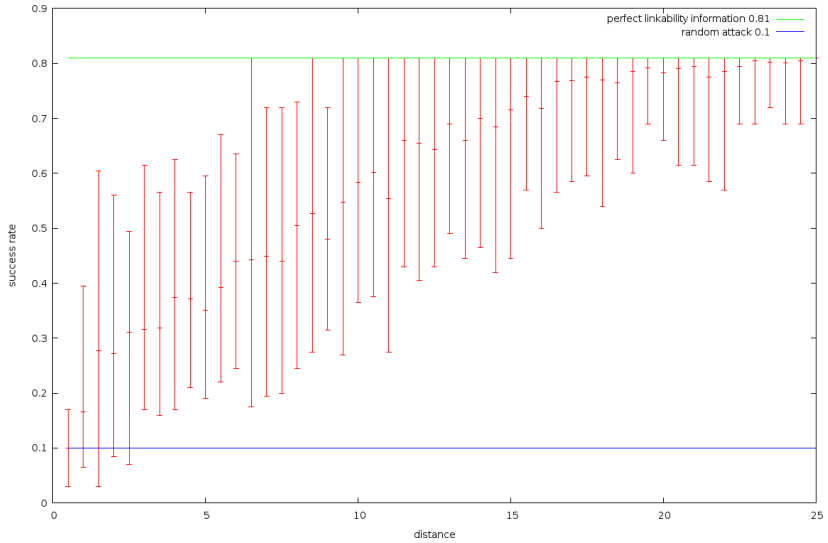

Fig. 4. Simulation results for 100 messages, distribution of cardinalities $|c_i|$: [4,5,7,8,9,9,10,10,11,12,15]. For each noise distance displayed, 25 experiments have been made. For each noise distance, the minimum, maximum and average success rate is displayed.

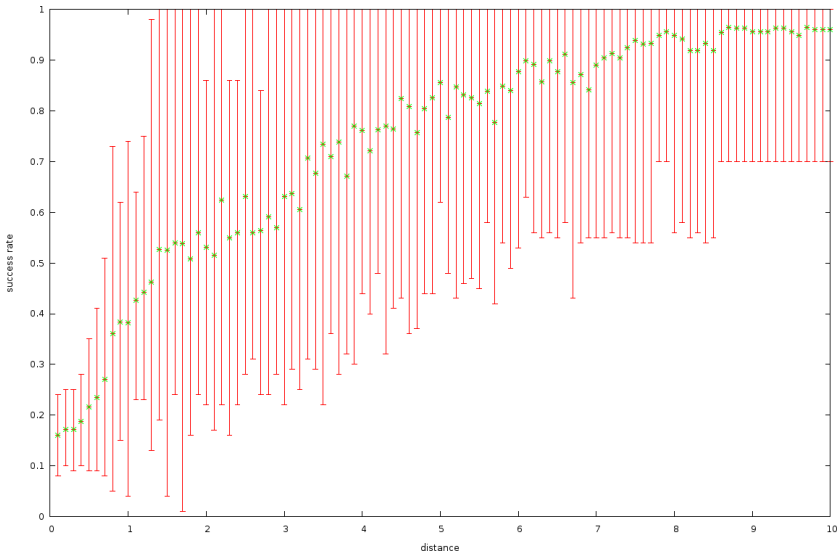

Fig. 5. Simulation results for 100 messages, distribution of cardinalities $|c_i|$: [4,7,8,12,14,15,18,22], i. e. all clusters have different sizes. In this case we reach total deanonymization.

where the number of messages sent by each two senders is either equal or differs by at least 5. As we can see, already for small noise distances our simulation reaches its maximum. In contrast to that, the blue stars represent results from a system where the number of messages per sender is much closer to each other. Hence, noise has much more influence on the simulation results since already a small change of the degree of a message might lead to a different clustering.

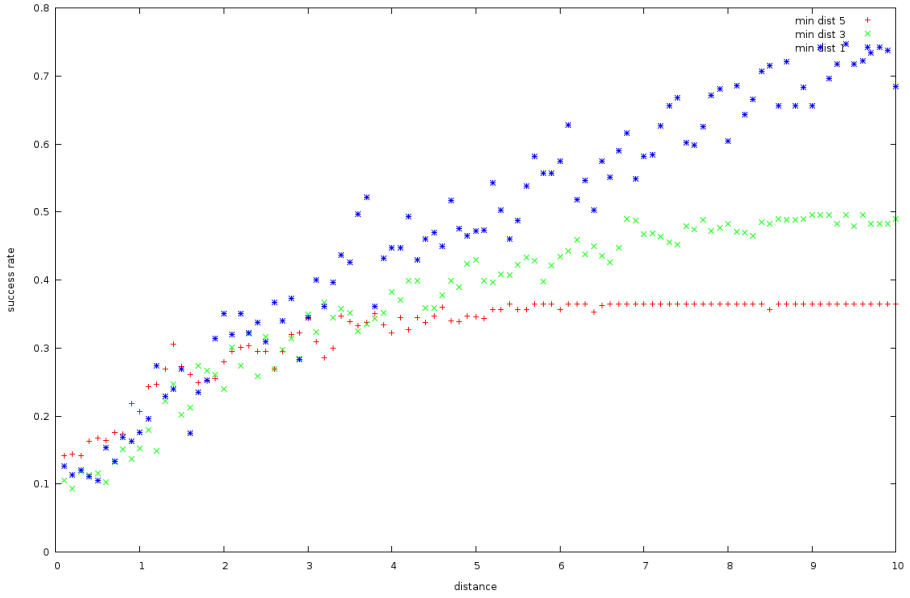

Fig. 6. Faster convergence with larger distances between clusters. Red pluses: each two senders sent either the same number of messages or the number of messages differs by at least 5. Green crosses: same number of messages or at least 3 messages difference for each two senders. Blue stars: same number of messages or at least 1 message difference for each two senders

5 Conclusion

In this paper we show how information about linkability between messages (gathered, e. g., by a service provider from knowledge of the content of messages) can be used to reduce sender anonymity beyond what is possible by traffic analysis alone. Therefore, we present a model which integrates information gathered from the network layer with information about linkability between messages.

In order to show the usefulness of incorporating such linkability information for deanonymizing users, we present an appropriate attack using both infor-

mation from observing the network and linkability information. Thereby, we consider an abstract service provider that receives all messages. This models many realistic attackers, such as coalitions of service providers or attackers that observe the exit node of a mix cascade.[6] However, even if the service provider does not receive all messages he can assume for every unknown message that it was sent equally likely by the same user as any other message, that is, it has an edge to any other message with the same wight. This would introduce the more noise, the more unknown messages are introduced, i.e., the less observations the attacker had made. The hidden assumption thereby is that users which heavily use the attacker's service also use the other services often.

The information we consider from the network layer consists of the number of messages sent by different senders. From the application layer we regard information about linkability between messages. We show upper and lower bounds for the success rate of the attack. We simulate our attack in order to show that messages belonging to the same sender are grouped together even in case of rather noisy linkability information. Further, the attack is the more successful in assigning messages to actual senders the more different amounts of messages the senders have sent.

However, in further research we will deal with better clustering algorithms especially with regards to proven quality bounds (e. g. branch and bound). Furthermore we will extend our model to more comprehensive network as well as application layer models. With regards to network layer models we will combine our attack with more sophisticated traffic analysis attacks.

We expect that our attack is generalizable to pool mixes, since the incoming message stream can be used to count the number of messages sent by the users. Furthermore the attacker can exploit the expected delay of the messages (which depends on the pool size) to determine which part of the outgoing message stream should match the ingoing message stream.

Another interesting field of research will be to show how we can use our model to directly derive sender anonymity measures in terms of Shannon entropy as metric, that is calculating the probability distribution that a given user has sent a given message.

References

1. Clauß, S., Pfitzmann, A., Hansen, M., Van Herreweghen, E.: Privacy-enhancing identity management. The IPTS Report **Special Issue: Identity and Privacy** (2002) 8–16
2. Serjantov, A., Danezis, G.: Towards an information theoretic metric for anonymity. In Dingledine, R., Syverson, P., eds.: Proceedings of Privacy Enhancing Technologies Workshop (PET 2002). Number 2482 in LNCS, Springer-Verlag (April 2002) 41–53

[6] Note, that most of the traffic is sent to service providers that do not support end to end encryption.

3. Díaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In Dingledine, R., Syverson, P., eds.: Proceedings of Privacy Enhancing Technologies Workshop (PET 2002), Springer-Verlag, LNCS 2482 (April 2002)
4. Steinbrecher, S., Köpsell, S.: Modelling unlinkability. In Dingledine, R., ed.: Proceedings of Privacy Enhancing Technologies workshop (PET 2003). Number 2760 in LNCS, Springer-Verlag (March 2003) 32–47
5. Franz, M., Meyer, B., Pashalidis, A.: Attacking unlinkability: The importance of context. In Borosov, N., Golle, P., eds.: Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007), Ottawa, Canada, Springer (June 2007)
6. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM **24**(2) (February 1981) 84–88
7. Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology **1** (1988) 65–75
8. Berthold, O., Federrath, H., Köpsell, S.: Web MIXes: A system for anonymous and unobservable Internet access. In Federrath, H., ed.: Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, Springer-Verlag, LNCS 2009 (July 2000) 115–129
9. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium. (August 2004)
10. Kesdogan, D., Pimenidis, L.: The hitting set attack on anonymity protocols. In: Proceedings of 6th Information Hiding Workshop (IH 2004). LNCS, Toronto (May 2004)
11. Clauß, S.: A framework for quantification of linkability within a privacy-enhancing identity management system. In Müller, G., ed.: Emerging Trends in Information and Communication Security (ETRICS). Volume 3995 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer (2006) 191–205
12. Clauß, S., Schiffner, S.: Structuring anonymity metrics. In Goto, A., ed.: DIM '06, Proceedings of the 2006 ACM Workshop on Digital Identity Management, Fairfax, Virgina, USA, ACM (November 2006) 55–62
13. Sweeney, L.: Guaranteeing anonymity when sharing medical data, the datafly system. Journal of the American Medical Informatics Association (1997) Washington, DC: Hanley & Belfus, Inc.
14. Fischer-Hübner, S.: IT-security and privacy: Design and use of privacy-enhancing security mechanisms. Volume 1958 of Lecture Notes in Computer Science. Springer (2001)
15. Díaz, C., Troncoso, C., Danezis, G.: Does additional information always reduce anonymity? In Yu, T., ed.: Proceedings of the Workshop on Privacy in the Electronic Society 2007, Alexandria,VA,USA, ACM (2007) 72–75
16. Díaz, C., Troncoso, C., Serjantov, A.: On the impact of social network profiling on anonymity. In Borisov, N., Goldberg, I., eds.: Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008), Leuven, Belgium, Springer (July 2008) 44–62
17. Berthold, S., Böhme, R., Köpsell, S.: Data retention and anonymity services – introducing a new class of realistic adversary models. In Švenda, P., ed.: The Future of Identity in the Information Society – Challenges for Privacy and Security, Springer Verlag (2008) To appear.
18. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science (1983)