Federated Authorization for Managed Data Sharing: Experiences from the ImPACT Project

Jeffrey S. Chase
Computer Science
Duke University
Durham, NC, USA
chase@cs.duke.edu

Ilya Baldin
RENCI/UNC Chapel Hill
Chapel Hill, NC, USA
ibaldin@renci.org

***Abstract*—This paper presents the rationale and design of the trust plane for ImPACT, a federated platform for managed sharing of restricted data. Key elements of the architecture include Web-based notaries for credential establishment based on declarative templates for Data Usage Agreements, a federated authorization pipeline, integration of popular services for identity management, and programmable policy based on a logical trust model with a repository of linked certificates. We show how these elements of the trust plane work in concert, and set the ideas in context with principles of federated authorization. A focus and contribution of the paper is to explore limitations of the resulting architecture and tensions among competing design goals. We also point the way toward future extensions, including policy-checked data access from cloud-hosted data enclaves with enhanced defenses against data leakage and exfiltration.**

***Index Terms*—Privacy-restricted data, Data Use Agreement, Authorization Logic**

I. Introduction

ImPACT [1] is a federated platform enabling networked research collaborations to discover and share restricted datasets in a controlled way. Its purpose is to facilitate safe sharing of sensitive data for approved research purposes. Sharing helps to obtain scientific value from data, but increases the risk of leaks or exposure of sensitive data to unauthorized parties.

This paper focuses on the problem of managing authorization and trust under multi-institutional sharing scenarios involving multiple parties operating under various interests and agreements. Data is produced and consumed by researchers affiliated with institutions and research projects. Projects may involve researchers at multiple institutions, governed by collaborative or virtual organizations (CO or VO) with their own authority structure. Data may be subject to usage conditions and constraints imposed by law or proprietary concerns.

The ImPACT project addresses how to enable sharing where sufficient trust exists under terms set by the data owners in their policies for access and usage. ImPACT seeks to enable data owners to control which parties and facilities are authorized to participate, and generally to maintain control over their data and restrict its distribution and use as they see fit. Our approach is based on strong authorization using declarative trust metadata describing access policies, approval workflows, usage conditions, user identities, research project affiliations, and/or security properties of the infrastructure used to process the data.

This material is based upon work supported by the National Science Foundation under grants OAC-1659367 and CNS-1330659.

We define a system as *federated* when multiple services operated by different principals contribute to an authorization decision. Data access in ImPACT involves multiple server instances to discover the data, establish credentials, and retrieve the data. We refer to the set of servers involved as the *authorization pipeline* for a request. The ImPACT pipeline introduces *Notary Services* that interpret conditions for *Data Usage Agreements* (DUAs), collect approvals from Web users, and issue digitally signed attestations to witness those approvals. It also leverages popular services for federated identity management to make it simpler to deploy, use, and manage.

The idea of a federated authorization pipeline is common to other complex federated services, including network testbeds such as NSF GENI [2] and its successors. Our approach to ImPACT is derived in part from our experience with GENI. ImPACT addresses representational challenges through the use of **declarative formalisms** including **logical trust**. It employs a combination of approaches to collect and transport authorization metadata through the pipeline to the decision point, involving design choices to support both web-based and "hands-free" access from hosted software tools.

We also explore three key challenges drawing on experience and limitations from the ImPACT prototype and pilot:

1) Obtaining authoritative **user attributes** for rich authorization. ImPACT leverages CILogon [3] as a key enabler to supply trust metadata. However, the cost is that it introduces a "trust bottleneck"—a central point of trust or attack, standing in the path of all access control and our goal of decentralized end-to-end trust.
2) Balancing **modular composition** of the authorization pipeline with ease of use. ImPACT's trust plane is sufficiently powerful to capture policy-based eligibility for the service instances that establish credentials, enabling flexible deployments. The prototype allows Web users to "click through" to gain data access, but that solution relies on a manual and static pipeline configuration.
3) Supporting data access via **trusted computing enclaves**. This support is complicated by reliance on Web standards: while they simplify the steps to traverse the

pipeline and propagate trust metadata, they presume that requests originate from a user agent running under the user's control and with its identity. In cloud settings, data access is mediated by servers that access data on the user's behalf.

§II gives an overview of authorization and trust management in ImPACT. §III explores the elements of the architecture in more detail, relating it to core concepts of federated authorization. §IV focuses on the role of federated identity and CILogon. §V outlines limitations of the prototype, challenges for realizing the platform's vision in extended scenarios of future interest, and our approach to address those challenges. As a motivating example it focuses on data access from cloud-hosted data enclaves with enhanced defenses against data leakage and exfiltration. §VI summarizes other related work, and §VII concludes.

II. ImPACT Overview

The ImPACT trust architecture is based on the principles of decentralized identity and attestation, policy autonomy, and point-of-use enforcement. It separates functions and concerns into separate component types (software agents), which various parties may deploy locally and link together by mutual consent. These principles make it suitable for a wide range of deployment scenarios.

Figure 1 depicts the core ImPACT components relating to authorization: a catalog for data discovery, notaries to establish credentials, and guarded storage. There may be many instances of each service, operated by different principals. Each step to obtain access may involve interactions with other parties via those services. A key challenge is to integrate all of the trust metadata into a policy compliance check (guard) at the point of access—a storage server at the end of the authorization pipeline.

Data discovery: Dataverse. To enable data discovery, ImPACT integrates with Dataverse [4], a federated data repository system widely used in social sciences and many other domains. A key step was to decouple the repository function from the Dataverse discovery service. We worked with the Dataverse team to extend Dataverse to interoperate with external data repositories that control their own data access. The data owner uses a tool to select a set of meta-data attributes to export to Dataverse ingest functions. The dataset itself remains in place and is served from owner-approved storage.

Presidio storage service. Decoupling storage provision from Dataverse preserves owner autonomy and enables use of Dataverse with a decentralized, distributed storage plane that can evolve independently for speed and scale. As a building block for such services, we developed Presidio, a simple Flask Web service that exposes an API to list and download datasets. What distinguishes Presidio from any Web file server is its integration at the tail of the authorization pipeline. Before allowing access, it invokes a *guard* to check for compliance with an access policy and decide whether or not to permit the access. The policy is a declarative document specified by the dataset owner: Presidio's guard imports and interprets the policy, assembles relevant trust data, and applies the policy.

Notary service. A primary goal of ImPACT is to automate various elements of policy-based data access including DUAs, which are often managed manually today. We designed a graphical workflow model to capture semantic aspects of real-world DUAs, based in part on studies of written DUA clauses [5], [6]. A DUA workflow template is a property graph whose nodes encode the phases and facets of a DUA, including agreement tasks for users in various roles, and precedence dependencies among tasks.

A DUA graph is represented declaratively as a GraphML document produced by an authoring tool. A notary interprets these documents and instantiates the template as needed to establish credentials for user access for specific projects.

Users interact with the notary through its Web UI within authenticated web sessions. The notary presents views of each DUA instance to associated users, allowing them to accept and/or certify various conditions required in a DUA. The DUA tasks may include approvals from other parties, such as project authorities, institutional governance, or a delegate of the data owner. To this end, the graph nodes are tagged with attributes that match against the roles or other identity attributes of users who must attest or certify each described task. As eligible users with matching attributes view and interact with a workflow instance, the notary traverses the workflow to dispatch tasks to those users, collecting their assertions for the DUA and updating the workflow instance state. The notary acts as a digital witness by issuing attestations that the required approval tasks are complete.

SAFE logical trust. To integrate the elements of the authorization pipeline, ImPACT employs a logical trust fabric based on SAFE [7]. SAFE defines a simple logic language (trust Datalog, §VI) to express assertions of fact and policy rules. Components manage trust by exchanging these statements and checking for compliance with applicable policy in guards, like the Presidio guard. SAFE includes an off-the-shelf logic interpreter; a scripting engine for guards and scripts that issue logic; and primitives to issue, store, index, retrieve, assemble, and query logic content. ImPACT issuers export trust metadata as logic certificates and store them in a shared certificate store (§V-C). ImPACT illustrates various aspects of how to use SAFE trust logic, as discussed below.

Threat model. The role of ImPACT's authorization pipeline and trust plane is to defend against unauthorized data access. Its purpose is to allow for expressive policy that governs access and use, and qualifies the component instances eligible for roles in the pipeline under the policy. The trust plane validates that each restricted dataset traverses only authorized users and components, including secure processing infrastructure (e.g., data enclaves, §V). If an authorized entity possesses restricted data, the system offers no defense if that entity violates its trust and leaks the data. ImPACT certifies all policy decisions and may provide some accountability for policy violations. Acccountability for data leakage is out of scope.

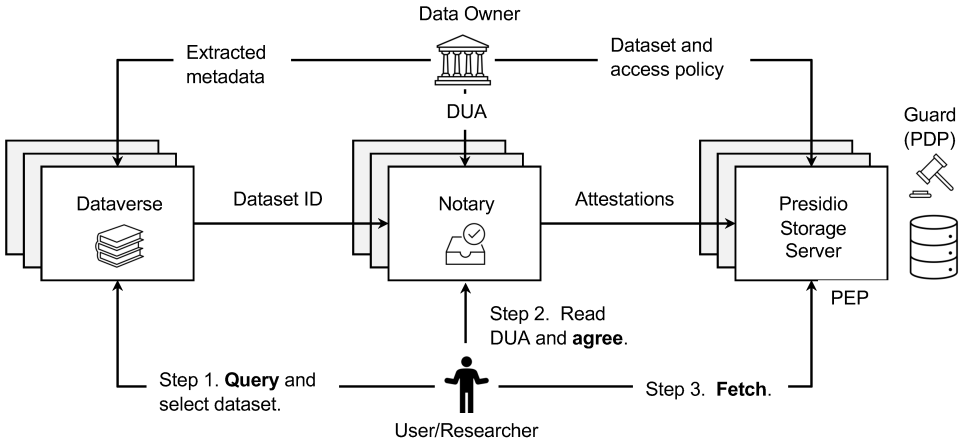

Fig. 1. Overview of the ImPACT authorization pipeline. A Data Owner exports a dataset together with metadata for a catalog, Data Usage Agreement (DUA) workflows, and access policies. A User/Researcher selects a dataset of interest from the catalog, interacts with one or more notaries to establish credentials, and retrieves the dataset from a guarded storage server, which applies the policy to check compliance.

III. Elements of the ImPACT Trust Plane

This section summarizes important concepts in authorization, and how they manifest in the design of ImPACT as an illustrative federated trust plane. We can understand a trust plane as a set of software processes, each acting on behalf of some principal, which exchange trust metadata over a network and make their local trust decisions based on that metadata.

In contrast to cloud authorization services such as Google's Zanzibar [8], a federated system has no central provider that is globally trusted. Instead, trust is decentralized: each participating process holds a cryptographic keypair to sign its statements and issue them as certificates to share with other parties. The recipients determine what trust to place in each issuer keypair according to local policy, based in part on what other principals say about it.

For this purpose ImPACT employs trust logic, a language to express security metadata and rules to evaluate trust. In the SAFE logical trust platform all trust metadata comprises sets of logic statements representing assertions of fact or policy. Principals may name one another in their statements. If the name is a public key hash, then a recipient can recognize any certificate signed under the key as issued by that named principal. Listings 1 and 2 illustrate with an example below.

In this way, trust logic subsumes the need for an external PKI: the logic captures and generalizes the relationships that a PKI might certify. It also encompasses within it all of the challenges and difficulties faced by any PKI, such as revoking and rotating keypairs.

ImPACT uses SAFE in concert with common standards and tools for federated Web identity, which employ other standards for digitally signed assertions. §IV discusses these systems and their role in user identity for ImPACT, and relates them to the concepts in this section.

A. Separating policy and mechanism

In policy-based authorization systems it is common to refer to a separation of Policy Enforcement Point (PEP) from a Policy Decision Point (PDP). A PEP is a software instance that accepts a request, initiates an access control decision, and services or rejects the request based on the result of the decision. In Figure 1 the PEP is a guarded file server, but it could be any application service. A PDP is a software instance that applies a policy to make an access control decision given parameters for a request, which include a requested action, a target object, and an identity of a requesting subject.

The separation of PEP from PDP embodies the concept of a reusable platform for authorization. It suggests that the PDP software to interpret and apply policies is written once and is interoperable with various PEP services. A PEP is a *relying party* (RP): it delegates or outsources its policy decisions to a PDP and is freed from the burden of managing authorization other than to gather relevant metadata and invoke the PDP. It also suggests that the PDP may interpret and apply many different policies expressed as documents in a declarative form. The PDP identifies the policy in force for a given decision and applies it to the request parameters and other authorization state, which may include assertions from other parties, as described below.

For example, XACML is a standard for authorization policy for a PDP. The PEP/PDP separation is fundamental to XACML, but it dates from earlier systems that manage policy for access to resources. It is common terminology for Internet (IETF) standards for policy-based networked systems [9], e.g., the IntServ framework for Internet Quality of Service [10].

We may understand the SAFE trust platform in these terms as a logical alternative to XACML. An application process (a PEP or RP) communicates with a trusted SAFE instance (also a process) to act as its PDP. The instance may run under the direct control of the RP (the application's operator), or as a common authorization service shared by multiple RPs. In the ImPACT scenario, the PEP/RP is a guarded file server, and the PDP is a guard script that runs within a SAFE instance. See Figure 2, discussed below.

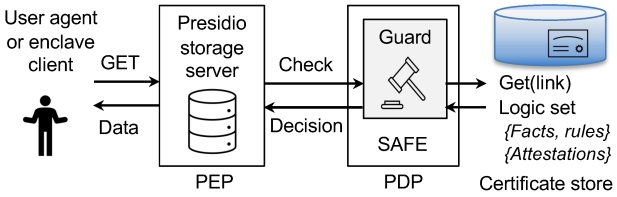

Fig. 2. The storage server applies access policy checks by invoking a guard script in a trusted SAFE logic engine, passing certain parameters (III-A). The guard retrieves policy rules, attestations, and other trust metadata as sets of signed logic statements, validates the signatures, assembles the logic, and runs programmed guard queries against it (§III-B). It retrieves the logic sets from a certificate store, indexed by links derived from the request parameters (§V-C).

B. Policy

SAFE is programmable: an instance loads scripts that define a set of entry points for requests from RPs, with various parameters. A *guard script* defines the PDP function: it assembles indexed logic statements into a context and queries it to apply a policy and return a decision. A SAFE instance may also run *issuer scripts* to issue templated logic as certificates signed under the issuer's keypair.

A key question is how the system represents and applies the access policies. We can view any policy as a set of rules: each rule specifies preconditions to match to apply the rule, taking a step toward a decision, e.g. to grant or deny access [9]. Related tooling includes software to author policies and to issue and assemble security assertions.

SAFE policy rules are expressed in logic. A policy may issue from a single principal, or it may freely compose logic issued by multiple sources. In ImPACT, this flexibility makes it possible for a Presidio server to serve files on behalf of a data owner or multiple owners, who may delegate elements of their policy to other parties. In essence, policy rules define a second level of programmability for SAFE: a guard's decision is guided by transportable policy rules imported from other parties, without changing the guard itself.

A logic rule enables an inference of a fact (the head) from a conjunction of known or inferred facts, which match preconditions to apply the rule (the *goals*). See *Rule A* in Listing 1, which we cite as a running example. An asserted or believed logical fact is a ground atom: a named predicate (a boolean function, also called a relation) applied to a list of constant parameters. For example, given the facts in Listing 2, Rule A infers `approve(alice)` meaning "Alice is approved". The goals of the rule may have variables, but they must match in a consistent way against the parameters of the matched facts, such that all goals are satisfied.

A guard script defines an authorization query that grants access if and only if one or more parameterized facts are true under the policy rules and other known facts in the context. For example, a guard might grant access for Alice if it infers `approve(Alice)` is true. SAFE's logic engine applies the rules recursively to generate proofs of compliance in a rigorous and provable way.

Listing 1. **Logic rule A.** An exemplary trust logic rule stating that a user principal U is "approved" if two preconditions (goals) match as true: a principal P asserts that U has attribute "qualified", and a designated *Root* principal grants "approval authority" to P. The issuer chooses the attribute vocabulary. The rule illustrates two ways to represent an attribute: as a special predicate (`approvalAuthority`) or as a parameter (`qualified`) of a more general predicate (`attr`). See Listing 2 for corresponding assertions.

```
Rule A.
approve(?U) :- ?P:attr(?U, qualified),
  $Root:approvalAuthority(?P).
```

Listing 2. **Logical assertions.** Exemplary logic statements $S1$ and $S2$ that assert attributes to delegate trust according to Rule A (Listing 1). Here `root`, `appService`, and `alice` stand for unique strings representing principals, e.g., a public key hash. The `root` says or speaks (issues and signs) $S1$, and `appService` says or speaks $S2$.

```
S1. root: approvalAuthority(appService).
S2. appService: attr(alice, qualified).
```

C. Attributes

Attributes are named properties of principals and objects. Attributes may have values. An example of an attribute is a distinguished name, whose value is a string that uniquely identifies the entity. Attributes may also represent properties such as roles or relations to other entities. A logic fact may assert that an entity possesses an attribute; see Listing 2.

Attributes may represent membership in a project or group. ImPACT makes use of an `isMemberOf` predicate, whose value is a group name to which a user belongs. An external group service manages cross-institutional groups (§IV). Groups are an important basis for authorization: for example, Zanzibar [8] models all access policy in terms of membership in nested groups.

An access policy often reduces to a boolean function over attributes of the subject and object of an action. This model is known as *attribute-based access control* (ABAC). The US government (NIST) promotes ABAC to enrich the practice of access control, as put forward in NIST 800-162 [11]. ABAC subsumes and extends role-based (RBAC) and identity-based models.

Trust logic expresses ABAC policies as sets of logical rules that match against conjunctions of attribute conditions, like Rule A in Listing 1. The goals of a rule may interrogate attributes of the subject or object of the access decision, and also related objects and principals, such as projects, groups, intermediary servers, and so on.

One practical challenge for ABAC systems is to capture the information that policies need into attributes, and establish authoritative sources for those attributes. ImPACT obtains all user identity attributes from standard identity solutions already in use by research institutions (§IV).

D. Authority

The concept of authority is fundamental to trust platforms. As a simple example, consider the problem of *policy authority*. The owner of a file in ImPACT has authority to issue the file's access policy. No other principal has this authority unless the owner delegates it. Before a guard applies a policy, it must

verify the policy's authority for the request. In one aspect it is an authentication problem: the issuer digitally signs the policy to enable a receiver to verify its origin and integrity. Separately, the guard must verify that this origin is in fact also the owner of the requested file, or a trusted delegate of the owner. SAFE incorporates this aspect into the logical check. That choice allows the owner to delegate the authority in various ways through its statements, and for the policy to draw from off-the-shelf rules issued by other sources that the owner trusts.

To authenticate statements to a file owner, a Presidio file server is configured with a self-certifying identifier (a *scid*) for each stored dataset, a set of one or more files. The scid is issued by the dataset's owner. A scid is a string that contains the issuer's public key hash as a prefix, and a unique identifier chosen by the issuer as its suffix. SAFE has a scripting primitive to generate scids with UUIDs for uniqueness. Given a dataset scid for the requested file as a parameter, the guard can verify cryptographically that an imported policy rule or other statement was spoken by the file's owner.

Another aspect of authority is *attribute authority*: who has the authority to assert attributes as input to the policy decision? XACML and NIST ABAC leave attribute authority loosely specified. A typical XACML PDP draws attributes from an enterprise directory service (e.g., LDAP), or receives them from the PEP/RP. Thus the XACML PEP or PDP grants attribute authority to a service that it trusts, independent of the policy.

SAFE takes the philosophy that a policy should specify its own trust in authorities. Delegations by a policy can validate the authority of notaries, workflow producers, or other intermediaries in our federated authorization pipeline. §III-E summarizes how trust logic represents delegations, and §III-F relates this idea to the structure of federations.

E. Delegation

Trust logic is a powerful formalism to specify delegation of authority [12]. A logical assertion can represent a qualified, limited delegation of authority to another principal P. For example, it can simply assert that P has some attribute that confers the authority, as by statements $S1$ and $S2$ in Listing 2. A delegation is constrained by the choice of predicate to represent it, and by its parameters.

Policy rules specify formally what authority is granted by a delegation by matching it as a condition for specific inferences. For example, under Rule A in Listing 1, the delegated `approvalAuthority` empowers P to assert the attribute `qualified` on another principal, but confers no other authority to P.

But how to validate a delegation? In a trust logic, each asserted or inferred fact is attributed to a *speaker*, a principal who either issued the assertion or the rule from which it was inferred. A SAFE instance validates the speaker against a signature before importing the logic, and tracks this attribution across inference. A rule that matches against an attribute as a condition may also match against the speaker of a fact used to satisfy the condition. Other conditions of the rule may query attributes of the speaker, which may derive from other rules and assertions. Rule A in Listing 1 illustrates this concept.

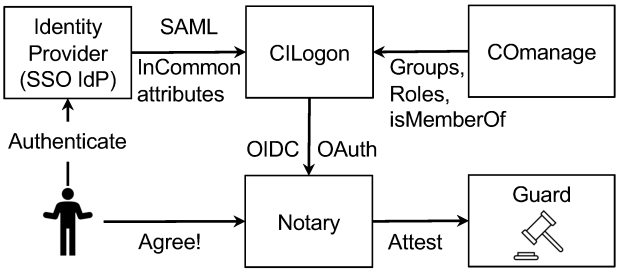

Fig. 3. The flow of identity attributes to an ImPACT storage guard—the access policy decision point. ImPACT relies on the CILogon service [3], [13] as an attribute authority that draws trust metadata from identity providers (IdPs) and from COmanage, a service to manage federated groups and roles (§IV-A, §IV-C). A notary obtains these attributes via Web single sign-on (SSO), and attests them under its keypair for use by the guard.

Logical validation checks are transitive. For example, the receiver of a delegation may issue a statement to delegate further to another principal, generating a chain. A guard validates such chains by applying policy rules recursively: the logic engine matches and applies rules automatically to search for a valid chain of inferences yielding a goal. In doing so, it applies the declarative rules to validate each step in the chain, possibly querying other attributes of each intermediary.

F. Federation

Delegated trust structures are ubiquitous in large-scale networked systems. In a federation, a common set of one or more trust anchors (*roots*) delegate authority to other servers for specific roles. Members delegate trust to the roots by accepting their authority, e.g., by issuing a logic fact designating the root as a trust anchor for the base case in recursive rules to infer authority of policies and attributes. Listing 1 illustrates how a policy may empower a designated root to delegate specific authority to other servers.

In more complex federations, delegates of the root may in turn delegate to others recursively, forming chains or trees. One familiar example is a PKI certificate chain, grounded in a root certifying authority, to validate a common TLS/SSL certificate. §IV discusses the trust structure of the federated identity services used in ImPACT, and §VI cites other examples of trust logic to represent secure Internet governance.

For ImPACT, logical policy enables data owners to subscribe to federations of eligible notaries and sources of authoritative attributes for projects, institutions, infrastructure, etc. Such structures may evolve and enter into use on a per-dataset basis over time without changing the software for the services or guards.

IV. Identity and Access Management

In addition to logical trust, the ImPACT trust plane incorporates other standards known broadly as Identity and Access Management (IAM). It relies on widely deployed

IAM systems to authenticate users and ingest their subject attributes. Figure 3 illustrates the flow of user attributes in the current prototype, which relies on the CILogon service [3], [13] (§IV-C). This section discusses that choice and the role of IAM systems in ImPACT.

Despite the value of trust logic to represent multi-domain trust in ImPACT, our design chooses to hide it from users. Instances of the core ImPACT services are SAFE principals, which authenticate their interactions with keypairs. Managing user identity is a different challenge: there are many more users, and they prefer to use familiar Web account logins rather than keypairs. It is costly to administer user accounts and track changes to their status, roles and affiliations.

Federated identity management systems support Web accounts with single sign-on (SSO). They reduce user password fatigue and free other services from the need to manage user accounts. Identity networks include the Shibboleth/Incommon federation of US research institutions, and OpenID Connect (OIDC) protocols for identity based on commercial social media platforms.

These systems also illustrate the idea of federated trust structures outlined in § III-F. They are based on open standards for digitally signed assertions: Security Assertion Markup Language (SAML; see [14]) and OIDC claims (see [15]). In contrast to trust logic, these standards represent assertions only, and not declarative policy: the code interprets the assertions according to a trust structure that is "baked in".

A. Identity and SSO

With single sign-on (SSO), users establish accounts with an institution or provider based on their affiliation as a customer, student, or employee. For example, US researchers are accustomed to using their institutional accounts to sign on to a range of services within and outside their institutions. The institution operates a trusted identity provider (IdP) service that maintains identity attributes for each user account,

When a user navigates to a Web-based application service (called a service provider or SP) in a secure session, the SP redirects the browser to the IdP, where the user logs in if needed. The IdP receives the identity of the referring SP from the browser, and releases user attributes, signed by the IdP, for the browser to return to the SP. The IdP determines what to release according to a privacy/release policy on a per-SP basis. An SP does not necessarily receive a distinguished name or other attribute sufficient to identify the user. In many cases, the affiliation and related attributes (e.g., student) may be sufficient for the SP's access control policy.

InCommon is a simple example of a federated trust structure that enables an SP to serve users from many participating institutions, using their home user accounts. A root principal (maintained by Internet2) endorses the public keys of member IdPs. Each SP knows the public key of the the root, and trusts the root to qualify and endorse legitimate IdPs. This structure eliminates the need for an IdP to establish trust with every SP independently: the federation scales without updating each SP with a new list of IdPs. Instead, the SP accepts attributes from any IdP that the root endorses as a member.

B. Service Providers

Trust structures may evolve over time as systems become more interconnected. Consider the approval of Service Providers (SPs) in the InCommon example. Due to privacy concerns with attribute release, it was initially necessary to register each SP with the operator of each IdP. This approach was workable for early use cases, when the goal was to enable SSO for SPs and users within an institution or enterprise: the SPs need register with only one IdP, and it is in the same administrative domain. As deployment expanded, interest in SSO for external services grew. That created new challenges for administrative scaling.

To illustrate, in the early days of the NSF GENI network testbed project the authors promoted the idea that GENI should use InCommon identity to free users and services from managing separate and duplicative accounts and credentials. That required registering GENI services as SPs with each participating institution. Other SPs also faced the challenge: IdP administrators are concerned about privacy leaks, but may have less incentive to understand the value that a candidate SP provides or to enable users to access the SP with institutional credentials. Additionally, this growth raised governance questions, given the lack of a standard to judge the stewardship and accountability of an external SP receiving identity information.

Ultimately, the solution was to extend InCommon to qualify and endorse SPs with value in the "research and scholarship category". IdP operators may choose to release attributes to an SP based on signed proof of endorsement by the InCommon root, even if the SP was not registered locally. The endorsements are represented in SAML security assertions: the root signs them and the SP passes them with the redirect through the browser to the IdP.

Services like ImPACT may seek approval in this category as they mature. However, the approval hurdle motivates a different approach: empowering users to select SPs and approve release of their own attributes, as described below.

C. CILogon

The ImPACT prototype relies on CILogon [13] as a bridge to InCommon and other identity networks. CILogon is a software platform for IAM for Web-based application services that support research collaborations. CILogon runs a service instance that is category-approved by InCommon for attribute release. Users may grant permission to re-release attributes to other user-approved services, such as an ImPACT notary, without involving the IdP. CILogon eliminates any need for ImPACT to maintain user accounts or for institutions to reconfigure their identity services to enable ImPACT. Users may authenticate to ImPACT Web services from any IdP that recognizes CILogon as an SP.

To use CILogon, logical policies in the ImPACT pilot delegate authority for user attributes to approved notaries. The implementation of the Notary Service collects user attributes

from CILogon using OIDC (see Figure 3). It issues logic to attest the attributes of users as they complete workflow tasks, enabling downstream policy checking. The downside of this approach is that all user attributes depend on trust in CILogon, and this trust is not explicit in the policy.

CILogon also provides supplementary user attributes to enrich access control, such as cross-institutional project groups. CILogon integrates with COmanage, a service that allows authenticated users to create collaborative organizations (COs) and grant membership to other users and groups with various roles. In this way, collaborating users can establish additional identity attributes (e.g., `isMemberOf`) to certify their collaborations. We also use COs to supply needed attributes not yet available from the IdPs. In particular, a notary must match users to DUA tasks that require approval from institutional governance, but governance roles are currently lacking from the standard attribute set for InCommon.

V. Extending Authorization

To summarize the architecture described above, ImPACT serves restricted datasets, discoverable through the Dataverse Project and its services to archive, share, and explore research datasets. Suppose that a researcher Alice establishes a Web session with Dataverse, queries its metadata catalog, and receives identifiers for one or more datasets of interest. Alice interacts with one or more notaries to complete DUA forms according to the data owner's policy and awaits any required approvals, e.g., from project authorities or institutional governance. Ultimately, software running on her behalf requests a file from a file server (a Presidio instance). The server's guard evaluates the policy against relevant credentials, including attestations from the notaries, and verifies compliance before returning the file.

With this background, we now return to the high-level vision of the ImPACT trust architecture and the role of the various elements in realizing that vision. A key goal of ImPACT is to lay the groundwork for rich authorization that is sufficiently powerful to meet future demands on our roadmap. The approach outlined above is based on three key elements: semantically expressive declarative representations, automated interpretation, and a modular and decentralized deployment.

Declarative representations. Data owners express access policy for their datasets as logic packages and DUA workflow graphs. The graphs display Web forms for DUA tasks organized into a workflow DAG to capture precedence and collect data populated into downstream forms. The logic language and policy rules can capture complex assertions extracted from DUAs as well as simple attributes.

Automated interpretation. These representations are transportable and machine-readable documents that a server interprets to execute the DUA or apply the policy. These properties make it possible to support a range of policies for different objects simultaneously, or change a policy without changing the software on the server. A notary can execute a wide range of DUAs, traversing the workflow to trigger user task prompts as dependencies are met. The logic engine combine policy rules and assertions from multiple parties; their union with a guard query forms a logic program that resolves using an off-the-shelf logic prover. Other tools may consume these representations to check various properties. For example, we built a tool to check soundness of DUA template graphs.

Modular authorization pipeline. Logical policy enables flexible deployment choices for the authorization pipeline because it can tag the intermediaries eligible to execute a given agreement and certify its completion. In principle, this property allows us to compose the authorization pipeline in a flexible way at deployment time or time of use. That goal is important because we envision many possible deployments of notaries in a data federation. For example, a notary may operate on behalf of a data owner, or a consortium of institutions or owners, or a VO with members in multiple institutions. The choice may be driven by a notary's need for specific attribute information: for example, an institutional notary might access an internal enterprise directory to extract needed governance roles (§IV-C). Complex scenarios may require multiple DUA workflows certified by different notaries.

We intend that these foundations will enable us to extend the ImPACT components in new directions to manage sharing of restricted data. For example, it is easy to express trust logic policies that consider extended attributes of principals or objects asserted by other eligible authorities in the trust structure—for example, attributes of the researcher's project or institution. We also consider scenarios that involve negotiated access via interacting DUA workflows in which, for example, a researcher proposes terms of use, and a delegate of the data owner approves or rejects them.

A related goal of ImPACT is to include infrastructure security under policy control, so that data owners may apply limits to modes of use. For example, data owners might mandate **cloud-mediated access**—processing on cloud providers with suitable defenses against leakage—and even check the software configurations used to access the data. We envision that data owners may associate the data with code modules that enable certain queries or analyses against the data, packaged for launch on policy-compliant cloud infrastructure. We now expand on our goals for the cloud-mediated access model (depicted in Figure 4), outline how it places new pressure on authorization beyond the Web-based access model of the initial prototype, and show how logical trust can support it.

A. Cloud-mediated access: Data enclaves

Once a file server approves a data access request and returns the data, it relinquishes any control over how the client uses the data. Security-enhanced cloud services offer potential to extend policy control end-to-end to assure the owner that the access complies with its terms. Our approach is to integrate data access with processing on secure cloud enclaves, and extend compliance checking to validate their configurations.

We use the term *enclave* to denote an infrastructure with storage and processing and a validated software stack with specific defenses against data leakage and exfiltration. A range of enclave architectures are possible with various levels of

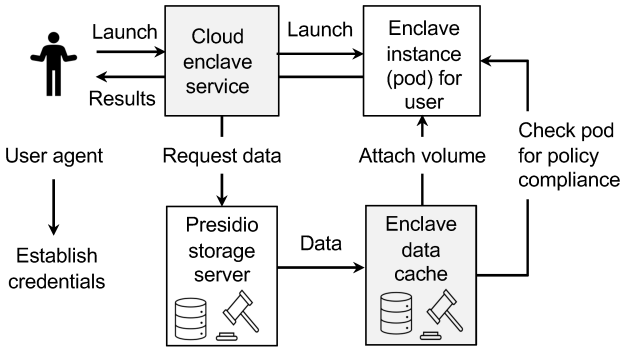

Fig. 4. Mediated data access from a secure cloud enclave. Here the policy may restrict the user to analyze the data from a compliant *enclave*: an instance with approved software hosted on trusted infrastructure with specific defenses against data leaks (§V-A). The user requests a campus cloud service (shaded regions) to launch an instance and obtain the data. The service is enhanced to obtain the data on the user's behalf and apply further policy checks against the instance before attaching the data (§V-D).

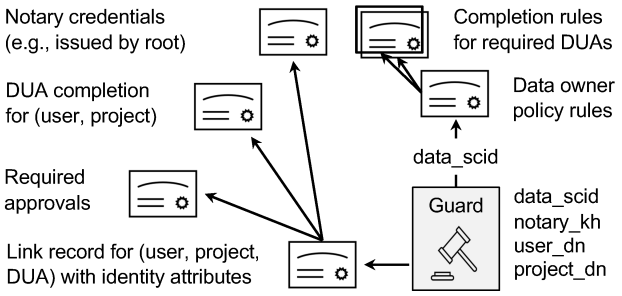

Fig. 5. Issuers of SAFE logic sets link their certificates to others with supporting evidence or authority. SAFE scripts may derive links from known templates and parameters. The ImPACT guard indexes relevant logic certificates from the request parameters and retrieves them from a store (§V-C).

protection. Hardware enclave abstractions continue to mature, e.g., with Intel Software Guard Extensions (SGX [16], [17]), and may enable trusted computing on sensitive data even with untrusted infrastructure providers (e.g., as in Ryoan [18]).

In ImPACT we focus on architectures for cloud enclave services operated by campuses or other infrastructure providers trusted by policy. We developed an exemplary data enclave solution deployed at Duke University, in which researchers access data only with approved tools through a remote desktop protocol with an authenticated InCommon identity. Such data enclaves enable institutions to control restricted data more closely and limit their exposure from data leaks. To address the descriptive challenge, ImPACT DUAs support attestations of infrastructure security properties.

However, the enclave model creates new challenges to track and maintain authorization state through the pipeline. The current ImPACT prototype lacks support for policy checking of enclaves. In particular, ImPACT is designed to simplify Web-based access (§V-B), but enclave access may occur "hands free"—outside of an authenticated Web session—and also requires validation of the enclave itself. For example, Figure 4 depicts a cloud enclave service with an API to launch an enclave instance from a configuration template—for example, a VM, container, or Kubernetes pod [19]. The storage guard is extended to allow an incoming request originating from an approved enclave service acting on the researcher's behalf. The service retrieves the data and attaches the local copy only if the template complies with policy (§V-D).

B. Web-based access in the ImPACT prototype

The current ImPACT prototype emphasizes simplicity for Web users, for which the user agent is a browser. It leverages single sign-on (§IV) to keep identity management familiar and largely transparent.

Additionally, the prototype preconfigures the authorization pipeline for Web-based access. When Alice selects a dataset of interest, Dataverse generates a URL to a notary selected by the data owner. After Alice completes any required DUA workflows with the notary—which may require attestations from other parties such as her institutional governance—the notary generates a URL to a directory for the dataset on a storage (Presidio) server preselected by the data owner. The notary response presents the link to Alice's user agent, which sends a well-formed request for the data to Presidio when Alice clicks on the link.

In this scenario, Alice merely clicks from one site to another until she can download the file. At each step, her browser authenticates her session with its common SSO identity. The notary also constructs a JSON Web Token containing certain parameters for the request. Importantly, the token is not a bearer token as defined by OAuth [20]: a bearer token confers the same privilege to any client who presents it. In this case, the token merely expedites transfer of parameters through the browser. These parameters include the user's distinguished name, which must match the authenticated client identity and attestations from the notary for the token to be of use.

While this approach is easy to use and so minimizes barriers to adoption by users, it raises a number of challenges. First, it does not adapt easily to scenarios in which a user (Alice) accesses the data from a hosted tool and/or through an enclave or other trusted service. In particular, the tool or enclave cannot authenticate with Alice's identity as required. It could if Alice obtains a keypair bound to her identity—CILogon can certify one—and shares a private key with the cloud instance. While such key sharing is common in hosting scenarios, it exposes her to leakage or misuse of the private key. [21]

Second, the current approach constrains each file for service by a specific notary and file server designated in a pre-established service chain. It relies on a manual configuration step for the data owner to designate the servers for each file: the data owner uses a Web browser to register the file with the notary, install an approval workflow, and designate the file server. In practice, it limits the system to a configuration in which the data owner itself or a known delegate operates the notary and the file server.

C. Linked logic certificates

SAFE's general approach to managing trust metadata can replace the need to rely on state transfer through Web tokens for ImPACT. SAFE uses certificate *linking* to track authorization state and discover and retrieve logic relevant to a given trust decision. Issuers export their logic in sets materialized as certificates. Each set is uniquely identified by an index derived from a hash of the issuer's public key and string parameters that the issuer selects. One set may link to another by including the target's index as an embedded link. Given a link, or the parameters to construct the link, a guard may import and cache the transitive closure of the indexed logic content.

The trust scripts developed for ImPACT make extensive use of certificate linking (Figure 5). Policy packages issued by the data owner link to compliance checking rules for each required workflow DUA. A notary issues attestations for required workflow elements, linked from a root receipt for the DUA keyed by the the researcher's distinguished name and affiliated project. The guard script assembles a query context by synthesizing links from the request parameters.

Issuers post their certificates to a shared repository based on a variant of a canonical key-value (put/get) storage abstraction. The architecture of the store is out of scope: in our pilot we use an enterprise key-value store (Riak) operated by a trusted party. If the store fails or is compromised, an attacker can mount a denial-of-service attack, but it cannot subvert the protection system because all certificates are signed.

SAFE's certificate store is suitable for decentralized operation with the trust and failure properties of a permissioned blockchain, but with a more scalable implementation. Specifically, it is designed to run with a Byzantine quorum system (BQS), following Phalanx [22]. BQS replication scales more easily than blockchains because it allows sharding, in which each put/get executes on only a subset of replicas. Blockchains induce consensus on a linear sequence of operations for state-machine replication, but it is expensive to obtain, and SAFE does not require it.

D. Authorization pipeline revisited

The handling of trust metadata through SAFE linking gives us a path to meet our goals for a modular pipeline and policy-checked access from enclaves. As a prerequisite, we must address common challenges for distributed systems: select notaries or other servers based on preferences, and pass needed string parameters through the APIs, e.g., to invoke the enclave. One challenge is that the client software must know how to traverse the pipeline. That requires a selection service to synthesize a Web page to click through, or a stateful user agent such as an app rather than a simple Web interface.

Beyond these requirements, it is straightforward to verify or extend the pipeline. The guard and selection software have all required parameters to retrieve logic content to validate candidate service instances against the policy. If the data request originates from an enclave, then the cloud service can request the data under its own identity (a keypair) on the user's behalf. It is straightforward to extend the guard to validate that the policy trusts the authenticated requester as an enclave. Presidio already supports authentication of a client by keypair using two-way TLS.

We envision access policies that limit processing to enclaves with specified security properties, or by qualified software stacks and configurations. To meet this goal, the authorization system must match logical (semantic) enclave descriptions against conditions in the policy. We are developing an enclave approach based on Kubernetes [19] that checks the software stack and configurations of cloud instances (pods). We use trust logic to endorse and qualify the pod configurations, following the model in TapCon [23] and in Nexus [24], which introduced trust logic for software attestation (see also [25]). A trusted enclave may apply extended checks against the data owner's policy locally to qualify the software stack and configuration, or it may pass a link to the attestation with the data access request to allow the Presidio guard to check it.

VI. Related Work

SAFE trust logic enables us to tie together ImPACT's elements of programmable authorization. Studies of authorization logic have yielded many approaches too numerous to detail here, including recent approaches that are expressive but also complex (e.g., NAL [26]). SAFE embraces direct use of Datalog [27], a rigorously defined and extensively studied general-purpose logic language that is a subset of Prolog, a popular language for logic programming with a standard syntax. Our approach merely adds a modal operator **says** to Datalog to identify or match the speaker of each statement or predicate, enabling its direct use as a logic of belief and attribution. This idea previously appears in Binder [28], SD3 [29], and SENDLOG [30]. Datalog-with-says is at least as powerful as the XACML web standard, and it enables reasoning from authenticated policy rules and assertions gathered from multiple sources, which is crucial in the federated scenarios characteristic of collaborative science.

Like these earlier systems, SAFE uses signed logic certificates as a transport for authenticated logic. To all of these systems, SAFE adds simple programmable indexing and sharing of logic certificates through a key-value store with scripted linking of related certificates via interpolated string templates. This combination facilitates rapid prototyping of trust applications like ImPACT.

Datalog trust logic is sufficiently powerful to capture trust structures of large distributed systems, and interpret the logic directly in an implementation. ImPACT illustrates this point, and this paper outlines how logic can represent the structure of IAM networks as well (§IV). We now cite further examples from network governance to complete the picture. SD3 [29] shows how to build secure Internet naming (DNSSEC) with logic. SENDlog [30] captures secure interdomain routing, and we use SAFE to extend that approach to construct secure interdomain networks on testbeds, with logical policy controls and IP prefix authentication [31]. Trust logic also captures the NSF GENI testbed security architecture [2], [7], [32] and its allocation of resources to hosted networks.

VII. Conclusion

This paper summarizes the design of the trust plane for ImPACT, a platform for managed sharing of restricted data. We outline the role and rationale for key elements of the architecture: notaries for credential establishment based on declarative templates for Data Usage Agreements, a federated authorization pipeline, integration of popular services for identity management, and programmable policy based on a logical trust model. We show how these elements of the trust plane work in concert, and set the ideas in context with principles of federated authorization. A focus and contribution of the paper is to explore limitations of the resulting architecture and tensions among competing design goals. We also point the way toward future extensions, including policy-checked data access from cloud-hosted data enclaves with enhanced defenses against data leakage and exfiltration.

Acknowledgments. Contributors to ImPACT include the co-authors of [1]: Jonathan Crabtree, Thomas Nechyba, Laura Christopherson, Michael Stealey, Charley Kneifel, Victor Orlikowski, Rob Carter, Erik Scott, Akio Sone, and Don Sizemore. Shujun Qi contributed ideas for enhanced cloud enclave services.

References

[1] I. Baldin, J. Chase, J. Crabtree, T. Nechyba, L. Christopherson, M. Stealey, C. Kneifel, V. Orlikowski, R. Carter, E. Scott, A. Sone, and D. Sizemore, "ImPACT: A networked service architecture for safe sharing of restricted data," *A journal submission*, 2021.

[2] M. Brinn, N. Bastin, A. Bavier, M. Berman, J. Chase, and R. Ricci, "Trust as the foundation of resource exchange in GENI," in *Proceedings of the 10th EAI International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom)*. European Alliance for Innovation, June 2015.

[3] "CI Logon website," https://www.cilogon.org/, 2019.

[4] M. Crosas, "The dataverse network®: an open-source application for sharing, discovering and preserving data," *D-lib Magazine*, vol. 17, no. 1, p. 2, 2011.

[5] S. Grabus and J. Greenberg, "Toward a metadata framework for sharing sensitive and closed data: an analysis of data sharing agreement attributes," in *Research Conference on Metadata and Semantics Research*. Springer, 2017, pp. 300–311.

[6] UMich ISR, " Data Use Agreement Portal," https://www.psc.isr.umich.edu/dis/data/restricted/rdc/, 2020.

[7] Q. Cao, V. Thummala, J. S. Chase, Y. Yao, and B. Xie, "Certificate linking and caching for logical trust," *arXiv:1701.06562*, 2017.

[8] R. Pang, R. Caceres, M. Burrows, Z. Chen, P. Dave, N. Germer, A. Golynski, K. Graney, N. Kang, L. Kissner, J. L. Korn, A. Parmar, C. D. Richards, and M. Wang, "Zanzibar: Google's consistent, global authorization system," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. Renton, WA: USENIX Association, Jul. 2019, pp. 33–46. [Online]. Available: https://www.usenix.org/conference/atc19/presentation/pang

[9] S. Waldbusser, J. Perry, S. Herzog, M. Carlson, J. Schnizlein, B. Quinn, M. Scherling, A. Westerinen, A. Huynh, and J. Strassner, "Terminology for policy-based management," Internet Requests for Comments, RFC Editor, RFC 3198, Nov 2001. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3198.txt

[10] D. Pendarakis, R. Yavatkar, and R. Guerin, "A framework for policy-based admission control," Internet Requests for Comments, RFC Editor, RFC 2753, Jan 2000. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2753.txt

[11] V. Hu, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations, Special Publication 800-162, National Institute of Standards and Technology (NIST)," http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf, 2014.

[12] N. Li, B. N. Grosof, and J. Feigenbaum, "Delegation logic: A logic-based approach to distributed authorization," *ACM Transactions on Information and System Security*, vol. 6, no. 1, pp. 128–171, Feb. 2003.

[13] J. Basney, H. Flanagan, T. Fleury, J. Gaynor, S. Koranda, and B. Oshrin, "CILogon: Enabling federated identity and access management for scientific collaborations," in *Proceedings of Science*, vol. 351, 2019, p. 031, appeared in International Symposium on Grids and Clouds (ISGC).

[14] J. Somorovsky, A. Mayer, J. Schwenk, M. Kampmann, and M. Jensen, "On breaking SAML: Be whoever you want to be," in *21st USENIX Security Symposium*, 2012, pp. 397–412.

[15] D. Fett, R. Küsters, and G. Schmitz, "The Web SSO Standard OpenID Connect: In-Depth Formal Security Analysis and Security Guidelines," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 2017, pp. 189–202.

[16] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for CPU-based attestation and sealing," in *Proceedings of the International Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*. ACM New York, NY, USA, 2013.

[17] V. Costan and S. Devadas, "Intel SGX explained," *IACR Cryptolology ePrint Archive*, vol. 2016, no. 86, pp. 1–118, 2016.

[18] T. Hunt, Z. Zhu, Y. Xu, S. Peter, and E. Witchel, "Ryoan: A distributed sandbox for untrusted computation on secret data," *ACM Transactions on Computer Systems (TOCS)*, vol. 35, no. 4, pp. 1–32, 2018.

[19] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes: Lessons learned from three container management systems over a decade," *ACM Queue*, vol. 14, no. 1, pp. 70–93, 2016.

[20] M. Jones and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage," Internet Requests for Comments, RFC Editor, RFC 6750, Oct 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6750.txt

[21] F. Cangialosi, T. Chung, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "Measurement and analysis of private key sharing in the HTTPS ecosystem," in *ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 628–640.

[22] D. Malkhi and M. K. Reiter, "Secure and Scalable Replication in Phalanx," in *Proceedings Seventeenth IEEE Symposium on Reliable Distributed Systems (Cat. No. 98CB36281)*. IEEE, 1998, pp. 51–58.

[23] Y. Zhai, Q. Cao, J. Chase, and M. Swift, "Tapcon: Practical third-party attestation for the cloud," in *9th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 17)*, 2017.

[24] E. G. Sirer, W. de Bruijn, P. Reynolds, A. Shieh, K. Walsh, D. Williams, and F. B. Schneider, "Logical attestation: an authorization architecture for trustworthy computing," in *23rd ACM Symposium on Operating Systems Principles*, 2011, pp. 249–264.

[25] A. Brown and J. Chase, "Trusted Platform-as-a-Service: A foundation for trustworthy cloud-hosted applications," in *3rd ACM Cloud Computing Security Workshop*, Oct. 2011.

[26] F. B. Schneider, K. Walsh, and E. G. Sirer, "Nexus authorization logic (NAL): Design rationale and applications," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 8:1–8:28, Jun. 2011. [Online]. Available: http://doi.acm.org/10.1145/1952982.1952990

[27] S. Ceri, G. Gottlob, and L. Tanca, "What You Always Wanted to Know About Datalog (And Never Dared to Ask)," *IEEE Transactions on Knowledge and Data Engineering*, vol. 1, no. 1, pp. 146–166, 1989.

[28] J. DeTreville, "Binder, A Logic-Based Security Language," in *IEEE Symposium on Security and Privacy*. IEEE, May 2002, pp. 105–113.

[29] T. Jim, "SD3: A trust management system with certified evaluation," in *IEEE Symposium on Security and Privacy*. IEEE, May 2001, pp. 106–115.

[30] M. Abadi and B. T. Loo, "Towards a declarative language and system for secure networking," in *Proceedings of the 3rd USENIX International Workshop on Networking Meets Databases*, ser. NETDB'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 2:1–2:6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1361430.1361432

[31] Y. Yao, Q. Cao, P. Ruth, M. Cevik, C. Wang, and J. Chase, "Logical peering for interdomain networking on testbeds," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) - Computer and Networking Experimental Research using Testbeds (CNERT)*, 2020, pp. 824–829, extended arXiv version. [Online]. Available: https://arxiv.org/abs/2010.04707

[32] Q. Cao, Y. Yao, and J. S. Chase, "A logical approach to cloud federation," *arXiv:1708.03389*, 2017.