Privacy-Preserving Interdomain Verification

Anonymous Author(s)

Abstract

In an interdomain network, autonomous systems (ASes) often establish peering agreements, so that one AS (agreement consumer) can influence the routing policies of the other AS (agreement provider). Peering agreements are implemented in the BGP configuration of the agreement provider. It is crucial to verify their implementation because one error can lead to disastrous consequences. However, the fundamental challenge for peering agreement verification is how to preserve the *privacy* of the parties involved in the agreement. To this end, this paper presents IVeri, the first privacy-preserving interdomain agreement verification system. IVeri models the interdomain agreement verification problem as a SAT formula, and develops a novel, efficient, privacy-serving SAT solver, which uses oblivious shuffling and garbled circuits as the key building blocks to let the agreement consumer and provider collaboratively verify the implementation of interdomain peering agreements without exposing their private information. A prototype of IVeri is implemented and evaluated extensively. Results show that IVeri achieves accurate, privacy-preserving interdomain agreement verification with reasonable overhead.

1 Introduction

In network research the term *autonomous system* (AS, plural: ASes) describes a network that is managed by a single administrative body (*e.g.*, universities, companies or Internet service providers). An interdomain network (*e.g.*, the Internet [19] and the Large Hadron Collider science network [30]) connects multiple ASes and exchanges traffic among them. The de facto interdomain routing protocol is the well known Border Gateway Protocol [45]. ASes use BGP to exchange routing information. In addition, BGP also allows each AS to independently apply its local routing policies to select and advertise interdomain routes.

Due to economic or collaborative incentives, ASes often establish *peering agreements*. In peering agreements, which are implemented in the configurations of BGP routers, ASes influence each other's routing policies. The correct implementation of peering agreements is crucial for ASes to realize their network management goals, *e.g.*, defense against distributed denial-of-service (DDoS) attacks, inbound and outbound traffic engineering, and fulfillment of regulations on data traffic traversal.

Nevertheless, technology news have been repeatedly reporting about peering agreements not being implemented correctly, leading to disastrous results [2, 4–6, 36]. For example, due to an error in a BGP configuration in 2018, Google lost control of several millions of IP addresses for more than one hour and rerouted the internet traffic through China [3].

Every statement in the peering agreement consists of one AS (also called agreement consumer) influencing the routing policies of the other AS (also called agreement provider). We, therefore, only consider a peering agreement between two ASes. Agreements can be either formal (in the form of a written contract) or informal (in the form of a public BGP community guide for an agreement consumer to influence the provider's routing policies). Based on the agreement, the agreement provider adds additional policies into its BGP configuration file implementing that agreement. Consider an interdomain network presented in Figure 1. An illustration of a peering agreement is when, for example, the ASes A and B agree to send all A's traffic towards F through D. The AS A is the agreement consumer, and B is the agreement provider – this means that the AS B adds additional rules to its BGP configuration implementing that agreement. To verify the peering agreement means that the agreement consumer is able to verify that the provider's BGP configuration indeed implements what was settled in the agreement.

Verification of an interdomain peering agreement is still an open, non-trivial problem, despite the fact that many recent tools are focused on finding configuration errors in routing protocols [9, 10, 16, 17, 20, 25, 52, 53]. To verify an implementation of a peering agreement using state-of-the-art network configuration verification tools would require that the involved ASes explicitly expose their private information (*i.e.*, configurations and the agreements-to-verify, due to their close ties to the ASes' internal operation policies), causing leakage of potentially sensitive information. Thus, the fundamental challenge for the verification problem for interdomain peering agreements is preserving the *privacy* of the parties involved in the agreement.

This paper presents IVeri, the first privacy-preserving interdomain agreement verification system. We leverage and expand the formalism from recent network verification tools and use it to model the interdomain peering agreement. In this formalism, the peering agreement is written as an SMT formula. In addition, using the same tools and formalism, the agreement provider extracts the BGP configurations into another SMT formula. This way the verification problem is reduced to checking the satisfiability of the formula. However, in such settings the main problem of sensitive information leakage would still be present. Therefore, our second contribution of this work is a novel, efficient, privacy-preserving SAT solver.

Submitted to PLDI'20, June, 2020, London, United Kingdom
2020.

Note that the SMT formulas derived before, describing the agreement policies and their implementation, do not contain quantifiers and all the variables are bounded, so they can easily be encoded into SAT formulas. To use the same language, the agreement producer sends to the agreement consumer a mapping of SMT variables to Boolean variables. For example: variable x is encoded with Boolean variable X_1, X_2 and X_3. Note that the agreement producer only sends the mapping of the variables, and does not send any additional constraints on variables. This mapping is needed so that every first-order variable in the original formalism is described with the same Boolean variables by both parties. We denote the derived SAT formulas with F_A (a formula describing the negation of the agreement policies from the perspective of the agreement consumer) and F_B (a formula describing the agreement implementation extracted from the BGP configuration of the agreement provider).

Our modeling reduces the verification problem of an interdomain peering agreement to checking if the formula $F_B \wedge F_A$ is unsatisfiable. Note that we encoded F_A as the negation of the actual agreement formula for simplicity of presentation. To this end, we define and develop a novel, efficient, privacy-preserving SAT solver. The basic idea of this solver consists of (1) an oblivious shuffling algorithm [24] that enables two parties to shuffle an encrypted version of the verification formula, and further ensures that the non-encrypted configuration and agreement formulas are always held private by the respective parties, and (2) the standard DPLL algorithm encoded with inexpensive garbled circuits [57], denoted by GC-DPLL. The solver determines the satisfiability of the verification formula contributed by the agreement consumer and provider without loss of their privacy on their contribution.

Contributions: This paper makes the following contributions:

- it systematically studies the important, real problem of interdomain peering agreement verification, and presents IVeri, a privacy-preserving interdomain agreement verification system, which is, to the best of our knowledge, the first such system that requires minimal deployment efforts;
- IVeri leverages and expands the modeling technique in recent network verification tools to model the interdomain peering agreement verification problem as a SAT problem;
- IVeri designs a novel, efficient privacy-preserving SAT solver that determines the satisfiability of the verification formula without letting the agreement consumer or provider sensitive information. In addition to interdomain verification, this SAT solver also has other application prospectives such as multi-domain resource orchestration via constraint programming [54];
- A prototype of IVeri is implemented and evaluated with extensive experiments. Results show that IVeri achieves accurate, privacy-preserving interdomain agreement verification with reasonable overhead.

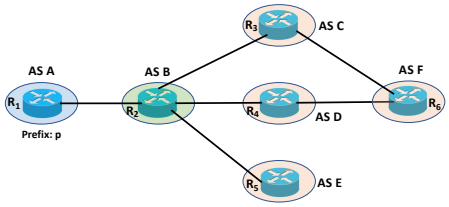


Figure 1. An example of interdomain network to illustrate BGP and interdomain peering agreements.

2 Background, Motivation and Challenge

This section provides a brief background on BGP and interdomain peering agreements, demonstrates the importance of interdomain peering agreement verification, and elaborates on its fundamental challenges.

2.1 Background

BGP [45] is the main interdomain routing protocol interconnecting ASes. Specifically, each AS assigns some of its routers as BGP border routers, which are connected to BGP border routers in neighboring ASes. BGP routers within the same AS are connected via internal intradomain routing protocols of the AS. Figure 1 shows an interdomain network with 6 ASes, for simplicity of the presentation assume that each of them has one BGP border router. Two ASes whose BGP routers are connected to each other are called BGP peers, *e.g.*, A and B are BGP peers.

Route announcements: A prefix represents a continuous sets of IP addresses. For example, the prefix 128.16.4.0/24 represents all IP addresses ranging from 128.16.4.0 to 128.16.4.255, or in other words, all the IP addresses sharing with the IP address 128.16.4.0 the first 24 bits. BGP peers exchange routing information on how to reach destination IP prefixes through so called *route announcements*. Abstractly, each route announcement carries a destination prefix and a sequence of ASes to traverse to reach the destination prefix, which we call an *AS path*. In Figure 1, suppose that an AS A has prefix p. Its router R_1 sends a route announcement $(p, [A])$ to router R_2 at B, indicating that R_1 has a route to p via the AS path A. When R_2 selects this route, it will add the AS B in the AS path, and send a route announcement $(p, [B, A]$ to routers R_3, R_4, and R_5, indicating that it can reach p via the AS path $B \rightarrow A$. Routers R_3 and R_4 can then propagate their received route announcements to R_6, *i.e.*, R_3 sends $(p, [C, B, A])$ to R_6 and R_4 sends $(p, [D, B, A]$ to R_6. This means, that router R_6 can select which route to prefix p it wants to use, $[C, B, A]$ or $[D, B, A]$.

In addition to the prefix and the AS path, a route announcement can also carry other attributes, such as *origin*,

which is the origin AS of the destination prefix, or *community tags*, which are numerical values whose semantics are predefined between two BGP peers via offline communication. For example, R_2 can send a route announcement $(p, [B, A], comtag : 10 : 30)$ to R_3, which includes a community tag attribute of value $10 : 30$.

BGP allows each AS to independently decide its own routing policies, which are implemented in the configurations of its BGP routers.

Route selection policy: Given an AS, its route selection policy specifies when a BGP router receives multiple route announcements to the same prefix, which route should be selected. Given a destination prefix, a route selection policy assigns a *local preference* value to each received route announcement to this prefix, and selects the one with the highest local preference. If there are more than one route having the same highest local preference, the selection policy breaks ties by comparing other attributes of the received route announcements, such as the length of the AS path, in an lexicographic order, to select only route to this prefix. To continue with our example, assume the route selection policy of AS F is to always prefer the route whose AS path starts with C over the one with D. This can be realized via the following configuration snippet of the Cisco IOS system:

```
ip as-path access-list 1 permit ^C_
ip as-path access-list 2 permit ^D_
match as-path 1
set local-preference 70
match as-path 2
set local-preference 50
```

It specifies that R_6 will set a local preference of 70 for any route announcement whose AS path starts with C, and a local preference of 50 for any route announcement whose AS path starts with D.

Route export policy: A route export policy specifies whether a BGP border router will propagate a route by sending a route announcement to its BGP peers. As an illustration, D decides that its export policy is to not announce any route that reaches prefix p to F. Suppose that prefix p is 172.218.8.14/24 and the IP address of router R_6 is 65.124.208.93. Then this policy can be realized via the following configuration snippet:

```
access-list 10 deny 172.217.8.14 0.0.0.255
neighbor 65.124.208.93 distribute-list 10 out
```

It specifies that R_4 will not announce any route toward the prefix 172.217.8.0/24 (p) to the BGP router whose IP address is 65.124.208.93 (R_6). With this export policy of D, F will not receive $(p, [D, B, A])$.

Encoding BGP configurations as SAT formulas: Recent network configuration verification tools [9, 52, 53] model router configurations as SMT formulas. Consider the previous configuration snippets of router R_4. It can be expressed as the following SMT formula, by using the Minesweeper [9] notation:

$$FBM(172.217.8.0, best_{BGP}.prefix, 24) \wedge \mathrm{R_i.IP} = 65.124.208.93 \Rightarrow \mathrm{out}_{\mathrm{R_4R_i}}.\mathrm{valid} = \mathrm{false}. \quad (1)$$

where $best_{BGP}$ is the selected route at R_4 and the function FBM tests the equality of the first 24 bits of $\mathrm{best_{BGP}.prefix}$ and 172.217.8.0. We convert all BGP configurations into such SMT formulas. All the variables of these formulas are finite-bounded, so those formulas can easily be encoded as SAT formulas.

2.2 Peering Agreements

Two ASes, who are BGP peers, often reach peering agreements to control the exchange of traffic. Peering agreements are implemented as part of the route selection/export polices in the BGP configurations of the agreement provider. Their correct implementation is essential for ASes to realize their network management goals, *e.g.*, defense against DDoS attacks, inbound and outbound traffic engineering, and fulfillment of regulations on data traffic traversal.

Example (DDoS attack): Consider the interdomain network given in Figure 1, and assume that AS E launches a DDoS attack to A, *i.e.*, keeps sending high volumes of malicious traffic to prefix p in A within a short time period. When A detects a DDoS attack, and suspects that E is the source of this attack, A can defend by reaching a *selective-export* agreement with B: *B will not export any route announced by A to E*. If such an agreement is implemented correctly by B, B will send a new route announcement $(p, [])$ to E, also called a withdraw announcement, indicating that B cannot reach p. As a result, E cannot continue the attack because it has no route to reach p, and A can observe the stop of the attack.

However, if B does not implement this agreement correctly, E will continue having a route $B \rightarrow A$ to reach prefix p, and use this route for the attack. As such, A will observe that the attack does not stop, and shift its suspicion to another AS, *e.g.*, C, and reaches another selective-export agreement with B to not export any route announced by A to C. If this new agreement is implemented by B correctly this time, not only does A still suffer from the DDoS attack, the normal traffic from C to A is also completely blocked, exacerbating the damage of this attack.

Encoding peering agreements as SAT formulas: The example above demonstrates the importance of verifying the correct implementation of peering agreements. While the work on verification of peering agreements was focused on route announcements [61], the output of peering agreements, we encode peering agreements as SMT/SAT formulas. Consider the selective-export agreement from the example given before, it can be modeled as the following SMT formula in Minesweeper syntax:

$$\forall \mathrm{i} = 3, 4, 5.\ \mathrm{FBM}(172.217.8.0, \mathrm{R_2.best_{BGP}.prefix}, 24) \wedge \mathrm{R_i.AS} = \mathrm{E} \Rightarrow out_{R_2R_i}.valid = false. \quad (2)$$

For the same reasons as when encoding BGP configurations, these SMT formulas can easily be transformed to SAT formulas.

2.3 Challenges

Recall that a basic peering agreement consists of two parties: one AS (also called agreement consumer) is influencing the routing policies of the other AS (also called agreement provider). One might think that by encoding a peering agreement as a SAT formula, we can easily verify it: a trusted third party collects formulas from both sides of the agreement, and feeds them to a SAT solver. However, this is not the case.

The reason, and the fundamental challenge of verifying interdomain peering agreements, is *privacy*. On one hand, given an agreement provider, its implementation of peering agreements (*i.e.*, the BGP router configuration files) is private for operational security and commercial reasons [7]. Although some work has shown it is possible to infer some aspects of ASes' BGP configurations [18], the inference results have limited accuracy and cannot be utilized for peering agreement verification.

On the other hand, for an agreement consumer, the peering agreements are also considered private. One may find this argument a little odd at a first glance, however, it is justified. Many agreements (*e.g.*, selective export, set local preference and AS path prepending) are invoked "on-demand" to achieve certain network management goals (*e.g.*, inbound traffic engineering). When to invoke which agreements is related to the operation policy of agreement consumers. As such, revealing which agreements to verify risk exposing its operation policy. For example, consider the not-export-to-E agreement between A and B. Assume additionally that it is on-demand requiring A attaching community tags in its route announcements to B. If A reveals to B that it wants to verify that B implements this agreement correctly, but does not attach any community tags in its route announcements, B may deduce that either A does not suspect E as a DDoS attack, or A and E reach additional commercial collaboration that allows E to send traffic to A, both of which are private to A.

From this perspective, verifying interdomain peering agreements, while allowing the agreement consumer to keep the agreement-to-verify private, may be considered unsafe for the agreement provider. By verifying arbitrary requirements, such privacy may allow the consumer to learn different aspects of the configuration of the provider that they should not know about. As it will be shown later, the design of IVeri mitigates the impact of such snooping-around attempts by restricting the programming abstraction exposed to the agreement consumer.

3 Overview

IVeri operates between the agreement consumer and the agreement provider (Figure 2). This section gives an overview

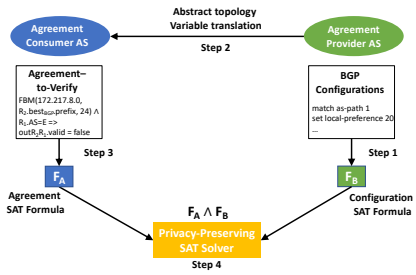


Figure 2. The basic architecture and workflow of IVeri.

of the key design points of IVeri, a SAT model of interdomain peering agreement verification and a novel, efficient, privacy-preserving SAT solver, and its basic workflow.

3.1 Key Design Points

A SAT model of interdomain peering agreement verification (Section 4): IVeri models the problem of verifying interdomain peering agreements as an SAT problem. To this end, IVeri leverages the modeling technique of recent network configuration verification tools (*e.g.*, Minesweeper [9], FSR [52], and Bagpipe [53]) to model the BGP configurations of the agreement provider, but goes beyond their focus on intradomain network properties (*e.g.*, reachability and blackhole freeness) and safety properties of interdomain network (*e.g.*, stability), to develop new models for interdomain peering agreements.

Specifically, in IVeri, the agreement provider extracts its BGP configurations into an SMT formula. It then uses bit blasting and Tseytin transformation to transform the formula to a SAT formula in Conjunctive Normal Form (CNF). This SAT formula is called the *configuration SAT formula*.

Next, the agreement provider exposes an abstract topology of a full-mesh of the provider's BGP routers and their connections to other BGP peers, and a SMT/SAT variable translation list. The administrator of the agreement consumer uses this information to express the agreement-to-verify as a SMT formula. The negation of this formula is transformed to a SAT formula in CNF. This SAT formula is called the *agreement SAT formula*.

With these two SAT formulas, the verification of interdomain peering agreement is then modeled as a SAT problem to decide the satisfiability of the *verification SAT formula*, *i.e.*, the conjunction of the configuration SAT formula and the agreement SAT formula. If this formula is unsatisfiable, the agreement is correctly implemented at the agreement provider. Otherwise, it is either not correctly implemented, or there is an error in the agreement itself.

A privacy-preserving SAT solver (Section 5): Although many SAT solvers have been developed to solve large SAT problems efficiently [13, 21, 37, 39, 60], they cannot be used to decide the satisfiability of the verification SAT formula due to the fundamental privacy challenge in interdomain

peering agreement verification. Specifically, the configuration formula and the agreement formula are private to the agreement provider and the agreement consumer, respectively. As such, exposing them to a trusted third party is not a satisfactory solution. To this end, IVeri introduces a novel, efficient, privacy-preserving solver that decides the satisfiability of the verification formula, such that neither the agreement provider nor the agreement consumer can infer the private formula of the other party.

The basic idea of the privacy-preserving SAT solver is leverages garbled circuit [58], a generic cryptography protocol in secure multi-party computation [15] to encode the SAT solving techniques developed in the DPLL algorithm [13] (*e.g.*, backtrack-searching and unit literal search) in a privacy-preserving, yet efficient manner. Specifically, it addresses the tradeoff between the high overhead of hiding the search history of private inputs (*i.e.*, Boolean variables of SAT formulas held private by agreement provider and consumer) in the garbled circuit and the severe privacy leakage of exposing such an search history using a simple, yet elegant idea: *instead of hiding the search history with high overhead, expose a disguised search history that no party can recover independently.*

Specifically, the solver consists of two key steps: (1) permute the order of variables in the verification formula, and share the permutation as pieces of secrets between agreement provider and consumer, and (2) encode the DPLL algorithm in a garbled circuit not hiding the search history of permuted variables. In this way, the solver avoids the high overhead of hiding the search history. In addition, although the search history of permuted Boolean variables is exposed to both parties, neither the agreement provider nor consumer can recover the search history of the original Boolean variables because the permutation is a secret neither of them holds independently.

A systematic study on the correctness, efficiency and preservation of privacy of this SAT solver through rigorous analysis (Section 5.4) and extensive evaluation (Section 6) is conducted in this paper. Not only can this solver be used for interdomain peering agreement verification, it also has extensive application prospectives such as multi-domain resource orchestration via constraint programming [54].

3.2 Basic Workflow

Before presenting the design details of IVeri, the basic workflow of IVeri to achieve privacy-preserving interdomain peering agreement verification (Figure 2) is summarized as follows:

- **Step 1**: The agreement provider transforms the configuration of its BGP routers to a configuration SAT formula $\mathbf{F_B}$;
- **Step 2**: The agreement provider exposes an abstract topology, and a list of SMT/SAT variable translation to the agreement consumer;
- **Step 3**: With the received topology and variable translation list, the network administrator of the agreement consumer specifies the agreement-to-verify in a SMT formula, whose negation is transformed to an agreement SAT formula $\mathbf{F_A}$;
- **Step 4**: The agreement provider and consumer execute the privacy-preserving SAT solver together to determine the satisfiability of the verification formula. If it is unsatisfiable, the agreement is implemented correctly. Otherwise, it is implemented incorrectly.

4 Interdomain Peering Agreement Verification Model

This section presents details on how IVeri models the interdomain peering agreements verification problem as a SAT model. Specifically, it first describes the model of BGP configurations of the agreement provider, and then specifies how the agreement consumer models the peering agreement-to-verify.

4.1 BGP Configuration Model

Unlike recent network configuration verification tools [9, 16, 20] that focus on the modeling of the interaction between multiple routing protocols within the same AS, IVeri focuses on modeling the BGP configurations of the agreement provider, where peering agreements are implemented. IVeri assumes that any two BGP routers belonging to the agreement provider can reach each other via the provider's intradomain routing protocols. The verification of this assumption can be accomplished with existing intradomain network verification tools (*e.g.*, [9, 16, 20, 27, 29, 56]), and is out of the scope of IVeri.

Specifically, IVeri first leverages the modeling technique of Minesweeper [9], a state-of-the-art intradomain network verification tool, to model the BGP configurations of the agreement provider as an SMT formula. Because all the variables of this formula are finite-bounded, it is encoded in a SAT formula in CNF (*i.e.*, using bit blasting and Tseytin transformation), which is called the *configuration SAT formula*.

For the purpose of presentation completeness, the following gives a short description on how IVeri models BGP configurations as an SMT model. Readers can refer to [9] for more details of this modeling technique.

Consider the BGP router R in Figure 3, where it is connected to three BGP routers R_1, R_2 and R_3 of other ASes. Its BGP configurations are conceptually categorized into three phases: route announcement processing, route selection and route export. The first two phases realize its route selection policy, and the third phase realizes its route export policy. The route announcement processing refers to the operations

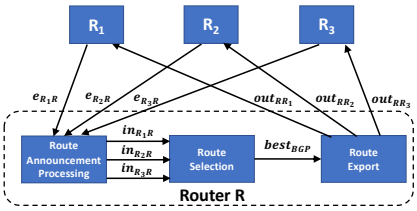
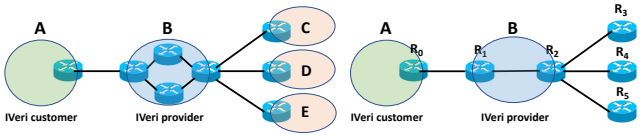


Figure 3. An abstract model of a BGP router R that is connected to three BGP peers R_1, R_2 and R_3.



(a) The physical topology of B. **(b)** The abstract topology that B exposes to A.

Figure 4. An example where the IVeri provider B provides IVeri consumer A an abstract topology.

of a BGP router dropping a received route announcement or modifying its attributes (*e.g.*, local preference). In Figure 3, arriving route announcements from different peers are represented as e_{R_1R}, e_{R_2R} and e_{R_3R}, and the corresponding route announcements after the processing are represented as variables in_{R_1R}, in_{R_2R} and in_{R_3R}. An example of route announcement processing configuration at R is the following configuration snippet:

```
ip community-list standard set_local_pref permit 20:30
match community set_local_pref
set local-preference 30
```

It specifies that when the router receives a route announcement, it needs to check the community tag in this announcement. If the value of the first two bytes (representing an AS number) of this tag equals 20, and the value of the second two bytes (representing the tag value) equals 30, the router should set a local preference of 30 for this route. This configuration is transformed to the following SMT formula:

$$\begin{aligned} &\forall i = 1, 2, 3.\ e_{R_iR}.\text{comAS} = 20 \wedge e_{R_iR}.\text{comVal} = 30 \\ &\quad \Rightarrow in_{R_iR}.\text{valid} = \text{true},\ in_{R_iR}.\text{lp} = 30, \\ &\quad \wedge in_{R_iR}.\text{prefix} = e_{R_iR}.\text{prefix} \wedge \ldots, \end{aligned} \tag{3}$$

where the ellipsis denotes that the value assignment of the other attributes of in_{R_iR} is the same as the value assignment of those attributes in e_{R_iR}.

Given a destination prefix and all received routes from its BGP peers, the route selection configuration specifies how a BGP router selects the best route toward this prefix. Given a destination prefix, a route selection configuration is modeled using a binary relation $\preceq$ between two routes, where $in_1 \preceq in_2$ means route in_1 is at least as preferred as route in_2, and the following SMT formula:

$$(\forall i = 1, 2, 3.\ \text{best}_{\text{BGP}} \preceq in_{R_iR}) \wedge (\exists_{i=1,2,3} \text{best}_{\text{BGP}} = in_{R_iR}), \tag{4}$$

where best_{BGP} is the selected route.

The route export configuration specifies, after a BGP router selects a best route, which BGP peers it should send a route announcement about this best route to, and the values of the attributes that should be set in this route announcement. How the route export configuration can be transformed to an SMT formula was illustrated in the example at the end of Section 2.1.

After getting the SMT formula of BGP configurations, the agreement provider further transforms it to the configuration SAT formula. Such transformation is standard and is omitted due to limited space.

4.2 Peering Agreement Model

IVeri develops new models for interdomain peering agreements. Specifically, in IVeri, the agreement provider exposes to the agreement consumer an abstract topology, which includes full-mesh logical connections between its BGP routers and their connections to other BGP peers, and a SMT/SAT variable translation list. The network administrator uses the received information to specify interdomain peering agreements he/she wants to verify as a SMT formula, whose *negation* is further transformed to a SAT formula in CNF (*i.e.*, *the agreement SAT formula*).

Consider the example in Figure 4a. The abstract topology in the programming abstraction exposed by the agreement provider B to the agreement consumer topology includes a connection between R_1 and R_2 (*i.e.*, the two border routers of B), the connection between R_1 to R_0 (*i.e.*, the BGP router of A), and the connections from R_2 to R_3, R_4, and R_5 (*i.e.*, the BGP routers of ASes C, D, and E).

Two reasons lead to the design of having the provider expose such information. First, the abstract topology provides sufficient topology information (*i.e.*, all BGP connections of the agreement provider) for the consumer to specify peering agreements to verify. Second, the exposed list of SMT/SAT variable translation allows the agreement provider to decide the scope of peering agreements that the agreement consumer can verify, and ensures that the variable semantics are consistent between the configuration SAT formula and the agreement SAT formula.

Note that the topology information provided by the abstract topology is sufficient, not "minimal". A simple example is that when all BGP routers of the agreement provider have the same configuration, the abstract topology can be simplified to have one virtual router, representing the agreement provider, connected to all the BGP peers of the agreement provider. The systematic investigation of the minimal abstract topology for interdomain peering agreement verification is left as future work.

When the agreement consumer receives the abstract topology and the variable translation list, its network administrator can specify peering agreements to be verified as SMT formulas. In addition to the selective-export agreement whose SMT formula has been illustrated in Section 2.2, this subsection introduces another representative peering agreement and illustrates its model on the abstraction in Figure 4. The models of another three representative peering agreements are provided in Appendix A.

Set local preference: Consider the case where AS A wants to verify a peering agreement that if A sets the community tag to $10:50$ in a route announcement, B will set the local preference of such a route to be 50. The administrator of AS A can express this agreement as:

$$\begin{aligned} & e_{R_0R_1}.\text{comAS} = 10 \wedge e_{R_0R_1}.\text{comVal} = 50 \\ & \Rightarrow in_{R_0R_1}.\text{valid} = \text{true} \wedge in_{R_0R_1}.\text{lp} = 50, \\ & \wedge in_{R_0R_1}.\text{prefix} = e_{R_0R_1}.\text{prefix} \wedge \ldots, \end{aligned} \tag{5}$$

where, again, the ellipsis denotes that the value assignment of the other attributes of $in_{R_0R_1}$ is the same as the value assignment of those attributes in $e_{R_0R_1}$.

5 Privacy-Preserving SAT Solver

This section presents details on the novel, efficient, privacy-preserving SAT solver in IVeri to verify interdomain peering agreements in a privacy-preserving manner. Specifically, we first formally define the problem of privacy-preserving interdomain peering agreement verification. Before diving into technical details, we provide a short review on the theoretical foundations this solver is built upon. At the end, rigorous analysis on the correctness, efficiency and privacy-preservingness of the designed solver is presented.

5.1 Problem Formulation

The following conventions are followed to differentiate scalar, vector and matrix variables. We use v to represent a scalar and $\mathbf{v} = \{v_1, v_2, \ldots, v_{|\mathbf{v}|}\}$ to represent a row vector of size $|\mathbf{v}|$. We use $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{|\mathbf{V}|}\}$, with $|\mathbf{v}_i| = |\mathbf{v}_j|$, for all $i \leq j \leq |\mathbf{V}|$ to represent a matrix of $|\mathbf{V}|$ rows. The element on the i-th row and j-th column of $\mathbf{V}$ is denoted by $\mathbf{V}_{ij}$. Unless explicitly specified, operators on vectors or matrices indicate component-wise operations.

A permutation π is a bijective function: $\{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$, for $n \in \mathbb{N}$. A *row permutation* of a matrix $\mathbf{V}$ is represented as $\{\mathbf{v}_{\pi^{-1}(1)}, \mathbf{v}_{\pi^{-1}(2)}, \ldots, \mathbf{v}_{\pi^{-1}(|\mathbf{V}|)}\}$ and written as $\pi(\mathbf{V})$.

A SAT formula in CNF of m clauses (*i.e.*, $c_1 \wedge c_2 \wedge, \ldots, \wedge c_m$) over n Boolean variables $x_1, x_2, \ldots, x_n$, is represented by an n-by-m matrix F, where each element $\mathbf{F}_{ij}$ of $\mathbf{F}$, consists of two bits. The first bit, denoted as $\mathbf{F}_{ij}.O$, is 1 if variable x_i occurs in clause c_j, and is 0 otherwise. The second bit, denoted as $\mathbf{F}_{ij}.P$, is 1 if and only if the variable x_i occurs as a positive literal (*i.e.*, x_i) in clause c_j.

Given a pair of peering ASes, A and B denote the agreement consumer and the agreement provider, respectively. The agreement SAT formula generated by A is represented as $\mathbf{F}^A$, and the configuration SAT formula generated by the provider B is represented as $\mathbf{F}^B$. As such, the privacy-preserving interdomain peering agreement verification problem is formally defined as follows:

Problem 1 (Privacy-Preserving Interdomain Peering Agreement Verification Problem)**.** *Given an agreement consumer A with associated agreement SAT formula $\mathbf{F}^A$ and an agreement provider B with an associated configuration SAT formula $\mathbf{F}^B$, the two parties A and B should decide if the formula $\mathbf{F}^A \wedge \mathbf{F}^B$ is satisfiable in a way such that A (respectively B) does not know what $\mathbf{F}^B$ (respectively $\mathbf{F}^A$) is.*

Security model: This paper assumes a *semi-honest* security model [57], *i.e.*, the agreement customer and the agreement provider will not deviate from the workflow specified in IVeri, but merely try to gather information during its execution [44]. This is sufficient for multiple scenarios of interdomain routing, including commercial Internet [23], collaboration science networks where member networks share resources to collaboratively conduct common tasks such as data transfers, storage and analytics [30], and military coalition networks [38].

5.2 Preliminaries

To address Problem 1, IVeri designs a privacy-preserving SAT solver. For clarity of presentation, this subsection presents a review of the theoretical foundations this solver is built upon.

DPLL algorithm: IVeri's Privacy-preserving SAT solver is based on the classic DPLL algorithm, as many modern SAT solvers do [14, 21, 37, 39, 48, 60]. In essence, DPLL is a backtracking search algorithm. Given a SAT formula in CNF represented by a matrix $\mathbf{F}$, the algorithm runs by choosing a Boolean variable x_i, assigning a truth value to it, simplifying the formula by removing all clauses that are satisfied by the truth value assignment of x_i as well as removing all literals involving the variable x_i that are false under this assignment, and then recursively checking if the simplified formula is satisfiable. If so, $\mathbf{F}$ is satisfiable. Otherwise, the algorithm uses the same recursive check by assigning the opposite truth value to x_i. To improve the efficiency of this backtracking search process, many optimizations are introduced [21, 37, 39, 48, 60]. One of the first optimizations introduced into the basic backtracking search algorithm that is the core of the DPLL algorithm, is *unit literal search and resolution*, which attempts to find a unit clause that contains only one unassigned variable, and assigns a truth value to this variable to satisfy this clause, before attempting anything else. This optimization substantially reduces the search space, leading to significant improvement in search efficiency.

Another optimization commonly associated with DPLL is *pure literal elimination.*

Secure multi-party computation (SMPC): SMPC refers to a set of frameworks for multiple parties to jointly compute a function over their inputs while keeping those inputs private [57]. Although there is a rich set of SMPC related techniques, this paper focuses on the following two that are adopted in the proposed privacy-preserving SAT solver.

- **Garbled circuits**: this is a cryptographic protocol, introduced in [15], that allows two mistrusting parties to jointly evaluate any function that can be expressed as a Boolean circuit over their private inputs. An example application of this protocol is the Millionaires' Problem, where the goal is for two parties to decide who has more money without exposing how much money they each have [58]. The output of a garbled circuit can be revealed to both parties, only one specific party, or no party but be taken as inputs of other garbled circuits. We refer the interested reader to [32] for more details on garbled circuits.
- **Oblivious shuffling**: this technique [24] enable two parties, collaboratively decide a permutation π over a vector of cipher texts $\Phi = \{\phi_1, \phi_2, \ldots, \phi_{|\mathbf{v}|}\}$) such that ϕ_i is the cipher text of some v_i. The protocol provides that (1) the output vector of this protocol is the vector $\{\phi_{\pi(1)}, \phi_{\pi(2)}, \ldots, \phi_{\pi(|\mathbf{v}|)}\}$ where $\phi_{\pi(i)}$ is the cipher texts of $v_{\pi(i)}$; (2) the permutation π is unknown to both parties unless they collaborate; (3) the plain text $v_1, \cdots, v_n$ is never revealed to any party during the protocol.

5.3 Privacy-Preserving SAT Solver

This subsection first describes the basic idea behind the privacy-preserving SAT solver, followed by the technical details of its key components.

Limitations of garbled circuits: A strawman solution to Problem 1 is to construct a garbled circuit of the entire DPLL algorithm, because it can be expressed as a Boolean circuit. However, the resulting circuit can blow up, especially for the algorithms such as DPLL that use a secret index (the index of unit literals) as a lookup to an array, for hiding which element was accessed. One may think of a second solution, which is to construct a garbled circuit to only implement the computation operations in DPLL, but leave the access pattern of private inputs of **A** and **B** outside the garbled circuit. In other words, the history of backtracking-search is exposed. However, the privacy leak of this solution is substantial and can lead to the complete exposure of $\mathbf{F^A}$ and $\mathbf{F^B}$.

Basic idea: The proposed SAT solver addresses this tradeoff between the high overhead of accessing private inputs in the garbled circuit and the severe privacy leakage of accessing private inputs outside the garbled circuit with a simple, yet elegant idea. It first encrypts and permutes the private inputs of **A** and **B** through an oblivious shuffling algorithm, and then constructs a garbled circuit of the computation operations in DPLL, but leave the access of the private inputs of **A** and **B** outside the garbled circuit. As a result, the high overhead of accessing private inputs in the garbled circuit is avoided. In addition, by the permutation only the structure of the backtracking-search steps is exposed. More precisely, the history of the search is exposed, but it is a history of permuted Boolean literals and thus neither of the two parties knows which literal of original formula is being accessed. It is more difficult for **A** and **B** to infer $\mathbf{F^B}$ and $\mathbf{F^A}$ by observing this permuted search pattern. The key steps of the privacy-preserving SAT solver are presented as follows:

Step 0: Initialization: The solver starts by zero-padding and aligning $\mathbf{F^A}$ and $\mathbf{F}^B$. This is because during the modeling phase, **A** and **B** each independently introduce auxiliary Boolean variables to transform their SAT formulas into CNF. As such, **A** and **B** each reveal to each other the number of auxiliary Boolean variables they introduced. Then $\mathbf{F^A}$ and $\mathbf{F^B}$ are both "stretched" to the same number of rows such that the row vectors $\mathbf{f}_i^{\mathbf{A}}$ and $\mathbf{f}_i^{\mathbf{B}}$ record the occurrence and polarity of the same Boolean variable x_i in $\mathbf{F^A}$ and $\mathbf{F^B}$, separately. Given an auxiliary Boolean variable introduced by **A**, its corresponding row vector in $\mathbf{F^B}$ is a zero vector. We denote $m^{\mathbf{A}}$ and $m^{\mathbf{B}}$ the number of variables of $F^{\mathbf{A}}$ and $F^{\mathbf{B}}$ respectively, assume $\mathbf{F^A}$ is of n-by-$m^{\mathbf{A}}$, $\mathbf{F^B}$ is of n-by-$m^{\mathbf{B}}$ and let $m = m^{\mathbf{A}} + m^{\mathbf{B}}$.

After $\mathbf{F^A}$ and $\mathbf{F}^B$ are aligned, the solver uses a simple procedure as shown in Algorithm 1 to store an encrypted version of the matrix $\{\mathbf{F^A}\ \mathbf{F^B}\}$ privately at **A** where that is the concatenation of matrix $\mathbf{F^A}$ and matrix $\mathbf{F^B}$. At the same time, it stores the encryption key k_B used privately at **B**. In this work, we explicitly for encryption use one-time pad with pseudo random function (**PRF**) that takes the a key and a seed as input and outputs an almost random strings. We refer interested readers to [32] for more details about **PRF** and the proof security of this encryption scheme, .

Algorithm 1: Preparing for oblivious shuffling.

Input: A and B each holds its private formula $\mathbf{F^A}$ and $\mathbf{F^B}$
Output: **A** holds an encrypted version of $\{\mathbf{F^A}\ \mathbf{F^B}\}$, and **B** holds the key $k_{\mathbf{B}}$

1 **A** locally a generates key $k_{\mathbf{A}}$ and an n-by-$m^{\mathbf{A}}$ random matrix $\mathbf{R^A}$;
2 **A** sends an n-by-$m^{\mathbf{A}}$ matrix $\Psi = \mathbf{F^A} \oplus \mathbf{PRF}(k_{\mathbf{A}}, \mathbf{R^A})$, to **B**;
3 **B** locally a generates key $k_{\mathbf{B}}$, an n-by-$m^{\mathbf{A}}$ random matrix $\mathbf{R}_{\mathbf{A}}^{\mathbf{B}}$, and an n-by-$m^{\mathbf{B}}$ random matrix $\mathbf{R}_{\mathbf{B}}^{\mathbf{B}}$;
4 **B** sends $\Phi^{\mathbf{A}} = \Psi \oplus \mathbf{PRF}(k_{\mathbf{B}}, \mathbf{R}_{\mathbf{A}}^{\mathbf{B}})$, $\Phi^{\mathbf{B}} = \mathbf{F^B} \oplus \mathbf{PRF}(k_B, R_{\mathbf{B}}^{\mathbf{B}})$, $\mathbf{R}_{\mathbf{A}}^{\mathbf{B}}$ and $\mathbf{R}_{\mathbf{B}}^{\mathbf{B}}$ then sends it back to **A**;
5 **A** computes $\Psi \oplus \Phi^{\mathbf{A}} = \mathbf{F^A} \oplus \mathbf{PRF}(k_{\mathbf{B}}, \mathbf{R}_A^B)$;
6 **A** stores $\mathbf{R^B} = \{\mathbf{R}_A^B\ \mathbf{R}_B^B\}$ and $\mathbf{F}_{k_{\mathbf{B}}, \mathbf{R^B}} = \{\mathbf{F^A} \oplus \mathbf{PRF}(k_{\mathbf{B}}, \mathbf{R}_A^B)\ \mathbf{F^B} \oplus \mathbf{PRF}(k_{\mathbf{B}}, \mathbf{R}_B^B)\} = \{\mathbf{F^A}\ \mathbf{F^B}\} \oplus \mathbf{PRF}(k_{\mathbf{B}}, \mathbf{R^B})$, and **B** stores k_B;

Step 1: Oblivious shuffling of private inputs: Given a matrix **V**, the oblivious shuffling algorithm in the proposed

solver (Algorithm 2) decides a row permutation π on $\mathbf{V}$, and distributes the secret $(\pi, \pi(\mathbf{V})$ between $\mathbf{A}$ and $\mathbf{B}$. Specifically, it takes $\mathbf{R}$ and $\mathbf{V} \oplus \mathbf{PRF}(k_B, \mathbf{R})$, the information stored at A at the end of the preparation (Line 6 of Algorithm 1), where $\mathbf{V} = \mathbf{F} = \{\mathbf{F_A}\ \mathbf{F_B}\}$ and $\mathbf{R} = \mathbf{R}^B$; and $k_\mathbf{B}$, the information stored at B (Line 7 of Algorithm 1), as the input. It then has $\mathbf{A}$ and $\mathbf{B}$ each introduce new private random matrices (*i.e.*, $\mathbf{R^A}$ by $\mathbf{A}$ and $\mathbf{R}^1, \mathbf{R}^2$ by $\mathbf{B}$) and private permutations (*i.e.*, $\pi_\mathbf{A}$ by $\mathbf{A}$ and $\pi_\mathbf{B}$ by $\mathbf{B}$), and achieves the goal of oblivious shuffling through garbled circuit encoded computations of the local permutation of these random matrices and the local encryption keys of the other party.

The key ingredient of this shuffling is the math behind Line 8, where it can be derived that $\mathbf{\Theta} = \pi_\mathbf{A}(\mathbf{V} \oplus \mathbf{\Gamma} \oplus \Omega)$. With this result, in Line 11-12, $\mathbf{A}$ can store a piece of secret $s_\mathbf{A} = \pi_\mathbf{B}\pi_\mathbf{A}(\mathbf{V}) \oplus \pi_\mathbf{B}(\mathbf{PRF}(k_\mathbf{B}, \mathbf{R}^1))$, and $\mathbf{B}$ can store another piece $s_\mathbf{B} = \pi_\mathbf{B}(\mathbf{PRF}(k_\mathbf{B}, \mathbf{R}^1))$. As such, the secret can only be recovered as $\pi = s_\mathbf{A} \oplus s_\mathbf{B} = \pi_\mathbf{B}\pi_\mathbf{A}$, and $\pi(\mathbf{V}) = \pi_\mathbf{B}\pi_\mathbf{A}(\mathbf{V}$ when both pieces are combined together. During this shuffling, no input from A (*i.e.*, $\mathbf{R}$ and $\mathbf{V} \oplus \mathbf{PRF}(k_B, \mathbf{R})$) or B ($k_\mathbf{B}$) is exposed to the other party.

In addition to permuting the SAT formula matrix $\mathbf{F} = \{\mathbf{F_A}\ \mathbf{F_B}\}$ by row with π and distributing this secret between $\mathbf{A}$ and $\mathbf{B}$, The proposed SAT solver also uses Algorithms 1 and 2 to permute two row vectors **prior** and **assign** with π and distribute each resulting secret between $\mathbf{A}$ and $\mathbf{B}$. These two vectors represent the branching search strategy specified by the agreement provider $\mathbf{B}$. Specifically, **prior** represents the priorities of all Boolean variables in the verification formula $\mathbf{F}$. During the DPLL search, when there is no more unit literal in the formula, the variable that is not decided and has the highest priority will be selected as the next variable x_i and first be assigned a truth value of $assign_i$ to simplify the formula. How **prior** and **assign** are computed at $\mathbf{B}$ is out of the scope of the proposed solver [1]

Step 2: A DPLL garbled circuit with exposed, permuted search history: After using an oblivious shuffling algorithm to permute the verification formula $\mathbf{F}$ and the branching search strategy (**prior**, **assign**), the proposed SAT solver designs a garbled circuit that encodes the DPLL algorithm with an exposed, permuted search history (Algorithm 3). Specifically, Algorithm 3 uses the same backtrack-searching process and the unit literal search optimization as in the original DPLL.[2]

Compared with the DPLL algorithm, the key differences in Algorithm 3 are the use of oblivious subroutines (*i.e.*, unit literal search, resolution, branching, contradiction detection and checking) that are encoded by garbled circuits. Specifically, the operator $[\cdot]$ is used in Algorithm 3 to indicate that the bracketed expression is computed using a garbled circuit and the result is not released to either $\mathbf{A}$ or $\mathbf{B}$ unless explicitly specified using the *release* keyword. We present Algorithm 3 in this paper to illustrate the core of our work while we put all subroutines in Appendix B because of limited space.

[1]The proposed solver may be more efficient if the search strategy is decided by $\mathbf{A}$ and $\mathbf{B}$ collectively, and this is left for future work.

[2]The proposed solver does not use pure literal elimination for the reason of future extensivity because most of modern DPLL-based SAT solving algorithms do not use pure literal elimination.

Algorithm 2: Oblivious shuffling over private inputs.

Input: $\mathbf{A}$ holds $\mathbf{R}$ and $\mathbf{V} \oplus \mathbf{PRF}(k_B, \mathbf{R})$; $\mathbf{B}$ holds $k_\mathbf{B}$;
Output: $\mathbf{A}$ holds $\pi_\mathbf{B}\pi_\mathbf{A}(\mathbf{V}) \oplus \pi_\mathbf{B}(\mathbf{PRF}(k_\mathbf{B}, \mathbf{R}^1))$;
$\mathbf{B}$ holds $\pi_\mathbf{B}(\mathbf{PRF}(k_\mathbf{B}, \mathbf{R}^1))$;

1 $\mathbf{A}$ generates a key $k_\mathbf{A}$, a random matrix $\mathbf{R^A}$, and a permutation $\pi_\mathbf{A}$;
2 $\mathbf{A}$ computes $\pi_\mathbf{A}(\mathbf{V} \oplus \mathbf{PRF}(k_B, \mathbf{R}) \oplus \mathbf{PRF}(k_\mathbf{A}, \mathbf{R^A}))$, $\pi_\mathbf{A}(\mathbf{R^A})$ and $\pi_\mathbf{A}(\mathbf{R})$;
3 $\mathbf{A}$ sends $\pi_\mathbf{A}(\mathbf{V} \oplus \mathbf{PRF}(k_B, \mathbf{R}) \oplus \mathbf{PRF}(k_\mathbf{A}, \mathbf{R^A}))$ and $\pi_\mathbf{A}(\mathbf{R^A})$ to B;
4 $\mathbf{B}$ generates two random matrices $\mathbf{R}^1$ and $\mathbf{R}^2$;
5 $\mathbf{A}$ and $\mathbf{B}$ construct two garbled circuits to collectively compute $\mathbf{\Delta} = \mathbf{PRF}(k_\mathbf{B}, \pi_A(\mathbf{R}))$ and $\mathbf{\Gamma} = \mathbf{PRF}(k_\mathbf{A}, \mathbf{R}^2)$, respectively, without revealing the input to each other; and the circuits do not release the results to any party;
6 $\mathbf{A}$ computes $\mathbf{\Lambda} = \mathbf{PRF}(k_\mathbf{A}, \pi_A(\mathbf{R^A}))$;
7 $\mathbf{B}$ computes $\Omega = \mathbf{PRF}(k_\mathbf{B}, \mathbf{R}^1)$;
8 $\mathbf{A}$ and $\mathbf{B}$ construct a garbled circuit taking the output of the previous circuits to collectively compute $\mathbf{\Theta} = \pi_\mathbf{A}(\mathbf{V} \oplus \mathbf{PRF}(k_B, \mathbf{R}) \oplus \mathbf{PRF}(k_\mathbf{A}, \mathbf{R^A})) \oplus \mathbf{\Delta} \oplus \mathbf{\Gamma} \oplus \mathbf{\Lambda} \oplus \Omega$, which equals to $\pi_\mathbf{A}(\mathbf{V}) \oplus \mathbf{\Gamma} \oplus \Omega$, and the circuit only release $\mathbf{\Theta}$ to $\mathbf{B}$;
9 $\mathbf{B}$ generates a permutation $\pi_\mathbf{B}$, computes $\pi_\mathbf{B}(\mathbf{\Theta}) = \pi_\mathbf{B}\pi_\mathbf{A}(\mathbf{V}) \oplus \pi_\mathbf{B}(\mathbf{\Gamma}) \oplus \pi_\mathbf{B}(\Omega))$, $\pi_\mathbf{B}(\mathbf{R}^1)$, and $\pi_\mathbf{B}(\mathbf{R}^2)$, and sends them to $\mathbf{A}$;
10 $\mathbf{A}$ computes $\mathbf{PRF}(k_\mathbf{A}, \pi_\mathbf{B}(\mathbf{R}^2)) = \pi_\mathbf{B}(\Gamma)$;
11 $\mathbf{A}$ stores $s_\mathbf{A}(\mathbf{V}) = \pi_\mathbf{B}(\mathbf{\Theta}) \oplus \pi_\mathbf{B}(\Omega) = \pi_\mathbf{B}\pi_\mathbf{A}(\mathbf{V}) \oplus \pi_\mathbf{B}(\mathbf{PRF}(k_\mathbf{B}, \mathbf{R}^1))$;
12 $\mathbf{B}$ stores $s_\mathbf{B}(\mathbf{V}) = \pi_\mathbf{B}(\mathbf{PRF}(k_\mathbf{B}, \mathbf{R}^1))$;

Specifically, the algorithm takes $\pi(\mathbf{F})$, $\pi($**prior**$)$ and $\pi($**assign**$)$, the permuted verficiation formula and search strategy from the oblivious shuffling step, as input. These inputs are split into two secrets and distributed over $\mathbf{A}$ and $\mathbf{B}$, As such, for the simplicity of presentation without causing confusion, when these inputs show in the psuedocode, they represent the operations of combining the secrets held at $\mathbf{A}$ and $\mathbf{B}$. For example, if $\pi(\mathbf{F})$ is distributed as $s_A(\mathbf{F})$ to $\mathbf{A}$ and $s_B(\mathbf{F})$ $\mathbf{B}$, where $s_A(\mathbf{F}) \oplus s_B(\mathbf{F})$, then $[\pi(\mathbf{F})]$ represents $[s_A(\mathbf{F}) \oplus s_B(\mathbf{F})]$.

Moreover, two parties also initialize other three vectors: secret shared $\mathbf{u}$ that represents whether the literals are unit or not; public $\mathbf{d}$ that records if the values of the literals have been decided, and $\mathbf{c}$ that records if the clauses have been removed (satisfied). Last but not least, a stack T is also instantiated for backtracking later.

5.4 Analysis

In this section, we provides statements about the correctness, privacy and efficiency on GC-DPLL and related proofs are available in the appendix C. The security and correctness of this protocol rely on the security and correctness of oblivious shuffling and the way it performs permutation. We hereby

Algorithm 3: Garbled-Circuit DPLL

Input: Secrets $\pi(\mathbf{F})$, $\pi(\mathbf{prior})$ and $\pi(\mathbf{assign})$ that are distributed stored at **A** and **B**;

```
u = {false, ··· , false} ;
d = {false, ··· , false};
c = {false, ··· , false};
// u, d, ∈ {true, false}^n ,  c ∈ {true, false}^m
T is an empty stack;
// T is the stack that is initialized to be empty
i = 0 ;
while true do
    if i ≠ 0 then
        OblivRes(i, π(F), π(assign), c) ;
        ([b_s], [b_c]) = OC(π(F), ) ;
        [b_c].release(A, B) ;
        [b_s].release(A, B) ;
        if b_c then
            while ¬T.empty do
                (π(F), i, c, d, π(assign), state) = T.pop();
                if state == FIRST then
                    break
            if T.empty then
                return false
            [π(assign)_i] = [¬π(assign)_i];
            T.push(π(F), i, c, d, π(assign), SECOND) ;
            continue
        else
            if b_s then
                return true
    [ind] = OblivUL(π(F), π(prior), u, π(assign)) ;
    [ind].release(A, B) ;
    if ind ≠ 0 then
        i = ind ;
        d_i = true ;
    else
        [i] = OblivBranch(π(prior), d) ;
        [i].release(A, B) ;
        d_i = true ;
        T.push(π(F), i, c, d, π(assign), FIRST) ;
```

state them first here:

Theorem 5.1. *Given a pseudorandom function (**PRF**) and input* $(\mathbf{V} \oplus \mathbf{PRF}(k_\mathbf{B}, R), R)$ *, Algorithm 2:*

- *outputs* $s_\mathbf{A}(\mathbf{V})$ *and* $s_\mathbf{B}(\mathbf{V})$ *satisfying* $s_\mathbf{A}(\mathbf{V}) \oplus s_\mathbf{B}(\mathbf{V}) = \pi_\mathbf{B}\pi_\mathbf{A}(\mathbf{V})$
- *where* $\pi_\mathbf{A}$ *(respectively* $\pi_\mathbf{B}$*) is unknown to* **B** *(respectively* **A***);*
- *keeps the* v_i *unrevealed. In particular, given two vectors* **V** *and* **V′***, there is no probabilistic polynomial time* **A** *or* **B** *that is able to individually distinguish the algorithm is performed with* $(\mathbf{V} \oplus \mathbf{PRF}(k_\mathbf{B}, R), R)$ *or* $(\mathbf{V} \oplus \mathbf{PRF}(k_\mathbf{B}, R), R)$ *as an input.*

Correctness: The correctness of our algorithm is very natural as a result of the simple but elegant idea behind our algorithm. Notice that permuting the literals will not change the satisfiablity of a formula, to prove the correctness of our protocol, it is sufficient to show that algorithm 3 indeed implements DPLL.

Theorem 5.2. ***Correctness****: Given any SAT propositional formula F in CNF as input, it is satisfiable if and only if algorithm 3 returns true.*

Security: We allow **A** and **B** to learn searching pattern , *i.e.*the structure of the decision tree in DPLL, and the searching time. Given a formula, the backtrack search algorithm organizes the search for a satisfying assignment by maintaining a decision tree. By searching pattern, we mean the structure of the traversal tree in DPLL and defined. The searching pattern essentially only shows when branching and backtracking occurs but does not reveals what the branching literal is. We stress that we can still show security of the resulting system by formally proving the leakage is nothing more than this. We accept this certain minimal amount of leakage is unavoidable for the sake of efficiency. The formal proof of the theorem is based on simulation paradigm of cryptography [31] and completed and written separately.

Theorem 5.3. *The algorithm 3 realizes secure MPC for deciding the satisfiability of* $F^\mathbf{A} \wedge F^\mathbf{B}$ *with the leakage profile as the following:*

- *The number of common variables of* $F^\mathbf{A}$ *and* $F^\mathbf{B}$*.*
- *The number of variables of* $F^\mathbf{A}$ *and* $F^\mathbf{B}$*.*
- *The size of the input formulae*
- *The searching pattern*

Efficiency: Considering SAT problem is NP complete, to theoretically evaluate the efficiency of our protocol, we use overhead as our metric. Theorem C.3, essentially says that if the satisfiablity of a formula f can be decided by DPLL with heuristics that can be encoded using **assign** and **prior**, then to gain the privacy when deciding f, the overhead of computation is no more than $O(mn)$, where m is the number of clauses and n is the number of literals.

Theorem 5.4. *Given a formula* f*, denote* T *the number of steps in DPLL to determine if* f *is satisfiable with the heuristics that can be expressed as two vectors* **assign** *and* **prior***, about which we discuss in 5.3, then the computational complexity as well as the communication complexity of GC-DPLL is* $O(T \cdot mn)$*.*

6 Evaluation

This section presents a prototype implementation of IVeri, and evaluate its performance via extensive experiments.

6.1 Prototype Implementation

To model the interdomain peering agreements verification problem, the IVeri prototype uses Batfish [17] and Minesweeper [9]

to parse vendor-specific BGP configurations to an SMT formula, and then transforms the SMT formula to the configuration SAT formula $\mathbf{F_B}$ using bit blasting [37] and Tseytin transformation [50]. The prototype also uses a similar transformation procedure to transform the SMT formula of peering agreements to the agreement SAT formula $\mathbf{F_A}$.

The privacy-preserving SAT solver of this prototype is implemented using OpenSSL [51] and Obliv-C [59], a GCC wrapper to embed secure computation protocols inside regular C programs. The prototype uses AES-128 as the choice of PRF (pseudorandom function). One implementation optimization in the prototype is that, during the oblivious shuffling phase, instead of using component-wise operations on each element in the matrix, we divide elements in the same row of matrix into blocks of 128 bits, and use block-wise operations. Because each element in a SAT formula matrix has only 2 bits, this optimization improves the efficiency (in terms of computation, storage and communication) of oblivious shuffling by approximately 128/2=64 times. Overall, the IVeri prototype has ~2700 lines of C and Obliv-C code.

6.2 Experiment Methodology

The goal is to examine the efficiency of IVeri on interdomain peering agreement verification, and evaluate its overhead. To this end, two sets of experiments are designed. The first set is the functionality check experiment. Specifically, we setup the example abstract topology in Figure 4b as the evaluation scenario, where the two BGP routers of the agreement provider B are configured using the Cisco IOS configuration language, and the agreement consumer specifies five representative peering agreements introduced in Section 4.2. The second set of experiment uses a standard SAT datasets (*i.e.*, the SATLIB datasets [1]) is to evaluate the overhead of the privacy-preserving SAT solver more extensively. In both sets of experiments, the agreement consumer and provider each runs its IVeri process on an Ubuntu 18.04 server with Intel(R) Xeon(R) Platinum 8168 CPU and 376GB RAM.

6.3 Results

Functionality check: In the first experiment, all five representative peering agreements introduced in Section 4.2 are specified. For each agreement, we create different interdomain agreement verification problems by using three different sets of correct BGP configurations of B. As such, a total of 15 experiments are executed. In all 15 experiments, IVeri successfully determines that the problem of $\mathbf{F_A} \wedge \mathbf{F_B}$ is unsatisfiable, indicating that the agreement-to-verify is implemented correctly at B. For each agreement, we then deliberately change one correct BGP configuration to be incorrect, and executes IVeri again. Results show that in all these 5 experiments, IVeri successfully finds a satisfiable solution to the problem of $\mathbf{F_A} \wedge \mathbf{F_B}$, indicating that these agreements are not correctly implemented. With this functionality check result, we conclude that IVeri provides accurate results for interdomain peering agreement verification.

Overhead study via SAT benchmark simulation: In the second experiment, we select an open SAT dataset [1] whose input SAT formulas have 50 variables and 100 clauses. In the simulation, we vary the number of variables (n) to be from 5 to 50, with a step size of 5, and vary the number of clauses (m) to be from 10 to 100, with a step size of 10. For each combination, 30 SAT formulas are generated from the dataset. In each experiment, each formula is approximately separated in half and held by A and B, respectively.

The first metric we study is *oblivious shuffling delay*, *i.e.*, the time consumption of the oblivious shuffling phase of the proposed solver. Figure 5a shows that (1) the worst case of oblivious shuffling delay is less than 10 seconds; and (2) the delay increases sharply when m reaches 64. This sharp increase is the result of the implementation optimization, where the oblivious shuffling operations in the same row are block-wise with 128/2=64 elements, not component-wise. Figure 5b shows that the oblivious shuffling delay increases linearly as the number of variables increases. The separation of two lines in this figure is again the result of our block-wise shuffling operation.

Second, we study the *verification delay*, which is the sum of the oblivious shuffling delay and the delay of the GC-DPLL algorithm. Figure 6 shows that the oblivious shuffling delay takes about 50% of the verification delay, and that the impact of m on the verification delay increases as n increases.

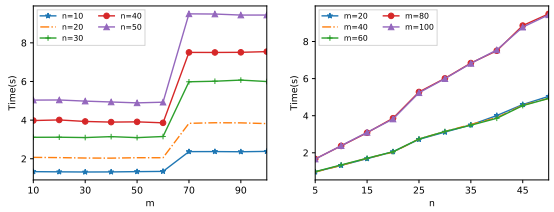
Third, we study the size of transmitted data during the verification. As shown in Figure 7, the size of total transmitted data is up to 3 GB. Although this seems like a large number, in modern networks where the interdomain bandwidth is often of hundreds of GBs, this data transmission overhead is very small.

Fourth, we compare the performance of the GC-DPLL algorithm with the original DPLL algorithm. Figure 8 compares their time consumption. We observe that the GC-DPLL algorithm increases the verification delay of the DPLL algorithm by three orders of magnitude. Considering that the absolute verification delay of GC-DPLL is of tens of seconds, this increase is acceptable.

With these overhead study, we conclude that IVeri is an efficient solution for privacy-preserving interdomain agreement verification.

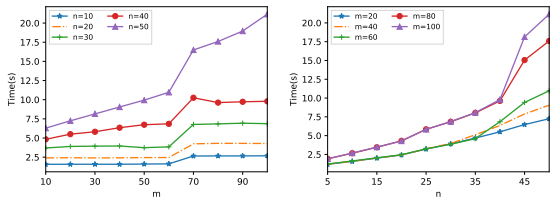
7 Related Work

Network verification: Network verification tools are roughly categorized in three classes. First, network configuration verification tools (*e.g.*, [9, 16, 20, 25]) focus on finding errors in the configurations of routing protocols (*i.e.*, the control plane of network) within the same network by building and analyzing a symbolic network model (*e.g.*, an SMT model). IVeri heavily leverages this approach to model BGP configurations, but goes beyond to build new models for interdomain



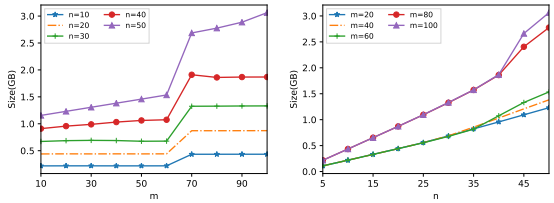
(a) Varying number of clauses m. **(b)** Varying number of variables n.

Figure 5. Oblivious shuffling delay of IVeri.



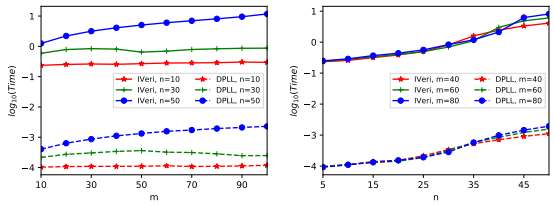
(a) Varying number of clauses m. **(b)** Varying number of variables n.

Figure 6. Verification delay of IVeri.



(a) Varying number of clauses m. **(b)** Varying number of variables n.

Figure 7. Data transmission overhead of IVeri: total message size.



(a) Varying number of clauses m **(b)** Varying number of variables n.

Figure 8. Computation delay: GC-DPLL vs. DPLL.

peering agreements. Second, network configuration testing tools (*e.g.*, [17, 33, 34, 43]) take the network configurations and a network environment as input, simulate the network behavior by generating data plane configurations, and analyze the resulting data plane configuration. Extending these tools to analyze the real behavior of interdomain networks requires the knowledge of the BGP configurations of different ASes, which are private information. Third, data plane verification tools (*e.g.*, [26, 27, 29, 40, 49, 56]) focus on verifying that the data plane configuration of a network, such as the forwarding tables and access control lists satisfies certain properties (*e.g.*, loop-freeness). Although tools such as Looking Glass [28] and RouteView [46] expose certain BGP data plane configurations (*e.g.*, selected routes) of participating ASes, such incomplete information makes it non-trivial to extend data plane verification tools to analyze interdomain networks, and left for future work.

Verification in interdomain networks: There are some efforts in interdomain network verification [11, 23, 47, 52, 53, 55, 61]. FSR [52] and BagPipe [53] focus on verifying the safety property (*i.e.*, stability) of BGP in interdomain networks and require the complete exposure of the BGP configurations of all the ASes. NetQuery [47] requires the agreement consumer to disclose agreement-to-verify to the agreement provider. SIDR [11] focuses on verifying the safety property of data plane configurations of multiple SDXex [22], and sacrifices accuracy for privacy and safety. NetReview [23] and VPerf [61] focus on the errors in BGP configurations in BGP peers and are work most related to IVeri. However, their functionalities are also affected by partial deployment. For example, given an agreement provider, VPerf [61] requires it and all its agreement consumers to deploy the system in order to verify BGP policies correctly. In contrast, a pair of BGP peers can achieve peering interdomain verification as long as both of them deploy IVeri.

SAT solver: SAT solver is an active research area with a rich literature [13, 21, 37, 39, 48, 60]. Studies in [41, 42] focus on the problem of how to disguise a SAT formula held by one party from the other party who holds an SAT solver, and they cannot solve the interdomain peering agreement verification problem To the best of our knowledge, the proposed SAT solver in IVeri is the first one that can decide the satisfiability of the conjunction of two private SAT formulas held by two parties, without exposing them.

Secure multi-party computation in networks: There have been proposals to leverage SMPC to design new interdomain routing protocols, which allow peering ASes to collectively compute interdomain routes for better network performance [8, 12, 35]. They are orthogonal to this paper, as IVeri focuses on verifying interdomina peering agreement instead of designing new routing protocols.

8 Conclusion

This paper systematically investigates the important problem of interdomain peering agreement verification. It identifies privacy as the fundamental challenge, and presents IVeri, the first privacy-preserving interdomain agreement verification system. IVeri models the interdomain agreement verification problem as a SAT formula, and develops a novel, efficient, privacy-serving SAT solver using oblivious shuffling and garbled circuits as key building blocks. Extensive evaluation on a prototype of IVeri demonstrates its efficiency and efficacy.

References

[1] 2011. SATLIB - Benchmark Problems. https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

[2] 2014. Time Warner Cable says outages largely resolved. http://www.seattletimes.com/business/time-warner-cable-says-outages-largely-resolved.

[3] 2018. Google goes down due to BGP errors. https://arstechnica.com/information-technology/2018/11/major-bgp-mishap-takes-down-google-as-traffic-improperly-travels-to-china/.

[4] 2019. BGP Churns in 2018. https://blog.apnic.net/2019/01/22/bgp-in-2018-bgp-churn/.

[5] 2019. BGP Route Leak Causes Cloudflare and Amazon AWS Problems. https://www.bleepingcomputer.com/news/technology/bgp-route-leak-causes-cloudflare-and-amazon-aws-problems/.

[6] 2019. BGPMon: Recent News and Updates: leak causing Internet outages in Japan, Brazil, India and etc. https://www.bgpmon.net/.

[7] Ruwaifa Anwar, Haseeb Niaz, David Choffnes, Ítalo Cunha, Phillipa Gill, and Ethan Katz-Bassett. 2015. Investigating interdomain routing policies in the wild. In *Proceedings of the 2015 Internet Measurement Conference*. ACM, 71–77.

[8] Gilad Asharov, Daniel Demmler, Michael Schapira, Thomas Schneider, Gil Segev, Scott Shenker, and Michael Zohner. 2017. Privacy-preserving interdomain routing at Internet scale. *Proceedings on Privacy Enhancing Technologies* 2017, 3 (2017), 147–167.

[9] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2017. A general approach to network configuration verification. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 155–168.

[10] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2018. Control plane compression. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 476–489.

[11] Rüdiger Birkner, Arpit Gupta, Nick Feamster, and Laurent Vanbever. 2017. SDX-Based Flexibility or Internet Correctness?: Pick Two!. In *Proceedings of the Symposium on SDN Research*. ACM, 1–7.

[12] Qingjun Chen, Shouqian Shi, Xin Li, Chen Qian, and Sheng Zhong. 2018. SDN-based privacy preserving cross domain routing. *IEEE Transactions on Dependable and Secure Computing* (2018).

[13] Martin Davis, George Logemann, and Donald W Loveland. 1961. *A machine program for theorem-proving*. New York University, Institute of Mathematical Sciences.

[14] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 337–340.

[15] David Evans, Vladimir Kolesnikov, Mike Rosulek, et al. 2018. A pragmatic introduction to secure multi-party computation. *Foundations and Trends® in Privacy and Security* 2, 2-3 (2018), 70–246.

[16] Seyed K Fayaz, Tushar Sharma, Ari Fogel, Ratul Mahajan, Todd Millstein, Vyas Sekar, and George Varghese. 2016. Efficient network reachability analysis using a succinct control plane representation. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 217–232.

[17] Ari Fogel, Stanley Fung, Luis Pedrosa, Meg Walraed-Sullivan, Ramesh Govindan, Ratul Mahajan, and Todd Millstein. 2015. A general approach to network configuration analysis. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*. 469–483.

[18] Lixin Gao. 2001. On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking (ToN)* 9, 6 (2001), 733–745.

[19] Lixin Gao and Jennifer Rexford. 2001. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking (TON)* 9, 6 (2001), 681–692.

[20] Aaron Gember-Jacobson, Raajay Viswanathan, Aditya Akella, and Ratul Mahajan. 2016. Fast control plane analysis using an abstract representation. In *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 300–313.

[21] Carla P Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. 2008. Satisfiability solvers. *Foundations of Artificial Intelligence* 3 (2008), 89–134.

[22] Arpit Gupta, Laurent Vanbever, Muhammad Shahbaz, Sean P. Donovanand Brandon Schlinker, Nick Feamster, Jennifer Rexford, Scott Shenker, Russ Clark, and Ethan Katz-Bassett. 2014. SDX: A Software Defined Internet Exchange. In *Proceedings og SIGCOMM 2014*. IEEE, 233–239.

[23] Andreas Haeberlen, Ioannis C Avramopoulos, Jennifer Rexford, and Peter Druschel. 2009. NetReview: Detecting When Interdomain Routing Goes Wrong.. In *NSDI*, Vol. 2009. 437–452.

[24] Ying Huang, David Evans, and Jonathan Katz. 2012. Private Set Intersection: Are Garbled Circuits Better than Custom Protocols?. In *NDSS*.

[25] Karthick Jayaraman, Nikolaj Bjørner, Jitu Padhye, Amar Agrawal, Ashish Bhargava, Paul-Andre C Bissonnette, Shane Foster, Andrew Helwer, Mark Kasten, Ivan Lee, et al. 2019. Validating datacenters at scale. In *Proceedings of the ACM Special Interest Group on Data Communication*. ACM, 200–213.

[26] Peyman Kazemian, Michael Chang, Hongyi Zeng, George Varghese, Nick McKeown, and Scott Whyte. 2013. Real time network policy checking using header space analysis. In *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*. 99–111.

[27] Peyman Kazemian, George Varghese, and Nick McKeown. 2012. Header Space Analysis: Static Checking for Networks.. In *NSDI*, Vol. 12. 113–126.

[28] Akmal Khan, Taekyoung Kwon, Hyun-chul Kim, and Yanghee Choi. 2013. As-level topology collection through looking glass servers. In *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 235–242.

[29] Ahmed Khurshid, Xuan Zou, Wenxuan Zhou, Matthew Caesar, and P Brighten Godfrey. 2013. Veriflow: Verifying network-wide invariants in real time. In *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*. 15–27.

[30] lhc [n. d.]. The Large Hadron Collider (LHC) Experiment. https://home.cern/topics/large-hadron-collider.

[31] Yehuda Lindell. 2017. *How to Simulate It – A Tutorial on the Simulation Proof Technique*. Springer International Publishing, Cham, 277–346.

[32] Yehuda Lindell and Benny Pinkas. 2009. A Proof of Security of Yao’s Protocol for Two-Party Computation. *J. Cryptol.* 22, 2 (April 2009), 161–188. https://doi.org/10.1007/s00145-008-9036-8

[33] Hongqiang Harry Liu, Yibo Zhu, Jitu Padhye, Jiaxin Cao, Sri Tallapragada, Nuno P Lopes, Andrey Rybalchenko, Guohan Lu, and Lihua Yuan. 2017. Crystalnet: Faithfully emulating large production networks. In *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 599–613.

[34] Nuno P Lopes and Andrey Rybalchenko. 2019. Fast BGP Simulation of Large Datacenters. In *International Conference on Verification, Model Checking, and Abstract Interpretation*. Springer, 386–408.

[35] Sridhar Machiraju and Randy H Katz. 2004. *Reconciling cooperation with confidentiality in multi-provider distributed systems*. Computer Science Division, University of California.

[36] Ratul Mahajan, David Wetherall, and Tom Anderson. 2002. Understanding BGP misconfiguration. In *ACM SIGCOMM Computer Communication Review*, Vol. 32. ACM, 3–16.

[37] Sharad Malik and Lintao Zhang. 2009. Boolean satisfiability from theoretical hardness to practical success. *Commun. ACM* 52, 8 (2009), 76–82.

[38] Vinod Mishra, Dinesh Verma, Chris Williams, and Kelvin Marcus. 2017. Comparing software defined architectures for coalition operations. In *2017 International Conference on Military Communications and Information Systems (ICMCIS)*. IEEE, 1–7.

[39] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. 2001. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th annual Design Automation Conference*. ACM, 530–535.

[40] Gordon D Plotkin, Nikolaj Bjørner, Nuno P Lopes, Andrey Rybalchenko, and George Varghese. 2016. Scaling network verification using symmetry and surgery. *ACM SIGPLAN Notices* 51, 1 (2016), 69–83.

[41] Ying Qin, ShengYu Shen, and Yan Jia. 2014. Structure-aware CNF obfuscation for privacy-preserving SAT solving. In *2014 Twelfth ACM/IEEE Conference on Formal Methods and Models for Codesign (MEMOCODE)*. IEEE, 84–93.

[42] Ying Qin, Xiao Yang Shen, and Zhen Yue Du. 2018. Privacy-Preserving SAT Solving Based on Projection-Equivalence CNF Obfuscation. In *International Symposium on Cyberspace Safety and Security*. Springer, 224–239.

[43] Bruno Quoitin and Steve Uhlig. 2005. Modeling the routing of an autonomous system with C-BGP. *IEEE network* 19, 6 (2005), 12–19.

[44] Mariana Raykova. 2012. *Secure Computation in Heterogeneous Environments: How to Bring Multiparty Computation Closer to Practice?* Columbia University.

[45] Yakov Rekhter, Susan Hares, and Dr. Tony Li. 2006. A Border Gateway Protocol 4 (BGP-4). RFC 4271. https://doi.org/10.17487/RFC4271

[46] Oregon RouteViews. [n. d.]. University of Oregon RouteViews project. *Eugene, OR.[Online]. Available: http://www. routeviews. org* ([n. d.]).

[47] Alan Shieh, Emin Gün Sirer, and Fred B Schneider. 2011. NetQuery: A knowledge plane for reasoning about network properties. In *ACM SIGCOMM Computer Communication Review*, Vol. 41. ACM, 278–289.

[48] Niklas Sorensson and Niklas Een. 2005. Minisat v1. 13-a sat solver with conflict-clause minimization. *SAT* 2005, 53 (2005), 1–2.

[49] Radu Stoenescu, Matei Popovici, Lorina Negreanu, and Costin Raiciu. 2016. SymNet: Scalable symbolic execution for modern networks. In *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 314–327.

[50] Grigori S Tseitin. 1983. On the complexity of derivation in propositional calculus. In *Automation of reasoning*. Springer, 466–483.

[51] John Viega, Matt Messier, and Pravir Chandra. 2002. *Network security with openSSL: cryptography for secure communications.* " O'Reilly Media, Inc.".

[52] A. Wang, L. Jia, W. Zhou, Y. Ren, B. T. Loo, J. Rexford, V. Nigam, A. Scedrov, and C. Talcott. 2012. FSR: Formal Analysis and Implementation Toolkit for Safe Interdomain Routing. *IEEE/ACM Transactions on Networking* 20, 6 (Dec 2012), 1814–1827. https://doi.org/10.1109/TNET.2012.2187924

[53] Konstantin Weitz, Doug Woos, Emina Torlak, Michael D Ernst, Arvind Krishnamurthy, and Zachary Tatlock. [n. d.]. Formal semantics and automated verification for the border gateway protocol. ([n. d.]).

[54] Qiao Xiang, Jingxuan Jensen Zhang, Xin Tony Wang, Yang Jace Liu, Chin Guok, Franck Le, John MacAuley, Harvey Newman, and Y Richard Yang. 2019. Toward Fine-Grained, Privacy-Preserving, Efficient Multi-Domain Network Resource Discovery. *IEEE Journal on Selected Areas in Communications* 37, 8 (2019), 1924–1940.

[55] Hongkun Yang and Simon S Lam. 2014. Collaborative verification of forward and reverse reachability in the Internet data plane. In *2014 IEEE 22nd International Conference on Network Protocols*. IEEE, 320–331.

[56] Hongkun Yang and Simon S Lam. 2015. Real-time verification of network properties using atomic predicates. *IEEE/ACM Transactions on Networking* 24, 2 (2015), 887–900.

[57] A. C. Yao. 1982. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*.

[58] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 162–167.

[59] Samee Zahur and David Evans. 2015. Obliv-C: A Language for Extensible Data-Oblivious Computation. *IACR Cryptology ePrint Archive* 2015 (2015), 1153.

[60] Hantao Zhang. 1997. SATO: An efficient propositional prover. In *International Conference on Automated Deduction*. Springer, 272–275.

[61] Mingchen Zhao, Wenchao Zhou, Alexander JT Gurney, Andreas Haeberlen, Micah Sherr, and Boon Thau Loo. 2012. Private and verifiable interdomain routing decisions. *ACM SIGCOMM Computer Communication Review* 42, 4 (2012), 383–394.

A Modeling of three other representative peering agreements

AS path prepending: Consider the case where AS A wants to verify a peering agreement that if A sets the community tag to 10:333 in any route announcement toward prefix 172.217.8.0/24, B will prepend itself three times in the AS path before exporting it to other ASes (*e.g.*, instead of announcing (172.217.8.0/24, [B A]), B should announce (172.217.8.0/24, [B B B A])). The administrator of AS A can express this agreement as:

$$\begin{aligned}&\forall \mathrm{i} = 3, 4, 5.\\ &\quad \mathrm{FBM}(172.217.8.0, \mathrm{e}_{\mathrm{R_0R_1}}.\mathrm{prefix}, 24)\\ &\quad \wedge \mathrm{e}_{\mathrm{R_0R_1}}.\mathrm{comAS} = 10 \wedge \mathrm{e}_{\mathrm{R_0R_1}}.\mathrm{comVal} = 333\\ &\quad \wedge \mathrm{FBM}(\mathrm{e}_{\mathrm{R_0R_1}}.\mathrm{prefix}, \mathrm{out}_{\mathrm{R_2R_i}}.\mathrm{prefix}, \mathrm{e}_{\mathrm{R_0R_1}}.\mathrm{prefixLength})\\ &\quad \wedge \mathrm{out}_{\mathrm{R_2R_i}}.\mathrm{nextHop} = \mathrm{A} \wedge \mathrm{out}_{\mathrm{R_2R_i}}.\mathrm{valid} = \mathrm{true}\\ &\quad \Rightarrow \mathrm{contain}(\mathrm{out}_{\mathrm{R_2R_i}}.\mathrm{asPath}, [\mathrm{B}, \mathrm{B}, \mathrm{B}]),\end{aligned} \tag{6}$$

where *nextHop* represents the next hop AS in an AS path. Such agreement are commonly used for A to balance its inbound traffic from different ASes.

After the agreement consumer's administrator specifies the agreement-to-verify in a SMT formula, the agreement consumer transforms the negation of this SMT formula to the agreement SAT formula. As such, the interdomain peering agreement verification problem is modeled as deciding whether the conjunction of the configuration SAT formula and the agreement SAT formula is satisfiable. If not, such an agreement is correctly implemented by the agreement provider. Otherwise, it is incorrectly implemented.

Prefer certain ASes: Consider the case where AS A wants to verify a peering agreement that when B receives multiple route announcements toward prefix 144.82.250.0/24 from peering routers R_3, R_4, and R_5, B will prefer to select a route whose next hop is AS C. The administrator of AS A can express this agreement as:

$$\begin{aligned}&\forall \mathrm{i}, \mathrm{j} = 3, 4, 5.\ \mathrm{FBM}(144.82.250.0, \mathrm{in}_{\mathrm{R_iR_2}}.\mathrm{prefix}, 24)\\ &\qquad \wedge \mathrm{FBM}(144.82.250.0, \mathrm{in}_{\mathrm{R_jR_2}}.\mathrm{prefix}, 24) \wedge \mathrm{i} \neq \mathrm{j}\\ &\qquad \wedge \mathrm{in}_{\mathrm{R_iR_2}}.\mathrm{nextHop} = \mathrm{C} \wedge \mathrm{in}_{\mathrm{R_jR_2}}.\mathrm{nextHop} \neq \mathrm{C}\\ &\qquad \Rightarrow \mathrm{in}_{\mathrm{R_iR_2}} \preceq \mathrm{in}_{\mathrm{R_jR_2}}.\end{aligned} \tag{7}$$

It is often used for A to balance its outbound traffic or fulfill the traffic traversal regulations. The Gao-Rexford guideline [19] is one special case of this agreement.

Interested in particular prefix: Consider the case where AS A wants to verify a peering agreement that B will only send route announcements toward prefix 144.82.250.0/24 to A. The administrator of AS A can express this agreement as:

$$\begin{aligned}&\neg\mathrm{FBM}(144.82.250.0, \mathrm{out}_{\mathrm{R_1R_0}}.\mathrm{prefix}, 24) \Rightarrow\\ &\qquad \mathrm{out}_{\mathrm{R_1R_0}}.\mathrm{valid} = \mathrm{false}.\end{aligned} \tag{8}$$

This agreement is also often used for A to balance its outbound traffic.

B Subroutines of Algorithm 3

This appendix presents the pseudocode of the subroutines used in the GC-DPLL algorithm.

Algorithm 4: OblivRes

Input: i_0, $[\pi(\mathbf{F})]$, $[\pi(\mathbf{assign})]$, $[\mathbf{c}]$
Output: None

for j = *1 to* m **do**
  $[b] = [\pi(\mathbf{assign})_{i_0}]$;
  $[\mathrm{cond}_0] = [\neg\pi(\mathbf{F})_{i_0,j}.O]$;
  $[\mathrm{cond}_1] = [(b \neq \pi(\mathbf{F})_{i_0,j}.P) \cdot \pi(\mathbf{F})_{i_0,j}.O]$;
  $[\mathrm{cond}_2] = [(b == \pi(\mathbf{F})_{i_0,j}.P) \cdot \pi(\mathbf{F})_{i_0,j}.O]$;
  for i = *1 to* n **do**
    $[\mathrm{cond}_3] = [i == i_0]$;
    $[\pi(\mathbf{F})_{i,j}.O] = [\mathrm{cond}_0 \cdot \pi(\mathbf{F})_{i,j}.O + \mathrm{cond}_2.0 + \mathrm{cond}_1 \cdot \mathrm{cond}_3 \cdot 0 + \mathrm{cond}_1 \cdot (1 - \mathrm{cond}_3) \cdot \pi(\mathbf{F})_{i,j}.O]$;
  end
  $\mathbf{c}_j = \mathrm{cond}_2$
end

Algorithm 5: OblivUL

Input: $\pi(\mathbf{F})$, $[\pi(\mathbf{prior})]$, $[\pi(\mathbf{assign})]$, $[\mathbf{u}]$
Output: [ind] such that ind $\in \{1, \cdots, n\}$

for $j = 1$ *to* m **do**
  $[b_j] = 0$;
  for $i = 1$ *to* n **do**
    $[b_j] = [\, b_j + \pi(\mathbf{F})_{i,j}.O]$
  end
  for $i = 1$ *to* n **do**
    $[\mathrm{cond}] = [(b_j == 1) \wedge \pi(\mathbf{F})_{i,j}.O == 1)]$;
    $[\mathbf{u}_i] = [\mathrm{cond} \cdot 1 + \neg\mathrm{cond} \cdot \mathbf{u}_i]$;
    $[\pi(\mathbf{assign})_i] = [\mathbf{u}_i \cdot \pi(\mathbf{F})_{i,j}.P + (1 - \mathbf{u}_i) \cdot \pi(\mathbf{assign})_i]$;
  end
end
[ind] = [0];
[pri] = [0];
for i = *1 to* n **do**
  $[\mathrm{cond}] = [(\mathbf{u}_i == 1) \cdot (\pi(\mathbf{prior})_i > \mathrm{pri})]$;
  $[\mathrm{ind}] = [\mathrm{cond} \cdot i + \mathrm{ind} \cdot (1 - \mathrm{cond})]$;
  $[\mathrm{pri}] = [\mathrm{cond} \cdot \pi(\mathbf{prior})_i + \mathrm{pri} \cdot (1 - \mathrm{cond})]$;
end
return [ind]

C Missing proofs from Section 5.4

Theorem C.1. ***(Restated)*** *Given a pseudorandom function (**PRF**) and input vector* $\{(\mathbf{V} \oplus \mathbf{PRF}(k_\mathbf{B}, R), R)\}$ *, Algorithm 2:*

- *outputs* $s_\mathbf{A}(\mathbf{V})$ *and* $s_\mathbf{B}(\mathbf{V})$ *satisfying* $s_\mathbf{A}(\mathbf{V}) \oplus s_\mathbf{B}(\mathbf{V}) = \pi_\mathbf{B}\pi_\mathbf{A}(\mathbf{V})$
- *where* $\pi_\mathbf{A}$ *(respectively* $\pi_\mathbf{B}$*) is unknown to* **B** *(respectively* **A***);*
- *keeps the* v_i *unrevealed. In particular, given two vectors* **V** *and* **V′***, there is no probabilistic polynomial time* **A**

Algorithm 6: OblivBranch

Input: $[\pi(\mathbf{prior})]$, $\mathbf{d}$
Output: $[\text{ind}']$ such that $\text{ind} \in \{1, \cdots, n\}$

```
[ind'] = [0];
[pri] = [0];
for i = 1 to n do
    [cond] = [¬d_i · (π(prior)_i > pri)] ;
    [ind'] = [cond · i + ind' · (1 − cond)];
    [pri] = [cond · π(prior)_i + ind' · (1 − cond)] ;
end
return ind' ;
```

Algorithm 7: OC

Input: $[\pi(\mathbf{F})]$, $[\mathbf{c}]$
Output: $[b_s]$, $[b_c]$

```
[b] = false;
[s] = 0;
for j = 1 to m do
    [z] = 0 ;
    for i = 1 to n do
        [z] = [z + π(F)_{i,j}.O · 1];
    end
    [b] = [b ∨ (z == 0 ∧ ¬c_j)] ;
    [s] = [s + z]
end
return ([s == 0], [b])
```

or $\mathbf{B}$ *that is able to individually distinguish the algorithm is performed with* $(\mathbf{V} \oplus \mathbf{PRF}(k_\mathbf{B}, R), R)$ *or* $(\mathbf{V} \oplus \mathbf{PRF}(k_\mathbf{B}, R), R)$ *as an input.*

Proof. The first property can be shown as

$$s_\mathbf{A}(\mathbf{V}) \oplus s_\mathbf{B}(\mathbf{V}) = \pi_\mathbf{B}(\mathbf{PRF}(k_\mathbf{B}, \mathbf{R}^1)) \oplus \pi_\mathbf{B}\pi_\mathbf{A}(\mathbf{V}) = \pi_\mathbf{B}\pi_\mathbf{A}(\mathbf{V})$$

For the last two on security, we here provides the high-level idea of the proof while while a full analysis with formal definitions and extensive proofs is completed and written separately. The security of this oblivious shuffling relies on security of one-time pad encryption and garbled circuits that is proved in [32]. The central idea of our proof is reduction from the attacks against the privacy of this oblivious shuffling scheme to the attacks against either garbled circuits or the one-time pad encryption. □

Theorem C.2. ***(Restated) Correctness****: Given any SAT propositional formula F in CNF as input, it is satisfiable if and only if algorithm 3 returns true.*

Proof. We provide sketch of proof that shows that algorithm 3 indeed implements DPLL. More formal proof can be easily derived. Essentially, we conduct as many **OblivUL** as possible until no unit literal is available (corresponding to searching unit literal and removing clauses). Then, **OblivBranch** is called and current state is pushed into stack, after which **OblivRes** and **OC** are called to simplify $\pi(\mathbf{F})$ and test validity of our guess (corresponding to the branching). Whenever a failure (b_c = false) arises, we backtrack the searching tree until a FIRST guess is found by popping out the stack (corresponding backtracking). On the other hand, if a success is met, *i.e.* b_s = true, the algorithm terminates and outputs true. Since each variable has at most 2 possible assignments, the algorithm either eventually returns true at some point, or it exhausts all possible assignments and reaches the bottom of stack T then returns false. □

Theorem C.3. ***(Restated)*** *Given a formula f, denote T the number of steps in DPLL to determine if f is satisfiable with the heuristics that can be expressed as two vectors* **assign** *and* **prior***, about which we discuss in 5.3, then the computational complexity as well as the communication complexity of GC-DPLL is $O(T \cdot mn)$.*

Proof. The proof can be directly derived from two facts (a) the overhead from garbled circuits is constant [] so the computational complexity and communication complexity of each subroutine are both bounded by $O(mn)$. (b) the computation and communication for oblivious shuffling is $O(mn)$. The other parts of our protocol are exactly the same as DPLL, therefore the total computation and communication will be bounded by $O(T \cdot mn)$. □