Evaluation of Anonymity Providing Techniques using Queuing Theory

Dogan Kesdogan

IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598
E-Mail: kesdogan@us.ibm.com

Abstract

In our work we use queuing theory both for security (i.e. anonymity) and performance analysis. A well-known anonymity technique, the MIX method, which is the basis of most of today's deployments, is the object of our investigation. We show shortcomings and problems in the MIX method and suggest possible workarounds. Our investigation reveals the level of security of MIX based systems and their performance characteristics on the Internet.

1. Introduction

Leonard Kleinrock starts his famous book "Queuing Systems" [15] with the words "How much did you waste waiting in line this week?" With this question, he expresses the aim of queuing theory: reducing waiting times. However, this is not true for all circumstances. For instance, in the anonymity area a short waiting time can be the first indication of an attack. In this work, we present this problem and analyze known techniques with the aid of queuing theory. We also discuss alternative techniques which guarantee a given delay time.

In the next chapter we present the most popular anonymity technique, the MIX method, and related works. We analyze the direct implementation of the MIX concept with the aid of queuing theory and show the importance of time. We present two variants, which delay messages independent of traffic, and we evaluate their security properties. We conclude our work with performance analysis.

2. Anonymity and the MIX Method

The challenge for anonymity-providing techniques is to accomplish their basic goal even if:

a) The underlying communication network is global and is not subject to any topology restrictions; and

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

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 the need to read the content of the packets.

Hence, the decisive point of anonymity techniques is to organize additional traffic in order to confuse the adversary and conceal the particular communication relationship. The sender and/or receiver of a message must be embedded in a so-called *anonymity set*.

The main questions related to an anonymity set are:

1. How is the anonymity set established?
2. What is the size of the anonymity set?

2.1 Related Works

Anonymity (i.e. privacy) has become a hot topic on the Internet, as illustrated by several recent works and deployments [4, 7, 11, 24]. However, anonymity is not a new topic. The first works, known to us, are [2, 3, 6, 18, 22, 23]. The main aim of these theoretical works is to provide full or even perfect security. Of the suggested techniques, the MIX concept is the most often investigated. Centralized MIX stations handle the bookkeeping associated with the anonymity sets, and provide flexible access to an anonymity service and can therefore serve a large number of users. Other known anonymity techniques require the end user to build his own anonymity set (see e.g. DC-Network [3]). This requirement can severely limit flexibility for the users, since anonymity-building software has to run on the user's computer and the members of the anonymity set have to be known beforehand (see [12, 13]).

Unfortunately, the MIX concept is not directly applicable to the Internet [13] and therefore most new works and deployments use a modified MIX (e.g. [1], [5], [8], [9], [20], [24] and distantly related [21]). None of them can provide the same security as the classical MIX method but they are practical.

0-7695-1321-2/01 $10.00 © 2001 IEEE

They do not build an anonymity set (e.g. [8, 21]) and thus assume that the eavesdropper cannot listened to all the lines between the intermediary nodes (in particular the attack by E described above can be applied here).

Some of these approaches (e.g. [1], [9], [20]) build an anonymity set, but this set is not protected against attacks: an attacker can contribute several times to the same anonymity set. Note, that if the anonymity size is n and the attacker can contribute *(n-1)* of members, then the remaining one is of course observable (compare it with [19]). In the security area, wiretapping (passive attack) and sending messages to an open network are not considered to be strong attacks (compare this also with the cryptography area). We analyze the MIX concept and the MIXmaster variant against this attack, since the MIXmaster is known in the literature as the strongest implementation of the classical MIX concept [16]. We do not analyze all other variants (e.g. [1, 8, 16, 20, 21]) since some of them provide less or same security and others are less practical (see [12]).

2.2 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 message to an output message [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 with 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. A sequence of MIXes is used to increase the reliability of the system.

MIX protocol: A MIX collects n 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 former received packets (management: store in a local database) in order to reject any duplicates. Every MIX (except the first) must include an anonymous *loop back*[1], because only the first MIX can decide whether or not the packets are from distinct senders.

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

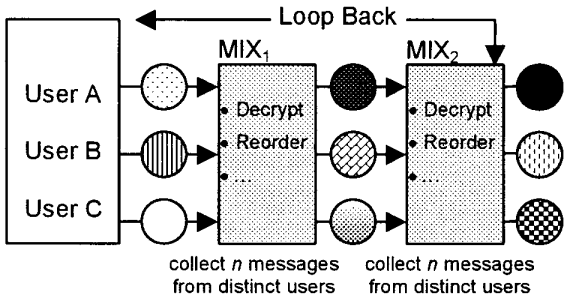

Fig. 1: Cascade of two mixes

Applying this protocol, 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 message passes.

b) The *(n-1)* other senders do not all cooperate with the attacker.

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

2.3 The (n-1) Attack

Before a MIX can forward a packet, it has to collect n messages from different users. This *group function* ensures that each packet is from a distinct user. However, a suggestion in the context of the Internet scenario cannot assume this functionality, mainly because it can only be ensured securely if a **P**ublic **K**ey **I**nfrastructure (PKI) exists. Since PKI is not generally available to the public, all implementations without this functionality are insecure (see for a suggestion of group function assuming PKI using blind signatures [1]). If a MIX cannot decide whether the packets are from different senders, the attacker can intercept the incoming packets, isolate each packet, and forward it together with *(n-1)* of his own packets. This is known as a *trickle attack* [9]). All MIX variants like MIXmaster [5] are insecure against this attack. One vulnerability of these variants is that the time until n messages are collected by a MIX and hence the end-to-end delay of a message is not known (asynchronous communication model). Therefore it is possible to delay a single message without any risk of

[1] 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 message 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.

detection. There is no solution for this problem (known to us), if the goal is complete (i.e. perfect) security is aimed [19, 12].

Furthermore, in this work we show that these MIXes and MIX variants are also insecure against a weaker attacker model $\underline{E}$: $\underline{E}$ is not able to intercept incoming packets from distinct users, but he is able to eavesdrop on the communication wire and to send his own messages with a constant rate λ_A.

The concrete attack scenario is as follows: $\underline{E}$ listens to the input and output lines of the one MIX station assumed to be honest, and is able to send λ_A of his own messages. One particularly feature of MIX enables this attack:

Deterministic output behavior: Sending messages to the MIX at a high rate results in a high output rate from the MIX.

In the next sections, we will analyze this disclosing attack.

2.4 Measuring the Success of Disclosing Attack on the MIX

In order to determine the probability of a successful attack, we assume that real messages arrive at the MIX according to a Poisson process with rate λ.

As before, let there be an attacker $\underline{E}$ sending messages to the MIX at the rate λ_A. The MIX outputs a batch when it has received n messages. If the attacker starts sending n-1 messages immediately after a batch has been processed, the attack is successful if, during the time he needs to send those messages, not more than one real message arrives at the MIX. The length of this interval is $t=(n-1)/\lambda_A$ and the number of real messages arriving during it is Poisson n distributed, so the probability that the attack was successful is [14]

$$\exp(-\lambda t)+\lambda t\exp(-\lambda t) = \left(1+(n-1)\frac{\lambda}{\lambda_A}\right)\exp\left(-(n-1)\frac{\lambda}{\lambda_A}\right)$$

To calculate the probability that an arbitrary message can be attacked successfully, we have to divide this probability by the mean number of real messages arriving within one interval, this is the expectation for the Poisson distribution λt plus a correction for the case that no real message arrives during the interval. Then, the MIX waits for the next arriving message, and the interval is extended until the message arrives. This case happens with probability $\exp(-\lambda t)$, so the mean number of arrivals is $\lambda t + \exp(-\lambda t)$ and the probability of a successful attack on an arbitrary message is (see Figure 2):

$$\frac{\exp(-\lambda t)+\lambda t\exp(-\lambda t)}{\lambda t+\exp(-\lambda t)}$$

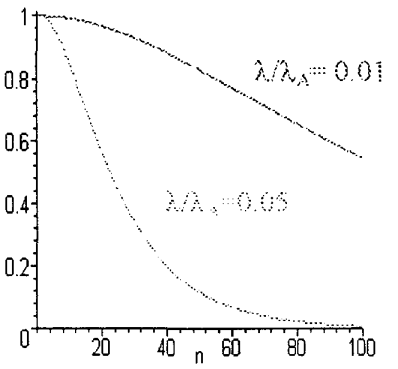

Figure 2: Probability of success for an attack on a message passing through a MIX with constant batch size n

2.5 Extending Mix to MIXmaster and the Success of Disclosing Attack

The MIXmaster for email by Cottrell [5] was the first real application in the Internet, which used the MIX concept. It includes the following parts of the MIX concept: same function for change of appearance (for encryption RSA (1024 Bit) and TDES (168 Bit) are used), replay detection by evaluating a package identity (32 Bit random number), uniform sizing. Instead of batching and reordering messages, a so-called *pool-mode* is used. (Strictly speaking there are two processing modes, but we will only analyze the pool-mode, since the security analysis is similar for both modes). The MIXmaster in pool-mode collects n packets. When packet $(n+1)$ arrives, the MIXmaster chooses one of the $(n+1)$ packets at random, and forwards it to its destination address. It is not possible to predict how long a message might be kept in the pool (nor is it known for the sender).

A successful attack consists of three steps:

i. Attacker fills up the pool with his own messages, sending at rate λ_A [2].

ii. A "real" message arrives.

iii. Attacker sends messages at the rate λ_A, until the MIXmaster forwards the real message.

To calculate the probability for step i, we assume that the attacker always sends at rate λ_A and that the system is in the steady state. At a given arbitrary time the probability

[2] The attacker knows the number of his messages in the pool, since he observes all incoming and outgoing messages.

for each message in the pool to be a genuine message is $\lambda/(\lambda+\lambda_A)$. Thus the probability that the attacker will successfully fill the pool is:

$$p_F = \left(1 - \frac{\lambda}{\lambda+\lambda_A}\right)^n = \left(\frac{\lambda_A}{\lambda+\lambda_A}\right)^n$$

Applying the PASTA-Property [10] this probability is always correct.

Now, we have to calculate the probability of forwarding the genuine message after m steps. Since a message in the MIXmaster is chosen with the probability $1/(n+1)$, the probability that the genuine message will be chosen in the m^{th} step is (if no other genuine message arrives in this time):

$$P(X=m) = \frac{1}{n+1} \cdot \left(1 - \frac{1}{n+1}\right)^m = \frac{n^m}{(n+1)^{m+1}}$$

The length of this interval is $t=m/\lambda_A$ and the probability that no other genuine message will arrive is $\exp(-\lambda t)$. Bringing all parts together and using the theorem of total probability the attack success is (see figure 3):

$$P(success) = p_F \cdot \sum_{m=0}^{\infty} \exp\left(-\frac{m\lambda}{\lambda_A}\right) \frac{n^m}{(n+1)^{m+1}}$$

$$= p_F \cdot \frac{1}{n+1} \sum_{m=0}^{\infty} \left(\frac{n\exp(-\lambda/\lambda_A)}{n+1}\right)^m$$

$$= \frac{(\lambda_A/(\lambda+\lambda_A))^n}{n+1-n\exp(-\lambda/\lambda_A)}$$

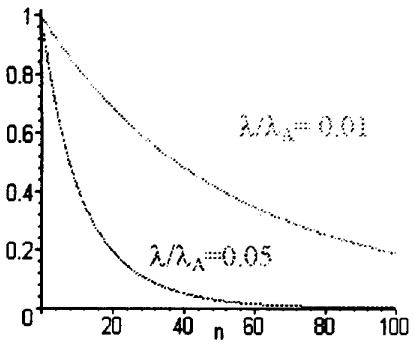

Figure 3: Probability of success for an attack on a message passing through a MIXmaster with constant pool size n

3. MIXes capable of Delaying

MIXes capable of delaying messages independent of traffic avoid the direct dependence on the packet arrival rate and are therefore not vulnerable to the attacks shown in the previous section. Such MIXes change the appearance of messages the same way as the classical MIX nodes but do not collect a fixed number of messages. We present and analyze two possible realizations T-MIX (Time-MIX) and SG-MIX (Stop-and-Go MIX) [12].

3.1 T-MIX

A T-MIX operates with a fixed time interval of length T between the outputs of two batches. The probability of a message being successfully attacked by an eavesdropper is simply the probability that no other message arrives in the same interval. The number of other messages arriving in that interval is Poisson distributed and therefore equals $\exp(-\lambda T)$.

Note that because messages from an active attacker do not influence the behavior of the T-MIX at all, a disclosing attacker has no advantage over a passive eavesdropper. We have seen that a T-MIX is not absolutely secure against even passive attacks. But, if we choose the interval length T so that $\lambda T = 50$, the probability that any message can be tracked from a passive and disclosing attacker is equally negligible. We could do even more and include a few dummy messages generated by the T-MIX in each batch, but we think small amount of additional obscurity gained is not worth the trouble. Server-generated dummies can only hide the true recipient, in any case, and are of no use if the recipient of a message cooperates with the attacker.

To use T-MIXes an appropriate time interval T must be chosen. If every participant selects the intermediary nodes to use independently and with equal probability, then the arrival rate λ will be nearly equal at every node. This value of the arrival rate should be digitally signed and published regularly by every node. These values can be collected and averaged by the users or T-MIX nodes. The mean value of λ can then be used to calculate the values of T accordingly. In practice, the arrival process is not time-homogeneous, and the fluctuations of λ should be taken into account.

3.2 SG-MIX

T-Mixes provide enough security against the disclosing attacker $\underline{E}$. What is needed is a technique that provides security against E without requiring identity verification. SG-MiXes provide such security and we will

next sketch the level of security that can be guaranteed with this technique without identity verification.

The SG-MIX [12, 13] operates in the same way as a classical MIX, but does not collect a fixed number of messages. A sender selects the SG-MIXes to be used from those available with equal probability. He calculates for every node *i* a time window (TS_{min}, TS_{max}) and draws a random delay time *T* from an exponential distribution with suitable parameter μ. This information is appended to the packet before encrypting it with the SG-MIX's public key. The SG-MIX *i* extracts $(TS_{min}, TS_{max})_i$ and T_i after decryption. If the arrival time of the packet is earlier or later than given by the time window the message will be discarded. After T_i units of time have elapsed, the SG-MIX *i* forwards the packet to the next hop or to its final destination.

The security of the SG-MIX does not rely on shuffling a batch of messages but rather on delaying each message individually and independently by a random amount of time. If the delay times are individually drawn from the same exponential distribution, the knowledge of the time a specific message arrived at the SG-MIX node does not help the attacker to identify the corresponding outgoing message as long as there is at least one other message in the queue at some time during the delay. Because of the memoryless property of the exponential distribution, if *n* messages are in the queue, it is equally probable for any one of them to depart next, regardless of their arrival times. Therefore, an attacker can correlate arrival and departure of a message only if no other message is in the queue during the whole delay time.

The resulting probability that an arbitrary message can be tracked by an eavesdropper is given by (see for detailed analysis [12, 13])

$$P(success) = \frac{\exp(-\lambda/\mu)}{1+\lambda/\mu}$$

with λ denoting the rate of message arrivals.

Let us consider an example: Assume a SG-MIX node with a mean arrival rate $\lambda = 10$ packets/s and parameter $\mu = 0.2$ packets/s, which implies a mean delay of 5 seconds. Then the probability of an arriving packet finding the server empty is $\exp(-50) \approx 1.9\ 10^{-22}$.

In order to provide probabilistic anonymity against *E* SG-MIX must be able to fend off delaying attacks. When running such an attack the intruder must delay all incoming data packets for a certain amount of time in order to "flush" the SG-MIX. Therefore we introduce the time stamps (TS_{min}, TS_{max}) to detect the delay of an incoming data packet and discard it. This prevents blocking attacks. The SG-MIX technique allows the calculation of the time windows very accurately as the user knows in advance the time a message will be delayed.

We define for the pair of time stamps of node i (TS_{min}, $TS_{max})_i$ the time window Δt_i during which a packet must arrive at SG-MIX i. If a Δt value is given, then the success probability of a blocking attack is (see for detailed analysis [12, 13]):

$$P(success) = \exp\left(\frac{-\lambda \exp(-\mu \Delta t)}{\mu}\right).$$

Obviously, when Δt is given, a linear decrease of μ leads to an exponentially decreasing probability for the success of a blocking attack. The only successful attack occurs when the adversary blocks the incoming messages of all SG-MIXes for quite a long time before the attacked message arrives, since he cannot know which SG-MIX node will be selected by the user for any particular message. This is usually impossible to do "on demand" and, in any case, would block the whole network, i.e. result in the loss of many messages due to time-outs, which would surely not go undetected.

The SG-MIX protocol allows the sender to calculate an accurate arriving time for each message. SG-MIXes or Trusted Third Parties (TTP) can use this feature to detect delaying attacks. For this purpose they send a message via an arbitrary path through the SG-MIX network to themselves using the SG-MIX protocol. If the related packets arrive within the calculated time period, the TTP can assume that there are currently no ongoing blocking attacks. If the packets arrive significantly later than expected or do not arrive at all, this is an indication for an ongoing blocking attack.

The keep-alive timer (i.e. timeout parameter) for such a feedback control mechanism must be chosen very carefully. If the parameter too small, then every variation of the transmission delay will result in an alarm (false positive). If the parameter is too large, blocking attacks will remain undetected (false negatives).

While it is clearly impossible to guarantee perfect untraceability by this method in the rigorous sense, it is secure in the same probablistic sense as most cryptosystems, if the parameter μ is suitably chosen.

4. Performance

To estimate the delays incurred for messages passing through an anonymity-providing node, we will model these nodes as 2-server systems (see Figure 3). The first server decrypts the messages and performs other tasks such as replay attack detection. The service times of this server are dependent only on server power and message size and therefore can be assumed to be constant. The appropriate model is then an M/D/1 server. The second server queues the messages until transmission. Because it does very little computing compared to the first server, it

can be viewed independently even if both servers are in reality one computer.

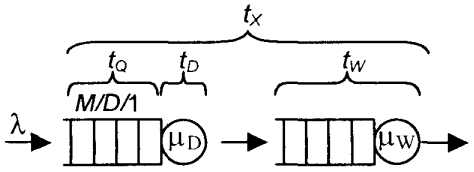

Figure 4: Anonymity providing node seen as a 2-server system

We will now derive the mean time a message spends in the server, that is, the expectation of t_X. Let $1/\mu_D$ be the time the first server needs to process a message and λ the rate of the message arrival process. Assuming that the system is stable, i.e. $\rho=\lambda/\mu_D<1$, we can determine the expectation for the delay in the first server using the Pollaczek-Khintchine formula and Little's Theorem [14]:

$$E(t_Q + t_D) = \frac{1}{\mu_D}\left(1+\frac{\rho}{2(1-\rho)}\right)$$

For a MIX with constant batch size n, the time spent in the second server is the time it needs to fill the batch. In a stable server system, the rate with which messages leave the server equals the arrival rate, the mean interarrival time at the second server is therefore $1/\lambda$ and the expectation of the delay

$$E(t_W) = \frac{n-1}{2\lambda}$$

and so we get (see Figure 4)

$$E(t_X) = \frac{1}{\mu_D}\left(1+\frac{\rho}{2(1-\rho)}+\frac{n-1}{2\rho}\right).$$

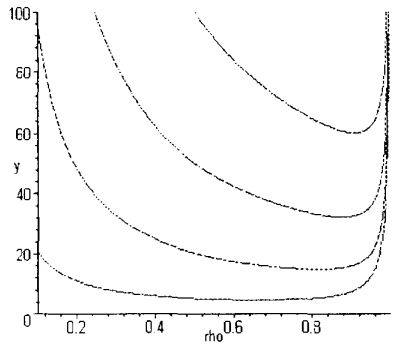

Figure 5: Mean total message delay $y=E(t_X)$ in a MIX with constant different batch sizes n (bottom-up) n=5, n=20 and n=50 and ($\mu_D=1$)

Obviously the expectation of t_W for a MIX with constant time intervals of length T is $T/2$ and the mean total delay

$$E(t_X) = \frac{1}{\mu_D}\left(1+\frac{\rho}{2(1-\rho)}\right)+\frac{T}{2}.$$

For an SG-MIX server, the mean delay in the second server is simply the expectation of the chosen exponential distribution $1/\mu_W$, and we get

$$E(t_X) = \frac{1}{\mu_D}\left(1+\frac{\rho}{2(1-\rho)}\right)+\frac{1}{\mu_W}.$$

While these average delay times are of the same order of magnitude for all three methods, it is important to note the differences:

SG-MIX is the only method where the user knows exactly how much delay the message will suffer, apart from the usually rather small network and queuing delays. This knowledge is important not only to compute the time windows mentioned in the previous section, but also to calculate turn-around times needed for congestion control in network protocols.

T-MIXes guarantee a finite upper bound for the delay time. This is true neither for MIXes with fixed batch size where delays depend on the arrival of other messages, nor for SG-MIX nodes because of the characteristics of the exponential distribution.

5. Conclusions

In this work we have analyzed the MIX concept and some variants with the aid of queuing theory. We show that queuing theory is not only good for performance evaluation but can also used for security evaluation.

Thanks are due to Joshua Mittleman for his helpful comments and reading of the manuscript. The author owes also Jan Egner a debt of gratitude for his contribution to this work.

6. References

[1] 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.

[2] D. L. Chaum, "*Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*" In: Comm. ACM, Feb. 1981, Vol. 24, No. 2, pp 84-88.

[3] D. L. Chaum, "*The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability*" Journal Cryptology, Vol. 1, No. 1, Springer-Verlag, 1988, pp 65-75.

[4] Communications of the ACM, "*Internet Privacy: The Quest for Anonymity*", vol. 42, num. 2, February 1999.

[5] L. Cottrell, "*MIXmaster and Remailer Attacks*", http://www.obscura.com/~loki/remailer/remailer-essay.html, 2001.

[6] D. J. Farber, K. C. Larson, "*Network Security Via Dynamic Process Renaming*, Fourth Data Communication Symposium, 7-9 Oct. 1975, Quebec City, Canada.

[7] H. Federrath (Ed.), "*Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*", LNCS 2009, Springer-Verlag 2001.

[8] A. Fasbender, D. Kesdogan, O. Kubitz, "*Variable and Scalable Security: Protection of Location Information in Mobile IP*", IEEE VTC'96, Atlanta, 1996.

[9] C. Gülcü, G. Tsudik, "*Mixing Email with Babel*" Proc. Symposium on Network and Distributed System Security, San Diego, IEEE Comput. Soc. Press, 1996, pp 2-16.

[10] B. R. Haverkort, "*Performance of Computer Communication Systems*", John Wiley&Sons Ltd 1998.

[11] IEEE Journal on Selected Areas, "*Copyright and Privacy Protection*", vol. 16. no. 4, May 1998.

[12] D. Kesdogan, "*Privacy im Internet*", Vieweg Verlag, ISBN: 3-528-05731-9 (in German).

[13] D. Kesdogan, J. Egner, R. Büschkes, "*Stop-And-Go-MIXes Providing Probabilistic Anonymity in an Open System*" Proc. 2nd Workshop on Information Hiding (IHW98), LNCS 1525, Springer-Verlag 1998.

[14] P. J. B. King, "*Computer and Communication Systems Performance Modelling*", Prentice Hall, 1990.

[15] L. Kleinrock, "*Queuing Systems*", Vol. I: Theory. John Wiley & Sons, 1975.

[16] D. Martin Jr., "*Local Anonymity in the Internet*", Ph.D. Dissertation, Boston University, 1999.

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

[18] A. Pfitzmann, M. Waidner, "*Networks without user observability – design options*" Advances in Cryptology – Eurocrypt '85, 219 LNCS, Springer-Verlag, 1985.

[19] J. F. Raymond, "*Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems*", International Workshop on Design Issues in Anonymity and Unobservability, Berkley, 2009 LNCS, Springer-Verlag, 2001.

[20] 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.

[21] M. K. Reiter, A. D. Rubin, "*Crowds: Anonymity for Web Transactions*", ACM Transactions on Information and System Security, volume 1, pages 66-92, 1998.

[22] C. Rackoff, D. R. Simon, "*Cryptographic defense against traffic analysis*", In Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing, pp 672-681, May 1993.

[23] M. Waidner, "*Unconditional sender and recipient untraceability in spite of active arracks*" Eurocrypt '89, LNCS 434, Springer-Verlag, 1989.

[24] Zero-Knowledge-Systems, Inc., The Freedom Network Architecture, http://www.freedom.net/ (2001)