Only Connect, Securely

Chandrika Bhardwaj, Sanjiva Prasad

Only Connect, Securely*

Chandrika Bhardwaj[0000−0003−0365−5478] and Sanjiva Prasad[0000−0001−5887−1237]

Indian Institute of Technology Delhi, India
{chandrika,sanjiva}@cse.iitd.ac.in

Abstract. The *lattice model* proposed by Denning in her seminal work provided secure information flow analyses with an intuitive and uniform mathematical foundation. Different organisations, however, may employ quite different security lattices. In this paper, we propose a connection framework that permits different organisations to exchange information while maintaining both security of information flow as well as their autonomy in formulating and maintaining security policies. Our prescriptive framework is based on the rigorous mathematical framework of *Lagois connections* given by Melton, together with a simple operational model for transferring object data between domains. The merit of this formulation is that it is simple, minimal, adaptable and intuitive, and provides a formal framework for establishing secure information flow across autonomous interacting organisations. We show that our framework is semantically sound, by proving that the connections proposed preserve standard correctness notions such as non-interference.

Keywords: Security class lattice · Information flow · Lagois connection · Atomic operations · Non-interference.

1 Introduction

Denning's seminal work [7] proposed *complete lattices*[1] as the appropriate mathematical framework for questions regarding *secure information flow* (SIF), *i.e.*, only authorised flows of information are possible. An information flow model (IFM) is characterised as $\langle N, P, SC, \sqcup, \sqsubseteq \rangle$ where: *Storage objects* in N are assigned *security classes* drawn from a (finite) complete lattice SC. P is a set of processes (also assigned security classes as clearances). The partial ordering $\sqsubseteq$ represents *permitted flows* between classes; reflexivity and transitivity capture intuitive aspects of information flow; antisymmetry helps avoid redundancies in the framework, and the join operation $\sqcup$ succinctly captures the combination of information belonging to different security classes in arithmetic, logical and computational operations. This lattice model provides an abstract uniform framework that identifies the commonalities of the variety of analyses for different applications – *e.g.*, confidentiality and trust – whether at the *language*

* Supported by Indo-Japanese project *Security in the IoT Space*, DST, Govt of India.

[1] Denning showed that the proposed structures, namely complete join semi-lattices with a least element, are in fact complete lattices.

level or at a *system* level. In the ensuing decades, the vast body of secure information flow analyses has been built on these mathematical foundations, with the development of a plethora of static and dynamic analysis techniques for programming languages [13, 15, 17, 19–21], operating systems [2, 8, 12, 20, 25], databases [22], and hardware architectures [9, 27], etc. The soundness of this lattice model was expressed in terms of semantic notions of system behaviour, for instance, as properties like non-interference [10] by Volpano *et al* [23] and others. Alternative semantic notions of security such as safety properties have been proposed as well, *e.g.*, [1], but for brevity we will not explore these further.

The objective of this paper is to propose a simple way in which large-scale distributed secure systems can be built by connecting component systems in a secure and modular manner. Our work begins with the observation that large information systems are not monolithic: Different organisations define their own information flow policies independently, and subsequently collaborate or federate with one another to exchange information. In general, the security classes and the lattices of any two organisations may be quite different — *there is no single universal security class lattice*. Moreover, *modularity* and *autonomy* are important requirements since each organisation would naturally wish to retain control over its own security policies and the ability to redefine them. Therefore, fusing different lattices by taking their union is an unsatisfactory approach, more so since the security properties of application programs would have to be re-established in this possibly enormous lattice.

When sharing information, most organisations limit the cross-domain communications to a limited set of security classes (which we call *transfer* classes). In order to ensure that shared data are not improperly divulged, two organisations usually negotiate agreements or memorandums of understanding (MoUs), promising that they will respect the security policies of the other organisation. We argue that a good notion of secure connection should require reasoning only about those flows from just the transfer classes mentioned in a MoU. Usually, cross-domain communication involves downgrading the security class of privileged information to public information using primitives such as encryption, and then upgrading the information to a suitable security class in the other domain. Such approaches, however, do not gel well with correctness notions such as non-interference. Indeed the question of how to translate information between security classes of different lattices is interesting [6].

Contributions of this paper. In this paper, we propose a simple framework and sufficient conditions under which secure flow guarantees can be enforced without exposing the complexities and details of the component information flow models. The framework consists of (1) a way to connect security classes of one organisation to those in another while satisfying intuitive requirements; (2) a simple language that extends the operations within an organisation with primitives for transferring data between organisations; and (3) a type system and operational model for these constructs, which we use to establish that the framework conserves security.

In §2, we first identify, using intuitive examples, violations in secure flow that may arise when two secure systems are permitted to exchange information in both directions. Based on these lacunae, we formulate *security* and *precision* requirements for secure bidirectional flow. We then propose a framework that guarantees the absence of such policy violations, without impinging on the autonomy of the individual systems, without the need for re-verifying the security of the application procedures in either of the domains, and confining the analysis to only the transfer classes involved in potential exchange of data. Our approach is based on *monotone functions* and an elegant theory of *connections* [16] between the security lattices. Theorem 1 shows that *Lagois connections* between the security lattices satisfy the security and precision requirements.

We present in §3 a minimal operational language consisting of a small set of *atomic primitives* for effecting the transfer of data between domains. The framework is simple and can be adapted for establishing secure connections between distributed systems at any level of abstraction (language, system, database, ...). We assume each domain uses *atomic transactional operations* for object manipulation and intra-domain computation. The primitives of our model include reliable communication between two systems, transferring object data in designated *output* variables of one domain to designated *input* variables of a specified security class in the other domain. We also assume a generic set of operations in each domain for copying data from input variables to domain objects, and from domain objects to output variables. To avoid interference between inter-domain communication and the computations within the domains, we assume that the sets of designated input and output variables are all mutually exclusive of one another, and also with the program/system variables used in the computations within each domain. Thus by design we avoid the usual suspects that cause interference and insecure transfer of data. The operational description of the language consists of the primitives together with their execution rules (§3.1).

The correctness of our framework is demonstrated by expressing soundness (with respect to the operational semantics) of a type system (§3.2), stated in terms of the security lattices and their connecting functions. In particular, Theorem 7 shows the standard semantic property of *non-interference* in *both domains* holds of all operational behaviours. We adapt and *extend* the approach taken by Volpano *et al* [23] to encompass systems coupled using the Lagois connection conditions, and (assuming atomicity of the data transfer operations) show that *security is conserved*. Since our language is a minimal imperative model with atomic transactions, reads and writes as the basic elements, we are able to work with a simplified version of the type system of Volpano *et al*. In particular, our language does not include conditional constructs in the transfer of data between domains, and assumes all conditional computation is absorbed within atomic intradomain transactions. Thus, we do not have to concern ourselves with issues of implicit flows that arise due to branching structures (*e.g.*, conditionals and loops in programming language level security, pipeline mispredictions at the architectural level, etc.) While non-interference is the property addressed in this

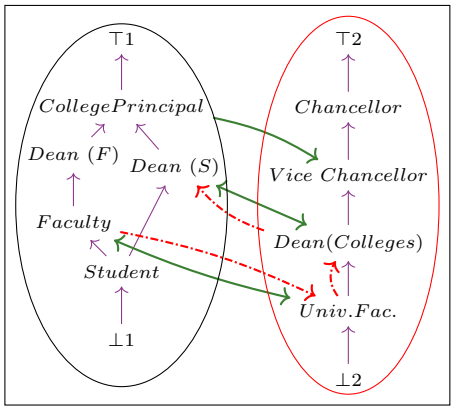

Fig. 1. Solid green arrows represent permitted flows according to the information exchange arrangement between a college and a university. Red dash-dotted arrows highlight a *new* flow that is a security violation.

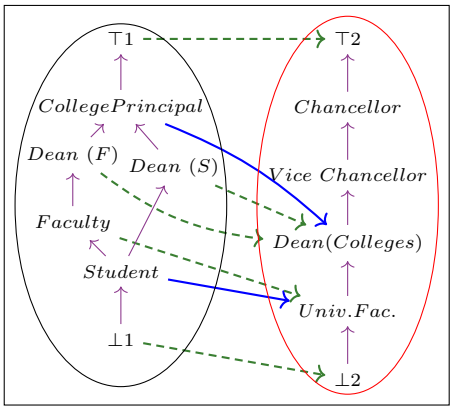

Fig. 2. Unidirectional flow: If the solid blue arrows denote identified flows connecting important classes, then the dashed green arrows are constrained by monotonicity to lie between them.

paper, we believe that our formulation is general enough to be applicable to other behavioural notions of secure information flow as a safety property [1].

In §4, we briefly review some related work. We conclude in §5 with a discussion on our approach and directions for future work.

2 Lagois Connections and All That

Motivating Examples. Consider a university system in which students study in semi-autonomously administered colleges (one such is C) that are affiliated to a university (U). The university also has "university professors" with whom students can take classes. We assume each institution has established the security of its information flow mechanisms and policies.

We first observe that formulating an agreement *between* the institutions which respects the flow policies within each institution is not entirely trivial. Consider an arrangement where the College *Faculty* and *University Faculty* can share information (say, course material and examinations), and the *Dean of Colleges* in the University can exchange information (*e.g.*, students' official grade-sheets) with the college's *Dean of Students*. Even such an apparently reasonable arrangement suffers from insecurities, as illustrated in Fig. 1 by the flow depicted using dashed red arrows, where information can flow from the college's *Faculty* to the college's *Dean of Students*. (Moral: internal structure of the lattices matters.)

As long as information flows *unidirectionally* from colleges to the University, *monotone functions* from the security classes of the college lattice C to those

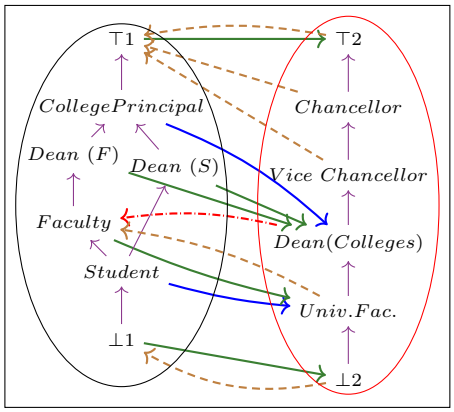

Fig. 3. The solid blue/green and dashed brown/red arrows respectively define monotone functions in each direction. However, the dash-dotted red arrow highlights a flow that is a security violation.

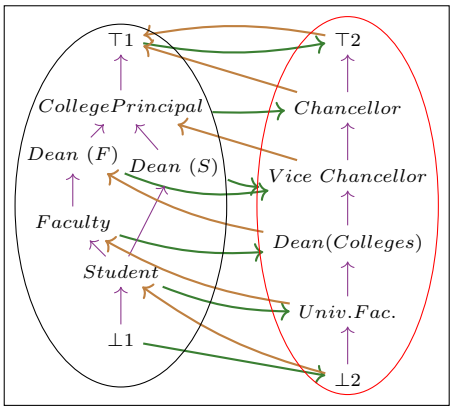

Fig. 4. The arrows define a secure and precise connection. However, the security classification escalates quickly in a few round-trips when information can flow in both directions.

in the university security lattice U *suffice* to ensure secure information flow. A function $\alpha : C \to U$ is called *monotone* if whenever $sc_1 \sqsubseteq sc_2$ in C then $\alpha(sc_1) \sqsubseteq' \alpha(sc_2)$ in U.[2] Monotonicity also constrains possible flows between classes of the two domains, once certain important flows between certain classes have been identified (see Fig. 2). Moreover, since monotone functions are closed under composition, one can chain them to create secure *unidirectional* information flow connections through a series of administrative domains. Monotonicity is a basic principle adopted for information flow analyses, *e.g.* [13].

However, when there is "blowback" of information, mere monotonicity is *inadequate* for ensuring SIF. Consider the bidirectional flow situation in Fig. 3, where data return to the original domain. Monotonicity of both functions $\alpha : C \to U$ and $\gamma : U \to C$ does *not* suffice for security because the composition $\gamma \circ \alpha$ may *not* be non-decreasing. In Fig. 3, both α and γ are monotone but their composition can lead to information leaking from a higher class, *e.g.*, *College Principal*, to a lower class, *e.g.*, *Faculty* within C — an outright violation of the college's security policy. Similarly, composition $\alpha \circ \gamma$ may lead to violation of the University's security policy.

Requirements. We want to ensure that any "round-trip" flow of information, *e.g.*, from a domain L to M and back to L, is a permitted flow in the lattice L, from where the data originated. Thus we require the following (tersely stated) "security conditions" **SC1** and **SC2** on $\alpha : L \to M$ and $\gamma : M \to L$, which preclude any violation of the security policies of both the administrative domains

[2] Note that it is not necessary for the function α to be total or surjective.

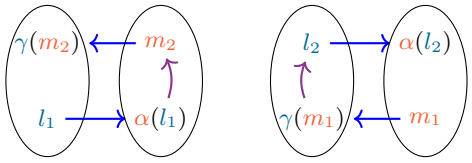

Fig. 5. Secure flow conditions: (**sc1**) $l_1 \sqsubseteq \gamma(m_2)$ (**sc2**) $m_1 \sqsubseteq' \alpha(l_2)$.

(see Fig. 5):

$$\textbf{SC1}\quad \lambda l.l \sqsubseteq \gamma \circ \alpha \qquad\qquad \textbf{SC2}\quad \lambda m.m \sqsubseteq \alpha \circ \gamma$$

In other words, the data can flow only in accordance with the flows permitted by the ordering relations of the two lattices.
We also desire *precision*, based on a principle of least privilege escalation — if data are exchanged between the two domains without any computation done on them, then the security level should not be needlessly raised. Precision is important for meaningful and useful analyses; otherwise data would be escalated to security classes which permit very restricted access.

$$\begin{array}{ll}\textbf{PC1} & \alpha(l_1) = \bigsqcup \{m_1 \mid \gamma(m_1) = l_1\},\ \ \forall l_1 \in \gamma[M] \\ \textbf{PC2} & \gamma(m_1) = \bigsqcup \{l_1 \mid \alpha(l_1) = m_1\},\ \ \forall m_1 \in \alpha[L]\end{array}$$

Further, if the data were to go back and forth between two domains more than once, the security classes to which data belong should not become increasingly restrictive after consecutive bidirectional data sharing (See Fig. 4, which shows monotone functions that keep climbing up to the top). This convergence requirement may be stated informally as conditions **CC1** and **CC2**, requiring *fixed points* for the compositions $\gamma \circ \alpha$ and $\alpha \circ \gamma$. Since security lattices are finite, **CC1** and **CC2** necessarily hold – such fixed points exist, though perhaps only at the topmost elements of the lattice. We would therefore desire a stronger requirement, where fixed points are reached as low in the orderings as possible.

Galois connections aren't the answer. Any discussion on a pair of partial orders linked by a pair of monotone functions suggests the notion of a Galois connection, an elegant and ubiquitous mathematical structure that finds use in computing, particularly in static analyses. However, Galois connections are not the appropriate structure for bidirectional informational flow control.

Let L and M be two complete security class lattices, and $\alpha : L \to M$ and $\gamma : M \to L$ be two monotone functions such that (L, α, γ, M) forms a Galois connection. Recall that a Galois connection satisfies the condition

$$\textbf{GC1}\quad \forall l_1 \in L, m_1 \in M, \quad \alpha(l_1) \sqsubseteq' m_1 \iff l_1 \sqsubseteq \gamma(m_1)$$

So in a Galois connection we have $\alpha(\gamma(m_1)) \sqsubseteq' m_1 \iff \gamma(m_1) \sqsubseteq \gamma(m_1)$. Since $\gamma(m_1) \sqsubseteq \gamma(m_1)$ holds trivially, we get $\alpha(\gamma(m_1)) \sqsubseteq' m_1$. If $\alpha(\gamma(m_1)) \neq' m_1$ then $\alpha(\gamma(m_1)) \sqsubset' m_1$ (strictly), which would violate secure flow requirement **SC2**. Fig. 6 illustrates such a situation.

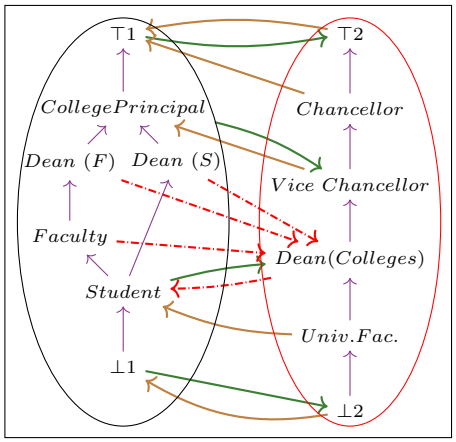

Fig. 6. The arrows between the domains define a Galois Connection. However, the red dash-dotted arrows highlight flow security violations when information can flow in both directions.

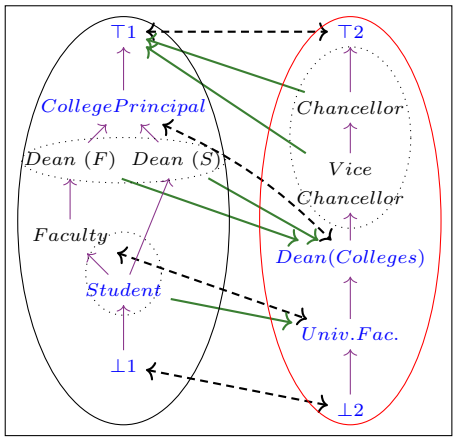

Fig. 7. A useful increasing Lagois connection for sharing data. Dashed black arrows define permissible flows between buds.

Why not Galois insertions? Now suppose L and M are two complete security class lattices, and $\alpha : L \to M$ and $\gamma : M \to L$ be two monotone functions such that (L, α, γ, M) forms a *Galois insertion*, *i.e.*, a Galois connection where α is surjective:

$$\textbf{GI} \quad \lambda l.l \sqsubseteq \gamma \circ \alpha \quad \text{and} \quad \lambda m'.m' = \alpha \circ \gamma$$

Then the flow of information permitted by α and γ is guaranteed to be secure. However, Galois insertions mandate conditions on the definitions of functions α and γ that are much too strong, *i.e.*,

- $\gamma : M \to L$ is *injective*, *i.e.*, $\forall m_1, m_2 \in M : \gamma(m_1) = \gamma(m_2) \implies m_1 = m_2$
- $\alpha : L \to M$ is *surjective*, *i.e.*, $\forall m_1 \in M, \exists l_1 \in L : \alpha(l_1) = m_1$.

Typically data are shared only from a few security classes of any organisation. Organisations rarely make public their entire security class structure and permitted flow policies. Organisations also typically do not want any external influences on some subsets of its security classes. Thus, if not all elements of M are transfer classes, it may be impossible to define a Galois insertion (L, α, γ, M) because we cannot force α to be surjective.

Lagois Connections. Further, the connection we seek to make between two domains should allow us to transpose them. Fortunately there is an elegant structure, *i.e.*, *Lagois Connections* [16], which exactly satisfies this as well as the requirements of security and bidirectional sharing (**SC1, SC2, PC1, PC2, CC1** and **CC2**). They also conveniently generalise Galois insertions.

Definition 1 (Lagois Connection [16]). *If $L = (L, \sqsubseteq)$ and $M = (M, \sqsubseteq')$ are two partially ordered sets, and $\alpha : L \to M$ and $\gamma : M \to L$ are order-preserving functions, then we call the quadruple (L, α, γ, M) an* increasing *Lagois connection, if it satisfies the following properties:*

$$\textbf{\textit{LC1}} \quad \lambda l.l \sqsubseteq \gamma \circ \alpha \qquad \textbf{\textit{LC2}} \quad \lambda m'.m' \sqsubseteq' \alpha \circ \gamma$$
$$\textbf{\textit{LC3}} \quad \alpha \circ \gamma \circ \alpha = \alpha \qquad \textbf{\textit{LC4}} \quad \gamma \circ \alpha \circ \gamma = \gamma$$

LC3 ensures that $\gamma(\alpha(c_1))$ is the least upper bound of all security classes in C that are mapped to the same security class, say $u_1 = \alpha(c_1)$ in U.

The main result of this section is that if the negotiated monotone functions α and γ form a Lagois connection between the security lattices L and M, then information flows permitted are secure and precise.

Theorem 1. *Let L and M be two complete security class lattices, $\alpha : L \to M$ and $\gamma : M \to L$ be two monotone functions. Then the flow of information permitted by α, γ satisfies conditions **SC1**, **SC2**, **PC1**, **PC2**, **CC1** and **CC2** if (L, α, γ, M) is an increasing Lagois connection.*

Proof. Condition **SC1** holds because if $\alpha(l_1) \sqsubseteq' m_2$, by monotonicity of γ, $\gamma(\alpha(l_1)) \sqsubseteq \gamma(m_2)$. But by **LC1**, $l_1 \sqsubseteq \gamma(\alpha(l_1))$. So $l_1 \sqsubseteq \gamma(m_2)$. (A symmetric argument holds for **SC2**.) Conditions **PC1** and **PC2** are shown in Proposition 3.7 of [16]. Conditions **CC1** and **CC2** hold since the compositions $\gamma \circ \alpha$ and $\alpha \circ \gamma$ are *closure* operators, *i.e.*, idempotent, extensive, order-preserving endo-functions on L and M.

In fact, Lagois connections ensure that information in a security class in the original domain remains accessible even after doing a round-trip from the other domain (Proposition 3.8 in [16]):

$$\gamma(\alpha(l)) = \sqcap\{l^* \in \gamma[M] \mid l \sqsubseteq l^*\}, \tag{1}$$
$$\alpha(\gamma(m)) = \sqcap\{m^* \in \alpha[L] \mid m \sqsubseteq' m^*\}. \tag{2}$$

Properties of Lagois Connections We list some properties of Lagois connections that assist in the construction of a secure connection, and in identifying those security classes that play an important role in the connection. Proposition 2 says that the two functions γ and α uniquely determine each other.

Proposition 2 (Proposition 3.9 in [16])**.** *If (L, α, γ, M) is a Lagois connection, then the functions α and γ uniquely determine each other; in fact*

$$\gamma(m) = \bigsqcup \alpha^{-1}\ [\ \sqcap\{\ m^* \in \alpha[L] \mid m \sqsubseteq' m^*\}\ ] \tag{3}$$
$$\alpha(l) = \bigsqcup \gamma^{-1}[\ \sqcap\{\ l^* \in \gamma\ [\ M] \mid l \sqsubseteq l^*\}\ ] \tag{4}$$

Proposition 3 shows the existence of dominating members in their pre-images, which act as equivalence-class representatives of the equivalence relations $\sim_M$ and $\sim_L$ induced by the functions γ and α.

Proposition 3 (Proposition 3.7 in [16]). *Let* (L, α, γ, M) *be a Lagois connection and let* $m \in \alpha[L]$ *and* $l \in \gamma[M]$. *Then* $\alpha^{-1}(m)$ *has a largest member, which is* $\gamma(m)$, *and* $\gamma^{-1}(l)$ *has a largest member, which is* $\alpha(l)$.

That is, for all $m \in \alpha[L]$ and $l \in \gamma[M]$, $\gamma(m)$ and $\alpha(l)$ exist. Also, the images $\gamma[M]$ and $\alpha[L]$ are isomorphic lattices. $L^* = \gamma[\alpha[L]] = \gamma[M]$ and $M^* = \alpha[\gamma[M]] = \alpha[L]$ define a system of representatives for $\sim_L$ and $\sim_M$. Element $m^* = \alpha(\gamma(m'))$ in M^*, called a *budpoint*, acts as the representative of the equivalence class $[m']$ in the following sense:

$$\textit{if } m \in M \textit{ and } m^* \in M^* \textit{ with } m \sim_M m^* \textit{ then } m \sqsubseteq' m^* \tag{5}$$

Symmetrically, $L^* = \gamma[\alpha[L]] = \gamma[M]$ defines a system of representatives for $\sim_L$. These budpoints play a significant role in delineating the connection between the transfer classes in the two lattices.

Further, Proposition 4 shows that these budpoints are closed under meets. This property enables us to confine our analysis to just these classes when reasoning about bidirectional flows.

Proposition 4 (Proposition 3.11 in [16]). *If* (L, α, γ, M) *is a Lagois connection and* $A \subseteq \gamma[M]$, *then*

1. *the meet of* A *in* $\gamma[M]$ *exists if and only if the meet of* A *in* L *exists, and whenever either exists, they are equal.*
2. *the join* $\hat{a}$ *of* A *in* $\gamma[M]$ *exists if the join* $\check{a}$ *of* A *in* L *exists, and in this case* $\hat{a} = \gamma(\alpha(\check{a}))$

3 An Operational Model

3.1 Computational Model.

Let us consider two different organisations L and M that want to share data with each other. We start with the assumptions that the two domains comprise storage objects Z and Z' respectively, which are manipulated using their own sets of *atomic* transactional operations, ranged over by t and t' respectively. We further assume that these transactions within each domain are internally secure with respect to their flow models, and have no insecure or interfering interactions with the environment. Thus, we are agnostic to the level of abstraction of the systems we aim to connect securely, and since our approach treats the application domains as "black boxes", it is readily adaptable to any level of discourse (language, system, OS, database) found in the security literature.

We extend these operations with a minimal set of operations to transfer data between the two domains. To avoid any concurrency effects, interference or race conditions arising from inter-domain transfer, we augment the storage objects of both domains with a fresh set of *export* and *import* variables into/from which the data of the domain objects can be copied *atomically*. We designate these sets X, X' as the respective *export* variables, and Y, Y' as the respective

import variables, with the corresponding variable instances written as x_i, x'_i and y_i, y'_i. These export and import variables form mutually disjoint sets, and are distinct from any extant domain objects manipulated by the applications within a domain. These variables are used exclusively for transfer, and are manipulated atomically. We let w_i range over all variables in $N = Z \cup X \cup Y$ (respectively w'_i over $N' = Z' \cup X' \cup Y'$). Domain objects are copied *to export* variables and *from import* variables by special operations $rd(z, y)$ and $wr(x, z)$ (and $rd'(z', y')$ and $wr'(x', z')$ in the other domain). We assume *atomic transfer* operations (*trusted by both domains*) T_{RL}, T_{LR} that copy data from the export variables of one domain to the import variables of the other domain as the only mechanism for inter-domain flow of data. Let "phrase" p denote a command in either domain or a transfer operation, and let s be any (empty or non-empty) sequence of phrases.

$$\begin{array}{llll} \text{(command)} & c ::= t \mid rd(z,y) \mid wr(x,z) & c' ::= t' \mid rd'(z',y') \mid wr'(x',z') & \\ \text{(phrase)} & p ::= T_{RL}(x',y) \mid T_{LR}(x,y') \mid c \mid c' & \text{(seq)} & s ::= \epsilon \mid s_1; p \end{array}$$

$$\textsc{T}\ \frac{\mu \vdash t \Rightarrow \nu}{\langle \mu, \mu' \rangle \vdash t \Rightarrow \langle \nu, \mu' \rangle} \qquad \textsc{T'}\ \frac{\mu' \vdash t' \Rightarrow \nu'}{\langle \mu, \mu' \rangle \vdash t' \Rightarrow \langle \mu, \nu' \rangle}$$

$$\textsc{Wr}\ \frac{}{\langle \mu, \mu' \rangle \vdash wr(x,z) \Rightarrow \langle \mu[x := \mu(z)], \mu' \rangle} \qquad \textsc{Wr'}\ \frac{}{\langle \mu, \mu' \rangle \vdash wr'(x',z') \Rightarrow \langle \mu, \mu'[x' := \mu'(z')] \rangle}$$

$$\textsc{Rd}\ \frac{}{\langle \mu, \mu' \rangle \vdash rd(z,y) \Rightarrow \langle \mu[z := \mu(y)], \mu' \rangle} \qquad \textsc{Rd'}\ \frac{}{\langle \mu, \mu' \rangle \vdash rd'(z',y') \Rightarrow \langle \mu, \mu'[z' := \mu'(y')] \rangle}$$

$$\textsc{Trl}\ \frac{}{\langle \mu, \mu' \rangle \vdash T_{RL}(y,x') \Rightarrow \langle \mu[y := \mu'(x')], \mu' \rangle} \qquad \textsc{Tlr}\ \frac{}{\langle \mu, \mu' \rangle \vdash T_{LR}(y',x) \Rightarrow \langle \mu, \mu'[y' := \mu(x)] \rangle}$$

$$\textsc{Seq0}\ \frac{}{\langle \mu, \mu' \rangle \vdash \epsilon \Rightarrow^* \langle \mu, \mu' \rangle} \qquad \textsc{SeqS}\ \frac{\langle \mu, \mu' \rangle \vdash s_1 \Rightarrow^* \langle \mu_1, \mu'_1 \rangle, \quad \langle \mu_1, \mu'_1 \rangle \vdash p \Rightarrow \langle \mu_2, \mu'_2 \rangle}{\langle \mu, \mu' \rangle \vdash s_1; p \Rightarrow^* \langle \mu_2, \mu'_2 \rangle}$$

Fig. 8. Execution Rules

A *store* (typically μ, ν, μ', ν') is a finite-domain function from variables to a set of values (not further specified). We write, *e.g.*, $\mu(w)$ for the contents of the store μ at variable w, and $\mu[w := \mu'(w')]$ for the store that is the same as μ everywhere except at variable w, where it now takes value $\mu'(w')$.

The rules specifying execution of commands are given in Fig. 8. Assuming the specification of intradomain transactions (t, t') of the form $\mu \vdash t \implies \nu$ and $\mu' \vdash t' \implies \nu'$, our rules allow us to specify judgments of the form $\langle \mu, \mu' \rangle \vdash p \implies \langle \nu, \nu' \rangle$ for phrases, and the reflexive-transitive closure for sequences of

phrases. Note that phrase execution occurs *atomically*, and the intra-domain transactions, as well as copying to and from the export/import variables affect the store in only one domain, whereas the *atomic transfer* is only between export variables of one domain and the import variables of the other.

3.2 Typing Rules

Let the two domains have the respective different IFMs:

$$FM_L = \langle N, P, SC, \sqcup, \sqsubseteq \rangle \qquad FM_M = \langle N', P', SC', \sqcup, \sqsubseteq' \rangle,$$

such that the flow policies in both are defined over different sets of security classes SC and SC'.[3]

The (security) types of the core language are as follows. Metavariables l and m' range over the sets of security classes, SC and SC' respectively, which are partially ordered by $\sqsubseteq$ and $\sqsubseteq'$. A type assignment λ is a finite-domain function from variables N to SC (respectively, λ' from N' to SC'). The important restriction we place on λ and λ' is that they map export and import variables X, X, Y', Y only to points in the security lattices SC and SC' respectively which are in the domains of γ and α, *i.e.*, these points participate in the Lagois connection. Intuitively, a variable w mapped to security class l can store information of security class l or lower. The type system works with respect to a given type assignment. Given a security level, *e.g.*, l, the typing rules track for each command *within that domain* whether all written-to variables in that domain are of security classes "above" l, and additionally for transactions within a domain, they ensure "simple security", *i.e.*, that all variables which may have been read belong to security classes "below" l. We assume for the transactions within a domain, *e.g.*, L, we have a type system that will give us judgments of the form $\lambda \vdash c : l$. The novelty of our approach is to extend this framework to work over two connected domains, *i.e.*, given implicit security levels of the contexts in the respective domains. Cross-domain transfers will require pairing such judgments, and thus our type system will have judgments of the form

$$\langle \lambda, \lambda' \rangle \vdash p : \langle l, m' \rangle$$

We introduce a set of typing rules for the core language, given in Fig. 9. In many of the rules, the type for one of the domains is not constrained by the rule, and so any suitable type may be chosen as determined by the context, *e.g.*, m' in the rules TT, TRD, TWR and TT_{RL}, and both l and m' in COM0.

For transactions *e.g.*, t entirely within domain L, the typing rule TT constrains the type in the left domain to be at a level l that dominates all variables read in t, and which is dominated by all variables written to in t, but places no constraints on the type m' in the other domain M. In the rule TRD, since a value in import variable y is copied to the variable z, we have $\lambda(y) \sqsubseteq \lambda(z)$,

[3] Without loss of generality, we assume that $SC \cap SC' = \emptyset$, since we can suitably rename security classes.

$$\textsc{Tt}\ \frac{}{\langle\lambda,\lambda'\rangle \vdash t : \langle l, m'\rangle}\ \text{if for all } z \text{ assigned in } t,\ l \sqsubseteq \lambda(z) \ \&\ \text{for all } z_1 \text{ read in } t,\ \lambda(z_1) \sqsubseteq l$$

$$\textsc{Tt'}\ \frac{}{\langle\lambda,\lambda'\rangle \vdash t' : \langle l, m'\rangle}\ \text{if for all } z' \text{ assigned in } t',\ m' \sqsubseteq' \lambda'(z') \ \&\ \text{for all } z'_1 \text{ read in } t',\ \lambda'(z'_1) \sqsubseteq' m'$$

$$\textsc{Trd}\ \frac{\lambda(y) \sqsubseteq \lambda(z)}{\langle\lambda,\lambda'\rangle \vdash rd(z,y) : \langle\lambda(z), m'\rangle}$$

$$\textsc{Trd'}\ \frac{\lambda'(y') \sqsubseteq' \lambda'(z')}{\langle\lambda,\lambda'\rangle \vdash rd'(z',y') : \langle l, \lambda'(z')\rangle}$$

$$\textsc{Twr}\ \frac{\lambda(z) \sqsubseteq \lambda(x)}{\langle\lambda,\lambda'\rangle \vdash wr(x,z) : \langle\lambda(x), m'\rangle}$$

$$\textsc{Twr'}\ \frac{\lambda'(z') \sqsubseteq' \lambda'(x')}{\langle\lambda,\lambda'\rangle \vdash wr'(x',z') : \langle l, \lambda'(x')\rangle}$$

$$TT_{RL}\ \frac{\gamma(\lambda'(x')) \sqsubseteq \lambda(y)}{\langle\lambda,\lambda'\rangle \vdash T_{RL}(y,x') : \langle\lambda(y), \lambda'(x')\rangle}$$

$$TT_{LR}\text{'}\ \frac{\alpha(\lambda(x)) \sqsubseteq' \lambda'(y')}{\langle\lambda,\lambda'\rangle \vdash T_{LR}(y',x) : \langle\lambda(x), \lambda'(y')\rangle}$$

$$\textsc{Com0}\ \frac{}{\langle\lambda,\lambda'\rangle \vdash \epsilon : \langle l, m'\rangle}$$

$$\textsc{ComS}\ \frac{\langle\lambda,\lambda'\rangle \vdash p : \langle l_1, m'_1\rangle \qquad \langle\lambda,\lambda'\rangle \vdash s : \langle l, m'\rangle}{\langle\lambda,\lambda'\rangle \vdash s;p : \langle l_1 \sqcap l, m'_1 \sqcap m'\rangle}$$

Fig. 9. Typing rules

and the type in the domain L is $\lambda(z)$ with no constraint on the type m' in the other domain. Conversely, in the rule Twr, since a value in variable z is copied to the export variable x, we have $\lambda(z) \sqsubseteq \lambda(x)$, and the type in the domain L is $\lambda(x)$ with no constraint on the type m' in the other domain. In the rule TT_{RL}, since the contents of a variable x' in domain M are copied into a variable y in domain L, we require $\gamma(\lambda'(x')) \sqsubseteq \lambda(y)$, and constrain the type in domain L to $\lambda(y)$. The constraint in the other domain is unimportant (but for the sake of convenience, we peg it at $\lambda'(x')$). Finally, for the types of sequences of phrases, we take the meets of the collected types in each domain respectively, so that we can guarantee that no variable of type lower than these meets has been written into during the sequence. Note that Proposition 4 ensures that these types have the desired properties for participating in the Lagois connection.

3.3 Soundness

We now establish soundness of our scheme by showing a non-interference theorem with respect to operational semantics and the type system built on the security lattices. This theorem may be viewed as a conservative adaptation (to a minimal

secure data transfer framework in a Lagois-connected pair of domains) of the main result of Volpano *et al* [23].

We assume that underlying base transactional languages in each of the domains have the following simple property (stated for L, but an analogous property is assumed for M). Within each transaction t, for each assignment of an expression e to any variable z, the following holds: If μ, ν are two stores such that for all $w \in vars(e)$, we have $\mu(w) = \nu(w)$, then after executing the assignment, we will get $\mu(z) = \nu(z)$. That is, if two stores are equal for all variables appearing in the expression e, then the value assigned to the variable z will be the same. This assumption plays the rôle of "Simple Security" of expressions in [23] in the proof of the main theorem. The type system plays the rôle of "Confinement". We start with two obvious lemmas about the operational semantics, namely preservation of domains, and a "frame" lemma:

Lemma 5 (Domain preservation)**.** *If* $\langle \mu, \mu' \rangle \vdash s \Rightarrow^* \langle \mu_1, \mu'_1 \rangle$*, then* $dom(\mu) = dom(\mu_1)$*, and* $dom(\mu') = dom(\mu'_1)$*.*

Proof. By induction on the length of the derivation of $\langle \mu, \mu' \rangle \vdash s \Rightarrow^* \langle \mu_1, \mu'_1 \rangle$.

Lemma 6 (Frame)**.** *If* $\langle \mu, \mu' \rangle \vdash s \Rightarrow^* \langle \mu_1, \mu'_1 \rangle, w \in dom(\mu) \cup dom(\mu')$*, and* w *is not assigned to in* s*, then* $\mu(w) = \mu_1(w)$ *and* $\mu'(w) = \mu'_1(w)$*.*

Proof. By induction on the length of the derivation of $\langle \mu, \mu' \rangle \vdash s \Rightarrow^* \langle \mu_1, \mu'_1 \rangle$.

The main result of the paper assumes an "adversary" that operates at a security level l in domain L and at security level m' in domain M. Note however, that these two levels are interconnected by the monotone functions $\alpha : L \to M$ and $\gamma : M \to L$, since these levels are connected by the ability of information at one level in one domain to flow to the other level in the other domain. The following theorem says that if (a) a sequence of phrases is well-typed, and (b,c) we start its execution in two store configurations that are (e) indistinguishable with respect to all objects having security class below l and m' in the respective domains, then the corresponding resulting stores after execution continue to remain indistinguishable on all variables with security classes below these adversarial levels.

Theorem 7 (Type Soundness)**.** *Suppose* l, m' *are the "adversarial" type levels in the respective domains, which satisfy the condition* $l = \gamma(m')$ *and* $m' = \alpha(l)$*. Let*

(a) $\langle \lambda, \lambda' \rangle \vdash s : \langle l_0, m'_0 \rangle$*;* *(*$s$ *has security type* $\langle l_0, m'_0 \rangle$*)*
(b) $\langle \mu, \mu' \rangle \vdash s \Rightarrow^* \langle \mu_f, \mu'_f \rangle$*;* *(execution of* s *starting from* $\langle \mu, \mu' \rangle$*)*
(c) $\langle \nu, \nu' \rangle \vdash s \Rightarrow^* \langle \nu_f, \nu'_f \rangle$*;* *(execution of* s *starting from* $\langle \nu, \nu' \rangle$*)*
(d) $dom(\mu) = dom(\nu) = dom(\lambda)$ *and* $dom(\mu') = dom(\nu') = dom(\lambda')$*;*
(e) $\mu(w) = \nu(w)$ *for all* w *such that* $\lambda(w) \sqsubseteq l$*, and* $\mu'(w') = \nu'(w')$ *for all* w' *such that* $\lambda'(w') \sqsubseteq' m'$*.*

Then $\mu_f(w) = \nu_f(w)$ *for all* w *such that* $\lambda(w) \sqsubseteq l$*, and* $\mu'_f(w') = \nu'_f(w')$ *for all* w' *such that* $\lambda'(w') \sqsubseteq' m'$*.*

Proof. By induction on the length of sequence s. The base case is vacuously true. We now consider a sequence $s_1; p$. $\langle \mu, \mu' \rangle \vdash s_1 \Rightarrow^* \langle \mu_1, \mu_1' \rangle$ and $\langle \mu_1, \mu_1' \rangle \vdash p \Rightarrow \langle \mu_f, \mu_f' \rangle$ and $\langle \nu, \nu' \rangle \vdash s_1 \Rightarrow^* \langle \nu_1, \nu_1' \rangle$ and $\langle \nu_1, \nu_1' \rangle \vdash p \Rightarrow \langle \nu_f, \nu_f' \rangle$ By induction hypothesis applied to s_1, we have $\mu_1(w) = \nu_1(w)$ for all w such that $\lambda(w) \sqsubseteq l$, and $\mu_1'(w') = \nu_1'(w')$ for all w' such that $\lambda'(w') \sqsubseteq' m'$.

Let $\langle \lambda, \lambda' \rangle \vdash s_1 : \langle l_s, m_s' \rangle$, and $\langle \lambda, \lambda' \rangle \vdash p : \langle l_p, m_p' \rangle$. We examine four cases for p (the remaining cases are symmetrical).

Case p is t: Consider any w such that $\lambda(w) \sqsubseteq l$. If $w \in X \cup Y$ (*i.e.*, it doesn't appear in t), or if $w \in Z$ but is not assigned to in t, then by Lemma 6 and the induction hypothesis, $\mu_f(w) = \mu_1(w) = \nu_1(w) = \nu_f(w)$.
Now suppose z is assigned to in t. From the condition $\langle \lambda, \lambda' \rangle \vdash p : \langle l_p, m_p' \rangle$, we know that for all z_1 assigned in t, $l_p \sqsubseteq \lambda(z_1)$ and for all z_1 read in t, $\lambda(z_1) \sqsubseteq l_p$. Now if $l \sqsubseteq l_p$, then since in t no variables z_2 such that $\lambda(z_2) \sqsubseteq l$ are assigned to. Therefore by Lemma 6, $\mu_f(w) = \mu_1(w) = \nu_1(w) = \nu_f(w)$, for all w such that $\lambda(w) \sqsubseteq l$.
If $l_p \sqsubseteq l$, then for all z_1 read in t, $\lambda(z_1) \sqsubseteq l_p$. Therefore, by assumption on transaction t, if any variable z is assigned an expression e, since μ_1, ν_1 are two stores such that for all $z_1 \in Z_e = vars(e)$, $\mu_1(z_1) = \nu_1(z_1)$, the value of e will be the same. By this simple security argument, after the transaction t, we have $\mu_f(z) = \nu_f(z)$. Since the transaction happened entirely and atomically in domain L, we do not have to worry ourselves with changes in the other domain M, and do not need to concern ourselves with the adversarial level m'.

Case p is $rd(z, y)$: Thus $\langle \lambda, \lambda' \rangle \vdash rd(z, y) : \langle \lambda(z), m' \rangle$, which means $\lambda(y) \sqsubseteq \lambda(z)$. If $l \sqsubseteq \lambda(z)$, there is nothing to prove (Lemma 6, again). If $\lambda(z) \sqsubseteq l$, then since by I.H., $\mu_1(y) = \nu_1(y)$, we have $\mu_f(z) = \mu_1[z := \mu_1(y)](z) = \nu_1[z := \nu_1(y)](z) = \nu_f(z)$.

Case p is $wr(x, z)$: Thus $\langle \lambda, \lambda' \rangle \vdash wr(x, z) : \langle \lambda(x), m' \rangle$, which means $\lambda(z) \sqsubseteq \lambda(x)$. If $l \sqsubseteq \lambda(x)$, there is nothing to prove (Lemma 6, again). If $\lambda(x) \sqsubseteq l$, then since by I.H., $\mu_1(z) = \nu_1(z)$, we have $\mu_f(x) = \mu_1[x := \mu_1(z)](x) = \nu_1[x := \nu_1(z)](x) = \nu_f(x)$.

Case p is $T_{RL}(y, x')$: So $\langle \lambda, \lambda' \rangle \vdash T_{RL}(y, x') : \langle \lambda(y), \lambda'(x') \rangle$, and $\gamma(\lambda'(x')) \sqsubseteq \lambda(y)$. If $l \sqsubseteq \lambda(y)$, there is nothing to prove (Lemma 6, again). If $\lambda(y) \sqsubseteq l$, then by transitivity, $\gamma(\lambda'(x')) \sqsubseteq l$. By monotonicity of α: $\alpha(\gamma(\lambda'(x'))) \sqsubseteq' \alpha(l) = m'$ (By our assumption on l and m'). But by **LC2**, $\lambda'(x') \sqsubseteq' \alpha(\gamma(\lambda'(x')))$. So by transitivity, $\lambda'(x') \sqsubseteq' m'$. Now, by I.H., since $\mu_1'(x') = \nu_1'(x')$, we have $\mu_f(y) = \mu_1[y := \mu_1'(x')](y) = \nu_1[y := \nu_1'(x')](y) = \nu_f(y)$.

4 Related Work

The notion of Lagois connections [16] has surprisingly not been employed much in computer science. The only cited use of this idea seems to be the work of Huth [11] in establishing the correctness of programming language implementations. To our knowledge, our work is the only one to propose their use in secure information flow control.

Abstract Interpretation and type systems [5] have been used in secure flow analyses, *e.g.*, [3, 4] and [24], where security types are defined using Galois connections employing, for instance, a standard collecting semantics. Their use of two domains, concrete and abstract, with a Galois connection between them, for performing static analyses *within a single domain* should not be confused with our idea of secure connections between independently-defined security lattices of two organisations.

There has been substantial work on SIF in a distributed setting at the systems level. DStar [26] for example, uses sets of opaque identifiers to define security classes. The DStar framework extends a *particular* Decentralized Information Flow Control (DIFC) model [12, 25] for operating systems to a distributed network. The only partial order that is considered in DStar's security lattice is subset inclusion. So it is not clear if DStar can work on general IFC mechanisms such as FlowCaml [19], which can use any partial ordering. Nor can it express the labels of JiF [17] or Fabric [13] completely. DStar allows bidirectional communication between processes R and S only if $L_R \sqsubseteq_{O_R} L_S$ and $L_S \sqsubseteq_{O_S} L_R$, *i.e.*, if there is an order-isomorphism between the labels. Our motivating examples indicate such a requirement is far too restrictive for most practical arrangements for data sharing between organisations.

Fabric [13, 14] adds *trust relationships* directly derived from a principal hierarchy to support federated systems with mutually distrustful nodes and allows dynamic delegation of authority.

Most of the previous DIFC mechanisms [2,8,12,17,20,25] including Fabric are susceptible to the vulnerabilities illustrated in our motivating examples, which we will mention in the concluding discussion.

5 Conclusions and Future Work

Our work is similar in spirit to Denning's motivation for proposing lattices, namely to identify a simple and mathematically elegant structure in which to frame the construction of scalable secure information flow in a modular manner that preserved the autonomy of the individual organisations. From the basic requirements, we identified the elegant theory of Lagois connections as an appropriate structure. Lagois connections provide us a way to connect the security lattices of two (secure) systems in a manner that does not expose their entire internal structure and allows us to reason only in terms of the interfaced security classes. We believe that this framework is also applicable in more intricate information flow control formulations such as decentralised IFC [18] and models with declassification, as well as formulations with data-dependent security classes [15]. We intend to explore these aspects in the future.

In this paper, we also proposed a minimal operational model for the transfer of data between the two domains. This formulation is spare enough to be adaptable at various levels of abstraction (programming language, systems, databases), and is intended to illustrate that the Lagois connection framework can *conserve* security, using non-interference as the semantic notion of soundness. The choice

of non-interference and the use of a type system in the manner of Volpano *et al.* [23] was to illustrate in familiar terms how those techniques (removed from a particular language formulation) could be readily adapted to work in the context of secure connections between lattices. In this exercise, we made suitable assumptions of atomicity and the use of fresh variables for communication, so as to avoid usual sources of interference. By assuming that the basic intra-domain transactions are atomic and by not permitting conditional transfer of information across domains in the language, we have avoided dealing with issues related to implicit flows. We believe that the Lagois connection framework for secure flows between systems is readily adaptable for notions of semantic correctness other than non-interference, though that is an exercise for the future.

In the future we intend to explore how the theory of Lagois connections constitutes a robust framework that can support the discovery, decomposition, update and maintenance of secure MoUs for exchanging information. In this paper, we concerned ourselves only with two domains and bidirectional information exchange. Compositionality of Lagois connections allows these results to extend to chaining connections across several domains. In the future, we also intend to explore how one may secure more complicated information exchange arrangements than merely chains of bidirectional flow.

We close this discussion with a reminder of why it is important to have a framework in which secure flows should be treated in a modular and autonomous manner. Consider Myer's DIFC model described in [18], where a principal can delegate to others the capacity to act on its behalf. We believe that this notion does not scale well to large, networked systems since a principal may repose different levels of trust in the various hosts in the network. For this reason, we believe that frameworks such as Fabric [13, 14] may provide more power than mandated by a principle of least privilege. In general, since a principal rarely vests unqualified trust in another in all contexts and situations, one should confine the influence of the principals possessing delegated authority to only specific domains. A mathematical framework that can deal with localising trust and delegation of authority in different domains and controlling the manner in which information flow can be secured deserves a deeper study. We believe that algebraic theories such as Lagois connections can provide the necessary structure for articulating these concepts.

Acknowledgments. The second author thanks Deepak Garg for insightful discussions on secure information flow. Part of the title is stolen from E.M. Forster.

References

1. Boudol, G.: Secure Information Flow as a Safety Property. In: Formal Aspects in Security and Trust, 5th International Workshop, FAST 2008, Malaga, Spain, October 9-10, 2008, Revised Selected Papers. pp. 20–34 (2008)
2. Cheng, W., Ports, D.R.K., Schultz, D.A., Popic, V., Blankstein, A., Cowling, J.A., Curtis, D., Shrira, L., Liskov, B.: Abstractions for Usable Information Flow Control

in Aeolus. In: 2012 USENIX Annual Technical Conference, Boston, MA, USA, June 13-15, 2012. pp. 139–151 (2012)
3. Cortesi, A., Ferrara, P., Halder, R., Zanioli, M.: Combining symbolic and numerical domains for information leakage analysis. Trans. Computational Science **31**, 98–135 (2018)
4. Cortesi, A., Ferrara, P., Pistoia, M., Tripp, O.: Datacentric Semantics for Verification of Privacy Policy Compliance by Mobile Applications. In: Verification, Model Checking, and Abstract Interpretation - 16th International Conference, VMCAI 2015, Mumbai, India, January 12-14, 2015. Proceedings. pp. 61–79 (2015)
5. Cousot, P.: Types as Abstract Interpretations. In: Conference Record of POPL'97: The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Papers Presented at the Symposium, Paris, France, 15-17 January 1997. pp. 316–331 (1997)
6. Deng, S., Gümüsoglu, D., Xiong, W., Gener, Y.S., Demir, O., Szefer, J.: SecChisel: Language and Tool for Practical and Scalable Security Verification of Security-Aware Hardware Architectures. IACR Cryptology ePrint Archive **2017**, 193 (2017), http://eprint.iacr.org/2017/193
7. Denning, D.E.: A Lattice Model of Secure Information Flow. Commun. ACM **19**(5), 236–243 (1976)
8. Efstathopoulos, P., Krohn, M.N., Vandebogart, S., Frey, C., Ziegler, D., Kohler, E., Mazières, D., Kaashoek, M.F., Morris, R.T.: Labels and event processes in the Asbestos operating system. In: Proceedings of the 20th ACM Symposium on Operating Systems Principles 2005, SOSP 2005, Brighton, UK, October 23-26, 2005. pp. 17–30 (2005)
9. Ferraiuolo, A., Zhao, M., Myers, A.C., Suh, G.E.: Hyperflow: A Processor Architecture for Nonmalleable, Timing-Safe Information Flow Security. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. pp. 1583–1600 (2018)
10. Goguen, J.A., Meseguer, J.: Security Policies and Security Models. In: 1982 IEEE Symposium on Security and Privacy, Oakland, CA, USA, April 26-28, 1982. pp. 11–20 (1982)
11. Huth, M.: On the equivalence of state-transition systems. In: Theory and Formal Methods 1993, pp. 171–182. Springer (1993)
12. Krohn, M.N., Yip, A., Brodsky, M.Z., Cliffer, N., Kaashoek, M.F., Kohler, E., Morris, R.T.: Information flow control for standard OS abstractions. In: Proceedings of the 21st ACM Symposium on Operating Systems Principles 2007, SOSP 2007, Stevenson, Washington, USA, October 14-17, 2007. pp. 321–334 (2007)
13. Liu, J., Arden, O., George, M.D., Myers, A.C.: Fabric: Building open distributed systems securely by construction. Journal of Computer Security **25**(4-5), 367–426 (2017)
14. Liu, J., George, M.D., Vikram, K., Qi, X., Waye, L., Myers, A.C.: Fabric: a platform for secure distributed computation and storage. In: Proceedings of the 22nd ACM Symposium on Operating Systems Principles 2009, SOSP 2009, Big Sky, Montana, USA, October 11-14, 2009. pp. 321–334 (2009)
15. Lourenço, L., Caires, L.: Dependent information flow types. In: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015. pp. 317–328 (2015)
16. Melton, A., Schröder, B.S.W., Strecker, G.E.: Lagois Connections - a Counterpart to Galois Connections. Theor. Comput. Sci. **136**(1), 79–107 (1994)

17. Myers, A.C.: Jflow: Practical mostly-static information flow control. In: POPL '99, Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, TX, USA, January 20-22, 1999. pp. 228–241 (1999)
18. Myers, A.C.: Mostly-static decentralized information flow control. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA (1999), http://hdl.handle.net/1721.1/16717
19. Pottier, F., Simonet, V.: Information Flow Inference for ML. ACM Trans. Program. Lang. Syst. **25**(1), 117–158 (Jan 2003)
20. Roy, I., Porter, D.E., Bond, M.D., McKinley, K.S., Witchel, E.: Laminar: practical fine-grained decentralized information flow control. In: Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2009, Dublin, Ireland, June 15-21, 2009. pp. 63–74 (2009)
21. Sabelfeld, A., Myers, A.C.: Language-based information-flow security. IEEE Journal on Selected Areas in Communications **21**(1), 5–19 (2003)
22. Schultz, D.A., Liskov, B.: IFDB: Decentralized Information Flow Control for Databases. In: Eighth Eurosys Conference 2013, EuroSys '13, Prague, Czech Republic, April 14-17, 2013. pp. 43–56 (2013)
23. Volpano, D.M., Irvine, C.E., Smith, G.: A Sound Type System for Secure Flow Analysis. Journal of Computer Security **4**(2/3), 167–188 (1996)
24. Zanotti, M.: Security Typings by Abstract Interpretation. In: Static Analysis, 9th International Symposium, SAS 2002, Madrid, Spain, September 17-20, 2002, Proceedings. pp. 360–375 (2002)
25. Zeldovich, N., Boyd-Wickizer, S., Kohler, E., Mazières, D.: Making Information Flow Explicit in Histar. In: 7th Symposium on Operating Systems Design and Implementation (OSDI'06), November 6-8, Seattle, WA, USA. pp. 263–278 (2006)
26. Zeldovich, N., Boyd-Wickizer, S., Mazières, D.: Securing Distributed Systems with Information Flow Control. In: 5th USENIX Symposium on Networked Systems Design & Implementation, NSDI 2008, April 16-18, 2008, San Francisco, CA, USA, Proceedings. pp. 293–308 (2008)
27. Zhang, D., Wang, Y., Suh, G.E., Myers, A.C.: A Hardware Design Language for Timing-Sensitive Information-Flow Security. In: Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '15, Istanbul, Turkey, March 14-18, 2015. pp. 503–516 (2015)