Toward Privacy-Preserving Interdomain Configuration Verification via Multi-Party Computation

Huisan Xu[†], Qiuyue Qin[†], Xing Fang[†], Qiao Xiang[†], Jiwu Shu[†◇]
[†]Xiamen Key Laboratory of Intelligent Storage and Computing, Xiamen University [◇]Minjiang University

ABSTRACT

Interdomain network configuration errors can lead to disastrous financial and social consequences. Although substantial progress has been made in using formal methods to verify whether network configurations conform to certain properties, current tools focus on a single network. The fundamental challenge of configuration verification in an interdomain network is *privacy*, because each autonomous system (AS) treats its network configuration files as private information and is not willing to share it with others. In this paper, we take a first step toward interdomain network configuration verification and propose InCV, a privacy-preserving interdomain configuration verification system based on data-oblivious computation. Given an interdomain network, InCV allows ASes to collaboratively simulate the running of the network and verify the resulting interdomain routing information base (RIB) without revealing their network configurations to any party. Preliminary evaluation using real-world topologies and synthetic network configurations shows that InCV can verify an interdomain network of 32 ASes within ~52 minutes with reasonable overhead.

CCS CONCEPTS

• **Networks** → **Error detection and error correction**; **Network reliability**; • **Security and privacy** → **Domain-specific security and privacy architectures**;

KEYWORDS

Interdomain networks, Network verification, Secure multi-party computation

Huisan Xu and Qiuyue Qin are co-primary authors. Qiao Xiang is the corresponding author.

APNET 2023, June 29–30, 2023, Hong Kong, China

ACM ISBN 979-8-4007-0782-7/23/06…$15.00
https://doi.org/10.1145/3600061.3600064

1 INTRODUCTION

The Border Gateway Protocol (BGP) [26] is the de facto routing protocol used in interdomain networks. Interdomain BGP configuration errors can cause substantial social and financial losses in interdomain networks. In October 2021, Facebook, Instagram, and WhatsApp suddenly became unreachable [3], with browsers displaying DNS errors when attempting to open them. It was believed the outage was caused by a BGP-related configuration error, which took the operators over 5 hours to fix. Cloudflare suffered an outage in 2019 that affected websites around the world [2]. According to Cloudflare, over 16 million Internet sites utilized their services for performance enhancement, DDoS mitigation, or other features. In November 2018, Nigerian internet service provider MainOne admitted that it made a configuration error during a network upgrade [4]. That error caused a disruption of key Google services by routing traffic to China and Russia.

Given these frequent and serious outages in the real world, it is a top priority for network operators to ensure the correctness of interdomain BGP configurations. To this end, most interdomain network operators depend on query-based tools (*e.g.*, ping, traceroute, and looking glasses) and human interaction (*e.g.*, the NANOG mailing lists and the INOC-DBA phone systems). However, these tools are not only inefficient and error-prone, but also can only find errors after deployment.

Network configuration verification: preventing configuration errors before deployment using formal methods. Over the past decade, substantial progress has been achieved by network configuration verification tools [1, 6, 7, 13–15, 19, 22, 25, 28, 29, 35, 37]. These tools analyze network configurations before they are deployed on network devices to decide if they will compute a requirement-compliant data plane. Many companies have deployed configuration verification tools in their production networks (*e.g.*, Batfish [14] and Hoyan [35]).

Proposal: preventing interdomain network errors using interdomain configuration verification. With the success of these tools [1, 6, 7, 13–15, 19, 35, 37], we propose to extend network configuration verification to prevent network failures in interdomain networks. The fundamental challenge of the proposal lies in *privacy*. Specifically, in an

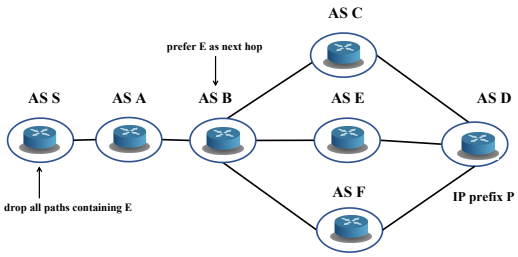

Figure 1: An example of an interdomain network to illustrate the need for interdomain configuration verification.

interdomain network, each AS considers its routing configurations as highly sensitive information and must keep them private. In contrast, existing network configuration verification tools require collecting all the routing configurations as input because they work under the setting of a single network. A strawman solution is to deploy a configuration verification tool at a trusted third party. However, such a third party would become a single point of failure and security vulnerability, and it is also hard to find such a party in the current operation of interdomain networks.

Another design is to build a shadow BGP network among ASes where ASes can interact to compute and check the correctness of their data plane. This simple approach appears reasonable but has the following drawbacks: (1) it is inefficient when verifying network properties under complex scenarios. For example, to verify reachability under k link failures [6], ASes have to run in the shadow network for C_n^k times to cover all possible failure scenarios; (2) it requires ASes to reveal their private data planes to other parties, which would cause unnecessary information leakage.

InCV: privacy-preserving interdomain configuration verification. To address the fundamental privacy challenge of interdomain configuration verification, we design InCV, a privacy-preserving interdomain configuration verification system, using secure multi-party computation (SMPC). InCV leverages Shamir's secret sharing [27] to allow participating ASes to collaboratively verify their BGP configurations while keeping them private. We implement a prototype of InCV and preliminary results using real-world topologies with synthesized configurations show that InCV can verify an interdomain network of 32 ASes within ~52 minutes with reasonable storage and communication overhead.

2 MOTIVATION AND BACKGROUND

In this section, we demonstrate the importance of interdomain configuration verification using a motivating example and introduce related background.

2.1 A Motivating Example

Consider the network in Figure 1. All AS routers are configured with standard BGP configurations, with the exception of AS B preferring AS E as the next hop and AS S filtering out all routes containing AS E. In such a setting, it is easy to observe that AS S cannot reach the IP prefix P. In real-world operations, AS S can only find that it cannot reach P after all ASes deploy their configurations. To troubleshoot this issue, operators typically need to use query-based tools (*e.g.*, ping, traceroute, and looking glasses) and ask for help on public forums (*e.g.*, the Nanog Mailing List). During the time of troubleshooting, the IP prefix P will keep being unreachable from AS S, causing financial and social loss. In contrast, if a network configuration verification tool can be deployed in this interdomain network, AS S can find this error before all ASes deploy their configurations, avoiding unnecessary service interruption and loss.

Privacy: the fundamental challenge of interdomain configuration verification. Given the fact that ASes prefer not to reveal their BGP configurations and that all current configuration verification tools require configurations as input, we argue that a practical interdomain configuration verification tool must be able to function while keeping ASes' configurations private. Operators' feedback to our brief survey on the NANOG Mailing List [30] also corroborates this point of view.

2.2 Background on Secure Multi-Party Computation

Secure multi-party computation (SMPC) [33] addresses scenarios in which n parties $P_i, ..., P_n$ hold private inputs $x_1, ..., x_n$ and wish to compute $y = f(x_1, ..., x_n)$ in such a way that all parties learn y but no P_i learns anything about $x_j, i \neq j$, except what is logically implied by the result y and the particular input x_i that he already knew.

An SMPC algorithm consists of multiple SMPC primitives (*e.g.*, addition, multiplication and comparison). There are different ways to build SMPC primitives. In this paper, we choose to use Shamir's secret sharing [27] because it has been proven to be efficient for designing SMPC primitives involving multiple parties (*e.g.*, $n > 2$) and has good scalability [9, 23].

Shamir's secret sharing. A (t, n)-Shamir's secret sharing scheme allows a secret S to be split and stored at n participants such that an adversary (1) can only reconstruct S when he/she has the pieces from at least t participants; (2) cannot gain any information about S if he/she has pieces from less than t participants. Shamir's secret sharing [27] is based on polynomial interpolation. It works in two phases:

(1) *share generation*: The owner of the secret S first selects a random polynomial $f(x)$ of degree $t-1$ to carry the secret and generates secret shares $S_i = f(i)$, where $i = 1, \ldots, n$. It then sends each S_i to participant i.

(2) *share reconstruction*: By collecting more than t shares, we can generate the original coefficients of the polynomial

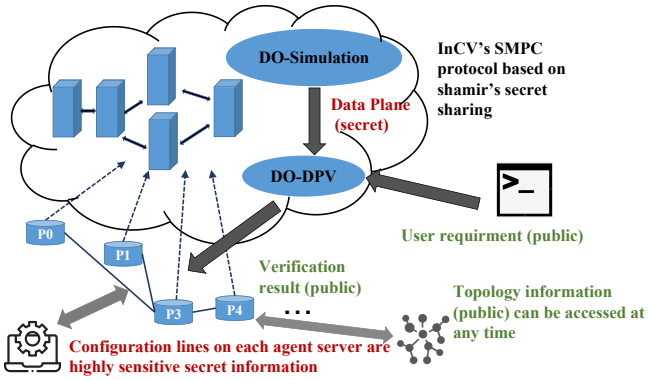

Figure 2: An illustration for InCV's architecture. All ground routers dispatch their configurations to agents on the cloud. The agents then collectively run the InCV's secure multi-party computation (SMPC) protocol

and reconstruct the secret by the Lagrange Interpolation method, as is shown in Equations (1) and (2). In the equations, t stands for the secret sharing threshold, S_j stands for the secret shares of the corresponding participant, and p is a prime number indicating the field of computation.

$$L_j(x) = \Pi_{i=0}^{t} \frac{x-i}{j-i} mod\ p\ (i \neq j); \quad (1)$$

$$f(x) = \Sigma_{j=0}^{t} S_j L_j(x) \quad (2)$$

Data-oblivious computation. Data-oblivious computation [16, 24] is a key concept in building complex SMPC algorithms. Given an algorithm, it is data-oblivious if and only if given any input data, the sequence of operations of this algorithm stays the same. Given an algorithm built exclusively on SMPC primitives, if it is also data-oblivious, it is also an SMPC algorithm [17]. Designing a data-oblivious algorithm is non-trivial because most data structures are not data-oblivious.

3 INCV DESIGN

We now present the design of InCV, an interdomain configuration verification system that enables operators of different ASes to collaboratively determine whether their configurations are correct while keeping their router configurations private. For ease of exposition, we assume a one-big-switch abstraction (*i.e.*, one BGP router) for each AS.

3.1 Overview

Security model. InCV assumes a semi-honest security model, where all ASes follow InCV's workflow specification but may collude to share information they obtain during its execution [34]. This model is sufficient for multiple scenarios of interdomain routing, including commercial Internet [18] and collaboration science networks where member networks share resources to collaboratively conduct exascale data transfers, storage, and analytics [11].

Architecture. Figure 2 presents the architecture of InCV. Each participating AS deploys an agent containing its BGP configurations in the cloud. The agents participate in the InCV's verification computation while keeping their configurations private. An alternative design is to let each AS deploy its agent in its local server and connect these local servers. We choose the cloud for deploying ASes' agents instead for better scalability. Specifically, InCV adopts secure multi-party computation for privacy-preserving verification, which requires exchanging a large number of encrypted messages among agents. As such, deploying agents in a cloud environment allows InCV to leverage the high bandwidth and low latency of the cloud to improve its efficiency.

We choose to use the simulation-based verification approach (*e.g.*, [14, 22, 25, 35]) in InCV, in which participating ASes collaboratively compute the data plane of the interdomain network with their private BGP configurations and examine the computed data plane to determine the correctness of configurations. We do not use the SMT-based approach [6, 29] because it cannot encode paths compactly, and checking the properties of paths (*e.g.*, waypoint and blacklist of ASes in AS paths) is important for interdomain networks.

Workflow. The workflow of InCV is illustrated in Figure 2. InCV provides an interface for operators from different ASes to specify their requirements on AS paths as public information. At the beginning of the verification, each AS locally uses a modified version of Batfish [14] to transform its private BGP configurations to a vendor-neutral intermediate representation (IR). It then encodes the IR in the form of vectors and matrices of integers and sends them to its own agent as the input of DO-Simulation. This encoding is necessary because vectors and matrices are data-oblivious data structures [20] but the original Batfish IR object is not. Note that the transformation process from configurations to vector and matrix formatted IR is conducted by each AS in plaintexts individually.

After each agent receives such information from its corresponding AS, they collaboratively execute a data-oblivious simulation (DO-Simulation) algorithm to compute a *secret* network data plane in an (IP prefix, AS path) mapping (*i.e.*, no agent can independently know the network data plane) without revealing their own IRs to any other agent. The plaintext version of DO-Simulation is shown in Algorithm 1. Afterward, agents continue to collaborate to execute a data-oblivious data plane verification (DO-DPV) using the secret network data plane computed by DO-simulation as the input to verify whether the computed AS paths satisfy operator-specified public requirements (*e.g.*, reachability, waypointing, and blacklisting).

Algorithm 1: BGP Simulation

Input: Origin is the announcement node of the given prefix, **N** is the set of nodes in the network
Output: Best path to the given prefix of each node
Init: $\forall n \in Origin : best(n) \leftarrow \varepsilon, adv(n) \leftarrow \varepsilon$;
Init: $\forall n \in N - Origin : best(n) \leftarrow \emptyset, adv(n) \leftarrow \emptyset$;
while *true* **do**
  $E \leftarrow \{n \in N \mid adv(n) \neq \emptyset\}$;
  if $E = \emptyset$ **then**
    break ;
  end
  $n \leftarrow pickNode(E)$;
  for ***each*** $peer \in peers(n)$ **do**
    $exports \leftarrow$ `Export(n,peer,adv(n))` ;
    $imports \leftarrow$ `Import(peer,n,exports)` ;
    $adv(peer) \leftarrow$ `Compare(best(peer),imports)`;
  end
  $adv(n) \leftarrow \emptyset$;
end

3.2 DO-Simulation: Data-Oblivious Network Simulation

We now present DO-Simulation, a privacy-preserving data plane simulation algorithm. The objective of DO-Simulation is to let AS agents collectively simulate their BGP configurations to compute a network data plane stored as a (t, n)-Shamir secret while keeping their configurations private.

A non-data-oblivious BGP simulation algorithm. Algorithm 1 gives the details of this algorithm for a given destination IP prefix. In the initial state, only the origin node has an origination route to the given prefix. The origination route will then be put into the advertisement list and later announced to corresponding neighbors. The main simulation process runs after initialization. In each iteration, one node whose advertisement list is not empty will be chosen. Then, the node will send the route advertisements in the list to its neighbors. The export and import functions filter out route advertisements that can not be sent out or received. The compare function compares the import routes with the current best path. If the import route has higher priority according to the route attributes, the best path and advertisement list will update. There are two types of route advertisements. One is for advertising the new best route to neighbors, and one is for notifying neighbors of the failure of the old route, just like what BGP does in reality. Finally, a converged state is reached if all node's advertisement lists are empty, meaning all nodes have computed their best route to the given prefix.

Transforming Algorithm 1 to DO-Simulation. We accomplish this goal in two steps. First, we implement all basic operations (*i.e.*, addition, multiplication, and comparison) in Algorithm 1 using n-party SMPC primitives based on Shamir's secret sharing. Second, we make the control flow of this algorithm data-oblivious. After these two steps, with the security analysis in Secure-RAM [17], the overall algorithm becomes an n-party SMPC algorithm that preserves the private configurations of participating ASes.

Algorithm 2: Oblivious PriorityCompare

Input: r1,r2 are two routes with the same origin
Output: a boolean value indicating whether **r2** is more preferred than **r1**
Function `Oblivious-PriorityCompare()`:
  $C0 = r1.lp.equal(r2.lp)$
  $C1 = r1.len.equal(r2.len)$
  $C2 = r1.med.equal(r2.med)$
  $C3 = r1.ID.equal(r2.ID)$
  $CompareRes =$
    $(1 - C0) * r.lp.greater\_than(r2.lp)$
  $+ C0 * (1 - C1) * r1.len.less\_than(r2.len)$
  $+ C0 * C1 * (1 - C2) * r1.med.less\_than(r2.med)$
  $+ C0 * C1 * C2 * (1 - C3) * r1.ID.less\_than(r2.ID)$
  return CompareRes;

A naive approach to make Algorithm 1 data-oblivious is to apply oblivious random access machine (ORAM) technique [5] to it. However, it would result in high computation and communication overhead due to the large amounts of padding and obfuscation operations in ORAM. In contrast, we transform Algorithm 1 to DO-Simulation with two key designs.

First, we leverage domain knowledge during the simulation to encode key data (*e.g.*, route map, route announcements, and RIBs) as vectors and matrices, which are data-oblivious data structures. As such, subroutines of Algorithm 1 (*e.g.*, *Import*, *Compare* and *Export*) can be rewritten as operations of linear scans of these data structures. Second, for each conditional statement (*i.e.*, if) in Algorithm 1, we let DO-simulation execute both branches to ensure that the execution of these statements is also data-oblivious. As an example, we illustrate our design using the oblivious priority-compare algorithm, a simple component in DO-Simulation that decides which route has a higher priority, in Algorithm 2. More examples of key components in DO-Simulation can be found in our technical report [31].

Accelerating the simulation. Making an algorithm into an SMPC version would cause high overhead. As such, we leverage the design of FASTPLANE [22] to accelerate DO-Simulation. The key idea is to let ASes with global optimal route announcements send their updates first. For monotonic networks, this design can generate the data plane efficiently by choosing a proper propagation order. However, FASTPLANE can not be applied in networks that are not monotonic. It is because the route withdrawal will occur when simulating according to the propagation order mentioned above

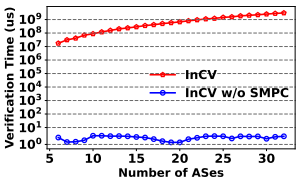

Figure 3: The verification time of InCV, compared with the non-SMPC version.

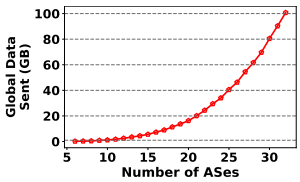

Figure 4: The global data sent of InCV on different networks.

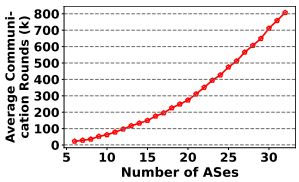

Figure 5: The communication rounds between parties on different networks.

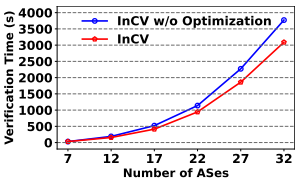

Figure 6: The verification time of InCV, compared with the version without optimization.

and FASTPLANE does not implement the withdrawal operation. To support the non-monotonic networks, we implement the route withdrawal operation on top of FASTPLANE. We note that in rare cases, such as when each router prefers the route with longer paths, this design may cause an efficiency decrease. However, experiments under all our datasets show that in most cases it can accelerate the simulation process by reducing the frequency of route withdrawals.

Current limitations. As a proof of concept, we have not yet included the implementation of more complex operations (*e.g.*, regular expression matching) in DO-Simulation. There is no theoretical obstacle preventing us from doing that. We will include these operations in our next version of the prototype.

3.3 DO-DPV: Data-Oblivious Verification

After DO-Simulation computes a secret network data plane in an (IP prefix, AS path) mapping that is stored by participating ASes' agents as a (t, n)-Shamir secret, the DO-DPV algorithm verifies network properties (*e.g.*, reachability, waypointing, blacklisting) using a DFS-based traversal algorithm to traverse from source AS to the destination AS holding the destination IP prefix. To accelerate DO-DPV, we construct an (IP-prefix, integer) mapping before the whole verification process starts. As such, DO-DPV does not need to perform any IP-prefix matching but only integer comparison. We will include IP-prefix matching in DO-DPV to handle verification queries on more specific IP prefixes.

3.4 Security Analysis

The security property of InCV follows the standard SMPC analysis under the semi-honest security model: the privacy of BGP configurations of n participating ASes will be preserved as long as less than $n/2$ ASes are corrupted.

4 EVALUATION

In this section, we target at showing the capability of InCV. We achieve this by making 3 comparisons. First, we make a comparison between InCV and its clear-text version (an implementation that does rely on SMPC) to show the overhead of SMPC. With this, we can also evaluate the feasibility of InCV. Second, we make a comparison between InCV and the verification tool Batfish [14] to evaluate if InCV can generate correct verification results while preserving privacy. Finally, we make a comparison on InCV before and after optimization. And the result suggests that our optimizations greatly improve efficiency in monotonic networks.

Testbed. We implement our system on MP-SPDZ [21], a versatile framework for multi-party computation. And all the experiments are run on a Linux server with kernel version 5.4.0 and two Intel Xeon Silver 4210R 2.40GHz CPUs and 128GB of DDR4 DRAM.

Dataset. We use 27 synthesized networks ranging from 6 to 32 nodes as datasets. For each topology, we synthesize configurations of each node that meets initial connection requirements using NetComplete [12]. All configurations are written in Cisco's IOS language.

Overhead of SMPC in InCV. To illustrate the overhead of SMPC in InCV, we additionally compose our clear-text verifier in Java by implementing our Algorithm 1 in the natural way (where the solver collects all configurations and run the simulation on the parsed intermediate representations). Figure 3 shows the running time for both verifiers. Figure 4 and Figure 5 show the global data sent and rounds of communication for InCV on networks scaling up to 32 ASes. The results show that achieving data-oblivious SMPC in interdomain verification is quite costly in time, as agents have to communicate hundreds of thousands of times on end. However, for smaller topologies, the overhead is still acceptable, at ~52 minutes for 32 ASes.

Accuracy. To confirm the accuracy of Algorithm 1, we compare the results of both InCV and its clear-text version with that of Batfish [14]. For all 27 networks, the verifiers generate consistent results with Batfish [14].

The performance of simulation optimization. To show the effect of the optimization technique based on FASTPLANE. We also make a comparison between the InCV before and after optimization on the synthesized networks. The comparison is plotted in Figure 6. It shows that ~19% acceleration has been achieved for the network with 32 ASes, due to fewer route withdrawals and fewer iterations during the simulation. However, some extra experiments for networks with complicated policies are required to better understand the capability of this optimization.

5 DISCUSSION

We discuss several open research questions in regard to interdomain network configuration verification.

Scaling the verification. The preliminary evaluation of InCV shows that it may take up to 3089 seconds (*i.e.*, 51 minutes) to verify an interdomain network of 32 ASes. In contrast, real-world interdomain networks have a larger scale, *e.g.*, the LHC science network has over 150 ASes and the Internet has over 60 thousand ASes. As such, for interdomain network configuration to be practical, we need to design optimizations to scale verification computation for larger networks. One direction toward this goal is to improve both DO-Simulation and DO-DPV by designing a customized and more-efficient SMPC protocol. A second approach is to design a more compact encoding of BGP configurations, *e.g.*, increase the level of abstraction of BGP route attributes [8], without compromising the correctness of the verification. In addition, whether certain intermediate results can be revealed during the verification without affecting the privacy-preserving of ASes' BGP configuration is also a possible way to improve the system's scalability.

Supporting more complex BGP route attributes and interdomain network properties. As a proof of concept of interdomain network configuration verification, InCV assumes that route reachability equals forwarding reachability and only implements limited BGP route attributes (*e.g.*, local preference). As such, how to verify interdomain BGP configurations containing policies on more complex BGP route attributes (*e.g.*, community, multiple exit discriminator, and AS set) is one of our ongoing investigations. In addition, we are also looking into how to extend DO-Simulation with the symbolic route [32, 35] to efficiently verify network properties under k-link-failure scenarios.

Incremental configuration verification. The current design of InCV does not support incremental verification. As such, when an AS wants to update its configurations, InCV has to rerun DO-simulation and DO-DPV from scratch to verify the interdomain network, which is time-consuming and may not be necessary. How to verify network configurations incrementally is an important yet not well-studied question for both interdomain networks and single-domain networks. DNA [36] takes a first step toward incremental network configuration verification. However, extending it to a privacy-preserving interdomain setting may be difficult because DNA builds its simulation on datalog, a programming paradigm whose extensibility for secure multi-party computation is still less understood. The approach we are currently studying is to extend DO-Simulation to an incremental setting. The critical challenge of this approach lies in how to store and access the intermediate results in a privacy-preserving yet efficient way.

Incremental deployment. Similar to interdomain routing protocols, incremental deployment is also a critical challenge for interdomain network configuration verification. Although we present the design of InCV assuming the participation of all ASes in an interdomain network, InCV supports incremental deployment. For example, some ASes can use InCV to collaboratively verify network properties that only involve themselves by treating other neighboring ASes' route announcements as external variables. After other ASes witness the efficacy of InCV in preventing interdomain configuration errors, they may be more willing to participate in the operation of InCV.

Stronger security models. Although we believe that the semi-honest security model adopted in designing InCV is sufficient for normal operations of interdomain networks, where operators are willing to help each other prevent network errors, verifying interdomain network configurations under stronger security models [10] (*e.g.*, malicious adversaries and covert adversaries model) is still an important and practical research question in security-critical scenarios such as financial networks. We plan to study this issue after we address other open questions discussed earlier.

6 CONCLUSION

In this paper, we advocate that interdomain network configuration verification is an important and emerging research area. We take a first step to bring privacy to network configuration verification and design InCV, an interdomain network configuration verification system where participating ASes' configurations are private. By leveraging SMPC, InCV achieved privacy-preserving configuration verification in a complicated interdomain network. Experiments show that InCV can verify an interdomain network of 32 ASes with reasonable computation and communication overhead. This preliminary work shed light on many open research questions such as scalability, incremental verification, and stronger security guarantee of interdomain network configuration verification.

Acknowledgments. We are extremely grateful for our shepherd, Ennan Zhai, and the anonymous APNet'23 reviewers for their wonderful feedback. We also thank Timos Antonopoulos, Hongyu Du, Franck Le, Ning Luo and Ruzica Piskac for their help and suggestions during the preparation of this paper. This work is supported in part by the National Key R&D Program of China 2022YFB2901502, NSFC Award #62172345, MOE of China Award #2021FNA02008, Open Research Projects of Zhejiang Lab #2022QA0AB05, and NSF Fujian China 2022J0-1004.

REFERENCES

[1] Anubhavnidhi Abhashkumar, Aaron Gember-Jacobson, and Aditya Akella. 2020. Tiramisu: Fast Multilayer Network Verification. In *NSDI'20*. USENIX, 201–219.

[2] Lawrence Abrams. 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/.

[3] Lawrence Abrams. 2021. Facebook, Instagram, and WhatsApp Back Online after BGP Fix. https://www.bleepingcomputer.com/news/technology/facebook-instagram-and-whatsapp-back-online-after-bgp-fix/.

[4] David Afolayan. 2018. How Bad is MainOne's BGP Error and Why They Must Prevent a Recurrence. https://technext24.com/2018/11/15.

[5] Miklós Ajtai. 2010. Oblivious RAMs without Cryptographic Assumptions. In *STOC'10*. ACM, 181–190.

[6] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2017. A General Approach to Network Configuration Verification. In *SIGCOMM'17*. ACM, 155–168.

[7] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2018. Control Plane Compression. In *SIGCOMM'18*. ACM, 476–489.

[8] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2019. Abstract Interpretation of Distributed Network Control Planes. In *POPL'19*. ACM, 1–27.

[9] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. 1988. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *STOC'88*. ACM, 1–10.

[10] Ran Canetti. 2001. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS'01*. IEEE Computer Society, 136–145.

[11] CERN. 2023. The Large Hadron Collider (LHC) Experiment. https://home.cern/topics/large-hadron-collider.

[12] Ahmed El-Hassanyr, Petar Tsankov, Laurent Vanbever, and Martin T Vechev. 2018. Netcomplete: Practical Network-wide Configuration Synthesis with Autocompletion. In *NSDI'18*. USENIX, 579–594.

[13] 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 *OSDI'16*. USENIX, 217–232.

[14] 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 *NSDI'15*. USENIX, 469–483.

[15] Aaron Gember-Jacobson, Raajay Viswanathan, Aditya Akella, and Ratul Mahajan. 2016. Fast Control Plane Analysis Using an Abstract Representation. In *SIGCOMM'16*. ACM, 300–313.

[16] Oded Goldreich. 1987. Towards a Theory of Software Protection and Simulation by Oblivious RAMs. *Journal of the Acm* 43, 3, 431–473.

[17] Steven Dov Gordon, Jonathan Katz, Vladimir Kolesnikov, Fernando Krell, Tal Geula Malkin, Mariana Raykova, and Yevgeniy Vahlis. 2012. Secure Two-party Computation in Sublinear (amortized) Time. In *CCS '12*. ACM, 513–524.

[18] Andreas Haeberlen, Ioannis C Avramopoulos, Jennifer Rexford, and Peter Druschel. 2009. NetReview: Detecting When Interdomain Routing Goes Wrong.. In *NSDI'09*. USENIX, 437–452.

[19] 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 *SIGCOMM'19*. ACM, 200–213.

[20] Marcel Keller and Peter Scholl. 2014. Efficient, Oblivious Data Structures for MPC. In *ASIACRYPT'14*. Springer, 506–525.

[21] Marcel Kellerl. 2020. MP-SPDZ: A Versatile Framework for Multi-Party Computation. In *CCS'20*. ACM, 1575–1590.

[22] Nuno P. Lopes and Andrey Rybalchenko. 2019. Fast BGP Simulation of Large Datacenters. In *VMCAI'19*. Springer, 386–408.

[23] Takashi Nishide and Kazuo Ohta. 2007. Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol. In *PKC'07*. Springer, 343–360.

[24] Rafail Ostrovsky. 1990. Efficient Computation on Oblivious RAMs. In *STOC'90*. ACM, 514–523.

[25] Santhosh Prabhu, Kuan Yen Chou, Ali Kheradmand, Brighten Godfrey, and Matthew Caesar. 2020. Plankton: Scalable Network Configuration Verification Through Model Checking . In *NSDI'20*. USENIX, 953–967.

[26] Yakov Rekhter and Tony Li. 1995. A Border Gateway Protocol 4 (BGP-4). *RFC Editor*. https://doi.org/10.17487/RFC1771

[27] A. Shamir. 1979. How to Share a Secret. *Commun. ACM* 22, 11, 612–613.

[28] 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, 1814–1827.

[29] Konstantin Weitz, Doug Woos, Emina Torlak, Michael D Ernst, Arvind Krishnamurthy, and Zachary Tatlock. 2016. Scalable Verification of Border Gateway Protocol Configurations With an SMT Solver. In *OOPSLA'16*. ACM, 765–780.

[30] Huisan Xu. 2022. Network Configuration Survey. https://mailman.nanog.org/pipermail/nanog/2022-November/220861.html.

[31] Huisan Xu, Qiuyue Qin, Xing Fang, Qiao Xiang, and Jiwu Shu. 2023. InCV-TR.pdf. http://sngroup.org.cn/publication.html.

[32] Rulan Yang, Xing Fang, Lizhao You, Qiao Xiang, Hanyang Shao, Gao Han, Ziyi Wang, Jiwu Shu, and Linghe Kong. 2023. Diagnosing Distributed Routing Configurations Using Sequential Program Analysis. In *APNET'23*. ACM, 85–92.

[33] Andrew Chi-Chih Yao. 1982. Protocols for Secure Computations (Extended Abstract). In *SFCS'08*. IEEE, 160–164.

[34] Andrew Chi-Chih Yao. 1986. How to Generate and Exchange Secrets. In *SFCS'86*. IEEE, 162–167.

[35] Fangdan Ye, Da Yu, Ennan Zhai, Hongqiang Harry Liu, Bingchuan Tian, Qiaobo Ye, Chunsheng Wang, Xin Wu, Tianchen Guo, Cheng Jin, Duncheng She, Qing Ma, Biao Cheng, Hui Xu, Ming Zhang, Zhiliang Wang, and Rodrigo Fonseca. 2020. Accuracy, Scalability, Coverage: A Practical Configuration Verifier on a Global WAN. In *SIGCOMM'20*. ACM, 599–614.

[36] Peng Zhang, Aaron Gember-Jacobson, Yueshang Zuo, Yuhao Huang, Xu Liu, and Hao Li. 2022. Differential Network Analysis. In *NSDI'22*. USENIX, 601–615.

[37] Peng Zhang, Dan Wang, and Aaron Gember-Jacobson. 2022. Symbolic Router Execution. In *SIGCOMM'22*. ACM, 336–349.

A DETAILS OF DO-SIMULATION

A.1 A Formal Definition for Data-oblivious Algorithms

Given a program Π. Let $\Pi(n, x)$ denote the execution of the program Π on an input x using n memory cells. Also, let $\tilde{\Pi}(n, x)$ denote the memory access pattern of $\Pi(n, x)$. $C(\Pi) = \Pi'$. μ denotes a function such that for any $n \in \mathbb{N}$, for any program Π and for any input $x \in \{0,1\}^*$, the probability that $\Pi'(n, x)$ outputs an overflow is at most $\mu(n)$.

In order to find such a $C(\Pi) = \Pi'$, which stands for the data-oblivious version of Π, the following Lemmas must be satisfied.

Lemma A.1. *Let $n \in \mathbb{N}$ and $x \in \{0,1\}^*$. Given a program Π, with probability at least $1 - \mu(n)$, the output of $\Pi'(n, x)$ is identical to the output of $\Pi(n, x)$. (Equivalence Lemma)*

Lemma A.2. *Given two programs Π_1 and Π_2 and two inputs $x_1, x_2 \in \{0,1\}^*$ such that $\left|\tilde{\Pi}_1(x_1, n)\right| = \left|\tilde{\Pi}_2(x_2, n)\right|$, with probability at least $1 - 2\mu(n)$, the access patterns $\tilde{\Pi}'_1(x_1, n)$ and $\tilde{\Pi}'_2(x_2, n)$ are identical. (Obliviousness Lemma)*

A.2 An Example Subroutine in DO-Simulation

Algorithm 3 illustrates how DO-Simulation manages to update ASes' RIBs obliviously. For concreteness of exposition, we use notation $[x]$ to indicate that the value of x is kept *secret* by the Shamir's secret sharing scheme.

In general, this algorithm uses the idea similar to Insertion sort to realize the function. $RIB(n)[i]$ stands for the i_{th} entry in the RIB of AS n.

Algorithm 3: Oblivious InsertToRIB

Input: **RIB(n)** is the RIB of AS n, **R** is the routing information to be processed, **MAX_SIZE** is the pre-defined maximum volume of a RIB.
Output: **R** is inserted into proper position in **RIB(n)**

```
[Position] ← 0;
[UpdateFlag] ← 0 ;
for i ← 0 to MAX_SIZE − 1 do
    [CompareRes] =
    PriorityCompare([R],[RIB(n)[i]]).greater_than(0)
    [Position] =
    [Position] * (1-[CompareRes]) + (i+1) * [CompareRes]
for i ← MAX_SIZE − 2 to 0 do
    [UpdateFlag] = 1 - i.less_than([Position])
    [RIB(n)[i+1]] =
    [RIB(n)[i+1]]*
    (1-[UpdateFlag])+[RIB(n)[i]]*[UpdateFlag]
    [RIB(n)[i]] =
    R * i.equal([Position])+[RIB(n)[i]]*(1-i.equal([Position]))
```