Privacy-Preserving Collaborative Verification Protocols

Andreas Haeberlen[†] Mingchen Zhao[†] Wenchao Zhou[†]
Alexander J. T. Gurney[†] Micah Sherr[‡] Boon Thau Loo[†]

[†] *University of Pennsylvania* [‡] *Georgetown University*

Abstract

We outline our vision of large-scale distributed systems that efficiently combine privacy and verifiability. In the systems we envision, a group of nodes would be able to verify that a given node N_i has been executing a given algorithm $A(N_i)$. If successful, the verification would not reveal any additional information.

1 Introduction

Security and privacy are both important goals for large-scale distributed systems. In systems with hundreds of nodes that are spread across different administrative domains, it is not uncommon that some nodes fail to correctly perform their assigned functions – e.g., because they have been manipulated or compromised by an adversary, or simply because of hardware defects or software bugs – and such faults are not always easy to see from a single domain. It would be useful if nodes from different domains could verify *collaboratively* that each node in the system is working as expected. However, this entails an exchange of additional information, which not all domains are comfortable with.

This tension between security and privacy is visible today, e.g., in the Internet's interdomain routing system. Internet routers can fail for a variety of reasons, and this results in a substantial amount of connectivity loss [6]. Solutions for detecting large and general classes of faults are available today [4], but they are being criticized for requiring ISPs to disclose additional information, e.g., the routing messages they have been exchanging with other ISPs, since this could potentially reveal sensitive information about an ISP's network setup or its routing policies.

At first glance, this dilemma between security and privacy seems hard to avoid: how can we tell whether a node is working correctly, without knowing what specifically the node did? Thus, most existing solutions have chosen one or the other: they either provide good privacy, with correspondingly weak security guarantees, or they choose to trade some privacy for better security.

However, there is evidence that applications do not *have* to make this choice: it is possible, in a sense, to have your cake and eat it too. Next, we state this problem more formally, and we present initial evidence that security and privacy can be combined efficiently.

2 The Private Verification problem

We consider a system with a set of nodes $N := \{E, C_1, \ldots, C_k\}$ that can communicate via message passing. An arbitrary subset $S \subseteq N$ of the nodes may be controlled by a malicious adversary. Let A be the algorithm that E is expected to use. We say that E is *correct* in a finite execution e of this system if the steps it has taken in e conform to $A(E)$; otherwise we say that E is *faulty*. $V(N, A, E)$ is a *private verification protocol* if it has the following properties when it is run after any finite execution e:

- **Nontriviality:** Every correct node C_i eventually outputs either `OK` or `FAIL`.
- **Detection:** If E is detectably faulty in e and $S = \{E\}$,[1] all correct C_i will output `FAIL`.
- **Accuracy:** If E is correct in e, each correct C_i will output `OK`.
- **Confidentiality:** If $V(N, A, E)$ is run and E is correct, no C_i learns any information *it did not already know before the run*, except that E is correct.

Briefly, E is detectably faulty if its network-visible behavior, as observed by the correct nodes, is inconsistent with any correct execution of $A(E)$. For instance, if E sends a single bad message m to another faulty node C_j but otherwise follows $A(E)$, E is faulty but not detectably faulty, since C_j may choose not to reveal m to the other nodes. See [5] for a more formal definition.

3 Strawman solutions

It is possible to approximate private verification using heavyweight cryptographic techniques. For instance, we can use general-purpose zero-knowledge proofs (ZKPs) [2]; however, the high overhead of ZKPs would be prohibitive for many applications. Another approach is to have the C_i compute E's expected outputs via secure multi-party computation (MPC) [1], and to then compare these outputs to the ones that were actually observed. But MPC is practical only for very simple functions. Moreover, this approach seems wasteful because private verification is a simpler problem than MPC: it merely requires the *verification* of a computation that was already performed by E.

[1]This definition is just a first step; it would be interesting to allow larger subsets of N here.

4 Approach: Collaborative verification

To enable private verification for realistic applications, we are exploring a different approach. We observe that each of the C_i already knows *some* facts about E's execution: it may have sent certain messages to E, and it may have received certain other messages from E. Moreover, the C_i collectively know the *entire* network-visible behavior of E. Hence, we can try to break the overall task of verifying E's execution into smaller subtasks, such that a) each subtask can be completed by at least one C_i, using only on information that C_i already knows, and b) successful verification of all subtasks implies successful verification of E's entire execution. We call the result a *collaborative verification protocol (CVP).*

4.1 Sketch of a generic CVP

To see that this is possible at least in principle, consider the following sketch of a simple (but highly impractical) protocol. E enumerates all the finite execution prefixes it could have completed by the time verification is triggered. It then instantiates a bit vector with one bit for each execution prefix, and sets all bits to zero except for the one that corresponds to the actual execution. E then constructs a Merkle hash tree [7] over the bits, signs the top-level hash value, and publishes it to all the C_i; this ensures that equivocation can be detected. Next, E determines, for each C_i, the set of executions that are inconsistent with the messages C_i has exchanged with E earlier, and it proves to each C_i that the corresponding bits are zero (by revealing the hashes along the path from the bits to the root). If a correct C_i does not receive a proof for a bit that is inconsistent with what it has seen, it broadcasts a challenge to the other C_i; if a bit is not zero, the corresponding proof is a proof of misbehavior.

If all C_i complete this step correctly, they have collectively ruled out all executions that would be inconsistent with E's observable behavior [5]. However, no C_i has learned anything it did not already know before the verification: if a good hash function is used, it is infeasible to learn the values of the bits from the tree's internal hash values, and the recipients of the bit proofs already know that the bits must be zero if E is correct, so they do not learn anything new from them. If verification succeeds, the only new information is that E is correct.

This approach seems attractive because it requires little cryptographic mechanism; all that is needed is a good hash function and a bit of public-key cryptography. However, it is clearly impractical to reserve a bit for *every* possible prefix of E's execution, so an obvious challenge is to find a more compact representation of this set. This is not straightforward because the representation must itself be private, i.e., its structure must not reveal any private information.

5 Status

As an existence proof that such representations can be found for practical applications, we have developed PVR, a collaborative verification system for the Internet's interdomain routing protocol [3, 8]. We have proven that PVR can provide a variant of the guarantees in Section 2, and we have shown that its overhead is low enough to be run for current routing table sizes in real time, and on commodity hardware. The protocol from [8] is domain-specific, but we have since developed generic variants that are currently under submission: one of them can be used for any $A(E)$ that evaluates a predicate on boolean inputs provided by the C_i, and another works for any $A(E)$ that can be formulated as a deterministic finite automaton.

6 Ongoing work

These protocols are evidence that private verification is general, and none of them use expensive cryptographic primitives. However, before CVPs can be widely applicable, several challenges remain to be solved, e.g.:

Optimizations: State machines are general but not necessarily efficient; for instance, a naïve state machine for a 64-bit counter would have 2^{64} states! Fortunately, realistic state machines tend to have a regular structure that can be exploited for optimizations. We have already developed one such optimization for numeric variables and simple arithmetic operations on them.

Stronger threat models: Our current protocols require the assumption that nodes do not collude; we would like to strengthen our protocols and remove this assumption.

Tool support: We are developing a protocol generator that can automatically produce CVPs from a specification of $A(E)$ written in a simple programming language.

References

[1] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. ACM STOC*, 1987.

[2] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38:690–728, 1991.

[3] A. Gurney, A. Haeberlen, W. Zhou, M. Sherr, and B. T. Loo. Having your cake and eating it too: Routing security with privacy protections. In *Proc. HotNets*, Nov. 2011.

[4] A. Haeberlen, I. Avramopoulos, J. Rexford, and P. Druschel. NetReview: Detecting when interdomain routing goes wrong. In *Proc. NSDI*, Apr 2009.

[5] A. Haeberlen and P. Kuznetsov. The Fault Detection Problem. In *Proc. OPODIS*, Dec. 2009.

[6] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP misconfiguration. In *Proc. ACM SIGCOMM*, Sep 2002.

[7] R. Merkle. Protocols for public key cryptosystems. In *Proc. Symposium on Security and Privacy*, Apr. 1980.

[8] M. Zhao, W. Zhou, A. Gurney, A. Haeberlen, M. Sherr, and B. T. Loo. Private and verifiable interdomain routing decisions. In *Proc. SIGCOMM*, 2012. (accepted, pending shepherd approval).