SHALON: Lightweight Anonymization based on Open Standards

Andriy Panchenko
RWTH Aachen University
Computer Science Department
Ahornstr. 55, D-52074 Aachen, Germany
panchenko@cs.rwth-aachen.de

Benedikt Westermann
Q2S, Norwegian University of
Science and Technology,
N-7491 Trondheim, Norway
benedikt.westermann@q2s.ntnu.no

Lexi Pimenidis
RWTH Aachen University
Computer Science Department
Ahornstr. 55, D-52074 Aachen, Germany
lexi@cs.rwth-aachen.de

Christer Andersson
AḞ Consult
Hamntorget 3, P.O. Box 467,
SE-651 10 Karlstad, Sweden
Christer.L.Andersson@afconsult.com

***Abstract*—In this paper, we introduce a novel lightweight anonymization technique called *Shalon*. It is based on onion routing, aims to reduce complexity, and delivers high bandwidth. We have, compared to the widely known approach Tor, slightly reduced the level of security in favor for greatly increased performance.**

The most significant advantage compared to other approaches is that Shalon is fully based on standardized protocols, which makes our approach highly efficient and easy to deploy. It also makes Shalon easier to understand for normal users, eases protocol reviews, and increases the chance of having several implementations of Shalon available. In this work, we provide a description of the design and implementation of Shalon, a performance and anonymity analysis, and a discussion on the scalability properties.

Keywords: Anonymous Communications, Privacy Enhancing Techniques, Confidentiality

I. Introduction

Many approaches have been proposed to provide anonymity on the network layer. Still, only some of them have reached wide scale deployment, e.g., [1], [2]. Currently, the most popular and widespread system is Tor [2]. Tor, as well as other approaches for anonymization, relies on non-standardized and complex protocols. On the one hand it makes sense to develop a dedicated protocol for the purpose of anonymization due to its special requirements. On the other hand this approach has also disadvantages. First, the development effort for practical implementations is high. This often leads to the existence of at most one single implementation, which in turn runs the risk of creating so-called *software monocultures*. In this case, failures in the single implementation can paralyze the whole network, possibly compromising the overall security. Second, if the underlying protocol specifications are changed or updated often, additional implementations are even more difficult to maintain. Additionally, it is naturally more difficult to analyze the properties of a complex system.

Due to the increase of multimedia content transmitted over the Internet, bandwidth requirements have been drastically increased. This also holds for anonymization networks, where quality of service is becoming an increasingly more problematic aspect. While the backbones and access networks of the Internet could be adjusted to the increased needs, current anonymizers have not been able to meet these demands, thus leading to a decrease in network performance. Reasons include the necessity to handle multiple layers of encryption, a geographicaly spread routing, and limited over-all bandwidth. The implications of a poor performance have been shown in [3]: the user base of a network drops linear with increasing latency. Several papers have dealt with the performance analysis of anonymization systems [4], [5] and have shown the need and possibilities for better quality of service [6], [7].

In this paper, we address the aforementioned problems by introducing Shalon: a novel lightweight and efficient anonymization technique purely based on standardized protocols. We provide a performance evaluation and an anonymity analysis of Shalon. In Section III, we show that the level of anonymity offered in Shalon can be compared to the level of anonymity in Tor. Further, Section IV shows that our implementation of Shalon is superior to Tor in terms of performance and scalability (at least in the laboratory settings in which the tests were conducted). Section V and VI provide a discussion on some of our design choices and outline possible future modifications of Shalon, respectively. Finally, Section VII concludes the paper.

II. Related Work

This section gives a brief overview of techniques used for network layer anonymization.

The typical method for anonymizing network traffic is to send messages on a detour through several *middle nodes* rather than directly to the recipient. This can be done in a way such that the relaying nodes cannot determine for certain whether the relayed data streams originate from the predecessor or are forwarded on behalf of other users. In this paper, we focus on low-latency anonymization networks. These are designed for

real-time communication, like web-browsing or instant messaging. The oldest representative in this category is the *single hop proxy* approach which is very lightweight. One widely known implementation in this category is *anonymizer.com*. However, single hop proxies provide an unavoidable single point of failure and trust. Therefore, they pose no solution for users with a higher demand for protection. We divide the remaining approaches into the following three categories:

1) Layered Encryption Approaches: A typical representative of this approach is Tor. It is an overlay network consisting of servers that are called *onion routers* (ORs). To anonymize Internet communications, end-users run an *onion proxy* (OP) on their computer that is listening locally for incoming connections and redirects TCP-streams through the Tor network. When sending out the redirected TCP-streams, the OP constructs *circuits* of encrypted connections through a path of randomly chosen ORs. A Tor circuit, as default, consists of three ORs, where each OR only knows *(i)* which peer has sent him data (the predecessor) and *(ii)* to which peer he is relaying data (the successor). A circuit length of three constitutes a reasonable trade-off between security and performance, where the role of the middle OR is to hinder the last OR in the circuit (the *exit node*) to learn the identity of the first OR (the *entry node*). If the latter two cooperate, users can be deanonymized.

During circuit creation in Tor, Diffie-Hellman key exchanges are used to establish shared symmetric session keys with each OR in the circuit. The user's OP encrypts all traffic before it is sent over the circuit, using these keys in reverse order, starting with the key of the last OR. Upon receiving traffic, each OR on the circuit removes (or adds, depending on the direction) one layer of encryption while relaying the data to the next OR, so only the last OR (the exit node) knows the actual destination of a TCP stream.

AN.ON [1] (also known as JAP or Jondonym) is also based onion routing. One of the main differences to Tor is that users cannot choose the circuit freely between the relays. In AN.ON normally three relays are forming a predefined path, denoted a *cascade*. The user only has the possibility to choose one of the predefined cascades. Problems with AN.ON include missing perfect forward security[1] and unknown scalability behaviour.

Tarzan [8] and MorphMix [9] are two peer-to-peer (P2P) based anonymization techniques for implementing onion routing. Unlike the earlier approaches, a MorphMix node does not have to have knowledge about all other MorphMix nodes in the network. For the circuit setup, so-called *witness nodes* are used to facilitate the selection of nodes for circuit extension. In Tarzan every node has a set of peer nodes for exchanging cover traffic which are called *mimics nodes*. Nodes select their mimics in a pseudo-random universally verifiable way. Neither Tarzan nor Morphmix are in active use today.

Lastly, the Invisible Internet Project (I2P) system[2] is an approach that makes use of so-called *garlic encryption*, that is, a variant of onion encryption where multiple messages wrapped into a single "garlic message", encrypted with a particular public key. Problems with I2P include missing transparency for their network layer protocol and a complete lack of academic coverage.

2) Simple Randomized Routing Protocols: Crowds [10] is an alternative to the techniques described above. When, for example, a user requests a web page, the request is sent to another (randomly chosen) crowd member (called a *jondo*). By making a biased coin toss, this jondo decides whether to forward the message to the final destination or to another randomly chosen jondo. Communications between jondos are link encrypted, meaning that each jondo can see the content of passing messages, including the address of the final destination (they cannot easily determine the initiator though). The GNUNet system [11] also makes use of a simple randomized routing protocol, where the forwarding of messages on behalf of the other nodes (here denoted the *"indirection"*) depends, among other things, on the load in the network. Crowds never left the stage of a research implementation, and further Crowds and GNUNet offer fairly weak protection against strong attackers [12], [13].

3) Multicast/Broadcast Based Methods: A classic proposal in this category are DC-networks [14]. DC-networks can provide "perfect anonymity" (in an information theoretic sense), however under some rather demanding assumptions, as it is required that all DC-net nodes must communicate with all other nodes for every message transfer. Thus, secure, reliable and fast broadcast channels are prerequisites for a practical realization of a DC-network. Furthermore, the protocol is prone to channel jamming, inefficient in large networks, etc. [15]. P5 [16] is another approach in this category that aims to remedy the scalability problems of DC-networks by dividing the network group into a tree hierarchy containing smaller broadcast groups. Also Herbivore [17] try to improve scalability by combining an approach based on DC-networks with a hierarchical topology in which the users are grouped into smaller subsets (so-called *cliques*). However, all of these networks are either not used anymore or do not have a widespread user base.

III. Introducing Shalon

In this section, we introduce our proposed anonymization technique called "Shalon". We begin by identifying our assumptions and the targeted attacker model. The second part provides a protocol description.

A. Assumptions and attacker model

We start by listing our basic assumptions.

- Our first assumption is that adversaries cannot break cryptographic primitives. This is a standard assumption in the area of anonymous communication;
- Our second assumption is that we do not aim to protect against a (passive or active) global adversary. This is because such protection reduces the provided level of

[1]Forward secrecy ensures that a session key created from a set of long-term asymmetric keys will not be compromised if the long-term private key is compromised in the future.

[2]See http://www.i2p2.de/ for more information.

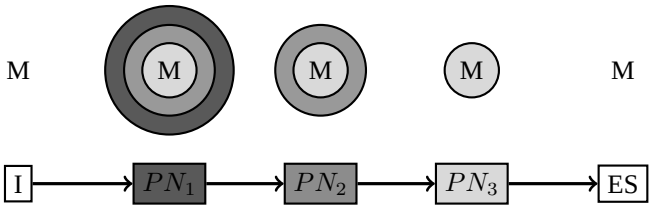


Fig. 1. Onion encryption for a tunnel of length three

performance and usability drastically. Also, it has been shown that most users prefer performance rather than very strong anonymity [3]. This is underlined by experiments indicating that the majority of users are only willing to wait up to four seconds when requesting a web page [4];
- Finally, we also assume that a Shalon client knows the list of nodes participating in the network.

Thus, our adversary constitutes an entity with the following capabilities:

- Passively observe some portion of network traffic;
- Actively operate its own nodes and/or compromise some fraction of honest nodes;
- Actively delete, modify and generate messages.

Inspired by the practical attacker classification in [18], this means that we aim to defend against corrupt end servers, local passive and active attackers (such as system administrators and small Internet service providers), and corrupt Shalon nodes.

B. Protocol Description

The wide success of HTTP and SSL constitutes a great incentive to base Shalon on these popular, standardized, and widely deployed protocols. Therefore, Shalon implements an onion encrypted HTTP-based tunneling method. This is done by using the HTTP CONNECT method/command. The CONNECT method extends a connection in the following way: it instructs the HTTP server or proxy to extend the connection by making a TCP connection to some specified server and port and relays the data transparently back and forth between this connection point and the client connection [19].

Anonymization tunnels are created in Shalon in the following way. After extending the connection from one Shalon node to a new node, a TLS handshake[3] is performed between the initiator and the new node. This enables the client to encrypt messages in an onion-routing like manner (see Figure 1) to achieve confidentiality of the next hop and application layer data. In Figure 1, I denotes initiator, ES denotes end-server, and M is the message sent along the circuit passing the proxy nodes denoted PN_i. The level of greyness of the proxy node corresponds to the key shared between the proxy node and the I. The encryption layers are produced with the corresponding keys. Due to the layered encryption, each node along the tunnel knows only its direct predecessor and successor in the tunnel. As in Tor, the default tunnel length of three hops is chosen as it constitutes a reasonable trade-off between performance and security.

[3]Although we use "SSL" and "TLS" as synonyms in this paper, we are aware of the differences between these protocols.

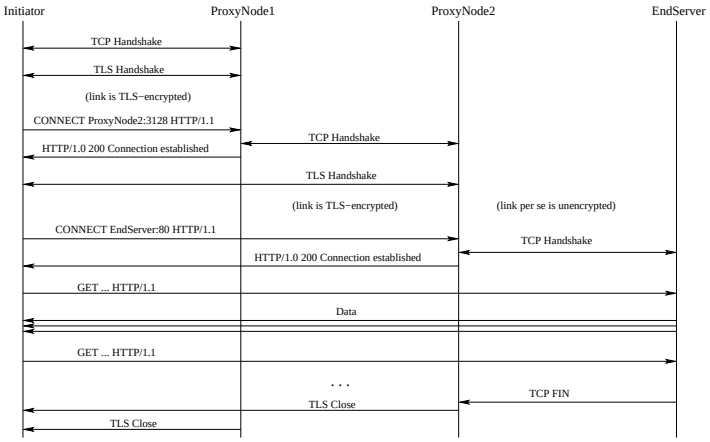


Fig. 2. Protocol run for a scenario where the initiating user builds a two-hop tunnel, fetches web site, and then closes the connection

The process of circuit establishment is depicted in Figure 2. For the sake of clarity, a circuit of length of two is used here. After establishing the encrypted connection with the PN_1, the circuit is extended to PN_2. TLS is used to implement the layered encryption. Finally, the connection is extended to the ES. Here, ES is a web server and it exchanges information using HTTP. Such tunnels, however, can be used for the transmission of arbitrary data on top of TCP.

C. Observations

Shalon relies on widely adopted standardized protocols for implementing onion routing. Any CONNECT-capable HTTP proxy with TLS/SSL [20] support is sufficient to serve as a base for an intermediary Shalon node, such as the widely used proxy server Squid with an enabled SSL-layer. While the idea of tunneling TCP-connections through a single web proxy has been described as early as [21] (maybe even before), implementing onion routing based on this principle, to the best of our knowledge, has not been proposed before. By using standardized protocols rather than proprietary and/or non standardized protocols, we achieve the following advantages:

- The availability of existing libraries eases the development process;
- The existing libraries are mature and thus well tested and scrutinized;
- There exists a sound and widely communicated body of knowledge regarding the functionality of these protocols;
- Since these protocols are simple and lightweight, they (and thus Shalon) can be easily studied in respect of their advantages and drawbacks.

Further, the low complexity of Shalon contributes to the following:

- Shalon can be easily understood by "the average users", not only by a small group of expert developers;
- Shalon is easier to implement than other approaches;
- Shalon and its properties are easier to evaluate;

- It is more likely that there will be a variety of different clients/servers;
- The risk of ending up with a software monoculture (where failure in a single implementation can, e.g., paralyze the whole network) is greatly reduced.

Our proposal, still, is not a silver bullet. Below, we list some observations regarding the design choices in Shalon:

- In Shalon, contrary to, e.g., Tor, packets are of variable size which may decrease the protection against traffic analysis;
- Shalon uses only one level of encryption between Shalon nodes, whereas in Tor packets are "double encrypted" both on cell level and on TLS level between each node;
- Each tunnel in Shalon only handles one connection per tunnel, in contrast to Tor circuits, which can handle multiple TCP streams in parallel;
- It is possible to quickly recreate recently used tunnels by employing SSL-key reuse.

In Section V, we further discuss our design choices and their implications.

D. Tunnel Establishment

In this section, we address the issue of tunnel establishments in Shalon. The user's client always maintains a few general purpose tunnels in a preemptive way, i.e. before they are required by any application. This saves tunnel build-up time, which constitutes the most significant delay when the anonymous communication channels are requested by the applications [5]. Also, by applying SSL reuse to reestablish recently used tunnels, we remedy the lack of stream multiplexing and a tunnel truncation after the connection to the end server is closed. SSL reuse, however, should only be used for a limited period of time and a limited data volume per tunnel. Tunnel rotation helps to strengthen the protection against long-time profiling issues and to achieve forward secrecy.

IV. Analyzing Shalon

In this section we provide an analysis of the most important issues with Shalon.

A. Anonymity and Security

This section gives an overview of most important security properties of Shalon. We have analyzed the perspectives of sender anonymity and relationship anonymity (on the network level) between the sender and the recipient.

It can be trivially shown that an entity which is unrelated to the network, i.e. does not operate a server node or participate as a client, and in addition has no physical relation to either communicating parties will neither learn the identity of the sender, nor the communication relationship.

The *recipient of the message* has only very limited chances to identify the true sender of the message he receives. However, it might be possible to use application level attacks, like profiling application layer data, injecting active content into HTML-pages in order to trick the sender's browser to bypass the proxy settings, or just set long-term cookies and hope that the sender will come back to the site without using an anonymization technology [22]. However, all of these techniques are independent of the anonymization technology being used, and thus Shalon is neither more nor less resistant to them compared to other anonymization networks.

A *local administrator* and the *first node* on the path have both several possibilities to learn the relationship between sender and receiver. First, there are denial-of-service attacks which block access to the anonymizing technology and thus force the sender to stop communicating or reveal his peer by communicating in plain. However, all major anonymizing techniques are susceptible to these kinds of attacks. Second, this type of attacker can use a database of traffic fingerprints to identify the website that the Shalon user visits (if the website's fingerprint is contained in the database) [23], [24], [25]. This attack might be more successful against Shalon compared to other anonymization techniques with fixed cell sizes because of variable packet sizes and visibility of new stream creations. However, this is a subject of current research and we expect Shalon to have similar susceptibility as e.g. Tor.

Colluding nodes can to a certain extend mount all previously listed attacks. They have the potential to be either a first and/or the last node in the tunnel (with respect to a certain user). However, the most serious case is when an attacker happens to occupy multiple nodes in a single tunnel; most notably – the first and the last position. In this case an attacker can mount traffic confirmation attacks by comparing input streams to output streams [26], by trying to identify the tunnel of a stream through the network [27], [28], or even other techniques [29]. To the extent of our analysis, Shalon is not more vulnerable against these attacks than any other practical approach.

Today, it is generally considered that an attacker with powers equal to those of *governments* (or stated in other words, a *passive global observer*) is able to break all currently available (and practical) low latency anonymization techniques. However, please note that it is not a design goal of Shalon to protect users against this kind of adversaries.

To sum up, as a preliminary conclusion we have showed that Shalon provides a similar level of protection as other comparable low-latency anonymization techniques.

B. Scalability

We estimate the scalability properties regarding the process of traffic anonymization to be in the same scale as in Tor. However, there are several advantages with Shalon that are explained below.

As previously mentioned, Tor encrypts everything twice: there is a TLS layer between the nodes as well as cryptography on the circuit layer between the client and corresponding ORs. Thus, each intermediary Tor node has to decrypt the incoming TLS message, then the corresponding onion layer, and finally encrypt the message on the TLS layer for the next node in the circuit. This is done by all Tor nodes on the path except the last one which does not do the final encryption step. Therewith, there are $(3l-1)$ encryption/decryption operations per circuit that have to be performed by the servers in Tor

circuit (where l is the circuit/tunnel length). In Shalon there are less cryptographic operations, as they are only between the client and the corresponding Shalon nodes. Hence Shalon only requires l operations on the servers' side instead of $(3l-1)$. This significantly reduces the amount of encryption/decryption operations performed by the intermediary nodes. Therefore, we expect Shalon to be able to serve more clients with the same amount of CPU power. Alternatively, given the same number of users, more CPU power per user should be available in Shalon (the CPU saturation is suspected to be the limiting factor for most Tor nodes [5]).

C. Performance Evaluation

In this section we present the results from the performance analysis of Shalon. The main focus of the analysis is data throughput, round trip time (RTT), and tunnel buildup time. Furthermore, we compare our results with the Tor network.

1) Experimental design: To provide a fair comparison, we performed our experiments in a laboratory under controlled and optimal conditions. In our laboratory environment, we have set up a private Tor and Shalon network (using the same machines) with the following experimental setup:

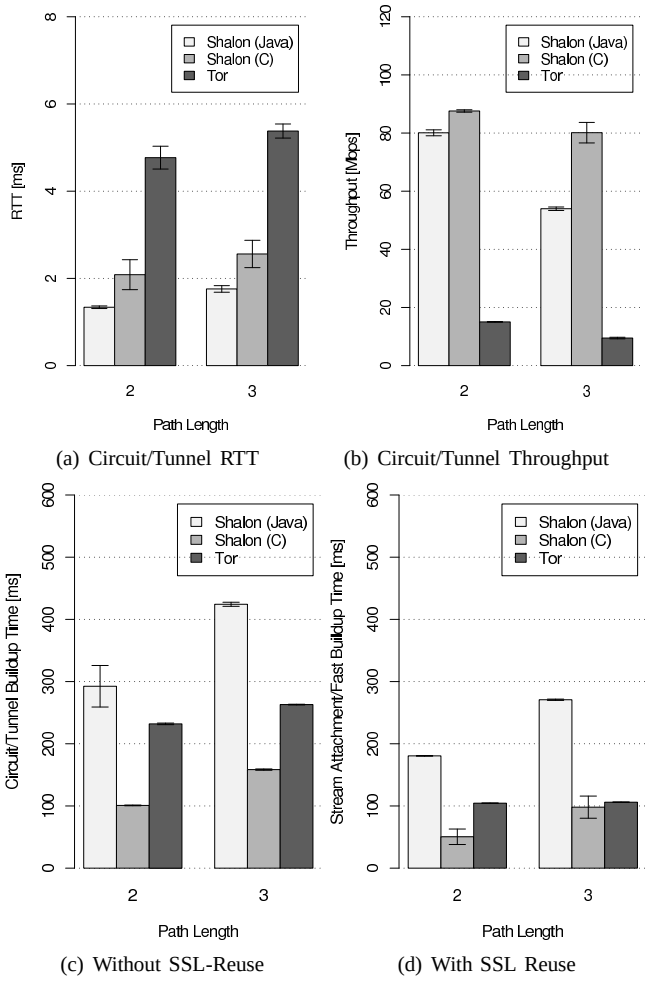
- Intel Pentium III Dual Core machines with 1 GHz CPUs and 2 GB RAM acting as nodes;
- Intel Core 2 Duo with 2.4 GHz and 2 GB RAM acting as client;
- Local network backbone bandwidth of 1 Gbps;
- The vanilla Tor implementation in the version *0.2.0.12-alpha* written in C (both as OP and ORs);
- Our Java and C versions of Shalon client.

The Shalon server (compare with OR in Tor) is built using the proxy software Squid (with enabled SSL). The end-server for both, Shalon and Tor is realized by a short and efficient Perl script running on the last node of the path.

Note that as both the clients and servers of Shalon and Tor were run in a private network, the result from our tests cannot be compared to performance evaluations of the real Tor network [4], [5], as the transmission and propagation delay in the later case are much larger due to bandwidth limitations and the geographical distances between nodes. Our tests, on the other hand, evaluates the best case for performance, that is, the upper limits. In the future, it would be possible to set up new tests where we emulate a limited bandwidth and geographical distance between the nodes in the laboratory network, for instance by using *dummynet*[4] or performing the measurements in the PlanetLab [30]. To calculate the mean values with a 95% confidence interval, each experiment was repeated 20 times.

2) Results: Figure 3(a) shows the *round trip time* of the circuits/tunnels of length 2 and 3, i.e. with 2 or 3 intermediary hops. We used these numbers because layered encryption approaches based on onion routing normally use this number of hops as a standard path length. Shalon outperforms Tor by a factor of 2 to 3 in terms of latency.

[4]See http://info.iet.unipi.it/~luigi/ip_dummynet/



(a) Circuit/Tunnel RTT (b) Circuit/Tunnel Throughput

(c) Without SSL-Reuse (d) With SSL Reuse

Fig. 3. Tor vs. Shalon

Figure 3(b) shows the data throughput of Tor and Shalon depending on path length. Both approaches use AES in CBC mode as a symmetric encryption algorithm. In terms of throughput, Shalon outperforms Tor by the factor 6. In Shalon (Java) a limiting factor was the client CPU for the case with tunnel length 3. The limiting factor for Tor was the ORs' CPUs which always had a 100% CPU load independent of circuit length. In the other cases we did not observe any obvious bottlenecks.

Because of the differences between circuit/stream concept in Tor and tunnels in Shalon it is difficult to directly compare these both approaches in terms of circuit/tunnel setup times. Recall that Tor uses circuits to tunnel various TCP streams over them. Shalon, in contrast, uses a single tunnel for each TCP connection.

To enable a comparison of both networks in this regard, we distinguish two different scenarios. In the first scenario we assume that both systems have no a-priori connection reservoirs. For Tor, this means that a new circuit must be built and a stream has to be attached to that circuit. Shalon also needs to buildup a new tunnel, but has no SSL session which could be reused. The second scenario covers the situation when

a connection is already established over the circuit/tunnel. Therefore, we assume that Tor already has an existing circuit and merely needs to attach a stream to that circuit. For Shalon this means that path exists that have recently been used for a tunnel. Due to this, every node in the path has valid SSL session keys which can be reused. Shalon now needs to create a new tunnel over the same nodes using SSL reuse.

Both states are depicted in Figures 3(c) and 3(d). In the first scenario (3(c)), Tor slightly outperforms the Java version of Shalon in the case of a 3 hop circuit, but the C version of Shalon is faster to the factor of $\frac{1}{3}$. In the second scenario, 3(d), Shalon in Java is actually about thrice as slow as Tor. The C version of Shalon, however, is still twice as fast as Tor in the case of a path length of 2. In case of a path length 3 the C version is as fast as Tor.

An interesting observation is that even in the case when Shalon needs to provide a new connection by reestablishing the whole tunnel, the procedure is as time consuming (in the C implementation) as merely attaching a stream to a circuit in Tor. Thus, from the performance point of view, "one way" usage tunnels in Shalon (because of the need to be compatible with the HTTP protocol) cannot be seen as a disadvantage compared to Tor.

All in all, this comparison shows that Shalon provides significantly improved performance compared to the state of the art anonymization network Tor. This performance gain is especially significant regarding data throughput. The main reason is the reduced complexity of the protocol, less encryption tiers and the use of a mature and performance tuned node software (Squid). However, the most significant impact on the performance is due to the lower number of de-/encryption operations needed for processing data packets. Our tests show, that the latter lowers the CPU load drastically.

Note that as both Shalon and Tor employ the cipher *TLS_DHE_RSA_WITH_AES_128_CBC_SHA*, the differences in the results are not caused by the used cipher.

V. Discussion

In this section, we motivate our design choices and discuss open issues in Shalon.

A. Single vs. Double Encryption

While Tor encrypts data on both the link layer and the circuit layer, Shalon encrypts on the tunnel layer (equivalent to circuit layer in Tor) alone. This leads to a significant performance increase as showed by the measurements in the previous section. The downside of this is that there is an additional information leakage: a passive observer is able to recognize the creation of a new anonymization tunnel as well as to link the amount of transferred data to a specific connection. Even though this is an additional information leakage compared to Tor, we believe that this only eases existing attacks against Shalon to still tolerable threshold and does not introduce new attacks. Thus, we claim that this does not significantly change Shalon's threat model and provides anonymity in the same situations as Tor does. However, this is subject to further research. In case that new vulnerabilities are found, countermeasures can be developed even without making use of an additional encryption layer. E.g. a protocol on top of the anonymization tunnel could be introduced which supports multiplexing of various streams. From a property point of view, this would make the approach similar to Tor, but would not decrease the performance due to a twofold encryption: Shalon would still encrypt on the tunnel layer only. Note that this multiplexing would only protect streams starting from the same node.

B. Packet Padding

Tor's packet padding scheme achieves a constant packet size by extending every cell to 512 Bytes. The purpose of padding in Tor is twofold. First, the developers originally wanted to introduce active *mixing* (delaying and batching) in a future generation of the protocol; fixed size packets are an unconditional requirement for mixing functionalities. However, today it is unlikely that mixing will be introduced in low-latency anonymizing networks like Tor. Even though AN.ON supports mixing, it is not enabled by default. The performance of these approaches is often below the acceptance level of average users ([3], [4]), even without mixing. Second, padding[5] hinders several attacks, as discussed below.

With respect to a global attacker, padding obscures a packet's path through a network, which could be traced due to its unique size. For similar reasons it also hampers end-to-end traffic confirmation attacks. However, even if padding is used, there is also other information available (like timing, data volume) that enables end-to-end traffic confirmation attack [26], [31]. Moreover, it is a widely accepted fact that Tor does not protect against an adversary who can observe both the first and the last node in a path [26], [32], [33]. A global adversary is also out of the scope of Tor's attacker model.

Padding has also been suggested as a means to circumvent fingerprinting attacks [25]. These attacks can be mounted by a local adversary – one of the main protection goals of Tor's attacker model. Increased protection against this adversary, from our point of view, provides an incentive to apply packet padding. To the best of our knowledge, however, the necessity of a constant packet size with respect to fingerprinting attacks has so far not been demonstrated in practice. Moreover, as fingerprinting attacks require a-priori knowledge about the content in order to be successful, they are hardly possible when dealing with unknown traffic patterns (e.g. secret data which only needs to be transfered once) or on the data without specific unique characteristics[6] (e.g. VoIP, audio/video/bulk data[7]).

Shalon does not apply padding mainly due to the following reasons: first, as discussed above, the effectiveness of padding

[5]Please note that we do not refer do link padding but rather to packets' padding.

[6]Please note that we are interested in detecting the *content* of the communication and not the *type* of content.

[7]The effect of padding is especially marginal for bulk data, where only the last packet is padded.

with respect to the mentioned attacks is questionable. Second, in case of transferring many small messages/objects, packet padding produces a lot of overhead, and, thus, leads to a performance loss. Finally, it is not a trivial task to integrate padding in Shalon. One possibility would be to use the padding mechanism of TLS [20], where padding up to 255 bytes is possible. Unfortunately most TLS implementations only support the minimal block padding required by a cipher. Another possibility would be to introduce a proprietary protocol on top of an anonymization tunnel. Due to the fact that a part of the motivation of Shalon is the use of standardized protocols, we want to avoid these measures as long there is no hard evidence that padding is an effective countermeasure against the aforementioned or new attacks.

C. Circuits vs. Tunnels

As discussed above, using one anonymization tunnel per TCP connection leaks more information than tunneling various streams through a circuit (see Section V-A). On the other hand, it also leaks less information; an exit node in Tor can link all streams from a single circuit to the same originator – if there is no circuit, the connections can not be trivially linked. Third, if a TCP connection between two servers breaks or looses packets, only a single tunneled connection is affected. In Tor that would cause a delay or connection drop on all the streams involved in the circuit. Finally, in most cases Shalon requires higher number of active file descriptors used by each node, as compared to a system like Tor. To which extend this limits the scalability and how this problem could be mitigated is subject to future research.

To summarize, it would be nice to have the option of multiplexing connections, but it is also possible to provide a reasonable anonymization service without it.

D. Use of Standardized Protocols

The use of standard protocols and open architectures in Shalon facilitates extensibility, interoperability, and portability. All in all, Shalon achieves onion routing in a simple yet elegant way, using only existing standardized protocols. Our proposal, however, is not a silver bullet. Standardized protocols cause flexibility restrictions since the anonymization protocols have to follow predefined flows. Side effects which can arise due to these restrictions must be carefully considered.

The advantage of standardized protocols also justifies the following example: the Tor developers currently discuss how to modify the Tor protocol to look more like a standard TLS-connection in order to achieve a higher blocking resistance. Shalon will obviously never suffer from similar concerns.

Altogether, we believe that the use of existing standardized protocols in the area of anonymous communication is an interesting idea. Its possibilities, implications, as well as restrictions, need to be researched in a greater detail.

E. Choice of Underlying Protocol

The main reason for choosing HTTP as the basis for Shalon is because of the following four properties of HTTP. First, HTTP is one of the most important and well known network protocols in the world. Therefore, a multitude of different well tested libraries exist. These libraries can be used for the development of new clients. Second, HTTP can easily be combined with TLS/SSL to protect the content of a connection. Hence, different HTTP proxy implementations with TLS/SSL support can be found, which could be employed for the anonymization process in Shalon. Third, HTTP would possibly allow to transfer directory information (status and contact information about the nodes) in-band of the tunneling protocol. Finally, HTTP allows proxy connections to other servers with the help of the HTTP CONNECT method. This is one of the most important requirements for multi hop tunnels.

A downside of HTTP is missing possibility of multiplexing several CONNECT-requests over a single connection. However, to the best of our knowledge there is no common network protocol which supports multiplexing and also possesses the above mentioned properties.

Due to the described reasons we found HTTP to be best choice to build Shalon on top of. Also, a positive side effect with using HTTP and a corresponding proxy server is the caching of HTTP objects. Instead of using the CONNECT command of the HTTP protocol at the last node on path, the user can also issue the GET request to retrieve HTTP objects. In case that objects are already cached at the proxy server, the requests can be served right away without fetching the objects again. Therewith some attacks like the low cost traffic analysis [27] could be made more difficult.

VI. Future Work

In this paper we did not discuss every aspect or solve every problem regarding Shalon, and thus there are some issues that we have left as future research.

First, some features like the padding of packets are necessary to enable some degree of protection against global attackers, but still, to the best of our knowledge, padding have not been proven effective against a local attacker. Therefore more research is required to demonstrate the implications of padding against a local attacker.

Second, Shalon leaks more information than Tor. This raises the question how this influences the provided anonymity of Shalon. We claim that this does not change Shalon's attacker model, but this has not been fully proven yet.

Other aspects which need more research include, for example, the provision of hidden services in Shalon (possibly using the SOCKS protocol). Also, it is an open question how to integrate multiplexing of connections or padding without introducing proprietary protocols. Finally, having a variety of clients may contribute towards easier profiling, which should be considered in the clients' designs.

VII. Conclusions

In this paper we introduced Shalon – a simple, scalable, and innovative low-latency anonymization technique purely based on open standards. It makes use of out-of-the-box nested TLS connections to achieve a simple and elegant version of onion

routing. We described how it achieves privacy and scalability in open environments and evaluated its performance. The key feature of Shalon is the buildup of anonymous communication on top of the HTTP/SSL protocol suite. Because of the use of standardized protocols we were able to develop two performance competitive implementations within a short period of time. Due to the restrictions of the used standardized protocol our approach renonces fixed packet sizes as well as multiplexing of different streams through a single circuit.

The main objective of Shalon was to provide similar protection as Tor without the use of proprietary protocols. Due to its performance Shalon is well suitable for applications with a high demand for bandwidth.

Because of its simplicity we envision the availability of different independent implementations of Shalon.

REFERENCES

[1] O. Berthold, H. Federrath, and S. Köpsell, "Web MIXes: A system for anonymous and unobservable Internet access," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, H. Federrath, Ed. Springer-Verlag, LNCS 2009, July 2000, pp. 115–129.

[2] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *Proceedings of the 13th USENIX Security Symposium*, 2004.

[3] S. Köpsell, "Low Latency Anonymous Communication - How Long Are Users Willing to Wait?" in *ETRICS*, ser. Lecture Notes in Computer Science, G. Müller, Ed., vol. 3995. Springer, 2006, pp. 221–237.

[4] R. Wendolsky, D. Herrmann, and H. Federrath, "Performance Comparison of low-latency Anonymisation Services from a User Perspective," in *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, N. Borisov and P. Golle, Eds. Ottawa, Canada: Springer, June 2007.

[5] A. Panchenko, L. Pimenidis, and J. Renner, "Performance Analysis of Anonymous Communication Channels Provided by Tor," in *Proceedings of the Third International Conference on Availability, Reliability and Security (ARES 2008)*. Barcelona, Spain: IEEE Computer Society Press, March 2008.

[6] R. Snader and N. Borisov, "A Tune-up for Tor: Improving Security and Performance in the Tor Network," in *Proceedings of the Network and Distributed Security Symposium - NDSS '08*. Internet Society, February 2008.

[7] S. J. Murdoch and R. N. Watson, "Metrics for security and performance in low-latency anonymity systems," in *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, N. Borisov and I. Goldberg, Eds. Leuven, Belgium: Springer, July 2008, pp. 115–132.

[8] M. J. Freedman and R. Morris, "Tarzan: A Peer-to-Peer Anonymizing Network Layer," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.

[9] M. Rennhard and B. Plattner, "Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.

[10] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for Web Transactions," *ACM Transactions on Information and System Security*, pp. 66 – 92, April 1998.

[11] K. Bennett and C. Grothoff, "GAP – practical anonymous networking," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*, R. Dingledine, Ed. Springer-Verlag, LNCS 2760, March 2003, pp. 141–160.

[12] M. Wright, M. Adler, B. N. Levine, and C. Shields, "The predecessor attack: An analysis of a threat to anonymous communications systems," in *ACM Transactions on Information and System Security TISSEC'04*, vol. 7 (4). ACM Press, November 2004, pp. 489 – 522.

[13] A. Panchenko and L. Pimenidis, "Crowds revisited: Practically effective predecessor attack," in *Proceedings of the 12th Nordic Workshop on Secure IT-Systems (NordSec 2007)*, Reykjavik, Iceland, October 2007.

[14] D. L. Chaum, "The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability," *Journal of Cryptology*, no. 1, pp. 65 – 75, 1988.

[15] J.-F. Raymond, "Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, H. Federrath, Ed. Springer-Verlag, LNCS 2009, July 2000, pp. 10–29.

[16] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, "P5: A protocol for scalable anonymous communication," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, May 2002.

[17] S. Goel, M. Robson, M. Polte, and E. G. Sirer, "Herbivore: A Scalable and Efficient Protocol for Anonymous Communication," Cornell University, Ithaca, NY, Tech. Rep. 2003-1890, February 2003.

[18] A. Panchenko and L. Pimenidis, "Towards Practical Attacker Classification for Risk Analysis in Anonymous Communication," in *IFIP CMS 2006, Communications and Multimedia Security*, Heraklion, Greece, October 2006.

[19] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol: Http/1.1," Internet Engineering Task Force: RFC 2616, June 1999.

[20] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," Internet Engineering Task Force: RFC 2246, Januar 1999, http://www.ietf.org/rfc/rfc2246.txt.

[21] A. Luotonen, "Tunneling tcp based protocols through web proxy servers," IETF Internet Draft, August 1998.

[22] T. G. Abbott, K. J. Lai, M. R. Lieberman, and E. C. Price, "Browser-Based Attacks on Tor," in *Privacy Enhancing Technologies*, 2007, pp. 184–199.

[23] A. Hintz, "Fingerprinting Websites Using Traffic Analysis," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)*, R. Dingledine and P. Syverson, Eds. Springer-Verlag, LNCS 2482, April 2002.

[24] G. D. Bissias, M. Liberatore, and B. N. Levine, "Privacy Vulnerabilities in Encrypted HTTP Streams," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2005)*, May 2005, pp. 1–11.

[25] M. Liberatore and B. N. Levine, "Inferring the Source of Encrypted HTTP Connections," in *Proceedings of the 13th ACM conference on Computer and Communications Security (CCS 2006)*, Alexandria, Virginia, USA, October 2006, pp. 255–263.

[26] L. Øverlier and P. Syverson, "Locating hidden servers," in *Proceedings of the 2006 IEEE Symposium on Security and Privacy*. IEEE CS, May 2006.

[27] S. J. Murdoch and G. Danezis, "Low-Cost Traffic Analysis of Tor," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. Oakland, California, USA: IEEE Computer Society Press, May 2005.

[28] G. Danezis and R. Clayton, "Route fingerprinting in anonymous communications," in *Peer-to-Peer Computing*, A. Montresor, A. Wierzbicki, and N. Shahmehri, Eds. IEEE Computer Society, 2006, pp. 69–72.

[29] N. Hopper, E. Y. Vasserman, and E. Chan-Tin, "How Much Anonymity does Network Latency Leak?" in *Proceedings of the 14th ACM Conference on Computer and Communications Security (ACM CCS)*, Alexandria, Virginia, USA, October 2007.

[30] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, 2003.

[31] G. Danezis, "The traffic analysis of continuous-time mixes," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, ser. LNCS, vol. 3424, May 2004, pp. 35–50.

[32] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-Resource Routing Attacks Against Anonymous Systems," University of Colorado at Boulder, Tech. Rep. CU-CS-1025-07, February 2007, http://www.cs.colorado.edu/department/publications/reports/docs/CU-CS-1025-07.pdf.

[33] S. J. Murdoch and P. Zielinski, "Sampled Traffic Analysis by Internet-Exchange-Level Adversaries," in *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, Ottawa, Canada, June 2007. [Online]. Available: http://www.cl.cam.ac.uk/~sjm217/papers/pet07ixanalysis.pdf