Accountability in Security Protocols (preprint)

Robert Künnemann[1], Deepak Garg[2], and Michael Backes[3]

[1] CISPA, Saarland University
[2] MPI-SWS
[3] CISPA
Saarland Informatics Campus

Abstract. A new paradigm in secure protocol design is to hold parties accountable for misbehaviour instead of postulating that they are trustworthy. Recent approaches in defining this property, called accountability, have highlighted the difficulty of characterising malicious behaviour. So far, no satisfactory solution has been found. Consequently, existing definitions are either not truly protocol agnostic or require complete monitoring of all parties.
To our knowledge, this work is the first to formalize misbehavior in the following sense: a deviation from the behaviour prescribed by the protocol that caused a security violation. We propose a definition for the case where it is known which parties deviated in which respect, and extend this definition to the case where neither these deviations are known, nor the complete trace of the protocol. We point out that, under realistic assumptions, it is impossible to determine all misbehaving parties, however, we show that completeness can be relaxed to exclude spurious causal dependencies. We demonstrate the use of our definition with two case studies, a delegation protocol with a central trusted authority, and an actual accountability protocol from the literature. In both cases, we discover accountability violations and apply our definition to the fixed protocols.

1 Introduction

Trust in other parties is the foundation of all security protocols. In scenarios like electronic voting, certified e-mail, online transactions, or processing of personal data, however, the agents involved cannot be trusted to behave according to the protocol. Nevertheless, if the protocol can detect agents causing security violations, there is an incentive to avoid malicious deviation from the protocol. The ability of a protocol to provide the necessary information for detection is what we call *accountability* in this work. While juridical notions of accountability also require intent, foreseeability etc. to actually assign blame, we concentrate on the protocol's part in this process: assuring that all violations of some security property can be traced back to the agent or agents who caused it.

We call these parties misbehaving. Intuitively, a protocol provides accountability if it can always hold all misbehaving parties accountable. Many security

properties (soundly) approximate misbehaving parties by instead using the notion of a *dishonest* party, a party controlled by some arbitrary global adversary. A party may be *dishonest* from the start, but whether it is *misbehaving* in the actual case depends on whether this adversary actually decides to do something harmful. Moreover, it may depend on other parties, e.g., when two parties need to collude to break security. A *dishonest* party may not be *misbehaving*, but obviously a *misbehaving* party needs to be *dishonest* to deviate from the protocol.

Despite the conceptual difference between these two notions, analysing dishonest parties instead of misbehaving parties has shown sufficient for many security properties. Typically, the aim is to show that the protocol exhibits a certain behaviour (e.g., a secure multi-party computation protocol computes the correct results) if all parties are honest, but also if some parties are dishonest. The property to be achieved may be weaker in the latter case (e.g., only honest parties compute the correct results), but the guarantees for the honest parties are not weakened by the misclassification of dishonest parties that are not misbehaving. This is different for accountability. Ideally, the protocol should hold all misbehaving parties accountable, but only those parties. The latter, called soundness or fairness, is achieved if the judge never holds an honest party accountable. The former, however, which is called completeness, requires a proper notion of a misbehaving party, as any dishonest party might just behave according to protocol and thus be unrecognizable.

Hence previous definitions had no means of capturing completeness w.r.t. truly misbehaving parties, and thus had to rely on protocol specific policies specifying completeness; some even going as far as saying that accountability cannot be defined generally, but requires a policy [KTV10]. In Section 2 we go into detail about why such policies are either incomplete or need to be adapted to the protocol's auditing mechanism to an extent that calls their use as a specification into question.

There have been approaches to capture causation in protocols [GM15,FJW11, DGK+15], some with the aim of identifying misbehaving parties, but they focus on protocol actions as causes. While this is a useful building block for removing causally irrelevant communication from traces, so far the approach was not able to fulfil its promise to capture the completeness aspect of accountability. The underlying problem is that parties effectuating protocol actions that are causally related to a violation may do so without misbehaving, e.g., if A uses a confused deputy B to attack C, then B would be a cause for the attack, but probably not be held accountable. We therefore propose a new approach in which *the fact that A deviates* from the protocol is considered a (potential) cause rather than the individual action performed. In the above example, A's deviation alone suffices to cause the attack; even if B and C were modified to follow the protocol to the letter. We hence formalize and explore the following idea in this paper: (I) Accountability is the ability to point out all misbehaving parties. (II) Misbehaving parties are those who deviated and whose deviations alone are sufficient to violate the given security property.

Following this idea, we specify what constitutes a correct (a sound and complete) verdict in the case where these deviations are known a posteriori (Section 4.2). In practical scenarios, however, these deviations are not known. Moreover, we understand accountability as a property of the protocol itself, which manifests itself as a guarantee for every possible combination of deviations. Unfortunately, a fundamental problem negates a straightforward lifting of correctness in the above sense to all possible deviations, which we call the *provocation problem*. It occurs whenever two parties may communicate in secret, as any attacking party A can claim to be provoked by a message another party B sent. It would be illusory to assume all communication is visible; even if that is the case in the model, off-band communication can hardly be preempted in reality. Thus, it is always possible for A to shift the blame by claiming causal relations that are not inherent to its epistemic state or the protocol itself.

It is thus not possible to achieve completeness w.r.t. our understanding of what constitutes misbehaviour. However, we can achieve completeness w.r.t. *optimal* deviations which assume the absence of some of those spurious relations. In practice, this is what we desire in cases like the above: we want the protocol to point to A only; causal dependencies to other parties like B should be neglected. We present and discuss two notions of optimality and show that accountability w.r.t. either preserves soundness — hence we strictly advance the state of the art. We show that our definition is applicable by analyzing two very different case studies (Section 5): First, a protocol with a centralized trusted accountability monitor; second, a practical protocol for accountable algorithms from the literature, where accountability is provided from distributed logs and where no party is trusted. For both, we discover that accountability is not achieved. We subsequently repair these protocols and show accountability w.r.t. to the second notion of optimality for both protocols.

2 Related work

Various works from social and political science to computer science, from regulatory frameworks to IT-management guidelines, from governance standards to risk assessment procedures provide definitions of what accountability means in their respective contexts. Some definitions point out key elements such as disclosure, liability and non-repudiation, others only consider these measures to implement accountability. But while the means by which accountability is achieved in these domains differ, there is wide agreement on the overarching goal, i.e., entities such as organisations or individuals need to give account for actions and thus be rewardable or punishable for them (see, e.g., the survey by Papanikolaou and Pearson [PP11]). Within the scope of this work, we presume the ability to reward or to punish (in contrast to, e.g., Feigenbaum et al. [FJW11]) and thus regard accountability as the requirement for a protocol to provide sufficient account for such actions.

This has been widely recognized, but most work on this subject provides or uses a notion of accountability in either an informal way, or tailored to the

protocol and its security [ASW98,Kro15,BCS05,BFM13]. There has been work on general definitions, but no definition to date defines completeness, i.e., the property that all parties that deviate in a malicious way are blamed.

Defining completeness. The difficulty lies in distinguishing whether a deviation from the behaviour prescribed by the protocol was harmless, or may have caused a security violation. For many security properties, e.g., secure multi-party computation, a misbehaving party is abstracted by giving control to the adversary, equating "dishonest", or potentially deviating, with "behaving maliciously". Expressing soundness is oblivious to this approximation, e.g., Küsters et al. define soundness as follows: whenever the judge gives some verdict, every party mentioned is dishonest [KTV10]. But the approach falls flat when expressing completeness naïvely: if a party is dishonest, the judge should mention it in the verdict, but the adversary might decide to not deviate after all, or deviate in some way that cannot be recognised. Some approaches still follow this idea, considering any trace that cannot be produced by the honest protocol malicious behavior [HKD07,JJPR09]. For distributed systems, which classically aimed at masking faults, systems like PeerReview [HKD07] can *detect* faults in the Byzantine setting. Our corruption model is also Byzantine, however, we work in an adversarial setting and we do not assume that a complete view of every component or all the communication is available. Even if all communication is public – which is unrealistic considering possible off-band communication – any accountable protocol would be required to verify the behaviour of every party in full detail, even if this party is not involved in later phases (e.g., registration authorities in e-voting protocols) or cannot meaningfully disrupt communication. Consequently, Haeberlen et al. focus on faults in distributed systems [HKD07], where the component's adherence to specification is a design goal, while Jagadeesan et al. admit that in their model "the only auditor capable of providing [completeness] is one which blames all principals who are capable of dishonesty, regardless of whether they acted dishonestly or not" [JJPR09].

Policies for accountability. There is a middle-ground between protocol-specific and fully protocol-agnostic definitions, where accountability is defined w.r.t. a policy. These policies, however, cannot reliably express completeness in a protocol-agnostic way. This is discussed in detail in the long version [Ano17].

In the following, we show that such policies sometimes have to depend on the specific protocol to be useful, hence they are not truly protocol agnostic. Küsters et al. propose policies of the form $\alpha \implies p_1 \mid \cdots \mid p_n$ and define completeness as follows: if a trace matches α, the verdict should imply some p_i. Consider the scenario where each of the two parties A and B might violate a security property by deviating on its own, and assume the log always provides indication that this is the case. We would like to express that A and B shall be held accountable in case both deviate, as each deviation on their own would entail a violation. Let α match traces with said security violation. How could such a simple policy be expressed?

- $\alpha \implies A$ is too weak, as B's participation is disregarded in case only B deviates (and unfair towards A). Same for $\alpha \implies B$.
- $\alpha \implies A \wedge B$ is unfair to A or B in cases only one of them deviates.
- $\alpha \implies A \vee B$ permits uncertainty in the verdict as it would suffice to blame $A \vee B$ in case either deviates, but as A and B's behavior is visible, this policy is unnecessarily weak.
- $\alpha \implies A \mid B$: In this case, the verdict needs to imply either A or B, hence a verdict A would suffice even if both deviate. The same holds for the policy $\alpha \implies A \mid B \mid A \wedge B$.

Hence the only choice is to split α into two formulas, α_A and α_B, which capture traces where A, respectively B, misbehaves. Then α_A and α_B may intersect in case both deviate and $\{\alpha_A \implies A, \alpha_B \implies B\}$ constitutes a policy that enforces actual completeness.

But at this point, the policy does not serve as a specification anymore – α_A and α_B describe the accountability mechanism itself, not the security property. For one, this means that the policy is protocol specific, i.e., not applicable to a class of protocols, e.g., all voting protocols, anymore. But most importantly, the policy validates the accountability mechanism with itself, hence the approach begs the problem.

Accountability from causal relations between protocol events The remaining approaches in the literature focus on protocol actions as causes for security violations [GM15,DGK+15,FJW11]. The main issue with the approach is that not all protocol actions that are related to an attack are malicious, e.g, a key-server distributing public keys used in the attack. While protocol actions as causes, i.e., cause traces, may help in filtering out parties that were altogether uninvolved in the attack, they nonetheless refer us back to the original question: What constitutes malicious misbehaviour? For this reason, none of the existing works formalising causality on protocol events arrive at answering this question. Datta et al. propose cause traces as a building block and sketch a procedure for blame assignment, but there the key-server would be blamed unless he followed the protocol to the letter [DGK+15]. First, this is overly strict, as, e.g., a slight deviation in the format of certificates would lead to the key-server being blamed even if this deviation is not causally related to the violation. Second, in cases where private communication is possible, the precise messages the key-server sent out might be unknown, and thus it would be unclear whether the server needs to be blamed. The same problem affects Feigenbaum et al.'s line of work, where causation on protocol events is used as a black-box. Their work focuses, however, on punishment (possibly protecting the parties' identities), a question we are currently not treating [FJW11]. Gössler et al. focus on traces and assume an explicit dependency relation to recognize faulty components in a system [GM15]. Due to this focus, the case of parties colluding privately is excluded from the start.

Summarizing, existing approaches are either protocol specific, or require the slightest deviation to be punished, even if the normative behaviour would have

had the same causal dependencies. In scenarios where parties can communicate privately, however, this approach reaches its limits.

3 Process calculus

In this section, we introduce a process calculus we use to discuss accountability in security protocols. It draws heavily from the applied-π calculus [AF01], with the difference that it incorporates the effectuating party of a process within the calculus. We describe a protocol in terms of the overall structure of the process, which determines which parties run in parallel and which parties share secrets, and the *normative behaviour*, which determines which process each party runs. Any party may choose to run a different process, which we call a *deviation*. We will first introduce the term algebra and the process calculus (closely following the applied-π calculus) before we specify how parties might deviate.

Notation Let $\mathbb{N}_n = \{0, .., n\}$. For two functions f and g, $f[g]$ denotes the function mapping any $x \in dom(g)$ to $g(x)$, and any $x \in dom(f) \setminus dom(g)$ to $f(x)$. For $l = (e_1, \ldots, e_n)$ and $1 \leq i \leq n$, we write $l|_i$ to refer to e_i and $l' \leq l$ if l' is a prefix of l. We denote domain restriction of a function f to a subset $S \subseteq dom(f)$ as $f|_S$. We filter a sequence l by a set S, denoted $l|_S$, by removing each element that is not in S, and by a partial function $\pi \colon S \rightharpoonup T$, denoted $l|_\pi$, by computing $l|_\pi = (s_1, \ldots, s_{n-1})|_\pi \cdot \pi(s_n)$ or $(s_1, \ldots, s_{n-1})|_\pi$, if $\pi(s_n)$ undefined.

Terms and equational theories We model messages by abstract terms. Assume a countably infinite set of names $\mathcal{X}$ used to model cryptographic keys and nonces, and a countably infinite set of variables $\mathcal{V}$. Given a signature Σ, i.e., a set of function symbols with an arity each, we write f/n for function symbol f of arity n. Let Terms_Σ be the set of terms built over Σ, $\mathcal{X}$, and $\mathcal{V}$, and $names(t)$, respectively $vars(t)$, denote the set of names, respectively variables, appearing in a term t. The set of ground terms, i.e., terms without variables, is denoted by $\mathcal{M}_\Sigma$. When Σ is fixed or clear from the context we omit it and simply write Terms for Terms_Σ and $\mathcal{M}$ for $\mathcal{M}_\Sigma$.

We equip the term algebra with an equational theory E, i.e., a finite set of equations of the form $m = m'$ where $m, m' \in \text{Terms}$. From the equational theory we define the binary relation $=_E$ on terms, which is the smallest equivalence relation containing equations in E that is closed under application of function symbols, bijective renaming of names and substitution of variables by terms of the same sort.

Example 1. The signature $\Sigma_{\text{sig}} := \{true/0, pk/1, sig/2, versig/3, extract/1\}$ and the following equational theory model digital signatures: $extract(sig(x, y)) = y$ and $versig(pk(x), sig(x, y), y) = true$. As usual, $extract/1$ over-approximates the capacity of parties to learn the value of the signed message, however, realistic protocols should not rely on this function symbol, given that many implementations of digital signatures do not provide this feature.

From now on, assume that E refers to some fixed equational theory and that the signature and equational theory always contain symbols and equations for pairing and projection, i.e., $\Sigma_{\text{pairs}} := \{\langle .,. \rangle, \pi_1, \pi_2\} \subseteq \Sigma$ and equations $\pi_1(\langle x, y\rangle) = x$ and $\pi_2(\langle x, y\rangle) = y$ are in E . We will sometimes use $\langle x_1, x_2, \ldots, x_n\rangle$ as a shortcut for $\langle x_1, \langle x_2, \langle \ldots, \langle x_{n-1}, x_n\rangle \ldots\rangle$. Set membership modulo E is denoted by $\in_E$ and defined as $e \in_E S$ iff $\exists e' \in S.\ e' =_E e$. A substitution σ is a partial function from variables x_i to terms t_i written $\sigma = \{^{t_1}/_{x_1}, \ldots, ^{t_n}/_{x_n}\}$. We homomorphically extend σ to apply to terms and use a postfix notation to denote its application, e.g., $t\sigma$ applies σ to t.

$\langle P,Q\rangle$ ::= (plain processes)	$\langle A,B\rangle$ ::= (extended process)
0	$A \mid B$
$\nu n; P$	$\nu n; P$
$\text{in}(x);\ P$	$\nu x; P$
$\text{out}(m);\ P$	$\{^m/_x\}$
if $m = m'$ then P else Q	$\cdot_p$ for $p \in \mathcal{P}$
event $t;\ P$	P

Fig. 1. Syntax, where $n \in \mathcal{X}$, $x \in \mathcal{V}$ and $m, m' \in$ Terms

Grammar and operational semantics In contrast to the applied-π calculus [AF01], this calculus incorporates the effectuating party of a process. Since processes running in parallel can represent threads or programs running in parallel as well as computers in an asynchronous network, this annotation is necessary. Assume a set of parties $\mathcal{P}$, a subset of which, $\mathcal{T} \subset \mathcal{P}$, is trusted to never deviate from normative behaviour.

Plain processes (defined by the grammar in Figure 1) usually define the behaviour of a single party as a combination of message input and out on a public channel, conditionals w.r.t. $=_E$ and scope restriction ($\nu n; P$) which creates a fresh name n and then behaves like P. Furthermore, plain processes are able to emit events, which can be used to model signalling behaviour (see Example 2 below) as well as append-only logs (see Example 11). We will omit else-branches with zero processes for brevity. We exclude parallel composition in plain processes and any kind of replication for simplicity, as otherwise we would need to track several processes per party. This could be easily achieved, e.g., by drawing fresh names as session identifiers [Bla02]. Extended processes combine extended or plain processes via parallel composition ($A \mid B$). Scope restrictions on names can be used to distribute shared secrets. An active substitution $\{^m/_x\}$ acts as a "floating" substitution operation and can, in combination with scope restriction on variables, be used to transmit terms between processes (see below). As usual, names and variables have scopes delimited by restrictions and by inputs. We write $fv(A)$, $fn(A)$ for the set of free variables and names of A, respectively. Finally, the "hole" operator $\cdot_p$ serves as a placeholder for the normative behavior of a party $p \in \mathcal{P}$ or its eventual deviations. We thus require the following condition for the well-formedness w.r.t. deviations.

Definition 1 (skeleton process). *A* skeleton process *is defined by the grammar for extended processes without the last production rule (which includes plain processes) and at most one hole $\cdot_p$ per party $p \in \mathcal{P}$.*

A protocol is now defined in terms of a skeleton process determining how information, i.e., names and terms, are initially shared between parties, and a function that maps every party to a plain process.

Definition 2 (protocol). *A protocol $\pi = (A, n)$ consists of a skeleton process A such that for every $p \in \mathcal{P}$, there is exactly one occurrence of the hole operator $\cdot_p$ in A, and a function n from $\mathcal{P}$ to plain processes, such that for all $p \in \mathcal{P}$, all $fn(n(p))$ and $fv(n(p))$ are bound in the scope of $\cdot_p$ in A. We call $n(p)$ p's* normative behaviour*.*

A straightforward approach to achieve accountability is to have a trusted monitor, which executes requests but only accepts signed requests. The next example follows this paradigm. We use it to explain the semantics of our calculus, but will show later that this particular monitor fails to provide accountability.

Example 2 (delegation example, broken). Assume the signature Σ consisting of Σ_{pairs}, Σ_{sig}, $\Sigma_{\text{log}} = \{Log/2, Exec/1\}$ and $\Sigma_{\text{act}} = \{NAct/0, SAct/0, UnAct/0, isAct/1\}$, along with the equation $isAct(a) = true$ for a either $NAct$, $SAct$ or $UnAct$, and the equations in Example 1. Assume four parties, A, B, I and T; among those only $\mathcal{T} = \{T\}$ is trusted. The following skeleton process defines the generation of A and B's signing keys and the distribution of their public parts:

$$\nu sk_A; \nu sk_B; \{^{pk(sk_A)}/_{pk_A}, ^{pk(sk_B)}/_{pk_B}\}; \cdot_A \mid \cdot_B \mid \cdot_I \mid \cdot_T.$$

The party A processes two kinds of actions, normal actions and special actions. Normal action are signed and sent to T for execution. Special actions are forwarded to B for authorisation, which signs the request identifier n_a. Finally, A sends both signatures T for processing.

$$\begin{aligned}
n(A) := \mathsf{in}(a);\ &\mathsf{if}\ a = NAct\ \mathsf{then}\ \mathsf{out}(\langle a, sig(sk_A, a)\rangle)\\
&\mathsf{else}\ \ \mathsf{if}\ \ a = SAct\ \mathsf{then}\ \nu n_a;\ \mathsf{out}(\langle n_a, sig(sk_A, \langle a, n_a\rangle)\rangle);\\
&\quad \mathsf{in}(r);\ \mathsf{if}\ \ versig(pk_B, r, n_a) = true\ \mathsf{then}\ \mathsf{out}(\langle a, n_a, sig(sk_A, \langle a, n_a\rangle), r\rangle)\\
n(B) := \mathsf{in}(m);\ &\mathsf{if}\ versig(pk_A, \pi_2(m), \langle SAct, \pi_1(m)\rangle) = true\\
&\quad \mathsf{then}\ \mathsf{out}(sig(sk_B, \pi_1(m)))
\end{aligned}$$

The trusted monitor T verifies the signature of incoming requests, but not which action they contain. But it takes note of the responsibility to later identify who request, e.g., an unusuaul action. This strategy is typical in scenarios where security violations may only be determined after the fact, e.g., if T cannot tell usual from unusual actions. For example, in a hospital, it may be unusual for a doctor (A) to retrieve data belonging to another doctor's (B's) patient. Still, in case of an emergency, B should be able to request this special action without further ado, but has to justify it later. The responsibility of T is to enforce that the necessary information is present. Let xa, xn, s_1, s_2 in the second branch abbreviate $xa = \pi_1(m)$, $xn = \pi_1(\pi_2(m))$, $s_1 = \pi_1(\pi_2(\pi_2(m)))$ etc. such that $m = \langle xa, xn, s_1, s_2\rangle$. Then,

$n(T) := \mathsf{in}(m);\ \mathsf{if}\ \mathit{versig}(pk_A, \pi_2(m), \pi_1(m)) = \mathit{true}\ \mathsf{then}$
$\quad \mathsf{event}\ \mathit{Log}(A, \pi_1(m));\ \mathsf{event}\ \mathit{Exec}(\pi_1(m))$
$\quad \mathsf{else}\ \ \mathsf{if}\ \ \mathit{versig}(pk_A, s_1, \langle xa, xn\rangle) = \mathit{true}\ \mathsf{then}$
$\quad\quad \mathsf{if}\ \ \mathit{versig}(pk_B, s_2, xn) = \mathit{true}\ \mathsf{then}\ \mathsf{event}\ \mathit{Log}(\langle A, B\rangle, xa);\ \mathsf{event}\ \mathit{Exec}(xa)$

The party I with $n(I) := \mathsf{out}\ (\mathit{SAct});\ \mathsf{out}\ (\langle pk_A, pk_B\rangle)$ models an intruder who knows A's and B's public key and triggers some communication between A and B leading to a normal run.

$$
\begin{array}{ll}
\textsc{Then} & (\mathsf{if}\ t_1 = t_2\ \mathsf{then}\ P\ \mathsf{else}\ Q)_{p_A} \xrightarrow{p_A} P_{p_A} \quad \text{if } t_1 =_E t_2 \\
\textsc{Else} & (\mathsf{if}\ t_1 = t_2\ \mathsf{then}\ P\ \mathsf{else}\ Q)_{p_A} \xrightarrow{p_A} Q_{p_A} \quad \text{if } t_1 \neq_E t_2 \\
\textsc{Comm} & (\mathsf{out}(x); P)_{p_A} \mid (\mathsf{in}(x); Q)_{p_B} \xrightarrow{(p_A, p_B, x)} P_{p_A} \mid Q_{p_B} \\
\textsc{Event} & (\mathsf{event}\ m; P)_{p_A} \xrightarrow{(p_A, m)} P_{p_A}
\end{array}
$$

Fig. 2. Reduction rules

The operational semantics are exactly the same as for the applied-π calculus, except that *a*) the top-most processes inserted at a hole operator $\cdot_p$ is annotated with the party p (skipping scope restrictions), and these annotations are preserved during reduction, and *b*) reductions are additionally labelled with the effectuating party for internal reductions, and with sending party and recipient in case of communication (see Figure 2). Appendix A recalls the applied-pi calculus, including our modifications, in detail. We obtain the following notion of traces.

Definition 3 (Traces). *Let* $\mathit{traces}(A) = \{(l_1, \ldots, l_n) \in ((\mathcal{P} \times \mathcal{P} \times \text{Terms}) \uplus (\mathcal{P} \times \text{Terms}) \uplus \mathcal{P})^* \mid A \xrightarrow{l_1} \cdots \xrightarrow{l_n}\}$.

Whenever $\mathit{fv}(A) = \emptyset$, i.e., all variables in A are initially bound, then $\mathit{traces}(A) \subseteq ((\mathcal{P} \times \mathcal{P} \times \mathcal{M}) \uplus (\mathcal{P} \times \mathcal{M}) \uplus \mathcal{P})^*$. Given a set of parties $\mathcal{Q} \subseteq \mathcal{P}$, we use $\rightleftarrows \mathcal{Q} = (\mathcal{Q} \times \mathcal{P} \times \text{Terms}) \cup (\mathcal{P} \times \mathcal{Q} \times \text{Terms}) \cup (\mathcal{Q} \times \text{Terms})$ to filter out everything except the communication with and the events emitted by members in $\mathcal{Q}$ from a trace.

Deviations A deviation is a function that overwrites the normative behaviour of untrusted parties and may only refer to names and variables the normative behaviour uses, e.g., a deviation for A in Example 2 may refer to pk_B, which is bound to $pk(sk_B)$, but not sk_B.

Definition 4 (protocol deviation / instance). *For a protocol* $\pi = (A, n)$*, any partial function* d *from* $\mathcal{P} \setminus \mathcal{T}$ *to plain processes such that* $\mathit{fv}(d(p)) \subseteq \mathit{fv}(n(p))$ *and* $\mathit{fn}(d(p)) \subseteq \mathit{fn}(n(p))$ *for every* $p \in \mathit{dom}(d)$ *is called a* deviation *and induces and instance of* π*,* $\pi[d]$*, where each occurrence of* $\cdot_p$ *is substituted by* $d(p)$*, if*

defined, and $n(p)$ otherwise. In both cases, the top-most process which is not of the form $\nu m; P$ within this subprocess is annotated with p.

For the empty deviation $\emptyset$, the instance $\pi[\emptyset]$ is an extended process modelling characterized by the normative behaviour of all parties. By Definition 2, the free names and variables of all plain processes in the domain of n are bound in the scope of $\cdot_p$ in A, hence any instance of the protocol is closed. We remove the non-determinism in the reduction of an extended process A and capture the order in which parties act, by means of a *context* $u \in ((\mathcal{P} \times \mathcal{P}) \uplus \mathcal{P})^*$. Let $\mathit{traces}(A, u)$ be the set of $t \in \mathit{traces}(A)$ such that $u = t|_\gamma$, where γ maps the reduction of conditionals and events to the executing party, i.e., $p_A \mapsto p_A$ and $(p_A, t) \mapsto (p_A)$, and communication to the sender and recipient of the message, i.e., $(p_A, p_B, m) \mapsto (p_A, p_B)$. This removes all non-determinism.

Lemma 1. *For any protocol $\pi = (A, n)$, deviation d, and context u, $\mathit{traces}(\pi[d], u)$ is either empty or singleton modulo E.*

Proof. W.l.o.g., let any chain of reduction $A_0 \xrightarrow{l_1} A_1 \xrightarrow{l_2} \cdots \xrightarrow{l_n} A_n$ be such that each variable x in any transition $A_i \xrightarrow{(p_A, p_B, x)} A_{i+1}$ is unique and follows some fixed order — this can always be achieved using Subst. We show the claim by showing the following stronger property: There is a context u' and a substitution σ such that for all $A_0 \xrightarrow{l_1} A_1 \xrightarrow{l_2} \cdots \xrightarrow{l_n} A_n$ such that $A_0 = \pi[d]$: and all $i \in \{0, \ldots, n\}$,

a) $P_i \equiv \nu\vec{n}.Q_1 \mid \cdots \mid Q_k|\{^{\vec{m}}/_{\vec{x}}\}$ where all Q_i, $i \in \mathbb{N}_k$, are annotated with a unique party p, $\mathit{fv}(Q_i) \subseteq \vec{x}$, $\mathit{fv}(Q_i) \subseteq \vec{n}$ and $\{^{\vec{m}}/_{\vec{x}}\} \subseteq_E \sigma$.
b) $l_i|_\gamma = u_i$,

Induction on n. In the initial case, *a)* follows from the fact that the holes in skeleton processes are unique and by application of New-P and Subst for renaming variables and moving the ν to the top. Given this structure, *b)* holds as the rules Then, Else, Comm, and Event are mutually exclusive and each carry a different label, i.e., whichever rule applies determines l_1, or l_i respectively.

Inductive step: *a)* holds because each reduction rule preserves this structure and since any variable x transmitted either was previously part of the substitution, or a new substitution was created by Alias, in which case the substitution maps x to some message m with $\mathit{fv}(m) \subseteq \mathit{vars}(\sigma)$ by induction hypothesis. This substitution can only be rewritten by Rewr. As E is closed under application of function symbols, this implies that the new variable always maps to the same message, modulo E. The argument for *b)* is identical to the argument in the base case. □

Hence $\mathit{trace}(\pi[d], u) := t$ s.t. $t \in \mathit{traces}(\pi[d], u)$ is a well-defined partial function. We define the *log* of a trace as the events emitted, e.g., $\mathit{log}(t) = t|_{\mathcal{P} \times \mathrm{Terms}}$ for a trace t, and define security in terms of properties on logs $\varphi : (\mathcal{P} \times \mathrm{Terms})^* \to \{0, 1\}$. We write $\varphi(t)$ for $\varphi(\mathit{log}(t))$ and require properties to be congruent w.r.t. E, i.e. for all t, t', $t' =_E t \implies \varphi(t) = \varphi(t')$, so $\varphi(\mathit{trace}(\pi[d], u))$ is well defined.

Definition 5 (satisfaction). *A protocol π with deviations d satisfies φ in context u, written $(\pi, u) \vDash [d]\varphi$, if $\varphi(trace(\pi[d], u))$ is defined and equals* 1.

Example 3. The delegation protocol (Ex. 2) shall hold parties accountable for effectuating actions that are neither normal nor special, i.e., for violating $\varphi(t) := \forall e \in t.\ e =_E Exec(a) \implies a \in_E \{NAct, SAct\}$.

4 Accountability

In this section, we formalise the notion of a *verdict*, which models the findings of a judge, a jury or a similar entity on the question "Which parties caused the misbehaviour (jointly or on their own)?". We then define the *a posteriori verdict*, which formalises which parties *should* be held accountable, if we know the processes executed by these parties are known a posteriori. The a posteriori verdict forms the basis for our later treatment of unknown deviations: Ideally, an accountability mechanism should always give a verdict that coincides with the a posteriori judgement. But if "always" means "for all deviations and contexts", then many protocols do not provide accountability, unless they make the unrealistic assumption that there are no private communication channels between untrusted parties.

We recover applicability by weakening the requirement to correctness w.r.t. short, rational, or in some other way optimal deviations that produce a given trace. We discuss two such notions and their respective limitations, and present our notion of accountability, which is parametric in the optimality notion used. This approach preserves soundness, but gives up completeness to some extent. However, we gain a clear understanding of which deviations are disregarded.

4.1 Verdict

In many scenarios, only parts of the trace are observable or considered trusted, consequently we may only consider a subset of the observations, $V \subseteq (\mathcal{P} \times \mathcal{P} \times \text{Terms}) \uplus (\mathcal{P} \times \text{Terms}) \uplus \mathcal{P}$. Given a protocol π, $traces(\pi[\emptyset])|_V$ denotes the visible projections of the traces of this protocol.

Definition 6 (verdict). *Given a protocol π and V, a verdict is a set of subsets of $\mathcal{P}$. It is derived from a trace using a total function verdict : $traces(\pi)|_V \to 2^{2^{\mathcal{P}}}$ congruent w.r.t. E. We sometimes write a verdict v as a propositional formula with parties as atoms of the form $\bigvee_{C \in v} \left(\bigwedge_{p \in C} p \right)$ to clarify its semantics.*

The verdict provides the set of *independently* accountable groups of agents, such that all agents within such a group are *jointly* accountable. For example, if a verdict $(A \wedge B) \vee (B \wedge C) \vee (A \wedge C)$ is given, the way any two of the parties A, B, C actually deviated was sufficient to provoke the violation, e.g., when a simple majority was sufficient to accept a faulty input, and all three did in fact misbehave. We discuss the choice of sufficient causation in Section 4.3.

The verdict function abstracts whatever entity is giving the verdict, be it an actual judge or jury, or a designated party which is part of the protocol. In the latter case, the verdict function would only take events emitted by this designated party into account. As *verdict* is total on $traces(P)|_V$ and congruent w.r.t. E, we abbreviate $verdict(trace(\pi[d], u)|_V)$ by $verdict(\pi[d], u, V)$.

Example 4 (Verdict for Example 2). In Example 2, the task of the monitor is to provide sufficient evidence of the parties deviating from protocol by issuing an unusual action. We thus assume only the events of the trusted party to be visible, $V = \{T\} \times \text{Terms}$, and consider the verdict function $verdict(t) :=$

$$\begin{cases} \{\{A, B\}\} & \text{if } Exec(a), Log(\langle A, B\rangle, a) \in t \wedge a \notin \{SAct, NAct\}, \\ \{\{A\}\} & Exec(a), Log(A, a) \in t \wedge a \neq NAct, \\ \emptyset & \text{otherwise.} \end{cases}$$

This example shows how events can provide for an append-only log, which in Example 2 is used by a trusted party. Example 11 will show that even untrusted parties can emit events, in which case their log may, however, not be trustworthy and thus the protocol design requires more sophistication. In the present example, a judge inspects the visible part of the trace, i.e., T's log, after the protocol run. A slightly different protocol could use the same conditions to derive a verdict while the protocol is running, in which case the verdict function would only interpret the corresponding events emitted, e.g., by T. The use of events carries the explicit assumption that this specific log can only be written to by T, that it is append-only and that it is never disputed. Even in the case where untrusted parties emit events, the semantics guarantee at least the append-only property.

Küsters, Truderung and Vogt pointed out the relationship between verifiability and accountability by relating the verdict to whether the security property was violated in the same run [KTV10].

Definition 7 (verifiability). *A verdict function verdict provides a protocol $\pi = (A, n)$ with verifiability for a property φ and visibility set V, if for any deviation d and $t \in traces(\pi[d])$, $\varphi(t) \iff verdict(t|_V) = \emptyset$.*

4.2 A posteriori verdict

The a posteriori verdict defines which parties should be held accountable (jointly or independently), given that the processes executed by these parties are known a posteriori, e.g., if forensic findings determine the program they ran. Given a deviation and a context, we consider a run of the protocol with only parts of this deviation by reverting some parties to their normative behaviour. For any set of parties S claimed to be sufficient to cause a violation of security property φ, it must be the case that reverting all *other* parties to their normative behaviour, i.e., restricting d to S, indeed results in a violation, i.e., $(\pi, u') \vDash [d|_S]\neg\varphi$ for an appropriately chosen context u'. The intuition is that the deviation of these parties alone was sufficient to ensure the violation takes place. We are interested

in minimal such sets, in order to capture independently accountable groups of agents. E.g., if either of A and B's deviation suffices to cause a violation, we want the verdict to be $\{\{A\},\{B\}\}$ (read as $A \vee B$).

Definition 8 (a posteriori verdict). *Given a protocol $\pi = (A, n)$, a property φ, a context u and a deviation d, the* a posteriori verdict $apv_{\varphi}(\pi, u, d)$ *is defined*

$$\{\mathrm{S} \mid (\pi, u) \vDash \neg\varphi \text{ and } \mathrm{S} \text{ is minimal s.t. } \forall \mathrm{S}' \supseteq \mathrm{S}. \exists u' : u' =_{\mathrm{S}'} u \wedge (\pi, u') \vDash [d|_{\mathrm{S}'}]\neg\varphi\}\,.$$

Here, $u' =_{\mathrm{S}'} u$ is short for $u'|_{\rightleftharpoons \mathrm{S}'} = u|_{\rightleftharpoons \mathrm{S}'}$, with $\rightleftharpoons \mathrm{S}' := (\mathrm{S}' \times \mathcal{P}) \cup (\mathcal{P} \times \mathrm{S}') \cup (\mathrm{S}')$.

As deviations can have a different structure from the normative behaviour, the context u', which controls the scheduling, allows parties that are reverted to their normative behaviour to behave differently. In contrast, the parties not reverted, e.g., those in S', are bound to their previous scheduling $u|_{\rightleftharpoons \mathrm{S}'}$. We discuss the quantification over all S' in the next section.

Example 5. Consider a deviation d for Example 2 with $d(B) := \mathsf{out}(sk_B)$ and $d(A)$ defined as follows:

$$\mathsf{in}(xsk);\ \nu n_a;\ \mathsf{out}(\langle UnAct, n_a, sig(sk_A, \langle UnAct, n_a\rangle), sig(xsk, n_a)\rangle)$$

Within context $u = ((B, A), A, (A, T), T, T, T, T)$, B sends her signing key to A, who fakes B's authorisation and then instructs T to execute an unusual action. W.r.t. the property that no unusual action was executed, i.e., $\varphi(t) := Exec(UnAct) \notin_E t$, the a posteriori verdict $apv_{\varphi}(\pi, u, d)$ is $\{\{A, B\}\}$, as reverting either A or B to their normative behaviour avoids φ. In contrast, if $d(A) := \mathsf{out}(sig(sk_A, UnAct))$ and B shares its signing key with I, i.e., $d(B) := \mathsf{out}(sk_B)$ and $d(I) := \mathsf{in}(m)$, then $apv_{\varphi}(\pi, u, d) = \{A\}$, for the obvious context u, as B's behaviour, even if it was reckless, had no bearing on the coming about of φ.

4.3 Discussion

In this section, we discuss different aspects of the definition and motivate the decisions we made with distinguishing examples.

Relation to causation Similar to the structural-model approach to causation [HP13], we determine causal relations between deviating from the normative behaviour and the occurrence of a violation by intervening on potential causal factors, in our case: the fact whether a party deviates at all. A host of recent definitions of actual causation follows this approach [HP13,Hal15] going back to Lewis [Lew73] and possibly Hume [Hum, Section VII]. These definitions all follow the idea that A causes B, if B would not have happened, had it not been for A, i.e., intervening in a way that removes A results in B. This kind of intervention is reflected within the deviation d and the manipulation thereof. Still, this causation defines A as a necessary cause, also known as *condicio sine qua non*. Recent work, however, points out that parts of necessary causes lack a clear interpretation, whereas sufficient causes *a*) are dual to necessary causes and *b*) capture joint causation directly [BGK17]. A is a sufficient cause for B, if

intervening on every other object beside A still leads to B in any case. Furthermore, as with necessary causes, minimality is required. The only intervention we regard here is restriction of the domain of d, i.e., reverting the deviation of some party to its normative behaviour. Hence, the a posteriori verdict equal to the set of deviations that are sufficient causes for φ with respect to the mentioned notion of intervention, with one difference: sufficiency is tested for all supersets of S, i.e., all subsets of $\mathcal{P} \setminus \mathrm{S}$ are reverted to their normative behaviour.

Coordination. Definition 8 requires a violation to occur for all supersets S' of S instead of just S. This is motivated by the following example.

Example 6 (coordination attack). Let $\mathcal{P} \setminus \mathcal{T} = \{A, B, C\}$. Assume C could mount an attack, but $n(C)$ prescribes to not do so. Both $n(A)$ and $n(B)$ send 0 to C, which is ignored by $n(C)$. Consider a deviation d where $d(C)$ attacks if A's and B's message are equal, and $d(A)$ and $d(B)$ both send 1. The attack is part of the actual trace, disappears if any of them is reverted to their normative behaviour, but is present if *both* are reverted. Hence there is a causal relation between $d(A)$ sending 1 and $d(C)$ mounting the attack. As, e.g., $d|_{\{A,C\}}$ does not cause the attack, the a posteriori verdict is $\{\{A, B, C\}\}$.

Some readers might have a different intuition about this example, as the attack would still have happened had A and B behaved normally. As Pearl put it: "causal explanation is a man-made concept" [Pea00]. If this is desired, the pragmatic solution is to only regard $\mathrm{S}' = \mathrm{S}$.

Choosing the counterfactual context The context u' for each counterfactual $d|_{\mathrm{S}'}$ is chosen existentially, but has to match u for the parties that are not reverted to their normative behaviour, enforcing that they behave the same and that messages they send reach the same party. This permits deviating parties to "talk behind the backs" of the other parties, however, if in u they address a party outside S, this party has to receive *some* message.

Consider the case where A and B are not supposed to do anything, but deviate as follows: A corresponds with B, and then mounts an attack on C. B would be part of a verdict merely for the fact that A waits for a response, which is acceptable. Consider, however, the case where A sends a message to B before mounting the attack, but does not wait for B's response. Here B would be part of the verdict $\{\{A, B\}\}$, as $n(B)$ would never pick up the message sent by A, and thus there would be no $u' =_{\mathrm{S}'} u$ for $\mathrm{S}' = \{A\}$ that contains (A, B). This is clearly an artefact of the semantics of our protocol calculus, however, similar to the applied π-calculus, the network attacker should be regarded as an evaluation context or a party running in parallel to the protocol as, e.g., represented by I in Example 2. For the purpose of automated verification, the operational semantics of the applied π-calculus were later streamlined and complemented by verification methods that consider an arbitrary network attacker by default [Bla09]; we expect something similar in future work. For now, we prefer to model direct communication for generality and clarity of presentation.

4.4 Accounting for hidden deviations

We extend the definition introduced in the previous section to cases where the deviation of some parties may remain unknown. This gives rise to a definition of accountability as a meta property of a protocol. We say that a protocol provides accountability for a property, if there is a verdict for this property, and this verdict is always correct w.r.t. the a posteriori verdict. But if "always" means "for all deviations and contexts", then many protocols do not provide accountability, unless they make the unrealistic assumption that there are no private communication channels between untrusted parties.

We recover applicability by weakening the requirement to correctness w.r.t. short, rational, or in some other way optimal deviations that produce a given trace. We discuss two such notions and their respective limitations, and present our notion of accountability, which is parametric in the optimality notion used. This approach preserves soundness, but gives up completeness to some extent. However, we gain a clear understanding of which deviations are disregarded.

The trouble with provocations One would expect that accountability for φ can be captured by requiring the actual verdict to coincide with the a posteriori verdict for $\neg\varphi$ for all d and u.

Example 7. Consider a deviation d with $dom(d) = \{A, B\}$ and $d(A)$ as follows:

$\mathsf{in}(m)$; if $m = \mathit{Hello}$ then *behave maliciously, violating* φ else *behave honestly*

and $d(B)$ is out (Hello). Then $(\pi, u) \vDash [d]\neg\varphi$, but $(\pi, u) \vDash [d|_{\{A\}}]\varphi$, thus A alone is not sufficient to cause $\neg\varphi$, hence $\{A, B\}$ is minimal.

Now assume there is no way to discover whether this communication between B and A took place – it is impossible to determine whether A was running $d(A)$ or some d' with $dom(d') = \{A\}$, mounting the attack herself. Hence A can plausibly discredit B.

This problem arises whenever private communication with a potentially provoking party is possible, hence, unless one were willing to assume all communication was public, no protocol could provide accountability. Note that this problem would also arise if Definition 8 took necessary causation rather than sufficient causation as a basis. Instead, we chose to only regard deviations which are *a*) plausible regarding the observed trace and *b*) optimal. Obviously there is a risk that the guarantees provided are weakened if the optimality notion is too strong. But consider the following two arguments for weakening accountability in this respect: First, we want the definition to be practically applicable; a definition that applies to no real protocol is useless, no matter how appealing it is. Example 7 is not an artefact of our definition: intuition requires that both A and B be held accountable for the first deviation, but only A for the second.

Second, if we step back from the idea that an accountable protocol needs to blame all accountable parties, but rather consider accountability as the capability of supplying enough information to a judge to do so, provocation loses its scare.

In Example 7, if the protocol would only blame A, but A can somehow prove (e.g., by revealing d) that B was implicated, the judge would take this claim into consideration. However, as nothing points to B, A deviating alone provides the simplest explanation consistent with the observation, and is plausible in the sense that A was indeed able to mount exactly the attack observed. There is no uncertainty that A was deviating in a way enabling the attack, and could have done with or without provocation. Thus it makes sense to only require the verdict to point to A — if A acted on provocation, e.g., A was part of a bot-net, A needs to provide evidence that this was the case. As we will see, we sacrifice completeness for applicability, but we can do so in a controlled manner.

Note that provocation is *not* simply a matter of excluding publicly derivable messages, such as the message used for provocation, from consideration. If that were the case, the problem could be solved by considering an adversarial context C, running in parallel to A and B, that could input messages deducible from the observable output, i.e., a network adversary. In this (potential, but failing) solution, we would always consider an arbitrary such context C, but not consider it a party and thus never include it in a verdict. This would solve the provocation problem, as the context itself could trigger A's malicious behaviour and thus reflect that B's input was not necessary to produce the observed message.

Nevertheless, this solution is not adequate in a different scenario where A exploits a 'confused deputy' B, i.e., B is trusted but can be misused to mount an attack. While C may exploit B just like A did, A is in fact the party to be held accountable in this case. Hence, we really do care about optimality of deviations, which is the subject of Section 4.5.

Accountability We provide a definition of accountability which specifies the correctness of a verdict function, i.e., its soundness and completeness, and takes a posteriori knowledge about the actual deviation d_p into account. This a posteriori knowledge models certainty about the way some parties deviated, it may however be incomplete or even empty. Correctness should hold for any visible trace produced by the protocol, but there are many deviations and contexts that produce a given trace. Nevertheless, an accountable protocol needs to provide a verdict with certainty. These are conflicting goals, which we resolve by only considering optimal deviations. In Section 4.5, we discuss several notions of optimality. For the moment, we will just require this notion of optimality to be *sane*, i.e., for all deviations d',d_p and every visible trace $t_V \in \mathit{trace}(\pi[d'[d_p]])$, every optimal deviation d and context u has w.r.t. d_p, $t_V = \mathit{trace}(\pi[d[d_p]], u)|_V$, i.e., it *explains* t_V. Furthermore, there is at least one optimal deviation and context for each such t_V. Then, the idea is the following: no matter which trace t we observe, the verdict that the accountability mechanism derives from the visible part $t|_V$ equals the a posteriori verdict, for *any* optimal deviations (and contexts) explaining $t|_V$.

Definition 9 (accountability). *A function* $\mathit{verdict}_{d_p} : \mathit{traces}(\pi)|_V \to 2^{2^P}$ *provides a protocol* $\pi = (A, n)$ *with accountability for a property* φ *assuming a posteriori knowledge of some deviation* d_p*, if for any trace* $t \in \mathit{traces}(\pi[d'[d_p]])$

(for some d'), and any optimal d, u explaining $t|_V$ w.r.t. d_p, $verdict_{d_p}(t|_V) = apv_\varphi(\pi, u, d[d_p])$.

We define optimality in Section 4.5. In the common case where all deviations are unknown, d_p is the empty map and thus $d[d_p] = d$.

If we consider the weakest sane optimality notion, e.g., all all deviations reproducing $t|_V$ are optimal, then this notion, of course, suffers from the provocation problem: As *verdict* is a function on $t|_V$, all optimal deviations explaining t_V need to have the same a posteriori verdict. Hence, independent of the verdict function, should V permit unobservable communication, then both deviations d and d' from Example 7 explain the same observation, but have different a posteriori verdicts, and therefore, no verdict function can provide accountability.

Discussion We relay the discussion of our choice of the optimality notion to the next section and discuss other aspects in the following.

Explicit verdict. We define accountability with respect to a verdict function. First, a functioning accountability mechanism needs the verdict function to be explicit. Second, the verdict function might be constrained to be efficiently computable, to be derivable from information that can be made public etc. Hence, different verdict functions may be appropriate for different contexts.

Enforcing certainty. Definition 9 obtains certainty from the fact that *verdict* is a function on traces. Hence all optimal deviations reproducing the visible part of the trace need to have the same a posteriori verdict. As the optimal deviations and contexts considered are not necessarily including the actual deviation (which may be non-optimal), they are universally quantified to capture cases where the remaining uncertainty prohibits accountability. Optimality should be defined so that all non-optimal deviations should be implausible, or at least acceptable to be postponed.

Example 8 (Whodunit). S and A coordinate on some value a as follows: $n(S)$ chooses a and sends it to A; $n(A)$ receives this value. Both report it to a trusted party T; only messages sent to T are visible. In actuality, $d(A)$ sends an a^* different from S's reported choice to T. In this case, a verdict is impossible without additional information on S's and A's deviation, as two minimal scenarios are plausible: S deviated and sent a^* to A, but reported a, or A deviated and received a, but reported a^* to blame S.

Both deviations are optimal w.r.t. to the notions discussed below. Blaming S or A based on one or the other could well be incorrect, hence we require the verdict to coincide for all optimal explanations of a given trace. As both deviations lead to different a posteriori verdict, no verdict can coincide with both. We stress that this is different from the disjunction within the verdict: if the verdict is $\{\{S\}, \{A\}\}$, i.e., $S \vee A$, it means that both a deviation of S or a deviation of A on their own are sufficient to cause $\neg\varphi$, however, the optimal deviation that explains t requires both S and A to deviate.

Soundness (Fairness). Our definition preserves soundness. As our case studies demonstrate applicability, and previous definitions could not capture completeness in realistic scenarios, this means we strictly improve the state of the art.

Lemma 2 (Soundness/Fairness). *If, assuming a posteriori knowledge d_p, $verdict_{d_p}$ provides π with accountability for φ w.r.t. a sane notion of optimality, then for all deviations d and contexts u, if $\mathrm{S} \in verdict(\pi[d], u, V)$ and $p \in \mathrm{S}$ for some S and p, then $p \in dom(d)$ and $d(p) \neq n(p)$.*

Proof. If $d(p) = n(p)$ or $p \notin dom(d)$, then $(\pi, u') \vDash [d|_{\mathrm{S}'}]\neg\varphi$, implies $(\pi, u') \vDash [d|_{\mathrm{S}'\setminus\{p\}}]\neg\varphi$, for any u' and $\mathrm{S}' \supseteq \mathrm{S}$. Thus, no $\mathrm{S} \in apv_\varphi(\pi, u, d[d_p])$ contains p for any d and u due to the minimality requirement in Definition 8. As any $\mathrm{S} \in verdict$ needs to be contained in at least one a posteriori verdict by sanity, this implies the claim. □

4.5 Notions of optimality

Many deviations and contexts can produce the same visible trace. This trace can indicate causal relations inherent to the protocol, e.g., if A signed a message that should never appear, however, some deviations introduce additional causal relations like provocation (Example 7), which we wish to ignore. Optimality should be weak enough to recognize uncertainty, e.g., both plausible deviations in Example 8, and preserve the causal information indicated by $t|_V$, yet strong enough to remove the second form of causal relations.

Example 9 (2-out-of-3 vote, [DGK+15, Section IV]). For an attack to be successful, two out of three servers A, B, C need to validate a compromised certificate for a trusted party T. All of them deviate by accepting it, which is publicly visible.

In order for $apv_\varphi(\pi, u, d)$ to enforce the correct verdict, $(A \wedge B) \vee (B \wedge C) \vee (A \wedge C)$, d and u need to contain all the causal information in the trace relating to the security violation. All optimality notions we discuss here thus require d and u to reproduce the visible part of the trace. Hence the question remains how to choose among these d and u.

Verdict-optimal deviation The first option is to minimize over the a posteriori verdict of each candidate deviation and context. We define the following order based on logical implication.

Definition 10 (verdict order). $\mathcal{S}_1 \leq \mathcal{S}_2$ *if* $\bigvee_{\mathrm{S}\in\mathcal{S}_2} \bigwedge_{p\in\mathrm{S}} p \implies \bigvee_{\mathrm{S}\in\mathcal{S}_1} \bigwedge_{p\in\mathrm{S}} p$.

The a posteriori verdict gives the empty verdict $\emptyset$ if no violation occurs in the given deviation and trace. As a propositional formula, this translates to $\bigvee_{\mathrm{S}\in\emptyset} \bigwedge_{p\in\mathrm{S}} p = \bot$. As there are no negative atoms in these formulas, $\emptyset \leq \mathcal{S}_2$ only if $\mathcal{S}_2 = \emptyset$. Hence $\emptyset$ is the bottom element of this order, while $\{\emptyset\}$ is the top element, however, this a posteriori verdict only occurs if the normative behaviour

of the protocol may produce a violation by itself. Note further, that, if $\mathcal{S}_2 \neq \emptyset$, then,

$$\mathcal{S}_1 \leq \mathcal{S}_2 \iff \forall \mathrm{S}' \in \mathcal{S}_2 \exists \mathrm{S} \in \mathcal{S}_1 . \mathrm{S} \subseteq \mathrm{S}'. \tag{1}$$

$$\begin{aligned}
&\forall \mathrm{S}' \in \mathcal{S}_2 \exists \mathrm{S} \in \mathcal{S}_1 . \mathrm{S} \subseteq \mathrm{S}' \\
\implies &\forall \mathrm{S}' \in \mathcal{S}_2 \exists \mathrm{S} \in \mathcal{S}_1 . \bigwedge_{p \in \mathrm{S}'} p \implies \bigwedge_{p \in \mathrm{S}} p \\
\implies &\forall \mathrm{S}' \in \mathcal{S}_2 . \bigwedge_{p \in \mathrm{S}'} p \implies \bigvee_{\mathrm{S} \in \mathcal{S}_1} \bigwedge_{p \in \mathrm{S}} p \\
\implies &\bigvee_{\mathrm{S}' \in \mathcal{S}_2} \bigwedge_{p \in \mathrm{S}'} p \implies \bigvee_{\mathrm{S} \in \mathcal{S}_1} \bigwedge_{p \in \mathrm{S}} p \quad \text{if } \mathcal{S}_2 \neq \emptyset. \\
\iff &\mathcal{S}_1 \leq \mathcal{S}_2.
\end{aligned}$$

For the other direction, we show that $\exists \mathrm{S}' \in \mathcal{S}_2 \forall \mathrm{S} \in \mathcal{S}_1 . \mathrm{S} \not\subseteq \mathrm{S}'$ implies $\mathcal{S}_1 \not\leq \mathcal{S}_2$ by contradiction. Hence we assume that $\mathcal{S}_1 \leq \mathcal{S}_2$ and fix $\mathcal{S}_1$, $\mathcal{S}_2$ and S' such that

$$\mathcal{S}_1 \leq \mathcal{S}_2 \iff \bigvee_{\mathrm{S}' \in \mathcal{S}_2} \bigwedge_{p \in \mathrm{S}'} p \implies \bigvee_{\mathrm{S} \in \mathcal{S}_1} \bigwedge_{p \in \mathrm{S}} p \tag{2}$$

As $\mathrm{S}' \in \mathcal{S}_2$,

$$\bigwedge_{p \in \mathrm{S}'} p \implies \bigvee_{\mathrm{S} \in \mathcal{S}_2} \bigwedge_{p \in \mathrm{S}} p.$$

Hence by (2),

$$\bigwedge_{p \in \mathrm{S}'} p \implies \bigvee_{\mathrm{S} \in \mathcal{S}_1} \bigwedge_{p \in \mathrm{S}} p.$$

And thus there must be $\mathrm{S} \in \mathcal{S}_1$ such that

$$\bigwedge_{p \in \mathrm{S}'} p \implies \bigwedge_{p \in \mathrm{S}} p.$$

Hence $\mathrm{S} \subseteq \mathrm{S}'$, contradicting the assumption.

Definition 11 (verdict-optimal). *Deviation and context d, u are verdict-optimal w.r.t. some (visible) trace t_V and an a posteriori deviation d_p if $t_V =_E trace(\pi[d[d_p]], u)|_V$ and $apv_\varphi(\pi, u, d[d_p])$ is minimal w.r.t. to the above order on verdicts.*

Coming back to the provocation example, given that only A's malicious behavior is visible and sufficient to provoke a violation, $apv_\varphi(\pi, u, d) = \{\{A\}\}$ for d mapping A to a process behaving maliciously. Assuming that no deviation may provoke the violation without A behaving this way, this deviation and context are verdict optimal, and the deviation where A is provoked by B is ignored in favour of this simpler explanation, as $A \wedge B$ implies A.

While Lemma 2 guarantees an empty verdict if every party is following its normative behaviour, the following theorem provides assurance that non-violating traces are recognized as such, and that at least *someone* is blamed if something bad happens.

Theorem 41 (accountability implies verifiability). *If verdict provides π with accountability for some φ, V such that $\varphi(t) \iff \varphi(t|_V)$ for all traces t and w.r.t. any sane notion of optimality, then it provides π with verifiability.*

Proof. Let t be fixed but arbitrary. Note first that, by the sanity requirement, and assumptions of φ, for any optimal deviation d and context u explaining $t|_V$:

$$\begin{aligned}\varphi(t) &\iff \varphi(t|_V)\\ &\iff \varphi(\mathit{trace}(\pi[d],u)|_V)\\ &\iff \varphi(\mathit{trace}(\pi[d],u))\\ &\iff (\pi,u) \vDash [d]\varphi\end{aligned}$$

The proof has two steps:

$$\varphi(t) \iff \mathit{apv}_\varphi(\pi,u,d) = \emptyset \quad \text{for all optimal } d,u \tag{3}$$

$$\iff \mathit{verdict}(t|_V) = \emptyset. \tag{4}$$

Step 1 holds because:

- ($\Longrightarrow$) If $\varphi(t)$, then for any optimal deviation and context and any subset of parties B, there is $\mathrm{S}' = \mathcal{P} \supseteq \mathrm{S}$ such that $u' = u$ reproduces a non-violating trace, i.e., $(\pi,u) \vDash [d|_{\mathrm{S}'}]\neg\varphi$ does not hold as $\varphi(t) \iff (\pi,u') \vDash [d]\varphi$.
- ($\Longleftarrow$) Assume $\neg\varphi(t)$, then there are optimal deviation and context d,u such that $t|_V \in \mathit{trace}(d,u)|_v\mathit{is}$ (by sanity requirement) and for $\mathrm{S} = \mathcal{P}$ and all $\mathrm{S}' \supseteq \mathrm{S}$, i.e., $\mathrm{S}' = \mathrm{S}$, $(\pi,u') \vDash [d|_{\mathrm{S}'}]\neg\varphi$ as $d|_{\mathrm{S}'} = d$. Thus, at least $\mathrm{S} = \mathcal{P}$ or a subset thereof is in $\mathit{apv}_\varphi(\pi,u,d)$.

The second step holds by assumption and Definition 9.

□

Verdict-optimal accountability gives a very strong guarantee: all deviations and contexts that are disregarded because they are not verdict-optimal actually imply the verdict, hence *a*) no one is blamed that would not be blamed otherwise, and *b*) from (1) we conclude that for every conjunct of the weaker verdict, which we can understand as a group of agents "working together" to produce the violation, at least one representative appears in a conjunct of the actual verdict, and can hence point out more subtle degrees of responsibility (or take all the blame).

Knowledge-optimal deviation The second notion of optimality we propose is based on the amount of knowledge parties need to share in order to reproduce the visible part of the trace. We assume that all parties can share information over invisible channels, but make the information shared explicit. Hence we relax Definition 4 so that all deviating parties can share some knowledge initially, but we will forbid communication between deviating parties later.

Definition 12 (relaxed deviation). *A* relaxed deviation *for a protocol* $\pi = (A, n)$*, is a partial function* d *from* $\mathcal{P} \setminus \mathcal{T}$ *to plain processes, such that* $fv(d(p)) \subseteq \bigcup_{p' \in dom(d)} fv(n(p'))$ *and* $fn(d(p)) \subseteq \bigcup_{p' \in dom(d)} fn(n(p'))$ *for every* $p \in dom(d)$*.*

The idea is that a deviating party can use any free name or variable used by another deviating party, giving us a measure of the information shared. (Although not a very precise one: a deviation of party A that obtains a single signature from B requires as much information sharing as a deviation that obtains hundreds of signatures: B's signing key.) We can now compare two deviations by comparing the information available to deviating parties, excluding the information allowed to be shared, i.e, $fv(n(p))$ and $fn(n(p))$.

Definition 13 (knowledge order). $d_1 \leq d_2$ *if* $dom(d_1) \subseteq dom(d_2)$*, and for all* $p \in dom(d_1)$*,* $fv(d_1(p)) \backslash fv(n(p)) \subseteq fv(d_2(p)) \backslash fv(n(p))$*, and* $fn(d_1(p)) \backslash fn(n(p)) \subseteq fn(d_2(p)) \setminus fn(n(p))$*.*

Definition 4 and Lemma 1 are trivially extended to relaxed deviations. For Definition 8, we slightly modify the restriction operator: $d|_{\mathrm{S}}(p)$ is undefined if $p \notin \mathrm{S}$, and defined $\nu \vec{n}'.d(p)\sigma$ where σ and the sequence of names $\vec{n}'$ are chosen such that names or variables that become unavailable as a result of the restriction are assigned fresh names, or structurally similar terms with fresh names.[4] E.g., sk_A and pk_A are substituted by sk_{dum} and $pk(sk_{dum})$ if the active substitution in the protocol mapped pk_A to $pk(sk_A)$. This preserves the requirements of Definition 12 and captures the intuition that removing any party from the deviation also removes information that only this party could have shared.

As all information is exchanged before the protocol run, we restrict all deviating parties to not communicate with each other, in order to quantify the amount of information they share by their free variables and names. We modify Definition 8, such that, for any $\mathrm{S} \subseteq \mathcal{P}$, we only consider contexts in which deviating parties do not communicate with each other, i.e., $(\pi, u) \vDash [d|_{\mathrm{S}}]\varphi$ implies that $(p_A, p_B) \notin u$ for all $p_A, p_B \in dom(d)$.

Definition 14 (knowledge-optimal). *A relaxed deviation and context* d, u *are knowledge-optimal w.r.t. some (visible) trace* t_V *and a posteriori deviation* d_p *if*

[4] Formally, we assume a bijection ρ between the set of names occurring in $fn(d(p)) \cup fn(fv(t)\sigma_A) \setminus \bigcup_{p' \in dom(d) \cap \mathrm{S}} fn(n(p'))$ (for σ_A the active substitutions in scope of $\cdot_p$ in $\pi = (A, n)$) and a sufficiently large set of fresh names. Then σ substitutes every name $n \in fn(d(p)) \setminus \bigcup_{p' \in dom(d) \cap \mathrm{S}} fn(n(p'))$, not available due to the restriction anymore by a (unique) fresh name according to ρ, and every variable $v \in fv(d(p)) \setminus \bigcup_{p' \in dom(d) \cap \mathrm{S}} fv(n(p'))$, by ρ applied to $\sigma_A(v)$.

trace$(\pi[d[d_p]], u)|_V =_E t|_V$, $(p_1, p_2) \notin u$ *for all* $p_1, p_2 \in dom(d)$, *and* d *minimal w.r.t. to the above order on relaxed deviations.*

Knowledge-optimality is motivated by the observation that deviations such as Example 7 rely on causal dependencies that are not apparent from the observed trace and the protocol itself, but are artificial in the sense that they result from the behaviour of the deviating parties alone. Hence these deviations are explanations for the observable behavior which can be constructed after the fact. Knowledge-optimality aims at excluding these "artificial" deviations when reviewing all possible explanations for a visible trace. It restricts communication to sharing of secrets, which induces causal dependencies between deviating parties which are inherent to the protocol, hence not "artificial". In the case studies we considered, this notion is indeed successful in eliminating these deviations.

For most applications we considered, the visible trace only consists of events emitted by parties, modelling claims or trusted public logs. Hence, for these examples, relaxed deviations are complete with respect to visible traces.

Lemma 3 (relaxed deviations completeness). *For any protocol* π, *deviation* d *and context* u *with* $t \in$ *trace*$(\pi[d], u)$, *there is a relaxed deviation* d' *and a context* u' *with* $(p_1, p_2) \notin u'$ *for all* $p_1, p_2 \in dom(d')$, *such that* *trace*$(\pi[d'], u') =_E t|_\delta$, *where* $\delta = \mathcal{P} \uplus \mathcal{P} \times \text{Terms} \uplus \{(p_A, p_B, m) \mid p_A, p_B \notin dom(d')\}$.

Proof. Let π, d, u and t be fixed but arbitrary. Let $m = |t|$. Then there is a chain of reductions $A_0 \xrightarrow{t_1} A_1 \xrightarrow{t_2} \cdots \xrightarrow{t_m} A_m$ with $A_0 = \pi[d]$. Any extended process A_i can be brought into form $A_i \equiv \nu\vec{n}.(\{^{\vec{m}}/_{\vec{x}}\} \mid \ldots \mid P_1 | \cdots | P_k)$ [AF01]. Here $\vec{n}$ is a sequence of names $n_1, \ldots n_m$ and $\nu\vec{n}$ is shorthand for $\nu n_1; \ldots, \nu n_m$. Moreover, observing that in the initial extended process $\pi[d]$, P_1 to P_k each correspond to exactly one party, we can require the form

$$A_i \equiv \nu\vec{n}.(\{^{\vec{m}}/_{\vec{x}}\} \mid P_{p_1} | \cdots | P_{p_k})$$

for some ordering of $\{p_1, \ldots, p_k\} = \mathcal{P}$ w.l.o.g. (i.e., keeping 0 processes even if parties have finished execution).

Contrary to the applied π-calculus, plain processes lack replication and thus w.l.o.g. we can assume all fresh values are contained in $\vec{n}$ and P_{p_1} to P_{p_k} contain only conditionals, events and message input and message output. Furthermore, w.l.o.g., any name transmitted is unique and every variable bound by in is unique. Thus we can assume

$$A_0 \equiv \nu\vec{n}_0.(\{^{\vec{m}_0}/_{\vec{x}_0}\} \mid P_{p_1} | \cdots | P_{p_k})$$

and every subsequent A_i, $i > 0$:

$$A_i \equiv \nu\vec{n}_0.(\{^{\vec{m}_0}/_{\vec{x}_0}\} \mid \nu\vec{x}_{p_1}\{^{\vec{m}_{p_1}}/_{\vec{x}_{p_1}}\} \mid P_{p_1} \\ \mid \cdots \mid \nu\vec{x}_{p_k}\{^{\vec{m}_{p_k}}/_{\vec{x}_{p_k}}\} P_{p_k})$$

where $\vec{x}_p$ are the variables transmitted to p so far. For each $p \in \mathcal{P}$ and $m \in \vec{m}_p$, any variable in m is bound either in $\sigma_{p'} := \{^{\vec{m}_{p'}}/_{\vec{x}_{p'}}\}$ where p' was the sending

party or in $\sigma_0 := \{^{\vec{m}_0}/_{\vec{x}_0}\}$. As there is a total order on the message transmission, this implies that the process is closed.

We now apply the following transformation to the reduction sequence: each transition $A_{i-1} \xrightarrow{t_i} A_i$ with $t_i = (p_A, p_B, x)$ for p_A and p_B both in $dom(d)$ is removed by equating both A_{i-1} and A_i to the following process A'_{i-1}. As this transition is an instance of Comm,

$$A_{i-1} \equiv \nu\vec{n}_0.(\sigma_0 \mid \sigma_{p_A} \mid (\mathsf{out}(x).P)_{p_A} \mid \sigma_{p_B} \mid (\mathsf{in}(x).Q)_{p_B} \\ \mid \sigma_{p_3} \mid P_{p_3} \mid \sigma_{p_k} \mid \cdots P_{p_k})$$

and

$$A_i \equiv \nu\vec{n}_0.(\sigma_0 \mid \sigma_{p_A} \mid P_{p_A} \mid \sigma_{p_A} \mid Q_{p_B} \\ \mid \sigma_{p_3} \mid P_{p_3} \mid \sigma_{p_k} \mid \cdots P_{p_k})$$

for some P and Q, we chose $A'_{i-1} = A_i$, i.e., instead of transmitting a message, we make sure they are contained a priori in $\vec{x}$. Then $A'_{i-1} \xrightarrow{t_{i+1}} A_{i+1}$. We can apply this process backwards and obtain $A'_0 \xrightarrow{t|_\delta|_1} A'_1 \xrightarrow{t|_\delta|_2} \cdots \xrightarrow{t|_\delta|'_m} A'_m$ for $m' = |t|_\delta|$. Syntactically, all corresponding (according to $\vec{t}$) input and output constructs in each P_p for $p \in dom(d)$ in A_0 are removed to produce A'_0. Hence we can formulate A'_0 as follows:

$$A'_0 \equiv \nu\vec{n}_0.(\{^{\vec{m}_0}/_{\vec{x}_0}\} \mid \sigma_{p_1} \mid P'_{p_1} \mid \sigma_{p_k} \mid \cdots P'_{p_k})$$

where the corresponding process P'_p in A_0 contains free variables for each $\vec{x}_p$, i.e., $fv(P'_p) = fv(n(p)) \cup \{x \in \vec{x}_p \mid (p', p, x) \in \vec{t}, p' \in dom(d)\}$ and $fn(P'_p) = fn(n(p))$, as opposed to P_p in A_0, where $fv(P_p) = fv(n(p))$ and $fn(P_p) = fn(n(p))$.

We can show by induction on the variables in

$$\{x \in \vec{x}_p \mid p', p \in dom(d), (p', p, x) \in \vec{t}\}$$

in order of transmission in $\vec{t}$, that for each $p \in dom(d)$, $x \in \vec{x}_p$, $fv(x) \subseteq \bigcup_{p' \in dom(d)} fv(n(p'))$ and $fn(x) \subseteq \bigcup_{p' \in dom(d)} fn(n(p'))$, since for each $m \in \vec{m}_p$, each variable is bound either by $\sigma_{p'}$ where p' was the sending party, or by σ_0. Hence we can conclude that by applying Subst for each $x \in \vec{x}_p$ for each $p \in dom(d)$, A_0 is equivalent to an instance $\pi[d']$ for a relaxed deviation d'. □

This lemma holds because communication is not authenticated, and relaxed deviations can share necessary secrets beforehand. In the extreme case, one deviating party can act on behalf of all others.

Similar to Theorem 41, which does not apply to relaxed deviations, we can assure ourselves that accountability w.r.t. knowledge-optimal parties provides verifiability.

Theorem 42 (knowledge-optimal acc. implies verifiability). *Assume V does not contain communication ($V \cap (\mathcal{P} \times \mathcal{P} \times \mathcal{M}) = \emptyset$) and $\varphi(t) \iff \varphi(t|_V)$ for all $t \in traces(\pi)$. Then, if verdict provides π with knowledge-optimal accountability for some φ, V, verdict provides π with verifiability.*

Proof. Let $\varphi(t)$. Then $apv_\varphi(\pi, u, d) = \emptyset$ for the deviation and context d', u' with $t \in trace(\pi[d], u)$. By Lemma 3, we can assume a relaxed deviation d^* and a context u^* with $(p_1, p_2) \notin u$ for all $p_1, p_2 \in dom(d)$, and $trace(d^*, u^*) =_E t|_\delta$ as above. Since $\varphi(t) \iff \varphi(t|_\delta)$, for $t' \in trace(\pi, d^*, u)$ $t'|_V|_\delta = t'|_V =_E t|_V$, $\varphi(t')$ and thus $apv_\varphi(\pi, u', d') = \emptyset$, too. Thus either the protocol does not provide verdict-optimal accountability, or $verdict(t|_V) = \emptyset$.

Let $verdict(t|_V) = \emptyset$. Thus for all knowledge-optimal d, u with $trace(\pi[d], u)|_V =_E t|_V$, $apv_\varphi(\pi, u, d) = \emptyset$. Hence there are no knowledge-optimal deviation d and context u such that $apv_\varphi(\pi, u, d) \neq \emptyset$, as otherwise the protocol would not provide knowledge-optimal accountability. By Lemma 3, there is a relaxed deviation and context d^* and u^* for the actual deviation and context d_a and u_a that produced t, i.e., d^* and u^* with $trace(\pi[d^*], u^*)|_V =_E t|_V$. W.l.o.g. d^* and u^* are knowledge-optimal. But if $apv_\varphi(\pi, u^*, d^*) = \emptyset$, then even for $\mathrm{S} = \mathcal{P}$, there is no $u' \sqsubseteq_\mathcal{P} u^*$, i.e., $u' = u^*$, such that $(\pi, u') \vDash [d]\neg\varphi)$. Hence $\neg\varphi(t)$ cannot be true, and thus $\varphi(t)$. □

There is a class of examples where knowledge-optimal accountability is required, as verdict-optimal accountability is not applicable. Unsurprisingly, they rely on artificial causal dependencies between deviating parties. Reconsider Example 9. It is plausible to assume that any of the three servers, say C, communicated with the two others, A and B, deciding to only deviate if A and B deviate. For such a deviation and context d, u, $apv_\varphi(\pi, u, d) = \{\{A, B\}\}$, since $(\pi, u') \vDash [d|_{i,C}]\varphi$ for $i \in \{A, B\}$ and some $u' =_{\{i,C\}} u$. Barring knowledge about the actual deviation, this argument can be made for B or A in place of C. Hence, the protocol described in Example 9 is not accountable w.r.t. verdict-optimal deviations. One might agree — after all, d is plausible — and decide this protocol does not provide accountability. On the other hand, C's behaviour was only observed in the case where A and B actually deviate. What C would have done if this was not the case can be neither proven nor refuted, d is just one possible explanation, and arguably not the simplest one. This situation, which is an instance of the provocation problem restricted to deviating parties, occurs whenever there are two or more parties in a set that is part of a verdict given by the verdict function. This is the case in both our case studies, however, there are scenarios where only one party is untrusted, e.g., Kroll's original analysis of the accountable computation protocol [Kro15].

Knowledge-optimal accountability is applicable in this case. Observe that no exchange of secrets is necessary to produce the observed trace, hence, no matter which knowledge-optimal deviation and context d^*, u^* are chosen, a restriction of the domain of d^* does not alter the message sent from either server to the client. Thus, for all knowledge-optimal deviations, $apv_\varphi(\pi, u^*, d^*) = \{\{A, B\}, \{B, C\}, \{A, C\}\}$. Note, however, that if we modify the example so that T provides a side-channel, e.g., it relays A's decision to B, knowledge-optimality again suffers the same problem. In this case, knowledge-optimality would conservatively reject the protocol. Protocols with such side-channels may require a more refined notion of optimality, which we leave as an open question. As the attacks and proofs in the next section will demonstrate, accountability w.r.t.

knowledge-optimality is applicable to both our case studies, which tackle two frequent use cases which are very different from each other.

We stress that, in the context of Definition 9, knowledge-optimality is *not* an assumption on the actual behaviour of the deviating parties, but an assumption on which kind of explanations should be considered when determining the verdict. If the reader considers Definition 8 appropriate, then the provocation problem witnesses that, for realistic protocols, no distinct verdict can capture all possible explanations for a violation. The conclusion is that either unambiguity of the verdict needs to be dropped, the corruption model be changed (e.g., to a single adversary controlling all deviating parties), or completeness be weakened to some extent. We chose to leave the first two options for future work and concentrate on the third.

We conclude that the provocation problem requires weakening accountability by regarding only optimal deviations and contexts producing the trace. Verdict-optimality provides the guarantee a represantative from each set of jointly misbehaving agents, but it can only apply in cases where all a posteriori verdicts are empty or consist of singleton sets (e.g., access control [BCS05], randomness generation [BFM13] and holding trusted third parties accountable [ASW98, KTV10]). Knowledge-optimality assumes that deviating parties share no more information than necessary, but still guarantees soundness (Lemma 2) and that at least some deviating party is blamed in case of misbehaviouar (Theorem 42).

5 Case studies

In this section, we come back to the central accountability monitor in Example 2, but will focus on an accountability protocol based on zero-knowledge proofs and commitments which is actually implemented. This protocol provides a "framework [..] to enable meaningful after-the-fact oversight [of computations conducted by authorities], consistent with the norm in law and policy" [Kro15], and hence our analysis may be of interest for protocol designers. As opposed to the centralized accountability monitor, this protocol has no trusted parties, hence the evidence collected for accountability might be tampered with by deviating parties.

The goal is to show that our definition is able to find subtle attacks in the centralized and the decentralized setting, but still allows to show accountability under reasonable assumptions, and in both settings. Furthermore, the proofs (in the long version [Ano17]) are of reasonable length and complexity.

5.1 Delegation (Example 2)

Consider the protocol defined in Example 2. The task of the monitor is thus to provide enough evidence of the parties deviating from the protocol by issuing an unusual action. We thus assume only the events of the trusted party to be visible, $V = \{T\} \times \text{Terms}$. As mentioned before, the monitor should hold parties

responsible for the execution of unusual actions accountable:

$$\varphi(t) := \forall e \in t.\ e =_E \mathit{Exec}(a) \implies a \in_E \{\mathit{NAct}, \mathit{SAct}\}.$$

It is obvious that the protocol provides for verifiability (Definition 7), since a *Log* event is always triggered before an *Exec* event appears, and any event $\mathit{Exec}(a)$ with an a which is not an action or a special action leads to a verdict. However, the protocol as defined in Example 2 allows A to trick B into making itself appear partially responsible for a violation, despite being honest. Further, any party can produce a verdict that blames A and only A.

It is obvious that the protocol provides for verifiability, since a *Log* event is always triggered before a *Exec* event appears, and any event $\mathit{Exec}(a)$ with an a which is not an action or a special action leads to a verdict. However, the protocol as defined in Example 2 allows A to trick B into making itself appear partially responsible for a violation, despite being honest, and any party to produce at verdict that blames A and only A. We think that it is instructive to first explain how these attacks manifests in our framework. We then proceed to repair the protocol and show it secure w.r.t. the same verdict, and knowledge-optimal deviations.

Two incorrect verdicts We will now show that both non-empty verdicts $\{\{A\}\}$ and $\{\{A, B\}\}$ may be incorrect w.r.t. Definition 9 and knowledge-optimality. Assume the verdict is $\{\{A\}\}$ for some fixed but arbitrary knowledge-optimal d and u, i.e., the trace $t \in \mathit{trace}(\pi, u, d)$ contains $\mathit{Log}(A, a) \in t \wedge a \neq \mathit{NAct}$. While this implies that a message m such that $\mathit{versig}(\mathit{pk}(\mathit{sk}_A), \pi_2(m), \pi_1(m))$ was transmitted to T, it is possible for any party to deduce such message and trigger the verdict. A responds to a term SAct with a pair $\langle n_a, \mathit{sig}(\mathit{sk}_A, \langle \mathit{SAct}(a'), n_a \rangle)\rangle$ for some name n_a. Thus, any single deviating party, e.g., I, can trigger the violation without any further knowledge. Hence the set of knowledge-optimal deviations and contexts disagree on their respective a posteriori verdicts.

Now assume the verdict is $\{\{A, B\}\}$ for some fixed but arbitrary knowledge-optimal d and u, i.e., the trace $t \in \mathit{trace}(\pi, u, d)$ contains $\mathit{Log}(\langle A, B\rangle, a)$ but $\neg(\mathit{isAct}(a) = \mathit{true})$. By definition of $n(T)$, this implies that T has received a message m such that $m =_E \langle a, y, \mathit{sig}(\mathit{sk}_A, \langle a, y\rangle), \mathit{sig}(\mathit{sk}_B, y)\rangle$ for some term y. The four terms that constitute this message can be constructed, even if A alone is deviating. As $n(B)$'s response is not bound to the action in particular, $d(A)$ can create a message $\langle n_a, \mathit{sig}(\mathit{sk}_A, \langle \mathit{SAct}, n_a\rangle)\rangle$, receive $\mathit{sig}(\mathit{sk}_B, n_B)$ and construct the pair m for any a. Hence there is a knowledge-optimal (d, u) with $\mathit{dom}(d) = \{A\}$ and $\mathit{fv}(d(A)) = \mathit{fv}(n(A))$ with $\mathit{apv}_\varphi(\pi, u, d) = \{A\}$, contradicting accountability in this case, too.

Example 10 (delegation, fixed). Consider the signature and equational theory described in Example 2, but the normal behaviour modified as follows:

$n(T) :=$ in(m);
  if $\mathit{versig}(\mathit{pk}_A, \pi_2(m), \pi_1(m)) = \mathit{true}$ then
if $\mathit{isAct}(\pi_1(m)) = \mathit{true}$ then

$$\begin{array}{l}
\quad \mathsf{event}\ \ \mathit{Log}(A, \pi_1(m))\mathit{;}\ \mathsf{event}\ \ \mathit{Exec}(\pi_1(m)) \\
\mathsf{else}\ \ \mathsf{if}\ \ \mathit{versig}(pk_A, s_1, \langle xa, xn\rangle) = \mathit{true}\ \mathsf{then} \\
\qquad \mathsf{if}\ \ \mathit{versig}(pk_B, s_2, xn) = \mathit{true}\ \mathsf{then} \\
\qquad \mathsf{if}\ \ \mathit{isAct}(xa) = \mathit{true}\ \mathsf{then} \\
\qquad\quad \mathsf{event}\ \ \mathit{Log}(\langle A, B\rangle, xa)\mathit{;}\ \mathsf{event}\ \ \mathit{Exec}(xa)
\end{array}$$

(As before, in the second branch $m = \langle xa, s_1, s_2\rangle$.)

$$\begin{array}{l}
n(A) := \mathsf{in}(a)\mathit{;}\ \mathsf{if}\ a = \mathit{NAct}\ \mathsf{then}\ \mathsf{out}\ \langle a, \mathit{sig}(sk_A, a)\rangle \\
\qquad \mathsf{else}\ \ \mathsf{if}\ \ a = \mathit{SAct}\ \mathsf{then} \\
\qquad\quad \mathsf{out}\ \ \langle a, \mathit{sig}(sk_A, \langle a, a\rangle)\rangle\mathit{;} \\
\quad \mathsf{in}\ \ r\mathit{;}\ \mathsf{if}\ \ \mathit{versig}(pk_B, r, \mathit{sig}(sk_A, \langle a, a\rangle)) = \mathit{true}\ \mathsf{then} \\
\quad \mathsf{out}\ \ \langle a, \mathit{sig}(sk_A, \langle a, a\rangle), r\rangle
\end{array}$$

$$\begin{array}{l}
n(B) := \mathsf{in}(m)\mathit{;} \\
\quad \mathsf{if}\ \ \mathit{versig}(pk_A, \pi_2(m), \langle \mathit{SAct}, \mathit{SAct}\rangle) = \mathit{true} \\
\qquad\qquad \mathsf{then}\ \mathsf{out}\ \ \mathit{sig}(sk_B, \pi_2(m))
\end{array}$$

$$n(I) := \mathsf{out}\ \mathit{SAct}\mathit{;}\ \mathsf{out}\ \langle pk_A, pk_B\rangle$$

Now, T verifies that its input contains an action (which might still be an unusual action). Instead of binding B's validation of an action via a nonce that A can later misuse, B's response contains the action it meant to validate. A's request to B only needs to contain the action now, but to avoid T confusing this message with a signed normal action, A sends the signature of a $\langle a, a\rangle$ instead of just a.

Theorem 1. *The verdict in Example 4 provides the protocol described in Example 10 with accountability for the property φ w.r.t. $V = \{T\} \times$ Terms and knowledge-optimality.*

Proof. Case distinction over an arbitrary but fixed t. Let t s.t. $\mathit{verdict}(t|_V) = \emptyset$. φ holds, as a *Log* event is always triggered before an *Exec* event appears (by the structure of $n(T)$), and any event $\mathit{Exec}(a)$ where a is not a normal action or a special action leads to a verdict. Hence $\mathit{apv}_\varphi(\pi, u, d) = \emptyset$ for all d and u such that $t|_V \in \mathit{trace}(\pi[d], u)$.

Let t s.t. $\mathit{verdict}(t|_V) = \{\{A\}\}$. Then $\mathit{Exec}(a) \in t|_V$, $\mathit{Log}(A, a) \in t|_V$ and $a \neq \mathit{NAct}$, and thus, by definition of $n(T)$, $t|_V = (\mathit{Log}(A, a), \mathit{Exec}(A, a))$. For any derivation producing t, a message $\langle a, m\rangle$ with $m =_E \mathit{sig}(sk_A, a)$ for $a \in \{\mathit{SAct}, \mathit{UnAct}\}$. must be transmitted Such a message can either be constructed knowing sk_A, or deduced from other protocol output. In the first case, we deduce that A must be deviating. As V defines only the events emitted by T as visible, we can chose any relaxed deviation d and context u that reproduce these exact event. Consider $d : A \mapsto \ \mathsf{out}\ m$ for above m, and $u = ((A, T), T, T, T, T)$, which explain t. We verify that $\mathit{apv}_\varphi(\pi, u, d) = \{\{A\}\}$ and $\mathit{fn}(d(A)) = sk_A$, $\mathit{fv}(d(A)) = \emptyset$. As argued before, the empty deviation cannot produce a violating trace, hence d and u are knowledge-minimal. Coming back to the case where m is deduced from some protocol output, we only need to consider cases where

A follows its normative behaviour, as any other S would not be minimal. Even considering arbitrary deviations of B and I, such an m cannot be constructed by either, nor can it be deduced from one of the three outputs within $n(A)$, as any reduction labelled (A, X, m') of $\pi[d]$ with $A \notin dom(d)$ for any $X \in \mathcal{P}$ and term m' has form $\langle a, sig(sk(pk_A), a)\rangle$, or $\langle a, sig(sk(pk_A), \langle a, a\rangle)\rangle$, for some a previously available, but of the form $NAct(a')$ or $SAct(a')$ for some term a', or is just a forwarding of the message received. Hence, if a message of form m can be deduced from this message, it is also deducible without it. Therefore, $\{\{A\}\}$ is the unique minimal verdict.

Let t s.t. $verdict(t|_V) = \{\{A, B\}\}$, which is the last case. Then $Exec(a) \in t|_V$, $Log(A, B, a) \in t|_V$ and $a \neq SAct$, and thus, by definition of $n(T)$, $t|_V = (Log(\langle A, B\rangle, a), Exec(\langle A, B\rangle, a))$. For any derivation producing t, a message $m =_E \langle xa, xn, s_1, s_2\rangle$ such that $versig(pk_A, s_1, \langle xa, xn\rangle) = true$, $versig(pk_B, s_2, xn) = true$ and $isAct(xa) = true$ was transmitted. Therefore, $(X, T, m) \in t$ for some $X \in \mathcal{P}$. By definition of $=_E$, any such m is of form $m =_E \langle a, sig(sk_A, \langle a, a\rangle), sig(sk_B, sig(sk_A, \langle a, a\rangle)\rangle$ for $a \in \{NAct, UnAct\}$. This message can either be constructed knowing sk_A and sk_B or deduced from other protocol output. For the first case, we show that this is possible indeed and thus $\{\{A, B\}\} = apv_\varphi(\pi, u, d)$ for some d and u with $dom(d) = \{A, B\}$ and $fn(d(A)) = fn(n(B)) \cup fn(n(A))$. Then, we show that for either A and I, or B and I deviating, it is impossible to deduce such an m. For one, this shows knowlege-minimality of d, second, this shows that $\{\{A, B\}\}$ is indeed a unique minimal a posteriori verdict. The deviation d mapping B to 0 and A to $\nu\ a;\ \mathsf{out}(m)$ along with the context $u = (A, (A, T), T, T, T)$ produces a trace with the same visible part $t|_V = (Log(A, B, a), Exec(\langle A, B\rangle, a))$. We can verify that $apv_\varphi(\pi, u, d) = \{\{A, B\}\}$, as either substituting A or B by their normative behaviour remedies the violation, in particular, $d|_{\{A\}}(A)$ undefined. It is left to show, that no derivation d with $dom(d) = \{A, I\}$ or $dom(d) = \{A, I\}$ can coincide with $t|_V$, which (as mentioned before) requires emitting m. First, if $dom(d) = \{A, I\}$, then the name sk_A available does not suffice to construct m. The only additional message that may be learned is the output of $n(B)$, but it is of form $sig(sk_B, SAct)$. Second, if $dom(d) = \{B, I\}$, then the name sk_B available does not suffice to construct m (due to its subterm $sig(sk_A, a)$). The only additional message learned is the output of $n(A)$, but it is either of form $\langle a, sig(sk_A, NAct)\rangle$ (in which case the same argument as in the last case applies) or of form

$$\langle SAct, sig(sk_A, \langle SAct, SAct\rangle), sig(sk_B, sig(sk_A, \langle SAct, SAct\rangle)\rangle$$

for some a'. While it is possible to extract the action $SAct$, again it is not possible to deduce m or the mentioned subterm. This shows that the verdict is correct in the last case, and thus concludes the proof. □

5.2 Accountable algorithms

We have analysed the accountable algorithms protocol due to Kroll [Kro15, Chapter 5], which lets an authority A, e.g., a tax authority, make accountable

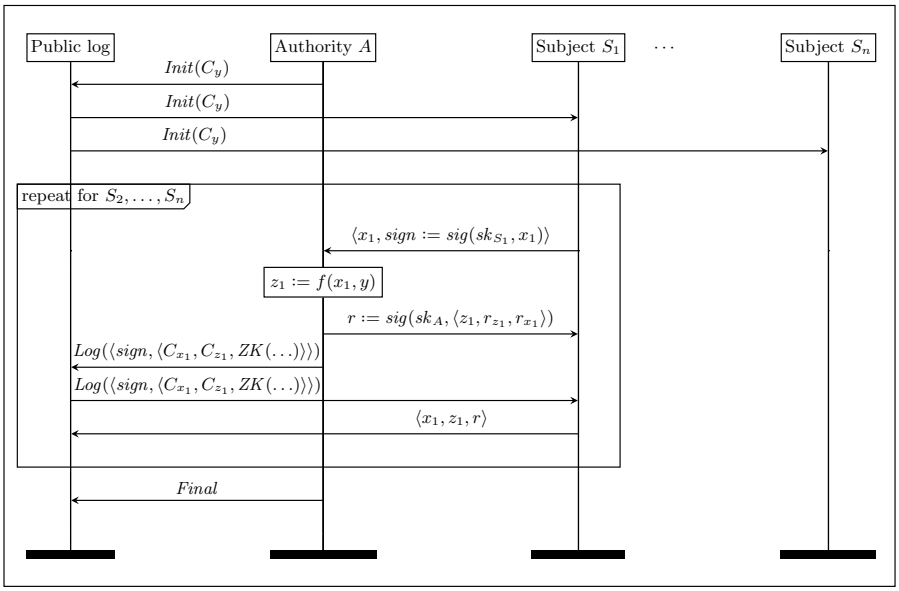

Fig. 3. Honest protocol run for Example 11

computations for any number of subjects S_1 to S_n. It is useful for many tasks in which a central authority computes a function for a large number of subjects that provide inputs, e.g., validation of tax returns. A secret input y may be used to hold elements of some policy secret, but fixed from the start, e.g., a threshold on the gross income to decide when someone will undergo a tax audit. The most interesting aspect is that, in contrast to Example 2 and many other protocols providing accountability, there is no central trusted party, all parties that provide evidence might deviate and thus lie.

The protocol has been implemented with hash functions and Zero-Knowledge Succinct Non-interactive Arguments of Knowledge (ZK-SNARKs). It is efficient on many examples; Kroll discusses applications ranging from credit scoring to various machine learning classifiers. Previous analysis [Kro15, Section 5.2.2] was informal and only considered holding the authority accountable. We discover that any subject can falsely accuse the authority A of misbehaviour, resulting in a situation reminiscent of Example 8. Such a claim by a single subject, e.g. an angry taxpayer, would subvert the trust into the system and hence render the accountability mechanism useless. We thus extended the protocol considerably to provide accountability w.r.t. knowledge-optimality.

Example 11 (Accountable algorithms). Assume the signature Σ to contain Σ_{sig} (see Example 1), pairs and the following sets:

$$\begin{aligned}\Sigma_{\text{ZK}} &:= \{ZK/9, verZK/1, Pub/1, true/0\}\\ \Sigma_{\text{comm}} &:= \{comm/2, open/2\}\\ \Sigma_{\text{channel}} &:= \{ch/2, retr/2, check/2\}\\ \Sigma_{\text{log}} &:= \{Init/1, Log/1, Final/0, readLog/1,\\ &\qquad isLog/1, readInit/1\}.\end{aligned}$$

Assume the following equations for zero knowledge proofs

$$\begin{aligned}Pub(ZK(C_x, C_y, C_z, x, y, z, r_x, r_y, r_z)) &= \langle C_x, C_y, C_z\rangle\\ verZK(ZK(C_x, C_y, C_z, x, y, z, r_x, r_y, r_z)) &= true,\end{aligned}$$

where $z = f(x, y)$, $C_x = comm(x, r_x)$, $C_y = comm(y, r_y)$ and $C_z = comm(z, r_z)$. Furthermore, the following equations for commitments, secure channels and logs:

$$\begin{aligned} open(comm(m, r), r) &= m & readLog(Log(l)) &= l\\ retr(x, ch(x, z)) &= z & readInit(Init(l)) &= l\\ check(x, ch(x, z)) &= true & isLog(Log(l)) &= true.\end{aligned}$$

Let $f \in \Sigma$ be a constructor that models an arbitrary function, the algorithm to be made accountable. The set of parties $\mathcal{P}$ consists of A, the authority, and a set of n subjects S_1 to S_n, as well as some bystander E which models a network adversary. Each subject S_i has some arbitrary but fixed input $x_i \in$ Terms which it signs and sends to A (see Figure 3). The authority, which has previously committed on some value y, computes the function f on each x_i and the value y which remains secret. It keeps a log that proves the authenticity of the commitments and the fact that f was indeed used to compute the resulting z. This z, along with information to open the commitments to x and z (but not y!) is then signed and sent to S_i. S_i checks the signature and uses this information to validate x and z with the commitments in the log. If those are correct, it checks the zero-knowledge proof in the log for consistency with the commitments to x and z, as well as the earlier comittment to y, to be sure that z was computed using f and said y. Only then S_i reports its input x_i and the output z received, along with the message signed by A, which is used to resolve disputes in case the log entries made by A and S_i are inconsistent. Formally, we define the protocol $\pi = (B, n)$ as follows:

$$\begin{aligned}B = {} & \nu sk_A; \nu sk_{S_1}; \cdots; \nu sk_{S_n}; \nu sk_L;\\ & \nu\ c_{A,S_1}, \ldots, c_{A,S_n}; \nu c_{S_1,A}, \ldots, c_{S_n,A};\\ & \{^{pk(sk_A)}/_{pk_A}, ^{pk(sk_{S_1})}/_{pk_{S_1}}, \ldots, ^{pk(sk_{S_n})}/_{pk_{S_n}}\};\\ & \cdot_A \mid \cdot_{S_1} \mid \cdots \mid \cdot_{S_n} \mid \cdot_E\end{aligned}$$

and $n(E) := \mathsf{out}(\langle pk_A, pk_{S_1}, \ldots, pk_{S_n}\rangle)$ so that $fv(n(E))$ contains all public keys. Names c_{A,S_i} and $c_{S_i,A}$ are used to model authenticated channel from authority to subjects and vice versa. The authority shall behave as follows:

$n(A) := \nu y,r_y;$ event $(Init(comm(y,r_y)));$
out $(comm(y,r_y));$ *(repeated n times)* ; $P_{A,1}$

where

$P_{A,i} :=$ in(m); if $check(c_{S_i,A}, m) = true$ then
if $versig(pk_{S_i}, sign, x) = true$ then
event$(Log(\langle sign, log\rangle))$;
out$(ch(c_{A,S_i}, sig(sk_A, \langle z, r_z, r_x\rangle)))$;
out$(Log(log))$; $P_{A,i+1}$

with

$$\begin{aligned} x &= \pi_1(retr(c_{S_i,A}, m)), & C_x &= comm(x, r_x),\\ sign &= \pi_2(retr(c_{S_i,A}, m)), & C_y &= comm(y, r_y),\\ z &= f(x, y), & C_z &= comm(z, r_z),\end{aligned}$$

$$log = \langle C_x, C_z, ZK(C_x, C_y, C_z, x, y, z, r_x, r_y, r_z)\rangle$$

and $P_{A,n+1} :=$ event $(Final)$. The subjects S_i, $i = 1, \ldots n$, have the following normative behaviour:

$n(S_i) :=$ in $(init)$; out$(ch(c_{S_i,A}, sig(sk_{S_i}, x_i)))$;
in(m_1); if $check(c_{A,S_i}, m_1) = true$ then
if $versig(pk_A, r, \langle z, r_z, r_x\rangle) = true$ then
in(m_2); if $C_x = comm(x_i, r_x)$ then
if $C_z = comm(z, r_z)$ then
if $\langle C_x, C_y, C_z\rangle = Pub(Z)$ then event$(\langle x_i, z, r\rangle)$

where

$$\begin{aligned} \langle z, r_z, r_x\rangle &= extract(r), & r &= retr(c_{A,S_i}, m_1),\\ \langle C_x, C_z, Z\rangle &= readLog(m_2), & C_y &= readInit(init).\end{aligned}$$

The subjects S_i need to be able to retrieve log entries correctly, hence we model the logging mechanism as a constraint on traces enforcing that the messages m_1 and m_2 received by $n(S_i)$ have matching entries in the log.

$$\begin{aligned} \alpha(t) :=& \forall l.(A, S_i, Log(l)) \in_E t \implies (A, Log(l)) \in_E t\\ & \wedge (A, S_i, Init(l)) \in_E t \implies (A, Init(l)) \in_E t\end{aligned}$$

Now we can formulate the goal of the protocol. Each party receives the correct output with respect to the input it claims to have sent, and the private information A has committed to with her first message. We model this violation separately for each party to facilitate analysis. Hence, the security property is $\varphi(t) = \bigwedge_{i\in\mathbb{N}_n} \varphi_i$, with

$$\begin{aligned} \varphi_i(t) :=& \alpha(t) \implies \forall C_y, r_y.(A, Final), (A, Init(C_y)) \in_E t\\ & \implies \exists x, y, x, r_x, r_y, r_z.\\ & (S_i, \langle x, z, r\rangle) \in_E t \wedge z =_E f(x, open(C_y, r_y))\\ & \wedge \forall (S_i, l) \in_E t.\ l =_E \langle x, z, r\rangle\end{aligned}$$

For simplicity, we focus on a single, but arbitrary, S_i and we assume only the claims of A and S_i are visible:

$$V := \{(A,t) \mid \exists t', f.t =_E f(t'), \\ f \in \{Init, Log\} \vee t =_E Final\} \cup \{(S_i,t)\}.$$

Simplifications. We have simplified the protocol in comparison to the original presentation [Kro15] by removing randomness generation for f. In the original protocol, A commits to a random seed which is then fed into a verifiable random function along with random bits from a public source. As the scope of this case study is in the decentralised accountability mechanism, we opted for a simpler presentation at the cost of generality. Kroll proposes an accountable green card lottery on the basis of a probabilistic f, which is an instance of an application that our simplification excludes. As previously mentioned, our modelling of the log via α is ad-hoc and we focus on individual correctness φ_i, as we want to emphasize the arguments concerning accountability.

Violation of accountability. On the other hand, we extended the protocol so that A and S_i authenticate the origin of their messages with signatures *sign* and r (see Figure 3). This is needed for accountability: Assume S_i decides to send a value $\hat{x}_i$ to A, which is different from the actual value x_i. S_i can claim to have sent x_i, and without these signatures, there is no way for A to show that S_i itself is behaving maliciously, similar to B's situation in Example 8. Similarly, if S_i brings forward a claim that A misbehaved, it needs to protect itself from unjustified claims by A that S_i's claim is based on false evidence, in essence: S_i needs to back its claims. Hence S_i requires a signature on z_i and other values.

This attack translates to the original protocol, which was apparently designed to only hold A accountable. As blind trust in the subjects to only submit correct claims would be the downfall of such a system in the real-world, we extend the protocol to provide authenticity of origin for whatever x_i the authority A receives, as to hold A *and* S_i accountable. Note that Kroll assumes an authenticated and secret channel between A and S_i, but this does not guarantee non-repudiation, which we need here.

Similar to many application-specific definitions, Kroll's modelling and analysis focuses on accountability as a property of the cryptographic algorithms executed by A and the subjects, thus disregarding the actual communication between (possibly deviating) parties and therefore neglecting the possibility that a subject can make false claims. This confirms the need for a formal notion of accountability such as the one we provide in this work. Our preliminary analysis pointed to this possible attack, as for an inconsistent log t with $(S_i,\hat{x}),(A,zk) \in t|_V$ with $zk = ZK(C_x,C_y,C_z,x,y,z,r_x,r_y,r_z)$, $verZK(zk) = true$ and $\hat{x} \neq_E x$, both a single deviating A and a single deviating S_i explain the visible part of this trace, if the signature were left out.

Note that our extension of the protocol comes with a loss of confidentiality: although most digital signature schemes do not reveal the signed message in full, they might leak some information about it. One could avoid disclosing the signature, e.g., by incorporating it within the zero-knowledge proof, however, we found a patch which stays close to the original protocol and its existing implementation more instructive within the context of accountability.

Security analysis Let us now define a verdict function which provides accountability for the repaired protocol. Similar to the security property, we have one verdict per subject. This is simpler to define, as otherwise the verdict would need to point out *all* misbehaving S_i at once. To achieve the same goal, these verdicts can be combined. Hence we define

$$verdict_i(t) := \begin{cases} \emptyset & \text{if } \varphi_i(t) \\ \{\mathrm{AC}_i \cup \mathrm{AD} \cup \mathrm{SC}_i \cup \mathrm{SD}_i\} & \text{otherwise,} \end{cases}$$

where AC_i validates the consistency of A's logs:

$$\begin{aligned} \mathrm{AC}_i := \{A \mid & \neg\exists C_x, C_y, C_z, r_x, Z, sign. \\ & (A, Init(C_y)), (A, Final), (A, Log(\langle sign, \langle C_x, C_z, Z\rangle) \in_E t \wedge \\ & \langle C_x, C_y, C_z\rangle =_E Pub(Z) \wedge verZK(Z) =_E true \wedge \\ & versig(pk(sk_{S_i}), sign, open(C_x, r_x)) =_E true\}. \end{aligned}$$

SC_i validates the consistency of S_i's claim with A's log: the claimed input and output should be consistent with the randomness A has signed, the commitments A has published; the input claimed should be the one previously signed by S_i.

$$\begin{aligned} \mathrm{SC}_i := \{S_i \mid & \neg\exists r, r_z, r_x, C_x, C_z, Z, sign. \\ & (S_i, \langle open(C_x, r_x), open(C_z, r_z), r\rangle) \in_E t \wedge \\ & (A, Log(\langle sign, \langle C_x, C_z, Z\rangle)) \in_E t \wedge \\ & versig(pk(sk_{S_i}), sign, open(C_x, r_x)) = true \wedge \\ & versig(pk(sk_A), r, \langle open(C_z, r_z), r_z, r_x\rangle) = true\}. \end{aligned}$$

SD_i and AD complement these consistency checks by verifying that S_i, respectively A, has not written contradicting entries to the log.

$$\begin{aligned} \mathrm{SD}_i := \{S_i \quad \mid \quad & \exists t_1 t_2.(S_i, t_1), (S_i, t_2) \in_E t \wedge t_1 \neq t_2\} \\ \mathrm{AD}_i := \{A \quad \mid \quad & \exists t_1 t_2.(S_i, Init(t_1)), (S_i, Init(t_2)) \in_E t \vee \\ & ((S_i, Log(t_1)), (S_i, Log(t_2)) \in_E t) \wedge t_1 \neq t_2 \wedge \\ & \exists m_1, m_2.versig(pk(sk_{S_i}), \pi_1(t_1), m_1) \\ & =_E versig(pk(sk_{S_i}), \pi_1(t_2), m_2) =_E true)\}. \end{aligned}$$

Theorem 2. *The verdict described above provides the protocol in Example 11 with accountability for φ_i w.r.t. knowledge-minimal deviations.*

Proof. Case distinction over an arbitrary but fixed t. Let t s.t. $\mathit{verdict}_i(t|_V) = \emptyset$. This is equivalent to $\mathrm{AC}_i = \mathrm{SC}_i = \mathrm{SD}_i = \mathrm{AD} = \emptyset$. Hence, there are $C_y, \mathit{sign}, C_x, C_z, Z$ such that $(A, \mathit{Init}(C_y))$, $(A, \mathit{Log}(\langle \mathit{sign}, \langle C_x, C_z, Z\rangle\rangle)$ and $(A, \mathit{Final}) \in_E t$, where $\mathit{verZK}(Z) =_E \mathit{true}$, and $\langle C_x, C_y, C_z\rangle =_E \mathit{Pub}(Z)\wedge$ thus $C_z = \mathit{comm}(f(x,y), r_z)$, $C_y = \mathit{comm}(y, r_z)$, $C_x = \mathit{comm}(x, r_x)$, for some x, y and r_y, r_z, r_x. Furthermore, from $\mathrm{SC}_i = \emptyset$ and $\mathrm{AD} = \emptyset$, we obtain $(S_i, \langle x', z, r\rangle) \in_E t$ with $\langle z, r'_z, r'_x\rangle = \mathit{extract}(r)$, such that $C_x =_E \mathit{comm}(x', r'_x)$, and $C_z =_E \mathit{comm}(z, r'_z)$, for some x', z, r, r'_z and r'_x. From $C_z =_E \mathit{comm}(z, r'_z)$, and $C_z = \mathit{comm}(f(x,y), r_z)$, we conclude that $r_z = r'_z$ and $z =_E f(x,y)$. From $C_x = \mathit{comm}(x, r_x)$, and $C_x =_E \mathit{comm}(x', r'_x)$, we conclude that $r_x = r'_x$ and $x =_E x'$. Hence $(S_i, \langle x, z, r\rangle) \in_E t$ with $z =_E f(x,y) =_E f(x, \mathit{open}(C_y, r_y))$. This shows that an empty verdict implies $\varphi(t|_V)$. The converse follows by definition of $\mathit{verdict}_i$.

Let t s.t. $\mathit{verdict}_i(t|_V) = \{A\}$. By definition, either $\mathrm{AC}_i = \{A\}$ or $\mathrm{AD} = \{A\}$, or both. In the first case, there is no consistent log by A, however, by $\mathrm{SC}_i = \emptyset$, we see that S_i's output is correct w.r.t. to the inconsistent log, and by SD_i, we see that it is unique. In the second case, A produced differing *Init*-events, in which case the violation is triggered immediately, or differing *Log*-events, in which case one of these was not received, which contradicts α, i.e., the assumption that $\neg\varphi_i(t)$ which is necessary for $\mathit{verdict}_i(t|_V) = \{A\}$. There is a knowledge-optimal deviation d and a context u that reproduce $t|_V$ (by Lemma 3), but $\mathit{dom}(d)$ might be any subset of parties containing A, as the events could contain subterms otherwise not derivable. However, no matter what the deviation, A can only communicate with parties that are not deviating by definition of relaxed deviations and context. For any violation resulting from $t|_V$, substituting all terms that are not deducible by A with a fresh name still yields a violation by definition of φ_i. Hence $(\pi, u) \vDash [d|_{\mathrm{S}'}]\neg\varphi$ for all $\mathrm{S}' \supseteq \{A, S_i\}$. As $\mathrm{SC}_i = \mathrm{SD}_i = \emptyset$, the observations produced by $d(S_i)$ are consistent with its normative behaviour, hence $(\pi, u) \vDash [d|_{\mathrm{S}'}]\neg\varphi$ for all $\mathrm{S}' \supseteq \{A\}$, but $(\pi, u) \vDash [d|_\emptyset]\varphi$ by definition of π. Hence, $\mathit{apv}_\varphi(\pi, u, d) = \{A\}$. There are no other knowledge-optimal deviation d' and context u' that produce the same visible trace but where A is not an element of some $\mathrm{S} \in \mathit{apv}_\varphi(\pi, u', d')$. Any such d' has A in its domain, since the normative behaviour is otherwise not able to produce any of the observations implied by AC_i or AD_i. But as $\mathrm{SC}_i = \mathrm{SD}_i = \emptyset$ for all $t \in \mathit{trace}(\pi[d'], u')$, and $d'(S_i)$ does not communicate with any other party (see above), $\mathrm{SC}_i = \emptyset$ also holds for all $t' \in \mathit{trace}(\pi[d'|_{\mathit{dom}(d')\setminus\{A\}}], u'')$, for all $u'' =_{\{A\}} u'$. Hence, $(S_i, \langle x, z, r\rangle) \in t'$ and $(A, \mathit{Log}(\langle \mathit{sign}, \langle C_x, C_z, Z\rangle\rangle), \in_E t$ with $x = \mathit{extract}(\mathit{sign})$, $\mathit{versig}(\mathit{pk}(sk_{S_i}), \mathit{sign}, x) = \mathit{true}$, $\mathit{versig}(\mathit{pk}(sk_A), r, \langle z, r_z, r_x\rangle)$, $\mathit{open}(C_x, r_x) = x$, and $\mathit{open}(C_z, r_z) = z$. By definition of $n(A)$, sign witnesses that A indeed received x and thus computed z such that $z =_E f(x, \mathit{open}(C_y, r_y))$ for the values r_y and C_y given upon initialisation and finalisation. Together with the statement implied by $\mathrm{SD}_i = \emptyset$, we conclude $\varphi(t')$. Thus A needs to be an element of any $\mathrm{S} \in \mathit{apv}_\varphi(\pi, u', d')$, for all d' and u' producing the same visible trace, as otherwise this S would be insufficient. For this reason, the verdict $\{\{A\}\}$ is the unique minimum.

Let t s.t. $\mathit{verdict}_i(t|_V) = \{S_i\}$. By definition, either $\mathrm{SC}_i = \{S_i\}$ or $\mathrm{SD}_i = \{S_i\}$, or both. In the first case, S_i's claim is inconsistent with the log, but by $\mathrm{AC}_i = \mathrm{AD}_i = \emptyset$, the A's log itself is consistent, including $(A, \mathit{Log}(\langle \mathit{sign}, \langle C_x, C_z, Z\rangle) \in_E t$ with sign signed by sk_{S_i}. In the second case, S_i produced differing claims, in which case a violation is triggered immediately. There is a knowledge-optimal deviation d and a context u that reproduce $t|_V$ (see Lemma 3), but $\mathit{dom}(d)$ might be any subset of parties, as the events could contain information otherwise not derivable. However, no matter what the deviation, A can only communicate with parties that are not deviating by definition of relaxed deviations and context. For any violation resulting from $t|_V$, substituting all terms that are not deducible by S_i with a fresh name still yields a violation by definition of φ_i Hence $(\pi, u) \vDash [d|_{\mathrm{S}'}]\neg\varphi$ for all $\mathrm{S}' \supseteq \{A, S_i\}$. As $\mathrm{AC}_i = \mathrm{AD}_i = \emptyset$, the observations produced by $d(A)$ are consistent with its normative behaviour, hence $(\pi, u) \vDash [d|_{\mathrm{S}'}]\neg\varphi$ for all $\mathrm{S}' \supseteq \{S_i\}$, but $(\pi, u) \vDash [d|_\emptyset]\varphi$ by definition of π. Hence, $\mathit{apv}_\varphi(\pi, u, d) = \{S_i\}$. There are no other knowledge-optimal deviation d' and context u' that produce the same visible trace but where S_i is not an element of some $\mathrm{S} \in \mathit{apv}_\varphi(\pi, u', d')$. Any such d' has S_i in its domain, since the normative behaviour is otherwise not able to produce any of the observations implied by SC_i or SD_i. But as $\mathrm{AC}_i = \mathrm{AD}_i = \emptyset$ for $t \in \mathit{trace}(\pi[d'], u')$, and $d'(A)$ does not communicate with any other party (see above), $\mathrm{AC}_i = \emptyset$ also holds for $t' \in \mathit{trace}(\pi[d'|_{\mathit{dom}(d')\setminus\{A\}}], u'')$, for all $u'' =_{\{A\}} u'$. Hence for all C_x, C_y, C_z, r_x, r_y, $(A, \mathit{Init}(C_y)), (A, \mathit{Final}) \in_E t'$, as well as $(A, \mathit{Log}(\langle \mathit{sign}, \langle C_x, C_z, Z\rangle) \in_E t'$ with $\langle C_x, C_y, C_z\rangle =_E \mathit{Pub}(Z)$, $\mathit{verZK}(Z) =_E \mathit{true}$ $\mathit{versig}(\mathit{pk}(sk_{S_i}), \mathit{sign}, \mathit{open}(C_x, r_x)) =_E \mathit{true}$. By Definition of α within φ, either $\varphi(t')$ is trivially true, or $n(S_i)$ indeed receives $\langle \mathit{sign}, \langle C_x, C_z, Z\rangle$ (which is unique by $\mathrm{AD} = \emptyset$). From the definition of $n(S_i)$ it follows that if S_i emits any event, it emits $\langle x_i, z, r\rangle$ where z is such that $C_z = \mathit{comm}(z, r_z)$ and $C_x = \mathit{comm}(x_i, r_x)$ holds for x_i. From $\mathit{verZK}(Z) =_E \mathit{true}$ we can therefore conclude that $z = f(x_i, y^*)$ for some y^* and r_y such that $C_y = \mathit{comm}(y^*, r_y)$, e.g., $z = f(x_i, \mathit{open}(C_y, r_y))$. This shows that $\varphi(t')$. Thus S_i needs to be an element of any $\mathrm{S} \in \mathit{apv}_\varphi(\pi, u', d')$, for all d' and u' producing the same visible trace, as otherwise this S would be insufficient. For this reason, the verdict $\{\{S_i\}\}$ is the unique minimum.

Let t s.t. $\mathit{verdict}_i(t|_V) = \{A, S_i\}$. In this case, it is trivial to find a knowlege-optimal deviation d with a context u that reproduces $t|_V$. No matter what the deviation, communication occurs only between S_j, $j \in \mathbb{N}_n$ and A, either by definition of relaxed deviations and context, or by n. For any violation resulting from $t|_V$, substituting all terms that are not deducible by S_i with a fresh name still yields a violation by definition of φ_i. Hence $(\pi, u) \vDash [d|_{\mathrm{S}'}]\neg\varphi$ for all $\mathrm{S}' \supseteq \{A, S_i\}$. We proceed to show that all knowledge-optimal deviations produce the same a posteriori verdict. There are no knowledge-optimal deviation d' and context u' that produce the same visible trace but where $\{A, S_i\}$ is not a subset of some $\mathrm{S} \in \mathit{apv}_\varphi(\pi, u', d')$. Any such d' has S_i and A in its domain, since the normative behaviour is otherwise not able to produce any of the observations implied by SC_i or SD_i, and AC_i or AD, respectively. But as $\mathrm{AC}_i \neq \emptyset \vee \mathrm{AD}_i \neq \emptyset$

and $\mathrm{SC}_i \neq \emptyset \vee \mathrm{SD}_i \neq \emptyset$ for all $t \in \mathit{trace}(\pi[d'], u')$, and $d'(A)$ and $d'(S_i)$ both do not communicate with any other party or each other, the same holds for any $t' \in \mathit{trace}(\pi[d'|_{dom(d')\setminus\{S_i,A\}}], u'')$, for all $u'' =_{\{A\}} u'$. In the previous two cases, we have argued that, for any knowledge-optimal deviation and context d'', u'' *a)* if $\mathrm{AC}_i \neq \emptyset \vee \mathrm{AD} =\neq \emptyset$, but $\mathrm{SC}_i = \mathrm{SC}_i = \emptyset$, A is an element of all $\mathrm{S} \in \mathit{apv}_\varphi(\pi, u'', d'')$. *b)* if $\mathrm{SC}_i \neq \emptyset \vee \mathrm{SD}_i =\neq \emptyset$, but $\mathrm{AC}_i = \mathrm{AC} = \emptyset$, S_i is an element of all $\S \in \mathit{apv}_\varphi(\pi, u'', d'')$. This d'' and u'' can be instantiated with $d'|_{dom(d)\setminus\{S_i\}}$ and some $u'' =_{\{S_i\}} u$, or $d'|_{dom(d)\setminus\{A\}}$ and some $u'' =_{\{A\}} u$, respectively. Hence applying the previous arguments conjunctively, we conclude that $\{A, S_i\}$ is a subset of all parts of a posteriori verdicts for relaxed deviations and contexts such that their trace agree with t. □

6 Conclusion and future work

We have presented a definition of accountability that presumes parties to behave optimally. We found verdict-optimality to have a sound interpretation, but not applicable in scenarios where several parties may be blamed at once. Knowledge-optimality is useful in this case, it is applicable to real protocols and helped identifying attacks in both case studies. In our opinion, the most promising solution to the dilemma between completeness and applicability implied by provocation are optimality notions that provide guarantees for non-optimal adversaries, like verdict-optimality does. We hope other optimality notions with similar guarantees can be identified in the future. Further assurance in our definition can be reached by connecting it to definitions of cause trace by restricting intervention on protocol events to similar events produced by the normative behaviour. In terms of Pearl's causality framework, e.g., this would correspond to a more precise definition of the range of a variable. The technical challenge lies in the dynamic nature of this range, as it is responsive to other interventions as well as the scheduling.

Another open question is whether our approach can be used to reason about accountability w.r.t. equivalence properties, in particular with regard to the computational setting. While a computational variant of our definition could be derived from our notion based on trace-properties (see Küsters et.al. [KTV10]), cryptographers favour indistinguishability notions. A viable approach may be to transform real-or-random games so the adversary only wins if he wins the game *and* the scheme produces an incorrect verdict, but this remains to be shown.

For practical use, we found that modelling a global log via events was tedious. A model in which a central trusted party is gathering the logs and checks the *absence* of protocol messages, could detect the adversary misbehaving by stalling progress, i.e., verify timeliness or accountability w.r.t. to timeliness. We wish to integrate our definition with recent work on liveness properties in security protocols [BDKK17], which provides a protocol calculus with resilient channels, local progress and a way to model timeouts.

Finally, handwritten proofs are very tedious to conduct and much of the work is similar to traditional protocol verification. We conjecture the existence of suf-

ficient conditions for accountability for some class of protocol and verdicts. The notion of relaxed deviations required for knowledge-optimality appears compatible with the prevalent paradigm of approximating dishonest parties by a single adversary that takes control over them. Then, e.g., minimality can be shown by restricting the a priori knowledge of this adversary and verifying impossibility of deriving any trace producing the verdict in question.

References

AF01. Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *28th ACM Symp. on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM, 2001.

Ano17. Anonymized. Accountability in security protocols. Technical report, 2017. available at `http://upload.preliminary.bplaced.net/accountability-long.pdf`.

ASW98. N. Asokan, Victor Shoup, and Michael Waidner. Asynchronous protocols for optimistic fair exchange. In *IEEE Symposium on Security and Privacy (S&P'98)*, pages 86–99. IEEE Comp. Soc., 1998.

BCS05. Michael Backes, Jan Camenisch, and Dieter Sommer. Anonymous yet accountable access control. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES 2005*, pages 40–46. ACM, 2005.

BDKK17. Michael Backes, Jannik Dreier, Steve Kremer, and Robert Künnemann. A novel approach for reasoning about liveness in cryptographic protocols and its application to fair exchange. In *Proceedings of the 2nd IEEE European Symposium on Security and Privacy (Euro S&P '17)*. IEEE Computer Society, 2017.

BFM13. Michael Backes, Dario Fiore, and Esfandiar Mohammadi. Privacy-preserving accountable computation. In *18th European Symposium on Research in Computer Security*, pages 38–56. Springer, 2013.

BGK17. Michael Backes, Deepak Garg, and Robert Künnemann. Capturing causality in security protocols. Technical report, Saarland University, 2017.

Bla02. Bruno Blanchet. From Secrecy to Authenticity in Security Protocols. In *9th International Static Analysis Symposium (SAS'02)*, pages 342–359. Springer, September 2002.

Bla09. Bruno Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 17(4):363–434, July 2009.

DGK+15. Anupam Datta, Deepak Garg, Dilsun Kaynar, Divya Sharma, and Arunesh Sinha. Program actions as actual causes: A building block for accountability. In *2015 IEEE 28th Computer Security Foundations Symposium*, pages 261–275. IEEE, 2015.

FJW11. Joan Feigenbaum, Aaron D. Jaggard, and Rebecca N. Wright. Towards a formal model of accountability. In *Proceedings of the 2011 New Security Paradigms Workshop*, NSPW '11, pages 45–56. ACM, 2011.

GM15. Gregor Gößler and Daniel Le Métayer. A general framework for blaming in component-based systems. *Sci. Comput. Program.*, 113:223–235, 2015.

Hal15. Joseph Y. Halpern. A modification of the halpern-pearl definition of causality. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 3022–3033. AAAI Press, 2015.

HKD07. Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. Peerreview: Practical accountability for distributed systems. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 175–188. ACM, 2007.

HP13. Joseph Y. Halpern and Judea Pearl. Causes and explanations: A structural-model approach — part 1: Causes. *CoRR*, abs/1301.2275, 2013.

Hum. David Hume. *An Enquiry concerning Human Understanding.*

JJPR09. Radha Jagadeesan, Alan Jeffrey, Corin Pitcher, and James Riely. Towards a theory of accountability and audit. In *Computer Security–ESORICS 2009*, pages 152–167. Springer, 2009.

Kro15. Joshua A. Kroll. *Accountable Algorithms.* PhD thesis, Princeton University, 2015.

KTV10. Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and relationship to verifiability. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pages 526–535. ACM, 2010.

Lew73. David Lewis. Causation. *Journal of Philosophy*, 70(17):556–567, 1973.

Pea00. Judea Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, New York, NY, USA, 2000.

PP11. Nick Papanikolaou and Siani Pearson. A cross-disciplinary review of the concept of accountability. In *Proceedings of the DIMACS/BIC/A4Cloud/CSA International Workshop on Trustworthiness, Accountability and Forensics in the Cloud (TAFC)*, 2011.

A Operational semantics

In this section we define the denotational semantics of our calculus for extended processes that do not contain holes, but in which the top-most process of any plain process is annotated with an *effectuating party*. Initially (cf. Definition 4), the top-most subprocess of every plain-process inserted at a hole operator $\cdot_p$ is annotated with the party p, but scope restrictions are skipped. As we will see, only plain processes are annotated, but never scope restriction.

Structural equivalence is the smallest relation closed under α-conversion of names and variables, as well as application of evaluation contexts[5], such that:

PAR-0	$A_{p_A} \mid 0_{p_B}$	$\equiv A_{p_A}$
PAR-C	$A_{p_A} \mid B_{p_B}$	$\equiv B_{p_B} \mid A_{p_A}$
PAR-A	$A_{p_A} \mid (B_{p_B} \mid C_{p_C})$	$\equiv (A_{p_A} \mid B_{p_B}) \mid C_{P_C}$
NEW-0	$\nu u; 0_{p_A}$	$\equiv 0_{p_A}$
NEW-C	$\nu u \nu v; 0_{p_A}$	$\equiv \nu v \nu u; 0_{p_A}$
NEW-P	$A_{p_A} \mid \nu u; B_{p_B}$	$\equiv \nu u; (A_{p_A} \mid B_{p_B})$ (if $u \notin fv(A) \cup fn(A)$)
ALIAS	0_{p_A}	$\equiv \nu x; \{^M/_x\}$
SUBST	$\{^M/_x\} \mid A_{p_A}$	$\equiv \{^M/_x\} \mid A'_{p_A}$ (where $A' = A\{^M/_x\}$)
REWR	$\{^M/_x\}$	$\equiv \{^N/_x\}$ if $M =_E N$

(For brevity, A_p or B_p can stand for an unannotated process, in which case $p = \bot$). Like in the applied pi calculus, we always assume that active substitutions are cycle-free, and that there is at most one active substitution for each variable in an extended process. Furthermore, there is exactly one active substitution when the variable is restricted. Later on, we will use variables to identify transmitted messages, hence we assume all variables to be unique from the start and define $P\{^m/_x\}$ as usual, but leave it undefined whenever P contains a subprocess of form $\mathsf{in}(x); P'$ to avoid dynamic renaming of variables. Using these rules, every closed extended process A can be brought into form [AF01]: $A \equiv \nu n_1; \ldots \nu n_m; (\{^{m_1}/_{x_1}\} \mid \ldots \mid \{^{m_l}/_{x_l}\} \mid P_1 | \cdots | P_k)$, where $P_1, \ldots, P_k$ are closed plain processes, i.e., all variables are bound or defined by an active substitution. (The proof to Lemma 1 precises this statement w.r.t. the annotations.)

Internal reduction is the smallest relation on extended processes closed by structural equivalence and application of evaluation contexts such that:

THEN	$(\mathsf{if}\ t_1 = t_2\ \mathsf{then}\ P\ \mathsf{else}\ Q)_{p_A}$	$\xrightarrow{p_A} P_{p_A}$ if $t_1 =_E t_2$
ELSE	$(\mathsf{if}\ t_1 = t_2\ \mathsf{then}\ P\ \mathsf{else}\ Q)_{p_A}$	$\xrightarrow{p_A} Q_{p_A}$ if $t_1 \neq_E t_2$
COMM	$(\mathsf{out}(x); P)_{p_A} \mid (\mathsf{in}(x); Q)_{p_B}$	$\xrightarrow{(p_A, p_B, x)} P_{p_A} \mid Q_{p_B}$
EVENT	$(\mathsf{event}\ m; P)_{p_A}$	$\xrightarrow{(p_A, m)} P_{p_A}$

W.l.o.g., $p_A, p_B \neq \bot$, as structural equivalence preserves that the top-most non-ν position of any plain process remains annotated with an effectuating party.

[5] Evaluation contexts are defined by the grammar

$$\langle C \rangle ::= \cdot \mid \nu n; \langle C \rangle \mid \nu x; \langle C \rangle \mid (A \mid \langle C \rangle) \mid (\langle C \rangle \mid A).$$

Along with Alias and Subst, Comm permits the transmission of terms: Assume $x \notin fv(M) \cup fv(P)$, then

$$\begin{aligned}(\mathsf{out}(t); P)|(\mathsf{in}(x); Q) &\equiv \nu x; (\{^t/_x\}|(\mathsf{out}(x); P)|(\mathsf{in}(x); Q)) \\ &\to \nu x; (\{^t/_x\}|P \mid Q) \equiv P \mid Q\{^t/_x\}.\end{aligned}$$

Hence, we write $A \xrightarrow{(P_A, P_B, m)} B$ if $A \xrightarrow{(P_A, P_B, x)} B$ and x is in scope of an active substitution $\{^m/_x\}$ in A.