Introducing Perfect Forward Secrecy for AN.ON

Benedikt Westermann[1] and Dogan Kesdogan[1,2]

[1] Q2S*, NTNU, 7491 Trondheim, Norway
[2] Chair for IT Security, FB5, University of Siegen, 57068 Siegen, Germany

Abstract. In this paper we discuss AN.ON's need to provide perfect forward secrecy and show by an estimation of the channel build up time that the straight forward solution is not a practical solution. In the remaining paper we propose an improvement which enables AN.ON to provide perfect forward secrecy with respect to their current attacker model. Finally, we show that the delay, caused by our improvement, does not decrease the performance significantly.

1 Introduction

Anonymity systems are used by various users with manifold motivations. Some users just want to preserve their privacy, while other users need anonymity systems to circumvent censorship systems. While the state of being anonymous is for some users just a positive side effect that however is not needed, it is of major importance for other users to avoid serious consequences. The latter group is for instance represented by a whistler-bowler who wants to inform the public or a law enforcement agency about a serious crime within an organisation. Obviously, the use of his real identity or an easy to trace pseudonym is not a good idea if the whistle-bowler would face serious consequences due to his action.

Therefore, the whistler-bowler has a strong motivation to be anonymous during his action and naturally wants to be anonymous also after he has provided the information to a third party. Especially the part of staying anonymous is important. Thus, it should not be possible after a week or even several years to reveal the identity of the user of an anonymity network. Therefore special care must been taken to protect the identity of the users after their actions have been taken. The simplest solution is not keeping any records of the user's communication. Although, this solution is simple and represents the usual policy of the operators of nodes in an anonymity network, it does not hinder a third party or a malicious node to record the transfered messages.

The probable most popular anonymity systems are Tor[1] and AN.ON[2]. Both systems provide different solutions to anonymize users. This leads naturally to different requirements, preconditions and problems. One problem AN.ON faces is the lack of *perfect forward secrecy.* Thereby perfect forward secrecy is the

* "Center for Quantifiable Quality of Service in Communication Systems, Center of Excellence" appointed by The Research Council of Norway, funded by the Research Council, NTNU and UNINETT. http://www.q2s.ntnu.no

property that if a long-term private key used in a key establishment protocol gets compromised it does not affect the security of session keys that had once generated with help of the compromised private key[3]. Due to the lack of perfect forward secrecy the anonymity of all users can be revoked even weeks after their connections took place due to compromised private keys. Obviously, a period of several weeks provides an attacker with various possibilities to compromise the keys. He can mount various targeted attacks, for example he can blackmail operators or can attack the servers itself to gain access to private keys. Once he has accomplished this and has a record of the previous communications the attacker can deanonymize all users with all their connections during the usage period of the compromised keys. Obviously, if this situation occurs it is without doubt a threating situation for the users. Hence, countermeasures are necessary to protect the users against such attacks.

In this paper we propose how this kind of attacks can be prevented by introducing perfect forward secrecy for AN.ON without decreasing its performance notably.

The paper is structured as followed. In Section 2 we describe Tor and AN.ON in more detail. Based on the description we point out the differences between AN.ON and Tor, the implications of the differences and the challenges for AN.ON regarding perfect forward secrecy. In Section 3 we describe the idea and propose an improvement to counter the described threat which arises due to compromised keys. Additionally, we estimate the additional delay that our improvement would introduce to the channel build up time. The security implications are discussed in Section 4. Section 5 concludes the paper.

2 Two Popular Anonymity Networks

In this section we describe the functionality of AN.ON [2] and Tor [1] and discuss their differences.

2.1 Tor: The Onion Routing

In Tor [1], the most popular low latency anonymization network, a user sends packets over so-called *circuits.* A circuit is a user selected path through the Tor network and it consists out of several nodes. A node in Tor is called *Onion Router (OR).* By routing the message over several ORs it is achieved, that only the user is aware from whom to whom his message travels. This state is called *relationship anonymity.* Tor as well as AN.ON aim to provide this kind of anonymity for their users against the network operators or a local attacker.

A user who wants to send a message anonymously with Tor has to establish a circuit. During a circuit setup, a user authenticates a selected OR and creates a session key with help of a DH key exchange. Additionally, he can extend the circuit to another OR. Usually, a circuit consists out of three ORs. After the circuit is successfully established, a user can tunnel over the circuit various (data) *streams.* A stream is an end-to-end connection between the user and the final destination, for example a web server. With the first packet in a stream a user informs the last OR in the circuit about the final destination which subsequently establishes a TCP connection to the destination.

The *telescopic* path-building design with its different layers of encryption prevents the first OR within a circuit to see the final destination of a stream[1]. Each OR in the circuit can remove exactly one layer of the encryption. This ensures that only the last OR can read the final destination.

The last OR in a circuit can link the different streams of a user, since the streams are tunneled over a single circuit which is only used by a single user. On the long run the linkability of the streams can threaten the anonymity of users. The Tor client, also called *onion proxy (OP)*, changes therefore periodically the circuits of its user.

To authenticate the ORs a custom made protocol is used. The protocol uses a DH key exchange which is authenticated with help of a known public encryption key. The protocol was proven to be secure under the random oracle model in [4].

2.2 AN.ON: Anonymity Online

Another popular anonymity system is AN.ON[2]. Nowadays it is also called JonDonym. On the first glace it seems that AN.ON and Tor are quite similar. Both systems use layered encryption to achieve anonymity for their users. A closer look shows that AN.ON and Tor are quite different. One important difference is, that the sequence of servers in AN.ON is fixed and cannot be chosen by the users. The user only has the option to choose from a set of predefined sequences of nodes. A sequence of nodes is determined by the nodes' operators and it is called a *cascade*. Figure 1 depicts a cascade. In AN.ON a node is named *mix*.

Similar to Tor, AN.ON's users encrypt data in layers and send it along a cascade. Data intended for the same TCP connection is sent over a so-called *channel*. Every time an application requests a new TCP connection, a new channel has to be established. Each mix removes exactly one layer of encryption from a packet. Due to the fix sequence of the mixes it's important to provide unlinkability for channels of a user especially with respect to the last mix. Otherwise an attacker

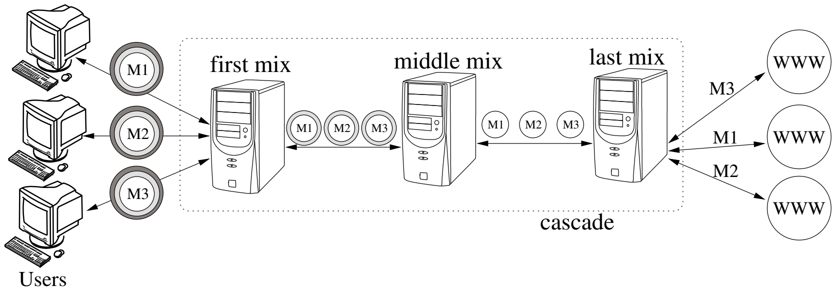

Fig. 1. Sketch of a Cascade in AN.ON

could deanonymize a user and all his prior channels by just attacking a single channel. Thus, for each channel established by a user, it is necessary to generate new keys in a way such that the mixes can neither link the key nor the channel to prior keys and channels respectively.

When a user wants to establish a new channel, for example if he wants to request a website, he generates two new session keys for each mix by uniformly selecting two 128 bit words from all possible 128 bit words. The generated keys are sent along the cascade and they are encrypted with the corresponding public encryption keys of the mixes. Thus, only the first mix in a cascade can link different channels with each other due to the user's IP address. This procedure fulfills therefore the unlinkability requirement of AN.ON.

2.3 The Challenge of Introducing Perfect Forward Secrecy

As mentioned above, Tor uses a DH key exchange to generate session keys between the OP and the ORs. In Tor the forward secrecy is achieved by the properties of the DH key exchange. Additionally, the fact that a DH key exchange results in a different key even though one side uses the same exponent, helps also to protect against replay attacks.

Unfortunately, AN.ON does not perform a DH key exchange. Contrary to Tor, AN.ON uses the cascade principle instead of the free routing principle and requires therefore the unlinkability property for each channel/stream. Thus, if AN.ON used a DH key exchange in way Tor does, it would require a user to perform a DH key exchange for each TCP connection an application requests. This would introduce an additional delay. The delay however greatly influences the user experience regarding the performance. For example various channels are needed to retrieve a single web page. Most often these channels are created in a sequential way. An additional delay of 100 ms can therefore extend a complete website retrieval by several seconds.

In order to roughly estimate the dimension of the additional delay, which would be introduced when every channel requires a DH key exchange, we conducted a small experiment with AN.ON. In this experiment we measured the build up times for a channel from the user to a destination at a cascade of length three. To do so, we connected to a cascade via the AN.ON client. After that our measurement started. The following procedure was repeated 200 times. We began to stop the time and requested a new channel over the cascade back to ourself. As soon as a TCP connection was established by the exit node, we stopped the time measurement. Please not, that this does not mean that we received already a confirmation about successful establishment on the other side of the channel. In total we measured the time it took to establish one TCP connection, to transmit three mix packets with a size of 998 bytes as well as the time the processing of the packets took. In addition we measured in a separate measurement the round trip times between the last mix in the cascade and our measurement node.

For the measurement we chose the payment cascade "Koelsch-Rousseau-SecureInternet"[1]. We collected 200 samples with an average channel buildup time of 229 ms with a 0.95-confidence interval of $[225.7\ ms, 232.56\ ms]$ and a standard deviation of 48.5 ms. For the round trip time we measured 52 ms on average from the last mix to the server. Thus, we can roughly estimate that 78 ms are used to establish the TCP connection from the last mix to the server under the assumption that no retransmissions of packets for the TCP handshake were necessary. The processing and the transmission of three mix packets took therefore 151 ms.

A DH-Key exchange for each new channel is likely to introduce $(n+1) \cdot n - 2$ additional messages as depicted in Figure 2 assuming that the client does not need to establish each time a key exchange with the first mix. In the current protocol n mix packets are exchanged between the mixes to establish a connection to the retriever. If we assume that each additional message introduces an additional delay of 20 ms[2] an additional delay of 200 ms would be introduced for a cascade with three mixes. This is more as twice as much as an establishment in the current version. Additionally, this estimation is quite rough, it does not consider the extra load on the mixes which is introduced due to the asymmetric cryptography. The estimation strongly indicates that the straight forward solution is not a suitable solution. Another solution would be, to perform a DH key exchange only to the middle mix for every channel. However, this would already introduce 4 new messages leading to an estimated additional delay of 80 ms. This delay is significant and therefore this solution is also not practical. Especially considering two facts. Firstly, the anonymity depends on the number of active users. Secondly, there is a linear correlation between the delay and the number of users[5]. In other words, the anonymity, a low latency anonymity system can provide, depends among others on the performance of the system. Thus, the solution of performing a DH key exchange for every new channel is not acceptable.

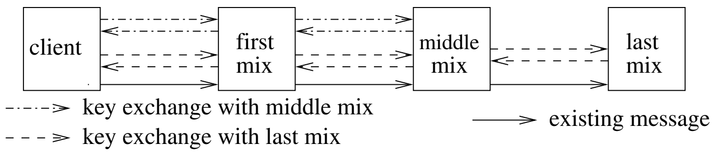

Fig. 2. Additional messages if a DH key exchange would be introduced

The challenge in AN.ON is to introduce a DH key exchange with neither risking the anonymity of the user nor decreasing the performance significantly.

[1] `http://anonymous-proxy-servers.net/de/status`

[2] This is an optimistic value considering that 3 messages took already 150 ms.

3 Introducing Perfect Forward Secrecy

A DH key exchange provides the property of perfect forward secrecy. AN.ON does not use a DH key exchange, instead it uses the private keys directly to decrypt the session keys sent by the clients and therefore it cannot provide perfect forward secrecy. In this section we present an approach which introduces a DH key exchange without influencing significantly the build up phase of a new channel which is an important performance parameter for low latency anonymity networks.

3.1 Observable Information

The idea of our approach is based on the fact, that different mixes have different knowledge about the channels of users. In a cascade we can distinguish between three types of mixes. The first type is the *first mix.* This mix sees the IP addresses of the users and it is the entry point for the cascade. This mix does not see the final destination. *Middle mix*es represent the second type and they are only present in cascades having more than two mixes. They neither see the final destination of a message nor the user's IP address. The last type of mix is the *last mix* that performs the request on behalf of the user. This type of mix sees the final destination of the messages, but it does not see the user's IP address.

As mentioned before, it is important in a cascade that the last mix cannot link single channels of a user. Therefore, a user should generate new session keys for every new channel. However, we claim that the channels of a user have to be unlinkable only with respect to the last mix, but they can be linkable for the first and the middle mixes. In the case of a first mix, the mix can already link user's channels due to the user's IP address and the user's established TCP connection respectively. Hence, a newly generated key cannot establish the unlinkability property with respect to the first mix.

Contrary to the first mix, middle mixes cannot link the channels of their users. Compared to the information that is available to a middle mix, the first mix has at least the same information available. If a middle mix can link different requests, the information that is available to the middle mix is still a strict subset of the information available to the first mix. This implies, that if a middle mix can revoke the relationship anonymity due to the introduced linkability of a user's requests, then a first mix is able to do the same, since it has at least the same information available. Therefore, assuming that an attacker only controls a single mix, we do not weaken the security of AN.ON.

Therefore, we propose to introduce DH key exchanges between a user and middle mixes as well as the first mix. The DH key exchanges are performed once directly after a user has established a connection to a cascade. The generated key serves as a master key in order to derive the different session keys for every new channel. Note, that reusing the same keys for de- and encryption is dangerous as for example demonstrated in the case of the old AN.ON protocol in [6].

Due to the fact, that the master key is once generated when a user connects, it introduces only once an additional delay, namely when the user connects to a cascade.

3.2 Used Protocol: Tor Authentication Protocol

In order to perform an authenticated DH key exchange we reuse Tor's authentication protocol (Tap)[7]. AN.ON fulfills the requirements of the protocol completely, and therefore no modifications of the protocol are necessary. Thus, the risk of introducing vulnerabilities in the design is lower. Additionally, the Tap is proven to be secure under the random oracle model[4].

In this paper we use the protocol as it was formally described and proven in [4]. For the protocol it is assumed that a user knows the public encryption key of the node. Let p and q both primes with $q = \frac{p-1}{2}$ and g a generator of the subgroup $\mathbb{Z}_p^*$ of order q. Let l_x the maximum length of the exponent that is chosen from the interval $[1, \min(q, l_x) - 1]$ [4].

In the first steps of the protocol a user chooses uniformly an exponent x. This exponent is used to calculate g^x which is sent encrypted with the node's public encryption key to the node. The node decrypts the message and checks if the decrypted message m is within the interval $[2, p-2]$. If so, the node selects uniformly an exponent y from the interval $[1, \min(q, l_x) - 1]$. The node computes g^y. Additionally, it computes the hash of m^y. Both results are sent back to the user. The user can now compute the key K himself and can verify that the node uses the same key by hashing his own key and comparing it with the received hash (b).

3.3 Key Derivation Function

As mentioned above the generated key K is not used directly to encrypt data. Instead, the session keys have to be derived by this generated master key K. If we assume that the packets are received in the same order as they were sent, we can use a standard *key derivation function (kdf)* like KDF1 from the ISO/IEC 18033-2:2006 standard [8].

Due to the fact, that the first mix directly communicates with a user and hence the packets arrive in the same order as they were sent, it is safe to use such a key derivation function. This key derivation function is based on a counter which is hashed together with the master key. If AN.ON uses SHA-1 as a hash function which produces 20 bytes and the required key length is 16 bytes, then the ith key can be generated by taking the first 16 bytes of SHA-1(K, i). Thereby i is represented by a fixed length integer limiting the maximal number of keys that can be derived. In [8] i has a length of 4 bytes and thus at most 2^{32} keys can be generated. Due to the fact,that AN.ON requires two keys, one for the *forward direction* and one for the *backward direction*, two keys have to be generated for each channel. The forward direction describes the processing of packets which are conveyed by the client and the backwards direction represents the other case, namely the processing of packets that are sent by the last mix. For example, keys with an odd counter can be used for the forward direction while keys with an even counter are used for the other direction.

A positive side effect of this procedure is, that replays of an old message do not lead to a successful decryption of the messages. The reason for this is that different keys are used for the en- and the decryption. While the key, used for encrypting the message, is based on an old counter, the mix uses a newer counter to generate the key for the decryption. Thus, both keys are different and the decryption fails. Currently, AN.ON has to protect by other means against replays[2,9].

Usually, AN.ON's mixe do not reorder the packets, since this would result in a decreased performance. However, it is possible to activate this feature for a mix. In the case that a previous mix reorders packets, the above mentioned kdf cannot be used for the middle mixes. Therefore, another kdf has to be used.

In the current protocol, a user sends two session keys to every mix whenever a new channel has to be established. One key for the forward direction and a second key for the backward direction. We name the key used for the forward direction k_f and the key for the backward direction k_b.

To minimize the necessary changes in the software, we use the sent keys k_f and k_b to derive the new session keys. We propose the following function, thereby K represents the master key generated with the Tor authentication protocol. Since the required keys are shorter than the hash, only the first 128 bits are used for the keys:

$$\begin{aligned} k_{nf} &= H\left(K||k_f\right)[0,127] \\ k_{nb} &= H\left(K||k_b\right)[0,127] \end{aligned}$$

As no sequence number is involved in this key derivation, the protocol is again vulnerable to replay attacks. However, the replays are limited for the time a middle mix knows the master key. The master key should be deleted immediately after a user disconnects from the cascade.

In order to counter a replay attack, one of the described replay protections can be used.

3.4 Estimation of the Additional Delay

As discussed earlier, our new approach should not significantly lower the performance of AN.ON. Due to the fact, that the DH key agreement is performed once when the user connects to the cascade, the additional delay is of minor importance. Thus, even a delay of several seconds is acceptable. The important question is, if the key derivation as well as the lookup of the master key have a significant influence on the build up time of a channel. To answer this question we performed a simulation of the additional operations necessary to calculate the session keys with help of the key derivation function.

In our simulation we measured the time to lookup a key from a hash table with a size of 2000000 entries. For each round we populated the hash table with 1000000 randomly selected (master) keys, 16 bytes for each key, which where stored under a 4 byte identifier in the hash table. The identifiers for the keys were also randomly chosen. The chosen values represent reasonable values regarding the current protocol. After we have populated the hash table, a random identifier was picked and the corresponding key was retrieved. This key was then hashed together with a 4 byte key representing k_f and k_b respectively. SHA-1 was the used hash function. For each round we picked 200000 different identifiers and generated two session keys each. For every new round we generated and populated a new hash table.

In total we simulated 100 rounds. On average the generation of the two session keys together with the master key lookup took $4.53\mu s \approx 0.004ms$ with an 0.95-confidence interval of $[4.506\mu s : 4.549\mu s]$. The simulation was performed on a Intel Core2Duo P8400 CPU. This simulation indicates that the additional overhead, introduced by our improvement, is negligible considering a total delay of over $100ms = 100000\mu s$.

4 Discussion

In Section 3.1 we discussed the information which is observable by a mix. The last mix can observe the final destination and the content of a message. This is the reason to require the unlinkability property for AN.ON. A possible, but risky solution would be to perform a DH exchange periodically with the last mix. The problem with this approach is, that the last mix could correlate different key exchanges with each other. For example, a user who terminates a "session" is likely to be the initiator of one of the next key exchanges, since he uses the same cascade for his subsequent channels. Hence, we do not propose this solution in the case of AN.ON. Contrary to AN.ON, in Tor the whole circuit is changed and therewith also the exit node. Hence, it is unlikely that one of the next circuits, observed by the exit node, is originated by a user who recently has closed a circuit and therefore the linkability of requests is only limited to a short period in the case of Tor.

One problem arises when mixes collude, namely the last mix with its predecessor. Due to our approach the collaboration of these two mixes can lead to the identification of a single user with all of his connections from his current session. This happens for example, if he sends an identifying information in one of his channels. This could not even link every visit from the current session to him, but also his previous sessions. This is the case if he visits regularly unpopular sides in a single session. This risk is not present in the current approach. However, also in the current version of AN.ON a collaboration of two mixes, namely the first and the last mix, is critical and leads to a deanonymization of a user. In this case, a user must not even send an identifying information over one of his channels. To mount the attack the first and the last mix have to introduce artificial timing or burst patterns to correlate in and outgoing packets. The feasibility of such a traffic conformation attack was shown in [10] in the case of Tor. Thus, neither our nor AN.ON's current version can protect against a collaboration of two mixes.

Due to the fact that no DH key exchange is performed between the user and the last mix of a cascade, the perfect forward secrecy does not hold for the last mix and therewith is also limited for the whole cascade. The interesting question is, in which situations AN.ON can provide anonymity even if the used private encryption keys has been compromised at some time after the connection took place. In the original version of AN.ON, an attacker who recorded every packet on the first mix can deanonymize every user under the compromised private key assumption. The attacker sees the IP address of all users and since he knows every private key, he can decrypt all session keys and subsequently all messages of all users that have once passed the mix during the usage period of the compromised key. Hence, the attacker can see the final destination of the messages as well as the IP address of the sender. Therewith he has revoked the relationship anonymity of all users.

With the introduced approach, the attacker needs to control the first $n-1$ mixes to mount the attack. Hereby n is the length of the cascade. If an attacker wants to mount the attack, he has to control the first $n-1$ mixes while the connection takes place. In this case he can easily correlate the user's messages with the outgoing messages at the $n-1$th mix by decrypting every intermediate message. The reason that the attacker does not need to control the last mix, is due to the assumption that the attacker will eventually compromise the private encryption key of the last mix. Therefore, he can decrypt the messages of the last mix later and therewith retrieve the final destination.

Obviously, our improvement does not provide any additional protection if we consider a cascade with length 2. Here the attacker only needs to control the first mix to mount the attack.

However, an attacker who has the power to control $n-1$ mixes in a cascade when the connection takes place can usually mount simpler attacks requiring less resources to deanonymize users.

Another problem occurs if the last middle mix gets the private key of the last mix. In this case the middle mix has the possibility to link different sessions to a user if the user transmits identifying information over one of his channels. Thus, the middle mix can mount the same attacks a last mix could mount due to the introduced linkability. However, the middle mix cannot directly identify every user under the compromised key assumption. Thus, the impact of the attack is less.

5 Conclusion

In our paper we argue that the lack of perfect forward secrecy exposes all users of AN.ON to a great risk if an attacker compromises the long term keys of the mixes. By introducing perfect forward secrecy in AN.ON up to the middle mixes, this thread is diminished. To perform a DH key exchange for every channel and thereby keeping the unlinkability property of AN.ON is unfortunately not possible due to practical reasons. The additional delay is simply to high.

By allowing middle mixe to link different channels to the same originator we do a trade-off between different risks and their impacts. While all users are affected by compromised keys in the original version, even though they did everything correct, only a couple of users are affected with our improved version. Therefore we claim that the linkability of channels regarding middle mixes reduces the overall risk.

Additionally, our approach does not introduce a significant performance decrease. Our estimations indicate that the introduced delay is less than 0.01 %. Since a mix can predict the session keys due to the knowledge of the master key, it might be possible to eliminate several asymmetrical cryptographic decryption operations. Thus, it is even possible that our approach decrease the overall build up time of a channel.

With our improvement an attacker cannot deanonymize earlier connections of all users by compromising the long term private keys of the mixes. Thereby, we improve the security of AN.ON.

References

1. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The second-generation onion router. In: USENIX Security Symposium, USENIX, pp. 303–320 (2004)
2. Berthold, O., Federrath, H., Köpsell, S.: Web MIXes: A system for anonymous and unobservable Internet access. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, pp. 115–129. Springer, Heidelberg (2001)
3. Mao, W.: Modern Cryptography: Theory and Practice. Prentice Hall Professional Technical Reference (2003)
4. Goldberg, I.: On the security of the Tor authentication protocol. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 316–331. Springer, Heidelberg (2006)
5. Köpsell, S.: Low latency anonymous communication - how long are users willing to wait? In: Müller, G. (ed.) ETRICS 2006. LNCS, vol. 3995, pp. 221–237. Springer, Heidelberg (2006)
6. Westermann, B., Wendolsky, R., Pimenidis, L., Kesdogan, D.: Cryptographic protocol analysis of an.on. In: Proceedings of the 14th International Conference of Financial Cryptography and Data Security, Tenerife, Spain (2010)
7. Dingledine, R., Mathewson, N.: Tor protocol specification (visited Feburary 3, 2010)
8. ISO/IEC 18033-2: 2006: Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers. ISO, Geneva, Switzerland (2006)
9. Köpsell, S.: Vergleich der Verfahren zur Verhinderung von Replay-angriffen der Anonymisierungsdienste AN.ON und Tor. In: Dittmann, J. (ed.) Sicherheit. LNI, vol. 77, pp. 183–187. GI (2006)
10. Øverlier, L., Syverson, P.: Locating hidden servers. In: Proceedings of the 2006 IEEE Symposium on Security and Privacy. IEEE CS, Los Alamitos (2006)