Introducing Perfect Forward Secrecy for AN.ON

No Author Given

No Institute Given

Abstract. In this paper we enhance AN.ON in order to provide perfect forward secrecy for all mixes in a cascade except for the last mix. By this improvement, we limit the time frame an attacker has to compromise the used keys of the mixes from several weeks to a couple of hours. Thereby we respect the boundary conditions of AN.ON. Thus, our approach does not introduce an additional delay for the channel build-up phase. Beside this we discuss the required unlinkability property of AN.ON and discuss the approach of Tor with respect to perfect forward secrecy and linkability of different connections.

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 and 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.

Depending on the system, an attacker may try to replay recorded data at a later point of time again into the system, decrypt data if he manages to compromise the used keys, for example exploiting a software bug or blackmailing the operators, or try to analyse the data in other ways. Hence, countermeasures are necessary to protect the users against such attacks.

The probable most popular anonymity systems are Tor[1] and AN.ON[2]. Both systems provide different solutions to anonymize users, which naturally

leads to different requirements. Thus, it is most often not possible to map a proposed countermeasure from Tor to AN.ON or vice versa. In addition, both systems are deployed systems that are used by thousands[1] of users. Thus, modifications on the protocols have to satisfy certain boundary conditions, like performance and integrability constraints. The boundary conditions represent a important challenge with respect to deployed low latency anonymity systems.

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 the message travels. This state is called *relationship anonymity* and it is the objective of Tor as well as AN.ON to provide this kind of anonymity for their users against the network operators.

A user who wants to send a message anonymously with Tor has to establish first a circuit. During a circuit setup, a user authenticates a selected OR, creates a session key with the this OR, and possibly extends the circuit to another OR. Usually, a circuit consists out of 3 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 over the final destination who subsequently establishes a TCP connection to the destination.

The *telescopic* path-building design with its different layer of encryption prevents the first OR of 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 that 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 [3].

[1] In case of Tor even hundreds of thousands of users.

2.2 AN.ON - Anonymity Online

Another popular anonymity system is AN.ON[2] which is nowadays also called JonDonym[2]. 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*.

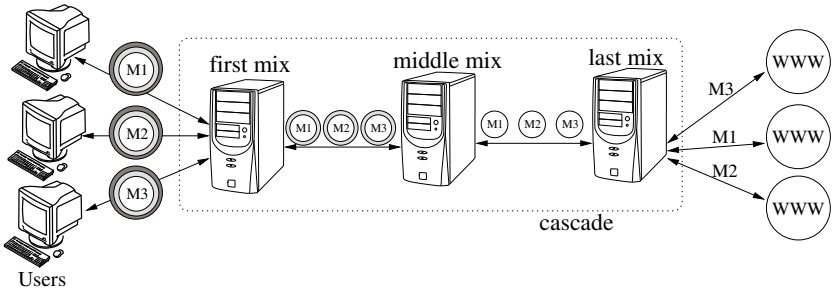

Fig. 1. Sketch of a Cascade in AN.ON

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*. A channel has to be established for every new TCP connection, which is required by an application. Each mix removes exactly one layer of encryption from a channel. Due to the fix sequence of the mixes it's important to provide unlinkability for channels of a user, otherwise an attacker 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[3], he generates a new session key for each mix by uniformly selecting a 128 bit word 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] JonDonym is a commercial version of former AN.ON research project.
[3] Note, this is done for almost every HTTP request.

The lack of a DH key exchange for every new channel allows the mixes to act deterministically, and therefore AN.ON requires extra countermeasures against replay attacks. Two different approaches were proposed for AN.ON. One approach, similar to the one mentioned in [4], was proposed in [2] as a solution for AN.ON. In this solution an identifier is stored in a database for each symmetric key that has once been used. The identifier remains in the database as long as the private key of the mix is in use. Every attempt to reuse the same key for a new channel will be rejected. In another approach [5], a time period is encoded within the mix packet. Only packets that are in the current time period of a mix are accepted. An identifier for each key, used in a time period, is stored in a database. The identifier is stored as long as the mix is in the time period. This approach aims to minimize the performance decrease that is introduced by the additional database lookups. However, a protection against replays was not active as demonstrated in [6]. Here the authors successfully replayed a whole session into a cascade.

2.3 Comparison Between AN.ON and Tor

As mentioned above, Tor uses a DH key exchange to generate session keys between the OP and the ORs. The DH key exchange in Tor serves two purposes. The first one is to provide perfect forward secrecy and the other one is to protect against replay attacks [1]. The forward secrecy is achieved by the properties of a DH key exchange and the deletion of the keys as soon as a circuit is closed. The fact that a DH key exchange results in a different key even though one side uses the same exponent, helps to protect against replay attacks. In combination with the *telescopic* circuits of Tor and the use of AES in *counter mode* the threat of replays is completely countered by Tor's design.

Unfortunately, AN.ON does not perform a DH key exchange. The reason for this is that contrary to Tor, AN.ON uses the cascade principle instead of the free routing principle and requires therefore the unlinkability property for the channel. 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 new channel. Although it is possible, it is not a suitable solution, due to the additional messages that have to be introduced to agree upon a key with each mix. The additional messages result in a higher delay for the channel build-up phase as well as in a bad user experience. As shown in [7] there is a linear correlation between performance and the number of users. Thus, in order to provide reasonable performance and therefore keep the delay to a minimum, a DH key exchange cannot be executed for every new channel.

Due to the fact that AN.ON does not make use of a DH key exchange, it is not able to provide perfect forward secrecy, instead it can only provide eventually forward secrecy. This means that AN.ON provides forward secrecy as soon as the used private keys are deleted. This happens usually several times a month. During the usage period of the keys a higher risk threatens the users' anonymity. This risk arises, since an attacker can get in possession of the keys, for example by compromising the mix or by blackmailing the mix operators

even though the connection was already closed. If the attacker succeeds in this and has a copy of the transfered data, he would be able to decrypt the entire data stream. The same attack is obviously not possible in the case of Tor, since the operators do not know the keys after the connection has been closed. If an attacker wants to decrypt messages in Tor, he needs to control the ORs already when the communication takes place. As a recent example of Tor shows, this can happen [8]. Fortunately, the attackers did not realize which computers they have compromised and so it seems that they did not try to attack the Tor process on the computers.

Thus, the challenge in AN.ON is to introduce a DH key exchange without risking the anonymity of the user or decreasing the performance of AN.ON.

3 Introducing Forward Secrecy for AN.ON

As in previous sections discussed a DH key exchange has two major benefits for anonymity system: perfect forward secrecy and a easy to integrate replay protection for every connection. AN.ON does not use a DH key exchange due to performance reasons. In this section we present an approach which introduces a DH key exchange without influencing the build up phase of a new channel which is important to low latency networks. In addition, it can be easily integrated in the current version of the protocol.

3.1 Observable Information

The idea of our approach is based on the fact, that different mixes have different knowledge about user's channels. 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 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[4] 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. This would mean that the system is in its current state insecure. However, there are no known attacks exploiting the linkability of the channels with respect to the first mix.

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 connected to the 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 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)[9]. AN.ON fulfills the requirements of the protocol completely, and therefore no modification on the protocol is 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[3].

Tor's authentication protocol is depicted in Figure 2 as it is formally described and proven in [3]. For the protocol it is assumed that a user knows the public encryption key of the mix. 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 $[1, \min(q, l_x) - 1]$ [3]. In the first steps 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 in 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 cannot be directly used to encrypt data. Instead, the session keys have to be derived by this generated key K. If we assume that the packets are received in the same order as they were sent, we

[4] The first mix can see contrary to the middle mix the user's IP address.

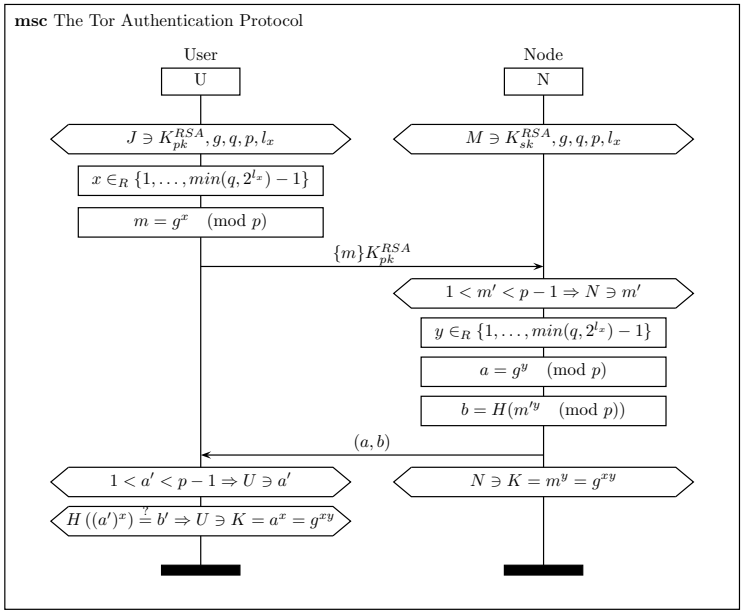

Fig. 2. The Tor Authentication Protocol

can use a standard *key derivation function (kdf)* like KDF1 from the ISO/IEC 18033-2:2006 Standard [10].

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[5], it is safe to use a standard kdf like the KDF 1. This key derivation function is based on a counter which is hashed together with the master key. If AN.ON uses SHA-1 as 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 [10] i has a length of 4 bytes and thus at most 2^{32} keys can be generated. Due to the counter that is synchronized between both parties, replay attacks are no longer possible. The reason for this is, that a replayed *channel open packet* is encrypted with an key which was produced by hashing an old counter. The mix uses however a higher counter and thus the packet cannot be decrypted within the cascade. Additionally, the user and the mixes are no longer synchronized. Thus, every attempt to open a new channel fails.

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.

[5] This is guaranteed by the underlying TCP Connection.

In the current protocol, a user sends new session keys to the mixes whenever a new channel has to be established. For each mix two keys are sent. One key for the *forward direction* and a second key for the *backward direction*. 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. We name the key used in 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:

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

Since 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. Due to the fact, that the possible keys are limited to a single client, the check should be faster.

To even more limit the number of key identifiers that have to be stored, a user can perform the Tor authentication protocol in periodical time intervals.

3.4 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 the 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. Due to this reason 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.

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 as been compromised at some time after the connection[6] took place. In the original version of AN.ON, an attacker who recorded every packet on the first mix can deanonymize the user under the compromised private key assumption. The attacker sees therefore the IP address of the user and since he knows every private key, he can decrypt the session keys and subsequently completely decrypt every message of a user that has 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 the user.

With the introduced approach, the attacker needs to control the first $n-1$ mixes to attack the proposed scheme. Hereby n is the length of the cascade. If an attacker wants to attack the proposed scheme, he has to control the first $n-1$ mixes. 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 eventually decrypt the messages for the last mix 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 controls $n-1$ mixes in a cascade when the connection takes place can usually mount simpler attacks requiring less resources to deanonymize a user. For example, he only needs to observe the first or exit mix in a cascade and simultaneously be able to delay packets on the other side of the cascade. If he can do this, he is able to mount a traffic confirmation attack against the AN.ON.

Under the assumption that the mixes do not reorder the packets, all mixes except the last can use the key derivation function which uses counter (KDF1). This has two major advantages. The first advantage is, that a user only need to generate a single session key. This key is intended for the last mix in a cascade. The first mix and the middle mixes can derive their keys by their own. Thus, they can save one asymmetric decryption operation for each connection that is established. The second advantage is, that the middle as well as the first mix do not need to implement one of the extra replay protections, since this is already countered by the design as discussed in Section 3.3. Thus, the delay by channel build-up phase should be significantly decreased.

4 Conclusion

In this paper we introduced an improvement for AN.ON in order to provide perfect forward secrecy for all mixes in a cascade except for the last mix. By this improvement, we limit the time frame an attacker has to compromise the used

[6] E.g. by blackmailing the operator during the usage period of the private key

key material from several weeks to a couple of hours without an additional delay for the channel build-up phase. Moreover, it is likely that the delay for the build-up phase even decreases. The reason for this is the replay protection mechanism. Here a mix has to store an identifier for every used key in a database. Each time a new channel request arrives, a mix has to check if the key was already used. Due to our approach the number of potential keys is limited to the keys a user produces during his usage of the cascade. Therefore, the mix only needs to check if the received key was used in the current "session" instead of checking if the key was once used during the last weeks. Thus, the storage requirements as well as the lookup times decreases. The latter obviously influences the build-up time of a new channel.

Thus, we significantly improved the security of AN.ON by introducing perfect forward secrecy. Additionally, we fulfill the boundary conditions of AN.ON. As a positive side effect our approach is likely to decrease the channel build-up time, which is an important factor regarding the user experience.

References

1. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The second-generation onion router. In: USENIX Security Symposium, USENIX (2004) 303–320
2. Berthold, O., Federrath, H., Köpsell, S.: Web MIXes: A system for anonymous and unobservable Internet access. In: Designing Privacy Enhancing Technologies. Volume 2009/2001 of Lecture Notes in Computer Science., Springer (2001) 115–129
3. Goldberg, I.: On the security of the Tor authentication protocol. In Danezis, G., Golle, P., eds.: Privacy Enhancing Technologies. Volume 4258 of Lecture Notes in Computer Science., Springer (2006) 316–331
4. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM **4**(2) (February 1981) 84–88
5. Köpsell, S.: Vergleich der Verfahren zur Verhinderung von Replay-angriffen der Anonymisierungsdienste AN.ON und Tor. In Dittmann, J., ed.: Sicherheit. Volume 77 of LNI., GI (2006) 183–187
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 (Jan 2010)
7. Köpsell, S.: Low latency anonymous communication - how long are users willing to wait? In: Emerging Trends in Information and Communication Security. Volume 3995/2006 of Lecture Notes in Computer Science., Springer (Juni 2006)
8. Dingledine, R.: Tor project infrastructure updates in response to security breach. http://archives.seul.org/or/talk/Jan-2010/msg00161.html (Jan 2010)
9. Dingledine, R., Mathewson, N.: Tor protocol specification visited 03.02.2010.
10. ISO/IEC 18033-2:2006: Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers. ISO, Geneva, Switzerland