Information Sciences 210 (2012) 99–117

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Anonymous overlay network supporting authenticated routing

Roman Schlegel *, Duncan S. Wong [1]

Department of Computer Science, City University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 6 July 2010
Received in revised form 21 August 2011
Accepted 27 April 2012
Available online 19 May 2012

Keywords:
Anonymous network
Privacy
Anonymous routing
Authentication
Path cost reduction attack

ABSTRACT

Typical anonymous networks mainly focus on providing strong anonymity at the price of having lower bandwidth, higher latency and degraded usability with limited routing support. They also often anonymize only a few specific applications. In this paper, we propose a new approach of constructing an anonymous network by building an overlay network atop a conventional IP network. The overlay network decouples the actual IP addresses of nodes and the virtual addresses that the nodes are using in actual applications. To do so, we use virtual addresses to anonymize the hosts and the physical IP address for efficient routing. The virtual addresses can also be dynamic for enhancing the nodes' anonymity further. This approach also allows the network to support almost any application running on it. Together with a new anonymous routing protocol, our simulation results show that the expected latency of our proposed anonymous system can be reduced by up to 50% compared to existing systems.

We also propose a suite of authentication methods which can be applied to the anonymous routing protocol we propose for preventing any malicious path cost reduction. Traditional routing protocols leak network topology information to nodes while existing anonymous routing protocols do not provide authentication for routing information. A malicious node can arbitrarily reduce the path cost value carried in an anonymous route announcement message for the purpose of negatively influencing routing efficiency or facilitating the launch of various attacks such as eavesdropping or man-in-the-middle attacks. We propose three generic schemes and several concrete instantiations to transform an anonymous routing protocol into an authenticated one which not only prevents path cost reduction attacks but also maintains anonymity. These schemes are based on three different primitives, namely one-way trapdoor functions, digital signature schemes and collision-resistant hash functions.

1. Introduction

Anonymity on the Internet is a topic which has been the subject of research of many papers and numerous anonymous networks have been proposed [9,10,18,15,13,44,35,16,21,27,41]. Most of these networks are based on one of two methods: onion routing and DC-nets [9,10,18]. Anonymous networks have many useful applications. For example, information which might be censored for political reasons in some countries may be published and accessed anonymously using anonymous networks over the Internet; similarly, a whistleblower may use an anonymous network to publish relevant information while keeping himself/herself anonymous; and organizations providing medical information can provide web services

* Corresponding author.
E-mail addresses: rs@ione.ch (R. Schlegel), duncan@cityu.edu.hk (D.S. Wong).
[1] The author was supported by CityU Grant (Project No. 7002711).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.04.042

within an anonymous network so that people can access information anonymously without worrying about being discriminated against because of a medical condition.

Many of the proposed anonymous networks [15,44,16,21,13,35] provide good anonymity. Nevertheless, most of them have drawbacks which limit their usefulness or hinder adoption. These disadvantages include high latency, limited bandwidth, narrow suitability (e.g. only individual protocols can be anonymized and applications have to be re-configured). Research into the performance of Tor [13] and JAP [35] shows a higher latency (up to an order of magnitude, from 0.4 s to around 4 s) and either a low (JAP) or inconsistent bandwidth (Tor) offered by existing anonymous networks [38], when compared to browsing on the conventional non-anonymous Internet.

The inconsistent bandwidth and the high latency discourage users from using an anonymous network. Fewer nodes in an anonymous network implies less anonymity for users who constitute the network. By increasing the number of users or nodes in the anonymous network, this can help enhance the anonymity for all the users of the network [12]. In order to attract more nodes to join an anonymous network, we need to build an anonymous network which can offer lower latency and higher bandwidth than existing anonymous networks. At the same time it should be versatile enough so that it can carry almost any Internet applications on top of it anonymously and will be available and useful to as large a number of Internet users as possible. Our approach therefore focuses on *reasonable anonymity* for everyday use rather than providing near-perfect anonymity for the more special cases where this is required.

In order to achieve both anonymity and good performance, an anonymous network has to make sure that the nodes in the anonymous network can communicate with each other efficiently. This requires an *anonymous routing protocol*, i.e., to determine how to route data packets from one node to another efficiently while maintaining the anonymity of the network, that is, without leaking any significant network topology information. Flooding is possible but neither scalable nor efficient. In [43,42], several anonymous routing protocols have been proposed. These protocols can ensure that when a node receives a routing announcement, it can only learn via which of its neighbors a particular destination can be reached with the smallest path cost. Apart from an estimated path cost (number of hops, latency, etc.), no additional information about the network topology is leaked. These protocols can create routes efficiently and also prevent routing loops. However, in these protocols, there is no *authentication*, they assume that nodes are honest and do not maliciously influence the routing process. In fact, if any node in their network is malicious it can influence the routing and launch attacks such as eavesdropping or man-in-the-middle through carrying out *path cost reduction attack* on the underlying anonymous routing protocols. A few non-anonymous routing protocols, such as S-BGP [22], soBGP [39] and psBGP [36] (all extended from BGP [33]), support authentication so that malicious tampering with routing information can be detected by the nodes in the network. However, these routing protocols inherently leak the network topology information.

In this article we describe a new approach for constructing an anonymous network. The idea behind it is twofold. First, we build an anonymous overlay network, then we propose an anonymous routing protocol which not only protects the anonymity of routing nodes but also helps route traffic for lower latency and higher bandwidth. Every node on our anonymous overlay network obtains a set of randomized identifiers called *overlay addresses*. The overlay addresses are dynamic and can be changed from time to time, multiple overlay addresses can co-exist on a single node at any given time and more importantly, we introduce mechanisms which ensure that no adversary is able to correlate the overlay address(es) of a particular node with its real IP address; under the condition that the adversary does not have full control of the entire network.

Furthermore, we present an anonymous routing protocol with three different authentication schemes. The anonymous routing protocol can be considered to be an anonymized version of path vector routing, preserving the performance characteristics of path vector routing but with the added advantage of hiding the overlay network topology. We also present concrete instantiations of the different schemes. Finally, we evaluate both the proposed anonymous network and the routing protocol and give some preliminary results.

1.1. Organization

In the next section, we review related work for anonymous networks and anonymous routing. This is followed by the description of our anonymous overlay network in Section 3 and the description of the anonymous routing protocol in Section 4. Section 5 contains the evaluation and in Section 6 we conclude and indicate some directions for future work.

2. Related work

2.1. Anonymous networks

The initial design of the Internet did not include any mechanism for anonymity. For almost all traffic an eavesdropper can easily determine who is communicating with whom and very often even the content of the communication is easily accessible. Because there are situations where anonymity is desirable there has been quite a lot of research into systems which provide anonymity on the Internet. Most of them go back to the idea of mixes [9], a system where messages are randomly passed between a number of hosts and each host performs some cryptographic operations on a message to obfuscate who is communicating with whom.

A popular technology building on these mixes is onion routing [18]. In onion routing, a message is encrypted repeatedly (“onion layers”) and then sent along a path of different hosts. Each host in the path removes the outermost layer of the encryption which reveals the next hop in the path. This is repeated until the packet reaches the last hop which can unwrap the last layer and access the message itself. Using onion routing, each host only knows the preceding host and the next host in the whole path, making the communication between the source and the destination anonymous. The probably best known public implementation of onion routing is Tor [13], an onion router network with more than a thousand server nodes [25]. A Tor client selects a number of Tor server nodes and builds a path along these nodes. The last node then connects to the destination intended by the client. This ensures that the client remains anonymous to the destination host, because it is a random Tor server which makes the actual connection to the destination. There are other systems based on the idea of onion routing like Tarzan [15] and Cashmere [44].

A disadvantage of onion routing is that to actually provide anonymity, the path length between the source and the destination has to be artificially inflated. Each communication passes through several intermediate hops which is less efficient in terms of latency and bandwidth than a direct connection. Measurements by Panchenko et al. show that performance is considerably worse compared to normal, direct network traffic, with a mean-time of 4*s* just to load the headers of a website compared to 0.4*s* for the non-anonymous case [29]. This is acceptable for users where the need for anonymity outweighs the problem of slow performance, but discourages casual users, leading to a smaller anonymity set and thus less anonymity as explained by Dingledine et al. [12].

Another class of systems for anonymity is based on the problem of the dining cryptographers introduced by Chaum [10], which uses XORing of bit vectors to transmit information with provable untraceability. The bandwidth available for transmitting data drops off with the square of the number of users and therefore does not scale well. One system building on DC-nets is Herbivore, which tries to solve the scalability problem through partitioning, reducing untraceability to individual partitions [16].

2.2. Anonymous routing

Routing in a network is the process of establishing a route for packet transmission between two nodes while observing a set of criteria like latency, bandwidth, path length, congestion or other kinds of cost or measure to optimize the overall performance of the network. Traditional routing protocols do not actively hide any information about the routing process. In networks like the Internet the inter-domain topology is freely advertised as part of the normal routing process.

In an anonymous network, routing protocol should help establish the necessary routes for the communication between any two entities in the network without disclosing the topology of the network or information about the path of individual routes. Anonymous routing has been studied especially in connection with mobile ad hoc networks (MANET). Protocols (including some non-anonymous routing methods) like [43,14,32,6,30,42,37,8,24,3,28,23] are all focused on MANETs and most of them use on-demand routing. These protocols try to preserve anonymity when establishing routes but the type of anonymity they achieve and the assumptions they make for the correct functioning of the protocol differ. In [43] for example, the source and destination node remain hidden to the intermediate nodes, which also do not have any information about their position in the route (e.g. number of hops). However, an intermediate node can arbitrarily reduce the path cost perceived by the destination, thus possibly re-routing data through itself (i.e. path cost reduction attack).

3. Anonymous overlay network

The anonymous network we propose here is in the form of an *overlay network*. Besides anonymity, another objective of proposing such a new anonymous network is to achieve a network performance in terms of low latency and high bandwidth. We start our description by specifying the underlying adversarial model.

3.1. Attack model

As of most related work [9,10,18,15,13,44,35,16,21], we consider an adversary which may have full control of several parts of the network but does not have the control of the entire network. Full control means that besides eavesdropping, the adversary may even have corrupted and taken control of several nodes in the network. If a node is taken control by the adversary, the adversary can perform anything as the compromised node and manipulate all the traffic that passes through or initiated from the node. However, we do not assume that all the nodes in the network have been corrupted (or are being malicious). Although an attacker can launch traffic analysis, timing attacks and many different types of statistical attacks, by assuming a “non-global” attacker, we can see that we are considering the scenario that the attacker is less capable of launching an effective attack. This is different from some previous attack model (e.g. Tarzan [15]), in which a global eavesdropper is considered, with the trade-off of having less desirable performance.

In the following, we start the description of our anonymous network by giving an overview of the overlay network we propose. This is followed by the detailed description of each component in the network.

3.2. Overview

The overlay network consists of nodes which form a virtual network on top of the Internet by creating logical links among them. Fig. 1 shows an example. The left part of the figure illustrates the physical connections among the nodes but only the shaded nodes (with IP addresses) are part of the overlay network. The right part shows the topology of the overlay network. As usual, the virtual links that delineate the topology of the overlay network also illustrate the relation of *virtual neighbors* on the overlay network. For example, the node with overlay address OA_5 has two virtual neighbors which have overlay addresses OA_1 and OA_4. However, the topology of the overlay network is hidden to all the nodes on the network (we will see shortly on how this is done). As a result, node OA_5 does not know the overlay addresses of its two virtual neighbors. Instead, it only knows that it can connect to the overlay network via two other nodes with real IP addresses IP_1 and IP_4.

A new node which is about to connect to the anonymous overlay network will take the following steps. For illustration, suppose the new node has IP address IP_5 (Figs. 2–5).

1. **Public directory service**: The new node first obtains a list of IP addresses of n_{IP_5} nodes which are already part of the overlay network from a public directory service (DS). The value n_{IP_5} is specified by the new node. In practice, n_{IP_5} can be chosen in the order of thousands. The purpose is to give the new node a large set of nodes to choose from as its *virtual neighbors*. Furthermore, each node in the list also has a public key associated. Fig. 2 illustrates this step.
2. **Virtual neighbor establishment**: The new node chooses uniformly at random v_{IP_5} nodes from the list where $v_{IP_5}, 1 \leqslant v_{IP_5} \leqslant n$, is the number of *virtual neighbors* that the new node intends to have after joining the overlay network and can be arbitrarily set by the new node. Then the new node establishes an encrypted channel with each of these nodes using the public key associated with each node (e.g. by running the authenticated Diffie-Hellman protocol [11,4]). This is exemplified in Fig. 3.

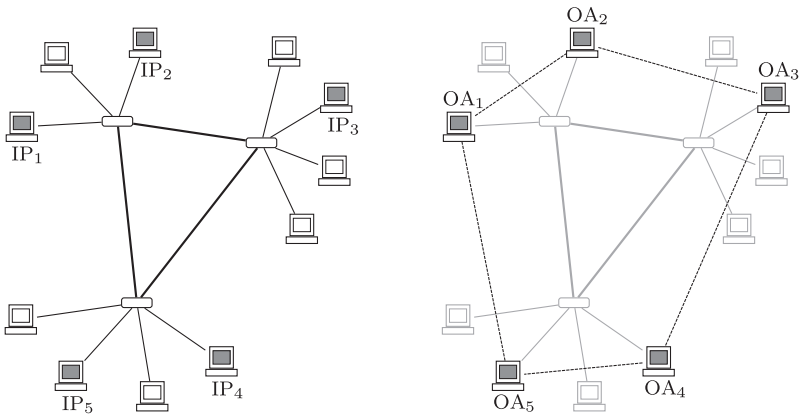

Fig. 1. Schematic drawing of an overlay network. The left drawing shows the physical connections, the right drawing shows the virtual/logical connections among the nodes which are part of the overlay (shaded nodes). Also, addresses in the physical network are different from the addresses in the overlay network.

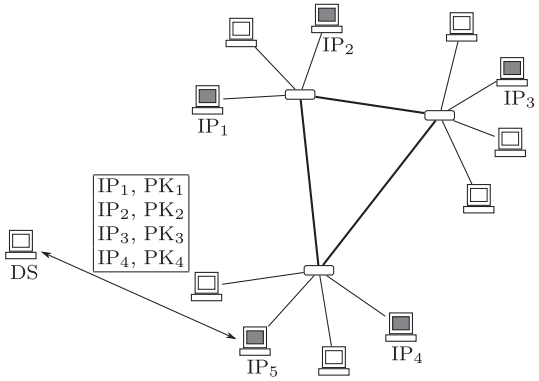

Fig. 2. The new node (IP_5) connects to DS and obtains v nodes which are already part of the overlay network. Each entry on the list also has an associated public key.

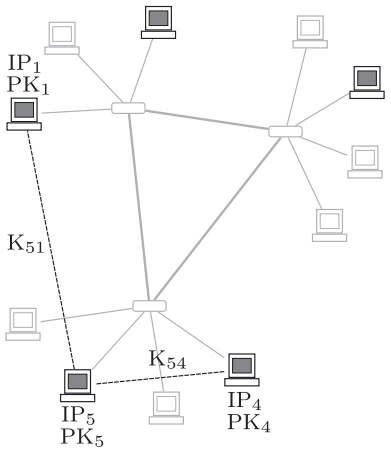

Fig. 3. The new node establishes session keys K_{51} and K_{54} with two nodes chosen from the list. The session keys are used for forming secure channels. Public keys (PK_5, PK_1, PK_4) are used during the authenticated key establishment process. The two nodes will become the *virtual neighbors* of the new node on the overlay network.

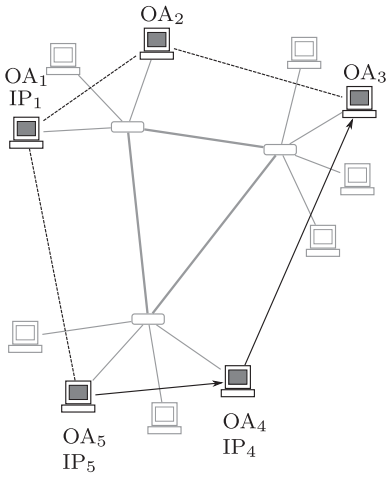

Fig. 4. The routing table in the node OA_5 indicates that routing data via a virtual neighbor with IP address IP_4 is better then going through another virtual neighbor which has IP address IP_1.

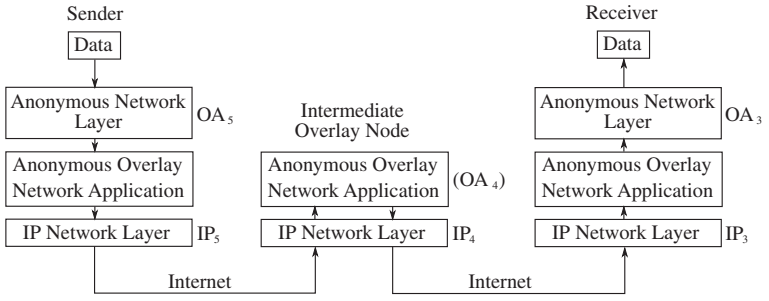

Fig. 5. Interaction between the anonymous network layer and IP network layer when data are sent from a sender to a receiver via an intermediate node.

3. **Overlay address**: The new node randomly chooses ℓ_{IP_5} overlay addresses, each of which is k-bit long (e.g. $k = 128$). Similar to n_{IP_5} and v_{IP_5}, the value $\ell_{IP_5} \geqslant 1$ can be set by the new node arbitrarily. The right drawing of Fig. 1 shows an example of having five virtual nodes over the overlay network, each having a unique but randomly generated overlay address. For simplicity, the drawing shows the scenario that $\ell_{IP_i} = 1$ for $1 \leqslant i \leqslant 5$. However, we stress that a physical node can have

multiple overlay addresses. As shown in Section 3.3, the probability of having a collision on the overlay addresses is negligibly small. The purpose of choosing multiple overlay addresses by each node is to de-link the number of virtual nodes (identified by their overlay addresses) on the overlay network and the actual number of physical nodes on the Internet that have joined the overlay network. One example of the overlay network namespace is given in Section 3.4.

4. **Routing on overlay network**: The set of new virtual nodes with overlay addresses starts running an anonymous routing protocol (Section 4) for learning how to reach other virtual nodes on the overlay network. The anonymous routing protocol will ensure that
   (a) the relationship between the real IP address of a physical node and the set of virtual overlay addresses of the node cannot be determined by any other nodes;
   (b) the overlay network topology will remain hidden to all nodes; and
   (c) the path of any individual route will not be disclosed.

Fig. 4 illustrates the path of sending data from a sender with overlay address OA_5 to a receiver with overlay address OA_3. As we can see, there are two paths. By running the anonymous routing protocol, the sender finds that the "best" way for reaching receiver OA_3 is by forwarding data to the virtual neighbor with IP address IP_4 rather than to the other virtual neighbor with IP address IP_1. The "best" way here refers to the routing objectives of the anonymous routing protocol, for example, achieving low latency and high bandwidth. We stress that no node is able to find out the overlay addresses of its virtual neighbors, instead, it only knows their physical IP addresses.

Unlike existing circuit-based anonymous networks, this new network is packet-based and the reverse path from the node with overlay address OA_3 back to the node with overlay address OA_5 may go through a different path, which is determined by the routing table of the node OA_3 and that of other intermediate nodes along the reverse path.

5. **Services**: A new virtual node can now communicate with all other virtual nodes on the overlay network and specify each other using their overlay addresses. Each physical node should periodically re-run all the steps above for generating a new set of virtual nodes.

The leave and fail processes for a physical node is handled similarly, by treating a failed node as a leaving node. The recovery of routing due to the lost of those routes via the leaving node is taken care of by the our anonymous routing protocol proposed in Section 4. In the following, we analyze the anonymity of this new anonymous network and discuss certain aspects of it in more detail.

3.3. Security analysis

We analyze the anonymity of this new anonymous network construction under the adversarial model given in Section 3.1. In step 1, DS is a simple public directory service such as an LDAP-based or a just bulletin board, containing the real IP addresses and the associated public keys of all the nodes which are in the anonymous network. Besides the size of the network (i.e. the number of physical nodes), there is no information on the topology of the network. In addition, the larger the value n_{IP_5} is, the less likely an adversary which is eavesdropping on the traffic of DS or has compromised the DS can find out which nodes are going to be the virtual neighbors of the new node. Intuitively, if a user receives a set of candidate peers, but only picks a small fraction of them as actual neighbors, then the smaller the fraction, the harder for an adversary to guess which nodes will become its neighbors (as there is no *a priori* information about the choice the node is going to make). The following is a more quantitative analysis.

In step 2, v_{IP_5} virtual neighbors are selected by the new node. If an adversary can monitor all traffic to and from the new node, the adversary can find out the value of v_{IP_5}. However, there is no information on the virtual overlay addresses of these virtual neighbors or that of the new node that the adversary can find out. Larger values for v_{IP_5} give more possible paths to route traffic to and from the new node, hence, improves efficiency and enhances anonymity.

In step 3, ℓ_{IP_5} virtual overlay addresses are chosen by the new node, each of them is k-bit long. They are generated independently from the real IP address of the node. Since all the overlay address are chosen uniformly at random, the chance of having a collision in the network on overlay addresses is at most

$$\binom{N\ell_{max}}{2}2^{-k} \leqslant \epsilon(k)$$

where N is the total number of physical nodes which have joined the anonymous network and ℓ_{max} is the maximum number of overlay addresses that each physical node chooses. $\epsilon : \mathbb{N} \to \mathbb{R}$ is called a negligible function which is defined as for any positive integer c, there exists an integer k_c such that for all $k > k_c$, $\epsilon(k) < 1/k^c$. As both N and ℓ_{max} are polynomial in k, the collision probability above is negligible.

The anonymous routing protocol in step 4 ensures that no information about the relationship between the real IP address and the set of virtual overlay addresses of a node is leaked. In the example illustrated in Fig. 4, this indicates that the sender only knows that the real IP addresses of its two virtual neighbors are IP_1 and IP_4. It has no idea about their overlay addresses, namely OA_1 and OA_4. Also notice that Routing is the only step which involves interaction between real IP addresses and virtual overlay addresses. In the interaction, the real IP addresses relate to the virtual neighbors of the sender only; and the overlay addresses relate to the sender and the destination within the overlay.

After a routing table has created, in step 5, each new virtual node can communicate with all other virtual nodes on the overlay network and identify each other using their overlay addresses. Fig. 5 illustrates how data is transferred from a sender with overlay address OA_5 to a receiver with overlay address OA_3. The intermediate node has IP address IP_4 and overlay address OA_4. As mentioned, the anonymous routing protocol ensures that the sender (OA_5) only knows that the "best" way to reach the receiver (OA_3) is to forward data to its virtual neighbor with IP address IP_4. However, the sender has no idea that the virtual neighbor with IP address IP_4 corresponds to the overlay address OA_4. Similarly, the intermediate node with IP address IP_4 knows (thanks to the anonymous routing protocol) that the "best" way to reach the receiver (OA_3) is to forward the data to its virtual neighbor with IP address IP_3. It has no idea that this virtual neighbor is the actual receiver with the overlay address OA_3. To elaborate further, the intermediate node also has no idea that the node with IP address IP_5 which forwards the data to it is actually the sender with overlay address OA_5. In addition, steps 1–4 are periodically re-run by each physical node so that different communication sessions between any two physically nodes cannot be traced from their virtual overlay addresses after the steps are re-run.

Apart from the encryption of the connection between individual hops (K_{51} and K_{54} in Fig. 3), communication between any two nodes on the overlay can also be encrypted. This end-to-end encryption prevents intermediate nodes that are routing traffic within the overlay network from learning the content of a communication. In this article we would not discuss any concrete implementation of such an end-to-end encryption.

In Section 4, we propose an anonymous routing protocol which not only achieves the three anonymity requirements specified in the step Routing above (Section 3.2), it also helps the node to find a path (i.e. the neighbor or the next hop) which offers low latency and high bandwidth for each destination within the overlay network.

3.4. Overlay network namespace and dissemination

As the overlay addresses for this new anonymous network, we propose to use IPv6 for a number of reasons. Many Internet applications already support IPv6 and most operating systems (Windows, Mac OS X, Linux, UNIX, *BSD) already have a dual protocol stack which makes access to the anonymous network transparent to most applications. As analyzed in Section 3.3, the chance of having a collision on the overlay addresses is in the order of $poly(128)/2^{128}$ which is negligible, where $poly(\cdot)$ is some polynomial function (in practice a little bit less than 128 bits would be used as to prevent collisions with the already assigned IPv6 address space). According to the birthday paradox, the expected number of overlay network nodes has to reach about 2^{64} before having a collision on the randomly generated addresses is likely to occur.

3.5. Geographical awareness

The geographical distribution of individual nodes of the overlay network has an influence on the overall performance of the network. By taking this into account when building the network, the overall performance of the overlay can be improved.

The Internet can be divided into several regions. Within one region latency is usually quite low, in the order of several tens of milliseconds. Between regions on the other hand the latency can go to more than 150 ms one-way (i.e. Asia–Europe). To take advantage of this fact, in this section, we propose an extension to our anonymous network. The idea of our extension is to divide the anonymous network into several regions. In the following, suppose that we divide it into three regions (say, corresponding to America, Asia, Europe). When a new node is joining the overlay network (step 2 **virtual neighbor establishment** in Section 3.2), the new node selects v_{IP_5} nodes from the list that are in the same region as itself. In addition to this, some nodes in the region will make connections to nodes in the other two regions. Those nodes function as gateways to the other two regions for local nodes. A node which sends data to a node in a different region will send the packet to the closest gateway node which forwards it to the destination region where it will then be forwarded locally.

To make this idea work efficiently, we also propose to make a slight modification on the format of the overlay addresses. In the modification, although the overlay addresses would still be chosen randomly, there will be a prefix added to each overlay address which indicates the region. The advantage of using a prefix is that it greatly reduces the size of the routing tables, because when a node wants to send packets to a node in a different region, it only needs to know how to forward the packets to the gateway node which connects to the region of the destination. The routing table of the sender does not need an entry for the specific destination. Instead, it only contains an entry for forwarding packets to the gateway node.

On the anonymity of this extension, one may notice that even though this scheme would reveal the region of a node through the prefix of its overlay address, this information may not provide much help to an adversary trying to compromise the anonymity. The reason is that narrowing the possible location of a node down to one region can already be done quite easily by using other means as well (e.g. latency measurements). The prefix would still leak no information about the relative location within a region or about the topology of the overlay network.

3.6. Anycast proxies

The ultimate goal of setting up such an anonymous network is to have the network be self-sustainable in the sense that all services are being offered within the network. However, it is also important to allow users to use this anonymous network for accessing services that are on the Internet at large anonymously.

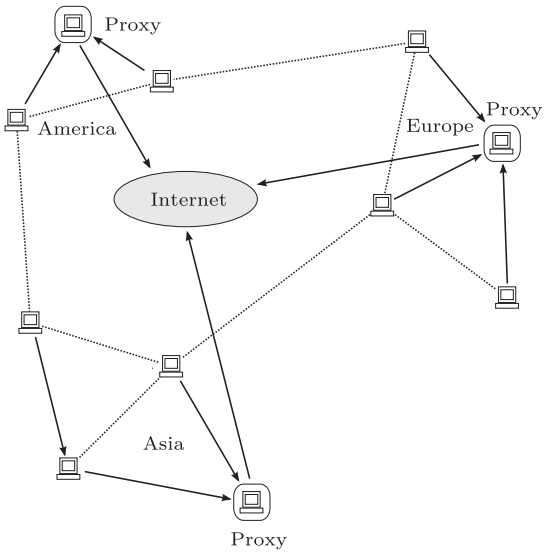

Fig. 6. Access to normal Internet servers is routed to one of the proxies.

For this purpose, we propose to use *anycast proxies* [2] to enable anonymous access to normal Internet services from within the overlay network. To do so, a node inside the overlay network will send traffic destined for a node (let us consider it to be a web server) on the normal Internet to a pre-defined anycast overlay address within the overlay network. Using the conventional anycast mechanism, the traffic will be routed to the "nearest" proxy available (due to the effect of running the anonymous routing protocol described in Section 4). An illustration of the anycast proxies in conjunction with the region notion introduced in Section 3.5 (for better performance) is shown in Fig. 6.

This mechanism can further be extended to allow *regional anycast proxies*, with each region having its own anycast address for proxies. A node can then decide whether to choose a proxy in the same region with better performance or a proxy in another region with better anonymity by using the respective anycast address. While the region of the proxy could be chosen, inside a region it would still be the closest proxy which is used. For some situations this can be optimized to have both good anonymity and good performance. For example, if a node in the region America accesses a web server in the region Europe, choosing a European anycast proxy could provide better performance and anonymity. On the performance, it should not be much worse than a non-anonymous connection because the traffic has to cross from America to Europe in any case, while anonymity should also be good because a more distant anycast proxy is used which can help confuse an adversary trying to determine whether the sender is in America or Europe.

The gateway nodes mentioned in Section 3.5 would also be addressed using anycast, such that packets for a different region are routed automatically to the closest gateway for that region. One disadvantage of using anycast proxies is that for example web browsers would need to be reconfigured.

4. Authenticated anonymous pseudo path vector routing

4.1. Introduction

We now propose an anonymous routing protocol, *authenticated anonymous pseudo path vector routing*, which is suitable for use in Step 4 **routing on overlay network** (Section 3.2). The authenticated anonymous routing protocol should ensure that.

1. the relationship between the real IP address of a physical node and the set of virtual overlay addresses randomly chosen by the node cannot be discovered by any other node;
2. the path of any individual route will not be disclosed; and
3. nodes cannot maliciously decrease the path cost of a route in order to disrupt routing or gain some advantage.

On the Internet, the inter-domain topology is freely advertised and can therefore be obtained easily by any node from public sources. If conventional routing protocols are applied to the overlay network proposed above, the path of any individual route and thus the topology of the overlay network will inherently be disclosed. Furthermore, routes are not usually authenticated and can be maliciously modified by any node in the path by reducing the path cost instead of increasing it. Authenticated anonymous pseudo path vector routing is derived from path vector routing, a version of the simpler distance

Destination	Path Cost	Pseudo Path (PSP)

Fig. 7. The structure of a routing entry in the authenticated anonymous pseudo path vector routing protocol.

vector routing. Path vector routing solves the problem of routing loops and counting to infinity by remembering the path of every route. The most typical example of path vector routing is the Border Gateway Protocol (BGP [33]) which is used to establish routes between autonomous systems in the Internet. In the following we will present the general construction of authenticated anonymous pseudo path vector routing.

4.2. General construction

Fig. 7 shows the structure of a route announcement in this routing protocol. As shown in the figure, there are three components in a route announcement. **Destination** indicates the *overlay address* of the destination of a route. **Path cost** contains a measure indicating the quality of the route. This can also be a measure such as the aggregated latency or the minimum bandwidth along a path. **Pseudo Path** (PSP) replaces the normal path in conventional path vector routing.

4.3. Path cost

The authenticity of the path cost is not only important for the performance of the overlay network but also to prevent malicious nodes from tampering with the path cost. By reducing the path cost, for example, a node can position itself as a traffic sink and then exploit this to launch other attacks. Fig. 8 illustrates this problem with a small example network. In the figure, S initiates an announcement, with an initial path cost of 1, represented by the number of black dots on the links. This could for example be the hop count. The path cost is increased by each node and forwarded. If node D wants to send data to node S, it will forward it to node A.

Now assume node M is malicious and tries to re-route traffic by leaving the path cost value carried by a route announcement message unchanged. D would still choose A as the next hop. However, if M succeeded in reducing the path cost, say by two, this would make B to be chosen as the next hop by D and B would choose M as the next hop and so on. In the following we show a construction which can prevent M from maliciously reducing the path cost.

4.3.1. Generic scheme

Our idea stems from the creation of a binary vector of length n bits: $\vec{v} = (b_1 b_2 \dots b_n)$. The value n denotes the maximum possible value the path cost of a route announcement may reach. The Hamming weight of this binary vector $\vec{v}$ is the path cost. We will also see shortly that this binary vector $\vec{v}$ does not need to be sent explicitly along with a route announcement message. Hence we refer to $\vec{v}$ as an 'imaginary' n-bit binary vector in the remaining part of this section. In the following we propose three schemes which are based on different constructions and have different properties.

Trapdoor functions. The first scheme depends on trapdoor one-way function families. Trapdoor functions are functions which are easy to compute in one direction, but their inverse can only be calculated easily if a certain *trapdoor information* is known [11].

Signatures. The second scheme makes use of digital signature as the primitive for ensuring that the path cost is monotonically increasing. We will show that using digital signatures can help reduce the space required compared to the trapdoor one-way function family based approach.

Hash-tree. The third scheme builds on hash trees and has the desirable property that the size of the imaginary vector decreases as it propagates within the network.

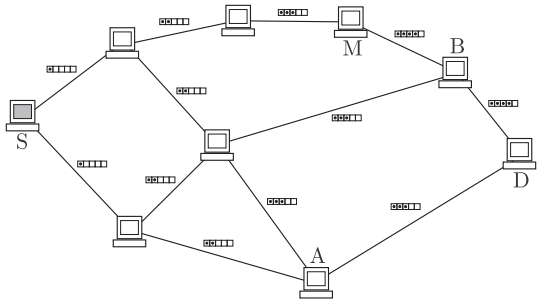

Fig. 8. Routing.

4.3.2. Trapdoor functions

Let $\mathcal{F} = \{f : K \times D \to R\}$ be a family of trapdoor functions where K is the index (i.e. public key), D is the domain and R is the range. By $f_k(M) = y$, we denote that $f(k,M) = y$. For each $k \in K$, let t_k be the trapdoor information (i.e. private key) such that for any $y \in R$, the inverse of f is computable, that is, given t_k, $M' \leftarrow f^{-1}(t_k, y)$ such that $f_k(M') = y$. As an abbreviation, we denote $f^{-1}(t_k, y)$ as $f_k^{-1}(y)$.

For generating a path cost of 0 (i.e. all the n bits of $\vec{v}$ are zeros) in a route announcement, the route initiator prepares the following tag:

$$PathCostTag = \langle \mathsf{S}, V_1, V_2, \mathsf{Sign}_{\mathsf{S}}(\mathsf{S}, T, V_1) \rangle$$

where S is the address of the source (e.g. node S in Fig. 8), T is a timestamp (for preventing replay attacks) and $\mathsf{Sign}_{\mathsf{S}}(\mathsf{S}, T, V_1)$ is the signature over (S, T, V_1) generated by the source. This signature is to prevent a malicious node from replacing V_1. (V_1, V_2) is a data structure which realizes $\vec{v}$. Below are the steps for creating (V_1, V_2):

1. Randomly pick an element $M \in_R R$ from the domain of $\mathcal{F}$.
2. Randomly pick n indices $k_1, k_2, \ldots, k_n \in K$ (i.e. n public keys) and their corresponding trapdoor information $t_{k_1}, t_{k_2}, \ldots, t_{k_n}$.
3. Set $V_1 = (M, k_1, \ldots, k_n)$ and $V_2 = (t_{k_1}, \ldots, t_{k_n})$. Alternatively, we denote (V_1, V_2) as follows using the abbreviations defined above:

$$V_1 = (M, f_{k_1}, f_{k_2}, \ldots, f_{k_n})$$
$$V_2 = \left(f_{k_1}^{-1}, f_{k_2}^{-1}, \ldots, f_{k_n}^{-1}\right)$$

Note that the n public key pairs in (V_1, V_2) correspond to the n bits in the 'imaginary' n-bit binary vector $\vec{v}$. In our solutions, $\vec{v}$ does not need to be explicitly included in the route announcement. Instead, (V_1, V_2) represents this n-bit binary vector. To increase the path cost by 1 (i.e. increase the Hamming weight of $\vec{v}$ by 1), we modify V_2 as follows.

1. Without loss of generality, suppose the first bit of $\vec{v}$ is to be flipped from zero to one, compute $y_1 \leftarrow f_{k_1}^{-1}(M)$.
2. Replace $f_{k_1}^{-1}$ in V_2 by y_1. Thus the updated V_2 becomes

$$V_2 = \left(y_1, f_{k_2}^{-1}, \ldots, f_{k_n}^{-1}\right)$$

Note that V_1 is never changed after it is created by the initiator of the route announcement. The change on V_2 reflects the update on the Hamming weight of the 'imaginary' n-bit binary vector $\vec{v}$.

To determine the path cost and verify whether the path cost is monotonically increasing, a node first checks if the timestamp T is reasonable and also checks if $\mathsf{Sign}_{\mathsf{S}}(\mathsf{S}, T, V_1)$ is valid. Then the node can determine the path cost by looking at (V_1, V_2) and for each i, $1 \leqslant i \leqslant n$, check if f_{k_i} is in V_1 and an element $y_i \in D$ is in V_2 such that $M = f_{k_i}(y_i)$. If so, set the ith bit of $\vec{v}$ to one, otherwise, set it to zero. Finally, the path cost is the Hamming weight of $\vec{v}$.

(*Security*) After $f_{k_i}^{-1}$ is removed from V_2 and is replaced by y_i such that $M_i = f_{k_i}(y_i)$, no one can recover $f_{k_i}^{-1}$ when this route announcement propagates further from the initiator. Otherwise, we contradict the assumption of the trapdoor one-way function family that given f_{k_i}, it is infeasible to calculate the trapdoor information t_{k_i}. In other words, once the 'imaginary' binary vector $\vec{v}$ has a bit position flipped to one, it is infeasible for a malicious node to flip it back to zero. One can imagine that the pair $\left(f_{k_i}, f_{k_i}^{-1}\right)$ in (V_1, V_2), respectively, corresponds to zero at the ith bit of $\vec{v}$ while the pair (f_{k_i}, y_i) in (V_1, V_2), respectively, corresponds to one at the ith bit of $\vec{v}$.

In the following, we propose two instantiations of this trapdoor one-way function family based solution.

RSA-based An RSA [34] public key consists of (N, e), where $N = pq$ with p, q prime and e is an integer such that $\gcd(e, \phi(N)) = 1$, with $\phi(N) = (p-1)(q-1)$. The corresponding trapdoor information, that is the private key, is d, where $ed \equiv 1 \bmod \phi(N)$.

Let $\mathcal{F}_{RSA} = \{f_{RSA} : K \times D \to R\}$ be a trapdoor one-way function family based on RSA, in which K is the set of all the pairs of (N, e) in the form above. Given a public key $k = (N, e) \in K$, the trapdoor one-way function is $f_{RSA}(k, y) = y^e \bmod N = M$. In other words, $D = R = \mathbb{Z}_n^*$. To compute the inverse, we need the trapdoor information d and compute the inverse as $f_{RSA}^{-1}(d, M) = M^d \bmod N$.

In this instantiation, we specify a security parameter $\ell = 1024$ so that all the public keys chosen randomly will have $|N| = \ell$ and p, q are of equal length, that is, $|p| = |q| = \ell/2$. Now we describe how to realize the initial state of (V_1, V_2).

Set $V_1 = (M, (N_1, e_1), (N_2, e_2), \ldots, (N_n, e_n))$ and $V_2 = (d_1, d_2, \ldots, d_n)$, where $N_i = p_i q_i$, and p_i, q_i are all distinct $\ell/2$-bit long primes chosen randomly, e_i are randomly chosen such that $\gcd(e_i, \phi(N_i)) = 1$, and d_i are then computed such that $d_i e_i \equiv 1 \bmod \phi(N_i)$, for $1 \leqslant i \leqslant n$. M is an integer chosen randomly from $[1, 2^{\ell-1}]$. This is to make sure that M is smaller than all the values of N_i, $1 \leqslant i \leqslant n$.

Now to flip a bit at position i of the 'imaginary' n-bit binary vector $\vec{v}$, we compute $y_i = M^{d_i} \bmod N_i$ and replace d_i in V_2 with y_i. Suppose $i = 1$, the updated V_2 becomes

$$V_2 = (y_1, d_2, \ldots, d_n)$$

Optimizations. We can optimize this instantiation so that it requires less space. The exponents $e_i\,(1 \leqslant i \leqslant n)$ can be standardized to a known value (e.g. $2^{16}+1 = 65{,}537$) so that e_i can be removed from V_1 and the new V_1 becomes $V_1 = (M, N_1, N_2, \ldots, N_n)$. Furthermore, in the actual route announcement, we do not need to send V_1 and V_2 exactly as described in *PathCostTag*. Instead, we can send the following which allows the receiving node to reconstruct V_1 and V_2:

$$PathCostTag - ActuallySent = \langle \mathsf{S}, \mathsf{Sign}_{\mathsf{S}}(\mathsf{S}, T, V_1), M, (p_1, q_1), (p_2, q_2), \ldots, (p_n, q_n)\rangle$$

From the message above, V_1 can readily be computed as $N_i = p_i q_i (1 \leqslant i \leqslant n)$, and V_2 can also be computed from (p_i, q_i) and the standardized exponent $e = 2^{16}+1$. When a bit, say the first bit, of the 'imaginary' n-bit binary vector $\vec{v}$ is flipped, the actually sent *PathCostTag* becomes:

$$PathCostTag - ActuallySent = \langle \mathsf{S}, \mathsf{Sign}_{\mathsf{S}}(\mathsf{S}, T, V_1), M, (N_1, y_1), (p_2, q_2), \ldots, (p_n, q_n)\rangle$$

Rabin-based. The public key of Rabin [31] can be considered as just the N component in the public key of RSA. The trapdoor information is the factorization of N, that is, p and q. Let ℓ be a security parameter, for example, $\ell = 1024$, let $\mathcal{F}_{Rabin} = \{f_{Rabin} : K \times D \to R\}$ be a trapdoor one-way function family based on Rabin cryptosystem, in which K is the set of all $N = pq$ such that p and q are distinct primes and $|p| = |q| = \ell/2$, $D = \mathbb{Z}_N^*$ and $R = QR_N$, with QR being the set of quadratic residues modulo N. Given a public key $k = N \in K$, the trapdoor one-way function is $f_{Rabin}(k, y) = y^2 \bmod N = M$. To compute the inverse, also given the trapdoor information (p,q), we can compute the square root $f_{Rabin}^{-1}((p,q), M) = M^{1/2} \bmod N$. Note that there are four square roots. Hence this trapdoor one-way function is a 4-to-1 function.

The initial state of (V_1, V_2) in this Rabin-based instantiation would be: $V_1 = (M, N_1, N_2, \ldots, N_n)$ and $V_2 = ((p_1, q_1), (p_2, q_2), \ldots, (p_n, q_n))$ which are generated in the same way as in the RSA-based instantiation above. To flip a bit at position i of the 'imaginary' n-bit binary vector $\vec{v}$, we carry out the following steps.

1. Set $j = 0$
2. Set $\widehat{M} = M + j \bmod N_i$. If $\widehat{M}$ is not a quadratic residue in $\mathbb{Z}_{N_i}^*$ [26, page 70], set $j = j + 1$ and repeat this step.
3. Compute $y_i = \widehat{M}^{1/2} \bmod N_i$. Note that $\widehat{M}^{1/2} \bmod N_i$ will give us four square roots, so we always set y_i to be the smallest one.

Suppose $i = 1$, the updated V_2 becomes $V_2 = (y_1, (p_2, q_2), \ldots, (p_n, q_n))$. Similar optimization method as described for the RSA case can also be applied to this Rabin-based instantiation.

4.3.3. Signatures

The second scheme is based on signatures. As described in the generic scheme we employ the 'imaginary' bit vector $\vec{v}$ and use its Hamming weight to indicate the path cost. The difference from the first scheme is that instead of using trapdoor functions, we use digital signature as the primitive for ensuring that the path cost is monotonically increasing. We will see that using digital signatures can help reduce the space required to represent the *PathCostTag-ActuallySent* compared to the trapdoor one-way function family based approach.

Let $SIG = (Gen, Sig, Ver)$ be a signature scheme. On input a security parameter $k \in \mathbb{N}$, the probabilistic polynomial-time algorithm (PPT) $Gen(k)$ generates a signing/verification key pair (sk, vk). On input a signing key sk and a message $m \in \{0,1\}^*$, $Sig(sk, m)$ generates a signature σ. On input a verification key vk, a message m and a signature σ, $Ver(vk, m, \sigma)$ outputs 1 if the signature is valid with respect to vk on m; otherwise it outputs 0. For security, we require SIG to be existentially unforgeable against chosen message attack [19].

In this generic scheme, we use the same structure of *PathCostTag* as shown in (4.3.2) on page 16. The only difference is the replacement of the trapdoor one-way functions $f_i (1 \leqslant i \leqslant n)$ in V_1 and their inverses $f_i^{-1} (1 \leqslant i \leqslant n)$ in V_2 by the verification keys $vk_i (1 \leqslant i \leqslant n)$ and the signing keys $sk_i (1 \leqslant i \leqslant n)$, respectively, each pair of the keys are generated independently using Gen. Without loss of generality, we assume that the message space of SIG is $\{0,1\}^k$. Hence the element M in V_1 is chosen uniformly at random from $\{0,1\}^k$.

Now suppose that we want to increase the path cost by 1, for example, by flipping the first bit of the 'imaginary' n-bit binary vector $\vec{v}$ from zero to one, we compute $y_1 \leftarrow Sig(sk_1, M)$ and replace sk_1 in V_2 by y_1. Thus the updated V_2 becomes $V_2 = (y_1, sk_2, \ldots, sk_n)$. To check whether the first bit of $\vec{v}$ is flipped or not, we run $Ver(vk_1, M, y_1)$ and determine if its output is 1, where vk_1 is obtained from V_1. If so, we conclude that the first bit is flipped.

We now also present two instantiations with slightly different properties.

BLS Short Signature Based The short signature by Boneh, Lynn and Shacham (BLS) [5] can be used to implement the scheme described above, with the advantage that each component in V_1 and V_2 is much smaller compared to RSA or Rabin. The signature uses bilinear pairings which are reviewed as follows.

Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order p, let g_1 a generator of $\mathbb{G}_1$ and g_2 a generator of $\mathbb{G}_2$. $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing if the following properties are satisfied: (1) **Bilinear**: for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p, e(P^a, Q^b) = e(P, Q)^{ab}$; (2) **Non-degenerate**: $e(g_1, g_2) \neq 1$; and (3) **Computable**: $e(P, Q)$ can be computed efficiently for all $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$. When practically implementing bilinear pairing, different types of pairings can be used with different speed and size

characteristics. We refer to this types as (A, D, F) Note that a standard curve (in Type A, D or F) can be used, in which case the curve parameters do not have to be included in *PathCostTag-ActuallySent*.

In BLS, a signing key sk is a random element in $\mathbb{Z}_p$, a verification key vk is computed as $vk = g_2^{sk}$. For a message $M \in \{0,1\}^*$, the signature is computed as $y \leftarrow H(M)^{sk}$ where $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ is a hash function. The verification is done by checking that $e(\sigma, g_2) = e(H(M), vk)$.

The size requirements for the different components in V_1 and V_2 are as follows:

	Type A (bits)	Type D (bits)	Type F (bits)
sk	≈160	≈160	≈160
vk	≈512	≈510	≈320
y	≈512	≈170	≈160

For all these types of bilinear pairings, we set the length of M to 160 bits.

Optimization through aggregation. In [5], Boneh et al. also suggested a method to aggregate the BLS short signatures. By applying aggregation, we can further reduce the size of *PathCostTag-ActuallySent* by aggregating the signatures in V_2 (i.e. those flipped bits of the 'imaginary' n-bit binary vector $\vec{v}$). Without loss of generality, suppose that the first l bits of $\vec{v}$ have been flipped, that is, V_2 would become $V_2 = (y_1, \ldots, y_l, sk_{l+1}, \ldots, sk_n)$ where $y_j = H(M)^{sk_j}$ for all j, $1 \leqslant j \leqslant l$. Instead of sending V_2 in this form, we can aggregate all the l short signatures into one single signature and the next V_2 would become $V_2^{new} = (y_{agg}, sk_{l+1}, \ldots, sk_n)$ where $y_{agg} = \prod_{j=1}^{l} y_j \in \mathbb{G}_1$. This aggregated signature can be verified by checking that $e(y_{agg}, g_2) = e(H(M), \prod_{j=1}^{l} vk_j)$. Note that not only has the size of the *PathCostTag-ActuallySent* (specifically, the size of V_2) been reduced, but we also reduced the computational complexity of the verification from $2l$ number of pairing evaluations to just two pairing evaluations.

ZSNS short signature based. Compared to BLS short signature, the short signature scheme from Zhang, Safavi-Naini and Susilo [40] (called ZSNS short signature in the later part of this paper) does not require a special map-to-point hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ which is generally implemented as a probabilistic algorithm. Instead, a conventional hash function $h : \{0,1\}^* \rightarrow \mathbb{Z}_p$ is enough. Below is a review of the ZSNS scheme.

The ZSNS scheme uses a symmetric pairing with $\mathbb{G}_1 = \mathbb{G}_2$ and therefore $g_1 = g_2 = g$. Note that the BLS short signature can also use a symmetric pairing (i.e. a bilinear pairing implementation which has a distortion map $\psi : \mathbb{G}_1 \rightarrow \mathbb{G}_2$). A signing key sk is a random element in $\mathbb{Z}_p$ and the corresponding verification key vk is computed as $vk = g^{sk}$. For a message $M \in \{0,1\}^*$, a signature is computed as $\sigma = g^{\frac{1}{h(M)+sk}}$. The verification is done by checking that $e(g^{h(M)} \cdot vk, \sigma) = e(g,g)$.

4.3.4. Hash tree

The third scheme is based on a Merkle hash tree to represent the imaginary bit vector. A Merkle hash tree is a data structure where data organized in the leaves of a tree is summarized upwards through hashing. The values of the whole tree can then be represented by a single hash value in the root node of the tree. Fig. 9 shows an example with 4 input values x_1, x_2, x_3, x_4. First, each of these values is hashed individually. Then, starting at the bottom, the two leaves of an intermediate node are hashed together and the intermediate node is replaced by the hashed value. This continues until the root node of the tree is calculated, which now depends on all the input values. A change in any of the leaves will change the value of the root node as well.

In this scheme the binary bit vector can be represented by a single vector V_1. Without loss of generality we assume the vector to be of length $n = 2^\tau$, in which case the hamming weight of the path cost has a range of 0 to n. We thus have a vector $V_1 = (x_1, \ldots, x_n)$ containing n randomly chosen values.

In the following we describe how the path cost for an announcement is created by a source node S and how it is processed as it propagates through the network.

1. S selects n random values $V_1 = (x_1, \ldots, x_n)$ of k bits each.
2. S builds a hash tree with the $x_1, \ldots, x_n$ as the leaves (Fig. 10a), then hashes the leaves and all intermediate nodes to calculate the root node r_0.
3. Node S calculates a signature $\mathrm{Sign}_S(S, T, r_0)$ over r_0, S and T.

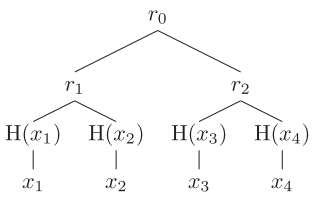

Fig. 9. A hash tree with input x_1, x_2, x_3, x_4. Consequently, $r_1 = H(H(x_1)||H(x_2))$, $r_2 = H(H(x_3)||H(x_4))$ and $r_0 = H(r_1||r_2)$.

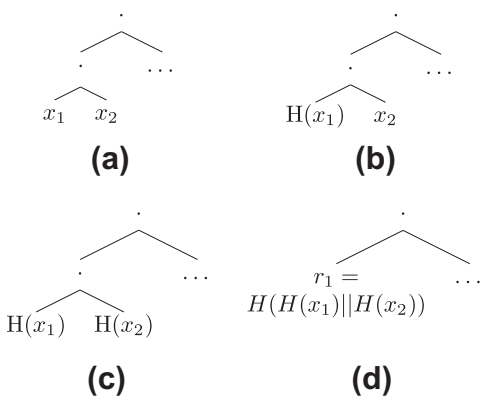

Fig. 10. Tree (a) is the initial tree. The first node transforms the first element, resulting in tree (b). The second node transforms the second element, obtaining tree (c) and then aggregates the two hashed values into one (tree (d)).

4. Assume that S already sets a path cost of *1* by hashing the first element x_1 (Fig. 10b).
5. Node S broadcasts an announcement with the following path cost to its neighbors:

   $$PathCostTag = \langle \mathsf{S}, r_0, \mathsf{Sign}_{\mathsf{S}}(\mathsf{S}, T, r_0), V_1 \rangle$$

   where $V_1 = (H(x_1), x_2, \ldots, x_n)$.
6. The next node R in the path receives the announcement and first calculates the root r'_0 of the hash tree, compares it with the r_0 included in the announcement and if they match it also verifies the signature.
7. Suppose that R wants to increase the path cost by 1, so it hashes x_2, then calculates the aggregate hash $H(H(x_1)||H(x_2))$ (Fig. 10c) and replaces x_1, x_2 (Fig. 10d), then forwards the announcement with the updated path cost to its neighbors:

   $$PathCostTag = \langle \mathsf{S}, r_0, \mathsf{Sign}_{\mathsf{S}}(\mathsf{S}, T, r_0), V'_1 \rangle$$

   where $V'_1 = (H(H(x_1)||H(x_2)), x_2, \ldots, x_n)$. Note that V'_1 only contains $n-1$ elements, one less than the initial V_1.
8. Subsequent nodes continue in the same fashion, first verifying the integrity of the tree and then hashing more elements and updating the tree.

If the announcement propagates far enough for the path cost to reach n, it will have the following structure:

$$PathCost = \langle \mathsf{S}, r_0, \mathsf{Sign}_{\mathsf{S}}(\mathsf{S}, T, r_0), V''_1 \rangle$$

where V''_1 only contains

$$V''_1 = (r_0)$$

We note that as the path cost increases, the size of the path cost tag decreases.

4.3.5. Comparison

Table 1 shows the size requirements for the different instantiations of the three schemes. In the comparison, we consider that the 'imaginary' n-bit binary vector $\vec{v}$ has a length of 100 bits (i.e. $n = 100$). For all schemes we consider 80-bit security, which determines the size of individual elements of the vector:

Table 1
Size comparison.

Instantiation	Empty vector (KB)	Full vector (KB)
RSA	12.5	25
Rabin	12.5	25
BLS (Type A)	2	12.5
BLS (Type D)	2	8.3
BLS (Type F)	2	5.9
BLS (aggr., Type A)	2	6.3
BLS (aggr., Type D)	2	6.2
BLS (aggr., Type F)	2	3.9
Zhang et al. (Type A)	2	12.5
Hash tree	2	0.02

Trapdoor: For both RSA and Rabin 80-bit security is equivalent to 1024 bit for M, N_i, y_i and 512 bit for p_i and q_i.
Signature: It takes about 320 bits for ECDSA (Elliptic Curve Digital Signature Algorithm) [1] and 160 bits for BLS or ZSNS short signature schemes (if Type F curve is used).
Hash tree: To achieve 80-bit security an element size of 160 bit is necessary. The individual elements should thus have a size of 160 bit and the hash function used should supply 160 bits of output.

In the table, the address S of the source, the signature $\mathrm{Sign}_S(S, T, V_1)$, and the random element M are not included in the size estimation as they are the same for all the instantiations and independent of the value of n. *Empty Vector* refers to the initial state of $\vec{v}$ where all the bits are zero, while *Full Vector* means that all the n bits of $\vec{v}$ have been flipped, so that it represents the maximum path cost supported.

4.4. Pseudo path

The Pseudo Path (PSP) component in a route announcement is a fixed-length random looking binary string which consists of N segments, each L bits long. For example, N could be 20 and L 128 bits. We will see shortly on how these values are set in practice. The purpose of the PSP is to detect routing loops. Each random-looking segment in the PSP component contains an L-bit pseudo-random binary string which is constructed so that no one can infer any useful information about the path. In the following, we describe the PSP component by describing how it is created and updated while propagating from one node to another.

In the anonymous overlay network, each node i randomly generates a k-bit secret symmetric key K_i (e.g. $k = 128$), a random string R_i of l_R bits long (e.g. $l_R = 80$) and a counter C_i of size l_C bits (e.g. $l_C = 48$) with an arbitrary initial value (*Note*: $L = l_R + l_C$). (K_i, R_i, C_i) are all kept secret and will not be shared with any other node.

When a new route announcement is created in the form shown in Fig. 7, the PSP component is initialized with random bits ($L \times N$ bits long in total). This is followed by replacing the last segment (i.e. the rightmost segment of the PSP component) with x_i where

$$x_i \leftarrow E(K_i, R_i \| C_i). \tag{1}$$

$E{:}\{0,1\}^k \times \{0,1\}^L \to \{0,1\}^L$ is a symmetric encryption algorithm (e.g. AES) and x_i is generated by encrypting "message" $(R_i \| C_i)$ under key K_i, in which the symbol '$\|$' denotes string concatenation.

After filling the last segment with x_i, the PSP component is rotated to the right so that the segment with x_i becomes the most significant segment (i.e. the leftmost segment of the PSP component). The two operations described above are illustrated in Fig. 11.

Note that under the assumption that the symmetric encryption algorithm is a pseudo-random function family [17], it is infeasible for any other node to distinguish x_i from any other randomly generated segment on the PSP component. After the two operations above, the route announcement is sent out to all the neighbors of node i. After sending the announcement, node i updates the counter C_i by increasing its value by one. The value of R_i remains static. When one of the neighbors, say node j, receives the route announcement, it checks for routing loops using the method described below in Section 4.4.1. If there is no routing loop, the node then updates its routing table for the entry of the overlay address specified in the **destination** component if the (accumulated) **path cost** of the route announcement are better than the current ones in its routing table.

If the entry in its routing table is updated, that is, a more efficient route to the destination specified in the route announcement is found, node j updates the route announcement and forwards it to all its neighbors except node i. To update the route

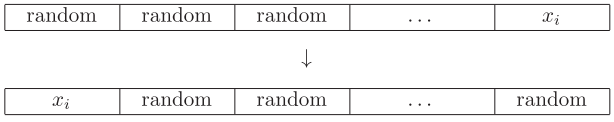

Fig. 11. This figure shows how the PSP component changes after replacing the last segment (upper diagram) and after rotation (lower diagram).

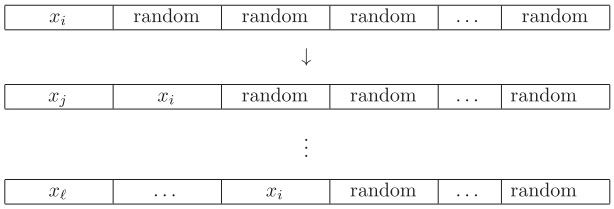

Fig. 12. The PSP component is updated by each node along the route.

announcement, node j computes the updated measures for the **path cost** component. Equivalently to Eq. (1) and the two operations described above, it computes $x_j \leftarrow E(K_j, R_j \| C_j)$, replaces the last segment of the PSP component and then rotates the PSP component to the right. The propagation then continues as illustrated in Fig. 12.

4.4.1. Loop detection

Suppose node i receives a route announcement from one of its neighbors. To detect if there is a routing loop, node i uses its symmetric key K_i to decrypt every segment in the PSP component. For each decrypted binary string, set the first l_R bits as R and the remaining l_C bits as C. If $\mathsf{R} \equiv R_i$ and C is strictly smaller than the current value of C_i, then the node i will conclude that a routing loop occurs and the announcement will be discarded.

The motivation for using the structure with K_i, R_i and C_i is that the node i can use them to generate random-looking strings $x_i \leftarrow E(K, R_i \| C_i)$ and only needs to remember these three values for being able to recognize any segment on a PSP component created by itself.

4.4.2. Pseudo path size

With reference to BGP-4 [33] used for inter-AS routing on the Internet, setting N to 20 should be a good starting point. The size l_R of the random component R_i and the size l_C of the counter C_i add up to L bits. If L is of size 128 bits, a possible division is 80 bits for l_R and 48 bits for l_C. This allows up to 2^{48} route announcements for each node before the counter wraps around. When the counter wraps around, the node will generate a new random string R_i' and a new counter C_i' and then starts using these new values while still remembering the old values for a period of time.

Remark 1. When a node leaves the system, all the nodes which have received routes from that node and propagated them to other neighbors will withdraw these routes, and they will also recalculate routes which were passing through the leaving node in the same way as establishing a route described above.

5. Evaluation

5.1. Overlay simulation

To verify the performance of the anonymous system, we conducted simulations using a custom-built Java-based, discrete routing simulator. The aim was to determine whether reasonable latency can be achieved within the overlay network. Furthermore, we used the *CAIDA IPv4 Routed /24 Topology Dataset* [20] to get more realistic estimates for latencies within the simulated network. We also used a geographical database for IP addresses[2] to properly distribute simulated users over the Internet. For the simulation we divided the overlay into 4 regions, America, Asia, Europe and Oceania.

5.1.1. CAIDA dataset

The *IPv4 Routed /24 Topology Dataset* records traceroutes to all routed /24 networks in the Internet, using a distributed set of so-called ARK monitors. The monitors are located in a number of countries, such as the Netherlands, the United States, Spain, Sweden, United Kingdom, Indonesia, Korea, Morocco, Australia, New Zealand, China and many more (for a complete list see [7]). The distribution is not uniform, however, with the US having 14 monitors, while China only has 2, one in Shenyang and one in Hong Kong. The ARK monitors are divided into 3 teams, and over a period of 2–3 days (called 1 *run*), each team will probe each /24 network once (which means that each individual monitor will only probe a small fraction of all /24 networks). Probing means that a monitor will send a traceroute to a random IP within a chosen /24 network and record the roundtrip time (RTT) for all intermediate nodes as well as the targeted node. Because of firewalls and packet filters, however, not all probes reach the targeted node, resulting in partial traceroutes.

For the simulation we used the accumulated data of 5 runs for each team (15 runs altogether), recorded in the time period from 2011/05/08 to 2011/05/17, which amounted to a total of approximately 135 million probes and 13 gigabytes of compressed data. Out of these 135 million probes, 12.7 million probes were complete (i.e., reaching the targeted node). This data was imported into a relational database for easy querying. During the import we also performed a few basic validity checks, e.g., removing measurements which were clearly too low (e.g., through inaccurate geographical IP information), or exceedingly high (possibly indicative of satellite Internet connections).

5.1.2. GeoIP database

A GeoIP database contains information about which country (or even which city, depending on the resolution of the database) a certain IP address block is located. This is not always completely accurate, but the database used in the simulation claims an accuracy of 99.5%.

[2] http://www.maxmind.com/app/geolitecountry.

5.1.3. Caveats

The CAIDA IPv4 routed /24 Topology Dataset is not perfectly suited to predict latencies between any two computers in the Internet, as all measurements originate from one of the more than 50 ARK monitors. This means that all estimates are in fact *triangular latencies* (see Fig. 13). As a consequence, estimated latencies are, except for some special cases, *higher* than they would be in reality.

5.1.4. Estimating latencies

Combining both the CAIDA dataset and the GeoIP database, it is possible to calculate estimates for the latency between two IP addresses IP_A and IP_B in the Internet. The following algorithms were used to estimate latencies:

1. Common monitor. If a specific monitor (located in either the country of IP_A or IP_B) has a complete traceroute for both IP_A and IP_B, the latency between those two hosts can be estimated by adding up the individual latencies lat_A and lat_B. This can be fairly accurate for latencies between IPs in different countries far apart in terms of latency, where the intra-country latency portion is small compared to the inter-country latency, but it can also be wildly inaccurate if the countries are close in terms of latency, or if both hosts are within the same country.
2. Distinct monitors. If a certain monitor X has a complete traceroute to IP_A, and another monitor Y has a traceroute to IP_B, a latency can be estimated by adding the latency from IP_A to monitor X, the latency between monitor X and Y, and the latency between monitor Y and IP_B. The accuracy of this again heavily depends on the placement of the monitors in the whole path between IP_A and IP_B.
3. Country-level statistics. If neither common, nor distinct monitors are available for a certain pair of IP addresses IP_A and IP_B, a country-level latency can be estimated by averaging the pair-wise latency of network blocks in two countries. Depending on the size of a country, these are typically fairly accurate estimates.

When using these algorithms to estimate latencies, only complete traceroutes were considered. Also, we only included countries in the simulation which contained at least one ARK monitor, as estimates between countries without ARK monitors are unpredictably inaccurate. The final list of countries or regions thus included is as follows:

America: Brazil, Canada, Mexico, United States of America
Asia: China, Hong Kong (China), Indonesia, Japan, South Korea, Philippines, Taiwan
Europe: Austria, Finland, Germany, Greece, Iceland, Ireland, Italy, Netherlands, Norway, Romania, Spain, Sweden, Switzerland
Oceania: Australia, New Zealand

5.1.5. Simulator

The simulator models how nodes send out routing information and the propagation of that routing information through the overlay network in discrete iterations until a steady state has been reached. From the steady state we then infer the latencies for different paths through the network. While the simulation currently only focuses on routing, the use of the CAIDA dataset to estimate latencies ensures that the results also include delays from other layers such as queuing delays, etc.

For each run the simulator creates a model network by adding 1500 nodes in total, distributed across the 4 regions (America, Asia, Europe, Oceania) according to the proportional distribution of /24 across these 4 regions. Specifically, the simulator loads a list of all /24 network blocks of the countries for each region and then randomly assigns each node to a /24 network block in that specific region. This ensures that the distribution of nodes among countries reflects the actual distribution in the Internet (as determined by the number of /24 network blocks assigned to each country). The CAIDA dataset is then used to estimate latencies between connected nodes, as described above.

5.2. Results

To create an anonymous overlay network over the modeled Internet described above, we let each node connect to a number of randomly chosen nodes and then determine the latencies between different parts of the overlay. Specifically, a node will connect to a certain number of nodes in the same region, and a certain number of nodes in different regions. Fig. 14 shows the continental latencies for different regions, i.e., the average one-way latency between two nodes in the same re-

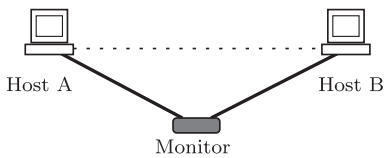

Fig. 13. Latency estimation using ARK monitors results in triangular latencies, which are typically higher than actual latencies.

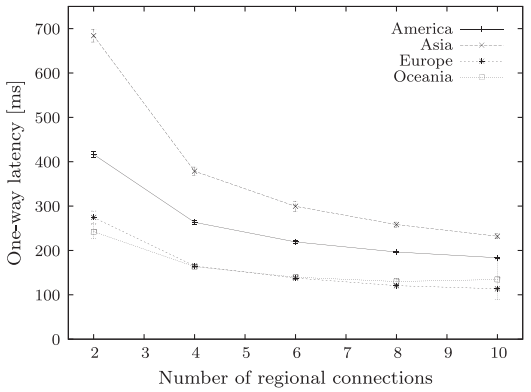

Fig. 14. Evolution of estimated latencies within regions as nodes increase the number of regional connections.

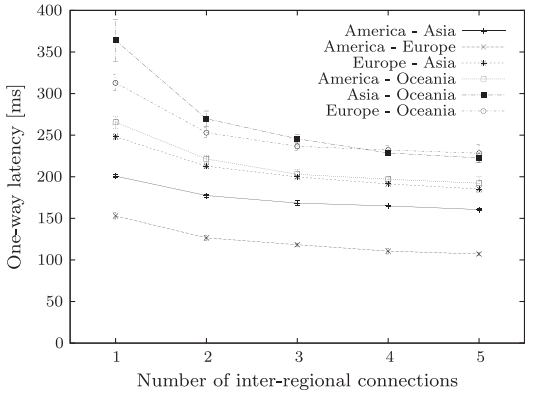

Fig. 15. Evolution of estimated latencies across regions as nodes increase the number of inter-regional connections.

gion. Fig. 15 show the intercontinental latencies between different regions, i.e., the average one-way latency from a specific node to the closest node in the destination region.

Table 2 shows the average latency (RTT) for the communication between any two nodes within the same region.

Similarly, Table 3 shows the RTT for inter-regional communication between the four different regions (i.e., the latency from any node to the closest node in the destination region). Combining these two results, we can estimate the RTT for the communication between any two nodes within the overlay, as shown in Table 4.

5.3. Discussion

For a connection made between any two nodes in the overlay network, no matter whether in the same region or not, the Round-Trip-Time (RTT) we simulated is (with one exception) consistently below 1 s. In addition, connections between nodes in the same region range from 227 ms to 464 ms RTT while inter-regional connections are in the order of 306 ms to 728 ms.

Furthermore, these estimates are based on real-world measurement data (the CAIDA dataset), which inherently includes phenomena such as queuing delays. Nevertheless, as described above, it is likely that latency estimates are in most cases slightly pessimistic, as the CAIDA dataset only allows to estimate triangular latencies.

A direct comparison with systems which are currently used to achieve anonymity is not straightforward, but according to the numbers determined by Wendolsky et al. [38], these systems have an initial delay when accessing a webpage of between

Table 2
RTT for intra-regional communications for 10 regional connections per node.

America	367 ms
Asia	464 ms
Europe	227 ms
Oceania	267 ms

Table 3
RTT for inter-regional communications for 5 inter-regional connections per node.

America–Asia	402 ms
America–Europe	306 ms
Asia–Europe	496 ms
America–Oceania	531 ms
Asia–Oceania	728 ms
Europe–Oceania	626 ms

Table 4
Total RTT for communications between any two nodes in different regions, for 10 regional and 5 inter-regional connections.

America–Asia	768–866 ms
America–Europe	533–673 ms
Asia–Europe	723–960 ms
America–Oceania	800–898 ms
Asia–Oceania	997–1192 ms
Europe–Oceania	853–895 ms

at least 1000 ms and 7000 ms with a mean of anywhere between 1200 ms to 4700 ms depending on the system. Some systems also incur occasional additional delays (i.e. circuit establishment in Tor).

6. Conclusion

In this article we presented a new anonymous network together with an authenticated anonymous routing protocol which detects and prevents path cost reduction attack. The anonymous network is in the form of an overlay built on top of the conventional non-anonymous Internet, creating a separate address space which is disjoint from the underlying nodes, providing anonymity for the overlay network. Furthermore, we presented an anonymous routing protocol designed to provide routing within the overlay network to achieve good performance, without leaking network topology information. The routing protocol described in this article not only allows the establishment of routes, it also contains mechanisms designed to prevent malicious nodes from interfering with the routing process to gain an advantage. In total we presented three schemes to secure the routing process, with a diverse number of concrete instantiations and we also compared the relative performance of the different instantiations.

Concerning the anonymous overlay network we also presented preliminary simulation results about the performance, which indicated that latency-wise performance is good compared to existing systems. We also discussed and proposed solutions for several important implementation issues, which include the overlay namespace, geographical awareness for better overlay network performance and anycast servers for anonymous access to normal Internet services.

Future work includes evaluating the performance of this new network not only in terms of latency but also in terms of bandwidth, looking into the performance of the anonymous routing protocol under highly dynamic situations (e.g. lots of nodes continuously joining and leaving the network) and finding ways to reduce the size of the different routing schemes even more.

References

[1] Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-3, National Institute of Standards and Technology, 2009.
[2] J. Abley, K. Lindqvist, Operation of anycast services, RFC 4786 (Best Current Practice), December 2006. <http://www.ietf.org/rfc/rfc4786.txt>.
[3] D. Bein, S. Zheng, Energy efficient all-to-all broadcast in all-wireless networks, Information Sciences 180 (10) (2010) 1781–1792.
[4] S. Blake-Wilson, A. Menezes, Authenticated Diffie–Hellman key agreement protocols, Lecture Notes in Computer Science 1556 (1999) 339–361.
[5] D. Boneh, B. Lynn, H. Shacham, Short signatures from the Weil pairing, in: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, ASIACRYPT '01, Springer-Verlag, London, UK, 2001. <http://portal.acm.org/citation.cfm?id=647097.717005>.
[6] A. Boukerche, K. El-Khatib, L. Xu, L. Korba, A novel solution for achieving anonymity in wireless ad hoc networks, in: Proceedings of the 1st ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, PE-WASUN '04, ACM, New York, NY, USA, 2004. <http://doi.acm.org/10.1145/1023756.1023763>.
[7] CAIDA, CAIDA Monitors: The Archipelago Measurement Infrastructure. <http://www.caida.org/data/monitors/monitor-map-ark.xml>.
[8] C.-Y. Chang, C.-T. Chang, T.-S. Chen, H.-R. Chang, Hierarchical management protocol for constructing a QoS communication path in wireless ad hoc networks, Information Sciences 177 (13) (2007) 2621–2641.
[9] D. Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms, Communications ACM 24 (2) (1981) 84–90.
[10] D. Chaum, The dining cryptographers problem: unconditional sender and recipient untraceability, Journal of Cryptology 1 (1) (1988) 65–75.
[11] W. Diffie, M. Hellman, New directions in cryptography, IEEE Transactions on Information Theory 22 (6) (1976) 644–654.
[12] R. Dingledine, N. Mathewson, Anonymity loves company: usability and the network effect, Designing Security Systems That People Can Use, O'Reilly Media, 2006.
[13] R. Dingledine, N. Mathewson, P. Syverson, Tor: the second-generation onion router, in: Proceedings of the 13th USENIX Security Symposium, 2004.

[14] K. El-Khatib, L. Korba, R. Song, G. Yee, Secure dynamic distributed routing algorithm for ad hoc wireless networks, in: International Conference on Parallel Processing Workshops (ICPPW 03), 2003.
[15] M.J. Freedman, R. Morris, Tarzan: a peer-to-peer anonymizing network layer, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, 2002.
[16] S. Goel, M. Robson, M. Polte, E. Sirer, Herbivore: a scalable and efficient protocol for anonymous communication, Technical Report TR2003-1890, Cornell University Computing and Information Science, 2003.
[17] O. Goldreich, Foundations of Cryptography: Basic Tools, Cambridge University Press, 2001.
[18] D. Goldschlag, M. Reed, P. Syverson, Onion routing for anonymous and private internet connections, Communications ACM 42 (1999) 39–41.
[19] S. Goldwasser, S. Micali, R. Rivest, A digital signature scheme secure against adaptive chosen-message attack, SIAM Journal on Computing 17 (2) (1988) 281–308.
[20] Y. Hyun, B. Huffaker, D. Andersen, E. Aben, C. Shannon, M. Luckie, kc claffy, The CAIDA IPv4 Routed/24 Topology Dataset – 2011/05/08 – 2011/05/17. <http://www.caida.org/data/active/ipv4_routed_24_topolog% y_dataset.xml>.
[21] I2P anonymous network. <http://www.i2p2.de/>.
[22] S. Kent, C. Lynn, K. Seo, Secure border gateway protocol (S-BGP), IEEE Journal on Selected Areas in Communications 18 (4) (2000) 582–592.
[23] C.-T. Li, M.-S. Hwang, A lightweight anonymous routing protocol without public key en/decryptions for wireless ad hoc networks, Information Sciences, in press (corrected proof, 2011). <http://www.sciencedirect.com/science/article/pii/S00200%25511003458>.
[24] S. Manvi, M. Kakkasageri, Multicast routing in mobile ad hoc networks by using a multiagent system, Information Sciences 178 (6) (2008) 1611–1628.
[25] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, D. Sicker, Shining light in dark places: understanding the Tor network, in: Privacy Enhancing Technologies, Springer, 2008.
[26] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 2001. <http://www.cacr.math.uwaterloo.ca/hac/>.
[27] P. Mittal, F. Olumofin, C. Troncoso, N. Borisov, I. Goldberg, PIR-Tor: scalable anonymous communication using private information retrieval, in: D.A. Wagner (Ed.), Proceedings of the 20th USENIX Security Symposium, San Francisco, CA, USA, August 10–12, 2011.
[28] C. Ok, S. Lee, P. Mitra, S. Kumara, Distributed routing in wireless sensor networks using energy welfare metric, Information Sciences 180 (9) (2010) 1656–1670.
[29] A. Panchenko, L. Pimenidis, J. Renner, Performance analysis of anonymous communication channels provided by Tor, in: Third International Conference on Availability, Reliability and Security, 2008 (ARES 08), 2008.
[30] P. Papadimitratos, Z. Haas, Secure routing for mobile ad hoc networks, in: SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), San Antonio, TX, 2002.
[31] M. Rabin, Digitalized signatures and public-key functions as intractable as factorization, Tech. rep., Massachusetts Institute of Technology Cambridge, MA, USA, 1979.
[32] T. Rajendran, K.V. Sreenaath, Secure anonymous routing in ad hoc networks, in: Proceedings of the 1st Bangalore Annual Compute Conference, COMPUTE '08, ACM, New York, NY, USA, 2008. <http://doi.acm.org/10.1145/1341771.1341791>.
[33] Y. Rekhter, T. Li, S. Hares, A Border Gateway Protocol 4 (BGP-4), RFC 4271 (Draft Standard), January 2006. <http://www.ietf.org/rfc/rfc4271.txt>.
[34] R.L. Rivest, A. Shamir, L.M. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM 21 (2) (1978) 120–126.
[35] University of Regensburg, JAP anonymity & privacy. <http://anon.inf.tu-dresden.de/>.
[36] T. Wan, E. Kranakis, P.C. van Oorschot, Pretty secure BGP, psBGP, in: Proceedings of the Network and Distributed System Security Symposium, NDSS 2005, San Diego, California, USA, 2005.
[37] N.-C. Wang, Y.-F. Huang, J.-C. Chen, A stable weight-based on-demand routing protocol for mobile ad hoc networks, Information Sciences 177 (24) (2007) 5522–5537.
[38] R. Wendolsky, D. Herrmann, H. Federrath, Performance comparison of low-latency anonymisation services from a user perspective, Lecture Notes in Computer Science 4776 (2007) 233–253.
[39] R. White, Securing BGP through secure origin BGP (soBGP), Business Communications Review 33 (2003) 47–53.
[40] F. Zhang, R. Safavi-Naini, W. Susilo, An efficient signature scheme from bilinear pairings and its applications, in: Public Key Cryptography, 2004.
[41] R. Zhang, Y. Zhang, Y. Fang, Aos: an anonymous overlay system for mobile ad hoc networks, Wireless Networks 17 (2011) 843–859. doi:10.1007/s11276-010-0319-2. <http://dx.doi.org/10.1007/s11276-010-0319-2>.
[42] Y. Zhang, W. Liu, W. Lou, Anonymous communications in mobile ad hoc networks, in: INFOCOM 2005. Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, 2005.
[43] B. Zhu, Z. Wan, M. Kankanhalli, F. Bao, R. Deng, Anonymous secure routing in mobile ad-hoc networks, in: 29th Annual IEEE International Conference on Local Computer Networks 2004, 2004.
[44] L. Zhuang, Z. Feng, B.Y. Zhao, A. Rowstron, Cashmere: resilient anonymous routing, in: NSDI 05: 2nd Symposium on Networked Systems Design & Implementation, 2005.