Evidence Tampering and Chain of Custody in Layered Attestations

Ian D. Kretz
ian.kretz@datadoghq.com
Datadog
France

Clare C. Parran
ccparran@mitre.org
The MITRE Corporation
USA

John D. Ramsdell
ramsdell@mitre.org
The MITRE Corporation
USA

Paul D. Rowe
paul.rowe@twosixtech.com
Two Six Technologies
USA

ABSTRACT

In distributed systems, trust decisions are made on the basis of integrity evidence generated via remote attestation. Examples of the kinds of evidence that might be collected are boot time image hash values; fingerprints of initialization files for userspace applications; and a comprehensive measurement of a running kernel. In layered attestations, evidence is typically composed of measurements of key subcomponents taken from different trust boundaries within a target system. Discrete measurement evidence is bundled together for appraisal by the components that collectively perform the attestation.

In this paper, we initiate the study of evidence chain of custody for remote attestation. Using the Copland attestation specification language, we formally define the conditions under which a runtime adversary active on the target system can tamper with measurement evidence. We present algorithms for identifying all such tampering opportunities for given evidence as well as tampering "strategies" by which an adversary can modify incriminating evidence without being detected. We then define a procedure for transforming a Copland-specified attestation into a maximally tamper-resistant version of itself. Our efforts are intended to help attestation protocol designers ensure their protocols reduce evidence tampering opportunities to the smallest, most trustworthy set of components possible.

*Kretz and Rowe were affiliated with MITRE when this work was done.

PPDP 2024, September 09–11, 2024, Milano, Italy

ACM ISBN 979-8-4007-0969-2/24/09.
https://doi.org/10.1145/3678232.3678244

KEYWORDS

Remote attestation, layered attestation, attestation protocols, digital evidence, evidence tampering, chain of custody

ACM Reference Format:
Ian D. Kretz, Clare C. Parran, John D. Ramsdell, and Paul D. Rowe. 2024. Evidence Tampering and Chain of Custody in Layered Attestations. In *26th International Symposium on Principles and Practice of Declarative Programming (PPDP 2024), September 09–11, 2024, Milano, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3678232.3678244

1 INTRODUCTION

In criminal investigations, chain of custody refers to evidence about the provenance and integrity of evidence relevant to a case. It describes where, when and by whom the evidence was collected, where it was stored and who had access to it, and whether and how it was modified from its original state. Chain of custody is crucial because it serves to convince judges and jurors that the evidence presented in court can be traced directly back to its collection and is therefore reflective of what really happened. The chain of custody also limits and documents who could be responsible for any discrepancy between the evidence collected and the evidence presented.

Remote attestation shares similarities with a criminal investigation. In a remote attestation, evidence of a target's integrity is collected by measuring various subcomponents of the target. The evidence can be collected in a variety of ways. Some examples are:

- the hashes of images used to boot an operating system can be digested into a report;
- a fingerprint of the initialization files of a critical userspace application can make a measurement; and
- an extensive analysis of a running kernel can be reduced to an informative measurement using a system such as LKIM [9].

This evidence is bundled together and presented to an appraiser who evaluates the evidence to infer whether the target has sufficient integrity. Thus, chain of custody for measurement evidence is also crucial for remote attestation. It should convince the appraiser that the evidence it receives can be traced directly back to the act of measurement itself. Any discrepancies can be attributed to whoever had access to the evidence along the chain of custody. The consequence of blindly trusting a component without sufficient integrity guarantees can have limitless repercussions since it would give an adversary a critical foothold on the system.

In this paper, we initiate the study of chain of custody for remote attestations. We focus on the following questions which are important for an appraiser to ask.

> Which components had an opportunity to tamper with a piece of evidence and remove signs of corruption?
>
> Could an adversary have combined these opportunities to eliminate all copies of a piece of incriminating evidence prior to appraisal?
>
> Can tampering opportunities be eliminated by enhancing an attestation's design?

Our primary goal is to help designers of remote attestation protocols ensure their protocols limit these evidence tampering opportunities as much as possible.

An important aspect of our approach is Copland [17], a language for specifying layered attestation protocols. Layered attestations take advantage of the hierarchical nature of many system architectures where measurements may be taken from a variety of *places* that may have different inherent levels of protection and different dependencies. Copland is equipped with formal semantics that allows for rigorous formal analyses of trust properties [19] while remaining connected to concrete implementations [11, 12, 14].

Copland allows us to introduce and formalize three problems corresponding to these questions.

Tamper Opportunities Problem (TOP): Given a Copland phrase and a measurement event v in its semantics, find all tampering opportunities for the measurement evidence produced at v.

Tamper Strategies Problem (TSP): Given a Copland phrase and a measurement event v in its semantics, find all adversarial "strategies" for tampering that consistently modify every copy of v that the appraiser sees.

Evidence-Protecting Program Problem (EPPP): Given a Copland phrase, produce another that preserves the given phrase's semantics while maximally constraining the set of tampering opportunities available to an adversary.

The major contributions of this paper are:

(1) Initiating the study of chain of custody for layered attestations
(2) Formalizing the TOP and TSP and introducing algorithms for computing solutions to them
(3) Formalizing the EPPP and introducing an idempotent function that maps each Copland phrase to a maximally tamper-proof iteration of itself

All of the paper's numbered definitions and theorems (including lemmas and corollaries) have been formalized and verified in the Coq proof assistant [2]. The proof scripts along with instructions for checking them and building their documentation are available in a public repository [1]. The algorithms for solving the TOP, TSP and EPPP have not been formalized nor formally proven to solve these problems, but we provide informal arguments to this effect herein.

C	$\longleftarrow$	$*P : T$	Start T at P
T	$\longleftarrow$	$S\ P\ S$	Measurement (Probe Place Target)
	$\mid$	$@_P\ [\ T\ ]$	At place
	$\mid$	$\_$	Copy
	$\mid$	$!$	Sign
	$\mid$	$\#$	Hash
	$\mid$	$(T\ \rightarrow\ T)$	Linear sequence
	$\mid$	$(T\ \ D{<}D\ \ T)$	Sequential branching
	$\mid$	$(T\ \ D{\sim}D\ \ T)$	Parallel branching
D	$\longleftarrow$	$-\mid+$	Splitting specification

Figure 1: Copland Syntax

The remainder of the paper is structured as follows. Section 2 introduces the syntax and semantics of the Copland language. Section 3 motivates the study of chain of custody for layered attestations. Section 4 formalizes the TOP and TSP and defines high-level algorithms for solving them. In Section 5 we introduce the notion of a *protected* data flow graph and prove that such graphs minimize tamper opportunities available to an adversary. The Copland Evidence Protection Program is defined in Section 6 and proven to solve the EPPP for any input phrase. We discuss how our work relates to other efforts from the literature in Section 7 before concluding. Supporting definitions for the Copland Data Flow Semantics can be found in Appendix A.

2 COPLAND SYNTAX AND SEMANTICS

Copland is a domain-specific language for specifying layered attestations. The semantics of Copland is fundamental to our understanding of what it means for an adversary to be able to tamper with evidence during an attestation. This section presents the syntax and semantics of Copland together with a small set of notational conventions. Ambitious readers can proceed straight to the motivating examples of Section 3, returning here as needed for technical reference.

The syntax of Copland is presented in Figure 1. The syntactic category P specifies places at which Copland actions take place, the category S specifies symbols that identify measurements or targets of measurements, C is the start symbol and the $\rightarrow$ operator is right-associative.

The data flow semantics of a Copland phrase is given as a *data flow graph*, a labeled, directed acyclic graph with distinguished input and output nodes. The nodes represent events such as measuring a component, signing evidence, or making a remote request; the edges encode data and control flow.

Definition 1 (Data Flow Graph). *A* data flow graph *is a tuple* $(V, v^i, v^o, \rightarrow, \ell)$ *consisting of*

(1) a set V of events,
(2) $v^i \in V$, the input event,
(3) $v^o \in V$, the output event,
(4) relation $\rightarrow \subseteq V \times V$, flow edges, and
(5) function $\ell : V \rightarrow L$, a map from events to labels.

The graph $(V, \rightarrow)$ must be acyclic, and there must be no flow edge to v^i or from v^o.

$$
\begin{array}{lcll}
E & \leftarrow & \xi & \text{Empty evidence} \\
 & \mid & \mathsf{m}(\mathsf{ms}(S,P,S),P,\Phi,E) & \text{Measurement} \\
 & \mid & !\{E\}_P & \text{Signature} \\
 & \mid & \#\{E\}_P & \text{Hash} \\
 & \mid & (E\ ;;\ E) & \text{Sequential composition} \\
 & \mid & (E \parallel E) & \text{Parallel composition} \\
\Phi & \leftarrow & \text{sequence of positive integers} &
\end{array}
$$

Figure 2: Evidence Syntax

$$
\begin{array}{lcl}
L & \leftarrow & P : \mathsf{msp}(E) \mid P : \mathsf{cpy}(E) \mid P : \mathsf{sig}(E) \\
 & \mid & P : \mathsf{hsh}(E) \mid P : \mathsf{req}(P,E) \mid P : \mathsf{rpy}(P,E) \\
 & \mid & P : \mathsf{split}(D,D,E) \mid P : \mathsf{join}(O,E) \\
O & \leftarrow & < \mid \sim
\end{array}
$$

Figure 3: Label Syntax

For $D = (V, v^i, v^o, \to, \ell)$ *we define* $D^V = V$, $\bot(D) = v^i$, *and* $\top(D) = v^o$.

Isolating the data flow portion of the semantics allows us to reason about how evidence propagates from component to component as an attestation executes. Such analyses form the basis of our ability to identify tampering opportunities.

Figure 5 in the appendix defines the data flow semantics inductively on the structure of Copland phrases. In this section, we seek to illustrate how the semantics uses this structure to build up a data flow graph. We rely on diagrams of the form

$$\xrightarrow{e} \boxed{\text{label}} \xrightarrow{e'}$$

to describe how executing a Copland phrase c at place p transforms and transmits evidence. Boxes represent labeled events and arrows depict the flow of evidence and the precedence ordering of events. The syntax of evidence is given in Figure 2. The syntax of event labels is given in Figure 3. Labels identify what kind of action (e.g., `msp` for measurement, `sig` for signature, `req` for request) an event represents, the place where the action occurs, the evidence produced at the event, and other action-specific data. These diagrams and the data flow semantics also refer to the Copland evidence semantics $\mathcal{E}$ defined in Figure 4. This semantics computes the form of the evidence that is produced by executing a Copland phrase at a place.

The most basic Copland phrase is a measurement $m\ q\ t$: symbols m and t name the probe and the target of measurement, respectively, and q is the place where t resides. When at place p, $m\ q\ t$ means that p should receive some evidence e, perform m targeting t at q, and then emit the resulting evidence. The structure of evidence produced by the measurement $m\ q\ t$ is $\mathsf{m}(p, m, q, t, v_\phi, e)$. When looking at the abstract syntax of a Copland phrase, ϕ provides a path from the root to the position of $m\ q\ t$ in the phrase, v_ϕ is a variable that holds the raw evidence value produced by the measurement, and e is the evidence term passed as input to the measurement event. The path is used to produce distinguished evidence when two occurrences of a measurement request in a phrase have the same measurement, target and place.

$$C(*p : t) = \mathcal{E}(t, p, \langle\rangle, \xi)$$

$$
\begin{aligned}
\mathcal{E}(m\ q\ t, p, \phi, e) &= \mathsf{m}(p, m, q, t, v_\phi, e) \\
\mathcal{E}(\{\}, p, \phi, e) &= \xi \\
\mathcal{E}(\_, p, \phi, e) &= e \\
\mathcal{E}(!, p, \phi, e) &= !\{e\}_p \\
\mathcal{E}(\#, p, \phi, e) &= \#\{e\}_p \\
\mathcal{E}(@_q\ [t], p, \phi, e) &= \mathcal{E}(t, q, 1 :: \phi, e) \\
\mathcal{E}(t_1 \to t_2, p, \phi, e) &= \mathcal{E}(t_2, p, 2 :: \phi, \mathcal{E}(t_1, p, 1 :: \phi, e)) \\
\mathcal{E}(t_1\ l{<}r\ t_2, p, \phi, e) &= \mathcal{E}(t_1, p, 1 :: \phi, \mathcal{F}(l, e))\ ;;\ \mathcal{E}(t_2, p, 2 :: \phi, \mathcal{F}(r, e)) \\
\mathcal{E}(t_1\ l{\sim}r\ t_2, p, \phi, e) &= \mathcal{E}(t_1, p, 1 :: \phi, \mathcal{F}(l, e)) \parallel \mathcal{E}(t_2, p, 2 :: \phi, \mathcal{F}(r, e))
\end{aligned}
$$

$$\mathcal{F}(d, e) = \begin{cases} \xi & \text{if } d = - \\ e & \text{if } d = + \end{cases}$$

Figure 4: Evidence Semantics

$$\xrightarrow{e} \boxed{p : \mathsf{msp}(\mathsf{m}(p, m, q, t, v_\phi, e))} \xrightarrow{\mathsf{m}(p, m, q, t, v_\phi, e)}$$

The semantics of the phrases copy $\_$, sign !, and hash # have the same form as a measurement. When executing at place p, their corresponding event labels are $p : \mathsf{cpy}(e)$, $p : \mathsf{sig}(e)$, and $p : \mathsf{hsh}(e)$. For input evidence e, the evidence sent along the outgoing arrow is e, $!\{e\}_p$, and $\#\{e\}_p$, respectively.

The semantics of the remainder of the Copland syntax is defined inductively. Let $p : \{c\}$ be the events and their labels and orderings associated with executing phrase c at p, and let $\mathcal{E}(c, p, \phi, e)$ be the evidence that results from the execution when e is provided as input evidence and when c is located at position ϕ inside a Copland phrase. Measurements can be combined in a pipeline fashion using the $\to$ operator. Thus, when at p, $c_1 \to c_2$ means

$$\xrightarrow{e} p : \{c_1\} \xrightarrow{\mathcal{E}(c_1, p, 1 :: \phi, e)} p : \{c_2\} \xrightarrow{\mathcal{E}(c_2, p, 2 :: \phi, \mathcal{E}(c_1, p, 1 :: \phi, e))}$$

A measurement can be taken at a remote location using the @ operator. When at p, $@_q\ [c]$ means

$$\xrightarrow{e} \boxed{p : \mathsf{req}(q, e)} \xrightarrow{e} q : \{c\} \xrightarrow{\mathcal{E}(c, q, 1 :: \phi, e)} \boxed{q : \mathsf{rpy}(p, e)} \xrightarrow{\mathcal{E}(c, q, 1 :: \phi, e)}$$

Phrases c_1 and c_2 can be combined using branching. There are two ways of combining phrases using branching, sequentially ($<$) and in parallel ($\sim$). A Copland phrase specifies whether or not evidence is sent along each branch. Syntactically, this is done by writing $d_1{<}d_2$ or $d_1{\sim}d_2$ with $d_i \in \{+, -\}$ on either side of the branching operator. If $d_i = +$, e is sent as input evidence along the corresponding branch. If $d_i = -$, no evidence is sent along that branch. We show the pattern for $c_1 +{<}- c_2$ executing at p.

$$
\begin{tikzcd}[row sep=small]
{} & {} & p : \{c_1\} \arrow[dr, "e_1"] & {} & {} \\
{} \arrow[r, "e"] & \boxed{p : \mathsf{split}(+, -, e)} \arrow[ur, "e"] & {} & \boxed{p : \mathsf{join}(<, e_1 ;; e_2)} \arrow[r, "e_1 ;; e_2"] & {} \\
{} & {} & p : \{c_2\} \arrow[ur, "e_2"'] & {} & {}
\end{tikzcd}
$$

In the above, $e_i = \mathcal{E}(c_i, p, i :: \phi, d_i(e))$. Since no evidence is passed along the branch where $d_i = -$, there is no data flow there and hence no arrow. In the case of $c_1 -{<}- c_2$, there is no data flow from any prior events to the events of c_1 or c_2.

The pattern for $c_1\ d_1{\sim}d_2\ c_2$ is identical except for the label of the last event and the resulting evidence. In this case, the last event has label $p : \mathsf{join}(\sim, e_1 \parallel e_2)$ and the resulting evidence is $e_1 \parallel e_2$. The sequential and branching operators differ in terms of their control flow semantics (hence the inclusion of both in the language): the former ensures the left phrase completes before the right one begins. However, the data flow semantics of the two branching operators are identical.

The following definition introduces notation used to define the data flow semantics in the appendix.

Definition 2 (Data Flow Semantics). *The* data flow semantics *of a Copland phrase* $*p : t$ *is denoted* $\bar{C}(*p : t)$. *It relies on an auxiliary function* $\bar{\mathcal{D}}(t, p, \phi, e)$ *which takes a Copland phrase* t, *a place* p, *a position* ϕ, *and evidence* e *and produces a data flow graph. The detailed inductive definition is shown in Figure 5 in the appendix.*

Evidence and chain of custody. The structure of evidence (see Fig. 2) contains "metadata" that records information about how the evidence was assembled from various pieces. For example, the fact that two pieces of evidence e_1 and e_2 were collected in sequence is notated as $e_1 ;; e_2$. If they were generated concurrently, the resulting evidence would be $e_1 \parallel e_2$. Similarly, the structure of a measurement $\mathsf{m}(p, m, q, t, v_\phi, e)$ consists mostly of metadata specifying who the measurer and the target are. All this metadata can be pre-computed based on the structure of the Copland phrase being executed. These pre-computed portions act as the documentation of the chain of custody of the evidence. The fact that it can be pre-computed means that any attempt to alter the structure of evidence will be detected by the appraiser, who is expecting it to look a certain way.

The only part of evidence that cannot be pre-computed is the value taken by the variable v_ϕ in a measurement. It is the variable v_ϕ that holds the actual result of measurement that should reflect the current state of the target component. Any incriminating evidence will be found in these v_ϕ variables, hence those are the parts of the evidence structure that an adversary may try to tamper with.

Notation and Terminology. Every event occurs at the place found to the left of the colon in its label. Since communication events (i.e. request or reply events) represent the coordinated transmission and reception of data between places, we identify a sending place at which these events occur as well as a "receiving" place that receives the evidence transmitted by the sending place. Although non-communication events do not have a receiving place, from a proof standpoint it is convenient to simply define their receiving places to be their sending places. The place(s) associated with an event are obtained by place projection functions.

Definition 3 (Place Projections). *Let function* ϱ *be the function that extracts the place out of an event's label, the place before the colon.*

For an event v, *define*

$$\varrho'(v) = \begin{cases} q & \ell(v) = p : \mathsf{req}(q, e) \text{ or } \ell(v) = p : \mathsf{rpy}(q, e), \\ \varrho(v) & \text{otherwise.} \end{cases}$$

For a communication event v, *we call* $\varrho(v)$ *the* sending place *and* $\varrho'(v)$ *the* receiving place *of* v.

As a small technical note, Copland syntax does not require that all requests go to a different place. That is, it is possible for the sending place and the receiving place of a communication event to be the same.

Definition 4 (Cross-place communication). *Event* v *is a cross-place communication event iff* $\varrho(v) \neq \varrho'(v)$.

Finally, we often have the need to refer to the evidence emitted at an event. The syntax and semantics interact to ensure that this evidence is always the last argument in the label of an event. We can thus easily project out this information.

Definition 5 (Evidence Projection). *Let function* ε *be the function that projects the last argument out of an event's label. This is always the outgoing evidence from that event.*

We end this section with a small lemma, a simple sanity check to ensure the consistency of the data flow and evidence semantics. The lemma says that the evidence of the output event of the data flow semantics is the same as the evidence computed by the evidence semantics on the same inputs.

Lemma 1 (Flow Graph Top). *If* $\bar{\mathcal{D}}(t, p, \phi, e) = D$ *then* $\varepsilon(\top(D)) = \mathcal{E}(t, p, \phi, e)$.

3 MOTIVATION

In this section, we motivate the study of chain of custody for layered attestations using a series of example attestation scenarios, modeled as Copland phrases in the syntax of Figure 1.

Consider a setting in which an appraiser app must assess the runtime corruption state of a system sys running on a target device. The appraiser can request that virus-checking software vc running in userspace us on the target take a runtime measurement of sys, which also runs in userspace. The appraiser can also request that a trusted virus-checker measurer vcm running in kernelspace ks take a runtime measurement of vc.

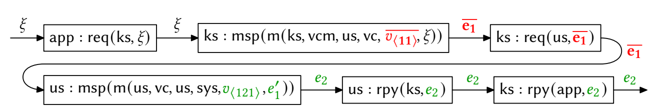
One way of structuring this attestation would see the vcm first measure the vc and then hand its evidence off to the latter for incorporation into the measurement of sys. The Copland phrase in Example 1 represents this structure.

Example 1.

$$*\mathsf{app} : @_{\mathsf{ks}} [\mathsf{vcm}\ \mathsf{us}\ \mathsf{vc} \to @_{\mathsf{us}} [\mathsf{vc}\ \mathsf{us}\ \mathsf{sys}]]$$

Suppose an adversary has corrupted the runtime state of vc sometime prior to the attestation. The measurement evidence produced by the vcm would reveal this corruption to the appraiser. Unfortunately, in this case the only copy of this evidence the appraiser receives is that which passes through the corrupt vc. The adversary in control of vc can direct it to alter the legitimate, corruption-revealing measurement evidence so that the appraiser concludes all is well.

The diagram below illustrates the Copland data flow semantics of this attestation and depicts how evidence tampering at vc allows the corruption to go undetected. Corruption-revealing evidence is shown in bold red, evidence that passes appraisal is in green.



Since vc receives a copy of the evidence collected by vcm, the vc measurement represents a *tampering opportunity* for this evidence.

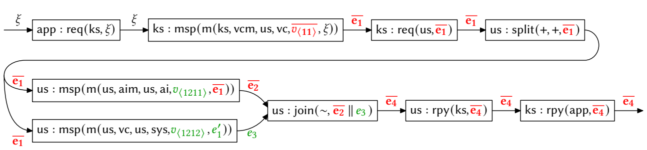
To take advantage of the tampering opportunity, an adversary must ensure that vc is corrupt when the evidence is received and that the evidence is not integrity protected in a way that prohibits tampering. Integrity protection is discussed further throughout the rest of the paper. This scenario exemplifies a conflict of interest: a component whose corruption state is at issue is nevertheless made sole custodian of the evidence establishing that state. Indeed, in this scenario, the adversary need only tamper at vc to completely avoid detection.

We might attempt to solve this problem by simply providing the appraiser with an additional copy of the vcm evidence via a third-party. This is the approach taken in the attestation of Example 2. Here, the appraiser also reviews the target device's asset inventory `ai`, which is measured by a userspace asset inventory manager `aim`. As indicated by the +~+ operator, the `aim` and vc measurements occur in parallel and both receive the vcm measurement evidence as input.

Example 2.

$$*\mathsf{app} : @_{\mathsf{ks}}\ [\mathsf{vcm}\ \mathsf{us}\ \mathsf{vc} \rightarrow @_{\mathsf{us}}\ [\mathtt{aim}\ \mathsf{us}\ \mathtt{ai}\ {+}{\sim}{+}\ \mathsf{vc}\ \mathsf{us}\ \mathsf{sys}]]$$

The following diagram depicts a possible execution of this attestation in which vcm once again collects evidence revealing corruption at vc. Separate copies of this evidence e_1 reach vc and `aim`, making tampering opportunities of both measurements. Since vc is corrupt, it is in a position to tamper with the copy of e_1 it receives. However, in order to tamper with the copy that reaches `aim`, the adversary must corrupt another component, namely `aim` itself or one of the components that receives the evidence generated by `aim`. If the adversary is unable to do so, this second copy of e_1 will ultimately reach the appraiser, revealing the corruption of vc.



Thus, the effect of providing several copies of evidence to the appraiser along separate *data flow paths* is to force an adversary to work harder in order to avoid detection. They might be required to tamper with multiple copies of evidence, that is, take advantage of multiple tampering opportunities for the evidence, requiring in turn multiple runtime corruptions in distinct components. As some of these corruptions may be more feasible than others, the adversary will be obliged to consider different patterns of corruption and tampering, which we refer to as alternative *tampering strategies*. Of course, an adversary will be interested in *minimal* tampering strategies, those involving the fewest corruptions.

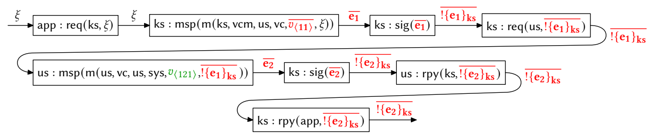
Unfortunately, providing redundant copies of evidence is not a panacea: `aim`, for instance, may be easy to corrupt, giving the adversary an easy out. A more robust approach to making tampering more difficult is to apply integrity protections to evidence. The Copland phrase in Example 3 is another variation of the phrase in Example 1. Here, the measurement evidence generated by vcm is digitally signed with the kernelspace signing key before it is forwarded to vc.

Example 3.

$$*\mathsf{app} : @_{\mathsf{ks}}\ [\mathsf{vcm}\ \mathsf{us}\ \mathsf{vc} \rightarrow\ !\ \rightarrow @_{\mathsf{us}}\ [\mathsf{vc}\ \mathsf{us}\ \mathsf{sys} \rightarrow\ !]]$$

The userspace component vc cannot access the kernelspace signing component. This means an adversary active at vc will be unable to obtain a new kernelspace signature on tampered evidence. Any tampering in spite of this fact will cause a signature verification failure at the appraiser, alerting it to the adversary's activities and hence defeating the purpose of tampering in the first place. In short, the addition of the kernelspace signature eliminates the tampering opportunity at vc with the vcm evidence. The tampering opportunities in the previous examples crucially depended on the vcm evidence lacking integrity protections.

As illustrated in the following diagram, evidence of corruption at vc now makes it all the way to the appraiser regardless of userspace corruptions.



Thus, the addition of digital signatures at key places in the data flow can limit—maximally, as we show—the number of tampering opportunities and strategies available to an adversary.

4 PATHS AND TAMPERING

Section 3 illustrated the central role of data flow analysis in identifying tampering opportunities and strategies. In this section, we develop the relevant theory to make these analyses precise.

For the remainder of this section, we fix a data flow graph $D = \bar{\mathcal{D}}(t, p, \phi, e)$ for some t, p, and e. Events v and v' range over D^V.

4.1 Defining Tampering Opportunities

A component can only tamper with evidence it receives. Thus, the first step in identifying tampering opportunities for a piece of evidence is to identify which components receive a copy of the evidence as the attestation executes. Using the data flow semantics, we can approach this task systematically by considering the data flow paths beginning at the event that generates the evidence of interest.

Definition 6 (Path). *For a data flow graph $D = (V, v^i, v^o, \rightarrow, \ell)$, a* path *is a sequence of events $\pi = \langle v_1, \ldots, v_n \rangle$ such that, for every $i < n$, $v_i \rightarrow v_{i+1}$. To append an event to the end of a path we write $\langle v_1, \ldots, v_n \rangle \S\, v_{n+1}$ for $\langle v_1, \ldots, v_{n+1} \rangle$. Concatenation is written as $\langle v_1, \ldots, v_n \rangle * \langle w_1, \ldots, w_k \rangle = \langle v_1, \ldots, v_n, w_1, \ldots, w_k \rangle$. We use $\Pi(v, v')$ to denote the set of all paths from v to v'.*

Data flow paths represent the trajectories of evidence through the attestation. Since the evidence emitted at a non-split data flow event always includes the evidence received as input there, every subsequent event along a path beginning with v receives the evidence emitted at v. This evidence is received at v' if and only if there is a data flow path starting at v and ending at v'.

As discussed in Section 3, the primary defense against tampering is to integrity-protect evidence by signing it. We assume that each place has a distinct evidence-signing function that all and only components residing at that place may access. One consequence of this assumption is that a signature generated at place p does not protect against tampering by other components at p. This is because components located at p can always obtain a new p signature on tampered evidence. Thus, it is important to keep track of which places have signed evidence as it flows along paths. An important case of this is when all signatures (if any) were generated at a single place.

Definition 7 (Place-signing path). *Let p be a place. A path π is a p-*signing path *iff for all $1 \leq i \leq |\pi|$, $\ell(\pi(i)) = q : \mathrm{sig}(e)$ implies $q = p$.*

With these definitions, we are ready to formalize the intuitions developed in Section 3 about an adversary's ability to tamper with evidence in such a way that an appraiser would not detect it.

Definition 8 (Tamper Opportunity). *A path $\pi = \langle v, \ldots, v' \rangle$* permits tampering *at v' with the evidence from v iff*

(1) v is a measurement event,
(2) $v \neq v'$ (i.e. $|\pi| > 1$), and
(3) π is $\varrho(v')$-signing or $\varrho'(v')$-signing.

The set of tamper-permitting paths from v to v' is denoted $\overline{\Pi}(v, v')$. We say v' is a tamper opportunity *for v iff $\overline{\Pi}(v, v') \neq \emptyset$.*

Condition 1 of Definition 8 restricts adversarial tampering to measurement evidence, that is, evidence created at measurement events. This focuses our analyses on the core objective of a tampering adversary, which is to change measurement results that reveal corruption into ones that do not. Measurement events are the only generators of these key values, with the other data flow events simply passing them along, bundling them into larger structures, or applying a cryptographic function to them. By homing in on the evidence generation sites, we ensure that we have the complete history of a measurement in mind when deciding its tampering opportunities. Condition 1 also serves the dual purpose of excluding tampering with other forms of evidence that we consider will always be detected by an appraiser. For instance, it excludes tampering with a hash or signature directly or altering the structure of evidence so that it does not match the appraiser's expectations.

Condition 2 disallows tampering at the event that generates the evidence of tampering interest. Such "self-tampering" is much more akin to evidence forgery, in which a corrupt measurer simply lies about the state of the target and forges a result that will pass appraisal. Prior work has considered this kind of evidence forgery in the layered attestation setting [18, 19].

Finally, Condition 3 codifies our assumptions about the integrity protection provided by signatures. By inspecting the signature events along a data flow path that delivers the evidence produced at v to v', we can determine which places sign that copy of the evidence before it is received. If v' occurs at p but a signature event at a different place q is found along the path, the q-signature prevents v' from tampering. However, p-signatures along the path provide no defense against tampering at v', and if all such signatures are p-signatures, the evidence produced at v lacks any integrity protections with respect to v'. The component executing v' can use its access to the signing component at p to obtain new signatures on tampered evidence.

Definition 8 allows us to more formally state the Tamper Opportunities Problem as follows.

Tamper Opportunities Problem (TOP):
Given a Copland phrase t, a place p, and a measurement event v in $\bar{C}(*p : t)$, compute $\mathtt{Opps}(v) = \{v' \mid \overline{\Pi}(v, v') \neq \emptyset\}$.

4.2 Defining Tamper Strategies

Definition 8 formalizes our intuitions surrounding an adversary's ability to tamper with evidence at a single event, that is, with a particular copy of evidence at a single component. However, the lesson of Example 2 is that tampering at a single event is sometimes insufficient to avoid detection. Rather, *all* copies of a particular piece of incriminating evidence that reach the appraiser must be tampered with.

As Lemma 1 shows, the evidence that results from executing a Copland phrase is that emitted by the output event of the phrase's data flow semantics. This means the appraiser receives the evidence emitted at any event v that has a data flow path ending in the output event, i.e., for which $\Pi(v, \top(D)) \neq \emptyset$. Thus, a successful strategy for an adversary to tamper with evidence created at v consists of a set of events v'_i such that each path $\langle v, \ldots, v'_i \rangle$ permits tampering and every path in $\Pi(v, \top(D))$ encounters one of the v'_i.

Definition 9 (Tamper Strategy). *A set of events S is a* tamper strategy for *$v \in D^V$ iff for each $\pi \in \Pi(v, \top(D))$, there is a $v' \in S$ such that π contains v' and the subpath $\langle v, \ldots, v' \rangle$ of π permits tampering.*

Of course, Definition 9 leaves room for wildly inefficient tamper strategies. For example, if any tamper strategy for v exists, then the set of all events is a tamper strategy for v. "Big" strategies tell us little about what events, if any, *must* be part of a tamper strategy for a given event. They are also inherently of less interest to an adversary, who will seek to minimize the number of additional corruptions needed to avoid detection. A more natural notion is therefore that of a minimal tamper strategy in which removing any of the events of S results in a set that is no longer a tamper strategy.

Definition 10 (Minimal Tamper Strategy). *S is a* minimal tamper strategy for *$v \in D^V$ iff*

(1) S is a tamper strategy for v and
(2) if S' is a tamper strategy for v and $S' \subseteq S$, then $S' = S$.

Observe that if $\Pi(v, \top(D))$ is empty, any set of events vacuously satisfies Definition 9, and hence the empty set is the unique minimal tamper strategy for v. This is reasonable because the evidence created at v never reaches the appraiser in this case, hence no tampering with this evidence is necessary. In general, however, there may be numerous minimal tamper strategies.

It is now clear how to formally state the Tamper Strategies Problem in terms of Definition 10.

Tamper Strategies Problem (TSP):
Given a Copland phrase t, a place p, and a measurement event v in $\bar{C}(*p : t)$, find all minimal tamper strategies S for v.

4.3 Computing Tamper Opportunities

We propose the following procedure that takes as input a starting event v and outputs the set O of tamper opportunities for v. It computes all paths that start with v and collects tamper opportunities within each path.

Algorithm to solve TOP:

(1) Set $P = \{\langle v\rangle\}$ and $O = \emptyset$.
(2) Remove a path $\pi \S v$ from P (initially $\pi = \langle\rangle$).
  (a) If $\pi \S v$ is tamper-permitting, add v to O.
  (b) Otherwise leave O unchanged.
(3) For each $\pi' = \pi \S v \S v'$ extending $\pi \S v$ by a single event, add π' to P.
(4) If P is non-empty, go to step 2, otherwise, return O.

It is easy to see that this procedure always terminates and outputs $O = \mathtt{Opps}(v)$. The procedure incrementally builds all paths starting at v, checking the three conditions of Definition 8 at each step until the paths cannot be extended further and recording all paths that are tamper-permitting along the way.

Line 2(a) of this algorithm to solve TOP requires verifying the conditions of Definition 8. Though doing so directly is straightforward, we instead introduce the *tamper set* function to simplify the process. This function applies to paths $\langle v_1, \ldots, v_n\rangle$ and computes the set of places that can tamper with measurement evidence generated at v_1 by the time it leaves v_n.

Definition 11 (Tamper set). *The* tamper set *function $ts(\cdot)$ outputs a set of places given a path and is defined as follows. Let $\mathcal{P}$ be the set of all places.*

$$ts(\langle v\rangle) = \begin{cases} \mathcal{P} & \text{if } \ell(v) = p : \mathsf{msp}(\ldots) \\ \emptyset & \text{otherwise} \end{cases}$$
$$ts(\pi \S v) = \begin{cases} \{p\} \cap ts(\pi) & \text{if } \ell(v) = p : \mathsf{sig}(e) \\ ts(\pi) & \text{otherwise} \end{cases}$$

A tamper set is either the set of all places, a singleton set, or the empty set. There are two observations to made about Definition 11. First, if π does not start with a measurement event, then $ts(\pi) = \emptyset$. This is in keeping with our exclusion of non-measurement evidence from tampering considerations. Secondly, it is easy to check that the function is weakly decreasing in the following sense:

$$ts(\pi_1 * \pi_2) \subseteq ts(\pi_1)$$

Theorem 1 shows that the tamper set function can be used to verify whether or not a path is tamper-permitting.

Theorem 1. *For any path $\pi = \pi' \S v$ that starts at a measurement event, π is tamper-permitting iff either $\varrho(v)$ or $\varrho'(v)$ is in $ts(\pi')$.*

4.4 Computing Tamper Strategies

Recall that a tamper strategy for v is a set of events at which the evidence of corruption produced at v can be can be changed so the appraiser detects nothing wrong. Observe that if v has a tamper strategy, then it has a minimal one. This is because sets of events form a well-founded partial order under inclusion. We also remark that, since tamper strategies are sets of tamper opportunities, every tamper strategy for v is a subset of $\mathtt{Opps}(v)$, the set of all tamper opportunities for v. Finally, we assert that $\mathtt{Opps}(v)$ itself is a tamper strategy for v. While not immediately obvious, this is a simple consequence of Theorem 2 in the next section.

Given these observations, our approach to solving TSP is to start with $\mathtt{Opps}(v)$, which we can compute via our algorithmic solution to TOP, and incrementally climb down the partial order of sets until we find the subsets of $\mathtt{Opps}(v)$ that are minimal tamper strategies for v. More concretely, we propose the following procedure that takes as input an event v and outputs the set M of minimal tamper strategies for v.

Algorithm to solve TSP:

(1) Set $\mathcal{S} = \{\mathtt{Opps}(v)\}$ and $M = \emptyset$.
(2) Remove a tamper strategy S from $\mathcal{S}$.
(3) For every $v \in S$, compute $S' = S \setminus \{v\}$.
  (a) If S' is a tamper strategy, add S' to $\mathcal{S}$.
(4) If no strategies were added to $\mathcal{S}$ in the above step, add S to M.
(5) If $\mathcal{S}$ is not empty, go to step 2, otherwise return M.

5 LIMITS TO CONSTRAINING TAMPER STRATEGIES

We are now equipped to explore the ways in which we can limit the tamper strategies available to an adversary.

The assumption that p signatures do not provide protection against tampering by other p-located components has significant implications for our ability to limit tamper strategies. It means, for instance, that if π is a path whose constituent events all occur at the same place, then a measurement event in π has a tamper opportunity at every subsequent event in π. We begin by formalizing this observation.

Definition 12 (Local Path). *A path $\pi = \langle v_1, \ldots, v_n\rangle$ in a data flow graph is* local *to place p iff $\varrho(v_i) = \varrho'(v_i) = p$ for all $1 < i < n$ and $\varrho(v_j) = p$ or $\varrho'(v_j) = p$ for $j \in \{1, n\}$.*

A local path is one that never leaves a given place. Definition 12 allows for a path local to place p to start or end with a cross-place communication event whose sending or receiving place is p. As anticipated, we can indeed prove that local paths always permit tampering.

Lemma 2 (Local Path Tampers). *Let $v \in D^V$ be a measurement event and suppose $\pi = \langle v, \ldots, v'\rangle$ is a path of length greater than one. If π is local, then π permits tampering.*

Might there be a way to structure a Copland phrase that precludes the existence of tamper strategies in spite of this fact? Unfortunately, the answer is negative. Every path from a measurement event to the output of a data flow graph has a local prefix which permits tampering.

Theorem 2 (Strategy Exists). *For every measurement event $v \in D^V$, if $v \neq \top(D)$, then there is a tamper strategy for v.*

Theorem 2 excludes the corner case in which the measurement event in question is the output event. This corresponds to a scenario in which the appraiser takes a measurement for its own consumption at the conclusion of the attestation. Since no events succeed this measurement, there are of course no tamper opportunities for it. However, such a measurement would be unusual. Appraisers typically do not measure themselves, and measurements of components on other systems are usually mediated by requests and replies, which makes them subject to Theorem 2. Thus, tamper strategies cannot be eliminated for most measurement events, because local events follow them. This negative result puts an upper limit on how much protection against tampering is possible.

We cannot hope to eliminate tampering along local paths. Is it possible to ensure these are the only available avenues for tampering? In other words, are there data flow graphs for which no non-local paths permit tampering? This time we can answer in the affirmative. The following definition specifies a property that a path or a graph can have to ensure that tampering is confined to occur only along local paths.

Definition 13. *Let $\pi = \langle v_1, \ldots, v_n \rangle$ be a path. We say π is* protected*, iff either*

(1) v_1 is not a measurement event, or
(2) for every $1 < i \le n$, if v_i is a cross-place communication event, then $ts(\langle v_1, \ldots, v_i \rangle) \subseteq \{\varrho(v_i)\}$.

We say a data flow graph is protected iff all paths in the graph are protected.

A protected data flow graph should be one in which adversarial tampering is maximally limited. That is, protected graphs should have the property that all tamper-permitting paths are local. This is indeed the case, as established by the following theorem.

Theorem 3. *If $\pi = \langle v_1, \ldots, v_n \rangle$ is protected and permits tampering, then π is local to $\varrho(v_1)$.*

6 TRANSFORMING COPLAND PHRASES

At the end of the previous section, we introduced the concept of a *protected* data flow graph and showed that these maximally limit tamper opportunities. Here, we present an algorithm that transforms any Copland phrase $*p : t$ into another phrase $*p : t'$ that preserves the evidence collected by the original phrase, but whose data flow graph is protected. In doing so, we provide a solution to the EPPP, which we can now formalize in terms of protected graphs.

> **Evidence-Protecting Program Problem (EPPP):**
> Given a Copland phrase $*p : t$, produce another phrase $*p : t'$ whose semantics is a protected graph that preserves the evidence collected by $*p : t$.

Informally, evidence preservation means that all events and data flows in the original semantics are carried over to the transformed phrase.

Constructing this algorithm involves shifting from the data flow-centric view of tampering developed in previous sections to an evidence-centric view. To illustrate this shift, consider the evidence term $e = !\{e_1\}_p \parallel !\{e_2\}_q$, where e_1 and e_2 are measurements. By examining the nested structure of signatures applied to embedded measurements, we can read off right from e which places can tamper with each measurement. Here, for instance, only events occurring at p can tamper with e_1 and only events at q with e_2; thus, a tamper opportunity for the measurements in e is an event that occurs at p or q. The structure of evidence encodes enough of each measurement's data flow history to draw such conclusions.

In order to make these observations precise, we define the *tamper places* function τ. Given an evidence term e, $\tau(e)$ is the set of places that can tamper with measurement evidence embedded in e.

Definition 14. *Let $\mathcal{P}$ be the set of all places. Given an evidence value e, the* tamper places *function returns a possibly empty set of places and is defined inductively as follows.*

$$\begin{aligned}
\tau(\xi) &= \varnothing \\
\tau(\mathsf{m}(\mathsf{msp}(m, q, t), p, \phi, e)) &= \mathcal{P} \\
\tau(\#\{e\}_p) &= \tau(e) \\
\tau(!\{e\}_p) &= \{p\} \cap \tau(e) \\
\tau(e_1 \;;;\; e_2) &= \tau(e_1) \cup \tau(e_2) \\
\tau(e_1 \parallel e_2) &= \tau(e_1) \cup \tau(e_2)
\end{aligned}$$

The tamper places function codifies our intuitions about the effects of the various Copland actions on the vulnerability of measurement evidence to tampering. An unsigned measurement is tamperable anywhere, signatures restrict tampering at most to the places that apply them, and operations that combine evidence have no effect.

We first establish that the evidence-centric view of tampering represented by τ has the correct connection to the path-centric view. Recall the tamper set function defined in Section 4: $ts(\langle v_1, \ldots, v_n \rangle)$ computes the set of places that can tamper with measurement evidence created at v_1 after it has propagated down the given data flow path through v_n. Theorem 1 characterizes tamper-permitting paths in terms of $ts(\cdot)$. We now establish the following relationship between $\tau(\cdot)$ and $ts(\cdot)$.

Lemma 3. *If $\pi = \langle v_1, \ldots, v_n \rangle$ is a path starting with a measurement event, then $ts(\pi) \subseteq \tau(\varepsilon(v_n))$.*

Thus, applying τ to the evidence emitted at the end of π overapproximates $ts(\pi)$. This is sufficient for our purposes, as it allows us to obtain an evidence-centric criterion for deciding whether a data flow graph is protected.

Lemma 4. *Given a flow graph D, if every cross-place communication event v in D satisfies $\tau(\varepsilon(v)) \subseteq \{\varrho(v)\}$, then D is protected.*

Lemma 4 provides us with a sufficient condition to aim for in constructing an algorithm to solve the EPPP. Namely, we can guarantee a Copland phrase's semantics is a protected graph, and therefore maximally limits tamper opportunities, by ensuring that $\tau(\varepsilon(v)) \subseteq \{\varrho(v)\}$ holds for every cross-place communication event v in its semantics. This suggests a procedure that computes $\tau(e)$ for input evidence e received at each cross-place communication event v and adds a signature immediately before v only if $\tau(\varepsilon(v)) \not\subseteq \{\varrho(v)\}$.

Definition 15 (Evidence Protection Program). *The procedure to transform one phrase into another is defined inductively as* *follows.*

$$\overline{\mathsf{epp}}(*p : t) = *p : \mathsf{epp}(t, p, \xi)$$

$$\begin{aligned}
\mathsf{epp}(m\ q\ t, p, e) &= m\ q\ t \\
\mathsf{epp}(-, p, e) &= - \\
\mathsf{epp}(\#, p, e) &= \# \\
\mathsf{epp}(!, p, e) &= ! \\
\mathsf{epp}(t_1 \to t_2, p, e) &= \mathsf{epp}(t_1, p, e) \to \mathsf{epp}(t_2, p, \mathcal{E}(\mathsf{epp}(t_1, p, e), p, \langle\rangle, e)) \\
\mathsf{epp}(@_p\ [t], p, e) &= @_p\ [\mathsf{epp}(t, p, e)] \\
\mathsf{epp}(@_q\ [t], p, e) &= \begin{cases} @_q\ [\mathsf{epp}(t, q, e)] & \text{if } \tau(e) \subseteq \{p\} \text{ and } \tau(e_1) \subseteq \{q\} \\ @_q\ [\mathsf{epp}(t, q, e) \to !] & \text{if } \tau(e) \subseteq \{p\} \text{ and } \tau(e_1) \not\subseteq \{q\} \\ ! \to @_q\ [\mathsf{epp}(t, q, !\{e\}_p)] & \text{if } \tau(e) \not\subseteq \{p\} \text{ and } \tau(e_2) \subseteq \{q\} \\ ! \to @_q\ [\mathsf{epp}(t, q, !\{e\}_p) \to !] & \text{if } \tau(e) \not\subseteq \{p\} \text{ and } \tau(e_2) \not\subseteq \{q\} \end{cases} \\
\mathsf{epp}(t_1\ l\ o\ r\ t_2, p, e) &= \mathsf{epp}(t_1, p, \mathcal{F}(l, e))\ l\ o\ r\ \mathsf{epp}(t_2, p, \mathcal{F}(r, e))
\end{aligned}$$

where $e_1 = \mathcal{E}(\mathsf{epp}(t, q, e), q, \langle\rangle, e)$ *and* $e_2 = \mathcal{E}(\mathsf{epp}(t, q, !\{e\}_p), q, \langle\rangle, !\{e\}_p)$ *and* $l, r \in \{-, +\}, o \in \{<, \sim\}$.

As an example, the Copland phrase in Example 3 of Section 3 is the output of Evidence Protection Program on Example 1.

This Evidence Protection Program was designed with two goals in mind. The primary goal is that it should provide a general solution to the EPPP by transforming any Copland phrase into a maximally tamper-resistant version of itself. A naive approach to our strategy of adding signatures before cross-place communication events would do so blindly, always inserting a signature whether or not it has any effect on tampering. While this would work, we can do much better. Our second goal for Evidence Protection Program is that it should minimally perturb the input phrase, adding only those signatures needed to ensure its semantics is a protected graph. Thus, a necessary feature of the program is idempotency, to guarantee that when Evidence Protection Program is applied to its own output, the phrase is left unchanged.

Several particular aspects of Evidence Protection Program merit discussion. First, epp only inserts signature events into the phrase and always does so using the $\to$ operator. This guarantees that the semantics of the original phrase is preserved by the transformation, one of the requirements of a solution to the EPPP. Note also that signatures are only added at the beginning or end of $@_q\ [t]$ phrases and only when q is known to be different from p, the place where the attestation is currently executing. This ensures that signing events are only added immediately before cross-place communication events. Finally, the side conditions in Definition 15 ensure that we do not add more signatures than necessary to ensure the condition of Lemma 4. In each of the four main cases, if the condition is already satisfied, no signature is added. This parsimonious approach allows us to establish subsequently that Evidence Protection Program is indeed idempotent.

Theorem 4 establishes that the transformations applied by Evidence Protection Program guarantee the data flow semantics of the transformed phrases meet the conditions of Lemma 4. As a simple corollary of these two lemmas, we conclude that the data flow semantics of a phrase output by $\overline{\mathsf{epp}}$ is protected.

Theorem 4. *For any phrase t, place p, evidence e, sequence ϕ, if v is a cross-place communication event in $\bar{\mathcal{D}}(\mathsf{epp}(t, p, e), p, \phi, e)$, then $\tau(\varepsilon(v)) \subseteq \{\varrho(v)\}$.*

Corollary 1. *For any phrase $*p : t$, $\bar{C}(\overline{\mathsf{epp}}(*p : t))$ is protected.*

Thus, Evidence Protection Program provides an algorithm to solve the EPPP for any input Copland phrase, which we state formally.

Algorithm to solve EPPP: Given $*p : t$ as input, return $\overline{\mathsf{epp}}(*p : t)$.

Finally, Theorem 5 establishes that Evidence Protection Program is idempotent. This proves that the side conditions in Definition 15 work as intended to ensure that the only signatures added are those necessary to guarantee the data flow graphs meet the conditions of Lemma 4.

Theorem 5. *The function* epp *is idempotent. That is, for any t, p, e, it follows that* $\mathsf{epp}(\mathsf{epp}(t, p, e), p, e) = \mathsf{epp}(t, p, e)$. *Hence also* $\overline{\mathsf{epp}}(\overline{\mathsf{epp}}(*p : t)) = \overline{\mathsf{epp}}(*p : t)$.

This shows that Evidence Protection Program meets both of its design goals, including resolving the EPPP.

7 RELATED WORK

The foundation of our the analyses in our work is Copland [17], a language for specifying layered attestations. We provide several formal semantics to facilitate precise reasoning about attestation matters. A language-based approach for specifying attestations provides flexibility to adapt to many different situations and architectures [7]. There is growing recognition that this approach is beneficial in supporting the full range of use cases that can benefit from remote attestation, from software defined networking [20], to smart cities [10], to flight planning for unmanned aerial vehicles [15]. Such recognition is even appearing in standards documents [4].

While such flexibility is useful, it introduces the challenge of having to determine the trustworthiness of any given protocol. A variety of efforts have investigated the trustworthiness of remote attestation. These range from low-level analyses of a “late launch” capability [5] to meta-analyses of high-level principles [21]. The ones most related to our current work are those that explore the possibility that an adversary could corrupt a component after it is measured in order to successfully fool an appraiser into accepting an attestation when it should not [3, 8, 16, 18, 19]. This is a time-of-check-time-of-use (TOCTOU) attack. The chain-of-custody attacks considered in this paper share some aspects with TOCTOU attacks. They both fool an appraiser due to adversarial actions that occur after the time of measurement. However, a key difference is that in a TOCTOU attack, the target of measurement is assumed to be uncorrupted at the time of measurement, yielding measurement evidence that is not incriminating. In a chain-of-custody attack, the target is assumed already to be corrupt at the time of measurement. This yields incriminating evidence that is tampered with by a downstream corrupt component.

Flexible attestation managers such as [11] and [13] incorporate a selection policy that dictates which attestation protocols a system is willing to run in different scenarios. Since the target and the appraiser may have differing priorities, they must negotiate a protocol that serves the needs of both parties [6]. Methods for analyzing the trustworthiness of attestations, such as the present work and those cited in the previous paragraph, can provide valuable input into the negotiation and selection process.

8 CONCLUSION

In this paper, we initiated the study of evidence chain of custody for layered attestations. Using the data flow semantics of Copland phrases, we can describe the complete history of components that had access to each copy of given measurement evidence on its way to an appraiser. This semantics allowed us to formally define tamper opportunities for measurement evidence and to develop a procedure for identifying all components that could tamper with given measurement evidence at runtime if corrupted by an adversary. As illustrated via examples, the possibility of branching execution means that tampering with a single copy of evidence is generally insufficient for an adversary to prevent incriminating measurement evidence from reaching the appraiser. This drove us to define tamper strategies, sets of tamper opportunities for given evidence that, if leveraged, would allow an adversary to avoid detection via evidence tampering. We then developed a procedure for identifying all minimal tamper strategies for given evidence.

While the ability to identify tamper strategies is useful, an attestation designer's chief interest lies in reducing or eliminating tamper strategies. For this reason, we also developed an Evidence Protection Program that transforms Copland phrases into maximally tamper-resistant iterations of themselves. The procedure preserves the semantics of the input phrase while maximally limiting tamper opportunities and strategies within it by adding digital signatures at key places in the semantics. As we show, Evidence Protection Program is idempotent, suggesting that it minimally perturbs the input Copland phrase. This permits Copland phrase designers to write their phrases agnostic of tamper-preventing signatures and use Evidence Protection Program to apply integrity protections afterward. Evidence Protection Program and the procedure for finding minimal tamper strategies can also be used in tandem during the initial design process to suggest alternatives with more desirable maximally-constrained tamper properties.

In the future, it would be fruitful to integrate tamper analyses into a larger trust analysis engine similar to the one presented in [19]. This would provide a more comprehensive understanding of the ways an adversary can avoid detection by combining evidence tampering with evidence forgery.

REFERENCES

[1] https://github.com/mitre/copland/tree/main/src/tampering.
[2] *The Coq proof assistant reference manual*, 2021. Version 8.17.1, https://coq.inria.fr.
[3] Tigist Abera, N. Asokan, Lucas Davi, Jan-Erik Ekberg, Thomas Nyman, Andrew Paverd, Ahmad-Reza Sadeghi, and Gene Tsudik. C-flat: Control-flow attestation for embedded systems software. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 743–754, New York, NY, USA, 2016. Association for Computing Machinery.
[4] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan. RFC 9334: Remote Attestation Procedures Architecture, 2023. https://datatracker.ietf.org/doc/rfc9334/.
[5] Anupam Datta, Jason Franklin, Deepak Garg, and Dilsun Kaynar. A logic of secure systems and its application to trusted computing. In *2009 30th IEEE Symposium on Security and Privacy*, pages 221–236. IEEE, 2009.
[6] Anna Fritz and Perry Alexander. A framework for policy based negotiation. In Kristin Yvonne Rozier and Swarat Chaudhuri, editors, *NASA Formal Methods*, pages 207–223, Cham, 2023. Springer Nature Switzerland.
[7] Sarah C. Helble, Ian D. Kretz, Peter A. Loscocco, John D. Ramsdell, Paul D. Rowe, and Perry Alexander. Flexible mechanisms for remote attestation. *ACM Trans. Priv. Secur.*, 24(4), sep 2021.
[8] Anum Khurshid and Shahid Raza. Autocert: Automated toctou-secure digital certification for iot with combined authentication and assurance. *Computers & Security*, 124:102952, 2023.
[9] Peter A. Loscocco, Perry W. Wilson, J. Aaron Pendergrass, and C. Durward McDonell. Linux kernel integrity measurement using contextual inspection. In *Proceedings of the 2007 ACM workshop on Scalable trusted computing*, STC '07, pages 21–29, New York, NY, USA, 2007. ACM.
[10] Louis Moreau, Emmanuel Conchon, and Damien Sauveron. An adaptive simultaneous multi-protocol extension of craft. *Sensors*, 23(8):4074, 2023.
[11] J. Aaron Pendergrass, Sarah Helble, John Clemens, and Peter Loscocco. A platform service for remote integrity measurement and attestation. In *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, pages 1–6, 2018.
[12] J. Aaron Pendergrass, Sarah Helble, John Clemens, and Peter A. Loscocco. Maat: A platform service for measurement and attestation. *CoRR*, abs/1709.10147, 2017.
[13] Adam Petz and Perry Alexander. A copland attestation manager. In *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*, pages 1–10, 2019.
[14] Adam Petz and Perry Alexander. An infrastructure for faithful execution of remote attestation protocols. In Aaron Dutle, Mariano M. Moscato, Laura Titolo, César A. Muñoz, and Ivan Perez, editors, *NASA Formal Methods*, pages 268–286, Cham, 2021. Springer International Publishing.
[15] Adam Petz, Grant Jurgensen, and Perry Alexander. 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*, pages 111–117, 2021.
[16] Yu Qin, Jingbin Liu, Shijun Zhao, Dengguo Feng, and Wei Feng. Ripte: runtime integrity protection based on trusted execution for iot device. *Security and Communication Networks*, 2020:1–14, 2020.
[17] John D. Ramsdell, Paul D. Rowe, Perry Alexander, Sarah C. Helble, Peter Loscocco, J. Aaron Pendergrass, and Adam Petz. Orchestrating layered attestations. In Flemming Nielson and David Sands, editors, *Principles of Security and Trust*, volume 11426, pages 197–221, Cham, 2019. Springer International Publishing. https://ku-sldg.github.io/copland/resources/copland-post-2019.pdf.
[18] Paul D. Rowe. Confining adversary actions via measurement. *Third International Workshop on Graphical Models for Security*, pages 150–166, 2016.
[19] Paul D. Rowe, John D. Ramsdell, and Ian D. Kretz. Automated trust analysis of copland specifications for layered attestations. In *23rd International Symposium on Principles and Practice of Declarative Programming*, PPDP 2021, New York, NY, USA, 2021. Association for Computing Machinery.
[20] Nik Sultana, Deborah Shands, and Vinod Yegneswaran. A case for remote attestation in programmable dataplanes. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, pages 122–129, 2022.
[21] Ahmad B Usman, Nigel Cole, Mikael Asplund, Felipe Boeira, and Christian Vestlund. Remote attestation assurance arguments for trusted execution environments. In *Proceedings of the 2023 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*, pages 33–42, 2023.

A COPLAND DATA FLOW SEMANTICS

The Copland data flow semantics associates a data flow graph to a Copland phrase. As we saw above, the semantics is recursively defined by stitching together smaller data flow graphs. We therefore need to specify functions to do this stitching. We rely on two methods depending on whether or not data is passed from the output of one graph as input to the next.

The before copy operation links two data flow graphs, passing the output of one to the input of the next.

Definition 16 (Before Copy). *Let $D_1 = (V_1, v_1^i, v_1^o, \to_1, \ell_1)$ and $D_2 = (V_2, v_2^i, v_2^o, \to_2, \ell_2)$ be two event systems where V_1 and V_2 are disjoint. The before copy operation $\rhd\!+$ is*

$$D_1 \rhd\!+ D_2 = (V_1 \cup V_2, v_1^i, v_2^o, \to_1 \cup \to_2 \cup \{v_1^o \to v_2^i\}, \ell_1 \cup \ell_2).$$

The before nil operation combines two data flow graphs but does not link them with a data flow edge. That is, the edge $v_1^o \to v_2^i$ is omitted.

Definition 17 (Before Nil). *Let $D_1 = (V_1, v_1^i, v_1^o, \to_1, \ell_1)$ and $D_2 = (V_2, v_2^i, v_2^o, \to_2, \ell_2)$ be two event systems where V_1 and V_2 are disjoint. The before nil operation $\rhd\!-$ is*

$$D_1 \rhd\!- D_2 = (V_1 \cup V_2, v_1^i, v_2^o, \to_1 \cup \to_2, \ell_1 \cup \ell_2).$$

$$\bar{C}(*p : t) = \bar{\mathcal{D}}(t, p, \langle\rangle, \xi)$$

$$\begin{aligned}
\bar{\mathcal{S}}(l) &= (\{v\}, v, v, \emptyset, \{v \mapsto l\}) \\
\bar{\mathcal{D}}(s_1\ q\ s_2, p, \phi, e) &= \bar{\mathcal{S}}(p : \mathsf{msp}(s_1, q, s_2, \phi, \mathsf{m}(\mathsf{ms}(s_1, q, s_2), p, \phi, e))) \\
\bar{\mathcal{D}}(@_q\ [t], p, \phi, e) &= \bar{\mathcal{S}}(p : \mathsf{req}(q, e)) \rhd\!+ \bar{\mathcal{D}}(t, q, 1 :: \phi, e) \\
&\qquad \rhd\!+ \bar{\mathcal{S}}(q : \mathsf{rpy}(p, \mathcal{E}(t, q, 1 :: \phi, e))) \\
\bar{\mathcal{D}}(\{\}, p, \phi, e) &= \bar{\mathcal{S}}(p : \mathsf{nul}(\xi)) \\
\bar{\mathcal{D}}(-, p, \phi, e) &= \bar{\mathcal{S}}(p : \mathsf{cpy}(e)) \\
\bar{\mathcal{D}}(!, p, \phi, e) &= \bar{\mathcal{S}}(p : \mathsf{sig}(!\{e\}_p)) \\
\bar{\mathcal{D}}(\#, p, \phi, e) &= \bar{\mathcal{S}}(p : \mathsf{hsh}(\#\{e\}_p)) \\
\bar{\mathcal{D}}(t_1 \to t_2, p, \phi, e) &= \bar{\mathcal{D}}(t_1, p, 1 :: \phi, e) \rhd\!+ \bar{\mathcal{D}}(t_2, p, 2 :: \phi, \mathcal{E}(t_1, p, 1 :: \phi, e)) \\
\bar{\mathcal{D}}(t_1\ l\ o\ r\ t_2, p, \phi, e) &= (V_5, v_5^i, v_5^o, \to_5, \ell_5)
\end{aligned}$$

$$\begin{aligned}
D_1 &= \bar{\mathcal{S}}(p : \mathsf{split}(l, r, e)) \\
D_2 &= D_1 \rhd l\ \bar{\mathcal{D}}(t_1, p, 1 :: \phi, \mathcal{F}(l, e)) \\
D_3 &= D_1 \rhd r\ \bar{\mathcal{D}}(t_2, p, 2 :: \phi, \mathcal{F}(r, e)) \\
D_4 &= \bar{\mathcal{S}}(p : \mathsf{join}(o, \mathcal{E}(t_1\ l\ o\ r\ t_2, p, \phi, e))) \\
V_5 &= V_2 \cup V_3 \cup V_4 \\
v_5^i &= \bot(D_1) \\
v_5^o &= \top(D_4) \\
\to_5 &= \to_2 \cup \to_3 \cup \{v_2^o \to v_4^i, v_3^o \to v_4^i\} \\
\ell_5 &= \ell_2 \cup \ell_3 \cup \ell_4
\end{aligned}$$

Figure 5: Data Flow Semantics

Using these two operations, we can now formally define a function that takes a Copland phrase and returns the data flow graph associated with it.

Definition 2 is the formal instantiation of the informal semantics described in the previous section.