A Case for Remote Attestation in Programmable Dataplanes

Nik Sultana
Illinois Institute of Technology

Deborah Shands
SRI International

Vinod Yegneswaran
SRI International

ABSTRACT

Programmability is a double-edged sword. It can better tailor solutions to problems, optimize resource use, and inexpensively patch deployed equipment. But programmability can also be abused to undermine the security of hardware and that of its unwitting users. Remote Attestation (RA) is a class of techniques to provide integrity assurance to remote users of resources such as hardware, OSs and applications. It is used to establish well-defined trust relationships among mutually distrustful principals who provide, use or delegate remote resources. RA could benefit, for example, tenants of a datacenter or users of IoT equipment such as health monitors.

This position paper considers how RA can be used to enable dynamic assessments of network security characteristics through automated generation, collection, and evaluation of rigorous evidence of trustworthiness. We introduce a set of use cases, sketch how the Copland and NetKAT languages can be combined and extended to make network-aware attestation policies, and propose an extension of P4-programmable hardware to enforce this mechanism in the network.

CCS CONCEPTS

• **Networks** → **Network security**;

KEYWORDS

Remote Attestation, Programmable Networking

ACM Reference Format:
Nik Sultana, Deborah Shands, and Vinod Yegneswaran. 2022. A Case for Remote Attestation in Programmable Dataplanes. In *The 21st ACM Workshop on Hot Topics in Networks (HotNets '22), November 14–15, 2022, Austin, TX, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3563766.3564100

HotNets '22, November 14–15, 2022, Austin, TX, USA

ACM ISBN 978-1-4503-9899-2/22/11…$15.00
https://doi.org/10.1145/3563766.3564100

1 INTRODUCTION

Programmability is a double-edged sword. It can better tailor solutions to problems, optimize resource use, and inexpensively patch deployed equipment. But programmability can also be abused to undermine the security of hardware and that of its unwitting users.

The abuse of programmable network equipment was central to the "Athens Affair" [20, 24], a cyberattack on a cellular network operator that "targeted the conversations of specific, highly placed government and military officials." This enabled eavesdropping on the private communications of the prime minister of Greece and least 100 other high-ranking officials. The attacker patched software running on programmable network equipment to duplicate digitized voice data streams associated with a specific list of phone numbers and direct the duplicate streams to other cellular phones, enabling eavesdropping. The rogue software patch activated existing, but unused, lawful-intercept functions of the equipment. The operators of the network were unaware that their equipment had been subverted. The attack came to light only by accident, when an upgrade resulted in a noticeable malfunction. *The pervasive programmability of modern wired and wireless networks is risky without rigorous assurance of exactly which programs are running on the network equipment.*

Remote Attestation (RA) [9] could have helped the network operator targeted in the Athens Affair to detect that rogue software was running on its equipment. RA is a class of techniques for validating remote resources, such as the remote execution of a program and the hardware on which it is executing. These techniques gather evidence to assure users of the integrity of the resources with which they are interacting. RA is orthogonal to *program verification*: RA can verify a claim that a specific program is running without verifying claims about the program's correctness.

In this paper, we sketch a path to tackle both the *specification* and the *mechanism* for RA on programmable dataplanes. We describe a specification approach that combines the state-of-the-art approaches of Copland [14] and NetKAT [2] which are used to reason about RA and SDN respectively. For the mechanism, we describe an extension of the Protocol Independent Switch Architecture (PISA) [7] that can participate in RA protocols by attesting its code as sketched in Fig. 2.

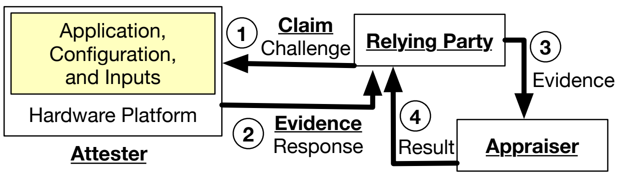

Figure 1: Principals in Remote Attestation. The Relying Party trusts a remote system based on evidence.

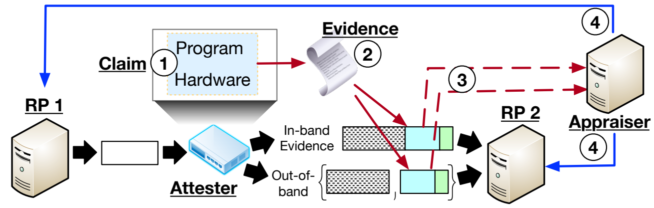

Figure 2: PERA: "PISA Extended with RA" (§5). Fig. 1 can be instantiated to leverage this in various ways.

The security of programmable networking equipment has not yet caught up with its flexibility, and research is needed to address the gap. Unfortunately, the security risks introduced by programmability can undermine the benefits of using programmable network hardware. For example, there has been excellent progress on leveraging programmable networking for monitoring [15, 16, 25], but without RA an adversary can replace a monitoring program with one that produces false readings, perhaps as part of a Denial-of-Service or Confused Deputy attack on the network being monitored.

The networking community needs fundamental techniques for using RA on programmable network hardware. Theoretical and practical RA techniques developed for other targets offer helpful stepping stones. For example, practical RA techniques for hosts [5, 10] can be reused for programmable network hardware (e.g., for securing the boot sequence and providing secure signing and verification of data.) Similarly, abstractions [9] and reasoning techniques about confinement [22] can be repurposed for programmable dataplanes.

Because existing RA techniques abstract away the network and focus on peers, new techniques are needed to enable dynamic, automated assessment of security-critical characteristics of networks implemented on programmable hardware. The central hypothesis in this position paper is that **RA can be used to enable dynamic assessments of network security characteristics through automated generation, collection, and evaluation of rigorous evidence of trustworthiness**.

A realization of the hypothesis is sketched in the remainder of the paper which introduces use cases, describes how NetKAT and Copland can be combined to express the use cases, and discusses the extension of a P4 [6] switch to produce and consume evidence. Related work is mentioned throughout the paper.

2 MOTIVATING USE CASES

We describe motivating practical situations in which having RA-capable programmable dataplanes improves the security of network users and operators.

UC1: *Configuration Assurance.* Using the wrong dataplane program can have serious consequences for the operator and its users. For example, using the wrong firewall, forwarding table, or load-balancer could degrade the network's security and performance. In this use case, RA protects against unvetted or unwanted dataplane programs that might have been mistakenly or deliberately swapped for the intended version. In the "Athens Affair" from §1, the equipment's software was deliberately modified to exfiltrate traffic, but it could also have been modified to malfunction and cause a denial-of-service.

Using RA in this use case would involve producing evidence at some granularity and frequency (at most, per hop and per packet) along the paths of a network flow, and giving peers a signed and suitably redacted form of that evidence. For example, the evidence for a packet p could indicate that p reached switch S_1[1] on a specific network port, and was processed by `firewall_v5.p4`[2] and forwarded to S_2 which was running `ACL_v3.p4`, that then forwarded p to a DPI appliance that forwarded it back to S_2, that forwarded the packet to the current node.

UC2: *(Authentication) Path Evidence as a Security Factor.* The evidence gathered along a path can be used as a factor for authentication [1]. For example, a user that forgets their password or connects from a new device could be permitted limited access to a resource if they can prove that they are connecting from their home via an acceptable network path. This evidence can be used to weakly authenticate different peers and complement other authentication methods.

In addition to chaining together evidence of the forwarding decisions made on programmable dataplanes, this evidence can attest to other packet processing or filtering. Our confidence in a path can be bolstered if the path has specific features (e.g., if it crosses a specific series of firewalls or appliances as discussed in **UC1**.)

UC3: *(Authorization) Path Evidence as a Tag.* In **UC2**, evidence was used for authentication. Evidence can also be used in *authorization* decisions that affect intra- or inter-domain

[1] Instead of revealing their actual serial number, switches could be assigned a per-user pseudonym by the operator.

[2] Programs can also be assigned pseudonyms that can be lifted by an auditor's request or court order.

handling of traffic. For example, the decision to forward packets could depend on whether those packet have been processed by a set of appliances, such as an IDS, firewall, and scrubber. Packets can be treated differently based on evidence that they have never left a particular segment of the network. RA can provide an evidential basis to use cases like those described for FlowTags [11].

Path evidence could be used for DDoS mitigation: while under attack, a network could drop traffic for which it lacks path-based evidence. The next section describes caching of evidence and varying the level of detail and sampling frequency of evidence to lower the overhead of providing and consuming evidence.

UC4: *(Auditing) Evidence as Documentation.* This use case has two complementary sub-cases: **(A)**: Programmable dataplanes can run filters that match packet types and fields [7] and distinguish flows [8]. These features can be used to characterize malware communications with command-and-control nodes [17], which in turn can fingerprint the presence of malware. This can be captured as evidence that is used to justify other actions, such as applying for a court order to deactivate that malware [12]. **(B)**: The subsequent action to deactivate the malware can also be documented in a similar way (i.e., as an appraisable set of network interactions) and stored for later use as evidence to show the limited and focused action that was taken to deactivate the malware, to prove compliance with the authorizing court order. Thus RA can be used to provide evidence on integrity, transparency, and compliance of network-related activities to third-parties at a later time [26].

A similar audit trail can be compiled as evidence when gathering data from sensors [13], such as those for atmospheric radiation sampling, as evidence of the data's provenance and to help ensure that the sensor traffic has not been spoofed.

UC5: *Cross-Referenced Attestation.* Evidence from host-based and network-based attestation could be composed together to offer a more complete picture of how host-based processes interacted with the network, and how the network processed their traffic. This can help detect and stop exfiltration attacks by checking whether outward traffic patterns have been authorized by an unmodified application.

This composed evidence could also be used to show that the host that produced a specific flow was running a specific version of a network stack or protocol implementation. This enables the enforcement of policies that are sensitive to the network behavior of software. For example, TLS packets that were produced by a verified implementation [3] could be allowed to leave the network, while packets produced by un-verified implementations are blocked.

Another application of this use case involves trusted redaction of evidence for compliance certification of information processing within a cloud environment: path evidence could be processed to redact details sensitive to the enterprise customer before giving the redacted evidence to a compliance officer. By using host-based RA, the customer can meet regulatory compliance obligations without disclosing unnecessary, sensitive information to the regulator.

3 THREAT MODEL

We assume that evidence-producing hardware components (e.g., those that initialize a chip or generate a digital signature) are trustworthy: they are correctly designed and manufactured to generate tamper-evident evidence. Larger components or products into which trustworthy components are integrated (e.g., switches, NICs) are not assumed to be trustworthy. Adversaries may attempt to exploit insecure intermediate nodes as well as limitations on hardware resources to potentially compromise path attestation. An adversary may also perform supply-chain or organizational insider attacks to actively interfere with hardware, software, and staff. RA leverages outputs from trusted components to derive security guarantees across distributed systems that are deployed on third-party nodes assembled by untrusted manufacturers and run by untrusted operators (and their employees).

The technique described in this paper enables verification of the integrity of dataplane programs and their state. This does not, by itself, ensure confidentiality, integrity, and availability of data processed by a programmable dataplane, but it supports improved confidence in the dataplane programs that are processing data. Our technique does not overcome physical insecurity of a network device. For example, it does not protect against a malicious insider installing a passive tap to siphon network traffic, or installing an intermediary device to introduce network traffic between two network elements. RA complements other security techniques, such as physical access control and network protocols for end-to-end security, such as TLS and IPsec.

4 REMOTE ATTESTATION

Fig. 1 shows the main principals in RA. The **Relying Party** (RP) is the user of a remote program for which **Claims (1)** about its execution are met by **Evidence (2)** produced by the **Attester** on behalf of the platform executing that program. The RP presents this Evidence to an **Appraiser (3)** (sometimes referred to as the Verifier) which verifies the evidence to produce an **Attestation Result (4)**. More details can be found in specifications such as RATS [4, §7].

4.1 Reasoning about Network-aware RA

To use RA to reason about network communication of multiple parties over time requires a language that enables us to describe how networking equipment is to generate and process attestation evidence.

The language needs primitives to describe principals and to chain together processing steps across the network. To work in general network settings, the language needs some special primitives. The language must enable us to **(Prim1)** *abstract over paths*, since paths might not be knowable in advance. To reason about abstract paths, the language must enable us to **(Prim2)** *abstract over places*, since the identities of intermediate hops along a path might not be known to us. The language must also enable us to **(Prim3)** reason about *reachability*, and predicate a policy on a collector of evidence being reachable by producers of evidence.

The next section describes Copland, an existing language for reasoning about RA. Later we describe our proposal to extend Copland to provide primitives **(Prim1)**-**(Prim3)**.

4.2 The Copland Language

Copland separates the specification of RA attestation protocols from the enforcement mechanisms provided by specific hardware features. It has formal semantics [19] and a verified compiler toolchain [18]. The separation between policy and mechanism plays a central role in describing fundamental principles for *flexible* RA [14] that accommodates the needs of different use cases.

We introduce Copland's syntax by adapting an example from Rowe et al. [23], in preparation for the language extension described in the next section. This example is like the host-based portion of the "verified TLS implementation" part of use case **UC5** in §2: a banking website uses evidence about the client's browser extensions to check for malware that could steal the client's credentials. The client runs a bmon process which measures the exts process that represents the client's browser extensions. The bank also receives measurements from av, an antivirus program running in the client device's kernelspace.

Expressions in Copland describe *measurements* done by *principals* of specific *values*. Measurements are to be carried out in certain *places* as described by the expression. The results of measurements can be transformed and sent to other places, in composition with other measurements. Consider the example below:

$$* \mathsf{bank} \ : \ @_{\mathrm{ks}}[\overbrace{\mathsf{av\ us\ bmon}}^{C_1}] \overset{++}{\sim} @_{\mathrm{us}}[\overbrace{\mathsf{bmon\ us\ exts}}^{C_2}] \quad (1)$$

In expression (1), the overbraces are not part of Copland syntax, rather, we use them to label two Copland subexpressions C_1 and C_2. Here, $*R \ : \ C$ indicates that the unique outermost principal R is requesting evidence for expression C, R is the *relying party*, and the principals mentioned in C include *attesters* and *appraisers*.

In the banking website example shown in (1), the bank is requesting a compound measurement. Starting with the inner expression C_1 (i.e., "av us bmon") it means that the antivirus principal av is to measure principal bmon that is running in place us (userspace). The expression $@_P[C]$ means that measurement C must be carried out in place P—so the measurement C_1 would be carried out in ks (kernel space). C_1 is used to check that bmon has not been tampered with. C_2 uses bmon to measure the userspace-located exts, to scan for suspicious browser extensions.

Measurements in (1) are composed using "$\overset{lr}{\sim}$": this means that both measurements are carried out in parallel ("$\sim$"). The evaluation of a Copland expression takes in evidence that has been accrued so far and transforms it into composite evidence. The values of l and r indicate whether evidence accrued so far is passed to each arm of the composition. Symbol "+" means that evidence is passed on, and "–" means that evidence is not passed on. Thus "$C_1 \overset{--}{\sim} C_2$" is a parallel composition that only allows information to flow back from evaluating C_i.

An active adversary could replace bmon in userspace to always give a positive measurement, even when exts includes malware. Ramsdell et al. [21] describe how an active adversary who has userspace but not kernelspace control can cheat example (1) as follows: it first evaluates C_2 using the corrupt bmon, then it "repairs" bmon in userspace (replacing it with the non-corrupt version), and only then would it allow C_1 to take place. av would then indicate to bank that bmon is authentic. This can be mitigated by sequencing the two measurements using the "<" composition instead of the parallel composition, to make it more difficult for an adversary to "hide their tracks." Rowe et al. [23] describe adversary capability models in more detail.

Version (2) improves on version (1), using "$\overset{--}{<}$" as described above and adding two other features.

$$\begin{aligned} *\mathsf{bank} \ : \ @_{\mathrm{ks}}[\mathsf{av\ us\ bmon} \rightarrow !] \ \overset{--}{<} \\ @_{\mathrm{us}}[\mathsf{bmon\ us\ exts} \rightarrow !] \end{aligned} \quad (2)$$

The first feature introduces the "$C \rightarrow D$" operator to express that evidence produced by C is to be passed along to be processed by function D. The second feature introduces the signature operator "!" to express that the measurements from C_1 and C_2 are to be separately signed and returned to bank.

5 REMOTE ATTESTATION IN PROGRAMMABLE DATAPLANES

We first convert the example shown in Fig. 2 into Copland to show the language being used in a simple network setting. We then extend Copland to express **UC1**-**UC5** from §2.

We start with the *out-of-band* variant from Fig. 2: here the evidence is sent from the switch directly to the appraiser for certification. The switch first hashes (the # operator) and then signs (! operator) this evidence. The first relying party, RP1, receives direct evidence of this appraisal and certification. RP2 can later retrieve this evidence from the appraiser.

Note that both expressions are bound by n, a nonce parameter following Helble et al. [14]. This parameter is negotiated separately by RP1 and RP2. Example (3) below uses simplified syntax to reduce clutter by eliding obvious place details, and steps ①-④ from Fig. 2 are shown in blue to distinguish them from Copland syntax. Example (3) is described by two expressions that are evaluated in parallel:

$$
\begin{aligned}
&*\mathsf{RP1}, n \;:\; @_{\text{Switch}}[\mathsf{attest}\;\overbrace{(\mathsf{Hardware} \overset{-\,-}{\sim} \mathsf{Program})}^{\text{①: Claim}} \rightarrow \#\\
&\quad \rightarrow !]\ \overset{++}{>}^{\text{② \& ③: Evidence}} \quad @_{\text{Appraiser}}[\mathsf{appraise} \rightarrow \mathsf{certify}(n)\\
&\quad \rightarrow ! \rightarrow \mathsf{store}(n)]^{\text{④: Result}}\\
&*\mathsf{RP2}, n \;:\; @_{\text{Appraiser}}[\mathsf{retrieve}(n)]^{\text{④: Result}}
\end{aligned} \tag{3}
$$

The *in-band evidence* variant (4) is similar except that the evidence first reaches RP2 who then makes the appraisal request. Since RP2 learns the result of the appraisal directly, it does not need to separately enquire for a certificate as in the out-of-band variant. In this variant, we do not need a nonce to link the requests by the two Relying Parties (though a nonce can be used for freshness), and there is no need to store a certificate for later retrieval by another party. Unlike the previous example, this setting is described by a single expression. At the end of this process, both RP2 and RP1 would have received a signed certificate from the appraiser.

$$
\begin{aligned}
&*\mathsf{RP1} \;:\; @_{\text{Switch}}[\mathsf{attest}(\mathsf{Hardware} \overset{-\,-}{\sim} \mathsf{Program}) \rightarrow \# \rightarrow !]\\
&\quad \rightarrow @_{\text{RP2}}[@_{\text{Appraiser}}[\mathsf{appraise} \rightarrow \mathsf{certify} \rightarrow !]]
\end{aligned} \tag{4}
$$

5.1 Network-aware Copland

Copland expressions incorporate network topology details which restricts applicability to network settings where we do not have full topological visibility into the network. Even if they have knowledge of the network's topology, the forwarding path between two peers is typically chosen outside their control, and the path might change without warning due to routing changes. In the banking example from §4.2, neither the client nor the bank is likely to have a complete picture of how their traffic is being forwarded. The banking service might be running in a cloud, the internal topology of which is not disclosed to the bank. The client app might be running behind a NAT which hides details from the rest of the Internet.

We sketch an extension of Copland that incorporates features of NetKAT [2] to describe attestation specifications over networks that can include programmable dataplanes.

This Copland+NetKAT hybrid has the following features: **(Prim1)** The $\overset{*}{\Rightarrow}$ operator is based on NetKAT's Kleene star operator and provides path abstraction: the phrase on the left of this operator can hold for zero or more hops along the path. **(Prim2)** The $\forall$ operator provides place abstraction by relaxing the requirement to explicitly name places at the time of writing a policy. **(Prim3)** Reachability testing is provided through a combination of the ► operator and path abstraction. The ► operator adapts NetKAT's Boolean test prefix, and applies a Boolean test to a device before having it produce an attestation. A node (for which a ►-test holds) is reachable if there is a path leading to it. That node can also attest the result of the test. This test is done for two design reasons: to "fail early" and avoid the attestation effort, and to apply different attestations based on which Boolean test succeeds.

AP1 in Table 1 is an example of **UC5**. It adapts the bank example we saw earlier, the original parts of which are shown in blue. Here, the bank is the relying party (RP) as before. The nonce n and property X are RP-chosen parameters. X is some property that bank wishes to be attested at each hop—such as which P4 program and tables were used for forwarding. **AP1** also serves as example of **UC1** by extending the property X to include other configuration details.

Terms *hop* and *client* are abstract place names—the first will be used for each hop along a path, and the second will be used to refer to the end of the path. The phrase to the left of $\overset{*}{\Rightarrow}$ describes the gathering of evidence from each *hop* (which must satisfy test $\mathsf{K}_{\mathsf{hop}}$). Evidence is sent to the specific Appraiser place. $\mathsf{K}_{\mathsf{hop}}$ and $\mathsf{K}_{\mathsf{client}}$ ensure a pre-existing relationship between bank, *client* and each *hop*. This is not necessary, and is done to strengthen the specification.

Unlike **AP1**, **AP2** has a switch be the relying party. This policy is an example of **UC4**. Except for the use of ►, this could have been written entirely in Copland. In this example, P is a test being made on a packet. If the test succeeds then the test result is signed and sent to the Appraiser for storing.

AP3 is a more complex example that shows an attestation for a path that has specific functions (F_1, F_2) running in abstract places (p, q). An interesting feature of this example is that p passes its evidence to q before it reaches Appraiser, and between q and r we do not require nodes that support RA.

5.2 Executing RA Policies

After authoring an RA policy, how do we deploy it? The policy will be compiled by the Relying Party and serialized into an options header in the transport layer, to be evaluated along the path of traffic that it is sending out. The Relying Party will then query named places for evidence.

To interpret this policy on flows we envisage a device that we call PERA, for "PISA [7] switch Extended with RA". This switch will have access to specialized hardware primitives that can produce and consume evidence. These primitives might be integrated into the ASIC or might be remotely invoked by the programmable switch [27].

Fig. 3 sketches this hardware, noting that evidence might be sent in-band or out-of-band, as illustrated back in Fig. 2. Here, (A) and (D) show in-band evidence being received and sent by

$*\mathsf{bank}\langle n, X\rangle : \forall hop, client : (@_{hop}[\mathsf{K}_{\mathsf{hop}} \blacktriangleright \mathsf{attest}(n)X \rightarrow !] \overset{-+}{>} @_{\mathsf{Appraiser}}[\mathsf{appraise} \rightarrow \mathsf{store}(n)]]) \overset{*}{\Rightarrow} @_{client}[\mathsf{K}_{\mathsf{client}} \blacktriangleright @_{\mathsf{ks}}[\mathsf{av\ us\ bmon} \rightarrow !] \overset{--}{<} @_{\mathsf{us}}[\mathsf{bmon\ us\ exts} \rightarrow !]]$
AP1: Bank example from (2) but with path attestation between bank and client. The original expression is shown in blue.
$*\mathsf{scanner}\langle P\rangle : @_{\mathsf{scanner}}[P \blacktriangleright \mathsf{attest}(P) \rightarrow !] \overset{-+}{>} @_{\mathsf{Appraiser}}[\mathsf{appraise} \rightarrow \mathsf{store}]$
AP2: Example of **UC4**: a switch scans for a traffic pattern P. RA's audit trail can then be referenced by other actions.
$*\mathsf{pathCheck}\langle F_1, F_2, Peer1, Peer2\rangle : \forall p, q, r, peer1, peer2 : @_{peer1}[Peer1 \blacktriangleright !] \overset{-+}{>} @_p[\mathsf{attest}(F_1) \rightarrow !] \overset{-+}{>} @_q[\mathsf{attest}(F_2) \rightarrow !] \overset{-+}{>} @_{\mathsf{Appraiser}}[\mathsf{appraise} \rightarrow \mathsf{store}] \overset{*}{\Rightarrow} @_r[Q \blacktriangleright !] \overset{-+}{>} @_{peer2}[Peer2 \blacktriangleright !] \overset{-+}{>} @_{\mathsf{Appraiser}}[\mathsf{appraise} \rightarrow \mathsf{store}]$
AP3: Example combining **UC2** and **UC3**, involving attested dataplane programs and network path segments.

Table 1: Examples of Attestation Policies (APs) in Network-aware Copland.

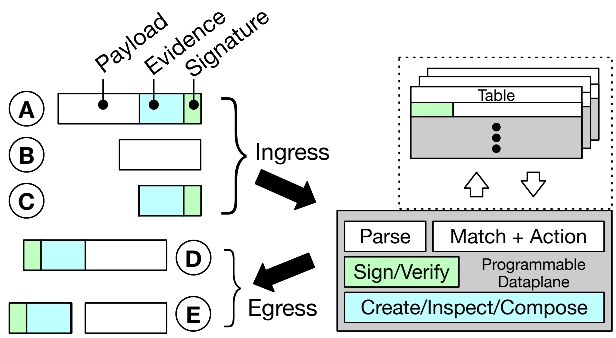

Figure 3: An RA-capable programmable switch. Evidence-handling is tuned to balance performance and security.

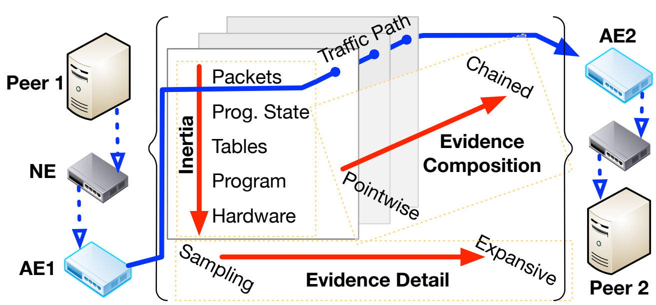

Figure 4: *Inertia*, *Detail* and *Composition* are the primary indices in our design space for PERA. The path between Peer 1 and Peer 2 includes Non-attesting Elements (NE) and Attesting Elements (AE).

the switch. Cases (B), (C) and (E) show evidence arriving and leaving separately. For some situations, it might be adequate to expect evidence to be gathered for each packet, and to add the RA policy to each packet. But in other situations, such per-packet overhead might be cumbersome and prohibitive.

In addition to the specification language and execution mechanism, we envisage a configuration interface that can tune the level of detail and frequency of evidence. Fig. 4 illustrates the main configuration choices. *Inertia* refers to the level of variability of attestable information across time: at one extreme, the model number of the hardware will not change, at the other extreme, a packet might be completely different than those that came before it. High-inertia attestations are more easily cached since they take longer to expire.

6 CONCLUSION

While programmable networking equipment offers great flexibility, networks implemented on such equipment are at risk from attacks that dynamically modify security-critical network behavior. This paper has sketched an approach for using RA techniques to enable dynamic assessment of network security characteristics. By extending the Copland RA policy language with elements of the NetKAT SDN programming language, we can define RA policies for programmable networks that specify the generation, collection, and evaluation of evidence of network program integrity. By implementing such RA policies, network dataplanes can participate in the process of proving their own trustworthiness.

ACKNOWLEDGMENTS

We thank the anonymous HotNets reviewers for their feedback. This material is based upon work supported by a Google Research Award, by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-19-C-0106, and by the National Science Foundation (NSF) under Grant No. ITE 2226443. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of funders.

REFERENCES

[1] [n. d.]. Multi-Factor Authentication. https://www.cisa.gov/mfa. ([n. d.]). Cybersecurity & Infrastructure Security Agency.

[2] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. 2014. NetKAT: Semantic Foundations for Networks. *SIGPLAN Not.* 49, 1 (jan 2014), 113–126. https://doi.org/10.1145/2578855.2535862

[3] Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue. 2017. A Messy State of the Union: Taming the Composite State Machines of TLS. *Commun. ACM* 60, 2 (jan 2017), 99–107. https://doi.org/10.1145/3023357

[4] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan. 2022. Remote Attestation Procedures Architecture. https://www.ietf.org/archive/id/draft-ietf-rats-architecture-17.html. (June 2022). RATS Working Group, IETF.

[5] Conor Black and Sandra Scott-Hayward. 2021. A Survey on the Verification of Adversarial Data Planes in Software-Defined Networks. In *Proceedings of the 2021 ACM International Workshop on Software Defined Networks & Network Function Virtualization Security (SDN-NFV Sec'21)*. Association for Computing Machinery, New York, NY, USA, 3–10. https://doi.org/10.1145/3445968.3452092

[6] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Comput. Commun. Rev.* 44, 3 (jul 2014), 87–95. https://doi.org/10.1145/2656877.2656890

[7] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. 2013. Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN. *SIGCOMM Comput. Commun. Rev.* 43, 4 (aug 2013), 99–110. https://doi.org/10.1145/2534169.2486011

[8] Xiaoqi Chen, Hyojoon Kim, Javed M. Aman, Willie Chang, Mack Lee, and Jennifer Rexford. 2020. Measuring TCP Round-Trip Time in the Data Plane. In *Proceedings of the Workshop on Secure Programmable Network Infrastructure (SPIN '20)*. Association for Computing Machinery, New York, NY, USA, 35–41. https://doi.org/10.1145/3405669.3405823

[9] George Coker, Joshua Guttman, Peter Loscocco, Amy Herzog, Jonathan Millen, Brian O'Hanlon, John Ramsdell, Ariel Segall, Justin Sheehy, and Brian Sniffen. 2011. Principles of Remote Attestation. *International Journal of Information Security* 10, 2 (2011), 63–81.

[10] Mihai Valentin Dumitru, Dragos Dumitrescu, and Costin Raiciu. 2020. Can We Exploit Buggy P4 Programs?. In *Proceedings of the Symposium on SDN Research (SOSR '20)*. Association for Computing Machinery, New York, NY, USA, 62–68. https://doi.org/10.1145/3373360.3380836

[11] Seyed Kaveh Fayazbakhsh, Luis Chiang, Vyas Sekar, Minlan Yu, and Jeffrey C. Mogul. 2014. Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions using FlowTags. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. USENIX Association, Seattle, WA, 543–546. https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/fayazbakhsh

[12] Eric Geller. 2022. How DOJ took the malware fight into your computer . https://www.politico.com/news/2022/06/13/how-doj-took-the-malware-fight-into-\your-computer-00038932. (2022). Politico.

[13] Andrew Gettelman, Alan J. Geer, Richard M. Forbes, Greg R. Carmichael, Graham Feingold, Derek J. Posselt, Graeme L. Stephens, Susan C. van den Heever, Adam C. Varble, and Paquita Zuidema. 2022. The future of Earth system prediction: Advances in model-data fusion. *Science Advances* 8, 14 (2022), eabn3488. https://doi.org/10.1126/sciadv.abn3488 arXiv:https://www.science.org/doi/pdf/10.1126/sciadv.abn3488

[14] Sarah C. Helble, Ian D. Kretz, Peter A. Loscocco, John D. Ramsdell, Paul D. Rowe, and Perry Alexander. 2021. Flexible Mechanisms for Remote Attestation. *ACM Trans. Priv. Secur.* 24, 4, Article 29 (sep 2021), 23 pages. https://doi.org/10.1145/3470535

[15] Hyojoon Kim, Xiaoqi Chen, Jack Brassil, and Jennifer Rexford. 2021. Experience-Driven Research on Programmable Networks. *SIGCOMM Comput. Commun. Rev.* 51, 1 (mar 2021), 10–17. https://doi.org/10.1145/3457175.3457178

[16] Srinivas Narayana, Anirudh Sivaraman, Vikram Nathan, Prateesh Goyal, Venkat Arun, Mohammad Alizadeh, Vimalkumar Jeyakumar, and Changhoon Kim. 2017. Language-Directed Hardware Design for Network Performance Monitoring. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 85–98. https://doi.org/10.1145/3098822.3098829

[17] Carlos Novo and Ricardo Morla. 2020. Flow-Based Detection and Proxy-Based Evasion of Encrypted Malware C2 Traffic. In *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security (AISec'20)*. Association for Computing Machinery, New York, NY, USA, 83–91. https://doi.org/10.1145/3411508.3421379

[18] Adam Petz and Perry Alexander. 2021. An Infrastructure for Faithful Execution of Remote Attestation Protocols. In *NASA Formal Methods Symposium*. Springer, 268–286.

[19] Adam Petz, Grant Jurgensen, and Perry Alexander. 2021. Design and Formal Verification of a Copland-Based Attestation Protocol. In *Proceedings of the 19th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE '21)*. Association for Computing Machinery, New York, NY, USA, 111–117. https://doi.org/10.1145/3487212.3487340

[20] Vassilis Prevelakis and Diomidis Spinellis. 2007. The Athens Affair. *IEEE Spectrum* 44, 7 (2007), 26–33. https://doi.org/10.1109/MSPEC.2007.376605

[21] John D. Ramsdell, Paul D. Rowe, Perry Alexander, Sarah C. Helble, Peter Loscocco, J. Aaron Pendergrass, and Adam Petz. 2019. Orchestrating Layered Attestations. In *Principles of Security and Trust*, Flemming Nielson and David Sands (Eds.). Springer International Publishing, Cham, 197–221.

[22] Paul D. Rowe. 2016. Confining Adversary Actions via Measurement. In *Graphical Models for Security*, Barbara Kordy, Mathias Ekstedt, and Dong Seong Kim (Eds.). Springer International Publishing, Cham, 150–166.

[23] Paul D. Rowe, John D. Ramsdell, and Ian D. Kretz. 2021. Automated Trust Analysis of Copland Specifications for Layered Attestations. In *23rd International Symposium on Principles and Practice of Declarative Programming (PPDP 2021)*. Association for Computing Machinery, New York, NY, USA, Article 23, 15 pages. https://doi.org/10.1145/3479394.3479418

[24] Bruce Schneier. 2006. Greek Wiretapping Scandal. https://www.schneier.com/blog/archives/2006/06/greek_wiretappi_1.html. (June 2006). Schneier on Security.

[25] John Sonchack, Adam J. Aviv, Eric Keller, and Jonathan M. Smith. 2018. Turboflow: Information Rich Flow Record Generation on Commodity Switches. In *Proceedings of the Thirteenth EuroSys Conference (EuroSys '18)*. Association for Computing Machinery, New York, NY, USA, Article 11, 16 pages. https://doi.org/10.1145/3190508.3190558

[26] Nik Sultana, Markulf Kohlweiss, and Andrew W. Moore. 2016. Light at the middle of the tunnel: middleboxes for selective disclosure of network monitoring to distrusted parties. In *Proceedings of the ACM*

SIGCOMM Workshop on Hot topics in Middleboxes and Network Function Virtualization, HotMiddlebox@SIGCOMM 2016, Florianopolis, Brazil, August, 2016, Dongsu Han and Danny Raz (Eds.). ACM, 1–6. https://doi.org/10.1145/2940147.2940151

[27] Nik Sultana, John Sonchack, Hans Giesen, Isaac Pedisich, Zhaoyang Han, Nishanth Shyamkumar, Shivani Burad, André DeHon, and Boon Thau Loo. 2021. Flightplan: Dataplane Disaggregation and Placement for P4 Programs. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, 571–592. https://www.usenix.org/conference/nsdi21/presentation/sultana