Sending unobservable messages across the internet

Martin Gwerder

Abstract—In this paper we intoduce the MessageVortex protocol. A unobservable message annonymisation protocol based on zero trust and a distributed P2P architecture without central aspects. It scores over existing work by blending its traffic into suitable existing transport protocol thus making it next to impossible to block it without affecting significantly normal users of the transport medium. It furthermore requires no protocol specific infrastructure and allows a sender to control all aspects of a message such as degree of anonymity, timing, redundancy of the message transport without disclosing any of these details to the routing or transporting node.

Index Terms—Data privacy, Message systems, Anonymity, Security

1 Introduction

Since whistle blower Edward Snowden disclosed documents it seems generally accepted that global monitoring of internet traffic is monitored. According to these documents (verified by NRC) NSA infiltrated more than 50k computers with malware to collect classified, or personal information. They furthermore infiltrated Telecom-Operators such as Belgacom to collect data, and targeted high member of governments even in associated states.

A normal message sent throughout the internet must, even when perfectly encrypted, disclose at least the recipient to the router transporting a message. The sender can be idenitified by the return path or is identifiable by following the source of packets. Meta information is valuable as frequency and message size disclose important facts about the association and intensity of relationship of the involved parties. Typical attacks are traffic capturing by network observation or by inserting one or more malicious nodes into a routing network.

This work addresses the above mentioned problems of message recording for either real time analysis or later processing by introducing a new protocol called MessageVortex. It furthermore addresses the leak of routing information most protocols have by applying a zero trust model to the whole network except the sending and receiving node. This protocol is able to sustain anonymity even under harsh assumptions such as an adverser possessing a huge founding and unlimited monitoring capability on the network and a huge amount of own nodes.

Numerous attempts such as in [1], [2], [3], [4], [5], [6] have been made to use relays [7], mixes [7], or DC-related-networks [8] to anonymize message flow. But most of them have problems as they rely at least on the partial trust to the nodes routing the messages, or some central infrastructures [9], [10], [11], [12]. Exit and entry points are important as they may leak information which is otherwise well hidden within the network. Additionally, a dedicated transport protocol is easy to block since their implementation can be easily identified by used ports or some protocol properties. Furthermore, most approaches require to have infrastructure with fixed addressing in the internet making owners of participating nodes easily identifiable and vulnerable.

All works analysed for this paper introduced a new transport layer solving these problems. In our approach we decouple the routing layer from the transport layer. By doing so we introduce new degrees of complexity to attack scenarios as messages may use any common transport protocol of the used network.

Our work consists of a routing layer which is completely P2P based without any central protocol specific infrastructure. Any node is a routing node and may be an endpoint. There is no implicit or explicit trust into any particular system of the network. Decoy traffic generation is controlled by the original sender of a message. Even a decoy traffic generating node is unable to differentiate between message and decoy traffic as a Solomon-Reed algorithm is used to blow the message up by adding redundancy information. This redundancy information may be decoy traffic or later required. The redundant blocks are always encrypted and a multitude of the cyphers block size. This fact makes it next to impossible to brute force the content.

As transport media we use common, well known store and forward based protocols. By doing so the protocol has no affiliation to the transport layer. Literally any free-mailer email address or chat account may be converted into a transport media for our protocol without any modification required on the server side. This makes the network very agile on one side at the cost of reliability as nodes may suddenly appear or disappear. To counter this phenomenon we are able to introduce a high degree of redundancy in our routes thus making the protocol stable again.

Using the MessageVortex protocol any device with a latent or permanent connectivity to the internet may act as routing node. It conceals its own traffic with the routed traffic. This is making it harder for any adverser to identify affiliated traffic.

By applying the zero trust model we give full control of all traffic to the original sender of the message. He controls message flow, redundancy, degree of anonymity, timing, and many more aspects of the message transport throughout the whole network. This is done without disclosing any of these parameters to the participating nodes.

To limit possibilities of DoS within the system and guarantee an efficient handling of messages, MessageVortex nodes (in short "node") rely on unlinked, ephemeral identities which are created in a proof of work system. While it is technically easy to use a node, it is hard to run traditional attacks against them. The amount of work required to disrupt services or do traditional attacks against the system grows significantly due to the non linear growth of calculation power required when maintaining more ephemeral identities. It is however still possible to exhaust external resources such as network bandwidth.

gwm
Mai 27, 2015

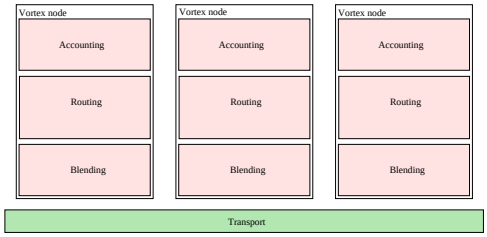

Fig. 1. Protocol stack.

2 Methods and Material

As shown in figure 1 we define the protocol on three different layers:

- A blending layer
  This is where messages are applied to the transport protocol.
- A routing layer
  This layer applies the logic to the message routing and prepares the message for the blending layer.
- An accounting layer
  This layer is a DoS and misuse protection. It keeps track of the transfers for each ephemeral identity and makes sure that queue and storage capacity are efficiently handled.

All three layers are connected through one or more store-and-forward based, common internet protocols. Protocols on this layer we refer as transport protocols. This layer is an unmodified, existing protocol layer on the internet.

All cryptographic operations such as encryption, decryption, hashing, or random number generation do not rely on a single algorithm. The protocol is able to signal what capabilities a node has and how exactly a message should be processed. This makes the protocol very robust if an used algorithm is broken. For this reason we defined for each capability at least two algorithms which are dependant on different mathematical problems (eg. RSA and EC as asymmetric encryption). This introduces a redundancy in algorithms allowing a user to swith it if required.

2.1 Protocol Layers

2.1.1 Transport

The transport layer provides the internet infrastructure. Unlike in most other approaches such as [4], [13], [14] this layer is not protocol specific. We use already existing, symmetrically built store and forward protocols. Attributes such as anonymity do not rely on the security of this layer.

By using this approach we remove the need for shaky technologies such as TCP or UDP hole punching to connect peer partners. It furthermore makes the use of "mostly connected" clients such as mobile phones or DSL connections suitable for this protocol as our transport endpoints are always within the global network connected and only the routing part may disconnect from time to time.

Protocols on this layer are typically well known and frequently used. They have no prerequisite for encryption or privacy and are store-and-forward based protocols with routing capabilities.

2.1.2 Blending

This layer is a translation-only layer and blends traffic onto the transport protocol. Protocol features such as anonymity or redundancy do not rely on this level. This layer embeds messages within the transport layer in such a way that an adverser is no longer able to identify VortexMessages from ordinary transport layer messages. Good blending is achieved if transport layer censorship measurements such as application level firewalls are be unable the difference between real world messages and MessageVortex messages. In an ideal application this applies to censorship applied by humans as well as censorship applied on the base of algorithms.

In a real scenario it is hard to achieve human proof censorship circumvention. If not done with care, problems as described in [15] arise. It is in our case not necessary as human censorship is very costly and slow compared to algorithms. For real time censorship as human censorship is too slow. Our transport layer is by definition frequently used for communication. We always consider an algorithm based censorship as existing.

At the moment the specification of this layer is limited to the two capabilities "embedd with offset" and "F5".

"Embedd with offset" is a plain embedding of a block in a file attached to a message. The offset allows to issue first a valid header of some sort in order to improve blending (eg. for a PCM encoded WAV file). While this is considered a very weak protection, analysis to detect such a file on a global transport scale is very demanding due to the sheer mass to be analysed.

"F5" means applying the F5 algorithm to hide a message within a random suitable JPEG image. "F5" is one of the very few steganographic works which have a real world implementation and attracted at least some interest in the research community. In [16] an approach to detect embedded information in steganographically modified images is presented. To obtain this information a considerable effort in terms of calculation power is required. This makes it impractical for real time censorship. It does furthermore only disclose the fact that F5 is being used. It does not leak the content of a message.

2.1.3 Routing

The routing layer is the mixer of the system. It processes messages received by the blending layer and is supported by the accounting layer. Any related set of messages is processed by the routing layer by recombining payload with operations defined in section 2.4. Due to the nature of these operations a node is unable to tell whether the traffic flow processed is decoy traffic or actual part of the message flow.

For a more precise working of the transport layer see section 2.3.

2.1.4 Accounting

The accounting layer protects a node from being overloaded or misused. Every sender must first apply for an ephemeral identity which is limited in lifetime prior to be able to route messages through a node. This is done by a proof of work (PoW) algorithm. The ephemeral identity is assigned with message and size transfer quotas. Any identity may apply as long as it is not expired for a raise of quota. It is up to the node to decicde whether this raise will be accepted with a new PoW puzzle or not at all. If rejected any sender might try to apply for a new ephemeral identity.

Due to the costs of maintaining multiple identities and their parental identities for anonymity of the original sender, the number of identities grow exponentially when growing a network of ephemeral identities. A sender might either introduce a new node to cut identity costs or maintain at higher identity costs a single node.

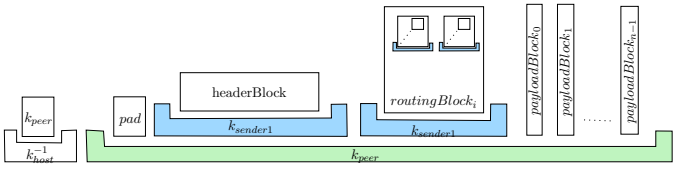

Fig. 2. Protocol block outline.

2.2 Protocol Outline

We define a protocol block which has an inner block structure as shown in figure 2.

These Blocks are passed from node to node. All blocks are binary proof which means that the same block sent twice will always result in exactly the same bit layout. There is no room for any adverser to tag the block without compromising the message. The message as a whole is replay protected. Routing and header blocks are linked with a chain secret to avoid hijacking of header or routing blocks.

2.2.1 Message Keys

Every protocol block is protected by two symmetric keys key_{peer} (in short k_p) , key_{sender} (in short k_s) and the private part of an asymmetric host key k_{host}^{-1} (in short k_h^{-1}). The public host key k_h^1 and both symmetric keys are known to the builder of the routing block structure.

This building is done by the sender. If using SURBs (Single Use Reply Blocks) or MURBs (Multi Use Reply Blocks) it is done by the builder of the reply block.

The header is protected by the symmetric k_s and is found in a preamble to the header protected by the receiving peers private key k_h^{-1}. The key k_s is known to the routing block builder only and the receiving node receives all important information protected by this key. k_p is known to two immediate peers and the builder of the routing block. The sending peer gets it from the routing block whereas the receiving peer finds it in the $headerBlock$.

2.2.2 Header Block

The header block contains vital static information for the message disclosed to only one peer of the network. It is protected ba k_s. The minimally contained information can be described in the tuple $headerBlock_i := \langle sendingIdentity, serial_i, replayAttributes_i, key_{p_i}, chainSecret, signature, optionalOperations \rangle$.

2.2.3 Routing Block

A routing block can be expressed with the following recursive tuple $routingBlock_i := \langle nexthopAddress, chainSecret, timingAttributes, E^{k_{s_{i+1}}}(headerBlock_{i+1}), E^{k_{s_{i+1}}}(routingBlock_{i+1}), payloadBuildInstructions_i, payloadId, optionalReplyBlocks \rangle$

2.2.4 Payload Block

A payload block is any number of bytes representing parts of a message, decoy traffic or a control block.

2.3 Message Processing

Unlike with traditional mix system a node has no choice of sending. It purely relies on the message processing facilities. A message is either handed over to the transport layer by the blending layer or may be induced internally (if local node is the sender).

First the preamble to the header is extracted. This proves that the sender possesses the public key of the node and contains the sender key k_{s_i}. With this information the node opens the $headerBlock$ revealing information regarding the ephemeral identity of the original sender. Based on the information given in the relatively small header the transport layer may decide whether further processing is desired or not. If desired, the node extracts the key k_{p_i} and decrypts the rest of the message which is considerably larger containing routing and payload information.

The routing block may contain instructions on processing information contained in this or any message related to this message and identity. These instructions are encoded in so called "Operations" as specified in section 2.4 and may be any combination of them. As soon as time arises for a routing block to be processed the blocks required for sending are built. If all prerequisites are satisfied the message blocks are built, concatenated with the new routing block, encrypted with $k_{p_{i+1}}$ this new block is concatenated with the already encrypted header and preamble of the routing block and then passed to the blending layer with the blending specification and the target address.

It is important to note that the blending specification contains vital information about how the message must be blended but not how the carrier message looks like. This is defined like that to minimise the risk of abuse (eg. sending plain text spam through the vortex system).

2.4 Operations

The operations are designed in such a way that they do allow variance of message size without telling to anyone including the generator which message part is used later. They include features to protect message content from bugging.

Some of the operations require a pseudo random number generator (PRNG). This PRNG is defined in section B. The definition of a reproduceable PRNG to be used by messages is important as we have to achieve binary proof messages.

2.4.1 addRedundancy *and* removeRedundancy *Operation*

This Operation is based on a modified Reed-Solomon redundancy function in order to accommodate the anonymity needs of this function. The Reed-Solomon function as defined in appendix A offers a varying number of redundant checksum blocks. When sending these blocks into multiple directions no mixing node is able to tell where the original message is being rebuilt. The general inner workings are described in figure 3.

We define a function $\mathrm{addRedundancy}_{n,m}(\mathbf{M}, k_0 \dots k_{m-1})$ where M denotes the message, n the number number of total output blocks, m the number of redundancy blocks k the encryption key and scheme to be used, and bs_k the block size required to accommodate scheme and key size described by k. It is important to note that the number of true data blocks is $d = n - m$ the rest of the output blocks are redundancy informations.

By encrypting all output blocks individually we make sure that no node having access to enough blocks may rebuild the data stream without the senders consent.

The message is length prefixed with a big endian 64 bit unsigned integer number and padded in such a way that $8 + \mathrm{len}(M) + \mathrm{len}(padding) \bmod bs_k = 0$. As padding stream

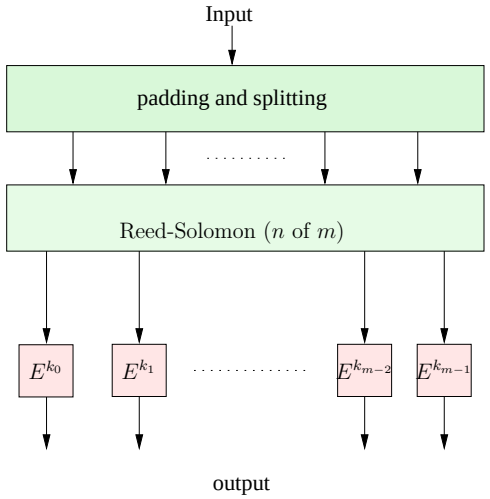

Fig. 3. AddRedundancy Operation

we take the output of $\mathrm{prng}_i\left(\left\lceil\frac{8+\mathrm{len}(M)}{b_s}\right\rceil b_s n\right)$. The first 64 bytes of the message (padded with 0 if required) is taken as initializer i for the PRNG function. By preparing our message block in such a way we guarantee that the output blocks are encryptable without further padding and that the output of all $addRedundancy$ functions is binary proof. If stream cyphers are used as output cyphers then padding is not required.

The reverse function for addRedundancy is called $\mathrm{removeRedundancy}_{m,n}(B_0 \dots B_{m-1}, k_0 \dots k_{m-1}) = \mathbf{M}$ and recovers the original data stream if enough blocks (at least m) and respective valid keys are provided.

2.4.2 splitPayload *and* mergePayload *Operation*

The splitPayload and mergePayload operations split and merge payload blocks into two chunks of different or equal sizes respectively joins them. We define the functions as follows:

If $\mathrm{len}(pb_1)$ expresses the size of a payload block called pb_0 in bytes then the two resulting blocks of the $splitPayload$ Operation pb_1 and pb_2 have to follow the following rules:

$$\mathrm{splitPayload}(f, pb_0) = \langle pb_1, pb_2\rangle \quad (1)$$
$$\mathrm{startsWith}(pb_0, pb_1) \quad (2)$$
$$endsWith(pb_0, pb_2) \quad (3)$$
$$\mathrm{len}(pb_2) = \lfloor \mathrm{len}(pb_0) \cdot f \rfloor \quad (4)$$
$$\mathrm{len}(pb_0) = \mathrm{len}(pb_1) + \mathrm{len}(pb_2) \quad (5)$$

respectively

$$\mathrm{mergePayload}(pb_1, pb_2) = pb_0 \quad (6)$$
$$\mathrm{startsWith}(pb_0, pb_1) \quad (7)$$
$$endsWith(pb_0, pb_2) \quad (8)$$
$$\mathrm{len}(pb_0) = \mathrm{len}(pb_1) + \mathrm{len}(pb_2) \quad (9)$$

2.4.3 xorSplit *and* xorMerge *Operation*

xorSplit and xorMerge are low cost obfuscation operations. These operations may be applied if a block is passed on without any required operation or as one-to-two blocks redundancy generating function.

They furthermore have the advantage that the output of the operation is not uniformely distributed. As encrypted messages may be easily identified with this feature. This may be an important feature for the blending layer.

The operations are trivially defined as follows:

$$\mathrm{xorSplit}(pb_0) = \langle pb_1, \mathrm{prng}_i(\mathrm{len}(pb_0))\rangle \quad (10)$$
$$pb_1 = pb_0 \oplus \mathrm{prng}_i(\mathrm{len}(pb_0)) \quad (11)$$
$$\mathrm{xorMerge}(pb_1, pb_2) = \langle pb_0\rangle \quad (12)$$
$$pb_0 = pb_1 \oplus pb_2 \quad (13)$$

2.4.4 encrypt *and* decrypt *Operation*

encrypt and decrypt are message obfuscation operations. These operations may be applied if a block is passed on without any required operation. They minimize the risk for a known plain text attack to a MessageVortex block. Both operations are defined as a padded or unpadded symmetrical encryption. $spec$ is the encryption specification and key provided by the routing block.

The operations are trivially defined as follows:

$$\mathrm{encrypt}(pb_0) = E_{spec}(pb_0) = pb_1\rangle \quad (14)$$
$$\mathrm{len}(pb_1) \geq \mathrm{len}(pb_0) \quad (15)$$
$$\mathrm{decrypt}(pb_1) = D_{spec}(pb_1) = pb_0 \quad (16)$$

3 Results

3.1 Message Building

Using previously defined operations we may build a message path. This path is typically built by first assigning an identity set I_k where k denotes the target identity. I_k is a static set of n ephemeral identities $I_k\langle eI_1 \dots eI_n\rangle$ which are always used to communicate with k. This set may be enriched with further m ephemeral identities when sending.

When building the message it has to be made sure that all nodes in I_k get enough information to rebuild the message. If an adverser is able to identify the full message flow and knows all the operations applied to the message except for those on the entry and exit node and at least a subset of $k = |I_{k_{uncompromised}}|$ where $k > 1$ exists then we are still at k-Anonymity as an absolute worst case scenario. Thus we can prove that attacks as described in [17] are of very limited use.

3.2 Attacking the Message Flow

In our thesis [18] we analyse various kinds of attacks. Such as illicit behaving nodes, hijacking of header and routing blocks, analysis on payload blocks, traffic replay, and analysis on operations. Results have shown that the protocol is very resistant against most kinds of attacks.

For block hijacking of a single block we can proof that probability for success is at least below $10E - 11$. We can furthermore prove the effectiveness of replay protection even when assuming misbehaving nodes. We can easily show the effectiveness of the tagging and bugging protection.

3.3 Routing Diagnosis

If an interruption of path is suspected parts of the message may be obtained by the message block builder at any time. He may do this by either introduce fixed diagnostic paths into a routing block which we refer as implicit diagnosis or he may send a second message picking up a block of the message at a node which should be tested. This we refer as explicit diagnostic.

Explicit diagnostics may be used as a kind of “receipt” from any node including but not limited to the terminal receiver of

a message. any block at any time of routing may be returned direct or indirect to the original sender. Arrival of such a packet and content tells the sender at which point a message did fail. If a diagnostic packet does not arrive, the routing block builder may build a implicit diagnostic message. This message may test any node between the last successful ant the last failing diagnostic message.

4 Discussion

4.1 Comparison to Existing Systems

The following section gives a short comparison to existing systems. It shows that the solution defined in this paper covers a different approach and what problems are solved.

It is important to note that this is not a ranking. It just outlines the differences between the system and shows where our system is different compared to existing solutions.

4.1.1 General

In [18] we show that the protocol is very secure. It is hard to block as messages may be redundant if required. It is hard to apply censorship in a real world scenario as messages are extremely hard to detect. It has however some flaws which must be outlined.

We always considered as an algorithmic censorship. If human censorship is applied we must assume that at least some of the messages are being identified as possible MessageVortex messages. If we assume a whitelisting, human, censoring adverser (everything which is not identified by a human as compliant is censored) we must conclude that at least some messages will fail to be delivered.Furthermore some of the participating nodes may be identified. This may be compensated with redundancy in message transmission.

4.1.2 TOR

TOR [19] is a synchronous or near synchronous routing system. The anonymisation is based on mixing by using a statical path consisting out of a entry node, an exit node and at least three more intermediate nodes.

Tor is criticised for several things. First it is easy attackable if a person does not use encryption in the transported protocol. It does rely on the trust to a centralized directory infrastructure. It is susceptible if more than $\approx 30\%$ of the nodes are controlled by an evil adversary as shown in [20]. Furthermore timing analysis on entry and exit nodes are particularly easy due to the fact that TOR is a low latency network [21], [22].

MessageVortex tries to address these problems in multiple ways. First there is no central infrastructure which defies the trust problem. There are no entry or exit nodes as all participating members are routers at the same time. As an immediate result all problems related to entry and exit nodes do not exist.

MessageVortex several has downsides compared to TOR. As it is asynchronously it introduces a timing component. It is therefore not suitable for real time communication. It is furthermore a closed system. TOR allows to tunnel almost any traffic through it. Whereas MessageVortex is a closed system. Only participating members may use it.

4.1.3 $\mathcal{P}^5$

The Peer-to-Peer Personal Privacy Protocol is defined in [13]. It provides sender-, receiver- and sender-receiver anonymity. Unlike many other protocols. According to the project page of $\mathcal{P}^5$ there is only a simulator available for the protocol.

The transport layer problematic has been completely ignored. As there is no true protocol specification but only a rough outline about the messaging and the crypto operations $\mathcal{P}^5$ offers very limited possibilities for analysis. It claims to be peer to peer which would result in some kind of NAT (Network Address Translation) circumvention technology. This technology does usually rely in at least partial central infrastructure (e.g. for hole punching).

While MessageVortex protocol is peer to peer the transport layer is not. It misuses already existing infrastructure for transport. This makes it not susceptible to approaches against infrastructure unless our messages are identified and filtered. This may be corrected by applying different blending schemes for the transport layer.

4.1.4 I^2P

The name I^2P is an derived from "Invisible Internet Project" according to geti2p.net. The system itself is comparable to Tor for its capabilities. Mayor differences are:

- P2P based
- Packet switched routing (tor is "circuit switched")
- Different forward and backward routes (called tunnels)
- Works pseudonymously
- Supports TCP and UDP

I^2P has not attracted as much attention as Tor so far. So it is hard to judge upon its real qualities.

Unlike TOR anonymity is here not fully granted. Instead a pseudonymity is granted.

In [23] an attack specific to I^2P is presented. As I^2Ps security model is chosen based on IP addresses the authors propose to use several cloud providers in different B-Class networks. By selectively flooding peers an adverser may extract statistical information. The paper proposes an attack based on the heuristic performance-based peer selection. The main critics of the paper were that the peer selection may be influenced by an adverser enabling him to recover data on a statistical base.

Due to the replay protection and the trust we do not rely on any node we show in [18] that attacks on this level is not possible.

4.1.5 Freenet

Freenet was originally designed to be a fully distributed data store [14]. Documents are stored in an encrypted form. Downloaders must know a document descriptor called CHK containing the file hash, the key, and some background about the crypto being used. A file is stored more or less redundantly based on the number of accesses to a stored file. The main goal of Freenet is to decouple authorship from a particular document. It furthermore provides a fault tolerant storage which improves caching of a document if requested more often.

Freenet is a storage system. It may even be used as store and forward transport layer for the MessageVortex protocol. This however would be a heavy misuse and would generate a huge overhead within Freenet as a block is stored once, then recovered by a different node and never again touched.

5 Conclusion

The protocol outlined in the previous sections does not solve all privacy issues which might arise. Furthermore it is complicated to implement and involves a considerable amount of book

keeping at runtime which is left to the sender of a message and the mixing nodes.

On the positive side we have a new protocol which addresses privacy in a holistic approach leaving very little attack surface. If handled with appropriate care by the sender and receiver, the protocol allows a sender controlled, high degree amount of anonymity. Message paths are diagnosable, may be built redundant and do not build on trust of any third party systems including all involved mixes except the sender's and receiver's one. Even closed group communication or broadcasting to multiple identities involving a specific subset of mixes is possible if desired by the sender.

APPENDIX A
REED-SOLOMON FUNCTION

The origins of the Reed-Solomon code go back to [24]. The method described in this paper was however not applicable in all cases. The publications [25] and [26] describe a more practical solution whereas [27] brings up the similarity to the Reed-Solomon code with a $GF(2^{\omega})$.

Reed-Solomon is used for many applications today. One of the most well known application is a redundancy generator for RAID-6 like systems. It is able to generate multiple linearly independent equation systems to a given set of data thus allowing to create systems where m out of n data storage systems may fail. The remaining $n - m$ data storages may then be used to rebuild the missing content. Traditionally the data and redundancy information is striped into blocks and distributed together with the redundancy information over all n storages. This is done to avoid data storages as bottleneck since a change to one data stripe in a stripe set results always in a change of the redundancy data on the other m storages. This would result in hot spots on redundancy information drives.

We use the Reed-Solomon function as redundancy generating function shown in figure 3. Unlike in storage technology we encrypt each redundancy block and all data stripes individually. By doing so we make it impossible to recover the contained information without knowledge of the keys. All blocks do then contain the same amount of data. Given we have enough blocks and the corresponding keys we may rebuild the message.

At the same time the generating node is unable to tell what blocks belong to the true message path and what blocks are sent for decoy traffic only.

As our resulting blocks are encrypted with a stream or block cypher we need to introduce some padding. The padding is applied before doing RS calculation. In the case of a stream cypher we need to pad so that the number of bytes is dividable by the number of data blocks. In the case of block cyphers we need to pad so that all resulting data blocks have exactly a size dividable by the block size. By applying the padding before spliting the blocks we achieve two goals. First we reduce overhead by adding only one instead of m paddings. Secondly an unpadded block is much harder to brute force. any resulting block to a key might be the right one as we have no longer padding to suggest that a decryption has been successful.

For more information of the used GF-Fields and exact matrices see [18].

APPENDIX B
PSEUDO RANDOM NUMBER GENERATOR

Our PRNG used for this work is an xorshift+ generator. It is based on the XSadd PRNG [28] and passes the bigcrush PRNG test suite. It is a fast, xor based PRNG which has two internal 64 bit seed states s_0 respectively s_1 and is defined as follows:

$$x = s_0 \tag{17}$$
$$s_0 = s_1 \tag{18}$$
$$x = x \oplus (x \ll 23) \tag{19}$$
$$s_1 = x \oplus s_1 \oplus (x \gg 17) \oplus (s_1 \gg 26) \tag{20}$$
$$nextNumber = s_1 + s_0 \tag{21}$$

We have chosen this comparably weak PRNG for the practical reasons. It is fast, simple, and is based on operations easy to implement on hardware. As we do not need a cryptographically strong PRNG, it is the primary choice so far.

As the protocol is heavily dependent on security we have introduced everywhere at least one alternate algorithm which may be used if one of the choices may become a problem. In order to have a second choice for the PRNG we define the Blum-Micali PRNG as described in [29]. This PRNG is a cryptographically secure PRNG and is defined as follows:

p is prime and g is a primitive root modulo p. x_0 reflects the seed state.

$$x_{i+1} = g^{x_i} \mod p \tag{22}$$

ACKNOWLEDGMENTS

The authors would like to thank their families for being so patient with them, and **many more FIXME name them** for their thoughts on the paper.

REFERENCES

[1] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type iii anonymous remailer protocol," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003, pp. 2–15. [Online]. Available: http://mixminion.net/minion-design.pdf

[2] C. Gülcü and G. Tsudik, "Mixing E-mail with Babel," in *Proceedings of the Network and Distributed Security Symposium - NDSS '96*. IEEE, Feb. 1996, pp. 2–16. [Online]. Available: http://citeseer.nj.nec.com/2254.html

[3] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman, "Mixmaster Protocol — Version 2," IETF Internet Draft, Jul. 2003. [Online]. Available: http://tools.ietf.org/pdf/draft-sassaman-mixmaster-03.pdf

[4] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, Aug. 2004. [Online]. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA465464

[5] R. Dingledine, M. J. Freedman, and D. Molnar, "The free haven project: Distributed anonymous storage service," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, H. Federrath, Ed. Springer-Verlag, LNCS 2009, Jul. 2000. [Online]. Available: http://freehaven.net/doc/berk/freehaven-berk.ps

[6] 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, Feb. 2003. [Online]. Available: http://www.cs.cornell.edu/People/egs/papers/herbivore-tr.pdf

[7] D. Chaum, "Untraceable electronic mail, return, addresses, and digital pseudonyms," *Communications of the ACM*, 1981. [Online]. Available: http://www.cs.utexas.edu/~shmat/courses/cs395t_fall04/chaum81.pdf

[8] ——, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, pp. 65–75, 1988. [Online]. Available: http://www.cs.ucsb.edu/~ravenben/classes/595n-s07/papers/dcnet-jcrypt88.pdf

[9] L. Øverlier and P. Syverson, "Locating hidden servers," in *Proceedings of the 2006 IEEE Symposium on Security and Privacy*. IEEE CS, May 2006. [Online]. Available: http://tor-svn.freehaven.net/anonbib/cache/hs-attack06.pdf

[10] M. V. Barbera, V. P. Kemerlis, V. Pappas, and A. Keromytis, "CellFlood: Attacking Tor onion routers on the cheap," in *Proceedings of ESORICS 2013*, Sep. 2013. [Online]. Available: http://www.cs.columbia.edu/~vpk/papers/cellflood.esorics13.pdf

[11] A. Biryukov, I. Pustogarov, and R. P. Weinmann, "TorScan: Tracing long-lived connections and differential scanning attacks," in *Proceedings of the European Symposium Research Computer Security - ESORICS'12*. Springer, Sep. 2012. [Online]. Available: http://freehaven.net/anonbib/papers/torscan-esorics2012.pdf

[12] A. Biryukov, I. Pustogarov, and R.-P. Weinmann, "Trawling for tor hidden services: Detection, measurement, deanonymization," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, May 2013. [Online]. Available: http://www.ieee-security.org/TC/SP2013/papers/4977a080.pdf

[13] 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. [Online]. Available: http://www.cs.umd.edu/projects/p5/p5.pdf

[14] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, Jul. 2000, pp. 46–66. [Online]. Available: https://freenetproject.org/

[15] M. Abadi, M. Burrows, M. Manasse, and T. Wobber, "Moderately hard, memory-bound functions," *ACM Transactions on Internet Technology (TOIT)*, vol. 5, no. 2, pp. 299–327, 2005. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/03/proceedings/papers/2.pdf

[16] J. Fidrich, M. Goljan, and D. Hogea, "Steganalysis of jpeg images: Breaking the f5 algorithm," 2002. [Online]. Available: http://www.ws.binghamton.edu/fridrich/research/f5.pdf

[17] G. Danezis and A. Serjantov, "Statistical disclosure or intersection attacks on anonymity systems," in *Proceedings of 6th Information Hiding Workshop (IH 2004)*, ser. LNCS, May 2004. [Online]. Available: http://www.cl.cam.ac.uk/~aas23/papers_aas/PoolSDA2.ps

[18] M. Gwerder, "Messagevortex – transport independent messaging anonymous to third parties," 12 2017.

[19] R. Dingledine and N. Mathewson, "Tor protocol specification." [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt

[20] R. Jansen, F. Tschorsch, A. Johnson, and B. Scheuermann, "The sniper attack: Anonymously deanonymizing and disabling the tor network," DTIC Document, Tech. Rep., 2014.

[21] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. IEEE CS, May 2005. [Online]. Available: http://www.cl.cam.ac.uk/users/sjm217/papers/oakland05torta.pdf

[22] S. Chakravarty, A. Stavrou, and A. D. Keromytis, "Traffic analysis against low-latency anonymity networks using available bandwidth estimation," in *Proceedings of the European Symposium Research Computer Security - ESORICS'10*. Springer, September 2010. [Online]. Available: http://www.cs.columbia.edu/~sc2516/papers/chakravartyTA.pdf

[23] M. Herrmann and C. Grothoff, "Privacy implications of performance-based peer selection by onion routers: A real-world case study using i2p," in *Proceedings of the 11th Privacy Enhancing Technologies Symposium (PETS 2011)*, Jul. 2011. [Online]. Available: http://freehaven.net/anonbib/papers/pets2011/p9-herrmann.pdf

[24] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[25] E. Karnin, J. Greene, and M. Hellman, "On secret sharing systems," *IEEE Transactions on Information Theory*, vol. 29, no. 1, pp. 35–41, 1983. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/1056621/

[26] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *J. ACM*, vol. 36, no. 2, pp. 335–348, Apr. 1989. [Online]. Available: http://doi.acm.org/10.1145/62044.62050

[27] F. P. Preparata, "Holographic dispersal and recovery of information," *IEEE Transactions on Information Theory*, vol. 35, no. 5, pp. 1123–1124, 1989. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/42233/

[28] G. Marsaglia *et al.*, "Xorshift rngs," *Journal of Statistical Software*, vol. 8, no. 14, pp. 1–6, 2003.

[29] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudorandom bits," *SIAM journal on Computing*, vol. 13, no. 4, pp. 850–864, 1984. [Online]. Available: http://dx.doi.org/10.1137/0213053

Martin Gwerder Martin Gwerder was born 20. July 1972 in Glarus, Switzerland. He is currently a doctoral Student at the University of Basel. After having concluded his studies at the polytechnic at Brugg in 1997, he did a postgraduate study as a master of business and engineering. Following that, he changed to the university track doing an MSc in Informatics at FernUniversität in Hagen. While doing this he constantly broadened his horizon by working for industry, banking and government as engineer and architect in security related positions. His main expertise lays in the field of networking related problems dealing with data protection, distribution, confidentiality and anonymity.