A Scalable Wireless Routing Protocol Secure against Route Truncation Attacks

Amitabh Saxena[1] and Ben Soh[2]

[1] Dept. of Computer Science, University of Mannheim, Germany
[2] Dept. of Computer Science, La Trobe University, Australia

Abstract. Wireless routing protocols allow transmitting nodes to have some knowledge of the topology in order to decide when to forward a packet (via broadcast) and when to drop it. Since a routing protocol forms the backbone of any network, it is a lucrative target for attacks. Routing protocols for wired networks (such as S-BGP) are not scalable in an ad-hoc wireless environment because of two main drawbacks: (1) the need to maintain knowledge about all immediate neighbors (which requires a discovery protocol), and (2) the need to transmit the same update several times, one for each neighbor. Although information about neighbors is readily available in a fairly static and wired network, such information is often not updated or available in an ad-hoc wireless network with mobile devices. Consequently, S-BGP is not suitable for such scenarios. We propose a BGP-type wireless routing protocol for such networks that does not suffer from such drawbacks. The protocol uses a novel authentication primitive called Enhanced Chain Signatures (ECS).

1 Introduction

With the advent of wireless technology, mobile ad-hoc networks (MANET) is an active area of research. The primary challenge in MANETs is equipping each device to continuously maintain the information required to properly route traffic. We investigate the use of BGP-type routing protocols in such networks.

The Border Gateway Protocol (BGP) [1,2] is a *path vector* routing protocol [3], in which routers repeatedly advertise routes to their immediate neighbors. On receiving an update, a router checks if the advertised route is better than an existing one. If so, the router updates its table and advertises the new route to all its other immediate neighbors. BGP, however, has many security vulnerabilities [4,5]. For instance, a rogue router could claim a shorter route to some destination in order to intercept traffic. Therefore, modern implementations use a modified variant called Secure-BGP (S-BGP) [6,7]. In S-BGP, routers authorize each neighbor using route attestations (RAs) and updates are peer-specific. In wireless networks, a node with several receivers in its vicinity must first establish the identity of each receiver, and then broadcast as many updates. Such control plane traffic is a bottleneck when the devices are densely distributed.

In [8], an authentication primitive called Chain Signatures (CS) is proposed. As an application, a secure routing protocol called Stateless Secure BGP

B. De Decker and I. Schaumüller-Bichl (Eds.): CMS 2010, LNCS 6109, pp. 4–15, 2010.

(SS-BGP) is also briefly discussed. The attack scenario described in [8] is that of a *route truncation attack* in wired networks. SS-BGP differs from S-BGP in that updates are not peer-specific and can therefore be broadcast. However, it is not clear if SS-BGP has any real advantage in wired networks, since true broadcast channels do not exist. Wireless networks, on the other hand, present a perfect application scenario for SS-BGP because they provide true broadcast channels without the ability to control or determine who receives a broadcast.

We extend the work in [8] by presenting an attack scenario for wireless networks that clearly demonstrates the advantages of SS-BGP. We then describe an improved SS-BGP protocol using an extension of CS called Enhanced Chain Signatures (ECS). Although we focus only on path vector routing using BGP, the main ideas carry over to other routing methods. In the rest of this paper, we consider a highly simplified variant of (S)-BGP that is sufficient for our purpose.

2 Route Truncation Attacks

The discussion of this section is based on the network of Fig. 1.

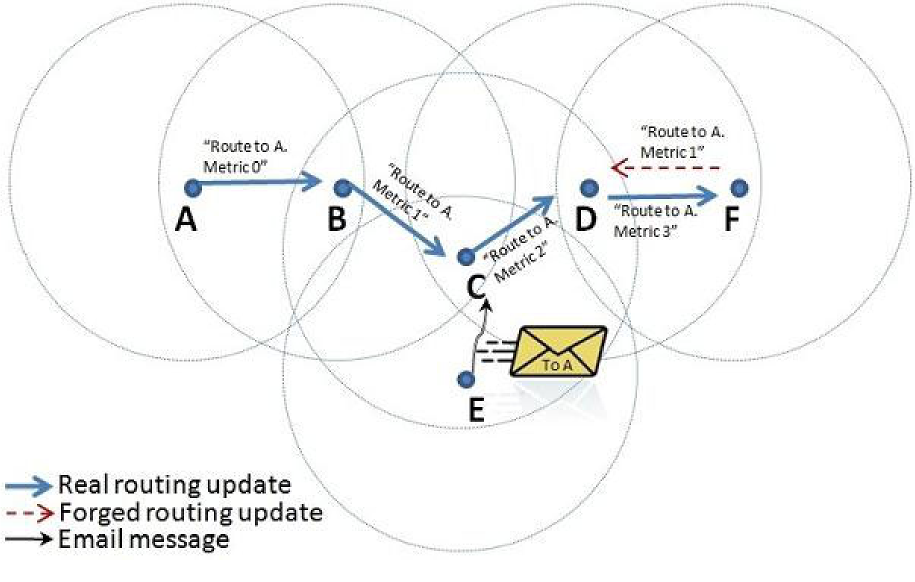

Fig. 1. A typical scenario for a route truncation attack in wireless networks

The circles represent coverages of the nodes located at their centers and the arrows represent broadcasts. Although the arrows point in particular directions, every node within a circle receives that message. Two non-overlapping nodes communicate by using the intermediate nodes as forwarders. $X \rightarrow: m$ indicates that m is broadcast by X. $\mathsf{S}_X(m)$ denotes a signature of X on m (for brevity, assume that signatures provide message recovery). $X \Leftarrow Y$ denotes the string

"*There is a metric 1 path from* Y *to* X" and $X \Leftarrow$ denotes the string "*There is a metric 0 path to* X". Each node acts as an autonomous system (AS).

BGP updates: (control plane) Refer to Fig. 1. The following updates are sent for routes to A. Each signature (except the first) implies a hop of metric 1.

1. $A \rightarrow$: $\mathsf{S}_A(A \Leftarrow)$
2. $B \rightarrow$: $\mathsf{S}_A(A \Leftarrow)$, $\mathsf{S}_B(A \Leftarrow B)$
3. $C \rightarrow$: $\mathsf{S}_A(A \Leftarrow)$, $\mathsf{S}_B(A \Leftarrow B)$, $\mathsf{S}_C(B \Leftarrow C)$
4. $D \rightarrow$: $\mathsf{S}_A(A \Leftarrow)$, $\mathsf{S}_B(A \Leftarrow B)$, $\mathsf{S}_C(B \Leftarrow C)$, $\mathsf{S}_D(C \Leftarrow D)$
   $E \rightarrow$: $\mathsf{S}_A(A \Leftarrow)$, $\mathsf{S}_B(A \Leftarrow B)$, $\mathsf{S}_C(B \Leftarrow C)$, $\mathsf{S}_E(C \Leftarrow E)$
5. $F \rightarrow$: $\mathsf{S}_A(A \Leftarrow)$, $\mathsf{S}_B(A \Leftarrow B)$, $\mathsf{S}_C(B \Leftarrow C)$, $\mathsf{S}_D(C \Leftarrow D)$, $\mathsf{S}_F(D \Leftarrow F)$

From this, each node updates its routing table with a tuple (*destination*, *next-hop*, *metric*) [1]. For instance C's table would contain the entry $(A, B, 2)$.

Forwarding: (data plane) Forwarding uses the following rule: if a data packet arrives from a node that is on the route to the destination, then it is dropped, otherwise it is forwarded (i.e., broadcast). In our example, node E transmits a data packet destined to A. On receiving this, node C would activate and broadcast this packet. Both B and D would receive this broadcast, but only B will activate to forward it further. Finally, B's broadcast is received by A.

Route truncation attack: Assume that F is an attacker who wishes to intercept the above data packet. First note that D's table is set to discard data packets received from C and addressed to A. Thus, such packets would never be received by F. To launch its attack, F claims a shorter route to A than C by replacing its routing update of Step 5 with the following:

$$F \rightarrow: \mathsf{S}_A(A \Leftarrow),\ \mathsf{S}_F(A \Leftarrow F)$$

On receiving this update, D would believe that the route to A via F is shorter than that via C. Consequently, every data packet sent by E and addressed to A will be received by F. This general attack is called *route truncation.*

One way to circumvent this attack is to use Secure-BGP (S-BGP) [6,9,7].

S-BGP updates: S-BGP updates are recipient specific, and therefore every node is required to be aware of its immediate neighbors. Nodes authorize their neighbors using Route Attestations (which are essentially the signatures in the updates shown below). Let X and Y denote nodes within E's and F's coverages respectively, but not covered by others. The S-BGP updates are as follows.

1. $A \rightarrow$: $\mathsf{S}_A(A \Leftarrow B)$
2. $B \rightarrow$: $\mathsf{S}_A(A \Leftarrow B)$, $\mathsf{S}_B(B \Leftarrow C)$
3. $C \rightarrow$: $\mathsf{S}_A(A \Leftarrow B)$, $\mathsf{S}_B(B \Leftarrow C)$, $\mathsf{S}_C(C \Leftarrow D)$
   $C \rightarrow$: $\mathsf{S}_A(A \Leftarrow B)$, $\mathsf{S}_B(B \Leftarrow C)$, $\mathsf{S}_C(C \Leftarrow E)$
4. $D \rightarrow$: $\mathsf{S}_A(A \Leftarrow B)$, $\mathsf{S}_B(B \Leftarrow C)$, $\mathsf{S}_C(C \Leftarrow D)$, $\mathsf{S}_D(D \Leftarrow F)$
   $E \rightarrow$: $\mathsf{S}_A(A \Leftarrow B)$, $\mathsf{S}_B(B \Leftarrow C)$, $\mathsf{S}_C(C \Leftarrow E)$, $\mathsf{S}_E(E \Leftarrow X)$
5. $F \rightarrow$: $\mathsf{S}_A(A \Leftarrow B)$, $\mathsf{S}_B(B \Leftarrow C)$, $\mathsf{S}_C(C \Leftarrow D)$, $\mathsf{S}_D(D \Leftarrow F)$, $\mathsf{S}_F(F \Leftarrow Y)$

Drawbacks of S-BGP: Although secure against route truncation, S-BGP has two drawbacks: (1) Each router must have knowledge of its neighbors, and (2) Router C can no longer broadcast the same update for every neighbor. That is, firstly, every router must discover all its neighbors, and secondly, since updates are peer-specific, therefore route changes would result in a large number of broadcasts by a node with many neighbors. This causes scalability issues.

Stateless Routing: Scalability could be improved if the routing protocol resisted route truncation attacks without requiring nodes to have prior knowledge of neighbors, thereby allowing a single broadcast per update. We call such a protocol *stateless* because nodes need not maintain states of their neighbors.

Related Work: Existing works are based on S-BGP and try to reduce the number and/or processing time of signatures. For instance, aggregate signatures keep the payload to a constant size [10]. *Signature Amortization* [11] coupled with (sequential) aggregate signatures [12] reduce the size of updates and signing time [6]. All the above works, however, assume that information about neighbors is known a priori before updates are sent, and are therefore not stateless.

Our Contribution: We present a stateless S-BGP variant, called Stateless S-BGP (SS-BGP), which is an optimized version of the protocol in [8].

It should be emphasized that we address only the scalability issues arising from using S-BGP in wireless networks, and not the security issues. In particular, any security issues arising from the use of S-BGP will be inherited by SS-BGP.

3 The BGP Routing Protocol

Our protocol is an extension of the basic BGP protocol [1], which we describe in slightly more detail here. In BGP, routing updates propagate in a tree structure rooted at the destination. For $i > 0$, consider any node at level $i+1$ of the tree, and label as $R_1, R_2, \ldots, R_{i+1}$, the nodes in the path from the root to this node (both inclusive). Denote by Sign_i, Verify_i the sign and verify functions of node R_i under an existentially unforgeable signature scheme. R_0 is a constant string used for notational convenience. BGP has two phases: Initialize and Update.

Initialize. Let t_1 be a timestamp. The initiator (R_1) sets $m_1 \leftarrow (R_0, R_1, t_1)$; $sig_1 \leftarrow \mathsf{Sign}_1(m_1)$; and $U_1 \leftarrow (m_1, sig_1)$. It broadcasts U_1.

Update. On receiving update U_i, node R_{i+1} sets $m_{i+1} \leftarrow (R_i, R_{i+1}, t_{i+1})$, where t_{i+1} is a timestamp. The update phase consists of two stages:

1. *Validation:* Parse U_i as $(m_1, sig_1), (m_2, sig_2), \ldots, (m_i, sig_i)$. Then ensure the following:
   (a) For each $j \in [1..i] : m_j$ is of the form (R_{j-1}, R_j, t_j).
   (b) For each $j \in [1..i] : R_{j+1} \notin \{R_1, R_2, \ldots, R_j\}$.
   (c) The route to R_1 given by the ordered tuple $(R_1, R_2, \ldots, R_i)$ is either new or better than an existing route.
   (d) For each $j \in [1..i]$: The difference in timestamps, $t_{j+1} - t_j$ is within a pre-defined threshold.

(e) For each $j \in [1..i]$: $\mathsf{Verify}_j(m_j, sig_j) = \mathsf{VALID}$.
Abort if any of the above checks fail, otherwise, proceed to the next step.

2. *Propagation:* Set $sig_{i+1} \leftarrow \mathsf{Sign}_{i+1}(m_{i+1})$ and $U_{i+1} \leftarrow (U_i, (m_{i+1}, sig_{i+1}))$. Update routing table and broadcast U_{i+1}.

We say that an update is *accepted* if it is successfully validated and propagated.

Correctness: Step 1(a). ensures that the update is of the correct format. If all nodes are honest then the only other steps where validation can fail are

- Step 1(b), when $R_{j+1} \in \{R_1, R_2, \ldots, R_j\}$ for some $j \in [1..i]$. For instance, referring to Fig. 1, when C's routing update reaches B.
- Step 1(c), when the existing routing table has a better route.

In either case, the protocol behaves correctly and implements BGP [1].

Security: The security is captured in the Validation stage as follows:

- Step 1(d) prevents replay attacks.[1]
- Step 1(e) ensures each node $j \in [1..i]$ accepted this update.

Weakness: The weakness in this protocol is that although Step 1(e). ensures that each node $j \in [1..i]$ accepted this update, it does not ensure that these are the only nodes that accepted this update. Specifically, it does not prevent the route truncation attack discussed in Section 2. For instance, when R_{i+1} receives this update, it cannot ensure that R_i received this update from R_{i-1}, since R_i could have truncated several intermediate entries.

Statelessness: The primary advantage is that of statelessness - any node R_i never needs knowledge of R_{i+1}. Therefore, there is no need to establish knowledge of immediate neighbors in order to use the protocol. Secondly, since update R_i is independent of R_{i+1}, the same update can be used by several nodes at level $i+1$.

4 Enhanced Chain Signatures

Notation: We first give some notation to deal with ordered elements.

1. A *sequence* is similar to a set except that the order of its elements is important. Elements of a sequence are written in order from left to right, and enclosed within the symbols $\langle, \rangle$. For instance, $\langle y_1, y_2, y_3 \rangle$ is a sequence. The symbol θ denotes the empty sequence with zero elements.
2. Let $\ell_a = \langle y_1, y_2, \ldots, y_k \rangle$ be some non-empty sequence. For any other sequence ℓ_b, we say that $\ell_b \prec \ell_a$ if and only if $\ell_b = \langle y_1, y_2, \ldots, y_i \rangle$ and $0 \leq i \leq k$. We say that two sequences $\{\ell_a, \ell_b\}$ **overlap** if there exists a non-empty sequence ℓ such that $\ell \prec \ell_a$ and $\ell \prec \ell_b$. For instance, $\{\langle x, y \rangle, \langle x \rangle\}$ overlap, while $\{\langle x, y \rangle, \langle y \rangle\}$ do not.

[1] Note that neither BGP nor S-BGP uses timestamps. In S-BGP, an expiration time carried in RAs is used to prevent replay attacks [7]. The timestamps here provide greater flexibility and may be used in conjunction with an expiration time in m_i.

3. For any two sequences ℓ_a, ℓ_b, the symbol $\ell_a \cup \ell_b$ denotes the **set** of elements that belong to at least one of $\{\ell_a, \ell_b\}$. Similarly $\ell_a \cap \ell_b$ denotes the **set** of elements that belong to both ℓ_a and ℓ_b. We denote by $\ell_a \odot \ell_b$ to be the **set** of elements from the largest sequence ℓ such that $\ell \prec \ell_a$ and $\ell \prec \ell_b$. For example, $\langle x, y, z \rangle \odot \langle x, z, y \rangle = \{x\}$ and $\langle x, y, z \rangle \cap \langle x, z, y \rangle = \{x, y, z\}$
4. Any Sequence $\langle \langle y_1, y_2, \ldots y_i \rangle, y_{i+1} \rangle$ is equivalent to $\langle y_1, y_2, \ldots y_{i+1} \rangle$.

We are now ready to describe Enhanced Chain Signatures (ECS), on which our routing protocol is based. As mentioned earlier, ECS are an extension of Chain Signatures (CS) proposed in [8]. The difference being that, while in CS, every signer signs the same message, in ECS, signers may sign different messages.

Algorithms: ECS are defined by 3 algorithms: **ECS-**(**KeyGen**, **Sign**, **Verify**).

ECS-KeyGen(τ) takes input $\tau \in \mathbb{N}$. It outputs a private-public key pair (x, y).

ECS-Verify(ℓ, σ) takes input $\ell = \langle (m_1, y_1), (m_2, y_2), \ldots \rangle$, a finite sequence of (message, public key) pairs, and a string σ.

1. If $\ell = \theta$ and $\sigma = 1$, the algorithm outputs VALID.
2. If $\ell = \theta$ and $\sigma \neq 1$, the algorithm outputs INVALID.
3. If any public key y_i repeats in ℓ, the algorithm outputs INVALID.
4. If this step is executed, the algorithm invokes a deterministic poly-time procedure after which it outputs either VALID or INVALID.

ECS-Sign$(x_i, y_i, m_i, \ell_j, \sigma_j)$ takes five inputs, which can be grouped into three parts: (1) a (private key, public key, message) tuple (x_i, y_i, m_i), (2) a sequence $\ell_j = \langle (m_1, y_1), (m_2, y_2), \ldots, (m_j, y_j) \rangle$ of j (message, public key) pairs for $j \geq 0$, and (3) a string σ_j (a purported ECS signature on ℓ_j).

1. If $y_i \in \{y_1, y_2, \ldots, y_j\}$, the algorithm outputs ERROR.
2. If **ECS-Verify**$(\ell_j, \sigma_j) =$ INVALID, the algorithm outputs ERROR.
3. If this step is executed, the algorithm computes an ECS signature σ_i on ℓ_i, where $\ell_i = \langle \ell_j, (m_i, y_i) \rangle$. It outputs (ℓ_i, σ_i).[2]

A string σ is an ECS signature on a sequence ℓ if **ECS-Verify**$(\ell, \sigma) =$ VALID.

Correctness: If the input (ℓ, σ) to **ECS-Verify** is the output of **ECS-Sign** then **ECS-Verify** algorithm must output VALID.

Difference with a CS scheme: If for every sequence $\langle (m_1, y_1), (m_2, y_2), \ldots \rangle$ to be signed, holds $\forall i : m_i = m_1$, then the ECS scheme is also a CS scheme.

Security Model: As in [8], we define the security of ECS under *adaptive known key and chosen message attack.* We define two variants using a parameter $\omega \in \{1, 2\}$ such that setting $\omega = 1$ results in the weaker variant.

Game ECS-UNF$_\omega(\tau)$

[2] Observe that in addition to the ECS signature σ_i on ℓ_i, we require **ECS-Sign** to also output the sequence ℓ_i. This is purely for notational convenience.

Setup. The challenger $\mathcal{C}$ gives the security parameter τ to the adversary $\mathcal{A}$, who then selects a game parameter n (the number of public keys) and an n bit string *extr*. Let $extr[i]$ denote the i^{th} bit of *extr*.

On receiving $(n, extr)$, $\mathcal{C}$ generates $(x_i, y_i) \overset{R}{\leftarrow}$ **ECS-KeyGen**(τ) for $i \in [1..n]$ and gives the set $Y = \{y_i\}_{1 \leq i \leq n}$ of n public keys along with set $X = \{x_i | extr[i] = 1\}_{1 \leq i \leq n}$ of extracted private keys to $\mathcal{A}$.
In the following, we denote by L the set of all non-empty sequences of pairs $(m, y) \in \{0,1\}^* \times Y$ such that no y is repeated.

Queries. Working adaptively, $\mathcal{A}$ makes queries as follows:

1. *Extract*(y): If $y \in Y \wedge \omega \neq 1$, $\mathcal{C}$ responds with the private key for y, otherwise it returns $\perp$.
2. *ECS-Sign*(ℓ): If $\ell \in L$, $\mathcal{C}$ responds with a valid ECS signature on ℓ, otherwise it returns $\perp$.

Output. $\mathcal{A}$ outputs a pair $(\ell_A, \sigma_A) \in L \times \{0,1\}^*$.

Result. $\mathcal{A}$ wins if **ECS-Verify**$(\ell_A, \sigma_A) = \mathsf{VALID}$ and ℓ_A is *non-signable* (Def. 1).

Definition 1. ***(Non-signable Sequence)*** *In the game, let $Y_X \subset Y$ and $L_S \subset L$ be the set of inputs to the extract and ECS-sign queries respectively (Note: $\{y_i | extr[i] = 1\}_{1 \leq i \leq n} \subseteq Y_X$). For any $\ell_A \in L$, define the set $L_A = \{\ell_i | \ell_i \in L_S \wedge \{\ell_A, \ell_i\}$ overlap$\}$. We say ℓ_A is* non-signable *if $\ell_A \notin L_S$, and:*

1. $\{(m_i, y_i) | (m_i, y_i) \in \ell_A \wedge y_i \notin Y_X\} \neq \emptyset$.
2. $\forall \ell_i \in L_A : \{(m_j, y_j) | (m_j, y_j) \in (\ell_i \cup \ell_A) \backslash (\ell_i \odot \ell_A) \wedge y_i \notin Y_X\} \neq \emptyset$.

In the following definition, we also consider the use of hash functions.

Definition 2. *The ECS scheme Σ is $(n, \tau, t, q_s, q_e, q_h, \epsilon)$-**UNF**-$\omega$-secure for $\omega \in \{1, 2\}$ if, for game parameters τ and n, there is no adversary $\mathcal{A}$ that runs for time at most t; makes at most (q_s, q_e, q_h) chain-sign, extract and hash queries respectively; and wins Game ECS-UNF$_\omega(\tau)$ with probability at least ϵ.*

Discussion: Roughly speaking, the above security notions imply security under two types of forgeries [8]. We illustrate this with an example. The first forgery (called *ordinary forgery*) occurs when the adversary manages to output a valid ECS signature on a sequence, say $\ell = \langle (m_1, y_1), (m_2, y_2) \rangle$ after making an ECS-sign query on $\ell_1 = \langle (m_1, y_1) \rangle$ but without making an extract query on y_2. This is the type of forgery that all multisignatures schemes (including the ones discussed in Section 2) resist. The second type of forgery in ECS (called *extraction forgery*) occurs when the adversary manages to output a valid ECS signature on $\ell = \langle (m_1, y_1), (m_2, y_2) \rangle$ after making an ECS-sign query on $\ell_3 = \langle (m_1, y_1), (m_2, y_2), (m_3, y_3) \rangle$ but without making an extract query on y_3 and one of $\{y_1, y_2\}$. A scheme secure against an extraction forgery is said to be *truncation resilient.* Note that neither aggregate signatures [10], nor sequential aggregate signatures [12,13] consider extraction forgery.

Construction: A construction of ECS is given in Appendix A.

5 Stateless Secure BGP Using ECS

This protocol fixes the weaknesses of BGP. As in Section 3, let $(R_1, R_2, \ldots)$ be any ordered tuple of nodes that would be affected by a given update, and t_i be the timestamp at which the update originated/arrived at node i. Let (x_i, y_i) be the (private, public) key-pair of node R_i under an ECS scheme. Assume that every node has a distinct public key. The SS-BGP protocol is as follows.

Initialize. Initiator R_1 sets $m_1 \leftarrow t_1$ and $(\ell_1, \sigma_1) \leftarrow$ **ECS-Sign**$(x_1, y_1, m_1, \theta, 1)$. Note that $\ell_1 = \langle (m_1, y_1) \rangle$. Finally it sets $L_1 \leftarrow (m_1, R_1)$; $U_1 \leftarrow (L_1, \sigma_1)$ and broadcasts the update U_1.[3]

Update. On receiving update $U_i = (L_i, \sigma_i)$, node R_{i+1} sets $m_{i+1} \leftarrow t_{i+1}$ and does the following:

1. *Validation:* Parse L_i as $(m_1, R_1), (m_2, R_2), \ldots, (m_i, R_i)$. Then ensure the following:
   (a) For each $j \in [1..i] : m_j$ is of the form t_j.
   (b) For each $j \in [1..i] : R_{j+1} \notin \{R_1, R_2, \ldots, R_j\}$.
   (c) The route to R_1 given by the ordered tuple $(R_1, R_2, \ldots, R_i)$ is either new or better than an existing route.
   (d) For each $j \in [1..i]$: The difference in timestamps, $t_{j+1} - t_j$ is within the pre-defined threshold t.
   (e) Construct the sequence $\ell_i = \langle (m_1, y_1), (m_2, y_2), \ldots, (m_i, y_i) \rangle$ and test if **ECS-Verify**$(\ell_i, \sigma_i) = \mathsf{VALID}$.

   Abort if any of the above checks fail, otherwise, update routing table and proceed to the next step.
2. *Propagation:* Set $(\ell_{i+1}, \sigma_{i+1}) \leftarrow$ **ECS-Sign**$(x_{i+1}, y_{i+1}, m_{i+1}, \ell_i, \sigma_i)$ and $L_{i+1} \leftarrow (L_i, (m_{i+1}, R_{i+1}))$. Note that $\ell_{i+1} = \langle \ell_i, (m_{i+1}, y_{i+1}) \rangle$. Finally, set $U_{i+1} \leftarrow (L_{i+1}, \sigma_{i+1})$ and broadcast U_{i+1}.

Correctness: Referring to the basic BGP protocol of Section 3, the only real difference is in Step 1(e). Assuming that the validation in this step always passes, the above protocol then implements BGP in the presence of honest users.

Security: Consider Step 1(e), where **ECS-Verify** is used. Observe that if the ECS scheme is **UNF**-1-secure then the protocol indeed ensures that each node $j \in [1..i]$ accepted this update. Thus, SS-BGP prevents *false route injection.* Next, we consider two further attacks on BGP-type protocols.

Route Truncation attacks: Let us analyze this using the example of Section 2, Fig. 1. The nodes in the path of the update received by F are (A, B, C, D). In the attack based on BGP, given the update

$$\mathsf{S}_A(A \Leftarrow),\ \mathsf{S}_B(A \Leftarrow B),\ \mathsf{S}_C(B \Leftarrow C),\ \mathsf{S}_D(C \Leftarrow D),$$

attacker F extracts the update $\mathsf{S}_A(A \Leftarrow)$. The corresponding SS-BGP update received by F is $U_D = (L_D, \sigma_D)$, where $L_D = ((t_A, A)(t_B, B)(t_C, C)(t_D, D))$, and the corresponding attack would amount to F extracting the ECS signature

[3] Our basic BGP variant only needs a timestamp. However, additional information may be included in m_i at node i. An example of such information is an expiration time of this update (see Footnote 1) or the GPS coordinates of i.

σ_A on ℓ_A given the ECS signature σ_D on ℓ_D. Under the **UNF**-1 security notion of ECS, this is not possible unless F has extracted the private keys of B, C, D. It can be seen that truncation attacks are not possible in SS-BGP.

Repeating Attacks: In this attack, F acts as a man-in-the-middle and simply repeats the message received from D to masquerade as D and make the route one hop shorter. In the more general case, called wormhole attacks [14,15,16,17,18], the adversary repeats the update at one or more distant regions using an out-of-band channel. As observed in [19], such attacks are unavoidable in BGP-type protocols (including both SS-BGP and S-BGP) if an attacking node can forward messages without modification such that its presence is never detected.[4] Thus, S-BGP does not offer any more protection than SS-BGP under these attacks.

Other attacks based on timestamps are possible on SS-BGP. However, these attacks are also possible in S-BGP, and so we do not consider them here.

Overhead: Assuming that public keys can be uniquely identified by identities, the sequences ℓ_i can be constructed at the receiver's end by knowing the timestamps $(t_1, t_2, \ldots, t_i)$ and identities $(R_1, R_2, \ldots, R_i)$. Consequently, the only overhead in this protocol is that of one single ECS signature σ_i, which is an element of G_1 and is less than 30 bytes using the parameters of [8]. Contrast this with basic BGP or S-BGP, both of which incur an overhead of i signatures.

Storage and Performance: The keys are elements of G_1 and can be stored in ≤ 30 bytes [8]. The benchmarks of [21] indicate the following performance estimates of the above protocol (assuming n nodes in the path):

1. **Update Propagation:** one exponentiation and addition in G_1, and one computation of $\mathcal{H}$ (total $<$ 2ms).
2. **Update Verification:** n pairing computations and multiplications in G_2, and n computations of $\mathcal{H}$ (giving $<$ 1 second for $n = 100$).

To conclude, SS-BGP based on ECS is as secure and as computationally efficient as S-BGP based on the signature schemes of [10,13] without the extra overhead of neighbor discovery, multiple signature computation, multiple broadcasts and multiple signatures in each broadcast.

Multiple Updates Aggregation: In the above description, we assumed that each advertisement U_i contains only one route and is transmitted instantaneously. In the real world, each advertisement contains multiple routes and is sent periodically. Fortunately, the ECS scheme allows *signature aggregation* and *aggregate verification* where a large number of ECS signatures are verified at once [10].

SS-BGP using Identity-Based ECS: SS-BGP using Identity-Based ECS (IBECS) would avoid the overhead of public-key management. However, at this stage a

[4] A possible way to detect this attack in wireless networks would be to encode GPS coordinates in the updates (see Footnote 3). It may then be possible to determine if a message was sent via a repeater by making assumptions on a transmitter's range. Other techniques to detect or prevent wormhole attacks are discussed in [17,18,20,15]

construction of IBECS is not known. A possible construction may arise from the Identity-Based Sequential Aggregate Signatures (IBSAS) of [13].

6 Conclusion

The Border Gateway Protocol (BGP) is a path vector protocol allowing nodes to build routing tables without prior knowledge of neighbors, and requires a single broadcast per node irrespective of the number of neighbors. We call such a protocol *stateless*. However, BGP does not resist route truncation attacks. Consequently, modern implementations use a stateful variant of BGP called Secure-BGP (S-BGP). S-BGP requires forwarding nodes to have prior knowledge of immediate neighbors and necessitates as many broadcasts per node as there are neighbors. Although sufficient for wired networks, this causes scalability problems in mobile ad-hoc wireless networks with low data plane traffic.

We presented a secure stateless variant of BGP for use in wireless networks by extending the work of [8]. The protocol, called Stateless Secure BGP (SS-BGP), is an augmentation of BGP that resists route truncation attacks. The main ingredient of SS-BGP is an authentication primitive called Enhanced Chain Signatures (ECS), which is an enhancement of the Chain Signatures (CS) of [8].

In summary, wireless S-BGP incurs the following overhead: (1) neighbor discovery, (2) multiple signature computation, (3) multiple broadcasts, and (4) multiple signatures in each broadcast. Although the signature schemes of [10,13] are able to address issue (4), they do not address the remaining three. SS-BGP based on ECS is as secure and computationally efficient as S-BGP based on the signature schemes of [10,13] without any of the above overheads. This feature makes SS-BGP particularly attractive for wireless networks.

References

1. Rekhter, Y., Li, T., Hares, S.: RFC 4271: A Border Gateway Protocol 4 (BGP-4) (Janaury 2006)
2. Rexford, J., Wang, J., Xiao, Z., Zhang, Y.: BGP routing stability of popular destinations. In: ACM SIGCOMM IMW (Internet Measurement Workshop) (2002)
3. Estrin, D., Rekhter, Y., Hotz, S.: A Unified Approach to Inter-Domain Routing. RFC 1322 (Informational) (May 1992)
4. Murphy, S.: RFC 4272: BGP security vulnerabilities analysis
5. Mahajan, R., Wetherall, D., Anderson, T.: Understanding BGP misconfiguration. In: SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 3–16. ACM Press, New York (2002)
6. Zhao, M., Smith, S.W., Nicol, D.M.: Aggregated path authentication for efficient BGP security. In: CCS '05: Proceedings of the 12th ACM conference on Computer and communications security, pp. 128–138. ACM Press, New York (2005)
7. Kent, S.: Securing the border gateway protocol. The Internet Protocol Journal 6(3), 2–14 (2003)

8. Saxena, A., Soh, B.: One-way signature chaining: A new paradigm for group cryptosystems. International Journal of Information and Computer Security 2(3), 268–296 (2008)
9. Kent, S.: Securing the border gateway protocol: A status update. In: Seventh IFIP TC-6 TC-11 Conference on Communications and Multimedia Security, pp. 2–3 (2003)
10. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
11. Park, J.M., Chong, E.K.P., Siegel, H.J.: Efficient multicast packet authentication using signature amortization. In: SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy, Washington, DC, USA, p. 227. IEEE Computer Society, Los Alamitos (2002)
12. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004)
13. Boldyreva, A., Gentry, C., O'Neill, A., Yum, D.H.: Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In: CCS '07: Proceedings of the 14th ACM conference on Computer and communications security, pp. 276–285. ACM, New York (2007)
14. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: Attacks and countermeasures. In: First IEEE International Workshop on Sensor Network Protocols and Applications, pp. 113–127 (2002)
15. Poovendran, R., Lazos, L.: A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks. Wirel. Netw. 13(1), 27–59 (2007)
16. Wang, W., Bhargava, B.: Visualization of wormholes in sensor networks. In: WiSe '04: Proceedings of the 3rd ACM workshop on Wireless security, pp. 51–60. ACM, New York (2004)
17. Maheshwari, R., Gao, J., Das, S.R.: Detecting wormhole attacks in wireless networks using connectivity information. In: INFOCOM, pp. 107–115. IEEE, Los Alamitos (2007)
18. Hu, Y.-C., Perrig, A., Johnson, D.B.: Wormhole attacks in wireless networks. IEEE Journal on Selected Areas in Communications 24(2), 370–380 (2006)
19. Torgerson, M., Leeuwen, B.V.: Routing data authentication in wireless networks. Technical Report SAND2001-3119, Sandia National Labs., Albuquerque, NM (US); Sandia National Labs., Livermore, CA, US (2001)
20. Khabbazian, M., Mercier, H., Bhargava, V.K.: Severity analysis and countermeasure for the wormhole attack in wireless ad hoc networks. Trans. Wireless. Comm. 8(2), 736–745 (2009)
21. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. Cryptology ePrint Archive, Report 2005/028 (2005)
22. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. J. Cryptology 17(4), 297–319 (2004)
23. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. SIAM J. Comput. 32(3), 586–615 (2003)

A Construction of ECS

The following construction of ECS is adapted from [8].

1. **Bilinear Maps:** Let G_1 and G_2 be two cyclic multiplicative groups both of prime order q such that computing discrete logarithms in G_1 and G_2 is intractable. A bilinear pairing is a map $\hat{e} : G_1 \times G_1 \mapsto G_2$ that satisfies the following properties [22,23,10].
   - *Bilinearity*: $\hat{e}(a^x, b^y) = \hat{e}(a, b)^{xy}$ $\forall a, b \in G_1$ and $x, y \in \mathbb{Z}_q$.
   - *Non-degeneracy*: If g is a generator of G_1 then $\hat{e}(g, g)$ is a generator of G_2.
   - *Computability*: The map $\hat{e}$ is efficiently computable.

   In a practical implementation, G_1 is a subgroup of the additive group of points on the elliptic curve and G_2 is the multiplicative subgroup of a finite field [22,23]. The security of our protocol depends on the hardness of the Computational Diffie-Hellman (CDH) problem in G_1 defined as follows: given $g, g^x, g^y \in G_1$ for $g \notin \{0_{G_1}, 1_{G_1}\}$, and unknowns x, y, compute $g^{xy} \in G_1$ [10].
2. **Common Parameters:** Let $\hat{e} : G_1 \times G_1 \mapsto G_2$ be a bilinear map over cyclic multiplicative groups (G_1, G_2) of prime order q, and g be any generator of G_1. The prime q is chosen so that the CDH problem in G_1 is requires $\approx 2^\tau$ operations for some security parameter τ. See [10] for details. Let $\mathcal{H} : \{0,1\}^* \mapsto G_1$ be a hash function. In the rest of this section, these parameters will be considered common and public.
3. **The Algorithms:** The following are the three algorithms in the scheme.

   ECS-KeyGen. Each user j generates $x_j \overset{R}{\leftarrow} \mathbb{Z}_q$. The private key of j is x_j. The corresponding public key is $Y_j = g^{x_j} \in G_1$.

   ECS-Sign. Let $\ell_i = \langle (m_1, Y_1), (m_2, Y_2), \ldots, (m_i, Y_i) \rangle$ be some sequence of i distinct (message, public key) pairs. A valid ECS signature on sequence ℓ_i is an element $\sigma_i \in G_1$, where:

   $$\sigma_i = \prod_{j=1}^{i} \mathcal{H}(\langle (m_1, Y_1), (m_2, Y_2), \ldots, (m_j, Y_j) \rangle)^{x_j},$$

   $\ell_0 = \theta$ and $\sigma_0 = 1_{G_1}$. Given $(\ell_{i-1}, \sigma_{i-1})$, the ECS signature σ_i on ℓ_i is computed by user $i \in \{1, 2, \ldots\}$ as

   $$\sigma_i \leftarrow \sigma_{i-1} \cdot \mathcal{H}(\ell_i)^{x_i}.$$

   ECS-Verify(ℓ_i, σ_i). Let $\ell_i = \langle (m_1, Y_1), (m_2, Y_2), \ldots, (m_i, Y_i) \rangle$. To verify the signature σ_i, check that all Y's are distinct and the following holds:

   $$\hat{e}(\sigma_i, g) \overset{?}{=} \prod_{j=1}^{i} \hat{e}(\mathcal{H}(\langle (m_1, Y_1), (m_2, Y_2), \ldots, (m_j, Y_j) \rangle), Y_j).$$

4. *Security:* The CS scheme of [8] is shown to be **UNF**-1-secure. We observe that the same proof of [8] can be made to work for the above ECS construction with an appropriate modification to the simulator. Hence, the above ECS construction is also **UNF**-1-secure in the random oracle model under the computational Diffie-Hellman (CDH) assumption in bilinear maps. Due to lack of space, the proof is deferred to the full version of this paper.

   Whether the above ECS scheme is also **UNF**-2-secure is an open question.