Monotonically Increasing Bit Vector for Authenticated Anonymous Routing

Roman Schlegel and Duncan S. Wong

Department of Computer Science
City University of Hong Kong
sschlegel2@student.cityu.edu.hk, duncan@cs.cityu.edu.hk

Abstract. Anonymous routing where data packets can be routed efficiently while hiding the topology of the network from all nodes is a crucial part of achieving anonymity in an efficient anonymous network. 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 launching various attacks such as eavesdropping or man-in-the-middle attacks. In this paper we propose a generic scheme and a concrete instantiation to transform a routing protocol into an authenticated one in the sense that the path cost cannot be reduced by a malicious node.

1 Introduction

Consider an anonymous network in which there are L nodes, each one of them has a unique address (which is not necessarily a real IP address but a virtual address and might already be providing some degree of anonymity to the underlying node). Apart from knowing its neighbors, a node does not have any additional information about the topology of the network. In other words, each node only knows how to reach its neighbors; but has no idea on how other nodes are connected in the network. A network of this type is typically referred to as an anonymous network [16,15,12].

One fundamental problem in an anonymous network is to find out 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 [16,15,12], 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.

(*Lack of Authentication*) However, in these protocols, there is no *authentication*. They assume that nodes are honest and do not maliciously alter the path cost when updating a routing announcement. In fact, if any node in their networks is malicious, the node can arbitrarily reduce the path cost value carried by a route announcement message. By doing this, this malicious node may be able to act as a sink in the network and route all the traffic to itself. This may help the node to launch various attacks such as eavesdropping or man-in-the-middle attacks.

Some non-anonymous routing protocols, such as S-BGP [5], soBGP [14] and psBGP [13] support authentication so that any malicious reduction of path cost value of a traversing route announcement message can be detected by other nodes in the network. However, these routing protocols inherently leak the network topology information.

A natural question to ask is whether it is possible to transform an existing anonymous routing protocol into an authenticated version so that it not only maintains the anonymity property, but also ensures that no malicious node can reduce the path cost value of any route announcement message when the message is traversing the node.

Our Results.

In this paper, we propose a solution which can be used to convert distance-vector based anonymous routing protocols such as [16,15,12] to authenticated ones so that the path cost of routing messages cannot be reduced maliciously. At the same time, the anonymity of these protocols is maintained so that there is no leakage of network topology information due to using this new solution.

We first propose a generic solution which is relying on signature schemes. By using elliptic curve and bilinear pairing, we then construct an efficient concrete scheme with low memory requirement.

Paper Organization. The rest of the paper is organized as follows: section 2 discusses related work and section 3 describes the attack model considered in our scheme. We then present the a solution based on signature schemes in section 4. This is followed by a performance evaluation of size and computational complexity in section 5 and in section 6 we conclude.

2 Related Work

Route authentication has especially been discussed in recent years concerning BGP [11], the inter-domain routing protocol used in the Internet. BGP works by exchanging routing information with peers, but this information is not authenticated which makes it akin to listening to and trusting *hearsay*.

There are a number of other acknowledged vulnerabilites of BGP [8] and several proposals like S-BGP [5,6], psBGP [13] and soBGP [14] have been made to remedy this situation. These three proposals have in common that they try to fix routing problems because of misconfiguration or possibly malicious rerouting of traffic by *authenticating* all routing information. To achieve this, AS numbers are certified, as are address ranges (to ensure that entities are authorized to announce a specific prefix), and, depending on the proposal, BGP speakers (to ensure that only authorized speakers can participate). The proposals differ mainly in the way these certifications are made.

The three protocols ([5,13,14]) all leak information about the nodes in the network. For an anonymous network outlined in Sec. 1, on the other hand, this is not desirable. In order to ensure authentication while providing anonymity to the network, we emphasize that the secure anonymous routing protocol should make sure that the cost along a path cannot be maliciously reduced by any malicious node along a route and *at the same time* the origin of every single route announcement message should be authenticated. This prevents a node from positioning itself as a sink and from advertising a destination other than itself.

Anonymous routing, on the other hand, has been studied especially in connection with mobile ad-hoc networks (MANET). Protocols like [16,3,10,2,9,15] are all focused on MANETs and 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.

The anonymous routing protocol described in [12] fulfills the requirements of establishing routes without leaking information about the topology of the network, but it is also vulnerable to a malicious node which arbitrarily reduces the path cost when forwarding a route announcement. Our solution below can be applied directly to this protocol for solving the path-reduction problem.

3 Attack Model

Before proposing our solution we discuss the adversarial capabilities of a malicious network in more detail in this section.

As introduced in Sec. 1, we consider a typical network of L nodes where each node is connected to a few neighbors. Each node knows the address of its neighbors, but other than that it has no information about how other nodes in the network are connected. By running a distance-vector based routing protocol, efficient routes can be establisehd. We refer readers to [12] for a recently proposed anonymous routing protocol which is distance-vector based.

Figure 1 shows an example of a network where $L = 9$. The shaded node denoted by S initiates an announcement, with an initial path cost of 1 (for example as the hop count), which is then increased by each node and forwarded.

In this example, suppose all the nodes are honest. 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 (which our solution will prevent), 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.

To prevent malicious node M from reducing the path cost value carried by a route announcement message when forwarding it to node D, one idea is to

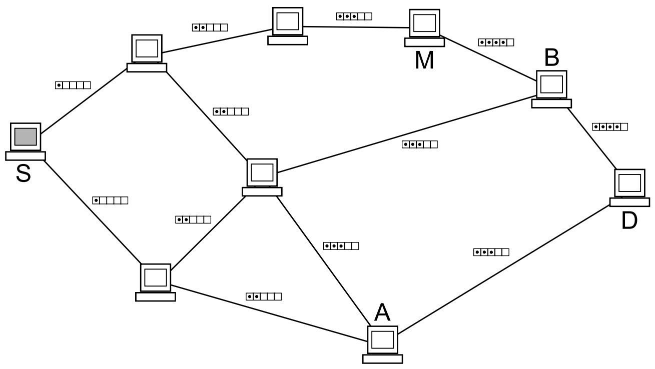

Fig. 1. An example network where the shaded node broadcasts an announcement. Each node increases the path cost value of the corresponding route announcement message by one and forwards it.

add some authentication mechanism to each route announcement message so that the next node (i.e. node D) can verify whether the path cost value in the receiving route announcement message has been maliciously reduced.

As discussed above, we target to convert existing anonymous routing protocols such as those in [16,15,12] *without* modifying any of their routing mechanisms. Our solution would only modify the path cost value. To achieve this, we introduce an authentication mechanism so that the path cost value carried by their route announcement messages cannot be maliciously reduced. Also, this additional authentication mechanism does not leak any information about the network topology. The advantages of this authentication approach is that the solutions can be applied directly to various anonymous routing protocols and also help maintain the anonymity properties of the anonymous routing protocols.

4 Generic Scheme

Our idea stems from the creation of a binary vector of length n bits: $\boldsymbol{v} = (b_1 b_2 \ldots b_n)$. The value n denotes the maximum possible value the path cost of a route announcement may reach. The Hamming weight of this binary vector $\boldsymbol{v}$ is the path cost. A node who wants to increase the path cost by 1 picks a zero bit on $\boldsymbol{v}$ and flips it to one, that is, it increases the Hamming weight of $\boldsymbol{v}$ by one. Our solution below will introduce mechanisms to ensure that the Hamming weight of $\boldsymbol{v}$ can only be monotonically increased (i.e. either leave it unchanged or increase it, but it cannot be reduced). The binary vector $\boldsymbol{v}$ also does not need to be sent explicitly but can instead be generated from the authentication mechanism introduced in our solution. Hence we refer to $\boldsymbol{v}$ as an 'imaginary' n-bit binary vector in the remaining part of this paper.

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

$$PathCostTag = < \mathsf{S}, V_1, V_2, \mathsf{Sign}_{\mathsf{S}}(\mathsf{S}, T, V_1) > \tag{1}$$

where S is the address of the source (e.g. node S in Fig. 1), 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 $\boldsymbol{v}$.

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 [4].

In the generic scheme, we use the structure of *PathCostTag* as shown in (1). V_1 is composed of the collection of verification keys vk_i $(1 \le i \le n)$ and V_2 contains the corresponding signing keys sk_i $(1 \le i \le n)$. Each pair of the keys are generated independently using Gen. Additionally, V_1 contains a message M. Without loss of generality, we assume that the message space of SIG is $\{0,1\}^k$. 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 $\boldsymbol{v}$ 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 $V_2 = (y_1, sk_2, \cdots, sk_n)$. To check whether the first bit of $\boldsymbol{v}$ 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.

In the following, we describe an instantiation of the generic scheme above. It is based on a short signature constructed from bilinear pairing.

4.1 BLS Short Signature Based

The short signature by Boneh, Lynn and Shacham (BLS) [1] can be used to implement the generic scheme above, with the advantage that each component in V_1 and V_2 is quite small. The signature uses bilinear pairing which is 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. The types we refer to (A, D, F) will be further explained in Sec. 5.2.

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 $H : \{0,1\}^* \to \mathbb{G}_1$ is a hash function. The verification is done by checking that $e(\sigma, g_2) = e(H(M), vk)$.

Optimization through Aggregation. In [1], Boneh et al. also suggested a method to aggregate the BLS short signatures. By applying aggregation, we can further reduce the size of *PathCostTag* by aggregating the signatures in V_2 (i.e. those flipped bits of the 'imaginary' n-bit binary vector $\boldsymbol{v}$). Without loss of generality, suppose that the first l bits of $\boldsymbol{v}$ have been flipped, that is, V_2 would become $V_2 = (y_1, \cdots, y_l, sk_{l+1}, \cdots, sk_n)$ where $y_j = H(M)^{sk_j}$ for all j, $1 \leq j \leq l$. 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 $V_2^{new} = (y_{agg}, sk_{l+1}, \cdots, sk_n)$ 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* (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.

5 Performance Evaluation

5.1 Size

Table 1 shows the size requirements depending on the signature scheme and the type used. In the comparison, we consider a 'imaginary' n-bit binary vector $\boldsymbol{v}$ of length 100 bits (i.e. $n = 100$). We only consider the size of the elements themselves for a security level of 80 bits and do not include the invariant parameters.

In the table, *Empty Vector* refers to the initial state of $\boldsymbol{v}$ where all the bits are zero, while *Full Vector* means that all the n bits of $\boldsymbol{v}$ have been flipped, so that it represents the maximum path cost supported. In other words, the sizes shown in Table 1 in the *Empty Vector* column indicate the lower bound for the size of each instantiation and under the *Full Vector* column the upper bound for the size of each instantiation is given.

Table 1. Size Comparison

Instantiation	*Empty Vector*	*Full Vector*
BLS (Type A)	2 KB	12.5 KB
BLS (Type D)	2 KB	8.3 KB
BLS (Type F)	2 KB	5.9 KB
BLS (aggr., Type A)	2 KB	6.3 KB
BLS (aggr., Type D)	2 KB	6.2 KB
BLS (aggr., Type F)	2 KB	**3.9 KB**

5.2 Computational Complexity

To test the computational complexity of our scheme, we performed benchmarks for typical operations in the different schemes. We used the Pairing Based Cryptography Library from Stanford University [7]. Once again, we consider 80-bit level security.

Table 2. Benchmark results for different instantiations

Scheme	Gen. Priv. Keys	Gen. Pub. Keys	Transf. Elem.	Verify Elem.
BLS (Type A)	$1601ms$	$524ms$	$524ms$	$1150ms$
BLS (Type D)	$1601ms$	$1925ms$	$163ms$	$2935ms$
BLS (Type F)	$1601ms$	$405ms$	$163ms$	$18754ms$
BLS agg. (Type A)	$1598ms$	$524ms$	$526ms$	$14ms$
BLS agg. (Type D)	$1595ms$	$1922ms$	$163ms$	$39ms$
BLS agg. (Type F)	$1602ms$	$402ms$	$163ms$	$188ms$

The benchmark consists of performing key generation, element transformation and element verification. For the tests we set $n = 100$, i.e. a vector contains 100 elements. All tests were done on a computer running Mac OS X 10.5.7 on an Intel Core 2 Duo 2.4 GHz processor. The computation time required for the different operations was determined using the `getrusage()` system call.

The PBC library [7] supports different types of bilinear pairings with different characteristics. For the BLS signature schemes we used types A, D and F. Type F has smaller elements, but is much slower than type D. Type A is the fastest for the pairings but has a larger element size than the other two types.

5.3 Discussions

Looking at tables 1 and 2 we can see that there is a distinct trade-off to be made between size and computational complexity depending on the type of bilinear pairing used.

The most important operation is the verification of elements of a vector received by a neighbor, because it is the most frequently done operation. Schemes where the verification is fast are therefore more desirable. A good compromise between size and computational complexity is the BLS aggregate signature with pairing of type A. Using type D or F would decrease the size further, at a slightly higher verification cost.

6 Conclusion

We proposed a scheme and a concrete instantiation which can be used to generically convert distance-vector based anonymous routing protocols such as [16,12] to authenticated ones so that the path cost of routing messages cannot be reduced maliciously. At the same time, the anonymity of these protocols is maintained so no network topology information is leaked due to using these new solutions. An interesting question which remains is how to further reduce the size of these schemes without introducing any significant computational burden to network nodes or making many assumptions on the nodes which may undermine the anonymity of the underlying network.

References

1. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
2. Boukerche, A., El-Khatib, K., Xu, L., Korba, L.: A novel solution for achieving anonymity in wireless ad hoc networks. In: Ould-Khaoua, M., Zambonelli, F. (eds.) PE-WASUN, pp. 30–38. ACM Press, New York (2004)
3. El-Khatib, K., Korba, L., Song, R., Yee, G.: Secure dynamic distributed routing algorithm for ad hoc wireless networks. In: International Conference on Parallel Processing Workshops (ICPPW 2003), pp. 359–366 (2003)
4. Goldwasser, S., Micali, S., Rivest, R.: A digital signature scheme secure against adaptive chosen-message attack. SIAM J. Computing 17(2), 281–308 (1988)
5. Kent, S., Lynn, C., Seo, K.: Secure Border Gateway Protocol (S-BGP). IEEE Journal on Selected Areas in Communications 18(4), 582–592 (2000)
6. Kent, S.T., Lynn, C., Mikkelson, J., Seo, K.: Secure Border Gateway Protocol (S-BGP) - Real World Performance and Deployment Issues. In: NDSS (2000)
7. Lynn, B.: PBC Library, `http://crypto.stanford.edu/pbc/`
8. Murphy, S.: BGP Security Vulnerabilities Analysis. RFC 4272 (Informational) (January 2006)
9. Papadimitratos, P., Haas, Z.: Secure routing for mobile ad hoc networks. In: SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), San Antonio, TX, pp. 01–27 (2002)
10. Rajendran, T., Sreenaath, K.V.: Secure anonymous routing in ad hoc networks. In: Shyamasundar, R.K. (ed.) Bangalore Compute. Conf., p. 19. ACM Press, New York (2008)
11. Rekhter, Y., Li, T., Hares, S.: A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard) (January 2006)
12. Schlegel, R., Wong, D.S.: Low Latency High Bandwidth Anonymous Overlay Network with Anonymous Routing. Cryptology ePrint Archive, Report 2009/294 (2009), `http://eprint.iacr.org/`
13. Wan, T., Kranakis, E., Van Oorschot, P.C.: Pretty Secure BGP (psBGP). In: NDSS (2005)
14. White, R.: Securing BGP through secure origin BGP (soBGP). Business Communications Review 33, 47–53 (2003)
15. Zhang, Y., Liu, W., Lou, W.: Anonymous communications in mobile ad hoc networks. In: INFOCOM, pp. 1940–1951. IEEE, Los Alamitos (2005)
16. Zhu, B., Wan, Z., Kankanhalli, M., Bao, F., Deng, R.: Anonymous Secure Routing in Mobile Ad-Hoc Networks. In: 29th Annual IEEE International Conference on Local Computer Networks, 2004, pp. 102–108 (2004)