Reduction-based Security Analysis of Internet Routing Protocols

Chen Chen*, Limin Jia†, Boon Thau Loo*, Wenchao Zhou‡
* University of Pennsylvania, Philadelphia, PA 19104, USA Email: chenche, boonloo@seas.upenn.edu
† Carnegie Mellon University, Pittsburgh, PA 15213, USA Email: liminjia@cmu.edu
‡ Georgetown University, Washington, DC 20057, USA Email: wzhou@cs.georgetown.edu

***Abstract*—In recent years, there have been strong interests in the networking community in designing new Internet architectures that provide strong security guarantees. However, none of these proposals back their security claims by formal analysis. In this paper, we use a reduction-based approach to prove the route authenticity property in secure routing protocols. These properties require routes announced by honest nodes in the network not to be tampered with by the adversary. We focus on protocols that rely on layered signatures to provide security: each route announcement is associated with a list of signatures attesting the authenticity of its subpaths. Our approach combines manual proofs with automated analysis. We define several reduction steps to reduce proving route authenticity properties to simple conditions that can be automatically checked by the Proverif tool. We show that our analysis is correct with respect to the trace semantics of the routing protocols.**

I. Introduction

In recent years, there have been strong interests in the networking community in designing new Internet architectures to address pressing security concerns. These range from security extensions to Internet routing [1], [2], [3], to "clean-slate" redesigns [4], [5]. One of the limitations of these proposals is that these new designs lack formal security proofs – these protocols are evaluated primarily via experimental evaluations and their security guarantees shown via informal reasoning.

This paper aims to develop techniques for proving security guarantees of the secure routing protocols. As a step towards eventually analyzing new Internet architectures, this paper focuses primarily on secure extensions to the current Internet architecture. The Internet runs a routing protocol called the *Border Gateway Protocol* (BGP), where routers are grouped into various Autonomous Systems (*AS*) administrated by Internet Server Providers (*ISP*). Individual ASes exchange route advertisements with neighboring ASes using the *path-vector* protocol. Each originating AS first sends a route advertisement (containing a single AS number) for the destination addresses that it owns. Whenever an AS receives a route advertisement, it will add itself to the AS *path*, and then advertise the best route to its neighbors based on routing policies.

Since these route advertisements are not authenticated, ASes can lie and advertise non-existent routes, or claim to own destination addresses that they do not. These faults may be a consequence of harmless misconfigurations, or malicious activities, e.g. traffic hijacking or violations of business agreements. They may lead to long periods of interruption of the Internet. These faults violate the *route authenticity* property, which requires that routes announced by honest ASes in the network not to be tampered with by the adversary.

978-1-4673-2447-2/12/$31.00

There have been a variety of proposed mechanisms [6] that aim to provide or improve the route authenticity of the Internet routing. For example, Secure BGP (S-BGP) [1] and pretty secure BGP [2] use cryptographic functions to sign routing information to prevent malicious routers from altering the routing information. Many such proposals rely on some form of layered signatures to provide security: each route announcement is associated with a list of signatures attesting the authenticity of its subpaths [7], [2], [4], [5]. Intuitively, attackers do not have the private keys of honest nodes, and therefore cannot forge the signatures that were created by the honest nodes. Consequently, route announcements sent out by the honest nodes cannot be tampered with by the attacker. We formalize this intuition and prove route authenticity properties on variants of S-BGP, a comprehensive routing security solution for BGP that uses layered signatures.

We define several *reduction* steps to reduce the route authenticity proofs to checking conditions that are simple enough to be either discharged manually or by an automated tool Proverif [8]. The reduction steps are generic to the class of routing protocols that we study. We show that our analysis is correct with respect to the trace semantics of the routing protocols. Our analysis either proves that a given property is met, or provides evidence of an attack. This provides a basis for comparing different protocols based on their security guarantees, costs of deployment and performance.

This paper makes the following contributions:

- We model routing protocols as transition systems, and formally define route authenticity properties in a first-order temporal logic (Section II).
- We define reduction steps that reduce the proof of route authenticity to conditions that can be either automatically checked or manually discharged. We have formally proven the correctness of our reduction steps (Section III).
- We use Proverif, an automated tool, to check the most complicated conditions, and present case study results (Section IV).

Detailed definitions and proofs can be found in the companion technical report [9].

Malicious Node	M	::=	(u, I, U)
Honest Node	H	::=	$(u, I, RtTb, Q)$
Signature	σ	::=	$\mathsf{sign}((d,p), \mathsf{sk}(u))$
Protocol	$\mathcal{P}$	::=	$(f_{orig}, f_{upd}, f_{ck}, f_a, \delta)$
Transition Function	δ	$\in$	*States* $\times$ *Actions* $\rightarrow$ *States*
Routing state	S	::=	$\emptyset \mid S, M \mid S, H$
Update	r	::=	(p, u_r, d, Σ)

Fig. 1. Constructs for Modeling Routing Protocols

II. Specifications of Secure Variants of BGP

We explore two variants of the S-BGP protocol, each with different security guarantees. We abstractly model the protocols as transition systems, which encode the underlying path-vector protocol used in BGP. Details of BGP's import and export policies are omitted from the model, since they are not directly relevant to our security analysis. We formally define route authenticity properties, which serves as a basis for our security analysis of these two protocol variants in Section III.

A. Formal Model for Routing Protocols

Syntax. We use a graph $G = (V, E)$ to represent the network topology, where V is the set of nodes (ASes) and E is the set of links. G_u represents the sub-graph of G that contains all the nodes and links directly connected to u. A node is either honest, which runs the prescribed routing protocol; or malicious, modeled using the Dolev-Yao attacker model: attackers can deconstruct terms, construct terms from known ones, including generating signatures using the private keys it knows. We write $\mathcal{A}$ to denote the set of malicious nodes. We assume that the topology (G) and the set of malicious nodes ($\mathcal{A}$) do not change during the execution of the protocol.

The syntactic constructs for modeling the routing protocols are summarized in Figure 1. M denotes the state of a malicious node, where u is the unique ID (AS number) for the node, I is its initial knowledge and U is the set of route updates it has learned so far. The initial knowledge I includes the public and private keys that the attacker knows, and other relevant information it needs to know to run the routing protocol.

H denotes the state of an honest node. In addition to a node ID, and initial knowledge, an honest node uses a routing table $RtTb$ to store known routes to different destinations and a queue Q to store the set of update messages to be processed.

We write H_u (or M_u) to denote the local state of an honest (or a malicious) node of ID u. A global state in the routing protocol S consists of all the nodes in the network.

A path p is a list of nodes $[u_1, \cdots u_n]$. σ denotes digital signatures. The signature of a pair of a path p and a destination prefix d signed by u's private key is written $\mathsf{sign}((d,p), \mathsf{sk}(u))$. A list of signatures $([\sigma_1, \cdots, \sigma_n])$ is written Σ. r denotes a route update message, where p is the path, d is the destination, u_r is the recipient node of this update, and Σ is a set of signatures associated with this update. The secure BGP protocols are parameterized over the following functions:

- $f_{orig}(d, u_s, u_r)$ computes a route update that originates at node u_s with the destination prefix d for node u_r.
- $f_{upd}(r, u_s, u_r)$ computes a new route update from node u_s to node u_r based on a route r that u_s has received.
- $f_{ck}(r)$ checks the validity of all the signatures in r. It returns true if r has the correct format and all the signatures are valid.
- $f_a(G_u, r, u)$ is a check on r by node u. f_a uses f_{ck} as a sub-routine, and additionally checks that r is sent from a direct neighbor. We assume that each node u knows its direct neighborhood G_u by using a separate neighbor discovery process which we assume is correct.

We say a route update is valid when $f_{ck}(r) = \mathsf{true}$. Actions are denoted α. The action relevant for our security analysis is the send action $sendA(u, v, r)$, which denotes node u sends node v a route update r. The transition function δ maps a pair of a state and an action to another state ($\delta(S, a) = S'$). A trace $\mathcal{T}$ is a sequence of state transitions: $S_0 \xrightarrow{\alpha_1} S_1 \cdots \xrightarrow{\alpha_n} S_n$. We write $\mathcal{R}[(\mathcal{A}, G, \mathcal{P}, S_0)]$ to denote the set of traces of executing the protocol $\mathcal{P}$ on a topology G given the set of malicious nodes $\mathcal{A}$, and an initial state S_0.

We write $\mathcal{K}(M_u)$ to denote all the knowledge that a malicious u can derive based on the standard Dolev-Yao attacker model. $\mathcal{K}(M_u)$ can be defined straightforwardly by a set of inference rules.

Transition function. δ allows three transitions: (1) an honest node originates a route r and r is inserted into the queue of the destination node; (2) an honest node u processes a route announcement r, and either discards it, or generates a new update r_{new}, in which case, u's routing table is updated, and the queue of the intended recipient of r_{new} is also updated; and (3) a malicious node sends out an announcement. For each transition, we additionally allow every malicious node to know the route announcement. The formal definitions of δ can be found in the companion technical report [9].

Two S-BGP variants. Both variants associate a route announcement r with a list of signatures: one for each subpath in r. They differ in the signature format. The signature in Protocol 1 does not contain the receiving node of the route. We define the auxiliary functions for Protocol 2 below. We use $\mathsf{verify}(\sigma, t, \mathsf{pk}(u))$ to represent the signature checking function that returns true if $\sigma = \mathsf{sign}(t, \mathsf{sk}(u))$, and we write $l_1 @ l_2$ to denote the concatenation of two lists l_1 and l_2.

$f_{orig}(d, u_s, u_r) = ([u_s, u_r], u_r, d, [\mathsf{sign}((d, [u_s]), \mathsf{sk}(u_s))])$
$f_{upd}((p, d, u_s, \Sigma), u_s, u_r)$
$\quad = (p@[u_s], d, u_r, \Sigma@[\mathsf{sign}((d, p@[u_s, u_r]), \mathsf{sk}(u_s))])$
$f_{ck}([u], u_r, d, [\sigma]) = \mathsf{true}$ iff $\mathsf{verify}(\sigma, (d, [u, u_r]), \mathsf{pk}(u)) = \mathsf{true}$
$f_{ck}(p@[u], u_r, d, \Sigma@[\sigma]) = \mathsf{true}$ iff $f_{ck}(p, u, d, \Sigma) = \mathsf{true}$
and $\mathsf{verify}(\sigma, (d, p@[u, u_r]), \mathsf{pk}(u)) = \mathsf{true}$
$f_a(r, v) = f_{ck}(r)$ and $uv \in E$ where $r = (p@[u], d, v, \Sigma)$

B. A Logic for Expressing Route Authenticity Properties

We present the syntax and semantics of a first-order logic for specifying the route authenticity properties.

Syntax. Figure 2 summarizes the syntax of formulas. Predicates, written P, include $\mathsf{link}(u_1, u_2)$ representing links in the topology, $\mathsf{honest}(u)$ stating u is honest, and several predicates on the send action. Route-dependent predicates, written $P_R(r, i)$, are parameterized over a route update r and

Predicates	P	::=	$\mathsf{link}(u_1,u_2) \mid \mathsf{honest}(u) \mid \mathsf{send}(u,r) \mid \mathsf{sendTo}(u_1,u_2,r) \mid \mathsf{sent}(u,r) \mid \mathsf{sentTo}(u_1,u_2,r)$
Route-dependent Pred	$P_R(r,i)$	::=	$\mathsf{linkPrev}(r,i) \mid \mathsf{linkNext}(r,i) \mid \mathsf{honestR}(i) \mid \mathsf{sentR}(r,i) \mid \mathsf{sentToR}(r,i)$
Route-dependent Form	$F_R(r,i)$	::=	$P_R \mid \neg F_R(r,i) \mid F_R(r,i) \wedge F_R(r,i) \mid F_R(r,i) \vee F_R(r,i)$
Universal Form	$F_\forall(r)$	::=	$\forall i.F_R(r,i)$
Formula	φ	::=	$P \mid F_\forall \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mid \varphi \supset \varphi \mid \neg\varphi \mid \forall x.\varphi \mid \exists x.\varphi$

Fig. 2. Syntax of Formulas

a natural number i indexing into a position in the path in r. A node can be uniquely determined by r and i. For instance if $r = ([u_1 \cdots u_n], u_r, d, \Sigma)$, then the i^{th} node of the path in r is u_i. Instead of relying on recursively-defined predicates, universally quantified route-dependent formulas can concisely specify properties that are true on every subpath in a route update. The route authenticity properties demonstrate how these route-dependent formulas are used.

Formulas φ are the standard first-order logic formulas that make use of the predicates and route-dependent formulas. To express trace properties, we use one temporal connective $\boxminus$. Formula $\boxminus\varphi$ means that φ is true in every state on the trace before the current state.

Semantics. We write $\mathcal{A}, G, \mathcal{T} \vDash \varphi$ to mean that φ holds in the last state of $\mathcal{T}$, given the topology G and the set of malicious nodes $\mathcal{A}$. Below is a summary of the interesting rules.

$\mathcal{A}, G, \mathcal{T} \vDash \mathsf{send}(u,r)$ iff u is an honest node implies $\mathcal{T} = \mathcal{T}' \xrightarrow{\alpha} S$ and $\alpha = sendA(u,v,r)$
$\mathcal{A}, G, \mathcal{T} \vDash \mathsf{linkPrev}(r,i)$ iff $i = 1$ or $(\llbracket i-1 \rrbracket_r, \llbracket i \rrbracket_r) \in E$
$\mathcal{A}, G, \mathcal{T} \vDash \mathsf{sentToR}(r,i)$ iff $r = (p, u_r, d, \Sigma), u = \llbracket i \rrbracket_r$, u is an honest node implies there exists an action $sendA(u,v,r') \in \mathcal{T}, v = \llbracket i+1 \rrbracket_r$ and $r' = (p', v, d, \Sigma')$, $|p'| = i$ and p' is a subpath of p
$\mathcal{A}, G, \mathcal{T} \vDash \forall i, F_R(r,i)$ iff $\forall i \in [1, |p|], \mathcal{A}, G, \mathcal{T} \vDash F_R(r,i)$, where $r = (p, u_r, d, \Sigma)$

Predicate $\mathsf{send}(u,r)$ is true if the action leading to the current state is node u sending a route update r. When u is a malicious node, $\mathsf{send}(u,r)$ is vicariously true, since there is little meaning to assert what a malicious node has sent out. Predicate sent and sentTo only require the send action to exist on the trace, as opposed to send and sendTo that require the send action to be the last action on the trace.

The rules for route-dependent predicates use $\llbracket i \rrbracket_r$ to denote the i^{th} node in the path of the update r. When n is the length of the route, the $(n+1)^{th}$ node is the recipient node of this update: $\llbracket n+1 \rrbracket_{(p,u_r,d,\Sigma)} = u_r$.

Predicate $\mathsf{linkPrev}(r,i)$ is true when there is a link between the $(i-1)^{th}$ (the previous node) and i^{th} node in the path in r. Predicate $\mathsf{linkNext}(r,i)$ is true when there is a link between the i^{th} and $(i+1)^{th}$ node (the next node). The semantics of sentR and $\mathsf{sentToR}$ are similarly defined by using i to index nodes and subpaths in r. A universally quantified route-dependent formula is true if for all the nodes i in the path of r, $F_R(r,i)$ is true.

Route authenticity property. The route authenticity properties $\mathsf{RouteAuth}$ are of the form $\boxminus\varphi_I$, where φ_I requires route announcements r from an honest node to satisfy certain properties ($F_\forall(r)$):

$$\varphi_I \quad = \quad \forall u \forall r, \mathsf{honest}(u) \wedge \mathsf{send}(u,r) \supset F_\forall(r)$$

We only care about routes sent out by honest nodes since a malicious nodes can send out any message. We call φ_I an *invariant* as it has to hold on all states of the trace. The invariant φ_I depends on the concrete definitions of $F_\forall(r)$. For instance, $\forall i, \mathsf{linkPrev}(r,i) \wedge \mathsf{linkNext}(r,i)$ requires that all the links in a route update exist in the network topology.

III. Security Analysis of Routing Protocols

There are two main ideas in our approach to verifying route authenticity. First, the property of a route sent out by an honest node u depends on the property of the routes u receives. Therefore, the proof needs to induct on the length of the execution traces to avoid circular dependencies. One main reduction step is to apply Rely-Guarantee reasoning principles, and break the induction into the base case (invariants holds initially) and the inductive case (transitions maintain the invariants). Second, both the protocol code and the attacker are analyzed, so that from a failed proof, an attack trace can be easily generated. Leveraging the security properties of the layered signatures, we are able to finitely enumerate all possible scenarios of an attacker's attempts to generate an attack route update.

Section III-A provides an overview of the reduction steps. We focus on explaining how to check whether malicious nodes can generate attack route updates in Section III-B. In Section III-C, we discuss how the conditions resulting from the reduction steps are verified, and how attack scenarios are generated. The soundness proofs can be found in the companion technical report [9].

A. Reduction steps

We summarize our reduction steps in Figure 3. These reduction steps form a proof tree proving that route authenticity property holds on all execution traces of a protocol (specified at the root of the tree), if all the leaf conditions can be verified. Each reduction step is represented as a reversed derivation (from conclusion to premises) in the tree.

First, we narrow down the traces to consider to those where attackers send and store route updates that pass the check $f_{ck}(r)$. This is sound because ill-formed updates will be discarded by the honest nodes right away.

Second, proving the φ_I is reduced to establishing a stronger invariant φ_I^S (defined below). Formula φ_I^S specifies properties of the internal states: all the route updates r stored in the routing table of an honest node must satisfy the required property $F_\forall(r)$, and that any route update in the queue of an honest node must also satisfy $F_\forall(r)$ if it passes the validity check. Further, for malicious nodes, any route update that a malicious node receives must satisfy $F_\forall(r)$ as well.

$$\begin{aligned}\varphi_I^S = \forall u, &(\mathsf{honest}(u) \supset \\ &((\forall r \in RtTb_u, F_\forall(r)) \wedge (\forall r \in Q_u, f_a(r,u) \supset F_\forall(r))) \\ &\wedge (\mathsf{malicious}(u) \supset (\forall r \in U_u, F_\forall(r))\end{aligned}$$

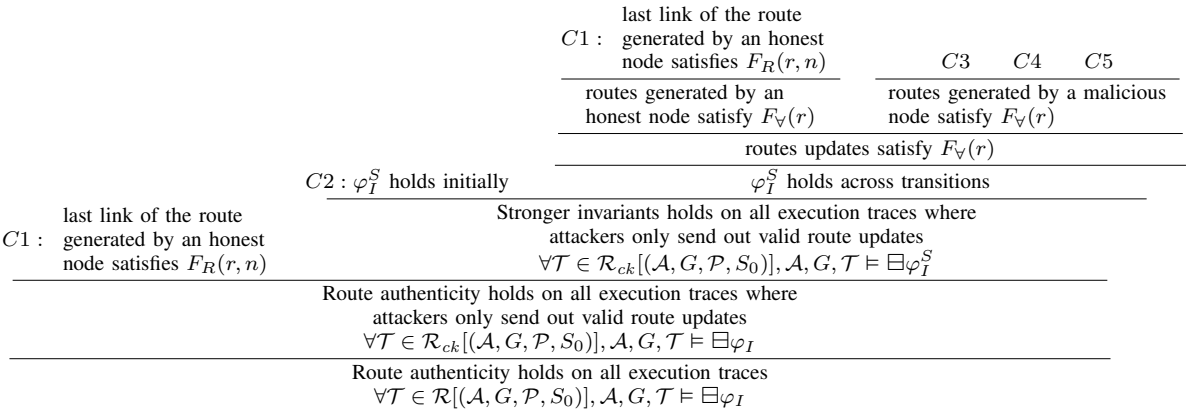

Fig. 3. Reduction Steps

The invariant φ_I^S and that the last link in the new route update generated by an honest node satisfies the property $F_R(r,n)$ imply path authenticity.

Third, we reduce proving φ_I^S is an invariant to: φ_I^S holds initially (C2) and φ_I^S is maintained (an inductive step).

The predicates used in specifying properties of routes talk about past events, and therefore, have the property that if they are true on a trace, then they are true on any extension of the trace. This allows us to check the invariant only on parts of the new state that differ from the preceding state. Therefore, in the fourth step, for the inductive condition, we only need to check that the newly generated updates r satisfies $F_\forall(r)$.

Next, we split the cases and consider honest nodes and malicious nodes separately. An honest node only updates its routing table with a route update directly taken from the queue and has passed the validity check $f_a(r)$. φ_I^S holds at the current state ensures that the new route update in the routing table indeed satisfies $F_\forall$; and that the last link generated by an honest node satisfies $F_R(r_{new}, n)$ implies that the honest nodes maintain the invariant. The next section details how to prove that malicious nodes cannot break the invariant.

B. Malicious Node Maintains the Invariant

Two challenges remain in checking whether a malicious node can violate the invariant φ_I^S: (1) given a route announcement r, how can we effectively check that $F_\forall(r)$ holds; (2) how can we identify all possible route announcements that an attacker can generate.

To address the first challenge, we leverage the fact that φ_I^S holds before the attacker generating a route update, and define *local* checks that only depend on the attacker's current state. For the second challenge, we quantify the attacker's capability and reduce the check to a finite set of scenarios where the attacker exactly knows one route update of length 2.

Local checks. We define semantics for formulas based on local states. We write $\mathcal{A}, G, M_u \vDash F_R(r,i)$ to mean that a route-dependent formula is true given the state of a malicious node u. The interesting cases are the rules for the predicate sentR and $\mathsf{sentToR}$. Predicate $\mathsf{sentR}\,(r,i)$ is true locally if there is a route announcement r' in U, and the common prefix between r and r' includes the prefix of r of length i. For example, assume that the route announcement $([u_1,u_2],d,u_{recv},\Sigma)$ is in U, then $\mathsf{sentR}\,(([u_1,u_2,v_1,...],d,v_{recv},\Sigma'),2)$ is true. The rule for $\mathsf{sentToR}$ is similar. Checking whether a formula is true based on local states is simple and can be automated.

The local semantics are sound with regard to the trace semantics, if the invariant φ_I^S satisfies Condition 3, which ensures that if a route update from an honest node exists in U then there is a corresponding send action on the trace. (Here $\vdash$ is the logical entailment)

Condition 3 (Invariant)

- *if* $\mathsf{sentR}\,(r,i)$ *is a subformula of* F_R*, then* $F_R(r,i) \vdash \mathsf{honest}\,(i) \supset \mathsf{sentR}\,(r,i)$
- *if* $\mathsf{sentToR}\,(r,i)$ *is a subformula of* F_R*, then* $F_R(r,i) \vdash \mathsf{honest}\,(i) \supset \mathsf{sentToR}\,(r,i)$

All the route announcements the attacker knows satisfy $\forall i.F_R(r,i)$, as specified by φ_I^S, together with the above condition, we know that each honest node appearing in the route announcement has sent out the announcement for that subpath. Therefore, if sentR is true based on the local checks, it is true on the trace.

Finite scenarios. An attacker has very limited capabilities to generate well-formed route updates: it can do so only by (1) constructing r without using any known route announcement, or (2) using a prefix of the list of signatures in an existing route announcement and adding more signatures signed using the secret keys that it knows. This makes it possible to enumerate all possible scenarios based on a fixed set of network topologies and initial knowledge of the adversary. We proved that checking Condition 4 is enough to prove that all the route updates that an attacker can generate satisfy $F_\forall(r)$ in protocol 2. The conditions for protocol 1 is similar and the only difference is that the topologies are listed in Figure 4, and that the signatures do not include the recipient node.

When supplied with one route update of length 2, an attacker would be able to perform all possible operations. Further, Condition 4 only checks that the section of the new route that differs from the existing route satisfies $F_R(r,j)$.

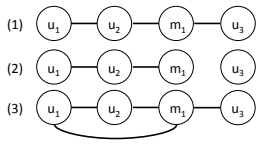

Fig. 4. Possible Topologies where m_1 is a malicious node

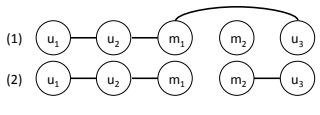

Fig. 5. Possible Topologies where m_1 is a malicious node, and it knows the private key of m_2

This is because the invariant φ_I^S ensures that existing sections satisfy $F_R(r,i)$. This condition can be finitely checked.

Condition 4 (New-Malicious)
Let $r_0 = ([u_1,u_2], m_1, d, \Sigma)$,
$\Sigma = [\mathsf{sign}((d,[u_1,u_2]),\mathsf{sk}(u_1)), \mathsf{sign}((d,[u_1,u_2,m_1]),\mathsf{sk}(u_2))]$,
$I = \{\mathsf{sk}(m_1),\mathsf{sk}(m_2),u_1,u_2,u_3,\mathsf{pk}(u_1),\mathsf{pk}(u_2),\mathsf{pk}(u_3)\}$,
$M_{m_1} = (m_1, I, \{r_0\})$,
$\forall G \in \mathcal{G}$ *(shown in Figure 5),* $\forall r \in \mathcal{K}(M_{m_1})$, *if* $f_{ck}(r) = \mathsf{true}$ *then* $\forall j \geq i$ $\mathcal{A}, G, \{([u_1,u_2],m_1,d,\Sigma)\} \vDash F_R(r,j)$, *where* i *is the length of the longest common prefix between* r_0 *and* r.

For Condition 4 to be sufficient to ensure that the attacker cannot generate any attack trace in any network topology, we need to justify why it is correct to only consider a route $[u_1,u_2]$, where u_1 and u_2 has a direct link between them. We again leverage the invariant φ_I^S. The following condition requires that the invariants for Protocol 1 include the property that for any honest node, the link to the previous node exists (I); and for Protocol 2, links to the previous and the next node of an honest node exist (II).

Condition 5 (Local to Trace)
(i) $F_R(r,i) \vdash \mathsf{honest}(i) \supset \mathsf{linkPrev}(r,i)$
(ii) $F_R(r,i) \vdash \mathsf{honest}(i) \supset (\mathsf{linkPrev}(r,i) \wedge \mathsf{linkNext}(r,i))$

These two conditions ensure that the attacker cannot generate any attack trace. Intuitively, for each attack trace, a boolean assignment for the predicates in $\neg F_R(r,i)$ makes it true in some topology. The scenarios in Condition 4 enumerate all possible boolean assignments (constrained by φ_I^S) to the predicates in $F_R(r,i)$. For instance, for $\mathsf{honest}(i) = \mathsf{true}$, $\mathsf{linkPrev}(r,i) = \mathsf{true}$, and $\mathsf{linkNext}(r,i) = \mathsf{false}$, for Protocol 2, topology (1) in Figure 4, the route update $([u_1,u_2,m_1,m_2], m_2, d, \Sigma)$ will have the above boolean assignment for $i = 3$. Assignments such as $\mathsf{honest}(i) = \mathsf{true}$, $\mathsf{linkPrev}(r,i) = \mathsf{false}$ will be discarded because Condition 5 ensures that such scenario need not be considered.

C. Summary and Discussion

The proof of route authenticity is reduced to checking the leaf conditions in Figure 3. Condition 1 is checked by careful examination of definitions of the protocol. Condition 2 holds since all the routing tables and queues are empty in the initial states. Condition 3 and Condition 5 are checked by manual inspection of the strong invariant φ_I^S. Condition 4 is checked using Proverif, which we discuss in Section IV.

Finally, we discuss how to generate attack traces from failed checks of Condition 4. Whenever Condition 4 is not true, it must be the case that the attacker can come up with a valid route ($f_{ck}(r) = \mathsf{true}$), but there is an index i such that $F_R(r,i)$ is not true. However, this route itself may not be an attack trace since an honest node may reject r after checking $f_a(r)$, which is true only if the sender of r is a direct neighbor of v. For a route update r such that $f_{ck}(r) = \mathsf{true}$, but $f_a(r,v) = \mathsf{false}$, it must be the case that the link between the sender of r to v doesn't exists. Let $r = (p@[u], v, d, \Sigma)$, then the attacker can send r to another colluding attacker node u_1, which has a link to an honest node v_1. u_1 can generate a route $r' = (p@[u,u_1], v_1, d, \Sigma@[\mathsf{sign}((d, p@[u,u_1]), \mathsf{sk}(u_1))])$ that the honest node v_1 will accept, but does not satisfy $F_\forall$. Thus r' is an attack route even though r is not.

	$F_R(r,i)$	**Prot. 1** (Fig 4)	**Prot. 2** (Fig 5)
Prop.(1)	$\mathsf{honest}(i) \supset (\mathsf{linkPrev}(r,i) \wedge \mathsf{sentR}(r,i))$	valid	valid
Prop.(2)	$\mathsf{honest}(i) \supset (\mathsf{linkPrev}(r,i) \wedge \mathsf{linkNext}(r,i) \wedge \mathsf{sentToR}(r,i))$	T(1);T(2);T(3): $([u_1], m_1, d, -)$	valid

Fig. 6. Analysis Results of Protocol 1 and Protocol 2

IV. Case Study

We present our analysis results of the two variants of S-BGP. Our analysis of checking Condition 4 is facilitated by Proverif [8], an automated cryptographic protocol verifier. The encoding is available at http://netdb.cis.upenn.edu/sbgp.tar.gz.

We encode the protocol as a π-calculus process, where the capabilities of an attacker are hard-coded as a process, as opposed to Proverif's standard attacker model[1]. As described in Condition 4, we provide the attacker process with a route update of length 2 and query whether the attacker can construct any route update violating the property $F_\forall(r)$.

Table 6 summaries results of the security analysis. Each row contains the results of verifying one specific property ($F_R(r,i)$) listed in the second column. Recall that the top-level formula (Section II-B) is of the form: $\boxminus(\forall u \forall r, \mathsf{honest}(u) \wedge \mathsf{send}(u,r) \supset \forall i, F_R(r,i))$. For each protocol, we list the attack route update–the route update that does not satisfy $\forall i, F_R(r,i)$–and the topology in which this route update is generated. The topology is indexed by the number as presented in Figure 4 and 5. We omit the signature list from the route update, which can be straightforwardly deduced.

Prop (1) requires that the incoming link to an honest node must exist and that route announcement is made by the honest node. It is possible that even if there is a physical link in the network, the link may not be included in a route announcement as it is not part of the best routing path to the destination prefix.

This property holds for both protocols because an honest node only accepts routes from its direct neighbors, and route announcements are cryptographically authenticated.

Prop (2) additionally requires that the outgoing link from an honest node must exist. This property highlights the difference between the security guarantees of Protocol 1 and Protocol 2.

The attack scenario for Protocol 1 is that the attacker reuses a subset of the signatures in an existing update: though there

[1]The reason is that the routing protocols are recursive in nature, and our attempts to directly use Proverif's attacker model cause non-termination.

is no link between u_1 and m_1 and u_1 did not send the route announcement to m_1, m_1 can still extract the u_1's signature from the route announcement from u_2. However, this property holds for Protocol 2. An honest node only announces routes to its neighbors, and this information is protected by digital signatures. This property does not prevent worm-hole attacks: two malicious nodes can collude to introduce non-existent links. Assuming no collusion, Protocol 2 guarantees that links in a valid route announcement exist in the topology. Protocol 1 provides weaker security guarantees: one attacker by itself can introduce non-existent links.

V. Related Work

Prior work on formal network verification (e.g. [10], [11], [12]) have focused primarily on functional correctness of the protocols, but not security properties. Compared to work on analyzing cryptographic protocols [13], [14], [15], [16], [17], [18], [19], secure routing protocols have to deal with arbitrary network topologies and the security properties are recursive. Most model-checking techniques are ineffective in proving these recursive security properties. Our proof techniques bare similarities with prior proof-based techniques for analyzing cryptographic protocols [14], [16]. The novelty lies in the reduction steps and combining manual and automated proofs to prove security properties for any arbitrary network topology.

Recent work [20], [21], [22] aim to verify similar route authenticity properties on wireless routing protocols for mobile networks Identifying these attacks are reduced to constraint solving. It is further shown that the security analysis of a specific route authenticity property solely based on topology on these wireless routing protocols can be reduced to checking these properties on several four-node topologies [22].

There are several key differences between our paper and the above body of work. First, wireless protocols are typical reactive in nature, where routes are requested on demand by the sender in a highly mobile environment. A route is valid in this case as long as an actual physical path exists between sender and receiver. In the BGP setting, given that each AS advertises a path based on its routing policies, our route authenticity property not only rely on the network topology, but also whether a node has announced a route in the past.

Second, the technique presented in [22] would not directly work for us, since the attack route we care about include those that exist in the topology, but not announced because of routing policy. Even though our proof technique also reduces the attack scenarios to a handful of scenarios, it differs greatly from their approach. Our technique makes use of the properties of the signature list and invariant properties that is assumed to hold in the current state, and therefore, we are able to use non-recursive conditions to identify attacks (or prove security).

VI. Conclusion

We presented a reduction-based techniques that enables the route authenticity property of S-BGP to be automatically checked for any arbitrary network topology. Our case study using Proverif demonstrates the utility of our approach on two variants of S-BGP. Our work provides not only a formal account for the path authenticity properties, but also a formal proof that these properties hold on certain routing protocols. As our future work, we plan to apply our framework for analyzing recent clean-slate designs of secure Internet routing infrastructures (e.g. SCION [4] and ICING [5]).

Acknowledgment

We thank anonymous reviewers for their comments and suggestions. This work was supported by NSF grants CNS-0845552, CNS-1065130, CNS-1117052, CNS-1115706, and IIS-0812270, ONR grant N00014-11-1-0555, and AFOSR MURI grant FA9550-08-1-0352.

References

[1] S. Kent, C. Lynn, J. Mikkelson, and K. Seo, "Secure border gateway protocol (S-BGP)," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 103–116, 2000.

[2] T. Wan, E. Kranakis, and P. C. Oorschot, "Pretty secure BGP (psBGP)," in *Proceedings of 12th NDSS*, 2005.

[3] R. White, "Securing bgp through secure origin BGP (soBGP)," *The Internet Protocol Journal*, vol. 6, no. 3, pp. 15–22, 2003.

[4] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen, "Scion: Scalability, control, and isolation on next-generation networks," in *Proceedings of Oakland S&P*, 2011.

[5] J. Naous, M. Walfish, A. Nicolosi, D. Mazieres, M. Miller, and A. Seehra, "Verifying and enforcing network paths with ICING," in *Proceedings of CoNEXT*, 2011.

[6] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford, "A survey of BGP security issues and solutions," *Proceedings of the IEEE*, vol. 98, pp. 100–122, January 2010.

[7] Secure BGP, http://www.ir.bbn.com/sbgp/.

[8] B. Blanchet and B. Smyth, "Proverif 1.86: Automatic cryptographic protocol verifier, user manual and tutorial," http://www.proverif.ens.fr/manual.pdf.

[9] C. Chen, L. Jia, B. T. Loo, and W. Zhou, "Reduction-based security analysis of internet routing protocols," CIS Dept. University of Pennsylvania, Tech. Rep. MS-CIS-12-13, July 2012.

[10] K. Bhargavan, D. Obradovic, and C. A. Gunter, "Formal verification of standards for distance vector routing protocols," *J. ACM*, vol. 49, no. 4, 2002.

[11] A. Goodloe, C. A. Gunter, and M.-O. Stehr, "Formal prototyping in early stages of protocol design," in *Proceedings of ACM WITS*, 2005.

[12] D. Engler and M. Musuvathi, "Model-checking large network protocol implementations," in *Proceedings of NSDI*, 2004.

[13] A. Datta, A. Derek, J. C. Mitchell, and A. Roy, "Protocol Composition Logic (PCL)," *Electronic Notes in Theoretical Computer Science*, vol. 172, pp. 311–358, 2007.

[14] A. Roy, A. Datta, A. Derek, J. C. Mitchell, and S. Jean-Pierre, "Secrecy analysis in protocol composition logic," in *Proceedings of ESORICS*, 2007.

[15] C. He, M. Sundararajan, A. Datta, A. Derek, and J. C. Mitchell, "A modular correctness proof of IEEE 802.11i and TLS," in *Proceedings of CCS*, 2005.

[16] L. C. Paulson, "Mechanized proofs for a recursive authentication protocol," in *Proceedings of CSFW*, 1997.

[17] S. Escobar, C. Meadows, and J. Meseguer, "A rewriting-based inference system for the NRL protocol analyzer: grammar generation," in *Proceedings of FMSE*, 2005.

[18] B. Blanchet, "Automatic verification of correspondences for security protocols," *J. Comput. Secur.*, vol. 17, no. 4, Dec. 2009.

[19] J. Bau and J. Mitchell, "A security evaluation of DNSSEC with NSEC3," in *Proceedings of NDSS*, 2010.

[20] M. Arnaud, V. Cortier, and S. Delaune, "Modeling and verifying ad hoc routing protocols," in *Proceedings of CSF*, 2010.

[21] ——, "Deciding security for protocols with recursive tests," in *Proceedings of CADE*, 2011.

[22] V. Cortier, J. Degrieck, and S. Delaune, "Analysing routing protocols: four nodes topologies are sufficient," in *Proceedings of POST*, 2012.