A Cut Principle for Information Flow

Joshua D. Guttman and Paul D. Rowe
The MITRE Corporation

***Abstract*—We view a distributed system as a graph of active locations with unidirectional channels between them, through which they pass messages. In this context, the graph structure of a system constrains the propagation of information through it.**

Suppose a set of channels is a cut set between an information source and a potential sink. We prove that, if there is no disclosure from the source to the cut set, then there can be no disclosure to the sink. We introduce a new formalization of partial disclosure, called *blur operators*, and show that the same cut property is preserved for disclosure to within a blur operator. A related compositional principle ensures limited disclosure for a class of systems that differ only beyond the cut.

I. Introduction

In this paper, we consider information flow in a true-concurrency, distributed model. Events in an execution may be only partially ordered, and locations communicate via synchronous message-passing. Each message traverses a channel. The locations and channels form a directed graph.

Evidently, the structure of this graph constrains the flow of information. Distant locations may have considerable information about each other's actions, but only if the information in intermediate regions accounts for this. If a kind of information does not traverse the boundary of some portion of the graph (a *cut set*), then it can never be available beyond that. We represent these limits on disclosure, i.e. kinds of information that do not escape, using *blur operators*. A blur operator returns a set of behaviors local to the information source; these should be indistinguishable to the observer. Blur operators formalize the semantic content of limited disclosures, and they cover similar ground to other forms of *what*-dimension declassification [51], [52]. Their definition, however, identifies the principles that localize information flow.

When disclosure from a source to a cut set is limited to within a blur operator, then disclosure to a more distant region is limited to within the same blur operator (see Thm. 28, the *cut-blur* principle). The cut-blur principle combines our *what*-dimension declassification with a *where*-dimension perspective. It gives a criterion that localizes those disclosure limits within a system architecture.

A related result, Thm. 32, supports *compositional* security. Consider any other system that differs from a given one only in its structure beyond the cut. That system will preserve the flow limitations of the first, assuming that it has the same local behaviors as the first in the cut set. We illustrate this (Examples 33–34) to show that secrecy and anonymity properties of a firewall and a voting system are preserved under some environmental changes. Flow properties of a simple system remain true for more complex systems, if the latter do not distort behavior at the edge of the simple system.

Our model covers many types of systems, including networks, software architectures, virtualized systems, and distributed protocols such as voting systems. Network examples, which involve little local state, are easy to describe, and rely heavily on the directed graph structure. Blur operators highlight their security goals as information-flow properties. Voting systems offer an interesting notion of limited disclosure, since they must disclose the result but not the choices of the individual voters. Their granularity encourages composition, since votes are aggregated from multiple precincts.

Motivation. A treatment of information flow that relies on the graph structure of distributed systems facilitates compositional security design and analysis.

Many systems have a natural graph structure, which is determined early in the design process. Some are distributed systems where the components are on separate platforms, and the communication patterns are a key part of their security architectures. In other cases, the components may be software, such as processes or virtual machines, and the security architecture is largely concerned with their communication patterns. The designers may want to validate that these communication patterns support the information flow goals of the design early in the life cycle. Thm. 32 justifies the designers in concluding that a set of eventual systems all satisfy these security goals, when those systems all agree on "the part that really matters."

Contributions of this paper. Our main result is the cut-blur principle, Thm. 28, which Thm. 32 brings to compositional form. The definition of *blur operator* is a supplementary contribution. We show that any reasonable notion of partial disclosure satisfies the conditions for a blur (Lemma 22). We regard these simple structural conditions as giving the "logical form" of composable limited disclosure. The conditions lead to very clean proofs of Thms. 28, 32.

Structure of this paper. After discussing motivating examples (Section II) and some related work (Section III), we introduce our systems, called *frames*, and their execution model in Section IV. In this static model, the channels connecting different locations do not change during execution. Section V proves the cut-blur principle for the simple case of no disclosure of information at all across the boundary.

Section VI formalizes partial disclosure via blur operators, and Section VII extends the cut idea to blurs (Thms. 28, 32).

Section VIII provides rigorous results to relate our model to the literature. We end by indicating some future directions. Appendix A contains longer proofs, and additional lemmas.

DOI 10.1109/CSF.2015.15

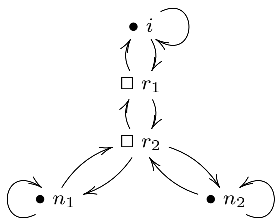

Fig. 1. A Two-Router Firewall

II. Two Motivating Examples

We first propose two problems we view in terms of information flow. One is about network filtering; the other concerns anonymity in voting. In each, we want to prove an information flow result once, and then reuse it compositionally under variations that do not affect the core mechanism itself.

Example 1 (Network filtering)**.** Fig. 1 shows a two-router firewall separating the public internet (node i) from two internal network regions n_1, n_2. The firewall should ensure that any packet originating in the internal regions n_1, n_2 reaches i only if it satisfies some property of its source and destination addresses, protocol, and port (etc.); we will call these packets *exportable*. Likewise, any packet originating in i reaches n_1, n_2 only if it satisfies a related property of its source and destination addresses, protocol, and port (etc.); we will call these packets *importable*.

These are information flow properties. The policy provides *confidentiality* for non-exportable packets within n_1, n_2, ensuring that they are not observable at i. It provides a kind of *integrity* protection for n_1, n_2 from non-importable packets from i, ensuring that n_1, n_2 cannot be damaged, or affected at all, if they are malicious.

We assume here that packets are generated independently, so that (e.g.) no process on a host in n_1, n_2 generates exportable packets encoding confidential non-exportable packets it has sent or received. If some process on a host is observing packets and coding their contents into packets to a different destination, this is a problem firewalls were not designed to solve, and security administrators worry about it separately.

A firewall configuration enforcing a flow goal against the internet viewed as a single node i should still succeed if i has internal structure. Similarly, the internal regions n_1, n_2 may vary without risk of security failure. ///

We will return to this example several times to illustrate how we formalize the system and specify its flow goals. Example 33 proves that some information flow goals of Fig. 1 remain true as the structure of i, n_1, n_2 varies.

Example 2. As another key challenge, consider an electronic voting system such as ThreeBallot [42]. Fig. 2 shows the voters $v_1, \ldots, v_k$ of a single precinct, their ballot box BB_1, a channel delivering the results to the election commission EC, and then a public bulletin board Pub that reports the results.

The ballot box should provide voter anonymity: neither EC nor anyone observing the results Pub should be able to

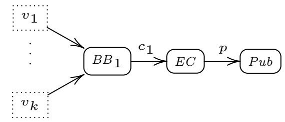

Fig. 2. A single precinct

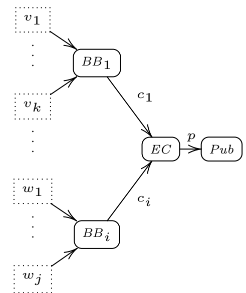

Fig. 3. Multiple precincts report to EC

associate any particular vote with any particular voter v_i. This also is an information flow goal.

However, elections generally concern many precincts. Fig. 3 contains i precincts, all connected to the election commission EC. Intuitively, a voter v_n cannot lose their anonymity in the larger system: BB_1 has already anonymized the votes in this first precinct. Accumulating precinct summaries at EC cannot change the causal consequences of BB_1's actions. ///

We formalize the flow goals of this example in Example 26, and justify Fig. 3 in Example 34.

These simple examples illustrate the payoff from a compositional approach to flow goals. Conclusions about a firewall should be insensitive to changes in the structure of the networks to which it is attached. An anonymity property achieved by a ballot mechanism should be preserved as we collect votes from many precincts. These are situations where we want to design, justify, and then reuse mechanisms, with a criterion ensuring the mechanisms remain safe under changes outside them. Thm. 32 below is the criterion we propose.

III. Some related work

Noninterference and nondeducibility. There is a massive literature on information-flow security; Goguen and Meseguer were key early contributors [20]. Sutherland introduced the non-deducibility idea [53] as a way to formalize lack of information flow, which we have adopted in our "non-disclosure" (Def. 11). Subsequent work has explored a wide range of formalisms, including state machines [47]; process algebras such as CSP [44], [43], [48] and CCS [18], [19], [6]; and bespoke formalisms [36], [29].

Irvine, Smith, and Volpano reinvigorated a language-based approach [25], inherited from Denning and Denning [16], in which systems are programs. Typing ensures that their behaviors satisfy information-flow goals; cf. [50]. Distributed execution has been considered also, e.g. [58], [9], [5]. Our work here is not specifically language-based, since the behaviors of our locations are sets of traces, not necessarily specified by programs. Moreover, language-based work emphasizes information flows from certain inputs to outputs, where the system is often regarded as a function. Our systems need not have any particular inputs, and information flow concerns the correlation of behaviors in different regions.

Declassification. Declassification is a major concern for us. A blur operator (Def. 21) determines an upper bound on what a system may declassify. It may declassify information unless its blur operators require those details to be blurred out. Like escape-hatches [51] or relaxed noninterference [28], this is disclosure along the *what*-dimension, in the Sabelfeld-Sands classification [52]. The cut-blur principle connects this *what* declassification to *where* the processing responsible for the declassification will occur in a system architecture. In this regard, it combines a semantic view of what information is declassified with an architectural view related to intransitive noninterference [47], [54]. Balliu et al. [4] connect *what*, *where*, and *when* declassification via epistemic logic, although without a compositional method.

McCamant and Ernst [34] study quantitative information flow when programs run. A directed acyclic graph representing information flow is generated dynamically from a particular execution or set of executions. The max-flow/min-cut theorem bounds flow in those runs by what can traverse minimal cuts. Apparently, other possible executions may not respect the bounds. Their flow conclusions are not compositional.

Composability and refinement. McCullough first raised the questions of non-determinism and composability of information-flow properties [35], [36]. This was a major focus of work through much of the period since, persisting until today [27], [57], [30], [31], [46], [41]. Mantel, Sands, and Sudbrock [32] use a rely/guarantee method for compositional reasoning about flow in the context of imperative programs. Roscoe [45], [44] offers a definition based on determinism, which is intrinsically composable. Morgan's [39] programming language treatment clarifies the refinements that preserve security. Our results do not run afoul of the refinement paradox either [26], [39]: our theorems identify the assumptions that ensure that blurs are preserved.

Van der Meyden [55] provides an architectural treatment designed to achieve preservation under refinement. Our work is distinguished from it in offering a new notion of composition, illustrated in Examples 1–2; in focusing on declassification; and in applying uniformly to a range of declassification policies, defined by the blur operators.

Van der Meyden's work with Chong [10], [11] is most closely related to ours. They consider "architectures," i.e. directed graphs that express an intransitive noninterference style of *what*-dimension flow policy. The nodes of an architecture are security domains, intended to represent levels of information sensitivity. The authors define when a (monolithic) deterministic state machine, whose transitions are annotated by domains, *complies* with an architecture. The main result in [10] is a cut-like epistemic property on the architecture graph: Roughly, any knowledge acquired by a recipient about a source implies that the same knowledge is available at every cut set in the architecture graph.

A primary contrast between this paper and [10] is our distributed execution model. We consider it a more localized link to development, since components are likely to be designed, implemented, and upgraded piecemeal. Chong and van der Meyden focus instead on the specifications, in which sensitivity levels of information (rather than active system components) form the directed graph. This new and unfamiliar specification is needed before analysis. Their epistemic logic allows nested occurrences of the *knowledge* modality K_G, or occurrences of K_G in the hypothesis of an implication. However, this surplus expressiveness is not used in their examples, which do not have nested K_G operators, or occurrences of K_G in the hypothesis of an implication. Indeed, our clean proof methods suggest that our model may have the right degree of generality, and be easy to understand, apply, and enrich.

Recently [11], they label the arrows by functions f, where f filters information from its source, bounding visibility to its target. They have not re-established their cut-like epistemic property in the richer model, however. Van der Meyden and Chong's refinement method [55], [11] applies when the refined system has a homomorphism *onto* the less refined one. It covers Example 1 but not Example 2, where the refined system contains genuinely new components and events.

We return to related work *passim*, and in Sections VIII–IX.

IV. Frames and Executions

We represent systems by *frames*. Each frame is a directed graph. Each node, called a *location*, is equipped with a set of traces defining its possible local behaviors. The arrows are called *channels*, and allow the synchronous transmission of a message from the location at the arrow tail to the location at the arrow head. Each message also carries some *data*.

A. A Static Model

In this paper, we will be concerned with a static version of the model, in which channel endpoints are never transmitted from one location to another. Section IX mentions a dynamic alternative, in which these endpoints may be delivered over other channels. Each frame uses three disjoint domains:

Locations $\mathcal{LO}$: Each location $\ell \in \mathcal{LO}$ is equipped with a set of traces, $\mathsf{traces}(\ell)$ and other information, further constrained below.

Channels $\mathcal{CH}$: Each channel $c \in \mathcal{CH}$ is equipped with two endpoints, $\mathsf{entry}(c)$ and $\mathsf{exit}(c)$. It is intended as a one-directional conduit of data values between the endpoints.

Data values $\mathcal{D}$: Data values $v \in \mathcal{D}$ may be delivered through channels.

We will write $\mathcal{EP}$ for the set of channel endpoints, which we formalize as $\mathcal{EP} = \{\mathsf{entry}, \mathsf{exit}\} \times \mathcal{CH}$, although we generally write $\mathsf{entry}(c)$ and $\mathsf{exit}(c)$ to stand for $\langle \mathsf{entry}, c\rangle$ and $\langle \mathsf{exit}, c\rangle$.

A frame $\mathcal{F}$ supplies sets of endpoints $\mathsf{ends}(\ell)$ and $\mathsf{traces}(\ell)$ for each location $\ell \in \mathcal{LO}$. When $\mathsf{entry}(c) \in \mathsf{ends}(\ell)$ we write $\mathsf{sender}(c) = \ell$; when $\mathsf{exit}(c) \in \mathsf{ends}(\ell)$ we write $\mathsf{rcpt}(c) = \ell$. Thus, $\mathsf{sender}(c)$ can send messages on c, while $\mathsf{rcpt}(c)$ can receive them. We write $\mathsf{chans}(\ell)$ for $\{c\colon \mathsf{sender}(c) = \ell$ or $\mathsf{rcpt}(c) = \ell\}$.

We say that λ is a *label* for ℓ if $\lambda = (c, v)$ where $c \in \mathsf{chans}(\ell)$ and $v \in \mathcal{D}$; and we categorize labels c, v as:

local to ℓ if $\mathsf{sender}(c) = \ell = \mathsf{rcpt}(c)$;
a transmission for ℓ if $\mathsf{sender}(c) = \ell \neq \mathsf{rcpt}(c)$;
a reception for ℓ if $\mathsf{sender}(c) \neq \ell = \mathsf{rcpt}(c)$.

With this notation we define frames:

Definition 3. Given domains $\mathcal{LO}, \mathcal{CH}, \mathcal{D}$, $\mathcal{F} = (\mathsf{ends}, \mathsf{traces})$ is a *frame* iff, for each $\ell \in \mathcal{LO}$:

1. $\mathsf{ends}(\ell) \subseteq \mathcal{EP}$ is a set of endpoints such that
   (a) $\langle e, c\rangle \in \mathsf{ends}(\ell)$ and $\langle e, c\rangle \in \mathsf{ends}(\ell')$ implies $\ell = \ell'$; and
   (b) there is an ℓ such that $\mathsf{entry}(c) \in \mathsf{ends}(\ell)$ iff there is an ℓ' such that $\mathsf{exit}(c) \in \mathsf{ends}(\ell')$;
2. $\mathsf{traces}(\ell)$ is a prefix-closed set, each trace $t \in \mathsf{traces}(\ell)$ being a finite or infinite sequence of labels λ. ///

In this definition, we do not require that the local behaviors $\mathsf{traces}(\ell)$ should be determined in any particular way. They could be specified by associating a program to each location, or a term in a process algebra, or a labeled transition system, or a mixture of these for the different locations.

Each $\mathcal{F}$ determines directed and undirected graphs:

Definition 4. If $\mathcal{F}$ is a frame, then the *graph of* $\mathcal{F}$, written $\mathsf{gr}(\mathcal{F})$, is the directed graph (V, E) whose vertices V are the locations $\mathcal{LO}$, and such that there is an edge $(\ell_1, \ell_2) \in E$ iff, for some $c \in \mathcal{CH}$, $\mathsf{sender}(c) = \ell_1$ and $\mathsf{rcpt}(c) = \ell_2$.

The undirected graph $\mathsf{ungr}(\mathcal{F})$ has those vertices, and an undirected edge (ℓ_1, ℓ_2) whenever either (ℓ_1, ℓ_2) or (ℓ_2, ℓ_1) is in the edges of $\mathsf{gr}(\mathcal{F})$. ///

B. Execution semantics

The execution model for frames uses partially ordered sets of events. The key property is that the events at any single location ℓ should be in $\mathsf{traces}(\ell)$. Our semantics is reminiscent of Mattern [33], although his model lacks the underlying graph structure. We require executions to be well-founded, but no later results in this paper depend on that.

Definition 5 (Events; Executions)**.** Let $\mathcal{F}$ be a frame, and let $\mathcal{E}$ be a structure $\langle E, \mathsf{chan}, \mathsf{msg}\rangle$. The members of E are *events*, equipped with the functions:

$\mathsf{chan}\colon E \to \mathcal{CH}$ returns the channel of each event; and
$\mathsf{msg}\colon E \to \mathcal{D}$ returns the message passed in each event.

$\mathcal{B} = (B, \preceq)$ is a *system of events*, written $\mathcal{B} \in \mathsf{ES}(\mathcal{E})$, iff (i) $B \subseteq E$; (ii) $\preceq$ is a partial ordering on B; and (iii) for every $e_1 \in B$, $\{e_0 \in B\colon e_0 \preceq e_1\}$ is finite.

Hence, $\mathcal{B}$ is well-founded. If $\mathcal{B} = (B, \preceq)$, we refer to B as $\mathsf{ev}(\mathcal{B})$ and to $\preceq$ as $\preceq_{\mathcal{B}}$.

Now let $\mathcal{B} = (B, \preceq) \in \mathsf{ES}(\mathcal{E})$, and define $\mathsf{proj}(B, \ell) =$

$$\{e \in B\colon \mathsf{sender}(\mathsf{chan}(e)) = \ell \text{ or } \mathsf{rcpt}(\mathsf{chan}(e)) = \ell\}.$$

$\mathcal{B}$ is an *execution*, written $\mathcal{B} \in \mathsf{Exc}(\mathcal{F})$ iff, for every $\ell \in \mathcal{LO}$,

1. $\mathsf{proj}(B, \ell)$ is linearly ordered by $\preceq$, hence—by the finiteness condition (iii)—a sequence, and
2. $\mathsf{proj}(B, \ell) \in \mathsf{traces}(\ell)$. ///

We often write $\mathcal{A}, \mathcal{A}'$, etc., when $\mathcal{A}, \mathcal{A}' \in \mathsf{Exc}(\mathcal{F})$. The choice between two structures $\mathcal{E}_1, \mathcal{E}_2$ makes little difference: If $\mathcal{E}_1, \mathcal{E}_2$ have the same cardinality, then to within isomorphism they lead to the same systems of events and hence also executions. Thus, we suppress the parameter $\mathcal{E}$, henceforth.

This semantics associates a set of executions with each frame, without imposing any notion of inputs and outputs, or regarding a frame as a program-like function.

Definition 6. Let $\mathcal{B}_1 = (B_1, \preceq_1), \mathcal{B}_2 = (B_2, \preceq_2) \in \mathsf{ES}(\mathcal{F})$.

1. $\mathcal{B}_1$ is a *substructure* of $\mathcal{B}_2$ iff $B_1 \subseteq B_2$ and $\preceq_1 = (\preceq_2 \cap B_1 \times B_1)$.
2. $\mathcal{B}_1$ is an *initial substructure* of $\mathcal{B}_2$ iff $\mathcal{B}_1$ is a substructure of $\mathcal{B}_2$, and for all $y \in B_1$, if $x \preceq_2 y$, then $x \in B_1$. ///

Lemma 7. 1. If $\mathcal{B}_1$ is a substructure of $\mathcal{B}_2 \in \mathsf{ES}(\mathcal{F})$, then $\mathcal{B}_1 \in \mathsf{ES}(\mathcal{F})$.
2. If $\mathcal{B}_1$ is an initial substructure of $\mathcal{B}_2 \in \mathsf{Exc}(\mathcal{F})$, then $\mathcal{B}_1 \in \mathsf{Exc}(\mathcal{F})$.
3. Being an execution is preserved under chains of initial substructures: Suppose that $\langle \mathcal{B}_i\rangle_{i\in\mathbb{N}}$ is a sequence where each $\mathcal{B}_i \in \mathsf{Exc}(\mathcal{F})$, such that $i \leq j$ implies $\mathcal{B}_i$ is an initial substructure of $\mathcal{B}_j$. Then $(\bigcup_{i\in\mathbb{N}} \mathcal{B}_i) \in \mathsf{Exc}(\mathcal{F})$. ///

Example 8 (Network with filtering)**.** To localize our descriptions of functionality, we expand the network of Fig 1; see Fig. 4. Regions are displayed as $\bullet$; routers, as $\square$; and interfaces, as $\triangle$. When a router has an interface onto a segment, a pair of locations—representing that interface as used in each direction—lie between this router and each peer router [21].

Let $\mathsf{Dir} = \{\mathsf{inb}, \mathsf{outb}\}$ represent the inbound direction and the outbound directions from routers, respectively. Suppose Rt is a set of routers r, each with a set of interfaces $\mathsf{intf}(r)$, and a set of network regions Rg containing end hosts.

Each member of Rt, Rg is a *location*. Each interface-direction pair $(i, r) \in (\bigcup_{r\in\mathsf{Rt}} \mathsf{intf}(r)) \times \mathsf{Dir}$ is also a location. The *channels* are those shown. Each interface has a pair of channels that allow datagrams to pass between the router and the interface, and between the interface and an adjacent entity. We also include a self-loop channel at each network region i, n_1, n_2; it represents transmissions and receptions among the hosts and network infrastructure coalesced into the region. Thus:

$\mathcal{LO} =$ $\mathsf{Rt} \cup \mathsf{Rg} \cup ((\bigcup_{r\in\mathsf{Rt}} \mathsf{intf}(r)) \times \mathsf{Dir})$;
$\mathcal{CH} =$ $\{(\ell_1, \ell_2) \in \mathcal{LO} \times \mathcal{LO}\colon \ell_1$ delivers datagrams directly to $\ell_2\}$;
$\mathcal{D} =$ the set of IP datagrams;
$\mathsf{ends}(\ell) = \{\mathsf{entry}(\ell, \ell_2)\colon (\ell, \ell_2) \in \mathcal{CH}\} \cup \{\mathsf{exit}(\ell_1, \ell)\colon (\ell_1, \ell) \in \mathcal{CH}\}$, for each $\ell \in \mathcal{LO}$.

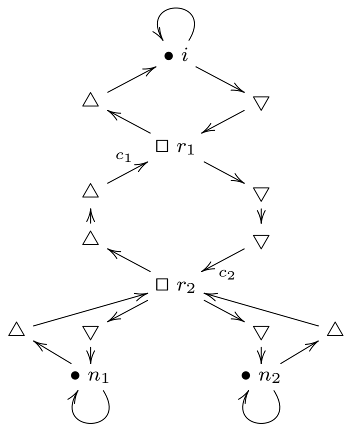

Fig. 4. Expanded representation of network from Fig. 1

The traces are easily specified. Each router $r \in \mathsf{Rt}$ receives packets from inbound interfaces, and chooses an outbound interface for each. Its state is a set of received but not yet routed datagrams, and the sole initial state is $\emptyset$. The transition relation, when receiving a datagram, adds it to this set. When transmitting a datagram d in the current set, it removes d from the next state and selects an outbound channel as determined by the routing table. For simplicity, the routing table is an unchanging part of determining the transition relation.

A directed interface enforces filtering rules. The state again consists of the set of received but not-yet-processed datagrams. The transition relation uses an unchanging filter function to determine, for each datagram, whether to discard it or retransmit it.

If $n \in \mathsf{Rg}$ is a region, its state is the set of datagrams it has received and not yet retransmitted. It can receive a datagram; transmit one from its state; or else initiate a new datagram. If it is assumed to be well-configured, these all have source address in a given range of IP addresses. Otherwise, the source addresses may be arbitrary. ///

If the router is executing other sorts of processing, for instance Network Address Translation or the IP Security Protocols, then the behavior is slightly more complex [1], [21], but sharply localized. Many other problems can be viewed as frames. Beyond voting schemes (Ex. 2), attestation architectures [13] and other secure virtualized systems are, at one level, sets of virtual machines communicating through one-directional channels.

Partially vs. totally ordered executions. Def. 5 does not require the ordering $\preceq$ of "occurring before" to be total. When events occur on different channels, neither has to precede the other. Thus, our executions need not be sequential.

This has three advantages. First, it is more inclusive, since executions with total orders satisfy our definition as do those with (properly) partial orders. Indeed, the main claims of this paper remain true when restricted to executions that are totally ordered. Second, reasoning is simplified. We do not need to interleave events when combining two local executions to construct a global one, as encapsulated in the proofs of Lemmas 15, 41. Nor do we need to "compact" events, when splitting off a local execution, as we would if we used a particular index set for sequences. This was probably an advantage to us in developing these results. Third, the minimal partial order is a reflection of causality, which can be used also to reason about independence. We expect this to be useful in future work.

There is also a disadvantage: unfamiliarity. It requires some caution. Moreover, mechanized theorem provers have much better support for induction over sequences than over well-founded orders. This inconvenienced a colleague who used PVS [40] to formalize parts of this work, and eventually chose to use totally ordered executions for induction-oriented proofs. With that difference, Thms. 28 and 32 have been confirmed in PVS, as have the basic properties of Example 34.

V. Non-disclosure

Following Sutherland [53], we think of information flow in terms of *deducibility* or *disclosure*. A participant observes part of the system behavior, trying to draw conclusions about a different part. If his observations exclude some possible behaviors of that part, then he can *deduce* that those behaviors did not occur. His observations have *disclosed* something.

These observations occur on a set of channels $C_o \subseteq \mathcal{CH}$, and the deductions constrain the events on a set of channels $C_s \subseteq \mathcal{CH}$. C_o is the set of *observed* channels, and C_s is the set of *source* channels. The observer has access to the events on the channels in C_o in an execution, using these events to learn about what happened at the source. The observed events may rule out some behaviors on the channels C_s.

Definition 9. Let $C \subseteq \mathcal{CH}$, and $\mathcal{B} \in \mathsf{ES}(\mathcal{F})$.

1. The *restriction* $\mathcal{B} \restriction C$ *of* $\mathcal{B}$ to C is (B_0, R), where $B_0 = \{e \in \mathcal{B} \colon \mathsf{chan}(e) \in C\}$, and $R = (\preceq \cap\, B_0 \times B_0)$.
2. $\mathcal{B} \in \mathsf{ES}(\mathcal{F})$ is a *C-run* iff for some $\mathcal{A} \in \mathsf{Exc}(\mathcal{F})$, $\mathcal{B} = \mathcal{A} \restriction C$. We write C-$\mathsf{runs}(\mathcal{F})$, or sometimes C-runs, for the set of C-runs of $\mathcal{F}$. A *local run* is a member of C-runs for the relevant C.
3. $J_{C' \triangleleft C}(\mathcal{B})$ gives the C'-runs *compatible with* a C-run $\mathcal{B}$:

$$J_{C' \triangleleft C}(\mathcal{B}) \quad = \quad \{\mathcal{A} \restriction C' \colon \mathcal{A} \in \mathsf{Exc}(\mathcal{F}) \text{ and } \mathcal{A} \restriction C = \mathcal{B}\}. \qquad ///$$

$\mathcal{B} \restriction C \in \mathsf{ES}(\mathcal{F})$ by Lemma 7. In $J_{C' \triangleleft C}(\mathcal{B})$, the lower right index C indicates what type of local run $\mathcal{B}$ is. The lower left index C' indicates the type of local runs in the resulting set. J stands for "joint." $J_{C' \triangleleft C}(\mathcal{B})$ makes sense even if C and C' overlap, though behavior on $C \cap C'$ is not hidden from observations at C.

Lemma 10.
1. C-$\mathsf{runs} = J_{C \triangleleft \emptyset}(\emptyset, \emptyset)$, i.e. the local runs at C are all those compatible with the empty event set $(\emptyset, \emptyset)$ at the empty set of channels.
2. $\mathcal{B} \notin C$-runs implies $J_{C' \triangleleft C}(\mathcal{B}) = \emptyset$.
3. $\mathcal{B} \in C$-runs implies $J_{C \triangleleft C}(\mathcal{B}) = \{\mathcal{B}\}$.
4. $J_{C' \triangleleft C}(\mathcal{B}) \subseteq C'$-$\mathsf{runs}$. ///

$\mathcal{A}$ *witnesses for* $\mathcal{B}' \in J_{C' \triangleleft C}(\mathcal{B})$ iff $\mathcal{A} \in \mathsf{Exc}(\mathcal{F})$, $\mathcal{B} = \mathcal{A} \restriction C$, and $\mathcal{B}' = \mathcal{A} \restriction C'$.

No disclosure means that any observation $\mathcal{B}$ at C is compatible with everything that could have occurred at C', where *compatible* means that there is some execution that combines the local C-run with the desired C'-run.

We summarize "no disclosure" by the Leibnizian slogan: *Everything possible is compossible*, "compossible" being his coinage meaning possible together. If $\mathcal{B}, \mathcal{B}'$ are each separately possible—being C, C'-runs respectively—then there's an execution $\mathcal{A}$ combining them, and restricting to each of them.

Definition 11. $\mathcal{F}$ has *no disclosure from C to C'* iff, for all C-runs $\mathcal{B}$, $J_{C' \triangleleft C}(\mathcal{B}) = C'$-runs. ///

A. Symmetry of disclosure

Like Shannon's mutual information and Sutherland's non-deducibility [53], "no disclosure" is symmetric:

Lemma 12. 1. $\mathcal{B}' \in J_{C' \triangleleft C}(\mathcal{B})$ iff $\mathcal{B} \in J_{C \triangleleft C'}(\mathcal{B}')$.
2. $\mathcal{F}$ has no disclosure from C to C' iff $\mathcal{F}$ has no disclosure from C' to C.

Proof. **1.** By the definition, $\mathcal{B}' \in J_{C' \triangleleft C}(\mathcal{B})$ iff there exists an execution $\mathcal{B}_1$ such that $\mathcal{B}_1 \restriction C = \mathcal{B}$ and $\mathcal{B}_1 \restriction C' = \mathcal{B}'$. Which is equivalent to $\mathcal{B} \in J_{C \triangleleft C'}(\mathcal{B}')$.

2. There is no disclosure from C' to C iff for every C-run $\mathcal{B}$ and C'-run $\mathcal{B}'$, $\mathcal{B}' \in J_{C' \triangleleft C}(\mathcal{B})$. By Clause 1, this is the same as $\mathcal{B} \in J_{C \triangleleft C'}(\mathcal{B}')$. □

Because of this symmetry, we speak of no disclosure *between* C and C'.

Lemma 13. 1. Suppose $C_0 \subseteq C_1$ and $C'_0 \subseteq C'_1$. If $\mathcal{F}$ has no disclosure from C_1 to C'_1, then $\mathcal{F}$ has no disclosure from C_0 to C'_0.
2. When $C_1, C_2, C_3 \subseteq \mathcal{CH}$,

$$J_{C_3 \triangleleft C_1}(\mathcal{B}_1) \subseteq \bigcup_{\mathcal{B}_2 \in J_{C_2 \triangleleft C_1}(\mathcal{B}_1)} J_{C_3 \triangleleft C_2}(\mathcal{B}_2). \qquad ///$$

This is not always an equality. $\mathcal{B}_1 \in C_1$-runs and $\mathcal{B}_3 \in C_3$-runs may make incompatible demands on a location ℓ. The location ℓ may have endpoints on channels in both C_1 and C_3; or paths may connect ℓ to both C_1 and C_3 without traversing C_2. Lemma 15 shows that otherwise equality holds. See Appendix A for this, and longer subsequent, proofs.

B. The Cut Principle for Non-disclosure

Our key observation is that non-disclosure respects the graph structure of a frame $\mathcal{F}$. If $\mathsf{cut} \subseteq \mathcal{CH}$ is a cut set in the undirected graph $\mathsf{ungr}(\mathcal{F})$, then disclosure from a source set $\mathsf{src} \subseteq \mathcal{CH}$ to a sink $\mathsf{obs} \subseteq \mathcal{CH}$ is controlled by disclosure to cut. If there is no disclosure from src to cut, there can be no disclosure from src to obs. As we will see in Section VII, this property extends to limited disclosure in the sense of disclosure to within a blur operator.

We view a cut as separating one set of channels as source from another set of channels as sink. Although it is more usual to take a cut to separate sets of nodes than sets of channels, it is easy to transfer between the channels and the relevant nodes. If $C \subseteq \mathcal{CH}$, we let $\mathsf{ends}(C) = \{\ell : \exists c \in C \,.\, \mathsf{sender}(c) = \ell$ or $\mathsf{rcpt}(c) = \ell\}$; conversely, $\mathsf{chans}(L) = \{c : \mathsf{sender}(c) \in L$ or $\mathsf{rcpt}(c) \in L\}$. For a singleton set $\{\ell\}$ we suppress the curly braces and write $\mathsf{chans}(\ell)$.

Definition 14. Let $\mathsf{src}, \mathsf{cut}, \mathsf{obs} \subseteq \mathcal{CH}$ be sets of channels; cut is an *undirected cut* (or simply a *cut*) between src, obs iff

1. src, cut, obs are pairwise disjoint; and
2. every undirected path p_1 in $\mathsf{ungr}(\mathcal{F})$ from any $\ell_1 \in \mathsf{ends}(\mathsf{obs})$ to any $\ell_2 \in \mathsf{ends}(\mathsf{src})$ traverses some member of cut. ///

For instance, in Fig. 4, $\{c_1, c_2\}$ is a cut between $\mathsf{chans}(i)$ and $\mathsf{chans}(\{n_1, n_2\})$. Lemma 15 serves as the heart of the proofs of the two main theorems about cuts, Thms. 16 and 28.

Lemma 15. Let cut be an undirected cut between src, obs, and let $\mathcal{B}_o \in$ obs-runs. Then

$$J_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o) = \bigcup_{\mathcal{B}_c \in J_{\mathsf{cut} \triangleleft \mathsf{obs}}(\mathcal{B}_o)} J_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c). \qquad ///$$

Proof. (Key idea; cf. App. A.) First, partition $\mathcal{LO}$ into three classes. Let left contain ℓ if ℓ has an endpoint on obs, or if ℓ can be reached by a path not traversing cut. Let right contain ℓ if ℓ has an endpoint on src, or if ℓ can be reached by a path not traversing cut. Let mid be the remainder, i.e. locations separated from both left and right by a channel in cut.

Suppose that $\mathcal{A}_1$ witnesses for $\mathcal{B}_c \in J_{\mathsf{cut} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$, and $\mathcal{A}_2$ witnesses for $\mathcal{B}_s \in J_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c)$. $\mathcal{A}_1$ and $\mathcal{A}_2$ agree for events involving mid, namely the events in $\mathcal{B}_c$ shared between them.

We build a witness $\mathcal{A}$ for $\mathcal{B}_s \in J_{\mathsf{cut} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$ by taking the events in $\mathcal{A}_1$ involving $\mathsf{left} \cup \mathsf{mid}$, union the the events in $\mathcal{A}_2$ involving $\mathsf{right} \cup \mathsf{mid}$. $\mathcal{A}$ is an execution because no location has a conflict between events from $\mathcal{A}_1$ and $\mathcal{A}_2$. □

The partial order semantics means that no arbitrary interleaving is needed to create the instance $\mathcal{A}$. Lemma 15 is in fact a corollary of Lemma 31, which makes an analogous assertion about a pair of overlapping frames.

Theorem 16. Let cut be an undirected cut between src, obs in $\mathcal{F}$. If there is no disclosure between src and cut, then there is no disclosure between src and obs.

Proof. Suppose that $\mathcal{B}_s \in$ src-runs and $\mathcal{B}_o \in$ obs-runs. We must show $\mathcal{B}_s \in J_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$. To apply Lemma 15, let $\mathcal{A} \in \mathsf{Exc}(\mathcal{F})$ such that $\mathcal{B}_o = \mathcal{A} \restriction \mathsf{obs}$; $\mathcal{A}$ exists by the definition of obs-run. Letting $\mathcal{B}_c = \mathcal{A} \restriction \mathsf{cut}$, we have $\mathcal{B}_c \in J_{\mathsf{cut} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$.

Since there is no disclosure between cut and src, $\mathcal{B}_s \in J_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c)$, and Lemma 15 applies. □

Example 17. In Fig. 4 let r_1 be configured to discard all inbound packets, and r_2 to discard all outbound packets. Then the empty event system is the only member of $\{c_1, c_2\}$-runs. Hence there is no disclosure between $\mathsf{chans}(i)$ and $\{c_1, c_2\}$. By Thm. 16, there is no disclosure to $\mathsf{chans}(\{n_1, n_2\})$. ///

Disconnected portions of a frame cannot interfere:

Corollary 18. If there is no path between src and obs in $\mathsf{ungr}(\mathcal{F})$, then there is no disclosure between them.

Proof. Then $\mathsf{cut} = \emptyset$ is an undirected cut set, and there is only one cut-run, namely the empty system of events. It is thus compatible with all src-runs. □

Thm. 16 and its analogue Thm. 28, while reminiscent of the max flow/min cut principle (cf. e.g. [14, Sec. 26.2]), are however quite distinct from it, as the latter depends essentially on the quantitative structure of network flows. Our results may also seem reminiscent of the Data Processing Inequality, stating that when three random variables X, Y, Z form a Markov chain, the mutual information $I(X; Z) \leq I(X; Y)$. Indeed, Thm. 16 entails the special case where $I(X; Y) = 0$, choosing $\mathsf{gr}(\mathcal{F})$ to be a single path $X \to Y \to Z$. For more on quantitative information flow, see the conclusion (Sec. IX).

VI. Blur operators

We will now adapt our theory to apply to partial disclosure as well as no disclosure. An observer learns something about a source of information when his observations are compatible with a proper subset of the behaviors possible for the source. Thus, the natural way to measure what has been learnt is the decrease in the set of possible behaviors at the source (see among many sources of this idea e.g. [17], [3]).

This starting point suggests focusing, for every frame and regions of interest $\mathsf{src} \subseteq \mathcal{CH}$ and $\mathsf{obs} \subseteq \mathcal{CH}$, on the compatibility equivalence relations on src-runs:

Definition 19. Let $\mathsf{src}, \mathsf{obs} \subseteq \mathcal{CH}$. If $\mathcal{B}_1, \mathcal{B}_2$ are src-runs, we say that they are obs-equivalent, and write $\mathcal{B}_1 \approx_{\mathsf{obs}} \mathcal{B}_2$, iff, for all obs-runs $\mathcal{B}_o$, $\mathcal{B}_1 \in J_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$ iff $\mathcal{B}_2 \in J_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$. ///

Lemma 20. For each obs and $\mathcal{B}_o \in$ obs-runs:

1. $\approx_{\mathsf{obs}}$ is an equivalence relation;
2. $J_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$ is a union of $\approx_{\mathsf{obs}}$-equivalence classes: let $\{\mathcal{S}_i\}_{i \in I}$ be the family of all $\approx_{\mathsf{obs}}$-equivalence classes. For some $I_0 \subseteq I$, $J_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o) = \bigcup_{i \in I_0} \mathcal{S}_i$. ///

No disclosure means that all src-runs are obs-equivalent, i.e. I_0 always equals I. Any notion of partial disclosure must respect obs-equivalence, since no observations can possibly "split" apart obs-equivalent src-runs. Partial disclosures always respect unions of obs-equivalence classes.

Rather than working directly with these unions of obs-equivalence classes, we instead focus on functions on sets of runs that satisfy three properties. These properties express the structural principles on partial disclosure that make our cut-blur and compositional principles hold. We call operators satisfying the properties *blur operators*. Lemma 22 shows that an operator that always returns unions of obs-equivalence classes is necessarily a blur operator.

When we want to prove results about all notions of partial disclosure, we prove them for all blur operators. When we want to show a particular relation is a possible notion of partial disclosure, we show that it generates an equivalence relation; Lemma 22 then justifies us in applying Thms. 28, 32.

Definition 21. A function f on sets is a *blur operator* iff it satisfies:

Inclusion: For all sets S, $S \subseteq f(S)$;
Idempotence: f is idempotent, i.e. for all sets S, $f(f(S)) = f(S)$; and
Union: f commutes with unions: If $S_{a \in I}$ is a family indexed by the set I, then

$$f(\bigcup_{a \in I} S_a) = \bigcup_{a \in I} f(S_a). \tag{1}$$

S is *f-blurred* iff f is a blur operator and $S = f(S)$. ///

By *Idempotence*, S is f-blurred iff it is in the range of the blur operator f. Since $S = \bigcup_{a \in S} \{a\}$, the *Union* property says that f is determined by its action on the singleton subsets of S. Thus, *Inclusion* could have said $a \in f(\{a\})$.

Monotonicity also follows from the *Union* property; if $S_1 \subseteq S_2$, then $S_2 = S_0 \cup S_1$, where $S_0 = S_2 \setminus S_1$. Thus $f(S_2) = f(S_0) \cup f(S_1)$, so $f(S_1) \subseteq f(S_2)$.

Lemma 22. Suppose that A is a set, and $\mathcal{R}$ is a partition of the elements of A. There is a unique function $f_{\mathcal{R}}$ on sets $S \subseteq A$ such that

1. $f_{\mathcal{R}}(\{a\}) = S$ iff $a \in S$ and $S \in \mathcal{R}$;
2. $f_{\mathcal{R}}$ commutes with unions (Eqn. 1).

Moreover, $f_{\mathcal{R}}$ is a blur operator.

Proof. Since $S = \bigcup_{a \in S} \{a\}$, $f_{\mathcal{R}}(S)$ is uniquely defined by the union principle (Eqn. 1).

Inclusion and *Union* are immediate from the form of the definition. *Idempotence* holds because being in the same $\mathcal{R}$-equivalence class is transitive. □

Although every equivalence relation determines a blur operator, the converse is not true: Not every blur operator is of this form. For instance, let $A = \{a, b\}$, and let $f(\{a\}) = \{a\}$, $f(\{b\}) = f(\{a, b\}) = \{a, b\}$. However, by Lemma 20 (cf. [24, Prop. 8]), *useful* partial disclosure is of this form:

Lemma 23. If $S = J_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$ is f-blurred, and $\mathcal{B}_s \in S$, then $f(\{\mathcal{B}_s\})$ is a union of $\approx_{\mathsf{obs}}$-equivalence classes. ///

The importance of Def. 21 is to identify the proof principles that make Thm. 28 true. The intuition comes from blurring an image: The viewer no longer knows the details of the scene, but only that it was some scene which, when blurred, would look like this, as the following example indicates.

Example 24. Imaginary Weather Forecasting Inc. (IWF) sells tailored, high-resolution weather data and forecasts to airlines, airports, etc., and low-resolution weather data more cheaply to TV and radio stations. IWF's low-tier subscribers should not learn higher resolution data than they have paid for. There is some disclosure about high resolution data because (e.g.) when low-tier subscribers see warm temperatures, they know that the high-resolution data is inconsistent with snow. We can formalize this partial disclosure as a blur.

Suppose IWF creates its low-resolution data d_L by applying a lossy compression function comp to high-resolution data d_H. When low-tier subscribers receive d_L, they know that the high-resolution data IWF measured from the environment is some element of $\mathsf{comp}^{-1}(d_L) = \{d_H : \mathsf{comp}(d_H) = d_L\}$. These sets are f-blurred where $f(\{d_H\}) = \{d'_H : \mathsf{comp}(d'_H) = \mathsf{comp}(d_H)\}$. ///

Curiously, IWF wants the low-tier customer, who receives one set of outputs, not to be able to infer too much about the outputs delivered to the high-tier customers. The inputs to the system—sensor values for temperature, wind, pressure etc. at different locations—are not of high value [22].

We will study information disclosure to within blur operators f, which we interpret as meaning $J_{C' \triangleleft C}(\mathcal{B}_c)$ is f-blurred. This is an "upper bound" on how much information about the local run at C' may be disclosed when $\mathcal{B}_c$ is observed. The observer will know an f-blurred set $S \in \mathcal{P}(C'\text{-runs})$ to which the behavior at C' belongs, without being able to infer anything finer than this f-blurred set.

Definition 25. Let $\mathsf{obs}, \mathsf{src} \subseteq \mathcal{CH}$ and $f \colon \mathcal{P}(\mathsf{src}\text{-runs}) \to \mathcal{P}(\mathsf{src}\text{-runs})$.

$\mathcal{F}$ *restricts disclosure from* src *to* obs *to within* f iff f is a blur operator and $J_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$ is f-blurred, for every $\mathcal{B}_o \in \mathsf{obs}\text{-runs}$.

We also say that $\mathcal{F}$ f*-limits* src*-to-*obs *flow*. ///

At one extreme, no-disclosure is disclosure to within a blur operator, namely the one that ignores S and adds all C'-runs:

$$f_{\mathsf{all}}(S) = \{\mathcal{A} \restriction C' : \mathcal{A} \in \mathsf{Exc}(\mathcal{F})\}.$$

At the other extreme, the maximally permissive security policy is disclosure to within the identity $f_{\mathsf{id}}(S) = S$. The blur f_{id} shows that every frame restricts disclosure to within *some* blur operator. Every set is a union of f_{id}-blurred sets.

$\mathcal{F}$ may f-limit src-to-obs flow even when the intersection $\mathsf{obs} \cap \mathsf{src}$ is non-empty, as long as f is not too fine-grained; see below (Def. 38).

Example 26. Suppose that $\mathcal{F}$ is an electronic voting system such as ThreeBallot [42]. Some locations L_{EC} are run by the election commission. We will regard the voters themselves as a set of locations L_V. Each voter delivers a message containing, in some form, his vote for some candidate.

The election officials observe the channels connected to L_{EC}, i.e. $\mathsf{chans}(L_{EC})$. To determine the correct outcome, they must infer a property of the local run at $\mathsf{chans}(L_V)$, namely, how many votes for each candidate occurred. However, they should not find out which voter voted for which candidate [15].

We formalize this via a blur operator. Suppose $\mathcal{B}' \in \mathsf{chans}(L_V)\text{-runs}$ is a possible behavior of all voters in L_V. Suppose that π is a permutation of L_V. Let $\pi \cdot \mathcal{B}'$ be the behavior in which each voter $\ell \in L_V$ casts not his own actual vote, but the vote actually cast by $\pi(\ell)$. That is, π represents one way of reallocating the actual votes among different voters. Now for any $S \subseteq \mathsf{chans}(L_V)\text{-runs}$ let

$$f_0(S) = \{\pi \cdot \mathcal{B}' \colon \mathcal{B}' \in S \wedge \pi \text{ is a permutation of } L_V\}. \quad (2)$$

This is a blur operator: (i) the identity is a permutation; (ii) permutations are closed under composition; and (iii) Eqn. 2 implies commutation with unions. The election commission should learn nothing about the votes of individuals, meaning that, for any $\mathcal{B} \in \mathsf{chans}(L_{EC})\text{-runs}$ the commission could observe, $J_{\mathsf{chans}(L_V) \triangleleft \mathsf{chans}(L_{EC})}(\mathcal{B})$ is f_0-blurred. Permutations of compatible voting patterns are also compatible.

This example is easily adapted to other considerations. For instance, the commissioners of elections are also voters, and they know how they voted themselves. Thus, we could define a (narrower) blur operator f_1 that only uses the permutations that leave commissioners' votes fixed.

In fact, voters are often divided among different precincts, and tallies are reported on a per-precinct basis. Thus, we have sets $V_1, \ldots, V_k$ of voters registered at the precincts $P_1, \ldots, P_k$ respectively. The relevant blur function says that we can permute the votes of any two voters $v_1, v_2 \in V_i$ within the same precinct. One cannot permute votes between different precincts, since that could change the tallies in the individual precincts. ///

Example 27. Suppose in Fig. 4: The inbound interface from i to router r_1 discards downward-flowing packets unless their source is an address in i and the destination is an address in n_1, n_2. The inbound interface for downward-flowing to router r_2 discards packets unless the destination address is the IP for a web server `www` in n_1, and the destination port is 80 or 443, or else their source port is 80 or 443 and their destination port is ≥ 1024.

We filter outbound (upward-flowing) packets symmetrically.

A packet is *importable* iff its source address is in i and either its destination is `www` and its destination port is 80 or 443; or else its destination address is in n_1, n_2, its source port is 80 or 443, and its destination port is ≥ 1024.

It is *exportable* iff, symmetrically, its destination address is in i and either its source is `www` and its source port is 80 or 443; or else its source address is in n_1, n_2, its destination port is 80 or 443, and its source port is ≥ 1024.

We will write $\mathsf{select}\, \mathcal{B}\, p$ for the result of selecting those events $e \in \mathsf{ev}(\mathcal{B})$ that satisfy the predicate $p(e)$, restricting $\preceq$ to the selected events. Now consider the operator f_i on $\mathsf{chans}(i)$-runs generated as in Lemma 22 from the equivalence relation:

$\mathcal{B}_1 \approx_i \mathcal{B}_2$ iff they agree on all *importable* events, i.e.:

$$\mathsf{select}\, \mathcal{B}_1\, (\lambda e \,.\, \mathsf{msg}(e) \text{ is importable}) \cong \mathsf{select}\, \mathcal{B}_2\, (\lambda e \,.\, \mathsf{msg}(e) \text{ is importable}).$$

The router configurations mentioned above are intended to ensure that there is f_i-limited flow from $\mathsf{chans}(i)$ to $\mathsf{chans}(\{n_1, n_2\})$. This is an *integrity* condition; it is meant to ensure that systems in n_1, n_2 cannot be affected by bad (i.e. non-importable) packets from i.

Outbound, the blur f_e on $\mathsf{chans}(\{n_1, n_2\})$-runs is generated from the equivalence relation:

$\mathcal{B}_1 \approx_e \mathcal{B}_2$ iff they agree on all *exportable* events, i.e.:

$$\mathsf{select}\, \mathcal{B}_1\, (\lambda e\, .\, \mathsf{msg}(e) \text{ is exportable}) \cong \\ \mathsf{select}\, \mathcal{B}_2\, (\lambda e\, .\, \mathsf{msg}(e) \text{ is exportable}).$$

The router configurations are also intended to ensure that there is f_e-limited flow from $\mathsf{chans}(\{n_1, n_2\})$ to $\mathsf{chans}(i)$.

This is a *confidentiality* condition; it is meant to ensure that external observers learn nothing about the non-exportable traffic, which was not intended to exit the organization.

In this example, transmission of an exportable packet is never dependent on reception of a non-exportable packet, and similarly for importable packets. In applications lacking this simplifying property, proving flow limitations is harder. ///

VII. The Cut-Blur Principle

The symmetry of non-disclosure (Lemma 12) no longer holds for disclosure to within a blur. We have, however, the natural extension of Thm 16:

Theorem 28 (Cut-Blur Principle)**.** Let cut be an undirected cut between $\mathsf{src}, \mathsf{obs}$ in $\mathcal{F}$. If $\mathcal{F}$ f-limits src-to-cut flow, then $\mathcal{F}$ f-limits src-to-obs flow.

Proof. By the hypothesis, f is a blur operator. Let $\mathcal{B}_o$ be a obs-run. We want to show that $J_{\mathsf{src} \lhd \mathsf{obs}}(\mathcal{B}_o)$ is an f-blurred set, i.e. $J_{\mathsf{src} \lhd \mathsf{obs}}(\mathcal{B}_o) = f(J_{\mathsf{src} \lhd \mathsf{obs}}(\mathcal{B}_o))$.

For convenience, let $S_c = J_{\mathsf{cut} \lhd \mathsf{obs}}(\mathcal{B}_o)$.

By Lemma 15, $J_{\mathsf{src} \lhd \mathsf{obs}}(\mathcal{B}_o) = \bigcup_{\mathcal{B}_c \in S_c} J_{\mathsf{src} \lhd \mathsf{cut}}(\mathcal{B}_c)$. Thus, we must show that the latter is f-blurred.

By the assumption that each $J_{\mathsf{src} \lhd \mathsf{cut}}(\mathcal{B}_c)$ is f-blurred and by idempotence, $J_{\mathsf{src} \lhd \mathsf{cut}}(\mathcal{B}_c) = f(J_{\mathsf{src} \lhd \mathsf{cut}}(\mathcal{B}_c))$. Now:

$$\begin{aligned} \bigcup_{\mathcal{B}_c \in S_c} J_{\mathsf{src} \lhd \mathsf{cut}}(\mathcal{B}_c) &= \bigcup_{\mathcal{B}_c \in S_c} f(J_{\mathsf{src} \lhd \mathsf{cut}}(\mathcal{B}_c)) \\ &= f(\bigcup_{\mathcal{B}_c \in S_c} J_{\mathsf{src} \lhd \mathsf{cut}}(\mathcal{B}_c)), \end{aligned}$$

applying the union property (Eqn. 1). Hence, $\bigcup_{\mathcal{B}_c \in S_c} J_{\mathsf{src} \lhd \mathsf{cut}}(\mathcal{B}_c)$ is f-blurred. ☐

This proof is the reason we introduced the *Union* principle Eqn. 1, rather than simply considering all closure operators [37]. Eqn. 1 distinguishes the closure operators that allow the "long distance reasoning" summarized in the proof.

Example 29. The frame of Example 27 has f_i-limited flow from $\mathsf{chans}(i)$ to the cut $\{c_1, c_2\}$. Thus, it has f_i-limited flow from $\mathsf{chans}(i)$ to $\mathsf{chans}(\{n_1, n_2\})$.

It also has f_e-limited flow from $\mathsf{chans}(\{n_1, n_2\})$ to the cut $\{c_1, c_2\}$. This implies f_e-limited flow to $\mathsf{chans}(i)$. ///

A. A Compositional Relation between Frames

Our next technical result gives us a way to "transport" a blur security property from one frame $\mathcal{F}_1$ to another frame $\mathcal{F}_2$. It assumes that the two frames share a common core, some set of locations L_0. These locations should hold the same channel endpoints in each of $\mathcal{F}_1, \mathcal{F}_2$, and should engage in the same traces. The boundary separating L_0 from the remainder of $\mathcal{F}_1, \mathcal{F}_2$ necessarily forms a cut set cut. Assuming that the local runs at cut are respected, blur properties are preserved from $\mathcal{F}_1$ to $\mathcal{F}_2$.

Definition 30. A set L_0 of locations is *shared between* $\mathcal{F}_1$ and $\mathcal{F}_2$ iff $\mathcal{F}_1, \mathcal{F}_2$ are frames with locations $\mathcal{LO}_1, \mathcal{LO}_2$, endpoints $\mathsf{ends}_1, \mathsf{ends}_2$ and traces $\mathsf{traces}_1, \mathsf{traces}_2$, resp., where $L_0 \subseteq \mathcal{LO}_1 \cap \mathcal{LO}_2$, and for all $\ell \in L_0$, $\mathsf{ends}_1(\ell) = \mathsf{ends}_2(\ell)$ and $\mathsf{traces}_1(\ell) = \mathsf{traces}_2(\ell)$.

When L_0 is shared between $\mathcal{F}_1$ and $\mathcal{F}_2$, let:

$\mathsf{left}_0 = \{c \in \mathcal{CH}_1 \colon$ both endpoints of c are locations $\ell \in L_0\}$;
$\mathsf{cut}_0 = \{c \in \mathcal{CH}_1 \colon$ exactly one endpoint of c is a location $\ell \in L_0\}$; and
$\mathsf{right}_i = \{c \in \mathcal{CH}_i \colon$ neither endpoint of c is a location $\ell \in L_0\}$, for $i = 1, 2$.

We will also use C-runs_1 and C-runs_2 to refer to the local runs of C within $\mathcal{F}_1$ and $\mathcal{F}_2$, resp.; and $J^1_{C' \lhd C}(\mathcal{B})$ and $J^2_{C' \lhd C}(\mathcal{B})$ will refer to the compatible C' runs in the frames $\mathcal{F}_1$ and $\mathcal{F}_2$, resp. ///

Indeed, cut_0 is an undirected cut between left_0 and right_i in $\mathcal{F}_i$, for $i = 1$ and 2. In an undirected path that starts in left_0 and never traverses cut_0, each arc always has both ends in L_0. We next prove a two-frame analog of Lemma 15.

Lemma 31. Let L_0 be shared between frames $\mathcal{F}_1, \mathcal{F}_2$. Let $\mathsf{src} \subseteq \mathsf{left}_0$, and $\mathcal{B}_c \in \mathsf{cut}_0\text{-}\mathsf{runs}_1 \cap \mathsf{cut}_0\text{-}\mathsf{runs}_2$.

1. $J^1_{\mathsf{src} \lhd \mathsf{cut}_0}(\mathcal{B}_c) = J^2_{\mathsf{src} \lhd \mathsf{cut}_0}(\mathcal{B}_c)$.
2. Assume $\mathsf{cut}_0\text{-}\mathsf{runs}(\mathcal{F}_2) \subseteq \mathsf{cut}_0\text{-}\mathsf{runs}(\mathcal{F}_1)$. Let $\mathsf{obs} \subseteq \mathsf{right}_2$, and $\mathcal{B}_o \in \mathsf{obs}\text{-}\mathsf{runs}_2$. Then
$$J^2_{\mathsf{src} \lhd \mathsf{obs}}(\mathcal{B}_o) = \bigcup_{\mathcal{B}_c \in J^2_{\mathsf{cut}_0 \lhd \mathsf{obs}}(\mathcal{B}_o)} J^1_{\mathsf{src} \lhd \mathsf{cut}_0}(\mathcal{B}_c).$$

Part 1 states that causality acts locally. The variable portions $\mathsf{right}_1, \mathsf{right}_2$ of $\mathcal{F}_1$ and $\mathcal{F}_2$ can affect what happens in their shared part left. But it does so only by changing which cut_0-runs are possible. Whenever both frames agree on any $\mathcal{B}_c \in \mathsf{cut}_0\text{-}\mathsf{runs}_1 \cap \mathsf{cut}_0\text{-}\mathsf{runs}_2$, then the left-runs runs compatible with $\mathcal{B}_c$ are the same. Distant effects from right_i to left occur only via local runs at the boundary cut_0.

The assumption $\mathsf{cut}_0\text{-}\mathsf{runs}(\mathcal{F}_2) \subseteq \mathsf{cut}_0\text{-}\mathsf{runs}(\mathcal{F}_1)$ in Part 2 and Thm. 32 is meant to limit this variability in one direction.

Theorem 32. Suppose that L_0 is shared between frames $\mathcal{F}_1, \mathcal{F}_2$, and assume $\mathsf{cut}_0\text{-}\mathsf{runs}(\mathcal{F}_2) \subseteq \mathsf{cut}_0\text{-}\mathsf{runs}(\mathcal{F}_1)$. Consider any $\mathsf{src} \subseteq \mathsf{left}_0$ and $\mathsf{obs} \subseteq \mathsf{right}_2$. If $\mathcal{F}_1$ f-limits src-to-cut_0 flow, then $\mathcal{F}_2$ f-limits src-to-obs flow.

The proof is similar to the proof of the cut-blur principle, which effectively results from it by replacing Lemma 31 by

Lemma 15, and omitting the subscripts on frames and their local runs. The cut-blur principle is in fact the corollary of Thm. 32 for $\mathcal{F}_1 = \mathcal{F}_2$.

B. Two Applications

Thm. 32 is useful as a compositional principle. It implies that in Example 29 non-exportable traffic in n_1, n_2 remains unobservable even as we vary the top part of Fig. 4:

Example 33. Regarding Fig. 4 as the frame $\mathcal{F}_1$, let L_0 be the locations below $\{c_1, c_2\}$, and let $\mathsf{cut} = \{c_1, c_2\}$. Let $\mathcal{F}_2$ contain L_0, cut as shown, and have any graph structure above cut such that cut remains a cut between the new structure and $\mathcal{F}_0$. Let the new locations have any transition systems such that the local runs agree, i.e. $\mathsf{cut}\text{-}\mathsf{runs}(\mathcal{F}_2) = \mathsf{cut}\text{-}\mathsf{runs}(\mathcal{F}_1)$. Then by Thm. 32, external inferences about $\mathsf{chans}(\{n_1, n_2\})$ are guaranteed to blur out non-exportable events. ///

It is appealing that our security goal is independent of changes in the structure of the internet that we do not control. A similar property holds for the integrity goal of Example 29 as we alter the internal network. The converse questions—preserving the confidentiality property as the internal network changes, and the integrity property as the internet changes—appear to require a different, refinement-oriented theorem.

Example 34. Consider a frame $\mathcal{F}_1$ representing a precinct, as shown in Fig. 2. It consists of a set of voters $\overline{v} = \{v_1, \ldots, v_k\}$, a ballot box BB_1, and a channel c_1 connecting that to the election commission EC. The EC publishes the results over the channel p to the public Pub.

We have proved that a particular implementation of BB_1 ensures that $\mathcal{F}_1$ blurs the votes; we formalized this within the theorem prover PVS. That is, if a pattern of voting in precinct 1 is compatible with an observation at c_1, then any permutation of the votes at $\overline{v}$ is also compatible.

The cut-blur principle implies this blur also applies to observations at channel p to the public. Other implementations of BB_1 also achieve this property. ThreeBallot and VAV [42] appear to have this effect; they involve some additional data delivered to Pub, namely the receipts for the ballots.[1]

However, elections generally concern many precincts. Frame $\mathcal{F}_2$ contains i precincts, all connected to the election commission EC (Fig. 3). Taking $L_0 = \overline{v} \cup \{BB_1\}$, we may apply Thm. 32. We now have $\mathsf{cut} = \{c_1\}$. Thus, to infer that $\mathcal{F}_2$ blurs observations of the voters in precinct 1, we need only check that $\{c_1\}$ has no new local runs in $\mathcal{F}_2$.

By symmetry, each precinct in $\mathcal{F}_2$ enjoys the same blur.

Thus—for a given local run at p—any permutation of the votes at $\overline{v}$ preserves compatibility in $\mathcal{F}_2$, and any permutation of the votes at $\overline{w}$ preserves compatibility in $\mathcal{F}_2$. However, Thm. 32 does not say that any pair of permutations at $\overline{v}$ and $\overline{w}$ must be jointly compatible. That is, does every permutation on $\overline{v} \cup \overline{w}$ that respects the division between the precinct of the $\overline{v}$s and the precinct of the $\overline{w}$s preserve compatibility? Although Thm. 32 does not answer this question, the answer is yes, as we can see by applying Lemma 41 to $\mathcal{F}_2$. ///

[1] Our claim is possibilistic. Quantitatively, this may no longer hold: Some permutations may be more likely than others, given the receipts [12], [38].

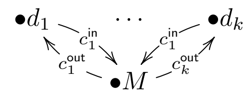

Fig. 5. Machine M, domains $\{d_1, \ldots, d_k\}$

Thm. 32 is a tool to justify abstractions. Fig. 4 is a sound abstraction of a variety of networks, and Fig. 2 is a sound abstraction of the various multiple precinct instances of Fig. 3.

VIII. Relating Blurs to Noninterference and Nondeducibility

If we specialize frames to state machines (see Fig. 5), we can reproduce some of the traditional definitions. Let $D = \{d_1, \ldots, d_k\}$ be a finite set of *domains*, i.e. sensitivity labels; $\hookrightarrow \subseteq D \times D$ specifies which domains are *visible* to others, and *may influence* them. We assume $\hookrightarrow$ is reflexive, though not necessarily transitive. A is a set of *actions*, and $\mathsf{dom}\colon A \to D$ assigns a domain to each action; O is a set of outputs.

$M = \langle S, s_0, A, \delta, \mathsf{obs}\rangle$ is a (possibly non-deterministic) state machine with states S, initial state s_0, transition relation $\delta \subseteq S \times A \times S$, and observation function $\mathsf{obs}\colon S \times D \to O$. M has a set of traces, and each trace α determines a sequence of observations for each domain [47], [54], [55].

M accepts commands from A along the incoming channels c_i^{in} from the d_i; each command $a \in A$ received from d_i has sensitivity $\mathsf{dom}(a) = d_i$. M delivers observations along the outgoing channels c_i^{out}. The frame requires a little extra memory, in addition to the states of M, to deliver outputs over the channels c_i^{out}.

$\mathcal{F}$ is star-like, since M holds an endpoint for each channel. Hence, if $\mathcal{A} \in \mathsf{Exc}(\mathcal{F})$, all events are in $\mathsf{proj}(\mathcal{A}, M)$, and $\preceq_{\mathcal{A}}$ is linearly ordered. Let us write:

$C_i = \{c_i^{\mathsf{in}}, c_i^{\mathsf{out}}\}$ for d_i's input and output for M;
$\mathsf{vis}(d_i) = \{c_j^{\mathsf{in}} \colon d_j \hookrightarrow d_i\}$ for inputs visible to d_i;
$\mathsf{IN} = \{c_x^{\mathsf{in}} \colon 1 \le x \le k\}$ for the input channels;
$\mathsf{input}(\mathcal{A}) = \mathcal{A} \restriction \mathsf{IN}$ for all input behavior in $\mathcal{A}$.

Noninterference and nondeducibility. *Noninterference* [20] and its variants are defined by *purge* functions p for each target domain d_i, defined by recursion on input behaviors $\mathsf{input}(\mathcal{A})$. The original Goguen-Meseguer (GM) purge function p_o for d_i [20] retains the events $e \in \mathsf{input}(\mathcal{A})$ satisfying the predicate

$$\mathsf{chan}(e) \in \mathsf{vis}(d_i).$$

A purge function for intransitive $\hookrightarrow$ relations was subsequently proposed by Haigh and Young [23]. In the purge function for domain d_i, any input event $e_0 \in \mathsf{input}(\mathcal{A})$ is retained if $\mathsf{input}(\mathcal{A})$ has an increasing subsequence $e_0 \preceq e_1 \preceq \ldots \preceq e_j$ where $\mathsf{dom}(\mathsf{chan}(e_j)) = d_i$ and, for each k with $0 \le k < j$,

$$\mathsf{chan}(e_k) \in \mathsf{vis}(\mathsf{dom}(e_{k+1})).$$

In [54], van der Meyden's purge functions yield tree structures instead of subsequences; every path from a leaf to the root in these trees is a subsequence consisting of permissible effects $\mathsf{chan}(e_k) \in \mathsf{vis}(\mathsf{dom}(e_{k+1}))$. This tightens the notion of security, because the trees "forget" ordering information between events that lie on different branches to the root.

We formalize a *purge function* for a domain $d_i \in D$ as being a function from executions $\mathcal{A}$ to some range set A. It should be sensitive only to *input* events in $\mathcal{A}$ (condition 1), and it should certainly reflect *all* the inputs *visible* to level d_i (condition 2). In most existing definitions, the range A consists of sequences of input events, though in van der Meyden's [54], they are trees of input events. In [11], the range depends on how declassification conditions are defined.

Definition 35. Let $\mathcal{F}$ be as in Fig. 5, and A any set. A function $p\colon \mathsf{Exc}(\mathcal{F}) \to A$ is a d_i-*purge function*, where $d_i \in D$, iff

1. $\mathsf{input}(\mathcal{A}) = \mathsf{input}(\mathcal{A}')$ implies $p(\mathcal{A}) = p(\mathcal{A}')$;
2. $p(\mathcal{A}) = p(\mathcal{A}')$ implies $\mathcal{A} \restriction \mathsf{vis}(d_i) = \mathcal{A}' \restriction \mathsf{vis}(d_i)$.

If p is a d_i-purge, $\mathcal{A} \approx^p \mathcal{A}'$ means $p(\mathcal{A}) = p(\mathcal{A}')$. ///

Each purge p determines notions of noninterference and nondeducibility.

Definition 36. Let p be a purge function for $d_i \in D$. $\mathcal{F}$ *is p-noninterfering*, written $\mathcal{F} \in \mathsf{NI}^p$, iff, for all $\mathcal{A}, \mathcal{A}' \in \mathsf{Exc}(\mathcal{F})$,

$$\mathcal{A} \approx^p \mathcal{A}' \text{ implies } \mathcal{A} \restriction C_i = \mathcal{A}' \restriction C_i.$$

$\mathcal{F}$ *is p-nondeducible* ($\mathcal{F} \in \mathsf{ND}^p$), iff, for all $\mathcal{A}, \mathcal{A}' \in \mathsf{Exc}(\mathcal{F})$,

$$\mathcal{A} \approx^p \mathcal{A}' \text{ implies } \mathcal{A}' \restriction \mathsf{IN} \in J_{\mathsf{IN} \lhd C_i}(\mathcal{A} \restriction C_i). \qquad ///$$

Here we take non-deducibility to mean that d_i's observations provide no more information about all inputs than the purge p preserves. Thus, $\mathcal{A} \restriction C_i$ is akin to Sutherland's *view* [53, Sec. 5.2], although slightly adapted.

Sutherland's *hidden_from* appears to mean $\mathcal{A}' \restriction \{c_j^{\mathsf{in}} \colon d_j \not\rightsquigarrow d_i\}$, i.e. the inputs that would not be visible to d_i. This agrees with our proposed definition in the case Sutherland considered, namely the classic GM purge for noninterference. The assumption $\mathcal{A} \approx^p \mathcal{A}'$ is meant to extend nondeducibility for other purges. As expected, noninterference is tighter than nondeducibility [53, Sec. 7]:

Lemma 37. Let p be a purge function for domain d_i. $\mathcal{F} \in \mathsf{NI}^p$ implies $\mathcal{F} \in \mathsf{ND}^p$.

Proof. Assume that $\mathcal{F} \in \mathsf{NI}^p$ and $\mathcal{A}, \mathcal{A}' \in \mathsf{Exc}(\mathcal{F})$, where $\mathcal{A} \approx^p \mathcal{A}'$. By the definition, $\mathcal{A} \restriction C_i = \mathcal{A}' \restriction C_i$. Thus, $J_{\mathsf{IN} \lhd C_i}(\mathcal{A} \restriction C_i) = J_{\mathsf{IN} \lhd C_i}(\mathcal{A}' \restriction C_i)$. But $\mathcal{A}' \restriction \mathsf{IN} \in J_{\mathsf{IN} \lhd C_i}(\mathcal{A}' \restriction C_i)$, because $\mathcal{A}'$ is itself a witness. □

NI^p and ND^p are not equivalent, as ND^p has an additional (implicit) existential quantifier. The witness execution showing that $\mathcal{A}' \restriction \mathsf{IN} \in J_{\mathsf{IN} \lhd C_i}(\mathcal{A} \restriction C_i)$ may differ from $\mathcal{A}'$ on channels $c \notin \mathsf{IN} \cup C_i$, namely the output channels c_j^{out} for $j \neq i$.

The symmetry of nondisclosure (Lemma 12) does not hold for NI^p and ND^p. For instance, relative to the GM purge for flow to d_i, there may be noninterference for inputs at d_j, while there is interference for flow from d_i to d_j. The asymmetry arises because the events to be concealed are only inputs at the source, while the observed events are both inputs and outputs [53].

The idea of p-noninterference is useful only when M is deterministic, since otherwise the outputs observed on c_i^{out} may differ even when $\mathsf{input}(\mathcal{A}) = \mathsf{input}(\mathcal{A}')$. For nondeterministic M, nondeducibility is more natural.

Purges and blurs. We can associate a blur operator f^p with each purge function p, such that ND^p amounts to respecting the blur operator f^p. We regard ND^p as saying that the input/output events on C_i tell d_i no more about all the inputs than the purged input $p(\mathcal{A})$ would disclose. We use a compatibility relation where the observed channels and the source channels overlap on c_i^{in}.

Definition 38. Let p be a purge function for d_i, and define the equivalence relation $\mathcal{R} \subseteq (\mathsf{IN\text{-}runs} \times \mathsf{IN\text{-}runs})$ by the condition: $\mathcal{R}(\mathcal{B}_1, \mathcal{B}_2)$ iff there exist $\mathcal{A}_1, \mathcal{A}_2 \in \mathsf{Exc}(\mathcal{F})$ s.t.:

$$\Big(\bigwedge_{j=1,2} \mathcal{B}_j = \mathcal{A}_j \restriction \mathsf{IN}\Big) \wedge \mathcal{A}_1 \approx^p \mathcal{A}_2. \tag{3}$$

Define $f^p \colon \mathcal{P}(\mathsf{IN\text{-}runs}) \to \mathcal{P}(\mathsf{IN\text{-}runs})$ to close under the $\mathcal{R}$-equivalence classes as in Lemma 22. ///

In fact, ND is a form of disclosure limited to within a blur:

Lemma 39. Let p be a purge function for domain d_i. For all $\mathcal{F}$, $\mathcal{F} \in \mathsf{ND}^p$ iff $\mathcal{F}$ f^p-limits IN-to-C_i flow.

Proof. **1. ND^p implies f^p-limited flow.** Suppose that $\mathcal{F} \in \mathsf{ND}^p$; $\mathcal{B}_i \in C_i$-runs; and $\mathcal{B}_1 \in J_{\mathsf{IN} \lhd C_i}(\mathcal{B}_i)$. If $\mathcal{B}_2 \in f^p(\mathcal{B}_1)$, we must show that $\mathcal{B}_2 \in J_{\mathsf{IN} \lhd C_i}(\mathcal{B}_i)$.

By Def. 38 there are $\mathcal{A}_1, \mathcal{A}_2$ such that

$$\mathcal{A}_1 \restriction \mathsf{IN} = \mathcal{B}_1, \quad \mathcal{A}_2 \restriction \mathsf{IN} = \mathcal{B}_2, \quad \mathcal{A}_1 \approx^p \mathcal{A}_2.$$

Furthermore, let $\mathcal{A}$ witness $\mathcal{B}_1 \in J_{\mathsf{IN} \lhd C_i}(\mathcal{B}_i)$. Then

$$\mathcal{A} \restriction \mathsf{IN} = \mathcal{B}_1 = \mathcal{A}_1 \restriction \mathsf{IN}.$$

So Def. 35, Clause 1 says $\mathcal{A} \approx^p \mathcal{A}_1$, and, by transitivity of $\approx^p$, also $\mathcal{A} \approx^p \mathcal{A}_2$. Since $\mathcal{F} \in \mathsf{ND}^p$,

$$\mathcal{A}_2 \restriction \mathsf{IN} \in J_{\mathsf{IN} \lhd C_i}(\mathcal{A} \restriction C_i).$$

That is, $\mathcal{B}_2 \in J_{\mathsf{IN} \lhd C_i}(\mathcal{B}_i)$ as required.

2. f^p-limited flow implies ND^p. Assume $J_{\mathsf{IN} \lhd C_i}(\mathcal{B}_i)$ is f^p-blurred for all $\mathcal{B}_i$. We must show, for all $\mathcal{A}_1, \mathcal{A}_2$,

$$\mathcal{A}_1 \approx^p \mathcal{A}_2 \text{ implies } (\mathcal{A}_2 \restriction \mathsf{IN}) \in J_{\mathsf{IN} \lhd C_i}(\mathcal{A}_1 \restriction C_i).$$

So choose executions with $\mathcal{A}_1 \approx^p \mathcal{A}_2$. By Def. 38, $\mathcal{R}(\mathcal{A}_1 \restriction \mathsf{IN}, \mathcal{A}_2 \restriction \mathsf{IN})$, since $\mathcal{A}_1, \mathcal{A}_2$ satisfy the condition. Thus,

$$\mathcal{A}_2 \restriction \mathsf{IN} \in J_{\mathsf{IN} \lhd C_i}(\mathcal{A}_1 \restriction C_i),$$

since $J_{\mathsf{IN} \lhd C_i}(\mathcal{A}_1 \restriction C_i)$ is f^p-blurred and contains $\mathcal{A}_1 \restriction \mathsf{IN}$. □

A frame $\mathcal{F}$ of this kind has definite inputs and outputs. The inputs are the events on IN, and the outputs are the events on $\mathsf{OUT} = \{c_x^{\mathsf{out}} : 1 \leq x \leq k\}$. We may thus regard it as a function from inputs to outputs (or, if M is non-deterministic, to sets of outputs). In this context, one could compare blurs with the partial equivalence relation model or abstract noninterference [24], which apply only when the system is a function mapping inputs to outputs. One can also regard some d_j as using a strategy for future inputs on c_j^{in} based on current outputs on c_j^{out}, recovering a form of nondeducibility on strategies [56].

Semantic sensitivity. Blur operators provide an explicit semantic representation of the information that will not be disclosed when flow is limited. This is in contrast to intransitive non-interference [47], [23], [54], which considers only whether the "$\hookrightarrow$ plumbing" among domains is correct.

Example 40. We represent Imaginary Weather Forecasting (IWF, see Example 24) as a state machine frame as in Fig. 5. It has domains $\{ws, \ell, p, cmp\}$ for the weather service, low-tier customer, premium-tier customer and compression service respectively. Let $\hookrightarrow$ be the smallest reflexive (but intransitive) relation extending Eqn. 4, where all reports must flow through the compression service:

$$ws \hookrightarrow cmp \hookrightarrow p \text{ and } cmp \hookrightarrow \ell. \tag{4}$$

The cmp service should compress reports lossily before sending them to ℓ and compress them losslessly for p. However, a faulty cmp may compress losslessly for both ℓ and p. Purge functions [23], [47], [54] do not distinguish between correct and faulty cmps. In both cases, all information from ws does indeed pass through cmp. The blur of Example 24, however, defines the desired goal semantically. With the faulty cmp, the high-resolution data compatible with the observation of ℓ is more sharply defined than an f-blurred set. ///

IX. Future Work

We have explored how the graph structure of a distributed system helps to constrain information flow. We have established the cut-blur principle. It allows us to propagate conclusions about limited disclosure from a cut set cut to more remote parts of the graph. These ideas are much more widely applicable than the simple examples that we have used here.

Quantitative treatment. It should be possible to equip frames with a quantitative information flow semantics. One obstacle here is that our execution model mixes some choices which are natural to view probabilistically—for instance, selection between different outputs when both are permitted by an LTS—with others that seem non-deterministic. The choice between receiving an input and emitting an output is an example of this, as is the choice between receiving inputs on different channels. This problem has been studied (e.g. [7], [8]), but a tractable semantics may require new ideas.

A Dynamic Model. Instead of building $\mathsf{ends}(\ell)$ into the frame, so that it remains fixed through execution, we may alternatively regard it as a component of the states of the individual locations. Let us regard $\mathsf{traces}(\ell)$ as generated by a labeled transition system $\mathsf{lts}(\ell)$. Then we may enrich the labels c, v so that they also involve a sequence of endpoints $\overline{p} \subseteq \mathcal{EP}$:

$$(c, v, \overline{p}).$$

The transition relation of $\mathsf{lts}(\ell)$ is then constrained to allow a transmission $(c, v, \overline{p})$ in a state only if $p \subseteq \mathsf{ends}(\ell)$ holds in that state, in which case $\overline{p}$ is omitted in the next state. A reception $(c, v, \overline{p})$ causes $\overline{p}$ to be added to the next state of the receiving location.

The cut-blur principle remains true in an important case: A set cut is an *invariant cut* between src and obs if it is an undirected cut, and moreover the execution of the frame preserves this property. Then the cut-blur principle holds in the dynamic model for invariant cuts.

This dynamic model suggests an analysis of security-aware software using object capabilities. Object capabilities may be viewed as endpoints $\mathsf{entry}(c)$. To use it, one sends a message to the object itself, which holds $\mathsf{exit}(c)$. To transfer a capability, one sends $\mathsf{entry}(c)$ over some c'.

McCamant and Ernst [34]'s quantitative approach generates a directed graph of this sort in memory at runtime. Providing a maximum over all possible runs would appear to depend on inferring some invariants on the structure of the graphs. Our methods might be helpful for this.

Cryptographic Masking. Encryption is not a blur. Encrypting messages makes their contents unavailable in locations lacking the decryption keys. In particular, locations lacking the decryption key may form a cut set between the source and destination of the encrypted message. However, at the destinations, where the keys are available, the messages can be decrypted and their contents observed. Thus, the cut-blur theorem implies it would be wrong to view encryption as a blur in this set-up: Its effects can be undone beyond the cut.

Several approaches are possible here. We would like to use the resulting set-up to reason about cryptographic voting systems, such as Helios and Prêt-à-Voter [2], [49].

We also intend to provide tool support for defining relevant blurs and establishing that they limit disclosure in several application areas.

Acknowledgments. We are grateful to Megumi Ando, Aslan Askarov, Stephen Chong, John Ramsdell, and Mitchell Wand. In particular, John Ramsdell formalized Thms. 28 and 32 in PVS, as well as Example 26; this suggested Example 34.

References

[1] P. Adao, C. Bozzato, R. F. Gian-Luca Dei Rossi, and F. Luccio, "Mignis: A semantic based tool for firewall configuration," in *IEEE CSF*, 2014.

[2] B. Adida, "Helios: Web-based open-audit voting," in *Usenix Security Symposium*. Usenix Association, 2008.

[3] A. Askarov and A. Sabelfeld, "Gradual release: Unifying declassification, encryption and key release policies," in *IEEE Symp. Security and Privacy*. IEEE, 2007, pp. 207–221.

[4] M. Balliu, M. Dam, and G. L. Guernic, "Epistemic temporal logic for information flow security," in *Programming Languages and Analysis for Security*. ACM, 2011.

[5] A. Bohannon, B. C. Pierce, V. Sjöberg, S. Weirich, and S. Zdancewic, "Reactive noninterference," in *ACM conference on Computer and Communications Security*. ACM, 2009, pp. 79–90.

[6] A. Bossi, R. Focardi, C. Piazza, and S. Rossi, "Verifying persistent security properties," *Computer Languages, Systems & Structures*, vol. 30, no. 3, pp. 231–258, 2004.

[7] R. Canetti, L. Cheung, D. K. Kaynar, M. Liskov, N. A. Lynch, O. Pereira, and R. Segala, "Analyzing security protocols using time-bounded task-PIOAs," *Discrete Event Dynamic Systems*, vol. 18, no. 1, pp. 111–159, 2008.

[8] K. Chatzikokolakis and C. Palamidessi, "Making random choices invisible to the scheduler," in *CONCUR*. Springer, 2007, pp. 42–58.

[9] S. Chong, J. Liu, A. C. Myers, X. Qi, K. Vikram, L. Zheng, and X. Zheng, "Secure web applications via automatic partitioning," in *ACM SIGOPS Operating Systems Review*, vol. 41. ACM, 2007, pp. 31–44.

[10] S. Chong and R. van der Meyden, "Deriving epistemic conclusions from agent architecture," in *Theoretical Aspects of Rationality and Knowledge*. ACM, 2009, pp. 61–70.

[11] ——, "Using architecture to reason about information security," Arxiv, Tech. Rep. arXiv:1409.0309, Sept. 2014.

[12] J. Clark, A. Essex, and C. Adams, "On the security of ballot receipts in e2e voting systems," in *Workshop on Trustworthy Elections (WOTE)*, 2007.

[13] G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. O'Hanlon, J. Ramsdell, A. Segall, J. Sheehy, and B. Sniffen, "Principles of remote attestation," *International Journal of Information Security*, vol. 10, no. 2, pp. 63–81, 2011.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2003.

[15] S. Delaune, S. Kremer, and M. Ryan, "Verifying privacy-type properties of electronic voting protocols," *Journal of Computer Security*, vol. 17, no. 4, pp. 435–487, 2009.

[16] D. E. Denning and P. J. Denning, "Certification of programs for secure information flow," *Communications of the ACM*, vol. 20, no. 7, pp. 504–513, 1977.

[17] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi, *Reasoning about Knowledge*. Cambridge, MA: MIT Press, 1995.

[18] R. Focardi and R. Gorrieri, "The compositional security checker: A tool for the verification of information flow security properties," *IEEE Transactions on Software Engineering*, vol. 23, no. 9, September 1997.

[19] ——, "Classification of security properties," in *Foundations of Security Analysis and Design*. Springer, 2001, pp. 331–396.

[20] J. A. Goguen and J. Meseguer, "Security policies and security models," in *IEEE Symposium on Security and Privacy*, 1982.

[21] J. D. Guttman and A. L. Herzog, "Rigorous automated network security management," *International Journal for Information Security*, vol. 5, no. 1–2, pp. 29–48, 2005.

[22] J. D. Guttman and M. E. Nadel, "What needs securing?" in *IEEE Computer Security Foundations Workshop*, 1988, pp. 34–57.

[23] J. T. Haigh and W. D. Young, "Extending the non-interference version of mls for sat," *IEEE Transactions on Software Engineering*, vol. 13, no. 2, 1987.

[24] S. Hunt and I. Mastroeni, "The PER model of abstract non-interference," in *Static Analysis Symposium SAS*, 2005, pp. 171–185.

[25] C. Irvine, D. Volpano, and G. Smith, "A sound type system for secure flow analysis," *Journal of Computer Security*, vol. 4, no. 3, pp. 1–21, 1996.

[26] J. Jacob, "Basic theorems about security," *Journal of Computer Security*, vol. 1, no. 3, pp. 385–411, 1992.

[27] D. M. Johnson and F. J. Thayer, "Security and the composition of machines." in *CSFW*, vol. 88, 1988, pp. 72–89.

[28] P. Li and S. Zdancewic, "Downgrading policies and relaxed noninterference," in *Principles of Programming Languages POPL*, 2005, pp. 158–170.

[29] H. Mantel, "Possibilistic definitions of security-an assembly kit," in *IEEE Computer Security Foundations*. IEEE, 2000, pp. 185–199.

[30] ——, "Preserving information flow properties under refinement," in *IEEE Computer Security Foundations*, 2001, pp. 78–91.

[31] ——, "On the composition of secure systems," in *IEEE Security and Privacy*, 2002, pp. 88–101.

[32] H. Mantel, D. Sands, and H. Sudbrock, "Assumptions and guarantees for compositional noninterference," in *IEEE Computer Security Foundations Symposium*, 2011, pp. 218–232.

[33] F. Mattern, "Virtual time and global states of distributed systems," in *Proc. Workshop on Parallel and Distributed Algorithms*, M. Cosnard, Ed., North-Holland / Elsevier, 1989, pp. 215–226, (Reprinted in: Z. Yang, T.A. Marsland (Eds.), "Global States and Time in Distributed Systems", IEEE, 1994, pp. 123-133.).

[34] S. McCamant and M. D. Ernst, "Quantitative information flow as network flow capacity," *ACM SIGPLAN Notices (PLDI)*, vol. 43, no. 6, pp. 193–205, 2008.

[35] D. McCullough, "Specifications for multi-level security and a hook-up property," in *IEEE Symposium on Security and Privacy*, 1987, pp. 161–161.

[36] ——, "Noninterference and the composability of security properties," in *IEEE Symposium on Security and Privacy*, 1988, pp. 177–186.

[37] J. McLean, "A general theory of composition for trace sets closed under selective interleaving functions," in *IEEE Symp. Security and Privacy*, 1994, pp. 79–93.

[38] M. Moran, J. Heather, and S. Schneider, "Automated anonymity verification of the ThreeBallot and VAV voting systems," *Software and Systems Modeling*, 2015, forthcoming.

[39] C. Morgan, "The shadow knows: Refinement of ignorance in sequential programs," in *Mathematics of program construction*. Springer, 2006, pp. 359–378.

[40] S. Owre, J. M. Rushby, and N. Shankar, "PVS: A prototype verification system," in *Conference on Automated Deduction*, ser. LNAI, Jun. 1992.

[41] W. Rafnsson and A. Sabelfeld, "Compositional information-flow security for interactive systems," in *IEEE Symp. CSF*, July 2014, pp. 277–292.

[42] R. L. Rivest and W. D. Smith, "Three voting protocols: ThreeBallot, VAV, and Twin," `http://people.csail.mit.edu/rivest/pubs/RS07.pdf`, 2007.

[43] A. W. Roscoe and M. H. Goldsmith, "What is intransitive noninterference?" in *12th IEEE Computer Security Foundations Workshop*. IEEE CS Press, June 1999, pp. 228–238.

[44] A. W. Roscoe, "CSP and determinism in security modelling," in *IEEE Security and Privacy*. IEEE, 1995, pp. 114–127.

[45] A. Roscoe, J. Woodcock, and L. Wulf, "Non-interference through determinism," in *Computer Security—ESORICS 94*. Springer, 1994, pp. 31–53.

[46] S. Rossi and D. Macedonio, "Information flow security for service compositions." in *ICUMT*, 2009, pp. 1–8.

[47] J. Rushby, *Noninterference, transitivity, and channel-control security policies*. SRI International, Computer Science Laboratory, 1992.

[48] P. Ryan and S. Schneider, "Process algebra and non-interference," *J. Comput. Secur.*, vol. 9, no. 1-2, pp. 75–103, Jan. 2001.

[49] P. Y. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia, "Prêt à voter: a voter-verifiable voting system," *Information Forensics and Security, IEEE Transactions on*, vol. 4, no. 4, pp. 662–673, 2009.

[50] A. Sabelfeld and A. C. Myers, "Language-based information-flow security," *IEEE Journal on Selected Areas in Communication*, vol. 21, no. 1, pp. 5–19, January 2003.

[51] ——, "A model for delimited information release," in *Software Security-Theories and Systems*. Springer, 2004, pp. 174–191.

[52] A. Sabelfeld and D. Sands, "Declassification: Dimensions and principles," *Journal of Computer Security*, vol. 17, no. 5, pp. 517–548, 2009.

[53] D. Sutherland, "A model of information," in *9th National Computer Security Conference*. National Institute of Standards and Technology, 1986.

[54] R. van der Meyden, "What, indeed, is intransitive noninterference?" in *Computer Security–ESORICS 2007*. Springer, 2007, pp. 235–250.

[55] ——, "Architectural refinement and notions of intransitive noninterference," *Formal Aspects of Computing*, vol. 24, no. 4-6, pp. 769–792, 2012.

[56] J. T. Wittbold and D. M. Johnson, "Information flow in nondeterministic systems," in *IEEE Symp. Security and Privacy*, 1990, pp. 144–144.

[57] A. Zakinthinos and E. S. Lee, "The composability of non-interference," *Journal of Computer Security*, vol. 3, no. 4, pp. 269–281, 1995.

[58] S. Zdancewic, L. Zheng, N. Nystrom, and A. C. Myers, "Secure program partitioning," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 3, pp. 283–328, 2002.

Appendix

We gather here additional lemmas, and a few longer proofs.

Lemma 13.

1. *Suppose $C_0 \subseteq C_1$ and $C'_0 \subseteq C'_1$. If $\mathcal{F}$ has no disclosure from C_1 to C'_1, then $\mathcal{F}$ has no disclosure from C_0 to C'_0.*
2. *When $C_1, C_2, C_3 \subseteq \mathcal{CH}$,*

$$J_{C_3 \triangleleft C_1}(\mathcal{B}_1) \subseteq \bigcup_{\mathcal{B}_2 \in J_{C_2 \triangleleft C_1}(\mathcal{B}_1)} J_{C_3 \triangleleft C_2}(\mathcal{B}_2).$$

Proof. **1.** Suppose $\mathcal{B}_0$ is a C_0-run, and $\mathcal{B}'_0$ is a C'_0-run. We want to show that $\mathcal{B}'_0 \in J_{C'_0 \triangleleft C_0}(\mathcal{B}_0)$.

Since they are local runs, there exist $\mathcal{A}_0, \mathcal{A}'_0 \in \mathsf{Exc}(\mathcal{F})$ such that $\mathcal{B}_0 = \mathcal{A}_0 \upharpoonright C_0$ and $\mathcal{B}'_0 = \mathcal{A}'_0 \upharpoonright C'_0$. But let $\mathcal{B}_1 = \mathcal{A}_0 \upharpoonright C_1$ and let $\mathcal{B}'_1 = \mathcal{A}'_0 \upharpoonright C'_1$. By no-disclosure, $\mathcal{B}'_1 \in J_{C'_1 \triangleleft C_1}(\mathcal{B}_1)$. So there is an $\mathcal{A} \in \mathsf{Exc}(\mathcal{F})$ such that $\mathcal{B}_1 = \mathcal{A} \upharpoonright C_1$ and $\mathcal{B}'_1 = \mathcal{A} \upharpoonright C'_1$.

However, then $\mathcal{A}$ witnesses for $\mathcal{B}'_0 \in J_{C'_0 \triangleleft C_0}(\mathcal{B}_0)$: After all, since $C_0 \subseteq C_1$, $\mathcal{A} \upharpoonright C_0 = (\mathcal{A} \upharpoonright C_1) \upharpoonright C_0$. Similarly for the primed versions.

2. Suppose that $\mathcal{B}_3 \in J_{C_3 \triangleleft C_1}(\mathcal{B}_1)$, so that there exists an $\mathcal{A} \in \mathsf{Exc}(\mathcal{F})$ such that $\mathcal{B}_1 = \mathcal{A} \upharpoonright C_1$ and $\mathcal{B}_3 = \mathcal{A} \upharpoonright C_3$. Letting $\mathcal{B}_2 = \mathcal{A} \upharpoonright C_2$, the execution $\mathcal{A}$ ensures that $\mathcal{B}_2 \in J_{C_2 \triangleleft C_1}(\mathcal{B}_1)$ and $\mathcal{B}_3 \in J_{C_3 \triangleleft C_2}(\mathcal{B}_2)$. □

We now consider different frames $\mathcal{F}_1, \mathcal{F}_2$ that overlap on a common subset L_0, and show how local runs in the two can be pieced together. In this context, we use the notation of Def. 30, such as left_0 for channels between locations L_0 shared between $\mathcal{F}_1$ and $\mathcal{F}_2$, cut_0 for the set of channels forming the boundary, and right_i for the channels unattached to L_0 in $\mathcal{F}_i$.

Lemma 41. Let L_0 be shared between frames $\mathcal{F}_1, \mathcal{F}_2$. Let

$$\mathcal{B}_{lc} \in (\mathsf{left} \cup \mathsf{cut})\text{-}\mathsf{runs}_1 \text{ and } \mathcal{B}_{rc} \in (\mathsf{right}_2 \cup \mathsf{cut})\text{-}\mathsf{runs}_2$$

agree on cut, i.e. $\mathcal{B}_{lc} \upharpoonright \mathsf{cut} = \mathcal{B}_{rc} \upharpoonright \mathsf{cut}$. Then there is an $\mathcal{A} \in \mathsf{Exc}(\mathcal{F}_2)$ such that

$$\mathcal{B}_{lc} = \mathcal{A} \upharpoonright (\mathsf{left} \cup \mathsf{cut}) \text{ and } \mathcal{B}_{rc} = \mathcal{A} \upharpoonright (\mathsf{right}_2 \cup \mathsf{cut}).$$

Proof. Since $\mathcal{B}_{lc}$ and $\mathcal{B}_{rc}$ are local runs of $\mathcal{F}_1, \mathcal{F}_2$ resp., they are restrictions of executions, so choose $\mathcal{A}_1 \in \mathsf{Exc}(\mathcal{F}_1)$ and $\mathcal{A}_2 \in \mathsf{Exc}(\mathcal{F}_2)$ so that $\mathcal{B}_{lc} = \mathcal{A}_1 \upharpoonright (\mathsf{left} \cup \mathsf{cut})$ and $\mathcal{B}_{rc} = \mathcal{A}_2 \upharpoonright (\mathsf{right}_2 \cup \mathsf{cut})$. Now define $\mathcal{A}$ by stipulating:

$$\begin{aligned} \mathsf{ev}(\mathcal{A}) &= \mathsf{ev}(\mathcal{B}_{lc}) \cup \mathsf{ev}(\mathcal{B}_{rc}) && (5)\\ \preceq_{\mathcal{A}} &= \text{the least partial order extending } \preceq_{\mathcal{B}_{lc}} \cup \preceq_{\mathcal{B}_{rc}}. && (6) \end{aligned}$$

Since $\mathcal{A}_1, \mathcal{A}_2$ agree on cut, $\mathsf{ev}(\mathcal{A}) = \mathsf{ev}(\mathcal{B}_{lc} \upharpoonright \mathsf{left}) \cup \mathsf{ev}(\mathcal{B}_{rc})$, and we could have used the latter as an alternate definition of $\mathsf{ev}(\mathcal{A})$, as well as the symmetric restriction of $\mathcal{B}_{rc}$ to right_2 leaving $\mathcal{B}_{lc}$ whole.

The definition of $\preceq_{\mathcal{A}}$ as a partial order is sound, because there are no cycles in the union (6). Cycles would require $\mathcal{A}_1$ and $\mathcal{A}_2$ to disagree on the order of events in their restrictions to cut, contrary to assumption. Likewise, the finite-predecessor property is preserved: $x_0 \preceq_{\mathcal{A}} x_1$ iff x_0, x_1 belong to the same $\mathcal{B}_{?c}$ and are ordered there, or else there is an event in $\mathcal{B}_{?c} \upharpoonright \mathsf{cut}$ which comes between them. So the events preceding x_1 form the finite union of finite sets. Thus, $\mathcal{A} \in \mathsf{ES}(\mathcal{F}_2)$.

Moreover, $\mathcal{A}$ is an execution $\mathcal{A} \in \mathsf{Exc}(\mathcal{F}_2)$: If $\ell \in L_0$, then $\mathsf{proj}(\mathcal{A}, \ell) = \mathsf{proj}(\mathcal{B}_{lc}, \ell)$, and the latter is a trace in $\mathsf{traces}_1(\ell) = \mathsf{traces}_2(\ell)$. If $\ell \notin L_0$, then $\mathsf{proj}(\mathcal{A}, \ell) = \mathsf{proj}(\mathcal{B}_{rc}, \ell)$, and the latter is a trace in $\mathsf{traces}_2(\ell)$.

There is no ℓ with channels in both left and right_2. □

What makes this proof work? Any one location either has all of its channels lying in $\mathsf{left}_0 \cup \mathsf{cut}_0$ or else all of them lying in $\mathsf{right}_i \cup \mathsf{cut}$. When piecing together the two executions $\mathcal{A}_1, \mathcal{A}_2$ into a single execution $\mathcal{A}$, no location needs to be able to execute a trace that comes partly from $\mathcal{A}_1$ and partly from $\mathcal{A}_2$. This is what determines our definition of cuts using the undirected graph $\mathsf{ungr}(\mathcal{F})$.

We next prove the two-frame analog of Lemma 15.

Lemma 31. *Let L_0 be shared between frames $\mathcal{F}_1, \mathcal{F}_2$. Let* $\mathsf{src} \subseteq \mathsf{left}$*, and* $\mathcal{B}_c \in \mathsf{cut}_0\text{-}\mathsf{runs}_1 \cap \mathsf{cut}_0\text{-}\mathsf{runs}_2$.

1. $J^1_{\mathsf{src} \triangleleft \mathsf{cut}_0}(\mathcal{B}_c) = J^2_{\mathsf{src} \triangleleft \mathsf{cut}_0}(\mathcal{B}_c)$.
2. *Assume* $\mathsf{cut}_0\text{-}\mathsf{runs}(\mathcal{F}_2) \subseteq \mathsf{cut}_0\text{-}\mathsf{runs}(\mathcal{F}_1)$. *Let* $\mathsf{obs} \subseteq \mathsf{right}_2$*, and* $\mathcal{B}_o \in \mathsf{obs}\text{-}\mathsf{runs}_2$. *Then*

$$J^2_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o) = \bigcup_{\mathcal{B}_c \in J^2_{\mathsf{cut}_0 \triangleleft \mathsf{obs}}(\mathcal{B}_o)} J^1_{\mathsf{src} \triangleleft \mathsf{cut}_0}(\mathcal{B}_c).$$

Proof. **1.** First, we show that $\mathcal{B}_s \in J^1_{\mathsf{src} \triangleleft \mathsf{cut}_0}(\mathcal{B}_c)$ implies $\mathcal{B}_s \in J^2_{\mathsf{src} \triangleleft \mathsf{cut}_0}(\mathcal{B}_c)$.

Let $\mathcal{A}_1$ witness for $\mathcal{B}_s \in J^1_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c)$, and let $\mathcal{A}_2$ witness for $\mathcal{B}_c \in \mathsf{cut}\text{-}\mathsf{runs}_2$. Define

$$\mathcal{B}_{lc} = \mathcal{A}_1 \upharpoonright (\mathsf{left} \cup \mathsf{cut}) \text{ and } \mathcal{B}_{rc} = \mathcal{A}_2 \upharpoonright (\mathsf{right}_2 \cup \mathsf{cut}).$$

Now the assumptions for Lemma 41 are satisfied. So let $\mathcal{A} \in \mathsf{Exc}(\mathcal{F}_2)$ restrict to $\mathcal{B}_{lc}$ and $\mathcal{B}_{rc}$ as in the conclusion. Thus, $\mathcal{A} \upharpoonright \mathsf{src} = \mathcal{B}_s$.

For the converse, we rely on the symmetry of "L_0 is shared between frames $\mathcal{F}_1, \mathcal{F}_2$."

2. By the assumption, whenever $\mathcal{B}_c \in J^2_{\mathsf{cut} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$, then also $\mathcal{B}_c \in \mathsf{cut}\text{-}\mathsf{runs}_1$. Thus, we can apply part 1 after using Lemma 13:

$$\begin{aligned} J^2_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o) &\subseteq \bigcup_{\mathcal{B}_c \in J^2_{\mathsf{cut}_0 \triangleleft \mathsf{obs}}(\mathcal{B}_o)} J^2_{\mathsf{src} \triangleleft \mathsf{cut}_0}(\mathcal{B}_c) \\ &\subseteq \bigcup_{\mathcal{B}_c \in J^2_{\mathsf{cut}_0 \triangleleft \mathsf{obs}}(\mathcal{B}_o)} J^1_{\mathsf{src} \triangleleft \mathsf{cut}_0}(\mathcal{B}_c). \end{aligned}$$

For the reverse inclusion, assume that $\mathcal{B}_s \in J^1_{\mathsf{src} \triangleleft \mathsf{cut}_0}(\mathcal{B}_c)$, where $\mathcal{B}_c \in J^2_{\mathsf{cut}_0 \triangleleft \mathsf{obs}}(\mathcal{B}_o)$. Thus, we can apply Lemma 41, obtaining $\mathcal{A} \in \mathsf{Exc}(\mathcal{F}_2)$ which agrees with $\mathcal{B}_s$, $\mathcal{B}_c$, and $\mathcal{B}_o$. So $\mathcal{A}$ witnesses for $\mathcal{B}_s \in J^2_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$. □

We now turn to the one-frame corollary, which we presented earlier as Lemma 15.

Lemma 15. *Let* cut *be an undirected cut between* src, obs*, and let* $\mathcal{B}_o \in$ src-runs*. Then*

$$J_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o) = \bigcup_{\mathcal{B}_c \in J_{\mathsf{cut} \triangleleft \mathsf{obs}}(\mathcal{B}_o)} J_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c).$$

Proof. Define L_0 to be the smallest set of locations such that

1. $\ell \in L_0$ if $\mathsf{chans}(\ell) \cap \mathsf{src} \neq \emptyset$;
2. L_0 is closed under reachability by paths that do not traverse cut.

L_0 is shared between $\mathcal{F}$ and itself. Moreover, for the set of channels cut_0 defined in Def. 30, we have $\mathsf{cut}_0 \subseteq \mathsf{cut}$: cut_0 is the part of cut that actually lies on the boundary of L_0.

By Lemma 31, we have

$$J_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o) = \bigcup_{\mathcal{B}_c \in J_{\mathsf{cut}_0 \triangleleft \mathsf{obs}}(\mathcal{B}_o)} J_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c).$$

Since $\mathsf{cut}_0 \subseteq \mathsf{cut}$,

$$\bigcup_{\mathcal{B}_c \in J_{\mathsf{cut}_0 \triangleleft \mathsf{obs}}(\mathcal{B}_o)} J_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c) \subseteq \bigcup_{\mathcal{B}_c \in J_{\mathsf{cut} \triangleleft \mathsf{obs}}(\mathcal{B}_o)} J_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c).$$

For the converse, suppose that $\mathcal{B}_s \in J_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c)$, for $\mathcal{B}_c \in J_{\mathsf{cut} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$. Then there is $\mathcal{A}$ such that $\mathcal{A} \upharpoonright \mathsf{src} = \mathcal{B}_s$ and $\mathcal{A} \upharpoonright \mathsf{obs} = \mathcal{B}_o$. Thus, $\mathcal{B}_s \in J_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{A} \upharpoonright \mathsf{cut}_0)$ and $\mathcal{A} \upharpoonright \mathsf{cut}_0 \in J_{\mathsf{cut}_0 \triangleleft \mathsf{obs}}(\mathcal{B}_o)$. □

The cut-blur principle is also the one-frame corollary of Thm. 32. The proofs are very similar.

Theorem 32. *Suppose that* L_0 *is shared between frames* $\mathcal{F}_1, \mathcal{F}_2$*, and assume* $\mathsf{cut\text{-}runs}(\mathcal{F}_2) \subseteq \mathsf{cut\text{-}runs}(\mathcal{F}_1)$*. Consider any* $\mathsf{src} \subseteq \mathsf{left}$ *and* $\mathsf{obs} \subseteq \mathsf{right}_2$*. If* $\mathcal{F}_1$ *f-limits* src*-to-*cut *flow, then* $\mathcal{F}_2$ *f-limits* src*-to-*obs *flow.*

Proof. By the hypothesis, f is a blur operator. Letting $\mathcal{B}_o \in \mathsf{obs\text{-}runs}_2$, we want to show that $J^2_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$ is an f-blurred set, i.e. $J^2_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o) = f(J^2_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o))$.

For convenience, let $S_c = J^2_{\mathsf{cut} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$. By Lemma 31,

$$J^2_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o) = \bigcup_{\mathcal{B}_c \in S_c} J^1_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c);$$

thus, we must show that the latter is f-blurred. By the assumption that each $J^1_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c)$ is f-blurred, we have $J^1_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c) = f(J^1_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c))$. Using this and the union property (Eqn. 1):

$$\begin{aligned} \bigcup_{\mathcal{B}_c \in S_c} J^1_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c) &= \bigcup_{\mathcal{B}_c \in S_c} f(J^1_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c)) \\ &= f(\bigcup_{\mathcal{B}_c \in S_c} J^1_{\mathsf{src} \triangleleft \mathsf{cut}}(\mathcal{B}_c)), \end{aligned}$$

Hence, $J^2_{\mathsf{src} \triangleleft \mathsf{obs}}(\mathcal{B}_o)$ is f-blurred. □