A Framework for Policy Based Negotiation*

Anna Fritz and Perry Alexander[0000−0002−5387−9157]

Institute for Information Sciences
The University of Kansas
{arfritzz,palexand}@ku.edu

Abstract. Semantic remote attestation is a process for gathering and appraising evidence to determine the state of a remote system. Remote attestation occurs at the request of an appraiser or relying party and proceeds with a target system executing an attestation protocol that invokes attestation services in a specified order to generate and bundle evidence. The appraiser may then reason about the evidence to establish trust in the target's state. *Attestation Protocol Negotiation* is the process of establishing a mutually agreed upon protocol that satisfies the appraiser's desire for comprehensive information and the target's desire for constrained disclosure. Here we explore formalization of negotiation focusing on a definitions of system specifications through manifests, protocol sufficiency and soundness, policy representation, and negotiation structure. By using our formal models to represent and verify negotiation's properties we can statically determine that a provably sound, sufficient, and executable protocol is produced.

1 Introduction

Establishing trust in a networked communicating peer is a difficult problem. Martin et al. (2008) state trust may be displayed through unambiguous identification, unhindered operation, and direct observation of good behavior or indirect observation by a trusted third party. One possible technique that allows a communicating peer to establish trust in a target system's execution is semantic remote attestation (Haldar et al., 2004). Shown in Figure 1 a *relying party* (RP) or *appraiser* (A) sends an attestation request ($r : (R, n, a)$) to a *target* (T) where attestation generates and returns evidence and meta-evidence ($e : (E, n)$) that can be appraised to determine trust.

Coker et al. (2011, 2008) define a remote attestation model where a target executes an *attestation protocol* that gathers evidence and generates meta-evidence. The protocol sequences the execution of attestation services that perform measurement, generate cryptographic signatures, and make requests of other sys-

* This work is funded in part by the NSA Science of Security initiative contract #H98230-18-D-0009 and Defense Advanced Research Project Agency contract #HR0011-18-9-0001 and Honeywell FMT Purchase Order #N000422909. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

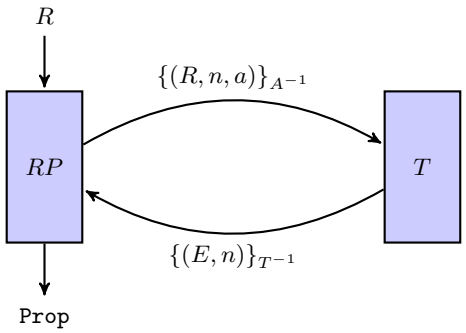

Fig. 1: Remote attestation architecture showing a *relying party* making an attestation request of a *target*.

tems. These protocols are executed by one or many *attestation manager(s)* associated with the relying party and target.

The design of attestation systems are guided by the following five principles taken from Coker et al. (2011). Summarized here these properties ensure evidence used by a relying party represents the system being appraised temporally and functionally:

1. Fresh information – Evidence gathered should reflect the system at the time it was gathered. This extends from boot-time evidence to run-time.
2. Comprehensive information – Evidence should provide a comprehensive view and attestation should access sufficient information about the target state.
3. Constrained disclosure – The target should control what is revealed to the relying party based on the relying party's identity and security context.
4. Semantic explicitness – Evidence should have a well-defined uniform and logical semantics.
5. Trustworthy mechanism – Evidence of attestation infrastructure trustworthiness must be provided to the relying party. Such evidence is called *meta-evidence*.

Most principles are upheld through the existing attestation infrastructure. However, the principles of comprehensive information and constrained disclosure require additional mechanisms to support their satisfaction. To meet the goal of comprehensive information, the relying party must have some idea of the target's measurement capabilities allowing them to select an comprehensive measurement. To meet the goal of constrained disclosure, the target must have some mechanism to distinguish measurements that would expose sensitive information. These contradictory goals can be difficult to mutually satisfy as the relying party would like the most descriptive evidence while the target would like to protect its information thus performing minimal measurements.

To solve this problem, we introduce *negotiation*: a networked communication scheme where a target and relying party may mutually agree upon a protocol that

satisfies conflicting objectives of comprehensive information and constrained disclosure. We define a negotiation scheme with the objective of selecting a sound, sufficient and executable attestation protocol for any situation. Using Coq we formally define protocol executability, soundness, and sufficiency as correctness criteria. By establishing these properties are decidable, we show they can be checked during negotiation and produce decision procedures for that purpose.

Verified decision procedures for soundness and sufficiency criteria are critical to negotiation. Running a protocol that is unsound—not executable or in violation of local policy—will at best fail and waste resources or at worst result in erroneous attestation results. Running a protocol that is insufficient will not satisfy an appraisers need for comprehensive information. Our results provide tools and models that prevent an ineffective negotiation procedure. All formal models are realized using the Coq (Bertot and Castéran, 2013) environment and are available publicly at `git@github.com:ku-sldg/nfm2023.git`

2 Background

Negotiation builds upon security associations (Maughan et al., 1998), attestation protocols Ramsdell et al. (2019); Coker et al. (2011), and attestation manager manifests. ISAKMP is a protocol for finding a security association among relying party and target that instantiates a common vocabulary and establishes a secure communication. Copland is a formally specified, domain-specific language for representing and reasoning about attestation protocols. Manifests are formally specified, abstract descriptions of attestation managers that define protocol execution capabilities and communication paths. Qualities of an attestation protocol can be determined by reasoning about it in the context of both a manifest and identities of the communicating peers.

2.1 ISAKMP

Negotiation begins with the establishment of a security association through the Internet Security Association and Key Management Protocol (ISAKMP) (Maughan et al., 1998). A security association (SA) is an agreement between communicating peers protecting all subsequent traffic. Within the security association, peers state their identities, define cryptographic primitives, identify the situation, and instantiate a domain of interpretation.

Identities and cryptographic primitives are trivial security services. More interestingly, the situation and domain of interpretation are unique, specialized fields critical to the success of negotiation. Within the situation, the relying party and target realize context specific information to instantiate policy. Additionally, the domain of interpretation solves any naming conflicts allowing the relying party and target a means to enforce a common understanding of measurement objects. A working implementation of ISAKMP is strongSwan (Steffen, 2021) which uses IKE as for key management and authentication protocols (Carrel and Harkins, 1998).

2.2 Copland

Copland (Ramsdell et al., 2019; Helble et al., 2021) is a formally specified, domain specific language designed specifically for attestation protocols. Copland interpreters are available on multiple execution platforms (Petz and Alexander, 2019); Copland's formal semantics denotationally and operationally define protocol execution (Ramsdell et al., 2019) and are captured and verified in Coq (Bertot and Castéran, 2013). Copland is effectively *parameterized over work* allowing arbitrary, distributed measurements over complex systems (Helble et al., 2021).

The Copland grammar allows for Copland *phrases* to specify measurement place, measurement target, and any meta-evidence, such as signatures or nonces. It also allows for the combination of measurements through sequencing operators. The grammar appears as follows:

$$t \leftarrow A \mid @_p\, t \mid (t \rightarrow t) \mid (t \overset{\pi}{\prec} t) \mid (t \overset{\pi}{\sim} t)$$
$$A \leftarrow \mathsf{ASP}\ m\ \bar{a}\ p\ r \mid \mathsf{CPY} \mid \mathsf{SIG} \mid \mathsf{HSH} \mid \cdots$$

Fig. 2: Copland Phrase Grammar

The terminal A is used to specify measurement operations through ASP invocation and meta-evidence. Attestation service providers (ASPs) are minimal work units that are necessary to preform measurement operations. Meta-evidence is an operation over evidence such as hashing or signing that enhances trustworthiness. The nonterminal t allows for terminal measurements to be combined through sequencing ($\rightarrow$) and parallel operators ($\overset{\pi}{\prec}$ and $\overset{\pi}{\sim}$). $@_p\, t$ is necessary for dispatching measurement operations to distinct attestation managers present within the attestation system. A term in the language may appear as follows:

$$\texttt{@p (m } \bar{a} \texttt{ q t)}$$

where `p` is the measurement place, `m` is the ASP or mechanism responsible for preforming measurement, and `q` is the place where the measurement target, `t` is located (Petz and Alexander, 2019). $\bar{a}$ is a list of input arguments necessary for measurement. For the remainder of this work, we assume no measurements require input evidence and can therefore safely omit this field.

Layered attestation is the act of combining various measurements across platforms to provide a more comprehensive view of the target. For example, we can strengthen the previous Copland phrase by first measuring t's operational environment. Say this operational environment has an attestation manager located at place `s` with some ASP `aOS` that measures the operating system target `os`. We can linearly sequence the measurements to make stronger guarantees about `t` with the following Copland phrase:

$$\texttt{@os (aOS s os)} \rightarrow \texttt{@p (m q t)}$$

Helble et al. (2021) reasoned about a variety of measurement orderings for different attestation scenarios eventually producing the flexible mechanisms. This

work introduced a collection of formally defined attestation scenarios where one such scenario is the certificate style attestation. In this case, a relying party wishes to determine trust in some attester through the appraisal of evidence where the result is summarized as a certificate.

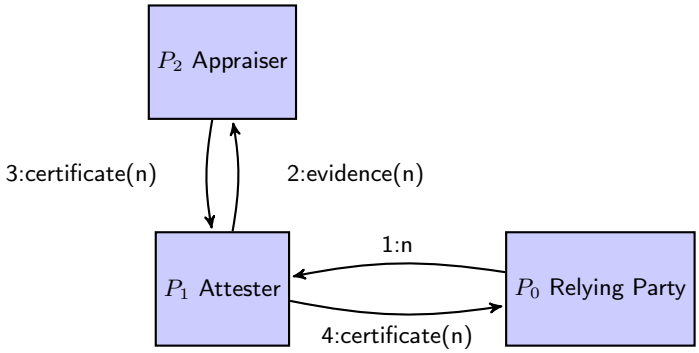

Fig. 3: Certificate-style remote attestation mechanism as seen in (Helble et al., 2021).

In the certificate style presented in Figure 3, the attester (previously target) preforms a measurement of their system. The appraiser is some trusted party which the attester knows of but runs a separate attestation manager to verify the attester's evidence value(s). P_0 begins the attestation sequence by prompting P_1 with a nonce and some measurement request. It is important to note that, without negotiation, the requested measurement operation must be provisioned by a knowledgeable user who has identified said phrase as executable. P_1 performs the measurement request to generate evidence that is then sent to be appraised by P_2. P_2 appraises the evidence and generates a certificate that contains the appraisal result. The certificate is sent back to P_1 where it is forwarded to P_0. The Copland protocol for this scenario appears as follows (Helble et al., 2021):

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

These flexible mechanisms are important not only because they distinguish various attestation scenario but they also introduce the idea that the attester and appraiser may be distinct. That is, in some situations, the appraiser and relying party are conflated but here they are disjoint. This motivates the need to capture each attestation component's capabilities in a formal way such that existing peers are able to realize their capabilities to formulate meaningful measurement.

2.3 Manifests

Manifests describe attestation manager specific information and minimally include a list of existing and operational ASPs, a context relation, a knows of

relation, and a local policy. The list of *ASPs* must be explicitly captured to realize measurement capabilities of an attestation manager. To understand the attestation manager's dependencies, we introduce the *context* relation (C) which lists all attestation managers which the current attestation manager depends on. This distinction is motivated by the idea that a trust decision of any higher level component depends on the assurance of the lower level components. To capture existing attestation managers which the current attestation manager is aware of and can request measurement from, we introduce the *knowsOf* relation (K). Capturing this relation is necessary to ensure we can preform @ operations to gain a comprehensive system view. The *local policy* is a context specific policy that applies constraints to measurement operations. The target's local policy is their privacy policy. It is enforced to uphold the principle of constrained disclosure and as such is a means to distinguish sensitive information in the context of some relying party. The relying party's local policy is their selection policy. This policy is applied to select a protocol that meets the goal of comprehensive information and as such describes the sufficiency of a protocol in some context.

Below is our formalization of manifests using Coq. We abstractly reason about other attestation managers using *Plc*. Currently, policy here represents the privacy policy as such representation is necessary for reasoning during *refinement*. The policy is written relationally and states which ASPs can share measurements with other specified peers (represented as *Plc*).

```
Record Manifest := {
    ASPs : list ASP ;
    K : list Plc ;
    C : list Plc ;
    Policy : ASP → Plc → Prop ; }.
```

An *environment* is a set of AM's each defined by a manifest. The domain of an environment provides names for each manifest. A collection of environments is known as a *system*. Within Coq, we realize these structures formally below.

```
Definition Environment : Type := Plc → (option Manifest).
Definition System := list Environment.
```

For implementation purposes, a manifest also includes public keys, addresses, and TPM initialization information. A future goal is to abstractly write manifests and compile them into attestation components. This is currently out of scope.

3 Negotiation

We introduce negotiation to provide communicating peers a means to mutually determine an attestation protocol that correctly describes the target's infrastructure, is executable on the target system, and introduces a mechanism by which the target can uphold their goal of constrained disclosure. These three goals can be formally defined as sufficiency, executability, and soundness. Our negotiation

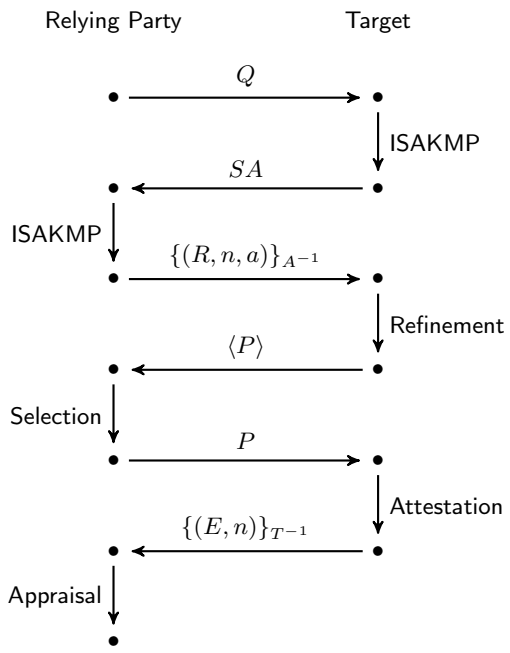

Fig. 4: Processing sequence for Negotiation, Selection, Attestation and Appraisal during remote attestation.

procedure aims to satisfy these three properties through the following protocol presented in Figure 4.

Negotiation begins with the establishment of a security association through ISAKMP. This multi-step procedure includes sharing necessary information, such as keys and identities, to establish a secure connection. Additional shared information includes instantiating the situation which is a function of identity and context. Together, these fields ultimately result in a security association SA which is valid for some predetermined length of time.

Once the network traffic is protected under the SA, the relying party sends a request ($\{(R, n, a)\}_{A^{-1}}$) which lists potentially many preferred measurement operations. The request is composed of the following four attributes:

- n – nonce
- R – list of protocols (Copland phrases)
- a – situational identifier
- $\{\cdot\}_{A^{-1}}$ – signature (by appraiser or relying party)

The most important piece of the request is R: a list of protocols which describe specific ASPs and measurement targets together as Copland phrases. Requesting evidence might be a useful concept but evidence reflects concrete values and we believe it is more useful to request abstract values in the form of Copland phrases. With the measurement operations abstractly outlined in R, the

target has the opportunity to map requested operations to one or many suitable attestation components such that the request is satisfied. In addition to R, the request includes a situational identifier a previously determined during ISAKMP. a is necessary to describe the context of the attestation and may be used to help the communicating peers realize policy. The request also includes a nonce n and signature $\{\cdot\}_{A^{-1}}$ which are necessary for security purposes namely freshness and authentication.

The process of mapping requested terms to existing, executable, and sound protocols is called *refinement*. Once the target receives a request, they may refine said request using specifications outlined in the *System* to generate terms that are either specifically requested or variations of those requested. The target may reason about existing attestation components using the list of existing ASPs, the $knowsOf$ relation, and *context* relation found within the manifest. The target may combine measurement operations within the manifests, within the environment, or within the system through Copland sequencing and parallel operators. As protocols are gathered for the proposal, the target recursively applies its privacy policy to ensure proposed protocols are sound. The target also recursively proves any proposed term is executable. Refinement ultimately produces the proposal $\langle P \rangle$ which is a collection of protocols that satisfy the request, are executable on the target system, and do not expose sensitive information.

Once the relying party receives the proposal, they must select the best protocol for attestation. To do so, the relying party orders the protocols based on some situationally determined criteria realized by the selection policy. One obvious ordering respects the goal of comprehensive information where the best protocol is the most comprehensive. Another possible ordering may be one that respects resource consumption and thus the best protocol is the one that consumes the fewest system resources. For any of these potential orderings, the arrangement of protocols naturally forms a lattice structure where the best protocol is at the top of the lattice and an empty protocol, or failed negotiation, is at the bottom.

The selection of a best protocol p ends the negotiation procedure. The relying party sends p to the target, signifying the beginning of the attestation sequence. Once received, the target preforms measurement operations specified in p to generated some evidence E. The evidence is packaged $(\{(E, n)\}_{T^{-1}})$ and delivered to an appraiser for assessment. The evidence package consists of:

- n – nonce
- E – bundled evidence
- $\{\cdot\}_{T^{-1}}$ – signature (by the target)

where the nonce is the same nonce sent in the attestation request to ensure freshness. Bundled evidence (Rowe, 2016b) is evidence gathered by the appraisal and the signature ensures authenticity and integrity of the evidence.

4 Verification

The commuting diagram in Figure 5 shows the relationship between attestation requirements and a correct implementation. At the requirements level a request

is received and transformed by the attestation system into evidence as originally defined in Figure 1. The evidence lattice $(E, \preceq, \top, \bot)$ captures evidence ordering reflecting the relying party's need for comprehensive information.

At the implementation level, a collection of ASPs run to generate measurements that are bundled into an evidence package and returned. If the implementation is correct, the correct ASPs run on the correct places in the correct order to generate measurements.

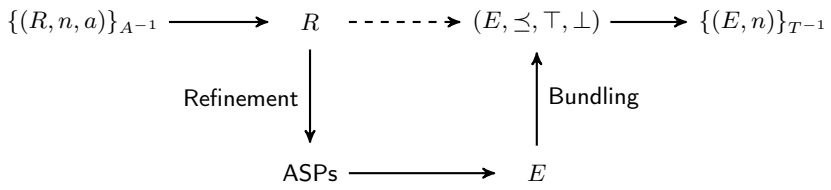

Fig. 5: Commuting diagram showing attestation correctness.

Verifying an implementation against requirements directly is infeasible, thus we work step-wise from requirements to implementation through a series of refinement steps. The certification stack in Figure 6 is a commuting diagram capturing attestation as defined in Figure 1 at multiple abstractions. Using the canonical approach for compiler verification, we verify each refinement against its requirements. Because each refinement serves as requirements for the next refinement, the separate verification steps plug together into the stack from requirements to implementation shown in Figure 6.

Figure 6 can be decomposed into two major activities. The request, R, through Selection producing P is *negotiation* while protocol P through ASPs is *execution*. Protocol execution is performed by compiling and running on an attestation manager. The attestation manager has been verified in earlier work by Petz and Alexander (2022). Thus, if we can verify a correct protocol results from negotiation, we will have a fully verified attestation process.

Primary requirements for negotiation are choosing a protocol that: (i) satisfies constrained disclosure requirements on the target; and (ii) satisfies comprehensive information requirements on the relying party. We refer to these properties as *soundness* and *sufficiency* respectively. Soundness is defined in terms of *executability* and *privacy policy enforcement*. Sufficiency is defined in terms of evidence ordering $(E, \preceq, \top, \bot)$.

Executability is a static guarantee that a proposed protocol can run on a target. This can be confirmed by knowing all protocol ASPs are available for execution on the target and all places referenced by dispatch commands are known. A manifest's `ASPs` list contains the set of available ASPs and its `knowsOf` list contains the set of attestation managers it can communicate with. The definition of executable is recursive implying that when a target dispatches a phrase to another attestation manager, that phrase must in turn be executable.

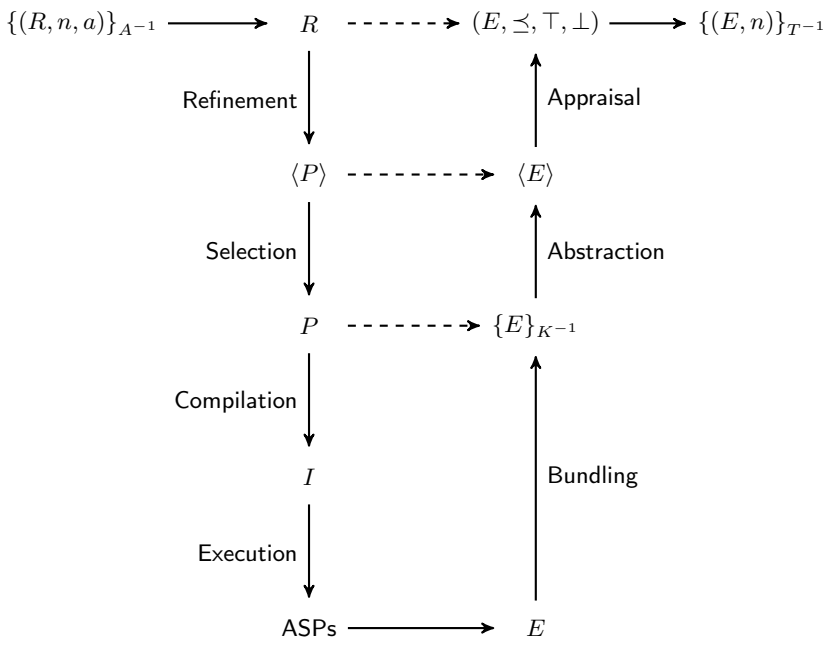

Fig. 6: Certification stack showing certification dependencies and execution path. Solid lines represent implementations while dashed lines represent mathematical definitions.

Definition 1 (Executability). *A protocol is* executable *with respect to a target, manifest, and environment if: (i) all necessary ASPs are available; (ii) all specified places are known, and (iii) all dispatched protocols are executable on their target.*

We formally represent executability with the Coq definition *executable*. One can see that given a term, place, and environment, any term may be recursively evaluated determine executability. For any term with an ASP, executable checks that the environment has the ASP. For any $@p(t)$ operation, *executable* ensures that the requesting place knows of the receiving place and that the term is executable at the receiving place. For sequencing and parallel operations, executability recurses into subterms.

We prove the decidability of executable with the theorem *executable_dec* from Figure 7.[1] *executable_dec* must produce a proof that *executable* is either true or false for every combination of term, place, and environment.

An executable protocol is *sound* if it does not violate the privacy policy of its target, including protocols dispatched to other attestation managers. To

[1] For those unfamiliar with Coq, the type $\{p\} + \{\sim p\}$ is a sum type whose values are lifted proofs of p or $\sim p$.

```
Fixpoint executable(t:Term)(k:Plc)(e:Environment):Prop :=
match t with
| asp a ⇒ hasASPe k e a
| att p t ⇒ knowsOfe k e p → executable t p e
| lseq t1 t2 ⇒ executable t1 k e ∧ executable t2 k e
| bseq _ t1 t2 ⇒ executable t1 k e ∧ executable t2 k e
| bpar _ t1 t2 ⇒ executable t1 k e ∧ executable t2 k e
end.

Theorem executable_dec:∀ t k e,{(executable t k e)}+{~(executable t k e)}.
```

Fig. 7: Definition and decidability theorem for *executable*.

reason about soundness, we must first capture policy to realize if protocols expose sensitive information.

Definition 2 (Privacy Policy). *Privacy policy applies to the target and relationally defines the information that may be shared with a communicating peer.*

Local policy is specified by the manifest as a relation among ASPs and places. Local policy, for the target, is synonymous with privacy policy. If the relation (*Policy a p*) exists and is specified in the manifest then a protocol from place p may ask the current attestation manager to run a. If all ASPs referenced by a protocol are allowed by a place's local policy, then that protocol may run on that place.

Like executability, we prove that policy enforcement is decidable. Given any local policy, p, any place *plc*, and any requesting ASP *asp*, it is decidable whether the request is allowed:

```
∀ p asp plc,{(p asp plc)}+{~(p asp plc)}.
```

Proving this theorem requires that the local policy relation be decidable. Specifically, for any ASP and place, it must be decidable whether the place can request an ASP's execution. When the local policy is defined as an inductive relation this proof is bulky, but straightforward. We have many examples of decidability proofs for individual policies and a rough Ltac tactic. However, when local policy is stated as an unqualified relation, we have not found a proof.

If we assume the local policy application is decidable we can prove decidability of access control checks over full protocols. The proof takes a form similar to the proof of executability:

```
Theorem checkTermPolicy_dec:∀ t k e,
    (∀ p0 e0 a0, {(checkASPPolicy p0 e0 a0)} + {~(checkASPPolicy p0 e0 a0)}) →
    {(checkTermPolicy t k e)}+{~(checkTermPolicy t k e)}.
```

The trade-off is assuming local policy decidability places the burden of proof on the system implementer. Before a full protocol can be checked for policy adherence, the implementer needs to provide a proof that local policy defined

over ASPs is decidable. With this assumption satisfied, policy enforcement over protocol terms is decidable and may be statically checked prior to execution.

With executability and policy enforcement proved decidable, it is a simple step to prove that *soundness* is decidable. Informally, soundness is defined as the combination of decidability and policy adherence:

Definition 3 (Soundness). *A protocol is* sound *with respect to a target manifest if: (i) it is executable; and (ii) it does not violate local privacy policy.*

The Coq definitions of *sound* and the associated theorem take a form similar to executable and policy adherence. However, work to prove decidability and policy adherence substantially reduce the complexity of the proof.

```
Definition sound (t:Term)(k:Plc)(e:Environment) :=
  (executable t k e) ∧ (checkTermPolicy t k e).

Theorem sound_dec: ∀ t p e,
    (∀ p0 a0, {(checkASPPolicy p0 e a0)} + {~(checkASPPolicy p0 e a0)})
    → {sound t p e}+{~(sound t p e)}.
```

We have shown that for any protocol, place, and environment soundness is decidable and may be determined statically given local policy enforcement is statically decidable. Thus, our system can determine soundness during negotiation, prior to execution and use the result to help select a protocol. Furthermore, Curry-Howard allows using decidability proofs as functions. A call to *sound_dec t p e* will determine if protocol t running at place p in environment e is sound. Furthermore, the soundness check can be synthesized to CakeML for inclusion in our fielded attestation manager implementation (**?**).

While soundness applies to the target, *sufficiency* applies to the relying party. If a protocol produces evidence that is *sufficient* for its trust assessment then the protocol is sufficient for the relying party.

Definition 4 (Sufficiency). *A protocol is* sufficient *with respect to a relying party if it meets the relying party's comprehensive information requirements.*

The evidence lattice $(E, \preceq, \top, \bot)$ shown in Figure 5 captures comprehensive information requirements of the relying party as an ordering of evidence types producible by available protocols. The Copland semantics (Ramsdell et al., 2019) defines a formal evidence semantics that maps a protocol to its evidence type. Thus, given an evidence type we know which protocol(s) produced it and can use the evidence lattice to determine protocol sufficiency. Unfortunately, Rowe (2016a,b) demonstrates that under certain conditions the same evidence type might be produced by different protocols. This problem can be avoided by restricting protocols considered, but the burden of this task falls to the user.

In the evidence lattice E is the set of evidence produced by protocols considered in negotiation with a target. E is partially ordered by $\preceq$, a partial order we refer to informally as the evidence order. $\top$ and $\bot$ define all evidence and no evidence respectively as required by the lattice definition.

Every negotiation defines evidence type $e_{min} : E$ that specifies minimally sufficient comprehensive information for the relying party. Any evidence e such that $e_{min} \preceq e$ is acceptable to the relying party and satisfies it's comprehensive information requirements. In turn, any protocol producing e is considered *sufficient* for the situation.

Protocol soundness and sufficiency are separate and potentially conflicting goals. A simple negotiation correctness definition requires producing a protocol that is both sound and sufficient.

Definition 5 (Negotiation Correctness). *Negotiation between a relying party and a target is* correct *if it produces a sound and sufficient protocol.*

This definition holds whenever negotiation produces a protocol that meets minimum requirements of a relying party while satisfying constrained disclosure and executability for the target.

While soundness and sufficiency are both statically predicted, Rowe (2016a,b)'s result establishing that the same evidence can be generated by different protocols implies that we cannot infer from otherwise good evidence that the correct protocol ran. Similarly, we cannot infer from evidence that a target's privacy policy is actually enforced. Thus, we are obligated to gather meta-evidence of protocol execution in addition to evidence of system behavior.

5 Example

An example attestation scenario used across our work is a relying party determining if an attester is running an acceptable, properly configured virus checker (Petz and Alexander, 2022). Figure 8 shows this architecture where P_1 is the location of the attestation manager responsible for measurement of the virus checker and local signature file, while P_2 is the location of the attestation manager responsible for measuring the signature file server.

Manifests for this scenario are specified as (with reformatting for readability):

P_0:={|asps:=[]; K:=[P_1]; C:=[]; Policy:={}|}
P_1:={|asps:=[aVC,aHSH]; K:=[P_0,P_2]; C:=[]; Policy:={(P_0 aVC),(P_0 aHSH)}|}
P_2:={|asps:=[aSFS]; K:=[P_1]; C:=[]; Policy:={(P_1 aSFS)}|}

Three Copland phrases describe potential attestation protocols available to P_0 for inclusion in proposals:

p_0 = @P_1 [(aVC P_1 vc)]
p_1 = @P_1 [(aVC P_1 vc) → (aHSH P_2 sf)]
p_2 = @P_1 [(aVC P_1 vc) → (aHSH P_2 sf) → @P_2 (aSFS P_2 sfs)]

In protocol p_0, the ASP aVC is used to measure vc, the virus checking target located at P_0. In p_1, the virus checking measurement is followed by aHSH, a measurement that hashes the signature file sf co-located at P_1. Finally, in p_2, the protocol invokes aVC and aHSH followed by a remote call to P_2 to invoke ASP aSFS to measure the signature file server sfs itself. While we are not limited to three protocols, this set defines a collection of increasingly informative attestations.

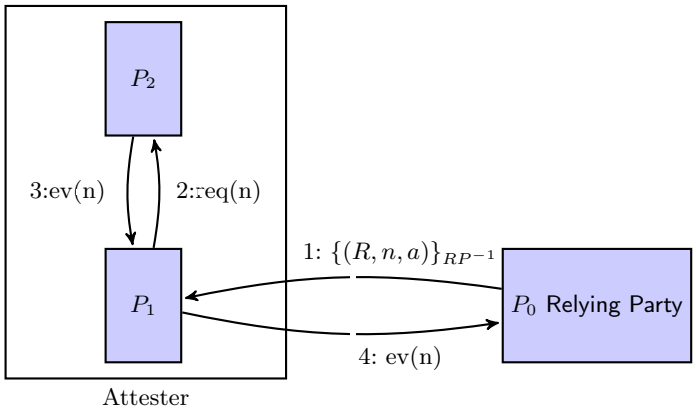

Fig. 8: Example remote attestation mechanism motivated by the flexible mechanisms presented in (Helble et al., 2021).

To begin negotiation the relying party sends the request $\{(R, n, a)\}_{RP^{-1}}$ to P_1 for a measurement of the target's virus checker where R is the request for measurement. When P_1 receives the request, it returns a subset its three protocols. The relying party selects a protocol based on soundness and sufficiency. In the current configuration, all three protocols are executable and satisfy policy and are thus selectable. However if the `aHSH` entry were removed from P_1's policy, protocols p_1 and p_2 would not be selectable because access control will not allow them to run. Similarly, if `aSFS` were not available to P_2, protocol p_2 would not be selected because the required ASP is not available.

The verified function *sound_dec* determines soundness correctly for each protocol. However, it is unwieldy as directly stated. Two helper functions customize *sound_dec* for each system being examined. *sound_local_policies* takes a specific environment and ensures *checkASPPolicy* is decidable. The resulting lemma is use to instantiate the assumption of *sound_dec* resulting in *sound_system_dec* that determines if a protocol is sound with respect to a place given the instantiated environment.

An alternative negotiation approach builds a proposal for each request. The *knowsOf* relation reveals P_1 knows of P_2 and therefore may request a measurement involving the signature file server in addition to its local measurements. The attester generates the protocol, $\langle p_1, p_2 \rangle$, for the request. Since p_2 is a more complex measurement, it may expose sensitive information in the context of P_0 not suitable for all appraisers. Privacy policy can be modified such that p_2 is provably unsound and thus not included in the proposal. However, both p_0 and p_1 are proven sound and included in the proposal. This produces in the following proposal $\langle p_0, p_1 \rangle$. Again, our verified soundness decision procedure guarantees the soundness of each proposed protocol.

In this example, the evidence latticed used by the relying party is trivial. If we comprehensive information is the goal, then our protocols should be ordered

$p_0 \preceq p_1 \preceq p_2$ from least information to most with $\top = p_2$ and $\bot = \emptyset$. The relying party sets e_{min}, the evidence in `ev(n)`, appropriately for the situation.

6 Related Work

Existing work in attestation protocol negotiation is minimal. (Helble et al., 2021) briefly introduce the negotiation scheme stating that it is needed to mutually satisfy the situational requirements of the target and appraiser. The specific formalisms and overall scheme of negotiation are not considered in detail. Pendergrass et al. (2017) introduce Maat, a measurement and attestation platform that performs negotiation for security services such as cryptographic hashing and signing algorithms. Similar to the negotiation work done in Maat, Huang and Peng (2009) introduce an automated negotiation model which exchanges keys and identities to authenticate attesting peers. ISAKMP Maughan et al. (1998) and IKE Carrel and Harkins (1998) both perform limited negotiation for defining security associations, but at an earlier stage in the communication process.

Protocol analysis is a critical part of selecting a situationally best protocol. In works by Rowe (2016a), authors introduce a mathematical representation of protocols concluding that measurement order directly impacts the protocols vulnerability to an adversary. CHASE is a first order model finder used to reason about protocol ordering in the context of an attacker (Ramsdell, 2020). Petz et al. (2021) detail the process of using CHASE to instantiate such reasoning. Baez (2022) also reason about Copland protocols using CHASE. In both works by Petz et al. (2021) and Baez (2022),authors provision Copland phrases as knowledge system users rather than mutually agreed upon protocols.

VRASED (**?**) and HYDRA **?** are remote attestation systems using formal verification in their design. VRASED uses a co-design approach that results in formally verified attestation systems. Similarly, HYDRA uses a verified microkernel to achieve attestation guarantees. Both of these projects focus on attestations during boot and system initialization in contrast to this work that focuses on layered, runtime attestation.

7 Conclusion and Future Work

We propose a negotiation framework allowing communicating peers to establish a mutually agreed upon attestation protocol satisfying a constrained disclosure and comprehensive information requirements. Using system manifests that describe attestation managers we derive a decision procedure for soundness and define an ordering relationship to establish sufficiency. Thus, for any communicating peers, an attestation protocol can be selected such that the goals of constrained disclosure and comprehensive information hold prior to execution.

This work represents an intermediate step in our longer term goal of verified attestation systems. We are currently exploring synthesis and selection of a optimal attestation protocol that satisfies semantic appraisal goals in addition to soundness and sufficiency. Additionally, we are investigating techniques for automatically generating proposals to reduce the burden prior to protocol selection. Finally, we are developing techniques for verifying manifest compilation to ensure implementations satisfy requirements.

Bibliography

Baez, F.E.V.: Evaluating sgx's remote attestation security through the analysis of copland phrases (2022)

Bertot, Y., Castéran, P.: Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions. Springer Science & Business Media (2013)

Carrel, D., Harkins, D.: The Internet Key Exchange (IKE). RFC 2409 (nov 1998), https://www.rfc-editor.org/info/rfc2409

Coker, G., Guttman, J., Loscocco, P., Herzog, A., Millen, J., O'Hanlon, B., Ramsdell, J., Segall, A., Sheehy, J., Sniffen, B.: Principles of remote attestation. International Journal of Information Security 10(2), 63–81 (June 2011)

Coker, G.S., Guttman, J.D., Loscocco, P.A., Sheehy, J., Sniffen, B.T.: Attestation: Evidence and trust. In: Proceedings of the International Conference on Information and Communications Security. vol. LNCS 5308 (2008)

Haldar, V., Chandra, D., Franz, M.: 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 (May 2004)

Helble, S.C., Kretz, I.D., Loscocco, P.A., Ramsdell, J.D., Rowe, P.D., Alexander, P.: Flexible mechanisms for remote attestation. ACM Transactions on Privacy and Security (TOPS) 24(4), 1–23 (2021)

Huang, X., Peng, Y.: An effective approach for remote attestation in trusted computing. WISA 2009: Proceedings of the 2nd International Symposium on Web Information Systems and Applications (01 2009)

Martin, A., et al.: The ten page introduction to trusted computing. Tech. Rep. CS-RR-08-11, Oxford University Computing Labratory, Oxford, UK (2008)

Maughan, D., Schertler, M., M., S., Turner, J.: Internet security association and key management protocol rfc 2048 (isakmp). Tech. rep., The Internet Engineering Task Force of The Internet Society (November 1998)

Pendergrass, J.A., Helble, S., Clemens, J., Loscocco, P.: Maat: A platform service for measurement and attestation. arXiv preprint arXiv:1709.10147 (2017)

Petz, A., Alexander, P.: A copland attestation manager. In: Hot Topics in Science of Security (HoTSoS'19). Nashville, TN (April 8-11 2019)

Petz, A., Alexander, P.: An infrastructure for faithful execution of remote attestation protocols. Innovations in Systems and Software Engineering (2022)

Petz, A., Jurgensen, G., Alexander, P.: 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. p. 111–117. MEMOCODE '21, Association for Computing Machinery, New York, NY, USA (2021), https://doi.org/10.1145/3487212.3487340

Ramsdell, J., Rowe, P.D., Alexander, P., Helble, S., Loscocco, P., Pendergrass, J.A., Petz, A.: Orchestrating layered attestations. In: Principles of Security and Trust (POST'19). Prague, Czech Republic (April 8-11 2019)

Ramsdell, J.: Chase: A model finder for finitary geometric logic. https://github.com/ramsdell/chase (2020)

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

Rowe, P.D.: Bundling Evidence for Layered Attestation. In: Trust and Trustworthy Computing, pp. 119–139. Springer International Publishing, Cham (Aug 2016b)

Steffen, A.: strongswan. https://www.strongswan.org/ (Oct 2021), accessed: 2022-06-29