Flexible Mechanisms for Remote Attestation

SARAH C. HELBLE, Johns Hopkins University Applied Physics Laboratory, Laurel, MD
IAN D. KRETZ, The MITRE Corporation,
Bedford, MA
PETER A. LOSCOCCO, National Security Agency,
Fort Meade, MD
JOHN D. RAMSDELL, The MITRE Corporation,
Bedford, MA
PAUL D. ROWE, The MITRE Corporation,
Bedford, MA
PERRY ALEXANDER, The University of Kansas,
Lawrence, KS

Remote attestation consists of generating evidence of a system's integrity via measurements, and reporting the evidence to a remote party for appraisal in a form that can be trusted. The parties that exchange information must agree on formats and protocols. We assert there is a large variety of patterns of interactions among appraisers and attesters of interest. Therefore, it is important to standardize on flexible mechanisms for remote attestation. We make our case by describing scenarios that require the exchange of evidence among multiple parties using a variety of message passing patterns. We show cases in which changes in the order of evidence collection result in important differences to what can be inferred by an appraiser. We argue that adding the ability to negotiate the appropriate kind of attestation allows for remote attestations that better adapt to a dynamically changing environment. Finally, we suggest a language-based solution to taming the complexity of specifying and negotiating attestation procedures.

ACM Reference Format:
Sarah C. Helble, Ian D. Kretz, Peter A. Loscocco, John D. Ramsdell, Paul D. Rowe, and Perry Alexander. 2020. Flexible Mechanisms for Remote Attestation. 1, 1 (October 2020), 21 pages. https://doi.org/0000

1 INTRODUCTION

In distributed systems, a component must frequently make trust decisions about other components it relies upon. These decisions are often made implicitly, and by default with little system information. For example, in an internal corporate network, a printer may implicitly trust any device that has been granted access to the network. Remote attestation [Coker et al. 2011, 2008; Haldar et al. 2004] procedures allow a relying party to make its decision on the basis of evidence provided by its peer

Authors' addresses: Sarah C. HelbleJohns Hopkins University Applied Physics Laboratory, Laurel, MD, Sarah.Helble@jhuapl.edu; Ian D. KretzThe MITRE Corporation,
Bedford, MA, ikretz@mitre.org; Peter A. LoscoccoNational Security Agency,
Fort Meade, MD, palosco@tycho.nsa.gov; John D. RamsdellThe MITRE Corporation,
Bedford, MA, ramsdell@mitre.org; Paul D. RoweThe MITRE Corporation,
Bedford, MA, prowe@mitre.org; Perry AlexanderThe University of Kansas,
Lawrence, KS, palexand@ku.edu.

XXXX-XXXX/2020/10-ART $15.00
https://doi.org/0000

systems. In this sense, remote attestation is an enabling capability that allows for more nuanced and context-dependent trust decisions.

Trust in the context of remote attestation is defined as: (i) having a strongly bound identity; (ii) being built of known, good parts; and (iii) being directly or indirectly observed via a trusted third party operating as expected. Making a trust decision involves a *relying party* requesting an attestation from a *target* (or *attester*) that performs an *attestation* of its state. The target returns collected trust evidence reflecting its configuration and behavior. That evidence is then evaluated by an *appraiser* allowing the relying party to make a trust decision based on results.[1] While mechanisms are important, we explore defining abstract protocols for orchestrating attestations.

Our work rests on two premises. First, remote attestation is potentially applicable whenever a trust decision about a peer system must be made. The sheer diversity of settings and contexts where trust decisions are relevant precludes the possibility of a small set of solutions to satisfy the needs of every situation. A collection of diverse mechanisms for gathering and appraising evidence is essential. Similarly, flexible mechanisms must be provided for coordinating both attestation and appraisal activities.

Second, evidence of trustworthiness may contain sensitive information that could expose the attesting system to attack, perhaps by revealing latent vulnerabilities. The attesting party must have a say in what is reported, how it is reported, and to whom it is reported during an attestation. The attesting party must be protected by local policy that describes how it will respond while protecting sensitive information.

2 TRUST AND EVIDENCE

What does it mean for a router to be trustworthy? A printer? A laptop? A mobile phone? An autonomous vehicle? There is no uniform answer to these questions. However, one thing should be clear: the answer will not be the same for each system. A printer and router perform different functions of differing criticality. Assessing the trustworthiness of a device must account for the function it should perform.

Equally important is the decision a trust assessment is supporting. For example, a user must decide whether to use a given printer to print a specific document. Depending on the contents of the document, it may be more or less important to believe the printer does not have hidden code to copy and send the contents elsewhere. The burden of trust is much lower when printing public information than for printing proprietary information. Similarly, for printing a given document, the burden of trust may differ depending on whether the printer is a public, shared resource, or whether it is a private resource sitting behind a firewall. Considerations only multiply as focus shifts from one device to another, from one trust decision to another, and from one context to another.

For any trust decision, a determination of what information a relying party will need to proceed must be made. Such information minimally depends on the kind of device it is and what decision the trust assessment is supporting. Some situations may require only simple evidence to support a trust decision, as an example a printer having a cryptographic key certified by the organization so it can be trusted for low-risk interactions. Other situations will require more extensive evidence. For example, hashes of key portions of code must match known provisioned values that have been previously statically analyzed for software assurance. Still others will find that evidence generated from boot-time measurements of a component set is less useful than evidence that speaks to their runtime states.

[1] Originally the roles of appraiser and relying party were conflated in a single principal that both appraises and makes a trust decision. Separating roles originates with the IETF RATS attestation standards study group.

In addition to base evidence attesting to a system, a relying party or appraiser may wish to receive *meta-evidence* attesting to the validity of the base evidence. Such meta-evidence may be as simple as a hardware signature over the base evidence, or something more complex such as an attestation of the evidence-generating code itself. Layered evidence might be provided that reflects a system's layered architecture.

In the end, a relying party must make its trust decision using the body of evidence provided by the attesting device and assessed by an appraiser. Such evidence ranges from a simple hash to complex, layered evidence with cryptographic meta-evidence. Ideally, evidence will be sufficient for a relying party to determine whether the device is in a known, expected state and acceptable for use. However, attestation can only observe a system and provide evidence. It cannot alter underlying mechanisms that make a component trustworthy or untrustworthy.

3 ATTESTATION USE CASES

There are a wide variety of use cases where the addition of attestation mechanisms might reasonably provide benefit. Any time a trust decision is required to support a use case, evidence provided through attestation can be expected to increase its quality. A generalized list of some possible use cases demonstrating the potential of attestation mechanisms follows.

- *Authentication*—Authentication is a means to support trust decisions with identity information. Attestation can offer evidence that the authentication mechanisms and the underlying system are indeed those expected without tampering, increasing confidence that correct identities are used in the trust decisions.
- *Release of sensitive information*—Decisions to release sensitive information to remote entities introduce risk. A poor decision regarding such things as cryptographic keys, private credentials, financial information, among others can cause significant damage. Attestation enables the decision process to incorporate information about the receiving system.
- *Network Access Control*—Any authorization decision about access to a remote resource will benefit by augmenting the presented credentials with integrity information about the system from which the request is made. This use case is applicable to all manner of remote resources from simple web requests to assessing the worthiness of a system to join a closed network.
- *Assessing Network Resources*—Just as a service provider can benefit from attestation about a requesting system, any request for service can be enhanced with a trust decision asking about the worthiness of the provider to service the request. This enables such things as detecting compromised servers or other networked devices.
- *Remote Host Monitoring*—Although not a traditional use case where attestation evidence directly supports a trust decision, remote host monitoring does exhibit key aspects to warrant its inclusion as a use case. Monitoring systems gather information and present it to remote servers where decisions are based on analysis of the data. Robust attestation mechanisms can be used to collect integrity evidence at a host and provide the analysis processes of the monitoring system key information in support of better assessments and responses.
- *System Assessments*—Every time a system is used there is an implicit trust decision that it is worthy for the purpose it is being used for. System audits and virus scanning are techniques used to gain this trust. Attestation mechanisms can be used to further establish trust in a system by providing evidence that it has not been compromised and is as expected. This is particularly true where attestation evidence contains runtime integrity data.

4 WORKFLOW VARIATIONS

When attestation mechanisms are integrated into a system in support of a use case, there are many factors that need to be considered. Among them are where attestation requests originate, where and how evidence is to be collected, how evidence is to be packaged and presented to appraisal processes, and how results from attestations will figure into the results of trust decisions. Collectively, these choices dictate an attestation workflow that describes how attestations will work. When visualizing workflows, it is useful to consider that each implies a distinct "shape" of the attestation. As there are a wide variety of possible choices, there can be a wide variety of workflows each taking on a distinctive shape.

In practice, many attestations fit two general workflows. The first is a *certificate-style* workflow where upon receiving an attestation request, an attester sends evidence to an appraiser that produces an unforgeable appraisal certificate. The certificate is sent to a relying party where an attestation policy governs how the trust decision is impacted (Figure 3). The second is a *background check* workflow where in response to an attestation request, an attester sends collected measurements to a relying party who interacts with an appraisal process directly for necessary, appraisal results. These results are used to consult the attestation policy that will impact the trust decision (Figure 2).[2]

Despite the utility of these two models, it would be a mistake to assume they are the only models that should be supported. Many other useful workflows are possible. For example, the use case might require that two parties attest to each other changing the shape of the workflow in either model. Further still, how that mutual attestation might proceed can result in variations of the workflow shape as one side or the other might be required to attest first or perhaps they engage in concurrent attestations.

Workflow shape can be impacted by variations in evidence type, how it is gathered and any previous relationships between the parties in the interaction. Variations of a simple Authentication example illustrate how simple changes result in multiple shapes. Throughout the example, the term user is used for interactions that take place on the user's behalf. In an implementation, these operations would all likely be handled automatically by the attestation service running on the user's local machine.

The simplest implementation of the Authentication use case is shown in Figure 1. The system first prompts the user for authentication credentials. Then the user then requests and appraises evidence that demonstrates the system's integrity.

Depending on the relationship between user and system, the user might desire a complex measurement that is more detailed than in the previous example. In this case, the user may be incapable of appraising the result and may wish to send the evidence to a trusted third party for appraisal. This interaction could take the form of the sequence diagram in Figure 2.

If the system is accustomed to handling this type of request, it could opt to keep a fresh copy of its appraisal result signed by a trusted third party, ready to offer the result to any users that request it (Figure 3). This final use case assumes the user and the system both trust the third party performing appraisal. The cached result must remain fresh enough for the user's needs as determined by user local policy.

As the Authentication example shows, variations in the use case scenario impact the workflow shape. It is also possible to see variations of the requested evidence itself impact workflow shape. Consider a use case where evidence about an attesting system must be augmented with evidence about a hypervisor the attesting entity is running on. This is perfectly reasonable to ensure a relying party will have as much relevant evidence as possible in making a trust decision. As the collection

[2]The certificate-style and background check style trace their origin to discussions with the IETF RATS attestation standards subcommittee. They use the term *passport* in lieu of certificate.

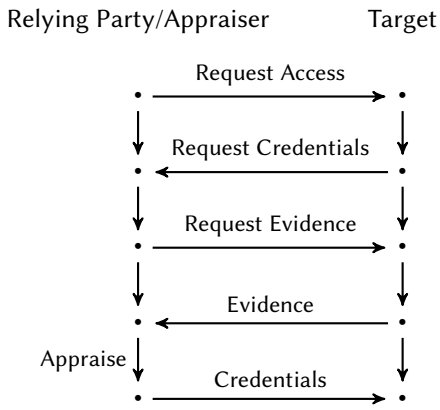

Fig. 1. Attestation process showing interaction among relying party and target.

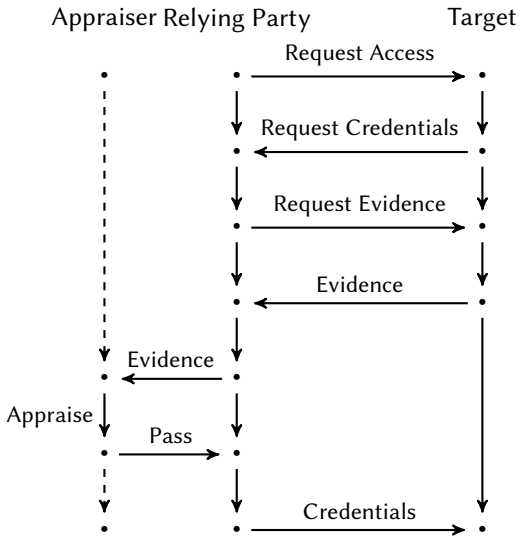

Fig. 2. Attestation process using a third party appraiser.

of hypervisor and guest system evidence are likely separate collection processes, a simple change in evidence for an otherwise identical use case results in a different shape as well as a potentially different result for a trust decision. (Figure 9).

Interestingly, all three examples of Authentication produce the same result type and engage in the same basic interaction. An access triggers a credential request that in turn triggers appraisal and a decision concerning release of credentials. Differences are embodied in how evidence is gathered and appraised. It is the same Authentication surrounded by different mechanisms supporting appraisal.

Understanding how attestation workflows imply shape enables trust analysis of variations that informs decisions about what shapes are suitable for given use cases. It is also a useful construct to help ensure that all parties involved in an attestation are cooperating to produce a proper result.

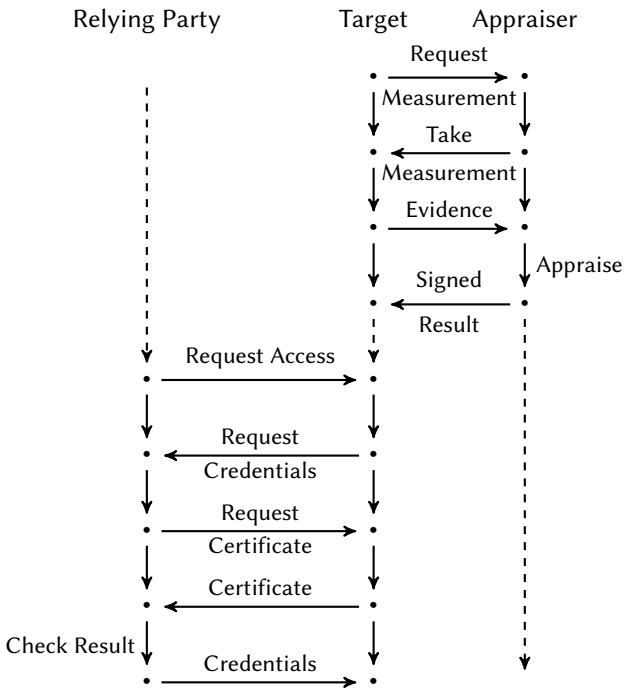

Fig. 3. Attestation process using a cached appraisal result.

Failures in these analyses might very well compromise the quality of the trust decisions rather than enhance them.

5 ORDER MATTERS

The previous section showed how slight variations in the workflow, including collected evidence types, can impact the attestation process and resulting trust decisions. The order measurements are taken and how they are composed to form evidence can be as important as the measurements themselves with regard to inferences about trust by a relying party. As ordering relations and composition can be reflected in workflow shape, this too can be important to inferences about trust.

Consider a simple example in which an integrity measurement of a target userspace process (TP) is taken at runtime by a local userspace measurer (UM) [Pendergrass et al. 2019]. For example, UM might be host-based antivirus software that periodically scans for malicious changes, or a tool that probes the host's browser for suspicious extensions. The most important attribute of UM is its location in userspace alongside TP, its target. In particular, there is no difference in the ambient protection level supplied by the underlying system to UM and TP.

The result of UM's measurement may be used to form evidence for an appraiser's consumption. Depending on its requirements, the appraiser might request, alongside the evidence about TP, additional evidence attesting to the integrity of UM's own runtime state. For example, this evidence might be based on a measurement taken by a *userspace integrity monitor* (UIM) embedded in the host's running kernel. By virtue of its location in the kernel, UIM is afforded a greater degree of protection from malicious runtime tampering than either of UM or TP. Figure 4 depicts UIM, UM and TP as components within the attesting host, along with their userspace or kernel space

locations and measurement relationships. The appraiser is assumed to be external to the host in this example.

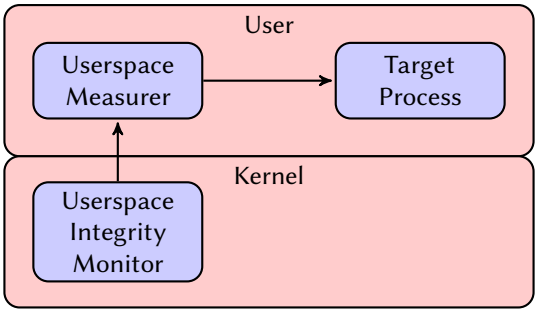

Fig. 4. Compound attestation structure showing internal processes involved in measurement.

In order to construct the compound evidence over UIM and UM's measurements, it is necessary to decide in what order these measurements should be taken. There are three possibilities:

(1) UM measures TP before UIM measures UM
(2) UIM measures UM before UM measures TP
(3) The measurements are taken simultaneously

Suppose the measurements are taken in the presence of a powerful, active adversary who will attempt to corrupt and repair components of the attesting host at runtime. In this context, *corrupt* refers to any modification of a component's runtime state that subverts its intended function, and *repair* refers to the return of a corrupt component to its *regular* state, in which the component carries out its intended functions faithfully. Assume that all components of the attesting host are initialized in regular states at system boot. Relative to intended measurement functions, the regular versus corrupt notion has two important implications:

- A regular measurer always accurately reports its measurement of the target's runtime state
- A corrupt measurer always reports a measurement indicating that its target is regular, even if it is corrupt

Suppose at some time before the attestation begins, the adversary has corrupted TP. In particular, TP is corrupt when UM measures it. If the compound evidence received by the appraiser indicates both UM and TP are regular, the adversary has successfully *avoided detection*. Two key questions are of interest in this scenario:

- What other components must the adversary have also corrupted, either before or during the attestation, in order to avoid detection?
- Can the order in which the measurements are taken be used to make avoiding detection more difficult?

There are several points to keep in view when addressing these questions. As noted, UIM runs wholly within the attesting host's kernel and is therefore more difficult to corrupt than UM or TP. Moreover, the act of repairing a component should be regarded as relatively easy to achieve even in narrow time windows, such as during an attestation. In particular, repairing a component is easier than corrupting one in a narrow time window.

Because UM reported that TP was regular when it was in fact corrupt, UM must also have been corrupt when it performed its measurement. This corruption of UM may have occurred before or during the attestation, depending on the order of the two measurements. Because UIM reported

that UM was regular, one can further infer that the adversary either additionally corrupted UIM or repaired UM after it measured TP but before it was measured by UIM [Rowe 2016b].

If UM measures TP before UIM measures UM, the adversary can avoid detection by corrupting UM at any time before the attestation begins and repairing it after it has measured TP but before it is measured by UIM. Under this ordering, the adversary has as much time as they need to corrupt a relatively less-protected component. During the attestation, the adversary is only required to repair UM, an easier objective to achieve in a narrow time window than a corruption. This ordering thus puts the adversary at an advantage.

By contrast, the adversary is much more constrained when UIM measures UM before it measures TP in turn. In order to avoid detection, the adversary must either 1.) corrupt both UIM and UM before the attestation begins or 2.) corrupt UM during the attestation, after it has been measured by UIM but before it measures the corrupt TP. Here, the adversary is required to choose between a relatively difficult corruption of UM *and* UIM, which enjoys kernel-level protections, before the attestation begins, or a corruption of the less-protected UM in the narrow time window between measurements. Either way, the adversary is at a disadvantage compared to the previous ordering.

It is reasonable to suppose this adversary we have considered is able to exploit slight natural delays between the measurements, or even induce them, to their advantage. The case of simultaneous measurements is therefore also likely to benefit the adversary, who could arrange delays to force the attestation into the UM-before-UIM case where they are less constrained.

In fact, there is a "fourth" ordering relationship for the two measurements, that in which no precise order is specified in the construction of the compound evidence. This approach disadvantages the appraiser and relying party, as it removes any possibility of reasoning about the trustworthiness of evidence based on the context in which measurements were taken.

As this discussion illustrates, the adversary's ability to avoid detection is most constrained when *deeper* components, those in positions of higher protection or greater trust, measure *shallower* ones before these perform their own measurements.

One may wonder how the runtime states of UIM or the attesting host's kernel might also be represented in evidence. In particular, the kernel must provide a clean runtime context for UIM, UM and TP to run in their regular states. By the same token, a corrupt kernel might be able to misrepresent the runtime state of any process, leaving little value in evidence produced on such a host. For more complex layered attestations or ones with more stringent assurance requirements, it is desirable to obtain a runtime measurement of the kernel that may be incorporated into evidence about shallower components the kernel supports. These kernel measurements would be backed by a hardware root-of-trust assumed to be beyond the adversary's ability to corrupt. Prior work has shown how the construction and deployment of a software *kernel integrity monitor* is practical and can promote greater assurance in the fidelity of evidence [Loscocco et al. 2007].

6 EVIDENCE STRUCTURE

Trustworthiness of an attestation depends on more than measurement evidence. The same evidence created in two different orders might provide different opportunities for a malicious actor to go undiscovered by the attestation. Given the importance of the order evidence is generated, the appraiser must have a way of determining the order in compound evidence. Additionally, when different entities were responsible for portions of the evidence, the appraiser must know what entity produced each portion. Compound evidence must contain not only the underlying measurement evidence, but also *meta-evidence* that allows an appraiser to determine evidence gathering order and the party responsible for its gathering.

Consider again the example from the previous section of an attestation that produces two evidence values: one generated by a Userspace Measurer (UM) about the integrity of a target

process (TP) and the other generated by a Userspace Integrity Monitor (UIM) about the integrity of UM. Since it is better to measure UM before it performs its measurement of TP, the compound evidence should reflect this order. There are several ways to structure the compound evidence for this purpose.

An orchestration agent on the attester may be responsible for coordinating the workflow. It could first make a request to UIM, resulting in evidence c_1. Then it could initiate a measurement by UM resulting in evidence c_2. The orchestration agent would then create a compound evidence $c = c_1; c_2$ whose structure indicates that c_1 was generated before c_2. This could be a simple concatenation of the two evidence values, or it could rely on some other syntactic marker like ";" to distinguish it from compound evidence in which the sub-evidence values were generated simultaneously such as $c = c_1|c_2$. In this approach, the appraiser must trust the orchestrating agent to honestly report the order of the workflow it managed, since a corrupted orchestrating agent could simply lie about the order it dictated.

A different approach would be for UIM to hand its evidence c_1 directly to UM who would be responsible for generating the compound evidence. It might be $c = c_1; c_2$ as in the previous approach. Or it might be something more along the lines of $c_2(c_1)$ in which the evidence c_2 incorporates evidence c_1 as an argument. In either case, if c_1 is not integrity protected, the appraiser must trust UM not to tamper with it. If c_1 is integrity protected, the appraiser need not place the same level of trust in UM. Nevertheless, it might still be possible for UM to create evidence c_2 based on an old (or even fabricated) measurement. This would be a way in which the appraiser might be fooled by a corrupt UM about the order of evidence generation.

In the two examples, the structure of compound evidence implies the shape of the workflow employed to construct them. They entail different trust relationships in order for the appraiser to believe that the reported order of evidence generation was actually implemented. It is this evidence structure that enables the relying party to ensure it uses the evidence correctly in support of the trust decision being made.

7 MECHANISM FLEXIBILITY

When integrating attestation mechanisms into a system, it may be possible to settle on a single workflow shape that will be sufficient to support the trust decision for the particular use case being considered. There might be homogeneity in the attesting systems where evidence generation is uniformly producing the same type of evidence. The appraisal process would likewise be uniform and enables a simple input to the attestation policy of the relying party. Unfortunately, it is not likely to be so straightforward.

In practice, multiple workflow shapes will be necessary, mandating flexible mechanisms for attestation. Different attesting systems have different attestation mechanisms. This causes the content and structure of evidence and meta-evidence to vary. Even when attesting systems for all intents and purposes are identical, appraisal may demand varying types and depths of evidence and meta-evidence as attesters may differ in relevant attributes such as whether or not they fall within a particular administrative domain. Such variation can impact the outcome of trust decisions. The appraisal process will need to be able to handle the variety.

Variability will be needed on attesting systems as well because they will likely be required to perform attestations in support of multiple use cases. The different use cases may have different shapes and even when they share a common shape, different types of evidence may be required. As was the case for appraisal, a seemingly identical use case may require different shapes because of attributes of the relying party, as different evidence or structure may be needed.

This argues for flexibility in attestation mechanisms in both attesting and appraisal systems. They must be able to support varied workflow shapes for the use cases where they are expected

to participate. The mechanisms should allow attesters and relying parties to request and perform those measurements necessary for the current situation dictated by the use case, and should prevent duplication and fragmentation of attestations.

Ideally, there should be an attestation service that supports various forms of attestations capable of generating all necessary evidence and meta-evidence. It must be able to support appraisal processes for all types of evidence necessary to the desired use cases and respond to relying parties in a manner appropriate to associated trust decisions. The mechanisms should be extensible so that new workflow shapes can be added when new cases arise.

Flexibility demands that there be a way of deciding on a workflow shape to use for any given situation. Requests for attestations must indicate what shape is desired, making it imperative there be a way to precisely specify what is required, requirements for evidence and structure of any compound evidence. There must be agreement between attester and appraiser that the proper shape and compatible mechanisms have been selected by both.

Another compelling reason to make selection a core component is to allow selective information release. Attestation evidence is sensitive information about a system and should not be released to a remote party indiscriminately. A system's willingness to release sensitive information depends on who the potential recipient is and why it is needed. Similarly, a decision about what to require from an attestation also depends on who is attesting and the reason. The trust decision made relies on both. When selection is independently governed by local policy at each end, only attestations consistent with both policies can proceed.

For any given attestation situation, selection policies at attesting and appraising systems might be coordinated *a priori* to guarantee identical selections. However, as the attestation service is generalized to be extensible and enable a wide variety of selection alternatives, analyzing and managing all local policies to ensure the best selection quickly becomes unmanageable. Instead as part of the selection process, it will be necessary for both sides to negotiate a protocol that satisfies goals of the competing interests of the attester's privacy versus the appraiser's desire for as much relevant evidence as possible for trust inferences.

Flexible attestation mechanisms require a means of precisely specifying workflow shapes, including the types and structure of evidence. Having a declarative language to describe the workflows enables unambiguous intent of a given workflow to be captured and shared. It can be the basis of analysis to ensure that workflows meet the trust requirements of supported use cases. Requests for attestations can include language expressions that clearly describe desired workflows. Selection policies can be expressed in terms of workflow descriptions in a common language with exact meaning. These same expressions facilitate negotiation as workflow preferences and alternatives can be included in proposals and counter-proposals.

Any attestation mechanisms, and especially a flexible and extensible attestation service, is necessarily a privileged entity that must have access to sensitive parts of a system. Special care must be taken to ensure secure integration into the system. Policy configurations must be carefully specified and protected from unauthorized modification. Sound security principles, like least privilege, should be applied to ensure that components, especially those like evidence collectors requiring higher privileges, do not enable system compromise. Failure will lead to the flexible attestation service becoming a vector for system compromise rather than a means of enhancing trust.

8 POLICY-BASED NEGOTIATION

The goal of negotiation should be the successful selection of an attestation workflow that mutually satisfies each participant's requirements as expressed in the local policies. Before an attestation can proceed the negotiation mechanism must establish agreement on the precise sequence of evidence

collection and cryptographic operations that will create the structured evidence that the appraisal mechanisms expect and can correctly interpret in support of the trust decision for the requested situation.

For negotiation to work, selection policies reflecting the particular goals of each party need to be established. It is assumed that attesters will guard the release of evidence just as appraisers will desire maximum disclosure. Selection policies may specify more than one possible selection, possibly indicating a preferred order. The negotiation service strives for a selection that will find the best fit satisfying both sides. Failure to agree on a selection should be treated differently than an appraisal failure as participating parties may wish to take different actions.

There are many ways that negotiation could take place. A simple example could be that a relying party requests an attestation for a specified situation, passing relevant information to both the attesting and appraising systems where each consults a local selection policy. From there a series of proposals and counter-proposals would hopefully result in a mutually satisfactory selection. Upon agreement both sides dispatch the appropriate mechanisms to execute the selected workflow. Although illustrative, this simple approach is too inefficient for actual use.

ISAKMP [Maughan et al. 1998] is a good starting point for the creation of a negotiation service. ISAKMP is an established protocol used to negotiate cryptographic associations that could easily be adapted for attestations. It is a two-phase negotiation protocol that first does a key exchange to establish a security association between the participants and then performs actual negotiation. The security association provides a secure session on which negotiation can proceed and provides necessary information about a peer that is useful input to the selection process. The negotiation phase is a simple proposal consisting of a list of acceptable choices followed by a selection that is assumed to be mutually agreeable. From there, compatible mechanisms are dispatched to execute the selected workflow.

Negotiation using the ISAKMP model works as follows. On request for an attestation, a security association is established and used to communicate the requested attestation situation. It can also convey other relevant input to selection not already available from the security association. Selection policies expressed using a declarative attestation orchestration language as suggested in the previous section are consulted. The initiator creates an ordered list of acceptable alternatives from its policy. Each element of the list is the workflow description expressed in that language. The proposal is sent to the peer who then chooses the best alternative matching expressions selected from its policy. Selection policies should never include unacceptable choices as they could be selected. Similarly, selection policies should never include selections that cannot be fulfilled. A failure of negotiation is preferred over either of these outcomes.

ISAKMP has several useful, applicable optimizations. Both the security associations and the results of previous negotiations are cached creating the possibility that the security association establishment and/or the actual negotiation might be bypassed. These caches are checked at appropriate points during execution with results of cache hits used in subsequent stages. This is particularly useful when attestations are frequent or when results of negotiations can be predicted and preloaded into the caches. They are also useful when attestation workflows include additional parties. Separate negotiations would be required to ensure proper mechanism selection at every party. The ability to establish ongoing relationships with other parties and have the previous results present in the cache certainly benefits performance. Examples of such parties include third-party appraisers or other attestation entities involved with collecting meta-evidence.

9 AN ATTESTATION PROTOCOL LANGUAGE

Previous sections argued for an attestation protocol language and showed that attestation services could benefit from one. This section expands on the need for such a language and describes

properties that it should have. We propose Copland [Ramsdell et al. 2019], a domain specific specification language that was designed for this purpose and with those properties in mind. A protocol writer must provide answers to many questions when defining a protocol for collecting evidence about a complex, layered system:

- Who measures what?
- In what order should measurements be taken?
- Which measurements should be packaged together?
- Which locations should be allowed to transfer sensitive evidence?
- When should evidence be signed and by whom?

Encoding this information in procedural code makes the protocol difficult to understand and get right. Such an encoding is also brittle in the face of changing requirements, especially if the people updating the policy are not the same as the original authors. To address this problem we propose declarative specifications whose interpretation is formally verified.

Attestation workflows have a structure that can be described by a machine-readable language. There are entities that are making and responding to requests. There are relations between entities that describe message exchanges. The content of messages can be abstracted to include what is essential to determine when a workflow achieves a desired goal, describe attestation policies, and define relationships between workflows.

A declarative description of workflows based on a machine-readable language is a desirable way to capture the variety of attestation possibilities. In addition to simply describing workflows the language must have the following properties:

- It should eliminate ambiguity, allowing different parties to agree on what each specification means.
- It should be modular in the evidence being requested.
- It should be capable of expressing the protocol structure of the requests and replies among the various parties.
- It should be able to express both control flow and data flow requirements.
- It should be able to express requirements for how composite evidence is bundled and given integrity/confidentiality protections.
- It should support modular specifications, not just simple names for complex interactions.

Copland is a protocol language capable of expressing complex interaction patterns in a modular style. It is a machine-readable language with an abstract, formal semantics specifying remote execution, ordering, and transformation of protocol to evidence. Most significantly Copland is parameterized over work to perform making it highly flexible and retargettable across many application domains.

Semantically, a Copland expression defines the mapping from input evidence to some output evidence via one or more attestation events. Each event occurs as a result of action taken by a well-defined principal. Events involve taking measurements, signing or hashing evidence, or routing evidence to produce combinations of evidence.

Principals in the specification language are logical components that participate in the specified attestation. Note that labels such as attester, appraiser and relying party are more appropriately identified with roles, not principals. A single principal may act in several roles over the course of an attestation.

Expressions in Copland are called *phrases*. The basic unit of any phrase is a *measurement*. In this context measurements represent any action taken by a principal that generates or consumes evidence. A measurement always produces at least the evidence that states the measurement

occurred. Measurements are represented as arbitrary text strings that are meant to be descriptive of the actions they designate. The value of a phrase is the type of evidence it produces.

Discrete measurements have the form `(M P T)` where `M` names the measurement, `P` is the place where the measurement target resides, and `T` is the measurement target. When the measuring and measured principals are the same or when a given measurement action has no target the principal and target parameters may be omitted.

Discrete measurements are composed using three operators, `->`, `<` and `~`. Given measurements M_1 and M_2:

- The phrase M_1`->`M_2 indicates that M_1 is taken before M_2 and that evidence produced by M_1 is passed directly as input to M_2. The result of M_1 logically appears in the resultant evidence as part of M_2's output evidence.
- The phrase M_1`+<+`M_2 indicates that M_1 is taken before M_2 and the two resultant evidence artifacts are composed sequentially according to some bundling strategy for doing so. The `+` on either side of the `<` operator indicate that both M_1 and M_2 receive any evidence provided as input to this phrase (for instance, when it is a subterm of a larger phrase). By contrast, M_1`-<-`M_2 indicates that neither measurement receives the input evidence. Both `+<-` and `-<+` are also possible.
- The phrase M_1`+~+`M_2 indicates that M_1 and M_2 are taken in parallel (unconstrained precedence) and their results composed to indicate parallel measurement according to some bundling strategy. Both M_1 and M_2 also receive any input evidence. The variants `+~-`, `-~+`, `-~-` are also possible.

The notation `@`$P[M]$ requests that phrase M be executed at *place* P. Copland places identify principals with the ability to execute phrases. Similarly, the notation $*P, n : M$ indicates that P will run M initialized with optional nonce, n. The `*` notation cannot be nested and must be the outermost construct.

Finally, several discrete actions are available for generating meta-evidence or transforming evidence. One is used throughout this paper. The `!` signs its input with the private key associated with the executing place. Thus `@P:m->!` results in `m` signed with `P`'s private key. Additionally, `_`, `#` and `{}` define copy, hash, and null respectively.

The syntax presented above was motivated by our formal analysis work. In the tradition of mathematics, it is concise. But some users find it too concise. We expect the syntax of Copland will evolve and become more user-friendly as we gain experience with the language.

10 ATTESTATION PROTOCOL EXAMPLES

Common attestation workflows or shapes provide motivation for using a language like Copland. We examine several shapes, each one typifying a common attestation pattern, and demonstrate how to express these using Copland. Some examples use Copland to explore variations on phrases with the same shape but different trust properties. The formal semantics of Copland are designed to support formal analyses of workflows expressed as Copland phrases.

Three roles recur throughout the examples. The relying party role identifies the principal depending on the trust assessment. The attester role identifies the principal providing evidence for appraisal. The appraiser role identifies the principal evaluating evidence resulting from an attestation. Often the appraiser and relying party roles are performed by the same principal. Such a principal requests an attestation, appraises the evidence, and makes a trust decision based on the results of appraisal.

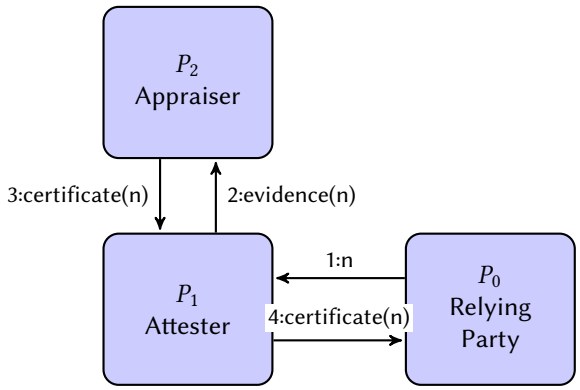

Fig. 5. Certificate-Style.

Certificate-Style

In the certificate-style remote attestation topology, an attester submits evidence to an appraiser and obtains in return a "certificate," an unforgeable, authoritative evidence artifact summarizing the results of appraising the submitted evidence. The attester may then provide the certificate to a relying party to facilitate a trust decision. The attester may obtain a certificate before receiving an attestation request from a relying party or in response to such a request. As the following variations illustrate, the relying party's decision-making should take this difference into account.

Figure 5 depicts a remote attestation scenario where a relying party depends on a logically separate appraiser to evaluate evidence produced by an attester. The relying party and attester both trust the appraiser to appraise evidence fairly and report the results of verification accurately.

The shape in Figure 5 depicts the variation in which the attester obtains the certificate in response to a request from the relying party. The attestation begins when the relying party makes an initial request to the attester that includes a nonce `n`. In response, the attester generates the requested evidence, cryptographically incorporating `n`, and submits the evidence to the appraiser. In Figure 5, this evidence is represented as `evidence(n)`, indicating that it is bound to `n`. The appraiser appraises the evidence and, in the case that it is acceptable, generates a verification result called `certificate(n)` in Figure 5 attesting to this fact for the relying party's consumption. This certificate is also bound to `n`, enabling the relying party to check that it was generated recently. The appraiser sends the certificate to the attester, who completes the attestation by forwarding the certificate to the relying party.

In Copland, this attestation may be expressed as:

```
*P0,n: @P1[(attest P1 sys) ->
         @P2[(appraise P2 sys) ->
               (certificate P2 sys) ]]
```

As in Figure 5, this phrase associates the relying party with principal `P0`, the attester with `P1` and the appraiser with `P2`. A request bound to nonce `n` is generated by `P0`. The result of this, `n`, is sent to `P1`, who passes it as input to its attestation system. The action `(attest P1 sys)` consumes `n` and produces `evidence(n)`, the evidence that is bound to `n`. This evidence is sent to `P2`, where it is consumed by `(appraise P2 sys)`. This action produces an appraisal result, which, if merited, is passed as input to the `(certificate P2 sys)` action that generates the certificate on it. This sequential composition of the `appraise` and `certificate` actions reflects the fact that the certificate is only generated after the attester's evidence has been appraised and found to be

acceptable. The certificate action might be as simple as the appraiser signing the results of appraisal. In this case the Copland phrase has the form:

```
*P0,n: @P1[(attest P1 sys) ->
         @P2[(appraise P2 sys) -> ! ]]
```

The pipeline of the `appraise` and `certificate` actions transforms the attester's evidence into an authoritative verification result. The certificate is bound to `n` because the appraisal results, via their dependence on the attester's evidence, are. The square brackets show that `P1` receives the `(certificate P2 sys)` result from `P2` after it sends `(attest P1 sys)` to `P2` and that `P0` receives `certificate(n)` from `P1` after the rest of the attestation has occurred, exactly as shown in Figure 5.

This example shows how a common remote attestation topology may be compactly encoded in an appropriate specification language like Copland. The relying party may use a phrase like this one to request recent evidence from the attester.

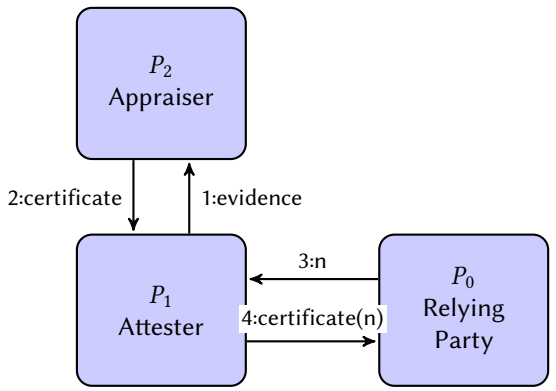

Fig. 6. Cached Certificate-Style.

Figure 6 shows a reworking of this example in which the attester can obtain a certificate before it is requested by a relying party. In this scenario, the attester caches the certificate and replies to requests from relying parties by quoting from this cache. This variant may be expressed as:

```
*P1:(attest P1 sys) ->
     @P2[(appraise P2 sys) -> (certificate P2 sys)] ->
       (store P1 cache)
*P0,n:@P1[((retrieve P1 cache) -<+ _) -> !]
```

Each of `P0`, `P1` and `P2` acts in the same role as before. Here, however, two distinct Copland phrases are used: one for `P1`'s certificate interaction with `P2` and another for the request made by `P0` to `P1` for attestation. These phrases execute independently, permitting the execution shown in Figure 6 where the certificate is obtained before the attestation request. In particular, this means the relying party's nonce is not cryptographically bound to the attester's evidence. As a result, the relying party is only guaranteed recency of the attester's response, not of the certificate (or the results of appraisal it certifies) contained in the response.

Background Check

Figure 7 depicts a background check remote attestation topology. In contrast to the certificate-style topology, in which the attester submits its evidence to the appraiser directly, here the attester submits its evidence to the relying party to forward to the appraiser on its behalf.

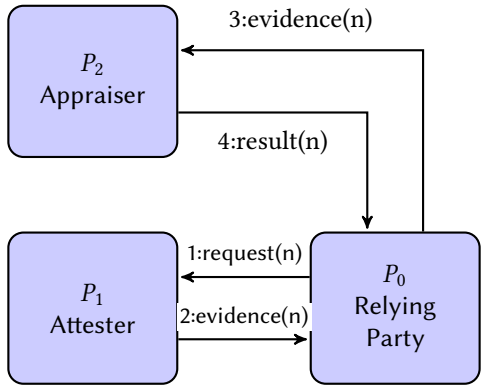

Fig. 7. Background Check.

The principals P0, P1 and P2 in the attestation act in the same roles as before. The relying party again includes a nonce n with its request to obtain a measure of recency. The attester performs some local measurement that generates evidence evidence(n) bound to the relying party's nonce. The attester then sends evidence(n) to the relying party, who forwards it to the appraiser P2. The appraiser verifies the attester's evidence and generates new evidence summarizing the results of appraisal, denoted by result(n) in Figure 7. The appraiser then sends result(n) to the relying party, completing the attestation. That n is bound to evidence(n) and result(n) permits the relying party to check the recency of both.

The Copland phrase for this topology is:

```
*P0,n: @P1[(attest P1 sys)] -> @P2[(appraise P2 sys)]
```

P0 begins the attestation with a request bound to the nonce n. This request feeds directly into P1's attest measurement, introducing the desired dependency on n. The result of attest is returned to P0, who forwards it to P2 to perform the appraise operation and generate result(n). Finally, the result of the appraise action is returned to P0, completing the interaction.

This example demonstrates an instance of a simple topology translating straightforwardly into a simple, recognizable Copland phrase.

Parallel Mutual Attestation

Figure 8 depicts a remote attestation topology where two principals, P_0 and P_1, act as relying party and attester at different points in a mutual attestation. P_0 and P_1 rely on a shared appraiser, shown as principal P_2. Each of P_0 and P_1 wish to request attestation from the other under a background check topology mediated by the shared appraiser. This topology may be expressed in Copland in two phrases, each representing one principal's attestation to the other. Literally, these are two background check protocols run in parallel with a shared appraiser:

```
*P0,n₀: @P1[(attest₀₁ P1 sys)] ->
          @P2[(appraise₀₁ P2 sys)]
*P1,n₁: @P0[(attest₁₀ P0 sys)] ->
          @P2[(appraise₁₀ P2 sys)]
```

The first phrase shows P_0 acting in the role of relying party and P_1 in the role of attester, and the second phrase has these assignments reversed. In the phrases and Figure 8, $\texttt{attest}_{01}$ is the measurement performed by P_1 when P_0 requests an attestation from it bound to nonce n_0, and $\text{evidence}_{01}(n_0)$ is the evidence produced by that measurement (analogously for $\texttt{evidence}_{10}$).

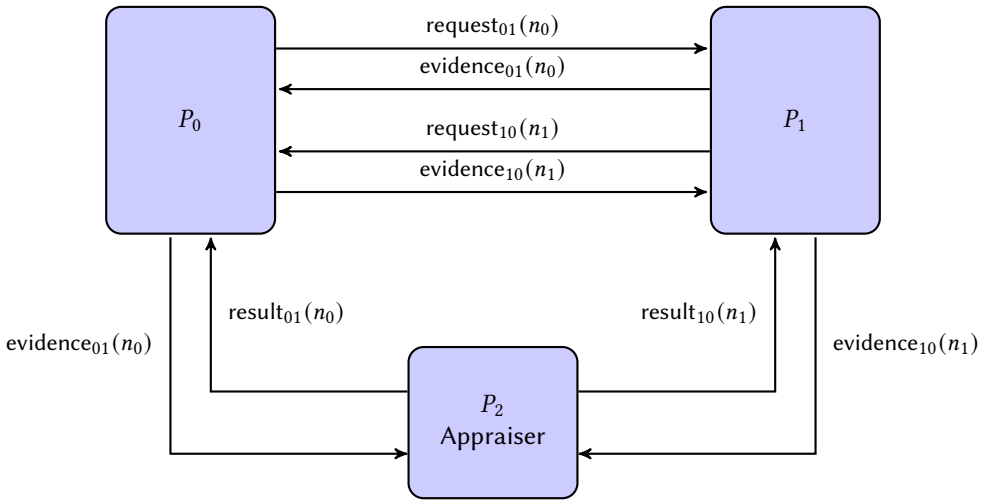

Fig. 8. Parallel mutual attestation.

Likewise, $\texttt{appraise}_{01}$ is the appraisal action P_2 performs on $\text{evidence}_{01}(n_0)$, and $\text{result}_{01}(n_0)$ is the evidence generated by P_2 based on appraisal results. (analogously for $\texttt{appraise}_{10}$ and $\text{result}_{10}(n_1)$). The two halves of this mutual attestation execute independently in parallel, that is, in no particular order.

This example shows how Copland phrases usually reflect the symmetry and complexity of the topologies they describe. Some care is needed during negotiation to ensure precedence relationships are specified correctly, especially where bottom-up guarantees are desired. The example also demonstrates the distinction between roles and the principals who execute them. Over the course of an attestation, a single principal may act in multiple roles.

Layered Background Check

Figure 9 shows a more complicated background check-style topology implementing a *layered attestation*. The relying party P_0 issues a request for attestation bound to a nonce `n` to an attester P_1 and sends collected evidence bound to `n` to a trusted appraiser P_2 for appraisal. The appraiser sends the results of appraisal to the relying party for use in a trust decision.

In this example the attestation is *layered* rather than performed exclusively by P_1. Here, P_1 gathers and composes evidence from two different targets by sending attestation requests to P_3 and P_4 and composing results. These layered attesters, P_3 and P_4, behave as any attesters by responding to the request with attestation evidence. P_1 bundles that evidence along with evidence it may have gathered and returns the bundle to the relying party.

The relying party P_0 need only know it needs evidence from P_1 and nothing about P_3 or P_4. P_1's local privacy policy specifies what it is willing to reveal, and P_0's local policy determines what protocol best suits its needs. Only the appraiser P_2 need understand the attestation result. P_1 may go so far as to encrypt evidence to prevent access by P_0.

This example topology in Figure 9 may be expressed in Copland as:

```
*P0,n: @P1[(attest P1 sys) ->
             (attest P3 att) ->
                (attest p4 att)
          +~+
```

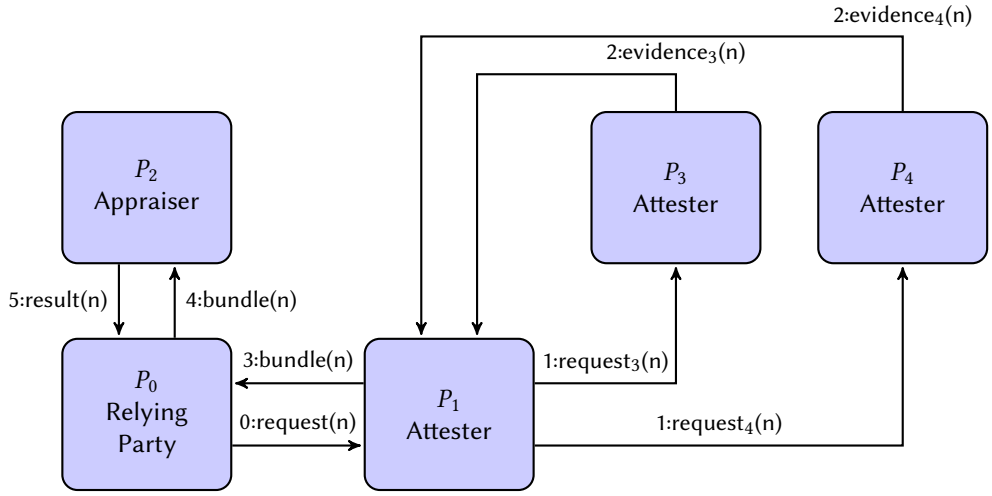

Fig. 9. Layered background check.

```
(@P3[(attest P3 sys)]
   +~+
 @P4[(attest P4 sys)]) ->
@P2[(appraise P2) -> !]]
```

Attestation begins with a nonce-bound request from relying party P_0 to attester P_1. P_1 performs `attest` on itself followed by off-platform attestations of P_3 and P_4's attestation subsystems. In parallel, P_1 requests P_3 and P_4 perform `attest` on themselves. The intention of this topology is to gather evidence from P_3 and P_4 to establish each can be trusted to attest to its own state. P_1 receives evidence and bundles it with its own evidence. This bundling of P_1, P_3 and P_4's outputs is represented in Figure 9 as `bundle(n)` and has the form:

`bundle(n) = evidence(n) | (evidence`$_3$`(n) | evidence`$_4$`(n))`

Parallel composition in the Copland phrase allows attestation on P_1, P_3 and P_4 to occur in any order. This is indicated in the bundled evidence by the `|` composition operator. When appraising `bundle(n)`, the parallel bundling operator indicates that no ordering may be assumed on the gathering of evidence. This in turn limits what appraisal can reveal about how attestation constrains an adversary.

Parallel composition of P_1's evidence with P_3 and P_4's implies P_1's attestation of P_3 and P_4 might have occurred after P_3 and P_4 performed their attestations. Thus, P_1's evidence says nothing about P_3 and P_4 at the time they perform their attestations. This is critical, as an adversary may have corrupted and repaired P_3 or P_4 before P_1's attestation. While running in parallel should improve runtime performance, it does so at the cost of assurance.

A minor modification of the phrase forces P_1's attestation to occur first:

```
*P0,n: @P1[(attest P1 sys) ->
             (attest P3 att) ->
                (attest p4 att)
           +<+
          (@P3[(attest P3 sys)]
             +~+
```

```
 @P4[(attest P4 sys)]) ->
@P2[(appraise P2) -> ![]]
```

The change from parallel to sequential composition means that P_1 must complete its work before P_3 and P_4 can start. The new evidence bundling has the form:

$\texttt{bundle(n) = evidence(n) ; (evidence}_3\texttt{(n) | evidence}_4\texttt{(n))}$

allowing the appraiser to conclude that `evidence(n)` was gathered first and describes the state of P_3 and P_4 prior to their local attestations. While an adversary may run an attack between attestations, this protocol makes the adversary's task more difficult. Ensuring P_1 completes its work immediately before P_3 and P_4 start could eliminate even this contingency [Rowe 2016b].

The layered background check example illustrates many foundational attestation principles.

- Precise semantics for phrases supports informed negotiation among appraisers and targets. The Copland representations are more detailed and precise than the diagrams, allowing the relying party and attester to interpret and negotiate protocols. The first protocol takes less time to complete, but places fewer constraints on an adversary.
- Order matters when appraising evidence and assessing how attestation constrains an adversary. A simple ordering change significantly changes constraints that appraisal places on an adversary. The first protocol allows a longer window for attack and guarantees less about the attestation infrastructure in P_3 and P_4.
- Flexible mechanisms allow both parallel and sequential ordering without modifying the attestation infrastructure. Simple changes in protocol structure allow different attestation shapes. Reasoning about those shapes is performed at a higher abstraction level.
- Evidence structure and meta-evidence allow the appraiser to make relatively stronger conclusions about the appraisal target. Concluding from evidence structure that attestation of P_3 and P_4 occurs prior to their local attestation allows the appraiser to make stronger conclusions about the overall system.

11 RELATED WORK

Our work follows principles set forth by Coker et al. [2011] in their description of remote attestation. They define an architecture centered on an attestation manager integrating measurements and executing attestation using an attestation protocol. Throughout their work they emphasize the need for flexible attestation mechanisms, a major goal of this work.

Prior theoretical work by Rowe [2016a,b] explores properties of layered attestation as presented specifically in section 10 and more generally across all presented examples. Layered attestation describes how collections of attestation assets are together used to establish trust in a system and is essential for addressing any multi-component system or systems that depend on others. Rowe describes how adversaries are constrained differently by different layering structures and how evidence must be composed to accurately reflect attestations.

The Linux Kernel Integrity Measurer (LKIM) [Loscocco et al. 2007] is a retargetable contextual measurer for kernel modules. LKIM exemplifies the kind of flexible measurement capability required by this approach. However flexible they might be, LKIM and similar tools target operating system measurement specifically and cannot easily be generalized to implement attestation architectures discussed here.

Maat [Pendergrass et al. 2018] and [Petz and Alexander 2019] are prototype attestation frameworks implementing the Copland approach. Maat [Pendergrass et al. 2018] is an open source prototype platform service built to explore and demonstrate the properties necessary for a centralized measurement and attestation service. The use cases presented here comprise a non-exhaustive list of those that have been explored during the development of the Maat prototype, and serve as

further illustration of the flexibility required for a centralized attestation service. Maat attestation managers rely on selection policy and negotiation in order to select a Copland phrase and its accompanying measurement agent to best suit the needs of the current attestation scenario.

Petz and Alexander [2019] implement a Copland attestation manager in Haskell used extensively for prototyping basic Copland capabilities. They define a JSON exchange format for Copland protocols and evidence and implement communication among multiple attestation managers. Mechanisms for dispatching work to measurement capabilities allow customization for specific applications.

Other attestation frameworks related to this work include Trusted Network Connect (TNC) [TCG 2012], SAMSON [Fisher et al. 2012], and OpenAttestation [IBM 2015]. Each framework provides mechanisms for executing measurements, a protocol for creating and exchanging evidence bundles, and mechanisms for appraising the results of protocol execution. Most provide a mechanism for plugging in customized measurement and appraisal capabilities. The lone exception being OpenAttestation where measurements are hard-coded. However, all are less general than the approach presented here. None have the concept of place or that one place may measure another's resources. None support changing roles or the definition of attestation shapes. Their protocol languages are *ad hoc* and tightly focused on appraisal of a single environment.

SAMSON (secure authentication modules) [Fisher et al. 2012] appraises client machines in a distributed computing environment. The appraiser is the SAMSON application while systems being appraised are clients. There is no mechanism for changing roles or defining different attestation architectures.

OpenAttestation [IBM 2015] is an extension to the popular OpenStack [Sefraoui et al. 2012] virtualization environment that provides an attestation capability. OpenAttestation uses an approach similar to Haldar et al. [2004] where the virtualization infrastructure directly supports the attestation process. However, OpenAttestation hard-codes its measurement and appraisal capabilities, making negotiation about protocol and implementation difficult.

Trusted Network Connect (TNC) [TCG 2012] is an attestation framework defined by the Trusted Computing Group (TCG) as a part of their suite of trusted computing capabilities. TNC centers on the implementation of a plug-and-play integrity measurement system. Integrity Measurement Collectors (IMCs) and Integrity Measurement Verifiers (IMV) define measurement and appraisal capabilities. The TNC client performs attestation by calling a predefined set of IMCs to gather attestation evidence. The TNC server executes a corresponding set of IMVs to appraise resulting evidence. Like Copland, IMC and IMV instances can communicate via a specialized protocol, TNCCS. In this way the client plays the role of attester while the server plays the role of appraiser. TNC provides some of the same flexibility of Copland and certainly more than OpenAttestation or SAMSON. However, roles are predefined, and defining attestation shapes is done in an *ad hoc* fashion.

12 CONCLUSION

Establishing trust among systems is critically important in today's distributed computing environments. From simple authentication to mutual appraisal among complex layered systems, trust is essential. We present a flexible mechanism for remote attestation that centers on coordinating attestation and appraisal tasks among various system actors including attesters, appraisers, and relying parties. Based on the Copland attestation protocol language, this mechanism supports representing, reasoning about, and executing attestation protocol shapes for a variety of attestation tasks.

To support our claim, we represented several common attestation shapes showing how Copland precisely represents remote attestation tasks. We modeled certificate-based and background

check-style attestations, then used those shapes as a basis for caching attestation results, layering attestation and bundling evidence, and mutual attestation among parties. We discussed how shapes are parameterized over work and how negotiation instantiates a shape with specific implementations for its tasks.

Work on attestation systems using Copland is accelerating. Systems such as Maat and the Haskell Attestation Manager have been constructed and empirically evaluated. New implementations are emerging that use Copland's formal semantics to construct verified systems on the seL4 [Klein et al. 2010] microkernel using CakeML [Kumar et al. 2014]. Each of these implementations provides building blocks for constructing remote attestation systems ranging from simple peer-to-peer attestation to complex, layered systems where precision and flexibility are critical.

REFERENCES

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 (June 2011), 63–81.

George S. Coker, Joshua D. Guttman, Peter A. Loscocco, Justin Sheehy, and Brian T. Sniffen. 2008. Attestation: Evidence and Trust. In *Proceedings of the International Conference on Information and Communications Security*, Vol. LNCS 5308.

C. Fisher, D. Bukovick, R. Bourquin, and R. Dobry. 2012. *SAMSON–Secure Authentication ModuleS.* General Dynamics C4S, https://sourceforge.net/p/secureauthentic/wiki/Home/.

Vivek Haldar, Deepak Chandra, and Michael Franz. 2004. Semantic Remote Attestation – A Virtual Machine directed approach to Trusted Computing. In *Proceedings of the Third Virtual Machine Research and Technology Symposium*. San Jose, CA.

IBM. 2015. *OpenAttestation (OAT) Project.* OpenStack Foundation, https://wiki.openstack.org/wiki/OpenAttestation.

Gerwin Klein, June Andronick, Kevin Elphinstone, Gernot Heiser, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. 2010. seL4: formal verification of an operating-system kernel. *Commununications of the ACM* 53, 6 (2010), 107–115. https://doi.org/10.1145/1743546.1743574

Ramana Kumar, Magnus O. Myreen, Michael Norrish, and Scott Owens. 2014. CakeML: A Verified Implementation of ML. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (San Diego, California, USA) *(POPL '14)*. ACM, New York, NY, USA, 179–191. https://doi.org/10.1145/2535838.2535841

Peter A. Loscocco, Perry W. Wilson, J. Aaron Pendergrass, and C. Durward McDonell. 2007. Linux kernel integrity measurement using contextual inspection. In *Proceedings of the 2007 ACM workshop on Scalable trusted computing* (Alexandria, Virginia, USA) *(STC '07)*. ACM, New York, NY, USA, 21–29. https://doi.org/10.1145/1314354.1314362

D. Maughan, M. Schertler, Schneider M., and J. Turner. 1998. *Internet Security Association and Key Management Protocol RFC 2048 (ISAKMP).* Technical Report. The Internet Engineering Task Force of The Internet Society.

J Aaron Pendergrass, Sarah Helble, John Clemens, and Peter Loscocco. 2018. A Platform Service for Remote Integrity Measurement and Attestation. (2018).

J. A. Pendergrass, N. Hull, J. Clemens, S. C. Helble, M. Thober, K. McGill, M. Gregory, and P. Loscocco. 2019. Runtime Detection of Userspace Implants. In *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*. 1–6.

A. Petz and P. Alexander. 2019. A Copland Attestation Manager. In *Hot Topics in Science of Security (HoTSoS'19)*. Nashville, TN.

J. Ramsdell, P. D. Rowe, P. Alexander, S. Helble, P. Loscocco, J. A. Pendergrass, and A. Petz. 2019. Orchestrating Layered Attestations. In *Principles of Security and Trust (POST'19)*. Prague, Czech Republic.

Paul D Rowe. 2016a. Bundling Evidence for Layered Attestation. In *Trust and Trustworthy Computing*. Springer International Publishing, Cham, 119–139.

P D Rowe. 2016b. Confining adversary actions via measurement. *Third International Workshop on Graphical Models for Security* (2016), 150–166.

Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. 2012. OpenStack: toward an open-source solution for cloud computing. *International Journal of Computer Applications* 55, 3 (2012), 38–42.

TCG. 2012. *TNC Architecture for Interoperability version 1.5* (1.5 ed.). Trusted Computing Group.