Bundling Evidence for Layered Attestation

Paul D. Rowe
prowe@mitre.org

The MITRE Corporation, Bedford, MA USA

Abstract. Systems designed with measurement and attestation in mind are often layered, with the lower layers measuring the layers above them. Attestations of such systems, which we call *layered attestations*, must bundle together the results of a diverse set of application-specific measurements of various parts of the system. Some methods of layered attestation are more trustworthy than others especially in the presence of an adversary that can dynamically corrupt system components. It is therefore important for system designers to understand the trust consequences of different designs. This paper presents a formal framework for reasoning about layered attestations. We identify inference principles based on the causal effects of dynamic corruption, and we propose a method for bundling evidence that is robust to such corruptions.

1 Introduction

Security decisions often rely on trust. Many computing architectures have been designed to help establish the trustworthiness of a system through remote attestation. They gather evidence of the integrity of a target system and report it to a remote party who appraises the evidence as part of a security decision. A simple example is a network gateway that requests evidence that a target system has recently run antivirus software before granting it access to a network. If the virus scan indicates a potential infection, or does not offer recent evidence, the gateway might decide to deny access, or perhaps divert the system to a remediation network. Of course the antivirus software itself is part of the target system, and the gateway may require integrity evidence for the antivirus for its own security decision. This leads to the design of layered systems in which deeper layers are responsible for generating integrity evidence of the layers above them.

A simple example of a layered system is one that supports "trusted boot" in which a chain of boot-time integrity evidence is generated for a trusted computing base that supports the upper layers of the system. A more complex example might be a virtualized cloud architecture. The virtual machines (VMs) at the top are supported at a lower layer by a hypervisor or virtual machine monitor. Such an architecture may be augmented with additional VMs at an intermediate layer that are responsible for measuring the main VMs to generate integrity evidence. These designs offer exciting possibilities for remote attestation. They allow for specialization and diversity of the components involved, tailoring the capabilities of measurers to their targets, and composing them in novel ways.

An important fact about such layered systems is that the trustworthiness of the system is not simply a function of the evidence produced by measurement; the relative order of the measurement events is crucial. In particular, a strong intuition that is manifest in the literature is that it is better to build trust "bottom-up" by first gathering evidence for components lower in the system *before* they measure the higher level components. A measurer is more likely to be uncorrupted at the time it takes its measurements if this order is respected. This intuition for "bottom-up" measurement underlies many architectures, most notably trusted boot [9] and the integrity measurement architecture (IMA) [15]. In a companion paper [14] we characterize the guarantees provided by a bottom-up measurement scheme in the presence of an adversary that can dynamically corrupt system components. Namely, if an adversary successfully corrupts a target component t without being discovered by measurements, then the adversary must have either performed a *recent* corruption of one of t's immediate dependencies, or else the adversary must have corrupted one of t's indirect dependencies *deeper* in the system. Thus bottom-up measurements confine undetectable corruptions to be either recent or deep. We schematize the main theorem of [14] in Eqn. (1).

$$\text{Bottom-up measurement} \Longrightarrow \text{Detectable, Recent or Deep} \tag{1}$$

Such a result is not enough, however. Since a remote appraiser cannot directly observe the order of measurements on a system, this information must be part of what is conveyed in the bundle of evidence during the attestation. In order to apply the result, the appraiser needs a way of inferring that the measurements were indeed taken bottom-up. If an adversary could make it look like measurements were taken in the desired order when they weren't then he could avoid the consequences of the theorem.

Much of the work on measurement and attestation relies on a Trusted Platform Module (TPM) to protect and report the evidence generated by measurement components. It is common to invoke the use of a TPM as sufficient for these purposes. Unfortunately, there are many natural ways to use a TPM that fail to accurately reflect the order of measurement. The ability of an adversary to dynamically corrupt components at runtime makes the problem all the more pronounced. This paper begins to address the issues surrounding the use of TPMs to bundle evidence in the presence of dynamic adversaries. We summarize our main contributions as follows:

1. We introduce a formalism for reasoning about the causal effects of dynamic corruption and repair of system components on the process of bundling measurement evidence for attestation using a TPM.
2. We prove correct a set of reusable principles for inferring the structure of system activity given that a certain structure of bundled evidence was produced by a TPM. Failure of these principles to prove some desirable property may indicate that the desirable property was not met.
3. We propose a particular method for using a virtualized TPM to bundle evidence, and we show (Theorem 5) that under some assumptions about

the behavior of uncompromised components, a remote appraiser can infer that either the measurements were taken bottom-up, or else the adversary performed a recent or deep corruption in the sense described above. Letting $\mathcal{Q}$ denote a set of quotes conforming to our method, we schematize this theorem in Eq. (2).

$$\mathcal{Q} \Longrightarrow \text{Bottom-up, Recent, or Deep} \tag{2}$$

The first two contributions are quite general, and, we believe, could be applied to the design and analysis of many systems. The third suggests a particular design recommendation. It says, roughly, that if our recommendation is followed, then either the hypothesis of Eq. (1) is satisfied, or else its conclusion is satisfied. The particular assumptions required might limit its applicability. In particular, it assumes some flexible access control to TPM registers which is hard to achieve in physical TPMs. Thus it is naturally applicable to virtualized systems incorporating virtualized TPMs (vTPMs) [1] that could allow for such access control. Although no industry standard currently exists for securing vTPMs, architectural designs and specifications for such systems are beginning to emerge [12, 13, 5, 2].

Paper structure. The rest of the paper is structured as follows. We begin in Section 2 by reviewing some basic facts about TPMs and introducing some notation. In Section 3 we build up some intuition about what types of inference an appraiser is justified in making and what types of problems can arise when using a TPM to bundle evidence from a layered system. Section 4 contains the description of our formal model which we will use to justify our intuitions. We develop our reusable principles and present our bundling strategy in Section 5, characterizing the guarantees provided by our strategy. We address related work in Section 6 before concluding.

2 Preliminaries

The results of this paper depend on some features of Trusted Platform Modules (TPMs). For reasons of space, a full review of the relevant features of TPMs is impractical. We present here only the most basic explanation of the notions necessary to proceed.

TPMs are stateful devices with a collection of platform configuration registers (PCRs) that contain information about the state of the system. These registers are isolated from the rest of the system and are thus protected from direct modification. They can only be updated in constrained ways, namely by *extending* a register or by *resetting* it. We explain below how this works. An additional restriction is imposed by a form of access control known as *locality*. This access control ensures that, for certain PCRs, only certain components with special privileges can extend or reset them. A TPM can also *quote* the state of a set of PCRs by emitting a digital signature over the contents of those PCRs. We will assume the signing key has not been compromised, as it never leaves the TPM unencrypted.

In order to describe how the state is updated and reported, we use elements of a term algebra. Terms are constructed from some base V of atomic terms using constructors in a signature Σ. The set of terms is denoted $\mathcal{T}_\Sigma(V)$. We assume Σ includes at least some basic constructors such as pairing $(\cdot,\cdot)$, signing $[\![\,(\cdot)\,]\!]_{(\cdot)}$, and hashing $\#(\cdot)$. The set V is partitioned into public atoms $\mathcal{P}$, random nonces $\mathcal{N}$, and private keys $\mathcal{K}$.

Our analysis will sometimes depend on what terms an adversary can derive (or construct). We say that term t is derivable from a set of term $T \subseteq V$ iff $t \in \mathcal{T}_\Sigma(T)$, and we write $T \vdash t$. We assume the adversary knows all the public atoms $\mathcal{P}$, and so can derive any term in $\mathcal{T}_\Sigma(\mathcal{P})$ at any time. We also assume the set of measurement values is public, so an adversary can forge acceptable evidence. We denote the set of potential measurement values for a target t by $\mathcal{MV}(t)$.

We represent both the values stored in PCRs and the quotes as terms in $\mathcal{T}_\Sigma(V)$. Extending a PCR by value v amounts to replacing its contents c with the hash $\#(v,c)$. Resetting a PCR sets its contents to a fixed, public value, say rst. Since PCRs can only be updated by extending new values, their contents form a hash chain $\#(v_n,\#(...,\#(v_1,\mathsf{rst})))$. We abbreviate such a hash chain as $\mathsf{seq}(v_1,\ldots,v_n)$. So for example, $\mathsf{seq}(v_1,v_2) = \#(v_2,\#(v_1,\mathsf{rst}))$. We say a hash chain $\mathsf{seq}(v_1,\ldots,v_n)$ *contains* v_i for each $i \leq n$. Thus the contents of a PCR contain exactly those values that have been extended into it. We also say v_i is *contained before* v_j in $\mathsf{seq}(v_1,\ldots,v_n)$ when $i < j \leq n$. That is, v_i is contained before v_j in the contents of p exactly when v_i was extended before v_j.

A quote from TPM t is a term of the form $[\![\, n,(p_i)_{i\in I},(v_i)_{i\in I}\,]\!]_{sk(t)}$. It is a signature over a nonce n, a list of PCRs $(p_i)_{i\in I}$ and their respective contents $(v_i)_{i\in I}$ using $sk(t)$, the secret key of t. We always assume $sk(t) \in \mathcal{K}$ the set of non-public, atomic keys. That means the adversary does not know $sk(t)$ and hence cannot forge quotes.

3 Examples of Weak Bundling

Before jumping into the technical details, we start with an example that illustrates some potential pitfalls of using TPMs for bundling evidence. Consider an enterprise that would like to ensure that systems connecting to its network provide a fresh system scan by the most up-to-date virus checker. The network gateway should ask systems to perform a system scan on demand when they attempt to connect. We may suppose the systems all have some component A_1 that is capable of accurately reporting the running version of the virus checker. Because this enterprise values high assurance, the systems also come equipped with another component A_2 capable of measuring the runtime state of the kernel. This is designed to detect any rootkits that might try to undermine the virus checker's system scan, for example by hiding part of the file system that contains malicious files. We may assume that A_1 and A_2 are both measured by a root of trust for measurement (rtm) such as Intel's TXT as part of a trusted boot process.

Figure 1 is a notional depiction of an architecture supporting this use case. In this architecture, the primary user virtual machine (VM) hosts the *kernel*, the virus checker *vc* and the file system *sys*. A sibling VM hosts the two measurement components A_1 and A_2. These virtual machines are managed by some hypervisor that runs on the underlying hardware containing the root of trust for measurement rtm. We have depicted a virtualized TPM (vTPM) for each VM while the hardware contains a physical TPM, although we might consider the possibility that the VMs only use the physical TPM. Such an architecture is reminiscent of those found, for example, in [4] or [2].

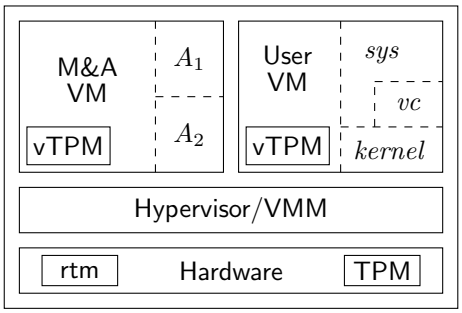


Fig. 1. Example Attestation System

If the gateway is to appraise the system, it might expect the measurements to be taken according to the order depicted in Fig. 2 (in which time flows downward). The event of o_m measuring o_t is represented by $\mathsf{ms}_{o_c}(o_m, o_t)$, where we include the subscript o_c only when it provides a clean runtime context for the measurer o_m. This order of events represents the intuitive "bottom-up" approach to measurement. It ensures that if *sys* is corrupted but not detected by the measurement event $\mathsf{ms}_{ker}(vc, sys)$ then the adversary must have either recently corrupted *vc* or *ker* or else he must have corrupted one of the more protected components A_1 or A_2 [14]. The $\mathsf{att\text{-}start}(n)$ event indicates a moment in time in which the gateway chooses a random nonce n. "Recent" corruptions are those that occur after this event. The bullet after the first three events is inserted only for visible legibility, to avoid crossing arrows.

Of course, the gateway cannot directly observe these events taking place. Rather, it must infer the order and outcome of measurements from evidence that is extended into a TPM and quoted for integrity protection. We now consider a couple natural ways one might think of doing this and point out some potential pitfalls in which the presence and order of the measurement events cannot be inferred from the quote structure.

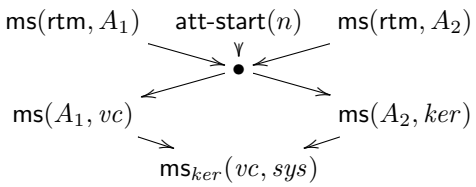


Fig. 2. Bottom-up order for measurement

Strategy 1: A single hash chain. Since PCRs contain an ordered history of the extended values, the first natural idea is for all the components to share a

PCR p, say in the physical TPM, each extending their measurements into p. The intuition is that the contents of p should represent the order in which the measurements occurred on the system. To make this more concrete, assume the measurement events of S_1 have the following results: $\mathsf{ms}(\mathsf{rtm}, A_1) = v_1, \mathsf{ms}(\mathsf{rtm}, A_2) = v_2, \mathsf{ms}(A_1, vc) = v_3, \mathsf{ms}(A_2, ker) = v_4, \mathsf{ms}(vc, ker) = v_5$. Then this strategy would produce a single quote $Q = [\![n, p, \mathsf{seq}(v_1, v_2, v_3, v_4, v_5)]\!]_{sk(t)}$. To satisfy the order of Fig. 2, any linearization of the measurements would do, so the appraiser should also be willing to accept $Q' = [\![n, p, \mathsf{seq}(v_2, v_1, v_3, v_4, v_5)]\!]_{sk(t)}$ in which v_1 and v_2 were generated in the reverse order.

Figure 3 depicts an execution that produces the expected quote Q, but does not satisfy the desired order. Since all the measurement components have access to the same PCR, if any of those components is corrupted, it can extend values to make it look as though other measurements were taken although they were not. Since the bottom-up order of measurement was not respected, the conclusions from [14] cannot be applied. Indeed, neither of the corruptions in Fig. 3 are recent. It is also troublesome since the adversary does not need to corrupt the relatively deep components A_1 or A_2. The corrupted vc, having access to p can extend the expected outcomes of measurement by A_1 and A_2 without those components even being involved.

Strategy 2: Disjoint hash chains. The problem with Strategy 1 seems to be that PCR p is a shared resource for many components of the system that should be trusted to varying degrees. The corruption of *any* component that can extend into the PCR can affect the results. This motivates a desire to separate access to the relevant PCRs, so that, in the extreme case, there is only a single component with the authority to extend each PCR. This could be done by making use of the virtualization architecture and vTPMs to ensure that each VM can only interact with its corresponding vTPM. Indeed, this separation may be much more natural for the architecture described above. The vTPM may further impose access control in the form of locality constraints for PCRs. Although locality is a relatively limited form of access control for physical TPMs, one opportunity provided by vTPMs is a more flexible notion of locality.

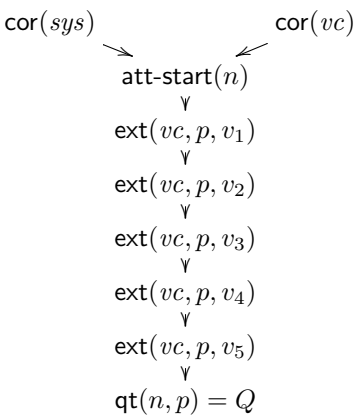


Fig. 3. Defeating Strategy 1

A natural next attempt given this assumption would be to produce three quotes, one from each (v)TPM over the set of PCRs that contain the measurement evidence. This would produce the quotes $Q_1 = [\![n, p_r, \mathsf{seq}(v_1, v_2)]\!]_{sk(t)}$, $Q_2 = [\![n, (p_1, p_2), (\mathsf{seq}(v_2), \mathsf{seq}(v_3))]\!]_{sk(vt_1)}$, $Q_3 = [\![n, p_{vc}, \mathsf{seq}(v_4)]\!]_{sk(vt_2)}$. Fig-

ure 4 demonstrates that the appraiser is not justified in inferring a bottom-up order of measurement from this set of quotes. The problem, of course, is that, since the PCRs may be extended concurrently, the relative order of events is not captured by the structure of the quote. An adversary may thus be able to alter the order in which these events take place, taking advantage of the different order to avoid detection by measurement. For example he could repair a corrupted *vc* just in time for it to be measured by A_1 so that it appears uncorrupted, when in fact it was previously corrupted when it performed its own measurement of *sys*.

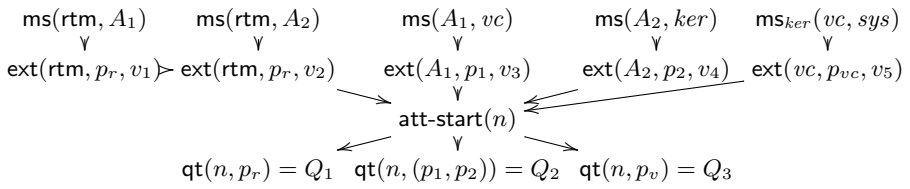


Fig. 4. Defeating Strategy 2

4 Attestation Systems

In this section we formalize the notions we used for the example in Section 3.

System architecture. We start with a definition of attestation systems that focuses on the relevant dependencies among components.

Definition 1. *An* attestation system *is a tuple* $\mathcal{AS} = (O, M, C, P, L)$*, where* O *is a set of objects (e.g. software components) with a distinguished element* rtm*.* M *and* C *are binary relations on* O*. We call*

M the measures *relation, and*
C the context *relation.*

$P = T \times R$ for some set T of TPMs and some index set R of their PCR registers, and L is a relation on $O \times P$.

M represents who can measure whom, so that $M(o_1, o_2)$ iff o_1 can measure o_2. rtm represents the root of trust for measurement. C represents the kind of dependency that exists between *ker* and *vc* in the example from the previous section. In particular, the *vc* depends on *ker* to provide it a clean runtime context. We can thus capture the fact that a corrupted *ker* can interfere with the *vc*'s ability to correctly perform measurements, for example by hiding a portion of the filesystem from *vc*. Many assumptions one might make about M and C

affect the dynamics of corruption on the outcome of measurement. This current paper instead focuses on the bundling of evidence, and so we make only a minor assumption that $M \cup C$ is acyclic. This ensures that the combination of the two dependency types does not allow an object to depend on itself. Such systems are stratified, in the sense that we can define an increasing set of dependencies as follows.

$$D^1(o) = M^{-1}(o) \cup C^{-1}(M^{-1}(o))$$
$$D^{i+1}(o) = D^1(D^i(o))$$

So $D^1(o)$ consists of the measurers of o and their context. Elements of P have the form $p = t.i$ for $t \in T$ and $i \in R$. The relation L represents the access control constraints for extending values into TPM PCRs. We assume each component in O can only access a single TPM, so that if $L(o, t.i)$ and $L(o, t'.i')$, then $t = t'$. As discussed in the example of Section 3, it may be desirable to have a relatively strict access control policy L. We can represent the extreme case in which each component has access to its own PCR by adding the assumption that L is injective. That is, if $L(o, p)$ and $L(o', p)$ then $o = o'$. Of course relaxations of this strict policy are also expressible.

Events, outputs, and executions. The components $o \in O$ perform actions on the system. In particular, as we have seen, components can measure each other, extend values into PCRs and the TPM can produce quotes. Additionally, an adversary on the system can corrupt and repair components with the aim of affecting the behavior of the other actions. Finally, an appraiser has the ability to inject a random nonce $n \in \mathcal{N}$ into an attestation in order to control the recency of events.

Definition 2 (Events). *Let $\mathcal{AS}$ be a target system. An event for $\mathcal{AS}$ is a node e labeled by one of the following.*

a. A measurement event *is labeled by* $\mathsf{ms}_{C^{-1}(o_2)}(o_2, o_1)$ *such that* $M(o_2, o_1)$*. We say such an event* measures o_1*, and we call* o_1 *the* target *of* e*. When* $C^{-1}(o_2)$ *is empty we omit the subscript and write* $\mathsf{ms}(o_2, o_1)$*.*
b. An extend event *is labeled by* $\mathsf{ext}(o, v, p)$*, such that* $L(o, p)$ *and* v *is a term.*
c. A quote event *is labeled by* $\mathsf{qt}(v, t_I)$*, where* v *is a term, and* $t_I = \{t.i \mid i \in I\}$ *is a sequence of PCRs belonging to the same TPM* t*. We say a quote event* reports on p*, or* is over p*, if* $p \in t_I$*.*
d. An adversary event *is labeled by either* $\mathsf{cor}(o)$ *or* $\mathsf{rep}(o)$ *for* $o \in O \setminus \{\mathsf{rtm}\}$*.*
e. The attestation start *event is labeled by* $\mathsf{att\text{-}start}(n)$*, where* n *is a term.*

The second argument to extend events and the first argument to quote events is called the input*.*

An event e touches *object* o *(or PCR* p*), iff* o *(or* p*) is an argument or subscript to the label of* e*.*

When an event e *is labeled by* ℓ *we will write* $e = \ell$*. We will often refer to the label* ℓ *as an event when no confusion will arise.*

A few observations about this definition: Measurement and extend events are constrained by the dependencies of the underlying system. So, for example, a component cannot extend a value into any PCR not allowed by the policy L. Notice that quote events have no component $o \in O$ as an argument. This is because (v)TPMs may produce quotes in response to a request by any component that has access to it. The only constraint on adversary events is that they do not affect the rtm. This is not essential, but it simplifies the statements and proofs of some theorems later on. We also do not consider the (v)TPMs as objects in O, so they are also immune from corruption. As for the $\mathsf{att\text{-}start}(n)$ event, since n is randomly chosen, extend or quote events that incorporate n must occur after $\mathsf{att\text{-}start}(n)$. We expect $\mathsf{ms}(\mathsf{rtm}, o)$ events not to occur after $\mathsf{att\text{-}start}(n)$ because they typically represent boot-time measurements of a system.

As we saw in the example from Section 3, an execution can be described as a partially ordered set (poset) of these events. We choose partially ordered sets rather than totally ordered sets because the latter unnecessarily obscure the difference between *causal* orderings and *coincidental* orderings. However, if we allow arbitrary posets of events we lose the causal structure. In particular, we need to ensure that in executions we can unambiguously identify (a) whether or not a component is corrupted at measurement and extension events, and (b) the contents of PCRs at extension and quote events. In the following, we thus impose two constraints on the posets of interest.

When no confusion arises, we often refer to a poset $(E, \prec)$ by its underlying set E and use $\prec_E$ for its order relation. Given a poset E, let $e{\downarrow} = \{e' \mid e' \prec_E e\}$, and $e{\uparrow} = \{e' \mid e \prec_E e'\}$. Given a set of events E, we let $adv(E)$, $meas(E)$, $ext(E)$, and $qt(E)$ denote respectively the set of adversary, measurement, extension, and quote events of E. For any poset $(E, \prec)$ of events over attestation system $\mathcal{AS} = (O, M, C, P, L)$, let $(E_o, \prec_o)$ denote the substructure consisting of all and only events that touch $o \in O$. Similarly we define $(E_p, \prec_p)$ for $p \in P$.

Definition 3 (Poset restrictions). *We say* $(E, \prec)$ *is* adversary-ordered *iff for every* $o \in O$, $(E_o, \prec_o)$ *has the property that if* e *and* e' *are incomparable events, then neither* e *nor* e' *are adversary events.*

We say $(E, \prec)$ *is* extend-ordered *iff for every* $p \in P$, $(E_p, \prec_p)$ *has the property that if* e *and* e' *are incomparable events, then they are both quote events.*

Adversary-ordered posets ensure that we can unambiguously define the corruption state of a component at an event that touches it. Extend-ordered posets ensure that we can unambiguously identify the contents of a PCR at events that touch it. Both these claims require justification.

Lemma 1. *Let* $(E, \prec)$ *be a finite, adversary-ordered poset for* $\mathcal{MS}$*, and let* $(E_o, \prec_o)$ *be its restriction to some* $o \in O$*. Then for any non-adversarial event* $e \in E_o$*, the set* $adv(e{\downarrow})$ *(taken in* E_o*) is either empty or has a unique maximal element.*

This lemma (proved in [14]) ensures the following conditions are well-defined.

Definition 4 (Corruption state). *Let $(E, \prec)$ be a finite, adversary-ordered poset for $\mathcal{MS}$. For each event $e \in E$ and each object o the* corruption state *of o at e, written $cs(e, o)$, is an element of $\{\bot, \mathsf{r}, \mathsf{c}\}$ and is defined as follows. $cs(e, o) = \bot$ iff $e \notin E_o$. Otherwise, we define $cs(e, o)$ inductively:*

$$cs(e,o) = \begin{cases} \mathsf{c} & : e = \mathsf{cor}(o) \\ \mathsf{r} & : e = \mathsf{rep}(o) \\ \mathsf{r} & : e \in meas(E) \wedge adv(e{\downarrow}) \cap E_o = \emptyset \\ cs(e', o) & : e \in meas(E) \wedge e' \text{ maximal in } adv(e{\downarrow}) \cap E_o \end{cases}$$

When $cs(e, o)$ takes the value c we say o is corrupt *at e; when it takes the value r we say o is* uncorrupt *or* regular *at e; and when it takes the value $\bot$ we say the corruption state is* undefined.

The above definition also allows us to define the result of a measurement event. In this work, to simplify the analysis, we assume there are no false positives or negatives as long as the measurer and its context are uncorrupted. However, we assume a corrupted measurer (or its context) can always produce evidence indicating that the target of measurement is uncorrupted.

Assumption 1 (Measurement accuracy). Let $\mathcal{G}(o)$ and $\mathcal{B}(o)$ be a partition for $\mathcal{MV}(o)$. Let $e = \mathsf{ms}(o_2, o_1)$. The *output* of e, written $out(e)$, is defined as follows.

$$out(e) = \begin{cases} v \in \mathcal{B}(o_1) & cs(e, o_1) = \mathsf{c} \text{ and } \forall o \in \{o_2\} \cup C^{-1}(o_2)\,.\ cs(e, o) = \mathsf{r} \\ v \in \mathcal{G}(o_1) & \text{otherwise} \end{cases}$$

If $out(e) \in \mathcal{B}(o_1)$ we say e *detects a corruption*. If $out(e) \in \mathcal{G}(o_1)$ but $cs(e, o_1) = \mathsf{c}$, we say the adversary *avoids detection at e*.

Assumption 1 can be used to reason in two ways. The first is to determine the result of measurement given the corruption states of the relevant components. It can also be used to infer the corruption states of some components given the corruption states of others and the result of measurement. That is, suppose we know the adversary avoids detection at $e = \mathsf{ms}_{C^{-1}(o)}(o, o_t)$. Then we can conclude that at least one member of $\{o\} \cup C^{-1}(o)$ is corrupt at e. This fact will be used in the proof of our main result.

Lemma 2. *Let $(E, \prec)$ be a finite extend-ordered poset for $\mathcal{AS}$, and let $(E_p, \prec_p)$ be its restriction to some $p \in P$. Then for every event $e \in E_p$, $ext(e{\downarrow})$ is either empty, or it has a unique maximal event e'.*

This lemma allows us to unambiguously define the value in a PCR at any event that touches the PCR.

Definition 5 (PCR value). *We define the* value *in a PCR p at event e touching p to be the following, where $e{\downarrow}$ is taken in E_p.*

$$val(e,p) = \begin{cases} \mathsf{rst} & : ext(e{\downarrow}) = \emptyset, e = \mathsf{qt}(n, t_I) \\ \#(v, \mathsf{rst}) & : ext(e{\downarrow}) = \emptyset, e = \mathsf{ext}(o, v, p) \\ state(e', p) & : e' = max(ext(e{\downarrow})), e = \mathsf{qt}(n, t_I) \\ \#(v, state(e', p)) & : e' = max(ext(e{\downarrow})), e = \mathsf{ext}(o, v, p) \end{cases}$$

When $e = \mathsf{ext}(o, v, p)$ *we say* e *is the event* recording *the value* v*.*

Lemma 2 and Definition 5 also allow us to determine the contents of a quote at a quote event. Recall that, to ensure the signature cannot be forged, we must assume the signing key is not available to the adversary.

Definition 6 (Quote outputs). *Let* $e = \mathsf{qt}(n, t_I)$*. Then its* output *is* $out(e) = [\![n, (t.i)_{i \in I}, (v_i)_{i \in I}]\!]_{sk(t)}$*, where for each* $i \in I$*,* $val(e, t.i) = v_i$*, and* $sk(t) \in \mathcal{K}$ *(the set of atomic, non-public keys). We say a quote* Q indicates a corruption *iff some* v_i *contains a* $v \in \mathcal{B}(o)$ *for some* o*.*

Finally, we formally define executions of a measurement system.

Definition 7 (Executions, specifications). *Let* $\mathcal{AS}$ *be an attestation system.*

1. *An* execution *of* $\mathcal{AS}$ *is any finite, adversary-ordered, extend-ordered poset* E *for* $\mathcal{AS}$ *such that whenever* e *has input* v*, then* v *is derivable from the set* $\mathcal{P} \cup \{out(e') \mid e' \prec_E e\}$*, i.e. the public terms together with the output of previous events.*
2. *A* specification *for* $\mathcal{AS}$ *is any execution that contains no adversary events.*

We denote by $\mathcal{E}(S)$ *the set of executions* E *that contain* S *as a substructure, and we say* S admits E*. When* S *consists only of quote events outputting a set* $\mathcal{Q}$ *of quotes, we say* E produces $\mathcal{Q}$*. We sometime abuse notation and write* $E \in \mathcal{E}(\mathcal{Q})$*.*

We thus further restrict executions to ensure that all inputs to extension and quote events are derivable at the time of the event. This reflects natural limitations on the adversary that he cannot, for example, break cryptography.

5 Bundling Evidence for Attestation

When evaluating evidence from a set of quotes $\mathcal{Q}$, the only information an appraiser has about the execution E that produced them is that $E \in \mathcal{E}(\mathcal{Q})$. According to [14], the appraiser should have a "bottom-up" specification S in mind, and she would like know whether $E \in \mathcal{E}(S)$. Thus, ideally, we could develop a strategy for bundling that would ensure $\mathcal{E}(\mathcal{Q}) \subseteq \mathcal{E}(S)$, at least for bottom-up specifications S. However, this is too much to ask for in the presence of dynamic corruptions. If the adversary completely owns the system, he can always create an $E \in \mathcal{E}(\mathcal{Q}) \setminus \mathcal{E}(S)$. The best we can do is ensure that it is difficult for the adversary to force the execution to be in $E \in \mathcal{E}(\mathcal{Q}) \setminus \mathcal{E}(S)$. In particular, we will aim to force the adversary to perform corruptions in small time windows, or to corrupt deeper (and presumably better protected) components (so-called "recent or deep" corruptions). This section develops the core set of inferences for characterizing executions in $E \in \mathcal{E}(\mathcal{Q}) \setminus \mathcal{E}(S)$, and proposes a particular strategy for bundling evidence relative to bottom-up measurements. The net result is that if an adversary would like to convince the appraiser the measurements were taken bottom-up when in fact they weren't, then he must perform recent or deep corruptions. That is, in order to avoid the hypothesis of the main result from [14] he must nonetheless subject himself to its conclusion!

5.1 Principles for TPM-based bundling

For the remainder of this section we fix an arbitrary attestation system $\mathcal{AS} = (O, M, C, P, L)$. The proofs of these lemmas can be found in the appendix. Our first lemma allows us to infer the existence of some extend events in an execution.

Lemma 3. *Let e be a quote event in execution E with output Q. For each PCR p reported on by Q, and for each v contained in $val(e, p)$ there is some extend event $e_v \prec_E e$ recording v.*

Lemma 4. *Let $e \in E$ be an event with input parameter v. If $v \in \mathcal{N}$ or if v is a signature using key $sk(t) \in \mathcal{K}$, then there is a prior event $e' \prec_E e$ such that $out(e') = v$.*

Lemma 5. *Let E be an execution producing quote Q. Assume v_i is contained before v_j in PCR p reported on by Q, and let e_i and e_j be the events recording v_i and v_j respectively. Then $e_i \prec_E e_j$.*

Proof. This is an immediate consequence of Def. 5. □

Corollary 1. *Let E be an execution producing quotes Q, and Q' where Q reports on PCR p. Suppose Q' is contained in p before v. Then every event recording values contained in Q' occurs before the event recording v.*

These results form the core of what an appraiser is justified in inferring about an execution on the basis of a TPM quote Q. Notice that the conclusions are only about extend events, and not about measurement events. This is due to one of the fundamental limitations of a TPM: Its isolation from the rest of the system causes it not to have very much contextual information about the measurement events. We are therefore very careful in what follows to identify the additional assumptions we must make about components in order to justify the inferences about measurements we would like to make.

5.2 Formalizing and justifying a bundling strategy

With these results in mind, we revisit the example of Section 3 to develop a strategy for bundling the evidence created by the measurers. In order to combine the benefits of the two strategies we considered, we are looking for a strategy that reflects the history of the events (in particular, their relative orders) while providing exclusive access for each component to its own PCR. The idea is to follow Strategy 2, but to ensure the evidence from lower layers is incorporated into the PCRs of the higher layers in a way that cannot be forged. This results in a layered, nested set of quotes of the following form.

$$
\begin{aligned}
Q_1 &= [\![\, n, p_r, \mathsf{seq}(v_1, v_2) \,]\!]_{sk(t)} \\
Q_2 &= [\![\, n, (p_1, p_2), (\mathsf{seq}(Q_1, v_3), \mathsf{seq}(Q_1, v_4)) \,]\!]_{sk(vt_1)} \\
Q_3 &= [\![\, n, p_{vc}, \mathsf{seq}(Q_2, v_5) \,]\!]_{sk(vt_2)}
\end{aligned}
$$

The quote Q_1 provides evidence that rtm has measured A_1 and A_2. This quote is itself extended into the PCRs of A_1 and A_2 before they take their measurements and extend the results. Q_2 thus represents evidence that rtm took its measurements before A_1 and A_2 took theirs. Similarly, Q_3 is evidence that vc took its measurement after A_1 and A_2 took theirs since Q_2 is extended into p_{vc} before the measurement evidence.

This quote structure is an instance of a more general strategy for bundling evidence from measurements that are taken bottom-up. The idea is that bottom-up measurements create temporal dependencies that reflect the M and C dependencies of the system. So each measurement agent o extends a quote containing measurements of $M^{-1}(o) \cup C^{-1}(o)$ before extending the evidence it gathers. This is why we assume $M \cup C$ is acyclic; this strategy would not be well-defined otherwise.

We formalize this strategy by giving a criterion for recognizing when a set of quotes conforms to the strategy. But first, we must finally formalize the as-yet intuitive notion of bottom-up measurement.

Definition 8. *A measurement event $e = \mathsf{ms}(o_2, o_1)$ in execution E is* well-supported *iff either*

i. $o_2 = \mathsf{rtm}$*, or*
ii. for every $o \in D^1(o_1)$, there is a measurement event $e' \prec_E e$ such that o is the target of e'.

When e is well-supported, we call the set of e' from Condition ii above the support *of e. An execution E* measures bottom-up *iff each measurement event $e \in E$ is well-supported.*

> **Bundling strategy criterion.** *Let $\mathcal{Q}$ be a set of quotes. We describe how to create a measurement specification $S(\mathcal{Q})$. For each $Q \in \mathcal{Q}$, and each p that Q reports on, and each $v \in \mathcal{MV}(o_2)$ contained in p, $S(\mathcal{Q})$ contains an event $e_v = \mathsf{ms}(o_1, o_2)$ where $M(o_1, o_2)$ and $L(o_1, p)$. Similarly, for each distinct n in the nonce field of some $Q \in \mathcal{Q}$, $S(\mathcal{Q})$ contains the event $\mathsf{att\text{-}start}(n)$. Let S_Q denote the set of events derived in this way from $Q \in \mathcal{Q}$. Then $e \prec_{S(\mathcal{Q})} e_v$ iff Q is contained before v and $e \in S_Q$. $\mathcal{Q}$ complies with the bundling strategy iff $S(\mathcal{Q})$ measures bottom-up.*

Using the results from the start of this section, we can prove that executions producing quotes that conform to the strategy contain a bottom-up extension structure that "shadows" the desired bottom-up measurement structure.

Definition 9. *Let $e = \mathsf{ext}(o, v, p)$ be an extend event in execution E such that $v \in \mathcal{MV}(o_t)$ for some $o_t \in O$. We say e is* well-supported *iff either*

i. $o = \mathsf{rtm}$*, or*
ii. for every $o \in D^1(o_t)$ there is an extend event $e' \prec_E e$ such that $e' = \mathsf{ext}(o', v', p')$ with $v' \in \mathcal{MV}(o)$.

We call the set of such e' the support *of e. A collection of extend events X* extends bottom-up *iff each $e \in X$ is well-supported.*

Lemma 6. *Suppose $E \in \mathcal{E}(\mathcal{Q})$ where $S(\mathcal{Q})$ measures bottom-up. Then E contains an extension substructure $X_{\mathcal{Q}}$ that extends bottom-up.*

Proof. Let $X_{\mathcal{Q}}$ be the subset of events of E guaranteed by Lemma 3. That is, $X_{\mathcal{Q}}$ consists of all the events $e = \mathsf{ext}(o, v, p)$ that record measurement values v reported in $\mathcal{Q}$. For any such event e, if $o = \mathsf{rtm}$ then e is well-supported by definition. Otherwise, since $S(\mathcal{Q})$ measures bottom-up, Lemma 3 and Corollary 1 ensure $X_{\mathcal{Q}}$ contain events $e' = \mathsf{ext}(o', v', p')$ for every $o' \in D^1(o)$ where $e' \prec_E e$. Thus e is also well supported in that case. □

Unfortunately, based on the lemmas from the start of the section, this is as far as we can go. Those lemmas do not allow us to infer the existence of *any* measurement events based only on the existence of extension events. In fact, this seems to be an important fundamental limitation of TPMs. Due to their isolation from the rest of the system, they have virtually no view into the activities of the system. Rather, we must rely on the trustworthiness of the components interacting with the TPM and knowledge of their specified behavior to infer facts about the behavior of the rest of the system.

We thus identify two assumptions about the behavior of uncorrupted measurers that will be useful in recreating the desired bottom-up measurement structure from the bottom-up extend structure.

Our first assumption is that uncorrupted measurers extend measurement values for only the most recent measurement of a given target. This translates to the following formal condition on executions.

Assumption 2. If E contains an event $e = \mathsf{ext}(o, v, p)$ with $v \in \mathcal{MV}(t)$, where o is regular at that event, then there is an event $e' = \mathsf{ms}(o, t)$ such that $e' \prec_E e$. Furthermore, the most recent such event e' satisfies $out(e') = v$.

Our second assumption is that when uncorrupted measurers extend a quote from a lower layer followed by measurement evidence it generates, it always generates the measurement evidence between those two extensions. This similarly translates to the following formal condition on executions.

Assumption 3. Suppose E has events $e \prec_E e'$ with $e = \mathsf{ext}(o, Q, p)$ and $e' = \mathsf{ext}(o, v, p)$, where Q contains evidence for $M^{-1}(o) \cup C^{-1}(o)$, and $v \in \mathcal{MV}(t)$. If o is regular at e' then there is an intervening event $e \prec_E e'' \prec_E e'$ such that $e'' = \mathsf{ms}(o, t)$.

The first assumption allows us to infer the existence of measurement events from extension events as long as the component is not corrupted. The second assumptions provides a way of inferring extra ordering information useful for reconstructing a bottom-up measurement structure.

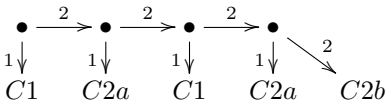
The second assumption in particular is crafted to correspond closely to our proposed strategy for bundling evidence, and so we should not expect every

architecture to satisfy these assumptions. While they may not be necessary for our purposes, we will show that they are jointly sufficient to guarantee that either the measurements were taken bottom-up, or else the adversary must have performed a recent or deep corruption relative to some component.

Theorem 4. *Let $E \in \mathcal{E}(\mathcal{Q})$ where $S(\mathcal{Q})$ measures bottom-up, and suppose it satisfies Assumptions 2 and 3. Suppose that $v_t \in \mathcal{G}(o_t)$ for each measurement value v_t contained in $\mathcal{Q}$. Then for each extension event e recording a measurement value, either*

1. *e reflects a measurement event that is well-supported by measurement events reflected by the support of e.*
2. *a. some $o_2 \in D^2(o_t)$ gets corrupted in E, or*
   b. some $o_1 \in D^1(o_t)$ gets corrupted in E after being measured.

Proof. First note that we can immediately apply Lemma 6 to infer that the extension events represented by $\mathcal{Q}$ form a bottom-up extension structure. The rest of the proof considers an exhaustive list of cases, demonstrating that each one falls into one of Conditions 1, 2a, or 2b. The following diagram summarizes the proof by representing the branching case structure and indicating which clause of the conclusion (C1, C2a, or C2b) each case satisfies.



Consider any extend event $e = \mathsf{ext}(o_1, v_t, p_1)$ of X extending a measurement value for some $o_t \in O$. The first case distinction is whether or not $o_1 = \mathsf{rtm}$.

Case 1: Assume $o_1 = \mathsf{rtm}$. Since rtm cannot be corrupted, it is regular at e, and by Assumption 2, e reflects the measurement event $\mathsf{ms}(\mathsf{rtm}, o_t)$ which is trivially well-supported, so Condition 1 is satisfied.

Case 2: Assume $o_1 \neq \mathsf{rtm}$. Since X extends bottom-up, it has events $e_i = \mathsf{ext}(o_2^i, v_2^i, p_2^i)$ extending measurement values v_2^i for every $o^i \in D^1(o_t)$, and for each i, $e_i \prec_E e$. Furthermore, by Corollary 1, there is an extend event $e_q = \mathsf{ext}(o_1, Q, p)$ with $e_i \prec_E e_q \prec_E e$ where Q is a quote containing the values recorded at each e_i. Now either some o_2^i is corrupt at e_i (Case 2.1), or each o_2^i is regular at e_i (Case 2.2).

Case 2.1: Assume some o_2^i is corrupt at e_i. Then there must have been a prior corruption of $o_2^i \in D^2(o_t)$, and hence we are in Condition 2a.

Case 2.2: Assume each o_2^i is regular at e_i. Then Assumption 2 applies to each e_i, so each one reflects a measurement event e_i'. In this setting, either o_1 is regular at e (Case 2.2.1), or o_1 is corrupt at e (Case 2.2.2).

Case 2.2.1: Assume o_1 is regular at e. Then since the events e_q and e satisfy the hypothesis of Assumption 3, we can conclude that e reflects a measurement event $e' = \mathsf{ms}(o_1, o_t)$ such that $e_q \prec_E e' \prec_E e$. Thus, e' is well-supported by the e_i' events which are reflected by the support of e, putting us in Condition 1.

Case 2.2.2: Assume o_1 is corrupt at e. Since $o_1 \in D^1(o_t)$ one of the e'_i is a measurement event of o_1 with output $v_1 \in \mathcal{G}(o_1)$ since X only extends measurement values that do not indicate corruption. Call this event e'_*. The final case distinction is whether o_1 is corrupt at this event e'_* (Case 2.2.2.1) or regular at e'_* (Case 2.2.2.2).

Case 2.2.2.1: Assume o_1 is corrupt at e'_*. Since the measurement outputs a good value, some element $o_2 \in D^1(o_1) \subseteq D^2(o_t)$ is corrupt at e'_*. This satisfies Condition 2a.

Case 2.2.2.2: Assume o_1 is regular at e'_*. By the assumption of Case 2.2.2, o_1 is corrupt at e with $e'_* \prec_E e$. Thus there must be an intervening corruption of o_1. Since e'_* is a measurement event of o_1, this satisfies Condition 2b. □

Theorem 4 guarantees that if there are no recent or deep corruptions, then we can infer the existence of a collection of measurement events reflected by the values in the quotes. It remains to show that this measurement substructure is precisely the one we want, namely that it is equal to $S(\mathcal{Q})$. Unfortunately, this may not be the case. $S(\mathcal{Q})$ may contain orderings that are not strictly necessary to ensure $S(\mathcal{Q})$ measures bottom-up. However, Assumption 3 can guarantee only that the orderings necessary to be bottom-up are present. For this reason we introduce the notion of the core of a bottom-up specification. The *core* of a bottom-up specification S is the result of removing any orderings between measurement events $e_i \prec_S e_j$ whenever e_i is not in the support of e_j. That is, the core of S ignores all orderings that do not contribute to S measuring bottom-up. We can then show that the measurement structure inferred from Theorem 4 is (isomorphic to) the core of $S(\mathcal{Q})$.

Theorem 5. *Let $E \in \mathcal{E}(\mathcal{Q})$ such that $S(\mathcal{Q})$ measures bottom-up, and let S' be its core. Suppose that $\mathcal{Q}$ detects no corruptions, and that E satisfies Assumptions 2 and 3. Then one of the following holds:*

1. *$E \in \mathcal{E}(S')$,*
2. *there is some $o_t \in O$ such that*
   a. *some $o_2 \in D^2(o_t)$ is corrupted, or*
   b. *some $o_1 \in D^1(o_t)$ is corrupted after being measured.*

6 Related Work

There has been much research into measurement and attestation. While a complete survey is infeasible for this paper, we mention the most relevant highlights in order to describe how the present work fits into the larger body of work.

Building on the early work of Trusted Boot [9], there have been numerous attempts to bring trust further up the software stack. Most notably, Sailer et al. [15] introduced an integrity measurement architecture (IMA) in which each application is measured as it is launched. More recently, this body of work on static measurement has been augmented with attempts to measure dynamic system properties that give a clearer picture of the current state of the system (e.g.

[8, 7, 6, 17]). Most of these focus on the low-level details of what to measure and how to implement it without considering how runtime corruption can affect the attestation process itself. In particular, it is common to invoke the use of Trusted Boot and IMA as a way to build a chain of trust from the hardware which the proposed measurement agent can extend. Our work could be applied to systems that incorporate these integrity measurers in order to better understand how they respond to dynamic corruption of the trusted computing base and measurement agents themselves.

We are not the first to discuss the dependencies that emerge in a layered system. Some work [10, 16] builds on the notion of a tree of trust [11] to tease out a structure for the integrity evidence required of an attestation. The focus in these papers is on ensuring the integrity of the system can be correctly inferred from the structure of the evidence. While we focus on only a subset of the trust dependencies considered in, say, [10], they do not take full account of the effects dynamic corruption of components might have on the bundling of the evidence. Rather they explicitly bracket out the problem of guaranteeing the trustworthiness of the integrity information itself. An interesting line of future work would be to investigate the causal effects of dynamic corruption on the wider variety of dependencies they consider.

Layered dependencies are also implicit in the design of many systems intended to support attestation of their runtime properties. Coker et al. [4] present 5 principles for remote attestation and propose a layered system designed from those principles. They do not investigate the low-level structure of evidence that must be created in order to attest to the layered dependencies or how to bundle such evidence using the TPM. Cabuk et al. [3] present a hierarchical system with a software-based root of trust for measurement that is connected to a lower-level chain of trust rooted in hardware. They demonstrate the variety of hierarchical dependencies that can naturally arise and propose ways to manage this complexity. Finally, in [2], Berger et al. propose a way to manage the complexity of appraising systems with layered dependencies as the systems scale. In all of these examples, to the extent that runtime corruptions are considered seriously, the problem of understanding how such corruptions break the chain of trust is not examined. Within our formalism we should be able to represent all these systems and characterize the ways in which runtime corruptions can occur without being reflected in the final bundle of evidence. Particular designs may enable bundling strategies that are tailored to the design which require weaker assumptions than those we used in this paper.

7 Conclusion

In this paper we have developed a formalism for reasoning about layered attestations. Within the framework we have identified some potential pitfalls when using a TPM to bundle measurement evidence. These pitfalls arise due to a fundamental limitation of TPMs. Namely, by virtue of being isolated from the main system, TPMs have very limited contextual information about the events

occurring on that system. This means further assumptions must be made about uncompromised components in order for an appraiser to infer desired behavior.

We also identified a core set of inference principles that can help system designers determine the consequences of a particular strategy for bundling evidence. Finally, we applied those principles to prove the robustness of a new layered approach to bundling evidence. We believe this new proposal gives easy to explain design advice. Namely, after identifying the temporal dependencies required for an attestation, the evidence should be extended into a TPM one layer at a time, ensuring the quotes from lower layers are incorporated into the quotes from higher layers as you go. This will remain robust as long as uncorrupted components can be trusted to take fresh measurements after receiving the results from below.

Although this proposal is most applicable to systems designed around the use of vTPMs, we believe the core idea illuminates the problems with certain naive ways of using a TPM to report evidence. In any case, we make no claims that this proposal represents a complete solution for all cases. Rather, we consider it the first attempt to seriously account for the possibility of runtime corruption during an attestation, and we would encourage others to develop complementary strategies. The formalism introduced here together with the inference principles would be a good way to evaluate such proposals.

Acknowledgments

I would like to thank Pete Loscocco for suggesting and guiding the direction of this research. Many thanks also to Perry Alexander and Joshua Guttman for their valuable feedback on earlier versions of this work. Thanks also to Sarah Helble and Aaron Pendergrass for lively discussions about measurement and attestation systems. Finally, thank you to the anonymous reviewers for helpful comments in improving the paper.

References

1. Stefan Berger, Ramón Cáceres, Kenneth A. Goldman, Ronald Perez, Reiner Sailer, and Leendert van Doorn. vTPM: Virtualizing the trusted platform module. In *Proceedings of the 15th USENIX Security Symposium, Vancouver, BC, Canada, July 31 - August 4, 2006*, 2006.
2. Stefan Berger, Kenneth A. Goldman, Dimitrios E. Pendarakis, David Safford, Enriquillo Valdez, and Mimi Zohar. Scalable attestation: A step toward secure and trusted clouds. *IEEE Cloud Computing*, 2(5):10–18, 2015.
3. Serdar Cabuk, Liqun Chen, David Plaquin, and Mark Ryan. Trusted integrity measurement and reporting for virtualized platforms. In *Trusted Systems, First International Conference, INTRUST 2009, Beijing, China, December 17-19, 2009. Revised Selected Papers*, pages 180–196, 2009.
4. George Coker, Joshua D. Guttman, Peter Loscocco, Amy L. Herzog, Jonathan K. Millen, Brian O'Hanlon, John D. Ramsdell, Ariel Segall, Justin Sheehy, and Brian T. Sniffen. Principles of remote attestation. *Int. J. Inf. Sec.*, 10(2):63–81, 2011.

5. Jordi Cucurull and Sandra Guasch. Virtual TPM for a secure cloud: Fallacy or reality? *Universidad de Alicante*, 2014.
6. Lucas Davi, Ahmad-Reza Sadeghi, and Marcel Winandy. Dynamic integrity measurement and attestation: towards defense against return-oriented programming attacks. In *Proceedings of the 4th ACM Workshop on Scalable Trusted Computing, STC 2009, Chicago, Illinois, USA, November 13, 2009*, pages 49–54, 2009.
7. Chongkyung Kil, Emre Can Sezer, Ahmed M. Azab, Peng Ning, and Xiaolan Zhang. Remote attestation to dynamic system properties: Towards providing complete system integrity evidence. In *Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2009, Estoril, Lisbon, Portugal, June 29 - July 2, 2009*, pages 115–124, 2009.
8. Peter Loscocco, Perry W. Wilson, J. Aaron Pendergrass, and C. Durward McDonell. Linux kernel integrity measurement using contextual inspection. In *Proceedings of the 2nd ACM Workshop on Scalable Trusted Computing, STC 2007, Alexandria, VA, USA, November 2, 2007*, pages 21–29, 2007.
9. Richard Maliszewski, Ning Sun, Shane Wang, Jimmy Wei, and Ren Qiaowei. Trusted boot (tboot). http://sourceforge.net/p/tboot/wiki/Home/.
10. Cornelius Namiluko and Andrew Martin. Provenance-based model for verifying trust-properties. In *Trust and Trustworthy Computing - 5th International Conference, TRUST 2012, Vienna, Austria, June 13-15, 2012. Proceedings*, pages 255–272, 2012.
11. Stéphane Lo Presti. A tree of trust rooted in extended trusted computing. In *Proceedings of the Second Conference on Advances in Computer Security and Forensics Programme (ACSF)*, pages 13–20, 2007.
12. Xen Project. http://www.xenproject.org.
13. QEMU. http://wiki.qemu.org.
14. Paul D. Rowe. Confining adversary actions via measurement. In *Graphical Models for Security - Third International Workshop, GraMSec 2016*, In press.
15. Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and implementation of a tcg-based integrity measurement architecture. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 223–238, 2004.
16. Andreas U. Schmidt, Andreas Leicher, Andreas Brett, Yogendra Shah, and Inhyok Cha. Tree-formed verification data for trusted platforms. *Computers & Security*, 32:19–35, 2013.
17. Jinpeng Wei, Calton Pu, Carlos V. Rozas, Anand Rajan, and Feng Zhu. Modeling the runtime integrity of cloud servers: A scoped invariant perspective. In *Cloud Computing, Second International Conference, CloudCom 2010, November 30 - December 3, 2010, Indianapolis, Indiana, USA, Proceedings*, pages 651–658, 2010.

A Proof of Lemmas

The following is a proof of Lemma 3

Proof. By definition, the values contained in a PCR are exactly those that were previously extended into it. Thus, since ext events are the only way to extend values into PCRs, there must be some event $e_v = \mathsf{ext}(o, v, p)$ with $e_v \prec_E e$. □

The following is a proof of Lemma 4

Proof. Definition 7 requires v to be derivable from the public terms $\mathcal{P}$ and the output of previous messages. Call those outputs $\mathcal{O}$.

First suppose $v \in \mathcal{N}$. Since v is atomic, the only way to derive it is if $v \in \mathcal{P} \cup \mathcal{O}$. Since $\mathcal{P} \cap \mathcal{N} = \emptyset$, $v \notin \mathcal{P}$, hence $v \in \mathcal{O}$ as required.

Now suppose v is a signature using key $sk(t) \in \mathcal{K}$. Then v can be derived in two ways. The first is if $v \in \mathcal{P} \cup \mathcal{O}$. In this case, since $v \notin \mathcal{P}$ it must be in $\mathcal{O}$ instead as required. The other way to derive v is to construct it from the key $sk(t)$ and the signed message, say m. That is, we must first derive $sk(t)$. Arguing as above, the only way to derive $sk(t)$ is to find it in $\mathcal{O}$, but there are no events that output such a term. □

We conclude with the complete proof of Theorem 5.

Proof. By Lemma 6, E contains a substructure $X_{\mathcal{Q}}$ of extend events that extends bottom-up. Thus by Theorem 4, Conditions 2a and 2b are possibilities. So suppose instead that E satisfies Condition 1 of Theorem 4. We must show that $E \in \mathcal{E}(S')$. In particular, we construct $\alpha : S' \to E$ and show that it is label- and order-preserving.

Consider the measurement events e_i^s of S'. By construction, each one comes from some measurement value v_i contained in $\mathcal{Q}$. Similarly, the well-supported measurement events e_i^m of E guaranteed by Theorem 4 are reflected by extend events e_i of E which are, in turn, those events that record each v_i in $\mathcal{Q}$. We need to show that $e_i^s = e_i^m$ for each i (i.e. that the labels agree), and that the orders among the e_i^s are reflected by corresponding orders among the e_i^m.

Consider first the label of e_i^s. It corresponds to a measurement value v_i contained in some p_i of $\mathcal{Q}$. So e_i^s is labeled $\mathsf{ms}(o, o')$ where $M(o, o')$, $v_i \in \mathcal{MV}(o')$, and $L(o, p_i)$. The event e_i^m also corresponds to the same v_i. Lemma 3 ensures that $e_i = \mathsf{ext}(o, v, p_i)$ with $L(o, p_i)$, and so the measurement event it reflects is $e_i^m = \mathsf{ms}(o, o')$ with $M(o, o')$ and $v_i \in \mathcal{MV}(o')$. Thus e_i^s and e_i^m have the same label.

We now show that if $e_i^s \prec_{S(\mathcal{Q})} e_j^s$ then $e_i^m \prec_E e_j^m$. The former ordering exists in S' because some quote $Q \in \mathcal{Q}$ is contained in p_j before v_j and v_i is contained in Q, and because e_i^s is in the support of e_j^s. By Corollary 1 $e_i \prec_E e_j$ and e_i is in the support of e_j and therefore Theorem 4 ensures that the measurements they reflect are also ordered, i.e. $e_i^m \prec_E e_j^m$.

Finally, consider any events $e = \mathsf{att\text{-}start}(n)$ in S'. They come from nonces n found as inputs to quotes $Q \in \mathcal{Q}$. By Lemma 4, E also has a corresponding event e^* with $out(e^*) = n$. Since att-start events are the only ones with output of the right kind, $e^* = \mathsf{att\text{-}start}(n)$ as well. Thus we can extend α by mapping each such e to the corresponding e^*. The rules for $S(\mathcal{Q})$ say that $e \prec_{S(\mathcal{Q})} e'$ only when Q has n in the nonce field, and Q occurs before the value recorded by e'. In E, e^* precedes the event producing Q (by Lemma 4) which in turn precedes e' by Lemmas 4 and 5. Thus the orderings in $S(\mathcal{Q})$ involving att-start events are also reflected in E. □