1-1-2012

Reduction-Based Security Analysis of Internet Routing Protocols

Chen Chen
University of Pennsylvania, chenche@seas.upenn.edu

Limin Jia
Carnegie Mellon University, liminjia@cmu.edu

Boon Thau Loo
University of Pennsylvania, boonloo@cis.upenn.edu

Wenchao Zhou
University of Pennsylvania, wenchaoz@seas.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-12-13.

Reduction-Based Security Analysis of Internet Routing Protocols

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 accepted and announced by honest nodes in the network are not 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 checks that can be automatically done by an automated tool Proverif. We show that our analysis is correct with respect to the trace semantics of the routing protocols.

Disciplines
Computer Engineering

Comments
University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-12-13.

Reduction-based Security Analysis of Internet Routing Protocols

Chen Chen[1], Limin Jia[2], Boon Thau Loo[1], and Wenchao Zhou[1]

[1] University of Pennsylvania, Philadelphia, PA 19104, USA
[2] Carnegie Mellon University, Pittsburgh, PA 15213, USA

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 accepted and announced by honest nodes in the network are not 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 checks that can be automatically done by an automated tool Proverif. We show that our analysis is correct with respect to the trace semantics of the routing protocols.

1 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 traditional security extensions to Internet routing [15, 20, 21], to "clean-slate" redesigns [22, 16]. 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 argued 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. Briefly, 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 Provider (*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. Given these faults may lead to long periods of interruption of the Internet, we focus on the *route authenticity* property in Internet routing protocols. Route authenticity requires that routes accepted and announced by honest ASes in the network should not be tampered with by the adversary.

There have been a variety of proposed mechanisms [7] that aim to provide or improve the route authenticity of the Internet routing. For example, Secure BGP (S-BGP) [15] and pretty secure BGP [20] 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 [19, 20, 22, 16]. Intuitively, attackers do not have the private key of honest nodes, and therefore cannot fake 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.

Our framework combines manual proofs with automated analysis, through the use of novel *reduction* techniques. Instead of analyzing arbitrary large network instances, we define several reduction steps to reduce the route authenticity proofs to checks that can either be performed by an automated tool Proverif [6] or simple enough to be discharged manually. 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 provides not only a formal proof that a given security property is guaranteed by a routing protocol, we also provide evidence that the protocol is vulnerable to particular forms of attacks. 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 define trace semantics for the protocols. We formally define route authenticity properties in a first-order temporal logic whose semantics is given by the traces (Section 2).
- We define reduction steps that reduce the proof of route authenticity to conditions that can be either automatically checked or simple enough to be manually discharged. We have formally proven the correctness of our reduction steps (Section 3).
- We use Proverif, an automated tool, to check the most complicated conditions, and present case study results (Section 4).

2 Formal 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. We omit the details of

Malicious Node	$M ::= (u, I, U)$	*Routing state*	$S ::= \varnothing \mid S, M \mid S, H$
Honest Node	$H ::= (u, I, RtTb, Q)$	*Update*	$r ::= (p, u_r, d, \Sigma)$
Path	$p ::= [u_1, \cdots u_n]$	*Trace*	$\mathcal{T} ::= S \mid \mathcal{T} \xrightarrow{\alpha} S$
Signature	$\sigma ::= \mathsf{sign}((d, p), \mathsf{sk}(u))$	*Action*	$\alpha ::= sendA(u, v, r)$
Signatures	$\Sigma ::= [\sigma_1, \cdots, \sigma_n]$		
Protocol	$\mathcal{P} ::= (f_{orig}(), f_{upd}(), f_{ck}(), f_a(), \delta)$		
Transition Function	$\delta \in States \times Actions \rightarrow States$		

Fig. 1. Constructs for Modeling Routing Protocols

BGP's import and export policies, since they are not directly relevant to our security analysis. Instead, we model the policies by a node's non-deterministic choice of routes to announce to its neighbors. We formally define route authenticity properties, which will be used as a basis for our security analysis of these two protocol variants in the next section.

2.1 Formal Model for Secure Routing

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, modelled using the Dolev-Yao attacker model. We write $\mathcal{A}$ to denote the set of malicious nodes. To simplify our analysis, we assume that the topology (G) and the set of malicious nodes ($\mathcal{A}$) do not change during the execution of the protocol.

We capture the behavior of routing protocols using a transition system. The syntactic constructs that we use to formally model 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 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.

We write H to denote the state of an honest node. In addition to a node id, and a set of initial knowledge, an honest node uses a routing table $RtTb$ to store known routes to different destinations, and Q to store the set of update messages to be processed by the honest node. We write $H(u)$ ($M(u)$) to denote an honest (a malicious) node of id u. A state in the routing protocol S consists of all the nodes in the network.

A path p is a list of nodes. We write $l_1@l_2$ to denote the concatenation of two lists l_1 and l_2. We write σ to denote a digital signature. 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 is written Σ. We write r to denote a route update message, where p is the path, d is the destination, u_r is recipient node of this update, and Σ is a set of signatures associated with this update. The path-vector protocol and the signature-based authentication mechanism is parameterized using the following functions:

- $f_{orig}(d, u_s, u_r)$ computes a route update for the destination prefix d that originates at node u_s.
- $f_{upd}(r, u_s, u_r)$ computes a new route update from node u_s to node u_r based on a route r that u_s received before.
- $f_{ck}(r)$ checks the validity of all the signatures in a route update. It returns true if the route update has the correct format and all the signatures are valid. We say a route update is valid when $f_{ck}(r) = \mathsf{true}$.
- $f_a(G(u), r, u)$ is a check on a route update by node u. Note that $f_a()$ uses $f_{ck}()$ as sub-routine. In our case studies, $f_a()$ additionally checks that the route is sent from a direct neighbor. We assume that each node knows its direct neighbors in the topology $G(u)$. This is typically done through a separate neighbor discovery process which we assume is correct.

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 caused by actions. 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, and an initial state S_0. The behavior of a routing protocol is captured by the set of traces defined by $\mathcal{R}[(\mathcal{A}, G, \mathcal{P}, S_0)]$.

We write $\mathcal{K}(M(u))$ to denote all the knowledge that a malicious node with node id u can derive based on the standard Dolev-Yao attacker model. $\mathcal{K}(M(u))$ is defined by a set of inference rules listed in Figure 7 in Appendix A. Attackers can deconstruct terms, construct terms from known ones, including generating signatures using the private keys it knows.

Transition function. The transition function δ for the path-vector protocol is defined below. For each data structure, we add an apostrophe to represent the same data structure in the resulting state. For instance, $RtTb'(u)$ is the routing table of node u in the resulting state. We write $RtTb[r]$ to denote the route table resulted from adding a route update r.
$\delta(S, sendA(u, v, r_{new})) = S'$

- u is an honest node, $r_{new} = f_{orig}(u, d, v)$, $uv \in E$; $\forall w \in \mathcal{A}$, $U'(w) = U(w) \cup \{r_{new}\}$, and if $v \notin \mathcal{A}$, $Q'(v) = Q(v) \cup \{r_{new}\}$;
- u is an honest node, $Q'(u) = Q(u) \backslash \{r_1\}$, $f_a(r_1, u) = \mathsf{true}$, $RtTb'(u) = RtTb[r_1]$, $uv \in E$, $r_{new} = f_{upd}(r', u, v)$ where $r' \in RtTb'(u)$; $\forall w \in \mathcal{A}$, $U'(w) = U(w) \cup \{r_{new}\}$, and if $v \notin \mathcal{A}$, $Q'(v) = Q(v) \cup \{r_{new}\}$;
- u is a malicious node, let $r_{new} = \mathcal{K}(M(u))$, $\forall w \in \mathcal{A}$ and $w \neq u$, $U'(w) = U(w) \cup \{r_{new}\}$, and if $v \notin \mathcal{A}$, $Q'(v) = Q(v) \cup \{r_{new}\}$.

There are three possible transitions: (1) an honest node originates a route; (2) an honest node receives a route and announces an update based on the received route; and (3) a malicious node sends out an announcement. For each transition, we allow every malicious node to know the route announcement. These rules, combined with the definitions of $\mathcal{K}(M(u))$, capture the adversary's capabilities: an attacker can eavesdrop on all route announcements, send out route announcements to any other node, and decompose and construct messages.

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

Fig. 2. Syntax of Formulas

Two S-BGP variants. Both variants of S-BGP associate a route announcement with a list of signatures: one for each subpath in the route announced. Both protocols use the same transition systems introduced in the previous section. The difference is the signature format, and its corresponding $f_{ck}(r)$ function.

We define the auxiliary functions for each protocol below. We use $\mathsf{verify}(\sigma, t, \mathsf{pk}(u))$ to represent the signature checking function that returns true if $\sigma = \mathsf{sign}(t, \mathsf{sk}(u))$.

Protocol 1
$f_{orig}(d, u_s, u_r) = ([u_s], d, u_r, [\mathsf{sign}((d, [u_s]), \mathsf{sk}(u_s))])$
$f_{upd}((p, d, u_s, \Sigma), u_s, u_r) = (p@[u_s], d, u_r, \Sigma@[\mathsf{sign}((d, p@[u_s]), \mathsf{sk}(u_s))])$
$f_{ck}([u], u_r, d, [\sigma]) = \mathsf{true}$ iff $\mathsf{verify}(\sigma, (d, [u]), \mathsf{pk}(u)) = \mathsf{true}$
$f_{ck}(p@[u], u_r, d, \Sigma@[\sigma]) = \mathsf{true}$
iff $f_{ck}(p, u, d, \Sigma) = \mathsf{true}$ and $\mathsf{verify}(\sigma, (d, p@[u]), \mathsf{pk}(u)) = \mathsf{true}$

Protocol 2
$f_{orig}(d, u_s, u_r) = ([u_s, u_r], d, u_r, [\mathsf{sign}((d, [u_s]), \mathsf{sk}(u_s))])$
$f_{upd}((p, d, u_s, \Sigma), u_s, u_r) = (p@[u_s], d, u_r, \Sigma@[\mathsf{sign}((d, p@[u_s, u_r]), \mathsf{sk}(u_s))])$
$f_{ck}([u], u_r, d, [\sigma]) = \mathsf{true}$ 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 $\mathsf{verify}(\sigma, (d, p@[u, u_r]), \mathsf{pk}(u)) = \mathsf{true}$

The difference between these two protocols is that the intended receiver is part of the signature in Protocol 2, while it is not the case in Protocol 1. Both have the same definition for the $f_a()$ function:

$f_a(r, v) = f_{ck}(r)$ and $uv \in E$ where $r = (p@[u], d, v, \Sigma)$.

2.2 A Logic for Expressing Route Authenticity Properties

We use a first-order logic with one temporal modality to express the route authenticity properties. We explain its syntax and trace semantics.

Syntax. We summarize the syntax of our logical formulas in Figure 2. Predicates are written P, which includes a predicate representing links in the topology, a predicate stating whether a node is honest, and predicates on actions on the traces. We also define a set of route-dependent predicates $P_R(r, i)$ that are indexed by a route update and a natural number, which represents a position in the path in the route update. A node can be uniquely determined by the path in the route update and the index number. For instance if the route update is $r = ([u_1 \cdots u_n], u_r, d, \Sigma)$, then the i^{th} node of the update r is u_i. We define

$\mathcal{A}, G, \mathcal{T} \vDash \mathsf{send}\,(u, r)$ iff u is an honest node implies $\mathcal{T} = \mathcal{T}' \xrightarrow{sendA(u,v,r)} S$

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

Fig. 3. Semantics of Formulas

universally quantified route-dependent formula, which is used to define recursive properties on a path.

Formulas φ are the standard first-order logic formulas that make use of the predicates and route-dependent formulas. To express trace properties, we make use of 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 on the last state of the trace $\mathcal{T}$, given the topology and the set of malicious nodes. It is inductively defined over the structure of φ. Figure 3 is a summary of the interesting rules.

Predicate $\mathsf{send}\,(u, r)$ is true if the action leading to the current state is node u sending a route update announcing r. In addition, when i 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.

The rules for route-dependent predicates use $[\![i]\!]_r$ 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: $[\![n+1]\!]_{(p,u_r,d,\Sigma)} = u_r$.

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. Predicates sentR and $\mathsf{sentToR}$ are in effect, past formulas. 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. We can formally specify the route authenticity properties based on properties on a given route update. 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.To give another example, $\forall i, \mathsf{sentR}\,(r, i)$ requires all the sub-routes in r were really announced as route updates.

The top-level property for the routing protocols is of the form:

$\mathsf{RouteAuth} = \boxminus\varphi_I$

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

This formula requires a route announcement r from an honest node has to satisfy certain properties ($F_\forall(r)$). 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 $F_\forall(r)$. The goal of our security analysis is to find properties $F_\forall(r)$ such that φ_I is an invariant for all execution traces of the protocol. For properties that we cannot prove it holds on all execution traces, we would like to generate attack scenarios automatically.

3 Security Analysis of Routing Protocols

In this section, we explain our reduction-based approach to verifying the route authenticity properties of secure routing protocols. We summarize our reduction steps in Figure 4. The section number of the explanations of the reduction step appears on the right hand side of the rule. These reduction steps form a proof tree proving that route authenticity property holds on all execution traces of a protocol, if all the leaf conditions can be verified. This figure serves as a road map for the rest of this section (Sections 3.1-3.6). In Section 3.7, we summarize our approach, discuss how these conditions can be verified, and explain how attack scenarios are generated.

3.1 Considering Only Well-formed Route Updates is Sufficient

First, we narrow down the traces to only consider those where attackers always send and store route updates that pass the check $f_{ck}(r)$. These traces have a much simpler invariant to specify and prove. We write $\mathcal{R}_{ck}[(\mathcal{A}, G, \mathcal{P}, S_0)]$ to denote the largest subset of $\mathcal{R}[(\mathcal{A}, G, \mathcal{P}, S_0)]$ such that for any action $sendA(u, v, r) \in \mathcal{T}$, where $\mathcal{T} \in \mathcal{R}_{ck}[(\mathcal{A}, G, \mathcal{P}, S_0)]$, it must be the case that $f_{ck}(r)$.

The soundness of this reduction step is proven by refinement. We detail the proofs in Appendix B.1. We first show that if $\mathcal{T}_1$ is an execution trace of the protocol, then there is a corresponding execution trace $\mathcal{T}_2$ where the attackers only send out well-formed route updates, and $\mathcal{T}_1$ and $\mathcal{T}_2$ agree on the actions and states of the honest nodes (Lemma 11). We further show that the invariant that we want to prove does not depend on the actions and the states of malicious nodes (Lemma 12). Combining the above, we can show that if φ_I is an invariant on all traces in $\mathcal{R}_{ck}[(\mathcal{A}, G, \mathcal{P}, S_0)]$, then it is an invariant for all the traces in $\mathcal{R}[(\mathcal{A}, G, \mathcal{P}, S_0)]$ (Theorem 1).

Theorem 1 *If* $\forall \mathcal{T}_1 \in \mathcal{R}_{ck}[(\mathcal{A}, G, \mathcal{P}, S_0)], \mathcal{A}, G, \mathcal{T}_1 \vDash \boxminus\varphi_I$ *then* $\forall \mathcal{T}_2 \in \mathcal{R}[(\mathcal{A}, G, \mathcal{P}, S_0)], \mathcal{A}, G, \mathcal{T}_2 \vDash \boxminus\varphi_I$

$C1$: Last link of the route generated by an honest node satisfies $F_R(r, n)$
$C2$: φ_I^S holds initially
$C5$: Local assumptions about sentR and sentToR are justified by the φ_I^S
$C6$: Updates generated by a malicious node given an update of length 2 satisfy $F_\forall(r)$
$C7$: Local assumptions about linkPrev and linkNext are justified by the φ_I^S

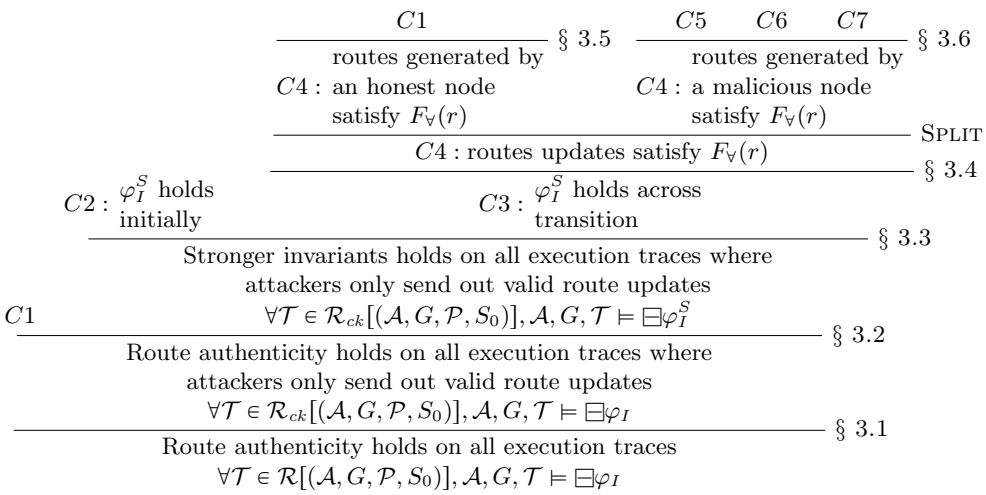

Fig. 4. Reduction Steps

3.2 Stronger Invariants

To prove the route authenticity properties RouteAuth, we need to establish a stronger invariant φ_I^S. Formula φ_I^S states that all the route updates stored in the routing table of an honest node must satisfy 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.

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

Additionally, the protocol needs to satisfy the following condition, which requires that the last link in the new route update satisfies the property $F_R(r, n)$.

Condition 1 (Honest) *If* $r_{new} = f_{upd}(r, u, v)$, *and* $|r_{new}| = n$
then $\mathcal{A}, G, \mathcal{T} \xrightarrow{sendA(u,v,r_{new})} S' \vDash F_R(r_{new}, n)$

The new route update is constructed by calling $f_{upd}(r, u, v)$ function, which extends an existing route in the routing table with one more link. From the strong invariant φ_I^S, we can infer that r satisfies the property $F_\forall$. If the last link in the new route update satisfies the property $F_R(r_{new}, n)$, where n is the length of r, then the entire update r_{new} satisfies $F_\forall$, formally:

Lemma 2 *If* $\mathcal{A}, G, \mathcal{T} \vDash \varphi_I^S$, $\mathcal{A}, G, \mathcal{T} \xrightarrow{sendA(u,v,r_{new})} S' \vDash F_R(r_{new}, n)$ *where* $r_{new} = f_{upd}(r, u, v)$ *and* $|r_{new}| = n$, *then* $\mathcal{A}, G, \mathcal{T} \xrightarrow{sendA(u,v,r_{new})} S' \vDash F_\forall(r_{new})$

We can prove that the strong invariant φ_I^S combined with Condition 1 imply that φ_I is an invariant (Theorem 3) using Lemma 2.

Theorem 3 *If* $\mathcal{A}, G, \mathcal{T} \vDash \boxminus \varphi_I^S$ *and Condition 1 holds then* $\mathcal{A}, G, \mathcal{T} \vDash \boxminus \varphi_I$

3.3 Non-inductive Conditions

The proof of an invariant holds on any execution trace is often carried out by induction over the trace. Applying ideas of rely-guarantee reasoning principles [14], we can reduce the inductive proofs into non-inductive conditions. We write $\mathcal{T}(i)$ to denote a trace from state S_0 to S_i.

Condition 2 (init) $\mathcal{A}, G, S_0 \vDash \varphi_I^S$

Condition 3 (inv) *if* $\mathcal{A}, G, \mathcal{T}(i) \vDash \varphi_I^S$ *and* $\delta(S_i, \alpha) = S_{i+1}$, *then* $\mathcal{A}, G, \mathcal{T}(i) \xrightarrow{\alpha} S_{i+1} \vDash \varphi_I^S$,

Theorem 4 *Condition 2 and 3 imply* $\forall \mathcal{T} \in \mathcal{R}[(\mathcal{A}, G, \mathcal{P}, S_0)]$, $\mathcal{A}, G, \mathcal{T} \vDash \boxminus \varphi_I^S$.

Proof (sketch): By induction over the trace. Condition 2 proves the base case. For the inductive case, we apply the Induction Hypothesis and Condition 3. □

Next, we further simplify Condition 3.

3.4 Checking New Route Updates is Sufficient

The predicates used in specifying properties of routes 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, we can further reduce Condition 3 to the following. For a state S_i, we use i to index the data structures in that state.

Condition 4 (inv-new)

- $\forall r \in RtTb_{i+1}(u)$ *s.t.* $r \notin RtTb_i(u)$, $\mathcal{A}, G, \mathcal{T}(i) \xrightarrow{\alpha} S_{i+1} \vDash F_\forall(r)$.
- $\forall r \in Q_{i+1}(u)$ *s.t.* $r \notin Q_i(u)$, *and* $f_{ck}(r) = \mathsf{true}$. $\mathcal{A}, G, \mathcal{T}(i) \xrightarrow{\alpha} S_{i+1} \vDash F_\forall(r)$
- $\forall r \in U_{i+1}(u)$ *s.t.* $r \notin U_i(u)$, $\mathcal{A}, G, \mathcal{T}(i) \xrightarrow{\alpha} S_{i+1} \vDash F_\forall(r)$

The following lemma states that a route-dependent formula remains true when a trace is extended. It can be proved straightforwardly by induction on the structure of the formula.

Lemma 5 *If* $\mathcal{A}, G, \mathcal{T}(i) \vDash F_R(r, k)$ *then* $\mathcal{A}, G, \mathcal{T}(i) \xrightarrow{\alpha} \mathcal{T}' \vDash F_R(r, k)$

Using Lemma 5, we can prove the soundness of this reduction step.

Theorem 6 *If* $\mathcal{A}, G, \mathcal{T} \vDash \varphi_I^S$ *and* $\delta(S_i, \alpha) = S_{i+1}$, *and Condition 4 holds then* $\mathcal{A}, G, \mathcal{T}(i) \xrightarrow{\alpha} S_{i+1} \vDash \varphi_I^S$,

Proof (sketch): We need to show that for every route update r in Q_{i+1}, $RtTb_{i+1}$ and U_{i+1}, $\mathcal{A}, G, \mathcal{T}(i) \xrightarrow{\alpha} S_{i+1} \vDash F_\forall(r)$. Since we know that φ_I^S holds at state S_i, by Lemma 5, we know that for any r that is in both state S_i and S_{i+1}, it is the case that $\mathcal{A}, G, \mathcal{T}(i) \xrightarrow{\alpha} S_{i+1} \vDash F_\forall(r)$. For route updates that are not in S_i, Condition 4 ensures that $\mathcal{A}, G, \mathcal{T}(i) \xrightarrow{\alpha} S_{i+1} \vDash F_\forall(r)$ as well. □

Given the transition system defined in § 2, we further simplify Condition 4 based on whether the transition is made by an honest or a malicious node.

3.5 Condition on Honest Nodes

Condition 4 can be reduced to Condition 1 when an honest node takes a step.

Theorem 7 *If Condition 1 holds, and φ_I^S holds on the current state (S_i), then Condition 4 holds*

Proof (sketch): First, an honest node only updates its routing table with a route update, which is directly taken from the queue and passes the validity check $f_a(r)$. Since we already assume φ_I^S holds at the current state, we can infer that the new route update in the routing table indeed satisfies $F_\forall$. Therefore, the first bullet of condition 4 is trivially satisfied.

Second, all the new route updates in state S_{i+1} are generated by an honest node. By Lemma 2, we know that $F_\forall(r)$ is true for each of these updates. □

3.6 Malicious Node Maintains the Invariant

The last piece of our security analysis of the protocols is to prove that any route announcement r generated by the malicious node satisfies the property $F_\forall(r)$. We face two challenges: (1) how can we identify all possible route announcements that an attacker can come up with; and (2) given a route announcement r, how can we effectively check that $F_\forall(r)$ holds.

Semantics for $F_\forall(r)$ based on local states. To decide the validity of predicates sentR and $\mathsf{sentToR}$, we need the entire execution trace. However, this would require us to consider all possible traces, which is hard for automated analysis. We can leverage properties stated in the invariant φ_I^S, and reduce deciding the validity of these predicates to only relying on an attacker's current state. The fact that φ_I^S is assumed to hold in the current state reduces the space of the traces that we need to consider. When specified properly φ_I^S, allows us to only consider the current state.

We first define semantics for route-dependent formulas based on an attacker's state. We write $\mathcal{A}, G, M(u) \vDash F_R(r, i)$ to mean that a route-dependent formula is true given the set of malicious nodes, topology, and the state of a malicious node u. The interesting cases are the rules for the predicate sentR and $\mathsf{sentToR}$. We explain the rule for sentR, and $\mathsf{sentToR}$ is similar, and can be found in Figure 8 in Appendix B.2. Predicate $\mathsf{sentR}\,(r, i)$ is true 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.

$\mathcal{A}, G, (u, I, U) \vDash \mathsf{sentR}\,((p, u_{recv}, d, \Sigma), i)$
iff u is an honest node implies
there exists a $r' \in (u, I, U)$, st.$r' = (p', u'_{recv}, d, \Sigma')$
p and p' has a subpath p'', and $|p''| = i$

The local semantics are sound with regard to the trace semantics, if the invariant φ_I^S satisfies the following conditions: (Here $\vdash$ is the logical entailment)

Condition 5 (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* $F_R(r, i) \vdash \mathsf{honest}\,(i) \supset \mathsf{sentToR}\,(r, i)$

Lemma 8 *If* $\mathcal{A}, G, \mathcal{T} \vDash \varphi_I^S$*, and* $M(u) \in S_i$*, where* S_i *is the last state on* $\mathcal{T}$*, then Condition 5 and* $\mathcal{A}, G, M(u) \vDash F_R(r, k)$ *implies* $\mathcal{A}, G, \mathcal{T} \vDash F_R(r, k)$*.*

Proof (sketch): By induction on the structure of $F_R(r, i)$. Most cases are straightforward. The interesting cases are predicate sentR and $\mathsf{sentToR}$.

Since we assume that all the route announcements the attacker knows satisfy $\forall i.F_R(r, i)$, and from the above condition, we know that each honest node appearing in the route announcement has sent out the announcement for that subpath. Therefore, sentR is true on the trace as well. □

Attackers have limited capabilities. An attacker can only generate a valid route update either by constructing it without using any known route announcement, or by 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. The proofs can be found in Appendix B.2.

Lemma 9 *Let* $M(u) = \{u, I, U\}$ *and* $\nexists\sigma \in I$*, if* $\forall r \in U$*,* $f_{ck}(r) = \mathsf{true}$*, then* $\forall r_0 \in \mathcal{K}(M(u))$ *such that* $f_{ck}(r_0)$*,*

1. *either* $r_0 \in \mathcal{K}(\{u, I, \varnothing\})$
2. *or* $\exists r_1 \in U$*, and* $\exists r_2$ *such that* $r_2 = (p, u_r, d, \Sigma)$*,* $r_1 = (p@[v_1 \cdots v_j], \cdots)$ $r_0 = (p@[u_1 \cdots u_k], \cdots)$*, where* $\mathsf{sk}(u_i) \in I$ *(*$i \in [1, k]$*)*

Reduction to finite scenarios We are able to reduce Condition 4 to a few scenarios where the attacker exactly knows one route update of length 2. Intuitively, Lemma 9 shows that there are only limited operations that an adversary can perform to generate new route updates. If we supply the adversary with a route update of length 2, then the attacker would be able to perform all of the possible operations.

Similar to the condition for the honest nodes, Condition 6 only requires that each node in the section of the new route that differs from the existing route to satisfy $F_R(r, j)$. There is a different condition for each of the protocols: Condition 6.(I) for Protocol 1 and Condition 6.(II) for Protocol 2. These two conditions only differ in the route updates format, and the topologies to consider.

Condition 6 (New-Malicious)

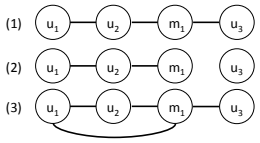

Fig. 5. Possible Topologies

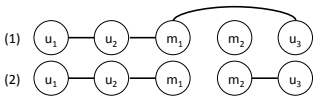

Fig. 6. Possible Topologies

(I) *Let* $r_0 = ([u_1, u_2], m_1, d, \Sigma)$,
$\Sigma = [\mathsf{sign}(\mathsf{sk}(u_1), (d, [u_1])), \mathsf{sign}(\mathsf{sk}(u_2), (d, [u_1, u_2]))]$,
$I = \{\mathsf{sk}(m_1), u_1, u_2, u_3, \mathsf{pk}(u_1), \mathsf{pk}(u_2), \mathsf{pk}(u_3)\}$, $M(m_1) = (m_1, I, \{r_0\})$,
$\forall G \in \mathcal{G}$, $\forall r \in \mathcal{K}(M(m_1))$, *if* $f_{ck}(r) = \mathsf{true}$ *then* $\forall j \geqslant i$ $G, \{([u_1, u_2], m_1, d, \Sigma)\} \models F_R(r, j)$, *where* i *is the length of the largest common prefix between* r_0 *and* r, *and* $\mathcal{G}$ *is defined in Figure 5.*

(II) *Let* $r_0 = ([u_1, u_2], m_1, d, \Sigma)$,
$\Sigma = [\mathsf{sign}(\mathsf{sk}(u_1), (d, [u_1, u_2])), \mathsf{sign}(\mathsf{sk}(u_2), (d, [u_1, u_2, m_1]))]$,
$I = \{\mathsf{sk}(m_1), \mathsf{sk}(m_2), u_1, u_2, u_3, \mathsf{pk}(u_1), \mathsf{pk}(u_2), \mathsf{pk}(u_3)\}$, $M(m_1) = (m_1, I, \{r_0\})$,
$\forall G \in \mathcal{G}$, $\forall r \in \mathcal{K}(M(m_1))$, *if* $f_{ck}(r) = \mathsf{true}$ *then* $\forall j \geqslant i$ $G, \{([u_1, u_2], m_1, d, \Sigma)\} \models F_R(r, j)$, *where* i *is the length of the largest common prefix between* r_0 *and* r, *and* $\mathcal{G}$ *is defined in Figure 6.*

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, which is assumed to hold at the current state. The following condition requires that the invariants for Protocol 1 include the property that for any honest node, the link to the previous node always exists (I); and for Protocol 2, links to the previous and next nodes always exist for an honest node (II).

Condition 7 (Local to Trace)

(I) $F_R(r, i) \vdash \mathsf{honest}(i) \supset \mathsf{linkPrev}(r, i, d)$
(II) $F_R(r, i) \vdash \mathsf{honest}(i) \supset (\mathsf{linkPrev}(r, i, d) \wedge \mathsf{linkNext}(r, i, d))$

We can prove that we can reduce Condition 4 to checking Conditions 5 - 7, assuming that φ_I^S holds in the current state. We list the intermediary lemmas in Appendix B.2, and present the final theorem below. The check function of Protocol 1 (Protocol 2) is $f_{ck}^1()$ ($f_{ck}^2()$) respectively.

Theorem 10

(I) *Let* $\mathcal{T} = S_0 \xrightarrow{\alpha} \cdots S$, $\mathcal{T}' = \mathcal{T} \xrightarrow{sendA(u,v,r_a)} S'$, $M(u) = (u, I, U) \in S$, *and* $f_{ck}^1(r_a) = \mathsf{true}$, *and* $\forall r \in U$, $\mathcal{A}, G, \mathcal{T} \models F_\forall(r)$ *then Conditions 5, 6.(I), 7.(I) imply* $\mathcal{A}, G, \mathcal{T} \models F_\forall(r_a)$ *(Condition 4)*

(II) *Let* $\mathcal{T} = S_0 \xrightarrow{\alpha} \cdots S$, $\mathcal{T}' = \mathcal{T} \xrightarrow{sendA(u,v,r_a)} S'$, $M(u) = (u, I, U) \in S$, *and* $f_{ck}^2(r_a) = \mathsf{true}$, *and* $\forall r \in U$, $\mathcal{A}, G, \mathcal{T} \models F_\forall(r)$ *then Conditions 5, 6.(II), 7.(II) imply* $\mathcal{A}, G, \mathcal{T} \models F_\forall(r_a)$ *(Condition 4)*

Proof (sketch): There are three steps. First, if an attacker can come up with an attack route that satisfies $\exists i.\neg F_R(r,i)$ given a set of updates, then it can come up with an attack route given just one of the route in the update U (Lemma 13). Using Lemma 9, we know that the attacker cannot use parts from two different route updates in one update.

Second, by definition, any attack trace must satisfy $\neg F_R(r,i)$ for some i. For each attack trace, there is a boolean assignment for the predicates in $\neg F_R(r,i)$, that makes $\neg F_R(r,i)$ true. The scenarios that we check in Condition 6 will enumerate all possible boolean assignments (constrained by φ_I^S) to the base predicates in $F_R(r,i)$. For instance, for the assignment $\mathsf{honest}\,(i) = \mathsf{true}$, $\mathsf{linkPrev}\,(i) = \mathsf{true}$, and $\mathsf{linkNext}\,(i) = \mathsf{false}$, for Protocol 1, topology (2) in Figure 5, the route update $([u_1, u_2, m_1], d, u_3, \Sigma)$ will have the above boolean assignment for $i = 3$. Of course, we create a scenario where $\mathsf{honest}\,(i) = \mathsf{true}$, $\mathsf{linkPrev}\,(i) = \mathsf{false}$ in Condition 6. This is when Condition 7 comes in, and we use φ_I^S to ensure that such scenario need not be considered. Therefore, if Condition 6 holds, then there is no attack trace on any topology (based on local semantics).

Third, using Lemma 8, we can conclude that all the traces generated by the attacker satisfies $F_\forall(r)$ given the trace semantics.

□

3.7 Summary and Discussion

By applying several sound reduction steps, the proof of route authenticity is reduced to checking the leaf conditions listed in Figure 4. Condition 1 is checked by careful examination of definitions of the protocol. Condition 2 holds trivially since all the routing tables and queues are empty in the initial states. Condition 5 and Condition 7 are checked by manual inspection of the strong invariant φ_I^S. We could use a theorem prover to discharge these two conditions if φ_I^S were more complicated. Condition 6 is the most complicated one to check, as we need to enumerate several cases. We use an automatic tool Proverif to help generate all possible route updates, and check whether they satisfy $F_\forall(r)$. We will briefly discuss how Proverif is used and summarize our analysis results in Section 4.

Finally, we would like to discuss how to generate attack traces from failed checks of Condition 6. Whenever Condition 6 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 in the sense that an honest node may reject it. The reason is that an honest node checks $f_a(r)$, not just $f_{ck}(r)$. Recall that for an honest node v to accept $f_a(r)$, it checks $f_{ck}(r) = \mathsf{true}$, and that 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 $r = (p@[u], d, v, \Sigma)$, then the attacker can similarly send r to another colluding attacker node u_1, which has a link to an honest node v_1. It is obvious that u_1 can generate a route $r' = (p@[u, u_1], d, v_1, \Sigma@[\mathsf{sign}((d, p@[u, u_1]), \mathsf{sk}(u_1))])$ that the honest node v_1 will accept, but that route does not satisfy $F_\forall$.

	$F_R(r,i)$	**Protocol 1** (Fig 5)	**Protocol 2** (Fig 6)
Prop.(1)	$\mathsf{linkPrev}\,(r,i)$	T(1);T(2): $([u_1,m_1],u_3,d,-)$	T(1);T(2): $([u_1,u_2,m_1,m_2],u_3,d,-)$
Prop.(2)	$\mathsf{linkNext}\,(r,i)$	T(1);T(2): $([u_1],m_1,d,-)$ T(2): $([u_1,u_2,m_1],u_3,d,-)$	T(1): $([u_1,u_2,m_1,m_2],u_3,d,-)$ T(2): $([u_1,u_2,m_1],m_2,d,-)$
Prop.(3)	$\mathsf{honest}\,(i) \supset$ $(\mathsf{linkPrev}\,(r,i) \wedge \mathsf{sentR}\,(r,i))$	valid	valid
Prop.(4)	$\mathsf{honest}\,(i) \supset$ $(\mathsf{linkPrev}\,(r,i) \wedge \mathsf{linkNext}\,(r,i)$ $\wedge\ \mathsf{sentToR}\,(r,i))$	T(1);T(2);T(3): $([u_1],m_1,d,-)$	valid

Table 1. Security Analysis of Protocol 1 and Protocol 2

4 Case Study

We discuss our analysis results of the two variants of S-BGP based on the techniques introduced in Section 3. In particular, we check whether Condition 6 holds. Our analysis is facilitated by Proverif [6], an automated cryptographic protocol verifier. We discuss the implication of the verification results, and how attack traces can be constructed based the results provided by Proverif. The source code can be found at `http://netdb.cis.upenn.edu/esorics12.tar.gz`.

For the protocols, we encode the data structures in Figure 1 and we encode the honest node as a process. For checking Condition 6, we decide to hard code the capabilities of the attacker as a process, as opposed to use Proverif's standard attacker model. The reason is that the protocols that we analyze are recursive in nature, and our attempts to directly use Proverif's attacker model cause non-termination. More concretely, we provide the attacker process with a well-formed updates of length 2 as described in Condition 6, and query whether any route update violating the property $F_\forall(r)$ can be constructed by the adversary. Since Condition 6 contains only finite number of possible routes that the adversary can generate, the analysis terminates.

Table 1 summaries the results of the security analysis of the two protocols presented in Section 2.1. 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 2.2) is of the form: $\boxminus(\forall u \forall r, \mathsf{honest}\,(u) \wedge \mathsf{send}\,(u,r) \supset \forall i, F_R(r))$ For each protocol, we list the attack route update – the route update that does not satisfy $\forall i, F_R(r)$– and the topology in which this route update is generated. For ease of presentation, we omit the signature list part from the route update, which can be straightforwardly deduced from other fields of the update. The topology is indexed by the number as presented in Figure 5 and 6.

Prop (1): The first property specifies the authenticity of incoming links. Protocol 1 allows the attacker to peel off both the last node in the route and the last signature in the signature list from an existing update, and attach itself to

the end of the route (with its signature). Since an honest node only checks the correspondence between nodes in the route and signature in the signature list, it will not detect the modification to the update.

This strategy does not apply to Protocol 2, as the destination of route updates is included in signatures. However, it is still feasible for two malicious nodes to collude with each other to make up a non-existent link in the update. An honest node can by no means detect such deception.

Prop (2): The second property specifies the authenticity of outgoing links. An attacker can come up with attack routes for both protocols.

There are two different attack scenarios for Protocol 1. The one is similar to the scenario in Prop (1), where the attacker reused a subset of the signatures in an existing update. There is no link between u_1 and m_1. In the second attack trace, there is no link between m_1 and u_3, however, this does not not directly correspond to a successful attack. The honest node u_3 will reject this route since m_1 is not its neighbor. Nonetheless, it does correspond to colluding attacks as discussed in Section 3.7: when the attacker colludes with another malicious node m_{new}, which does not have a direct link to m_1, but is connected to u_3, then $([d, u_1, u_2, m_1, m_{new}], u_3, d, \Sigma)$ is an attack trace, and will be accepted by u_3.

An analogous analysis applies in Protocol 2 as well.

Prop (3): In addition to the authenticity of the links included in a route update, we can also specify the authenticity of route announcements. In routing protocol, 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.

Intuitively, this property holds because an honest node only accepts routes from its direct neighbors and signs the route it sends out. By combining the analysis results here and the proofs in earlier sections, we have obtained a proof that all traces generated by the protocol satisfy this route authenticity property.

Prop (4): The final property captures the difference in terms of security guarantees between Protocol 1 and Protocol 2. The attack route for Protocol 1 coincides with the one for the property $\mathsf{linkNext}\,(r, i)$: there is no link between u_1 and m_1, and u_1 did not send the route announcement to m_1.

In comparison, this property holds for Protocol 2. An honest node only receives from its neighbors, and also only announces routes to its neighbors, and this information is protected by the digital signatures. However, this property does not prevent worm-hole attacks: two malicious nodes can collude to introduce non-existent links to the route update. If we assume that there is only one malicious node in the network, then Prop (4) implies that all links in a valid route announcement exist in the topology. Protocol 1 provides weaker security guarantees, and one attacker by itself can create one non-existent link.

5 Related Work

Prior work on formal network verification (e.g. [4, 12, 10]) have focused primarily on functional corrections of the protocols, but not the security properties. As compared to prior work on analyzing cryptographic protocols [9, 18, 13, 17, 11, 5, 3], secure routing protocols have to deal with arbitrary network topologies and that the security properties are recursive. Most model-checking techniques ineffective in proving these recursive security properties on routing protocols. Our proof techniques share some similarities with prior proof-based techniques [18, 17] for analyzing of cryptographic protocols. Our novelty lies in the reduction techniques and combining manual and automated proofs, in order to deal with proving security properties for any arbitrary network topology.

Recent work on verifying the security properties of wireless routing protocols for mobile networks [1, 2, 8] focus on similar route authenticity properties. 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 topology [8].

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 but also whether a node has announced a route in the past.

Second, the technique presented in [8] 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).

6 Conclusion

In this paper, we present 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 not only provide a formal account for the path authenticity properties, but also a formal proof that these properties hold on certain routing protocols. Beyond S-BGP, as our future work, we plan to apply our framework for analyzing recent clean-slate designs of secure Internet routing infrastructures (e.g. SCION [22] and ICING [16]).

References

1. Arnaud, M., Cortier, V., Delaune, S.: Modeling and verifying ad hoc routing protocols. In: Proceedings of the 2010 23rd IEEE Computer Security Foundations Symposium. CSF '10 (2010)
2. Arnaud, M., Cortier, V., Delaune, S.: Deciding security for protocols with recursive tests. In: Proceedings of the 23rd international conference on Automated deduction. CADE'11 (2011)
3. Bau, J., Mitchell, J.: A security evaluation of dnssec with nsec3. In: NDSS. The Internet Society (2010)
4. Bhargavan, K., Obradovic, D., Gunter, C.A.: Formal verification of standards for distance vector routing protocols. J. ACM 49(4) (2002)
5. Blanchet, B.: Automatic verification of correspondences for security protocols. J. Comput. Secur. 17(4) (Dec 2009)
6. Blanchet1, B., Smyth, B.: Proverif 1.86: Automatic cryptographic protocol verifier, user manual and tutorial, `http://www.proverif.ens.fr/manual.pdf`
7. Caesar, M., Rexford, J.: BGP Routing Policies in ISP Networks (2005)
8. Cortier, V., Degrieck, J., Delaune, S.: Analysing routing protocols: four nodes topologies are sufficient. In: Proceedings of the First Conference on Principles of Security and Trust. POST'12 (2012)
9. Datta, A., Derek, A., Mitchell, J.C., Roy, A.: Protocol Composition Logic (PCL). Electronic Notes in Theoretical Computer Science 172, 311–358 (2007)
10. Engler, D., Musuvathi, M.: Model-checking Large Network Protocol Implementations. In: USENIX Symposium on Networked Systems Design and Implementation (2004)
11. Escobar, S., Meadows, C., Meseguer, J.: A rewriting-based inference system for the nrl protocol analyzer: grammar generation. In: Proceedings of the 2005 ACM workshop on Formal methods in security engineering. FMSE '05 (2005)
12. Goodloe, A., Gunter, C.A., Stehr, M.O.: Formal prototyping in early stages of protocol design. In: Proc. ACM WITS'05 (2005)
13. He, C., Sundararajan, M., Datta, A., Derek, A., Mitchell, J.C.: A modular correctness proof of IEEE 802.11i and TLS. In: CCS '05: Proceedings of the 12th ACM Conference on Computer and Communications Security. pp. 2–15 (2005)
14. Jones, C.B.: Tentative steps toward a development method for interfering programs. ACM Trans. Program. Lang. Syst. 5(4) (Oct 1983)
15. Kent, S., Lynn, C., Mikkelson, J., Seo, K.: Secure border gateway protocol (s-bgp). IEEE Journal on Selected Areas in Communications 18, 103–116 (2000)
16. Naous, J., Walfish, M., Nicolosi, A., Mazieres, D., Miller, M., Seehra, A.: Verifying and enforcing network paths with icing. In: CoNEXT (2011)
17. Paulson: Proving properties of security protocols by induction. In: CSFW: Proceedings of The 10th Computer Security Foundations Workshop (1997)
18. Roy, A., Datta, A., Derek, A., Mitchell, J.C., Seifert, J.P.: Secrecy analysis in protocol composition logic. In: Proceedings of the 11th Asian computing science conference on Advances in computer science: secure software and related issues. pp. 197–213. ASIAN'06 (2007)
19. Secure BGP: http://www.ir.bbn.com/sbgp/
20. Wan, T., Kranakis, E., Oorschot, P.C.: Pretty secure bgp (psbgp). In: Proceeding of 12th Annual Network and Distributed System Security Symposium (NDSS05) (2005)

21. White, R.: Securing bgp through secure origin bgp (sobgp). The Internet Protocol Journal 6(3), 15–22 (2003)
22. Zhang, X., Hsiao, H.C., Hasker, G., Chan, H., Perrig, A., Andersen, D.G.: Scion: Scalability, control, and isolation on next-generation networks. In: Oakland (2011)

A Adversary model

$$\frac{t \in M(u)}{t \in \mathsf{analyze}(M(u))} \qquad \frac{(p, u_{recv}, d, \Sigma) \in \mathsf{analyze}(M(u))}{p \in \mathsf{analyze}(M(u))}$$

$$\frac{(p, u_{recv}, d, \Sigma) \in \mathsf{analyze}(M(u))}{d \in \mathsf{analyze}(M(u))} \qquad \frac{(p, u_{recv}, d, \Sigma) \in \mathsf{analyze}(M(u))}{\Sigma \in \mathsf{analyze}(M(u))}$$

$$\frac{\Sigma \in \mathsf{analyze}(M(u)) \qquad \sigma \in \Sigma}{\sigma \in \mathsf{analyze}(M(u))}$$

$$\frac{t \in \mathsf{analyze}(M(u))}{t \in \mathsf{syn}(M(u))} \qquad \frac{r \in \mathsf{syn}(M(u)) \qquad u_1 \in \mathsf{syn}(M(u)) \qquad u_2 \in \mathsf{syn}(M(u))}{f_{upd}(r, u_1, u_2) \in \mathsf{syn}(M(u))}$$

$$\frac{d \in \mathsf{syn}(M(u)) \qquad v \in \mathsf{syn}(M(u))}{f_{orig}(d, u, v) \in \mathsf{syn}(M(u))} \qquad \frac{\mathsf{sign}(p, \mathsf{sk}(u)) \in \mathsf{syn}(M(u))}{p \in \mathsf{syn}(M(u))}$$

$$\frac{p \in \mathsf{syn}(M(u)) \qquad u_{recv} \in \mathsf{syn}(M(u)) \qquad d \in \mathsf{syn}(M(u)) \qquad \Sigma \in \mathsf{syn}(M(u))}{(p, u_{recv}, d, \Sigma) \in \mathsf{syn}(M(u))}$$

$$\frac{\Sigma \in \mathsf{syn}(M(u)) \qquad \sigma \in \mathsf{syn}(M(u))}{\sigma :: \Sigma \in \mathsf{syn}(M(u))} \qquad \frac{p \in \mathsf{syn}(M(u)) \qquad u \in \mathsf{syn}(M(u))}{p@[u] \in \mathsf{syn}(M(u))}$$

$$\frac{p \in \mathsf{syn}(M(u)) \qquad \mathsf{sk}(v) \in \mathsf{syn}(M(u))}{\mathsf{sign}(p, \mathsf{sk}(v)) \in \mathsf{syn}(M(u))} \qquad \frac{t \in \mathsf{syn}(M(u))}{t \in \mathcal{K}(M(u))}$$

Fig. 7. Rules for Deriving Adversary Knowledge

B Additional Lemmas and Proofs for Section 3

B.1 Additional Definitions and Proofs for Section 3.1

We first define a projection $\widehat{\mathcal{T}}$ that extracts parts of the trace in $\mathcal{T}$ that contains only honest nodes' actions and states.

Definition 1 *$\widehat{\mathcal{T}}$ is a trace that only has honest nodes in the states, and the send actions performed by the honest node. Furthermore, ill-formed route updates are removed from the queues in honest nodes.*

The following lemma states that if there is a trace where the attacker did not send a route update that pass the check, then there is an equivalent trace where the attacker always send out a route update that pass the check.

Lemma 11 *For any trace $\mathcal{T} \in \mathcal{R}[(\mathcal{A}, G, \mathcal{P}, S_0)]$, there is a trace $\mathcal{T}' \in \mathcal{R}_{ck}[(\mathcal{A}, G, \mathcal{P}, S_0)]$ such that $\widehat{\mathcal{T}} = \widehat{\mathcal{T}'}$.*

Proof. We construct $\mathcal{T}'$ from $\mathcal{T}$ as follows. We examine the trace from the initial state. If we encounter a malicious node u sending a route update that is not well-formed, we extract all the valid signatures from it, and record on the side that u knows these signatures. We then remove this send action from the trace, and continue our examination.

When we encounter a malicious node v that sends a well-formed update, if all the signatures are derivable by v, then we continue. Otherwise, we must have removed a send action by some malicious node that contains the missing signatures. In this case, we can reconstruct a trace where all the updates are well-formed. Given the list of signatures $\sigma_1 \cdots \sigma_n$, we find the first signature σ_i that is not known by v, and the malicious node u that has σ_i, and send an update corresponds to signatures $\sigma_1 \cdots \sigma_{i-1}$ to u. Note that this signature is well-formed. Now u can send an update with signatures $\sigma_1, \cdots \sigma_i$ to v. We continue this process until no signature is missing from v. When $i = 1$, we do not need the first step.

This process continues until we reach the end of the trace. The resulting trace is the projection of the original trace, since we do not change what the honest nodes do, except remove ill-formed updates from its state, which is taken care of by the projection definition.

Next, we show that a trace $\mathcal{T}$ satisfies a formula if and only if the honest parts of the trace satisfies the formula.

Lemma 12 *$\mathcal{A}, G, \mathcal{T} \vDash \varphi$ iff $\mathcal{A}, G, \widehat{\mathcal{T}} \vDash \varphi$*

Proof (sketch): By induction on the structure of φ. The interesting cases are the cases for predicates. Predicate $\mathsf{link}\,(u, v)$ does not depend on the trace, so the conclusion holds. For predicate $\mathsf{sentR}\,(r, i)$, based on its semantics, it is always true when the i^{th} node is a malicious node, so it does not depend on the trace either. When it is is an honest node, $\mathcal{T}$ and $\widehat{\mathcal{T}}$ agree on actions and states for honest nodes, so the conclusion holds as well. □

B.2 Additional Definitions and Proofs for Section 3.6

Lemma 9 Let $M(u) = \{u, I, U\}$ and $\nexists \sigma \in I$, if $\forall r \in U$, $f_{ck}(r) = \mathsf{true}$, then $\forall r_0 \in \mathcal{K}(M(u))$ such that $f_{ck}(r_0)$,

$\boxed{G, (u, I, U) \vDash F_R(r, i)}$

$\mathcal{A}, G, (u, I, U) \vDash \mathsf{linkPrev}\,(r, i)$ iff $i = 1$ or $([\![i-1]\!]_r, [\![i]\!]_r) \in E$
$\mathcal{A}, G, (u, I, U) \vDash \mathsf{linkNext}\,(r, i)$ iff $([\![i]\!]_r, [\![i+1]\!]_r) \in E$
$\mathcal{A}, G, (u, I, U) \vDash \mathsf{sentR}\,((p, u_{recv}, d, \Sigma), i)$
iff u is an honest node implies
there exists a $r' \in (u, I, U)$, st.$r' = (p', u'_{recv}, d, \Sigma')$
p and p' has a subpath p'', and $|p''| = i$
$\mathcal{A}, G, (u, I, U) \vDash \mathsf{sentToR}\,((p, u_{recv}, d, \Sigma), i)$
iff u is an honest node implies
there exists a $r' \in U$, st.$r' = (p', u'_{recv}, d, \Sigma')$
either (1) p and p' has a subpath p'', and $|p''| = i + 1$
or (2) $i = |p|$ and $(p@[u_{recv}])$ is a subpath of p',
or (3) $r' = r$

Fig. 8. Local Semantics for Formulas

1. either $r_0 \in \mathcal{K}(\{u, I, \varnothing\})$
2. or $\exists r_1 \in U$, and $\exists r_2$ such that $r_2 = (p, u_r, d, \Sigma)$, $r_1 = (p@[v_1 \cdots v_j], \cdots)$ $r_0 = (p@[u_1 \cdots u_k], \cdots)$, where $\mathsf{sk}(u_i) \in I$ $(i \in [1, k])$

Proof (sketch): By induction on the length of r_0. The interesting case is when the last node v in r_0 is an honest node. In this case, r_0 contains a signature signed by v. By induction over the rules describing an attacker's ability, we know that it must be the case that this signature belong to some route update that the attack already knows. From there, and the fact that all the route updates in U are valid, we can derive that r_0 must be a prefix of an existing route update. □

Lemma 13 *There exists a route update $r \in \mathcal{K}(M(u))$ such that $f_{ck}(r) = \mathsf{true}$, and $\mathcal{A}, G, (u, I, U) \vDash F_R(r, i)$, iff there exists an update $r_1 \in U(u)$ and $\mathcal{A}, G, (u, I, \{r_1\}) \vDash F_R(r, i)$.*

Proof (sketch): Let r_1 be the route update that has the one that has the largest common path prefix as r.

The rest of the proof is done by induction on the structure of $F_R(r, i)$. The proofs for the cases where the base predicates that relate to links or whether a node is honest are trivial, since they do not depend on the trace. For action predicates, let us take the sentR predicate as an example. If sentR is true based on the set U, then it must share a common prefix with one of the update in U. Since r_1 has the largest common prefix as r, the predicate must be true based on r_1 as well.

The other direction is trivial. □