INVITED PAPER *Special Section on Log Data Usage Technology and Office Information Systems*

Elastic Trust Model for Dynamically Evolving Trust Frameworks

Hiroyuki SATO[†a)], ***Nonmember and*** **Noriyasu YAMAMOTO**[††], ***Member***

SUMMARY Today, trust plays a central role in services in distributed environments. Conventionally deployed trust has been based on static framework in which a server responds to a service request under statically determined policies. However, in accordance with evolution of distributed environments empowered with IoT and federated access mechanisms, dynamic behavior must be analyzed and taken into service provision, which conventional trust cannot properly handle. In this paper, we propose an extension of PDP (Policy Decision Point) in which assertions together with service requests are evaluated. Furthermore, the evaluation may be dynamically configured in dynamically evolving trust environment. We propose an elastic trust model in view of dynamic trust environment. This enables intuitionistic modeling of typical concrete elastic distributed services.

key words: *internet trust, policy decision point (PDP), assertion exchange*

1. Introduction

In several methodologies so far proposed and deployed for effective security, we have observed that we need a common framework or infrastructure on which organizations or societies rely. For organizational security, we need specifying a security border on which each security measure component is performed. For effective specification and operation of security border and security measures, it is required that the corresponding framework is correctly deployed and operated.

The same conclusion applies to the Internet trust. For the Internet trust to be effective, we need that the policies of trust are stipulated, that entities enroll into the community by contract representing the consent of the specified policies (trust boundary), and that the administrator of the trust commits the trust by regular compliance audit (trust measure). Thus operated community, called "trust framework," is established and some standard operation models have been proposed and deployed. For effective operations, we need an organizational framework in which enrolling entities are identified by the contract and the compliance audit is effectively performed.

Generally in modern distributed environments, the logic used to decide whether a server can accept or deny a given service request grows in complexity. To process complicated service requests, a modern server also collects evidence from peers and the surrounding environment to make a decision. This is implemented as collecting and evaluating assertions by the policies of a server. For an assertion to be elevated to evidence, a server must trust issuing entities of an assertion. Trust is indispensable for services in distributed environments also in this meaning.

In addition to this, the evolution of modern edge networks comprising of IoT devices including smartphones is also a driving force of dynamically changing network environments. Dynamic border reconfiguration in security and dynamic trust management of dynamically changing network are strongly required. While conventional trust management is designed for static trust environments, new technologies such as Self-Issued IdPs (Identity Providers) of OpenID Connect are emerging to handle such dynamic trust environments where smartphones play a central role. It is essential to analyze these dynamically changing distributed environments, and to build frameworks enabling dynamic evaluation of policies in terms of security and trust.

To this aim, this paper first defines a framework for evaluating assertions under given trust. Based on the framework, we propose Elastic Trust in which dynamically changing trust can be represented. Elastic Trust can dynamically update the trust values and the set of trusted entities, which is implemented upon assertion exchange among trusted entities and environments, including conventional assertion exchange seen in SAML and OpenID Connect used in access federations.

The rest of this paper is organized as follows: in Sect. 2, we specify the situations where dynamic trust management is essential. Section 3 gives a formal definition of policy decision points (PDP) that evaluate assertions to elevate to evidence. In Sect. 4, we propose Elastic Trust that enables dynamic and elastic trust management. In Sect. 5, we describe typical concrete trust scenarios by using Elastic Trust. Section 6 surveys related work. Section 7 concludes this paper.

2. Services and Trust in Distributed Environments

2.1 Classical Trust Frameworks

In modern architectures of distributed service environments, it is commonly observed that individual servers federate to provide a single mash-up service. This service design dates back to service oriented architecture (SOA). These days, abundant sets of service components are provided, and the deployment cost is drastically reduced. They include API

Manuscript received November 1, 2018.
Manuscript publicized June 25, 2019.

[†]The author is with the University of Tokyo, Tokyo, 113–8658 Japan.
[††]The author is with Fukuoka Institute of Technology, Fukuoka-shi, 811–0295 Japan.
a) E-mail: schuko@satolab.itc.u-tokyo.ac.jp

DOI: 10.1587/transinf.2018OFI0001

Copyright © 2019 The Institute of Electronics, Information and Communication Engineers

for service integration and dedicated protocols for federated authentication such as SAML and OpenID Connect, and for federated authorization such as OAuth.

In providing such services, a problem arises: how we can trust a communication peer and its data. This problem is critical especially in an access federation where authentication and authorization are delegated to third parties. Trust framework has been designed and deployed in order to solve this trust problem in access federations. In this framework, service providers delegate their authentication process to third parties called identity providers. If the identity providers belong to the same trust framework in which the authentication is actually completed, the service providers trust assertions issued by the identity providers. For this kind of trust to be established, a certain set of criteria is specified regarding verified operations of dedicated protocols and secure management of identities and their credentials. Furthermore, maturity of operating organization is required. Classical trust framework is designed and operated to guarantee that the operations of the service and identity providers are trustworthy enough so that criteria of the framework is fulfilled. Concretely,

1. the policies of governance and operations are defined first.
2. Next, they are enforced to enrolling entities in the form of contracts.
3. Enrolling entities provide their services, trusting that the policies are enforced to other entities by the trust framework.
4. The trust framework commits the trust by regularly or irregularly auditing the entities for the compliance with the policies, thus by guaranteeing the compliance posteriorly.

Given a single trust framework, there are two kinds of entities: Identity Providers (IdP) that provide identity information and Relying Parties (RP) that leverage given identity information. Within the framework, they trust each other.

Classical trust frameworks have been explicitly or implicitly deployed in several forms. One form is that so called "platformers" that provide social identities together with services of their own or those of third parties under federation. Here, IdP: RP = one: many. This form is widely accepted, and stably operated under the trust of platformers. Since around 2003 when the specification of SAML 2.0 was released, another form has emerged. Academia in the world has begun access federation in which universities access resources of RPs by using IdPs operated by them. Here, IdP: RP = many: many. This kind of academic access federations have been world-widely deployed. However, also in this early access federation, trust was implicit.

Following this trend, in 2010, United States announced the initiative of National Strategy for Trusted Identities in Cyberspace (NSTIC). Its objective includes replacement of the scheme of individual account issuance by each federal service provider with leveraging identities of social identities and those issued by IdPs operated by trustworthy organizations. To implement this scheme, establishment of trust between IdPs and RPs must be defined. For this purpose, trust frameworks are explicitly specified and deployed by using related policies and bodies for their enforcement (e.g. TFPAP [1], Kantara Initiative).

2.2 Trust Issues under Modern Distributed Environments

The forms of service and computing provision have drastically evolved since the design and deployment of classical trust frameworks. How to adapt trust so as to handle the changes described below is a central issue on trust deployment in modern distributed environments.

2.2.1 IoT Device Network

Classical trust has been designed on the assumption that a relatively small number of servers are operated by stable organizations. Servers are assumed to have sufficient power of computing and communications. Under these assumptions, the Internet-wide global trust has been established. However, modern Internet architectures have evolved so that a number of IoT devices play a major role as sensors and/or actuators. They communicate with each other to build a significant part of the Internet. Their behaviors are determined by their host nodes whose policies specify how IoT nodes enroll into the network, how they leave it, and how trustworthy the produced data is. This situation is very similar to the role of conventional trust framework. However, because all actions are completed online, we have to devise a different method from offline contracts. Furthermore, we have to solve the problem of how we elevate thus constructed trust circle around a host node to a global trust. While classical trust framework only determines the trust circle by offline contracts, we have to extend the evaluation of global trust levels based on dynamic trust related behaviors of enrolling entities.

2.2.2 Flexible Authentication Schemes

Authentication is now a significant gateway to service provision. As authentications have evolved and become diversified, we have to flexibly handle the diversity.

Flexible authentication has been partly proposed and implemented in various ways such as classification of authentication levels depending on the required levels of assurance (LoA) for providing services, and dynamic tuning of assurance levels in LoA elevation.

In addition, advanced authentication is affected by dynamic configuration of its trust circle and monitored data of its environment. For example, Self-issued IdP of OpenID Connect, assuming a trusted user environment provided by smartphones, is specified so that an RP can install an IdP dedicated to a specific user. In this framework, an RP can extend its own trust circle by using a specific trust anchor such as smartphones.

Continuous authentication based on behavior analysis

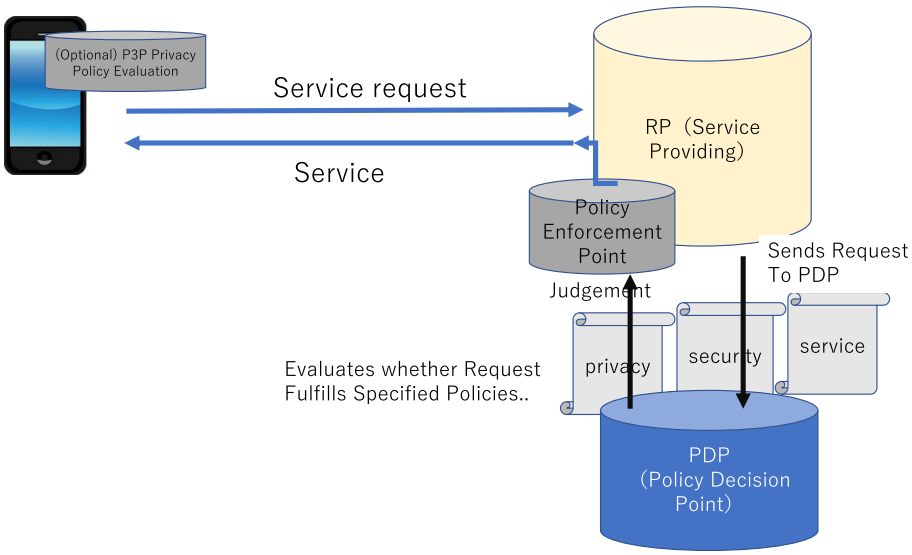

Fig. 1 Service request processing model by PDP and PEP

is another major example that enables dynamic tuning of levels of security and trust. In behavior analysis, behaviors of enrolling entities (typically, users) are monitored, and LoA of authentication is dynamically tuned by using predefined policies and logic. When LoA is elevated, entities obtain extra trust, which may result in additional offer of services. When degraded, entities lose trust, which may result in refusal of specific services. This scheme is actually deployed by some SNS providers. In this deployment, user behaviors are recorded. If any irregular use such as login from an unusual terminal and that from unusual places is detected, it is reported as anomalies, and sometimes service degradation such as login refusal is determined. More sophisticated anomaly detection to affect a level of trust given to individual entity is also proposed.

2.3 Service Model using Assertions and their Evaluation

Let us consider a general model to process service requests (Fig. 1). In this model, request processing is initiated by a client. A service request is sent to an RP by a client. The RP specifies and controls policies related to its own service, security and privacy. The request is evaluated by its Policy Decision Point (PDP) in terms of the policies. PDP makes a decision based on the evaluation, and the Policy Enforcement Point (PEP) enforces the decision. The result (allow/deny) is sent back to the client.

Service requests are generally access requests to a specific server resource. We consider their extension so that a client can add auxiliary data that is generally called "assertion." In this extended scheme, a client sends a service request together with a set of assertions.

[2] proposes a model in which in a given trust circle, an RP receives a service request and a set of assertions that are also evaluated under the trust policies of the RP. Some assertions are evaluated as trustworthy, and elevated to evidence. Thus collecting evidence in this way, PDP decides whether it allows the service request. Assertions may include general documents such as operation policies of a client. Also in this case, an RP evaluates the operation policies to make a decision. This model can be applied to the case where an RP requests service of another RP. RPs can integrate services of third parties whose policies match those of requesting RPs. However, in this model, dynamic trust growth is out of scope. In a modern service architecture, trust circle may dynamically grow or shrink, which must be represented and analyzed.

3. Formal Definitions of Trust and PDP

This section gives a formal definition of PDP. This models a PDP that decides whether a given service request is allowed or denied by evaluating assertions exchanged in a distributed environment. A distributed environment is a set of entities. An entity is an engine that executes given programs and communicates with other entities.

In this paper, we abstract PDP in an entity. In the rest of this paper, an entity runs under given policies *pol*, has a set of assertions exchanged in its communications, and decides allow or deny to a service request of another entity. Its set of assertions is exchanged in communications, and may be modified. Concretely,

Definition 1. An entity e has the attributes defined below:

Name An unique identifier $e.id$.
Policies A set of policies $e.pol$. It may consist of multiple policies regarding access control, privacy and security. A policy is represented by a "property."
Environment A set of entities $e.env$ that e can communicate with.
Trust The trust set $e.trust$.

"Properties" are defined as:

1. Terms consist of constants and variables for arithmetic and sets together with the ones below:
   - entity,
   - service requests,
   - internal representation $\lceil P \rceil$ for a property P.
2. A property is a formula of the first order logic that contains the following primitive formulas.
   - $\top$, $\bot$,
   - predefined primitive formulas P,
   - $Allow(e, r)$, $Deny(e, r)$ for entity e and service request r,
   - $assert(e, \lceil P \rceil)$ for entity e and a property P.

An assertion is defined as the internal representation of a formula of the form $assert(e, \lceil P \rceil)$. It has an intuitive meaning that an entity e claims a property P.

Trust for an entity e is a set of pairs of an entity f and its assertion that e trusts. Given a possible set S_f of assertions of f that e trusts, trust of e is represented by $\{(f, \lceil assert(f, \lceil P \rceil) \rceil) | f : \text{entity}, assert(f, \lceil P \rceil) \in S_f\}$

We denote by $A \models^e P$ that a property P holds in the condition that e has policies $e.pol$, trust $e.trust$, and a set of assertions A.

A set of assertions may grow in the communications with other entities, or may shrink by expiration of assertions. We use the symbol $\rightarrow$ to denote that a set of assertions of an entity changes from A by receiving assertions or observing expiration. We denote by $A \rightarrow A \cup B \models^e P$ that the set of assertions that e uses changes by receiving a new set of assertions B, and e makes a decision on P under a new set $A \cup B$.

$e.pol$ is specified and used for an entity e to decide $Allow(f, r)$ or $Deny(f, r)$ for a request r of an entity f.

Given a set of assertions A, $\models^e$ satisfies the relation under given $e.pol$ and $e.trust$ specified below:

Definition 2.
- For any A, $A \models^e \top$.
- For a primitive formula P, $A \models^e P$ if the predefined interpretation of P is true.
- If $e.pol \ni P$, $A \models^e P$
- If $e.trust \ni (f, \lceil assert(f, \lceil P \rceil) \rceil)$, and $A \ni \lceil assert(f, \lceil P \rceil) \rceil$, then $A \models^e assert(f, \lceil P \rceil)$ and $A \models^e P$.
- $A \models^e P \wedge Q$ is equivalent to $A \models^e P$ and $A \models^e Q$.
- $A \models^e P \vee Q$ is equivalent to $A \models^e P$ or $A \models^e Q$.
- $A \models^e P \rightarrow Q$ is equivalent to $A \models^e Q$ if $A \models^e P$.
- $A \models^e \neg P$ is equivalent to not $A \models^e P$.
- If $A \models^e P(t)$ for all entities $t \in e.env$, $A \models^e \forall x.P(x)$.
- If $A \models^e P(t)$ for some entity $t \in e.env$, $A \models^e \exists x.P(x)$.

In this system, an entity e makes an inference of P from $assert(f, \lceil P \rceil)$, the claim of a property P. In other words, it is essential that e trusts an assertion of f $((f, \lceil assert(f, \lceil P \rceil) \rceil) \in e.trust)$ so that it is elevated to evidence.

Example 1. Let $Authed(f, p)$ represent that an entity p is authenticated on entity f. Furthermore, let $e.pol \ni \forall p.\forall r.(Authed(f, p) \rightarrow Allow(p, r))$, and for all P, $e.trust \ni (f, \lceil assert(f, \lceil P \rceil) \rceil)$. An entity p issues a service request r together with $Authed(f, p)$ to entity e when f issues $\lceil assert(f, \lceil Authed(f, p) \rceil) \rceil$. When e receives this request and the assertion, the original set of assertions is augmented with the assertion, from the inference of $A \rightarrow A \cup \{\lceil assert(f, \lceil Authed(f, p) \rceil) \rceil\} \models^e Authed(f, p)$, and therefore $Allow(p, r)$ holds for e. This is the simplest model for access federation. By accepting trust and policies in enrolling a given access federation, e works as an entity of the access federation.

Example 2. Let us consider $assert(e_0, \lceil assert(e_1, \lceil P \rceil) \rceil)$. When $e.trust \ni (e_0, a)$ and $e_0.trust \ni (e_1, a)$ for every assertion a, $\{\lceil assert(e_0, \lceil assert(e_1, \lceil P \rceil) \rceil) \rceil\} \models^e assert(e_1, \lceil P \rceil)$ holds. Furthermore, $\{\lceil assert(e_0, \lceil assert(e_1, \lceil P \rceil) \rceil) \rceil\} \models^{e_0} P$ in e_0. For e to $\{\lceil assert(e_0, \lceil assert(e_1, \lceil P \rceil) \rceil) \rceil\} \models^e P$, the assertion $\lceil assert(e_0, \lceil P \rceil) \rceil$ must be issued by e_0 to e.

Consistency of Assertions

For a set of assertions A to enable meaningful inference, PDP must control A so that $A \not\models^e \bot$. When this holds, the set of assertions is called consistent. In the rest of this paper, we consider only consistent sets of assertions.

Policies of Assertion Issuance

The system so far defined represents the logic of PDP. The logic of PDP consists of the policies under which given assertions can be trusted. This logic allows internal representation of a property P, which means that assertions can be taken as an argument of *Allow* and *Deny*. This enables representation of the issuance policy of assertions.

Let us revisit Example 1. The entity p must initiate the process by requesting issuance of $\lceil assert(f, \lceil Authed(f, p) \rceil) \rceil$ to f. Let $issueReq$ be this service request. PDP of f can specify the policy of assertion issuance under a certain condition C as $C \rightarrow Allow(p, issueReq(\lceil assert(f, \lceil Authed(f, p) \rceil) \rceil))$. PDP of f, on the other hand, may infer $B \models^f Allow(p, issueReq(\lceil assert(f, \lceil Authed(f, p) \rceil) \rceil))$ under a certain set of assertions B such that $B \models^f C$. By this conclusion, f can issue the related assertion.

Policy Set of an Entity

A PDP often evaluates related policies of a peer entity. P3P (Platform for Privacy Preference Project) is its typical example. Assuming that all policies are made machine-readable, [3] gives a model in which policies of a peer such as security and privacy can be directly processed as data, and are sent to the peer PDP. This service model is subsumed in the model of this paper in the way that the policy set

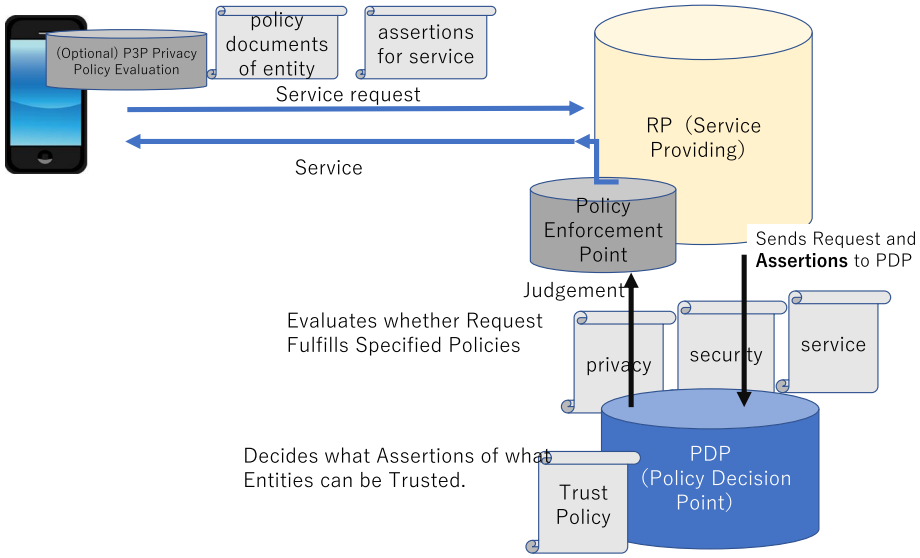

Fig. 2 Extended PDP model leveraging assertions

$\{P_1, P_2, \cdots, P_n\}$ of an entity e is represented as a set of assertions $\{\lceil assert(e, \lceil P_1 \rceil)\rceil, \cdots, \lceil assert(e, \lceil P_n \rceil)\rceil\}$ which can be exchanged as data between entities.

Figure 2 illustrates PDP extended so as to process assertions. In Fig. 2, we observe that policies and assertions are sent to PDP, and they are evaluated under the trust policy of PDP.

4. Elastic Trust

For a given entity e, we have observed that its PDP runs under $e.trust$ and $e.pol$. In order to manage them, we need to control $e.env$, dynamic behaviors of $e.trust$, and trust anchors because the initial set of trust must be separately determined. In this paper, we consider such control policies as meta-policies of PDP, and propose Trust PDP (TPDP) in order to dynamically control the trust of e. We denote by Elastic Trust dynamically growing/shrinking trust whose behaviors are specified by TPDP.

Concretely, TPDP controls the trust of an entity in the way below:

- an entity e, by specifying a set of trustworthy assertions S, and executing $create(S, TP, P)$, can create a uniquely identifiable entity in $e.env$ where TP is the TPDP policy and P is the PDP policy of the created entity.
- The name id of a created entity belongs to at least one name space. In the specified name spaces, the created name is unique.

Definition 3. TPDP properties are defined as formulas of the first order logic on $trust(t, A)$, $join(t, A)$, $leave(t, A)$ for a entity t and a set of TPDP assertions A defined below, $created(e, t, S, TP, P)$ for entities e, t, S a set of assertions, TP a set of TPDP properties, and P a set of properties, and $assert(t, \lceil P \rceil)$ for an entity t and a TPDP property P.

As the entailment in TPDP properties in an entity e, we add the axiom schema to the usual first order logic as below:

- $trust(e, \top)$.
- If $trust(t, A)$ is valid and $A \ni \lceil P \rceil$, then $assert(t, \lceil P \rceil) \rightarrow P$ is valid (corresponding to Definition 2).

TPDP policy is defined as a set of TPDP properties. *join* and *leave* in TPDP policies correspond to *Allow* and *Deny* in PDP policies, respectively. *join* is effective if another entity that creates the target entity is trustworthy, even if it itself does not create the target. Furthermore, an entity may follow its own policy and independently decide wheter it adopts claims of *leave*.

TPDP assertions are defined as assertions on TPDP properties and *created* assertions. They are issued by an entity e in the way below:

- $\lceil assert(e, \lceil created(e, t, S, TP, P)\rceil)\rceil$ is issued when an entity t is created by entity e with a certain S, TP, P by $create(S, TP, P)$.
- $\lceil assert(e, \lceil join(t, S)\rceil)\rceil$ is issued when e itself determines to trust the set of assertions S of t.
- $\lceil assert(e, \lceil leave(t, S)\rceil)\rceil$ is issued when e itself determines to untrust the set of assertions S of t.

Trust Status and Trust Status Transition

join and *leave* initiated by some entity are propagated to the trust of other entities. Trust status, the domain of assertions that the entity trusts, can grow or shrink in our system. An entity claims the range of its own trust status by continuously issuing its TPDP assertions. These assertions determine growth and shrink of the entity's trust $e.trust$. We define this growth and shrink as the trust status transition.

Definition 4. • A trust status of an entity e is defined as

$$\begin{array}{ccc} \mathbf{TPDP} : (e, TP, T, TA) & \Longrightarrow & (e, TP, T', TA') \\ \downarrow(\mathbf{determines}) & & \downarrow(\mathbf{determines}) \\ \mathbf{PDP} : T = e.trust \models^e & \overset{(update)}{\longrightarrow} & T' = e.trust \models^e \end{array}$$

Fig. 3 Relation between TPDP and PDP in trust status transition

a tuple (e, TP, T, TA) where TP is a trust policy comprising of a set of TPDP properties and T is the trust of e, and TA is a sequence of TPDP assertions.

- When TA, a given sequence of TPDP assertions, changes to TA', and T changes to T' accordingly, we denote it by $(e, TP, T, TA) \Rightarrow (e, TP, T', TA')$. $(e, TP, T, TA) \Rightarrow (e, TP, T', TA')$ is defined as follows.
- When $TA = \emptyset$ (initial status) and $trust(f, A)$ can be inferred in TP, then $T' \supset \{f\} \times A$.
- When $TA' = TA + \{\lceil assert(e, \lceil created(e, t, S, TP, P)\rceil)\rceil\}$, $(t, TP, \{e\} \times \top, \emptyset)$ is created as a new trust status for t.
- When $TA' = TA + \{\lceil assert(t, \lceil join(s, B)\rceil)\rceil\}$ and $join(s, B)$ can be entailed in TP, $T' = T \cup (\{s\} \times B)$.
- When $TA' = TA + \{\lceil assert(t, \lceil leave(s, B)\rceil)\rceil\}$ and $leave(s, B)$ can be entailed in TP, $T' = T \setminus (\{s\} \times B)$,

where + represents concatenation to a given sequence.

The inference of PDP depends on a trust status (e, TP, T, TA). Here, $e.trust = T$ is enforced. That is, an entity continuously receives TPDP assertions by which $e.trust$ is modified so as to determine the evaluation policy of PDP. Furthermore, $e.env$ is updated to contain or remove s that issues B. Figure 3 summarizes the relation of PDP and TPDP.

5. Analysis of Dynamic Trust Behaviors

Scenarios of Dynamic Trust Behaviors

We show some typical scenarios of distributed services where elastic trust plays an essential role.

Example 3. Let an entity e create a set of sensors $s_i (i \in N)$. An individual sensor s_i is created with its TPDP policy $\{trust(e, \top)\}$ and its PDP policy $\{e\} \times \top$. e will issue assertions $\lceil assert(e, \lceil created(e, s_i, \top, \{e\} \times \top, \emptyset)\rceil)\rceil$ and $\lceil assert(e, \lceil join(s_i, \top)\rceil)\rceil$ that will be received by e itself, and will make the PDP policy of e modified. Furthermore, $e.env$ is updated to contain s_is.

On the side of a sensor s, data d will be transmitted as an assertion $\lceil assert(s, \lceil sensed = d\rceil)\rceil$. e receives it, and issues $\lceil assert(e, \lceil assert(s, \lceil sensed = d\rceil)\rceil)\rceil$. Data d will be received by entities that trust e, and be utilized.

Example 4. We consider a scenario that a security monitor SM delivers its decision to enrolling entities. Enrolling entities e have a TPDP policy $TP \ni trust(SM, \top)$ which means that e trusts every TPDP assertions from SM. When SM delivers a TPDP assertion set $A = \{\lceil assert(SM, \lceil leave(f, \top)\rceil)\rceil\}$ to e, its trust status changes $(e, TP, T, TA) \Rightarrow (e, TP, T \setminus (\{f\} \times \top), TA + \{\lceil assert(SM, \lceil leave(f, \top)\rceil)\rceil\}$. This means that f is excluded from the trust circle, and removed from $e.env$.

Example 5. Let us revisit a classical trust framework of access federation. There, protocols for use are specified in the policy. The list of enrolling entities is published by a special server of the federation (called "metadata server"). The metadata server M publishes the set of enrolling entities $e_i (i = 1, \cdots)$ that use the specified protocol set Pr in the form of assertions $TA_M = \{\lceil assert(M, \lceil join(e_i, Pr)\rceil)\rceil | i = 1, \cdots\}$. An enrolling entity e initializes its trust policy as $TP_e = \{trust(M, Pr)\}$. Letting $trust_{init} = \{M\} \times Pr$, e trusts assertions sent in protocol Pr. Then, e updates its trust status by using TPDP assertions sent by M in the following way.

$$\begin{array}{l} (e, TP_e, \emptyset, \emptyset) \Rightarrow \\ (e, TP_e, trust_{init}, \emptyset) \Rightarrow \\ (e, TP_e, trust_{init} \cup (\{e_i | i = 1, \cdots\} \times Pr), TA_M) \Rightarrow \\ \cdots \end{array}$$

By this, entities listed in the metadata are trusted. M may issue $\lceil assert(M, \lceil leave(f, Pr)\rceil)\rceil$ for excluding f from its trust circle. This is processed in the similar way as Example 4.

Example 6. We formulate Self-Issued IdP in the following way. In addition to the processes of Example 1, we need the steps below.

1. We assume that a service provider s runs under the PDP policy of Example 1. Mobile apps are its typical example.
2. s creates IdP i for itself, and *join*s it in its trust. Then, the new trust circle $TC = \{trust(s, \top), trust(i, \top)\}$ is established between s and i. The trust status is updated from its initial TPDP policy $TP = \{trust(s, \top)\}$ and a trust policy T to the one below:

$$\begin{array}{l} (s, TP, T, TA) \Rightarrow \\ (s, TP, T \cup (\{i\} \times \top), TA \cup \\ \{\lceil assert(s, created(i, \top, TC, P)\rceil, \lceil assert(s, \lceil join(i, \top)\rceil)\rceil\}) \end{array}$$

3. i is operated under a specific policy P restricted to an IdP of Example 1 so that only authentication assertions for prespecified user p are issued.
4. s requests an authentication assertion from i to a service request. After certain communications, s receives assertions needed to infer $Authed(i, p)$. The PDP of s infers $Authed(i, p)$ by using the trust status where i is trusted and the fact that i is the issuer of the related assertion.

Trust Anchors

The trust $e.trust$ of an entity e grows or shrinks by receiving and evaluating TPDP assertions including *join* and *leave*. To add an entity s with assertion set B to e's trust, $trust(e, \lceil join(s, B)\rceil)$ must be valid as the result of a chain

of entailment in the e's TPDP. This chain of trust must start from a certain entity a that must be specified as $trust(a, A)$ for some A in the TPDP policy of e. This entity specified in the trust policy is called "trust anchor." Security monitors and metadata servers are its typical examples. Examples 3 to 6 explicitly assume fixed trust anchors. In Definition 4, in creating an entity, its trust anchor is defined as the one that creates it. In IoT where communications environment is limited, the trust anchor may be a national telecom carrier [4]. Controlling the set of trust anchors is one of major functions of a trust framework, which may lead further extension of our model.

6. Related Work

The Internet trust has been designed for assuring the existence of an entity requesting services by assessing the assurance of identities and authentication by using predefined policies. NIST SP800-63 [5] is a standard used in this assessment. Trust frameworks of access federations are deployed as a major application of the Internet trust. In Japan, GakuNin (https://gakunin.jp) for academia is in operation. Furthermore, the federation of academic access federations is being built as eduGain (https://edugain.org).

As Internet service architectures evolve, their trusts are forced to change, accordingly. Cloud trust analysis has started around 2010 (e.g. [6], [7]), and now in operation in the forms such as ISO (ISO 27017) and CSA (STAR). Related audit has been specified and performed. Effective operation of IoT trust [4], [8] is one of the most strongly required trusts. There have been proposed assurance levels and anchors of IoT trust. IoT is characterized differently to conventional distributed environments where each IoT node has its own local connections and name space. Furthermore, nodes can dynamically be created, grow and vanish, which means that it is hard to consistently control trust. [9] analyzes service management of IoT environments including the hierarchical control for name spaces. [10] discusses identity management of IoT.

Methodologies of trust assessment have been matured. Assessment of operating policies has been added to enable more effective assessment [3], [11]–[15]. It is generally based on evaluating machine-readable policies. There have been proposed some specifications for them including P3P [16] and privacy labels [17].

Today, it is commonly observed that trust status is never static, but may change to another, which may be initiated by changes of surrounding services and environments. Dynamic trust status transitions have been analyzed in a number of scenarios including multiple assurance levels [1], dynamic trust level elevation [18], creation of trust by Self-Issued IdP [19], reflection of security monitoring [20], [21] and continuous authentication by behavior monitoring and analysis [22], [23].

7. Concluding Remarks

In this paper, we have proposed a formal system in which PDP receives assertions from a peer, and evaluates them to elevate to evidence. Furthermore, we have proposed Elastic Trust in which we can dynamically evaluate trust status transition in order to analyze dynamically growing/shrinking trust.

Concretely, first, we have formalized behaviors of PDP where entities exchange service requests and assertions in distributed environments. It evaluates assertions to elevate to evidence, and makes a decision on the basis of thus generated evidence. Next, we have given a system Elastic Trust in which trust status may dynamically change. That is, TPDP has been established on top of PDP where trust status transits by exchanging TPDP assertions on *join* and *leave*. In addition, by using this system, we have analyzed typical scenarios of dynamic trust management such as IoT, a security monitor, classical trust for access federations, and dynamic trust for Self-Issued IdP.

Acknowledgments

This research was partially supported by KAKENHI (Grants-in-Aid for Scientific Research) (C) 19K11958 and (B) 19H04098.

References

[1] FICAM, "Trust Framework Provider Adoption Process for Levels of Assurance 1, 2, and Non-PKI 3," 2009.
[2] H. Sato, S. Tanimoto, T. Kobayashi, and A. Kanai, "Adaptive policy evaluation framework for flexible service provision," 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE), pp.124–131, March 2018.
[3] H. Sato, S. Tanimoto, and A. Kanai, "A Policy Consumption Architecture that enablesDynamic and Fine Policy Management," 3rd ASE International Conference on CyberSecurity 2014, 2014.
[4] H. Sato, A. Kanai, S. Tanimoto, and T. Kobayashi, "Establishing Trust in the Emerging Era of IoT," 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE), pp.398–406, 2016.
[5] P.A. Grassi and J.L. Fenton, "Digital authentication guideline," Technical Report SP800-63-3, NIST, 2017.
[6] H. Sato, A. Kanai, and S. Tanimoto, "A Cloud Trust Model in a Security Aware Cloud," 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet, pp.121–124, 2010.
[7] K.M. Khan and Q. Malluhi, "Establishing trust in cloud computing," IT Professional, vol.12, no.5, pp.20–27, Sept. 2010.
[8] S. Sicari, A. Rizzardi, L.A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead," Computer Networks, vol.76, pp.146–164, 2015.
[9] W. Shang, Z. Wang, A. Afanasyev, J. Burke, and L. Zhang, "Breaking out of the cloud: Local trust management and rendezvous in named data networking of things," 2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI), pp.3–14, April 2017.
[10] P. Mahalle, S. Babar, N.R. Prasad, and R. Prasad, "Identity management framework towards internet of things (iot): Roadmap and key challenges," Recent Trends in Network Security and Applications, eds. by N. Meghanathan, S. Boumerdassi, N. Chaki, and D. Nagamalai, vol.89, pp.430–439, Springer Berlin Heidelberg, Berlin,

Heidelberg, 2010.
[11] P. McDaniel and A. Prakash, "Methods and Limitations of Security Policy Reconciliation," ACM Trans. Inf. Syst. Secur., vol.9, no.3, pp.259–291, Aug. 2006.
[12] R. Sahay, G. Blanc, Z. Zhang, K. Toumi, and H. Debar, "Adaptive Policy-driven Attack Mitigation in SDN," Proceedings of the 1st International Workshop on Security and Dependability of Multi-Domain Infrastructures, pp.4:1–4:6, XDOMO'17, ACM, 2017.
[13] M. Burnside and A.D. Keromytis, "Asynchronous Policy Evaluation and Enforcement," Proceedings of the 2nd ACM Workshop on Computer Security Architectures, pp.45–50, CSAW '08, ACM, 2008.
[14] S.V. Macua, J. Chen, S. Zazo, and A.H. Sayed, "Distributed Policy Evaluation Under Multiple Behavior Strategies," IEEE Trans. Automatic Control, vol.60, no.5, pp.1260–1274, Nov. 2014.
[15] C. Basile, A. Lioy, C. Pitscheider, and S. Zhao, "A Formal Model of Policy Reconciliation," 2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp.587–594, 2015.
[16] L. Cranor, B. Doubs, S. Egelman, G. Hogben, J. Humphrey, M. Langheinrich, M. Presler-Marshall, J. Reagle, M. Schunter, D.A. Stampley, and R. Wenning, "The Platform for Privacy Preferences 1.1 (P3P1.1) Specification," Technical report, WWW Consortium, 2006.
[17] P.G. Kelley, J. Bresee, L.F. Cranor, and R.W. Reeder, "A "Nutrition Label" for Privacy," Proceedings of the 5th Symposium on Usable Privacy and Security, pp.4:1–4:12, SOUPS '09, ACM, 2009.
[18] H. Sato, "A Formal Model of LoA Elavation in Online Trust," ASE Science Journal, vol.1, no.4, pp.166–178, 2012.
[19] N. Sakimura, J. Bradley, M.B. Jones, B. deMedeiros, and C. Mortimore, "OpenID Connect Core 1.0," 2014.
[20] V.A. Desnitsky, I.V. Kotenko, and S.B. Nogin, "Detection of Anomalies in Data for Monitoring of Security Components in the Internet of Things," 2015 XVIII International Conference on Soft Computing and Measurements (SCM), pp.189–192, 2015.
[21] W.-L. Cheng, T.-C. Chuang, C.-W. Yang, Y.-H. Lin, M. Liu, and C. Yin, "An Integrated Security Monitoring System for Digital Service Network Devices," 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp.118–122, 2017.
[22] I. Brosso, A.L. Neve, G. Bressan, and W.V. Ruggiero, "A continuous authentication system based on user behavior analysis," 2010 International Conference on Availability, Reliability and Security, pp.380–385, Feb. 2010.
[23] C. Shen, Y. Chen, and X. Guan, "Performance evaluation of implicit smartphones authentication via sensor-behavior analysis," Information Sciences, vol.430-431, pp.538–553, 2018.

Hiroyuki Sato 1985, 1987, 1990 Dept. Information Science, the University of Tokyo, B.Sc., M.Sc., D.Sc, respectively. 1990 Assistant Professor of Kyushu University. Presently, Associate Professor of Information Technology Center, the University of Tokyo. Specialties: Security, Internet Trust. Accredited assessor of LoA 1 for Kantara Initiative.

Noriyasu Yamamoto received BE degree from Nagasaki University, Japan, in 1990. He received ME and PhD degrees from Kyushu University, Japan, in 1992 and 1998, respectively. He is currently a Professor at Fukuoka Institute of Technology, Japan. His research interests include computer vision, distributed and parallel image processing, network security and Life Log systems. He is a member of IPSJ and IEICE.