Invited Paper

Authorization by Documents

Hiroyuki Sato[1,a)]

Received: December 1, 2016, Accepted: June 2, 2017

Abstract: These days, ICT service environments have dramatically changed in their complexity. Accordingly, related business logics for business processes such as provisioning, resource limit, conditional authorization and delegation have grown in its complexity. In this paper, we generalize the idea of access tokens of OAuth, and propose "authorization by documents." In our model, a user submits a document as evidence of privilege claim, and a server verifies the document to prove the appropriateness of the user's privilege. A document can be complicated, reflecting some business flow in an institution. If the process and result of business flow are expressed by using documents, the evidence as documents can reflect arbitrarily complex business flow. For this purpose, we formalize documents, and define document tree logic (DTL) as a variant of CTL* to express the policies associated with documents. Typical business processes including request and approval, delegation, and approval by document circular are expressed in DTL, and verified by using documents as evidence.

Keywords: documents, policies, authorization, verification, document tree logic, provisioning, delegation

1. Introduction

These days, ICT service environments have dramatically changed in their complexity. Particularly, because access federations have become one of standard service frameworks, restructuring of services has been strongly promoted. In an access federation, authentication and authorization can separately be served by dedicated service providers for authentication and for services. Accordingly, data exchange between servers is required for maintaining service integrity which was once guaranteed as providing services by a single "server." Therefore, the authentication process has become complicated. Concretely, the data integrity between the service providers, identity providers and user agents (Web browsers) is the essential requirement in authentication. We need guarantee that each data piece cannot be falsified for proving that an entity actually authenticates. Furthermore, as the authorization logic is sophisticated to represent some complicated process such as delegation and resource limit, authorization has also become complicated. We can say that this complexity is caused partly by the fact that multiple servers must involve in completing a single task such as authentication and authorization.

Actually, tasks of a server are not limited to authentication and authorization. Particularly, provisioning is a basis of service initiation and authorization. As authorization has begun to include complicated logics, provisioning becomes even more complicated. Conventionally, when an authority decides to create an account for a user, its backyard office is responsible for enabling the user to work on a given set of servers. In short, a workbench in a server must be provisioned by the backyard by using a separate path from the access by a user. **Figure 1** depicts this scenario. Here, service providers accept the modification request by a backyard to configure for a user.

$$\begin{tikzcd}
\textbf{Authority} \arrow[r, Rightarrow, "\text{account/privilege}"] \arrow[d, "\text{Data on Authorization}"'] & \textbf{(user)} \\
\textbf{Backyard} \arrow[r, Rightarrow, "\text{Provisioning}"'] & \textbf{Service Provider} \arrow[u, "\text{Service}"']
\end{tikzcd}$$

Fig. 1 Role of backyard against a service provider.

Because provisioning has conventionally been considered to be an offline task of the backyard, its optimization has been out of scope of ICT, and the heavy load of backyard has not been alleviated. Obviously, this model does not scale. Because the separation of privileges on the side of four players is not clear, there are required duplicated processes and backchannel communications. For instance, when user privileges are decided by an authority, the related data must be sent both to the user and the backyard. To improve the service integrity, the task of provisioning must be optimized so that too much cost is not caused to the backyard.

Let us analyze the task of provisioning. If a server provides provisioning, it must *implicitly* communicate with an authority that gives privileges to a user. A given authority determines users' privileges, reflecting their roles and tasks in an institution. Furthermore, their privileges granted by an authority must be proved with a kind of evidence, if the communication between the authority and the server is indirect. This kind of evidence is implemented as assertions in SAML or access tokens in OAuth. Exchanging evidence among servers enables the multiple servers to cooperate for completing authentication or authorization. However, such assertions assume stateful and complicated transac-

[1] The University of Tokyo, Bunkyo, Tokyo 113–8658, Japan
[a)] schuko@satolab.itc.u-tokyo.ac.jp

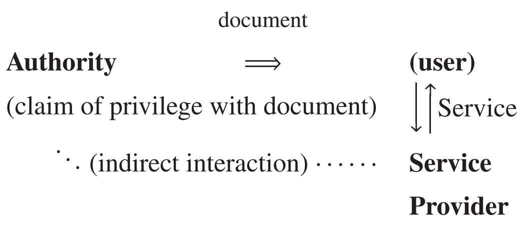

Fig. 2 Using documents for service claim – backyard-less approach.

tions to be appropriately processed by the corresponding servers. For example, in building assertions of SAML, we must take care so that the request id and the response id are consistent in processing the transaction. Moreover, access tokens of OAuth must be issued to a resource requester as the result of service registration and appropriate processing of the relevant request tokens.

However, when complicated logic for provisioning, resource limit, conditional authorization, and delegation must be considered, such evidence aforementioned is too simple, and therefore still causes careful processing by servers.

In this paper, we generalize the idea of access tokens of OAuth, and propose "authorization by documents." In our model, a user submits a document as evidence of privilege claim, and a server verifies the document to prove the appropriateness of the user's privilege. A document can be complicated, reflecting some business flow in an institution. If the process and result of business flow are expressed by using documents, the evidence as documents can reflect arbitrarily complex business flow. Thus, the service integrity is maintained by this evidence, and the flow of business processes including those of backyards is streamlined. **Figure 2** depicts this scenario. The indirect communication between Authority and Service Provider is implemented as submission of related documents via user (dotted line in Fig. 2). Compared with Fig. 1, the flow is streamlined, and the burden of backyard can be minimized.

The rest of this paper is organized as: Section 2 discusses the role of documents in business processes. Section 3 formalizes documents and document tree logic to express the evidence and policies. Section 4 applies thus defined documents and the logic to typical business processes. Furthermore, analysis in terms of expressive power is given. Section 5 surveys related work. Section 6 summarizes this paper.

2. Documents in Business Processes

In this section, we see how documents are used in business processes.

2.1 Publication of Policies of an Institution

Documents have a long history of publication and representation of policies. They are validated to claim some policies.

Particularly, if a document is published by an institution, it is regarded as official announcement of an institutional policies. This scheme is also used in IT services. When an institution provides a service, it publishes related policies such as a service policy and a privacy policy. A user or client validates the published policies, and subscribes to the service if the policy matches the client's requirement. In PKI, publication of certificate polices and certification practice statements (CP/CPS) plays a significant role in establishing trust.

Furthermore, there have been proposed mechanical validation schemes of policies such as P3P and Ref. [29] on the assumption that policies are digitally published.

2.2 Documents as Audit Trail

For audit, chronological records of documents that represent some sequence of activities are called "audit trail." It has a decisive power in verifying that some decision is appropriately made by following the policies of a given institution. The idea of audit trail is now widely applied to financial transactions, scientific research process together with traditional accounting. It is now a central topic in digital forensics. For trails, it is important that the sequence is chronologically correct so that one can prove that transactions are processed, following the policies of the institution. This also means that if a given trail is successfully verified, we can say that the related transaction is valid for a given policy.

2.3 Embedding Time in Documents

Traditionally, it was very hard to prove the chronological correctness of generated documents. Traditional forensics focused on direct or indirect evidence of time information in a given document. As modern forensics is being supported by modern information technologies, the objectivity has been much improved. Here, time-stamp and time-stamp certificates are the source of authority. Furthermore, if we do not care the precision of date and time, digital signature can verify the order of generation of documents. If a digital signature A is given to a document that has a digital signature B, we can say that signature A is given after signature B. Thus specified sequence is significant in determining the sequence of documents, which is essential in audit trail. Because chronologically correctness is essential in business processes and business transactions, we can say that digital signatures are very useful for sequencing the documents.

3. Documents and Document Tree Logic

As we have discussed so far, documents have the role of evidence for policies and trail. Evidence is validated for a given policy. Hereafter, we consider documents with digital signatures. They are shown to be very useful for representing the characteristics of policies and trail in a business transaction. In this paper, based on the notations of Ref. [27], we formalize documents with digital signatures and the related logic that represents the business processes.

3.1 Documents

Definition 1 We define terms in a document as:

person	$p ::= p_1, p_2, \cdots$
fieldname	$f ::= f_1, f_2, \cdots$
text	$t ::= t_1, t_2, \cdots$
val	$v ::= \text{person}\|\text{text}$
document name	$D ::= D_1, D_2, \cdots$
document form	$did ::= D_0[f_0, \cdots, f_{n_0}], D_1[f_0, \cdots, f_{n_1}], \cdots$

The D part of a document form represents its name. Its f part represents the arity of a document form.

Given a document form $D[f_0, \cdots, f_n]$ together with field names $f_0, \cdots, f_n$, documents and their contents are defined as below. First, documents *doc* is defined as:

Definition 2

$$doc ::= D[f_0, \cdots, f_n] : C_{(f_0,\cdots,f_n)}$$

Content is defined as below.

Definition 3 Content $C_{(f_0,\cdots,f_n)}$ of a document is defined as:

$$\begin{aligned} C_{(f_0,\cdots,f_n)} ::= & \; f_i = v \\ & \mid \; f_i = doc \; (0 \le i \le n) \\ & \mid \; C_{(f_0,\cdots,f_n)} \text{ signed by } \; p \\ & \mid \; (C_{(f_0,\cdots,f_n)}, \cdots, C_{(f_0,\cdots,f_n)}) \\ & \mid \; doc \end{aligned}$$

The form $(C_{(f_0,\cdots,f_n)}, \cdots, C_{(f_0,\cdots,f_n)})$ represents the concatenation of documents $C_{(f_0,\cdots,f_n)}$'s. In summary, a document is labeled with its name, meaning its intended use. Furthermore, a document contains content in which all values have their field name, meaning their intended use, and may contain a digital signature that will be used to represent its related workflow.

3.2 Document Tree Logic

In an institution, for a given policy for processing a document, the related document form is written and processed according to them. Thus written document is used as evidence of this business process. We observe the role of documents as follows: first, a set of document forms is determined by an embracing institution. Given a form, its objectives and intended use must be determined. Concretely, a form is understood to have request(s), and due process of its approval.

When a requester fills a form with some request, the filled form is processed according to the rules of approval. When the document is approved, then the document claims something that is authorized by an approver. The requester can claim some authority if one has the (approved) document, (and properly authenticates). In this meaning, a process of business workflow is embedded in a document.

The document tree logic (DTL) is defined to represent intention of the documents. DTL is a subset of CTL* (superset of CTL (computation tree logic) and LTL (linear temporal logic)) and built on persons and texts. We list their definitions in Definition 4. Let a finite set F of field names be given. For each field name $f_i \in F$, we define a unary predicate $F_i(\cdot)$. we denote such a predicate by *field predicate*. Let a set FP of field predicates be fixed. Moreover, we introduce a predicate Signed($\cdot$) for representing signing.

Definition 4 A Formula ϕ of DTL is defined as:

$$\begin{aligned} \phi ::= & \; true \mid false \\ & \mid \; P(t_0, \cdots, t_n, p_0, \cdots, p_m) \text{(primitive formulas)} \\ & \mid \; F_i(t) \quad (F_i \in \text{FP}) \\ & \mid \; \text{Signed}(p) \\ & \mid \; \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \\ & \mid \; \forall X.\phi \mid \exists X.\phi \\ & \mid \; \phi < \phi \mid \phi \dot{<} \phi \\ & \mid \; E\phi \mid A\phi \\ & \mid \; \overleftarrow{E}\,\phi \mid \overleftarrow{A}\,\phi. \end{aligned}$$

The formulas $\phi < \psi$ and $\phi \dot{<} \psi$ represents $(E\phi)\mathsf{U}\psi$ (there is a path that ϕ is true until ψ becomes true), and $\phi \wedge \mathsf{next}\psi$ in CTL*, respectively. Because our focus is on the order of fields and signatures, and we consider distinguished paths corresponding to signatures, we simply use the notation $<$ and $\dot{<}$ in DTL.

Next, we define a document tree associated to a given document.

Definition 5 Let a document $d \equiv D[f_0, \cdots, f_n] : C_{(f_0,\cdots,f_n)}$ be given. We define its associated document tree $[d]$, a variant of Kripke structure as follows:

(1) If $C_{(f_0,\cdots,f_n)}$ is of the form $f_i = v$, we have a node $[d] \equiv [D[f_0, \cdots, f_n] : f_i = v]$.

(2) If $C_{(f_0,\cdots,f_n)}$ is of the form $f_i = doc$, we have a node $[d] \equiv [D[f_0, \cdots, f_n] : f_i = doc]$, $[doc]$, and nodes and paths corresponding to *doc*. Moreover, we have the path $[d] \longleftarrow [doc]$.

(3) If $C_{(f_0,\cdots,f_n)}$ is of the form $C_{(f_0,\cdots,f_n)}$ signed by p, we have a node $[d] \equiv [D[f_0, \cdots, f_n] : C_{(f_0,\cdots,f_n)}$ signed by $p]$ and nodes and paths corresponding to $[D[f_0, \cdots, f_n] : C_{(f_0,\cdots,f_n)}]$. Moreover, we have the path $[d] \longleftarrow [D[f_0, \cdots, f_n] : C_{(f_0,\cdots,f_n)}]$. We call this single path component a *signing path component*.

(4) If $C_{(f_0,\cdots,f_n)}$ is of the form $(C_{0,(f_0,\cdots,f_n)}, \cdots, C_{m,(f_0,\cdots,f_n)})$, we have a node $[d] \equiv [D[f_0, \cdots f_n] : (C_{0,(f_0,\cdots,f_n)}, \cdots, C_{m,(f_0,\cdots,f_n)})]$ together with nodes corresponding to $[D[f_0, \cdots, f_n] : C_{i,(f_0,\cdots,f_n)}](i = 0, \cdots, m)$, and paths $[d] \longleftarrow [D[f_0, \cdots, f_n] : C_{i,(f_0,\cdots,f_n)}](i = 0, \cdots, m)$.

(5) If $C_{(f_0,\cdots,f_n)}$ is of the form *doc*, we have a node $[doc]$ together with nodes and paths corresponding to $[doc]$ and a path $[d] \longleftarrow [doc]$.

Naturally, the nodes and paths constructed as above form a tree. The root node corresponds to the whole document d, and nodes corresponding to $f_i = t$ are leaves. We call this tree a document tree associated with d.

Definition 6 In a document tree, we call a path that contains at least one signing path component a *signing path*.

Signing paths are associated with the sequence of document construction.

3.3 Document as Evidence Relation

Next, we define the interpretation $d \models P$ for a document d and a DTL formula P, meaning a document d is evidence to P. The interpretation is based on the Kripke semantics of CTL*. The (only) substantial difference is that a signing path is significant.

Kripke semantics on a partial order is constructed in the way as assigning a model to each node, and a transition between models to each arrow. To a document d, we assign a node $[d]$. To each node, we assign a model in which (set-theoretic) interpretations of primitive formulas P's are fixed.

Definition 7 Let us assume that the interpretations of primitive formulas are given. Then, given a document d and a path p in the document tree of d, we define the model relations $d \models$ and $p \models$ in the following way.

As for state formulas,

- $d \models true$ iff $true$.
- Never $d \models false$.
- $d \models P(t_1, \cdots, t_n, p_1, \cdots, p_m)$ iff $P(t_1, \cdots, t_n, p_1, \cdots, p_m)$ is true in the given interpretation of P

in $[d]$.

- $d \models F_i(v)$ iff d is of the form $D[f_0, \cdots, f_n] : f_i = v(i = 0, \cdots, n)$.
- $d \models \text{Signed}(p)$ iff d is the form $D[f_0, \cdots, f_n] : C_{(f_0, \cdots, f_n)}$ signed by p.
- $d \models \phi_0 \wedge \phi_1$ iff $d \models \phi_0$ and $d \models \phi_1$.
- $d \models \phi_0 \vee \phi_1$ iff $d \models \phi_0$ or $d \models \phi_1$.
- $d \models \phi_0 \longrightarrow \phi_1$ iff if $d \models \phi_0$, then $d \models \phi_1$.
- $d \models E\phi$ iff for some path p starting from $[d]$, $p \models \phi$.
- $d \models A\phi$ iff for all paths p starting from $[d]$, $p \models \phi$.
- $d \models \overleftarrow{E}\,\phi$ iff for some path p ending with $[d]$, $p \models \phi$.
- $d \models \overleftarrow{A}\,\phi$ iff for all paths p ending with $[d]$, $p \models \phi$.

As for path formulas, for a path p,

- $p \models \phi_0 < \phi_1$ iff p is a signing path of the form $d_0 \longrightarrow d_1 \longrightarrow \cdots \longrightarrow d_n$, $d_0 \models \phi_0$, and $d_n \models \phi_1$.
- $p \models \phi_0 \dot{<} \phi_1$ iff p is a signing path of the form $d_0 \longrightarrow d_1 \longrightarrow \cdots \longrightarrow d_n$, and only $d_{(n-1)} \longrightarrow d_n$ is a signing path component in a path p, $d_0 \models \phi_0$, and $d_n \models \phi_1$.
- Otherwise, $p \models \phi$ iff p is a path of the form $d_0 \longrightarrow \cdots$, and $d_0 \models \phi$.

We denote by $d \models P$ that "a document d is evidence to a policy P."

$\dot{<}$ is used typically in the case below:

Property 1 Let $d \equiv D[\cdots, f_i = t, \cdots] : (\cdots, f_i = t, \cdots)$ signed by p. Then,

$$d \models F_i(t) < \text{Signed}(p).$$

The proof is clear. From the node $[D[\cdots, f_i = t, \cdots] : f_i = t]$, we have one and the only one path to $[D[\cdots, f_i = t, \cdots] : (\cdots, f_i = t, \cdots)$ signed by $p]$.

Property 2 If d contains $f_i = t$ or Signed(p) as a component of d, we have $d \models \overleftarrow{E}\,(F_i(t))$ or $d \models \overleftarrow{E}$ Signed(p), respectively.

This says that in the document tree associated with d, there is a path from each document component to d, the whole document.

Property 3 If a document d has d' as d's component, then for all ϕ, if $d' \models \phi$, then $d \models \overleftarrow{E}\,\phi$.

3.4 Institutional Support

Our goal is to use documents in authorization in an institution. A document must be associated with its form because a policy in an institution must be associated with a document form. Who must sign a document, what must be filled in the form, and who must approve the document are all specified as a policy of an institution, and represented in a document form.

Specifically, associated with a document form D, there is a set of axioms that validates an authority claimed by a document of the form. This represents a policy related to the document form.

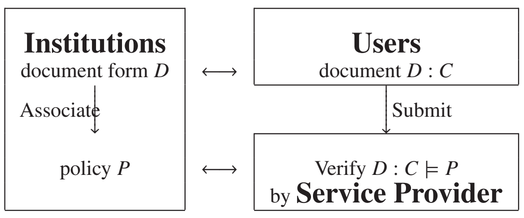

Fig. 3 Policies, documents and authorization.

We also denote it by D.

Figure 3 depicts this relation among policies, documents and authorization.

4. Applications and Analysis

4.1 Request and Grant

By using DTL, we can represent significant part of business workflow, including the cycle of request, approval, and execution. First, we show this simple scenario. In this scenario, fund managers can concentrate on grant. They do not need to interact with backyard servers. The interactions are indirectly performed by using the documents that the fund managers issue to an applicant.

Example 1 In this example, DTL is augmented with ordinary arithmetic.

Let document forms

`sbudget[applicant, budget, applyamount, offeramount]`

and

`lbudget[applicant, budget, applyamount, offeramount]`

be given. They are intended so that an `applicant` applies for budget `budget` with amount `applyamount`. An applicant expects that this document is approved by an authority with the amount `offeramount`. An applicant has right to withdraw `offeramount` if the document is appropriately filled with content, and the application with the document is approved. Within an institution, this form is used to process the application for a fund. The logic for processing this form is defined as the policy that claims that if the application is for Type S, and the amount of offer is less than \$1K, Bob can approve this application. Otherwise, it needs approval by a fund manager and finally by President. **Figure 4** illustrates this scenario. Here, the role of backyard is minimized, corresponding to Fig. 2. This policy is represented as

$$\forall X.\forall Y.\forall Z.\forall W.\text{FUND}(X, Y, Z, W),$$

where

$\text{FUND}(apply, applicant, budget, applyamount, offeramount) \equiv$
$\text{SBudget}(applicant, budget, applyamount, offeramount) \vee$
$\text{LBudget}(applicant, budget, applyamount, offeramount)$
$\longrightarrow \text{PermitWithDraw}(applicant, offeramount)$,

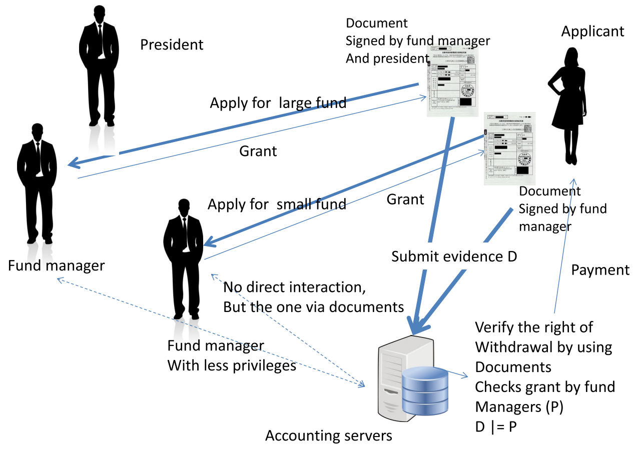

Fig. 4 Document as evidence of apply-grant – fund management.

SBudget(*applicant*, *budget*, *applyamount*, *offeramount*) ≡
Applicant(*applicant*) ∧ Budget(Type_S) ∧
$\overleftarrow{E}$ Applyamount(*applyamount*) ∧ applyamount < \$1K ∧
$\overleftarrow{E}$ Offeramount(*offeramount*) ∧
offeramount ≤ *applyamount* ∧ *offeramount* < \$1K ∧
$\overleftarrow{E}$ Signed(Bob) ∧
$\overleftarrow{E}$ (Offeramount(*offeramount*) $\dot{<}$ Signed(Bob)),
and
LBudget(*applicant*, *budget*, *applyamount*, *offeramount*) ≡
$\exists\, W_p$. (Applicant(*applicant*) ∧ ¬Budget(Type_S) ∧
$\overleftarrow{E}$ Applyamount(*applyamount*) ∧
$\overleftarrow{E}$ Offeramount(*offeramount*) ∧
$\overleftarrow{E}$ Signed(W_p) ∧ $\overleftarrow{E}$Appoint(W_p, FundManager) ∧
$\overleftarrow{E}$ (Offeramount(*offeramount*) $\dot{<}$ Signed(W_p)) ∧
$\overleftarrow{E}$ (Offeramount(*offeramount*) < Signed(President)).

Here, PermitWithDraw(Y, Z) is a primitive formula that represents the grant that Y can withdraw the amount of Z from the accounting server. If the amount of budget is large, the policy requires the signatures of the fund manager and the president. Furthermore, the fund manager must be appointed as a fund manager (represented by Appoint(W_p, FundManager)).
In the institution, let us suppose that the policy
Appoint(*person*, *role*) for processing the content in document form `appoint[person, role]` is defined as:

Appoint(*person*, *role*) ≡
$\overleftarrow{E}$ (Role(*role*) < Signed(President))∧
$\overleftarrow{E}$ (Person(*person*) $\dot{<}$ Signed(President)).

Let us suppose that someone submits a document below:
`sbudget[applicant, budget, applyamount,`
`offeramount]:`
`(applicant = Alice, budget = Type`$_S$`,`
`applyamount = $500 signed by Alice).`

Then, Bob returns it with his signature as document DS ≡
`sbudget[applicant, budget, applyamount,`
`offeramount]:`
`((applicant = Alice, budget = Type`$_S$`,`
`applyamount = $500 signed by Alice),`
`offeramount = $300) signed by Bob,`

When one submits document DS, then with simple inference, we have

$$DS \models \text{PermitWithDraw(Alice, \$300)},$$

and Alice acquires the right to withdraw \$300. Note that the document DS faithfully embeds the result of the workflow meaning that Alice applies to a fund, and Bob approves the application. In an institution, document forms are fixed for use in workflow. If a form is filled in a valid way, the document is considered to be evidence of the grant in the workflow. In this meaning, a set of document forms and their associated policies is essential in a business workflow of an institution.

Next, let us consider the document DL below that validates larger fund. This time, Alice is granted for a budget of Type_L. In this case, Alice attaches the additional document that Charlie, the signer is appointed to the FundManager.
`lbudget[applicant, budget,applyamount,offeramount]:`
`(((applicant = Alice, budget = Type`$_L$`,`
`applyamount = $1000 signed by Alice),`
`offeramount = $800) signed by Charlie`
`signed by President,`
`appoint[person,role]:`
`(person=Charlie, role=FundManager)`
`signed by President).`

With simple inference, we have
`appoint[person,role]:(person=Charlie,`
`role=FundManager) signed by President)`
⊨ Appoint(Charlie, FundManager),

and hence $\overleftarrow{E}$ Appoint(Charlie, FundManager) in `lbudget`, and hence LBudget(Alice, Type_L, \$1000, \$800). Therefore, DL ⊨ PermitWithDraw(Alice, \$800), and Alice has the right to withdraw \$800.

In **Fig. 5**, we show the document tree of DL and the process of inferring PermitWithDraw(Alice, \$800).

4.2 Delegation

We consider delegation of privileges of user A to user B.

Example 2 Suppose that Charlie likes to delegate his BoardMember role to access a document d. Let us suppose that Charlie already has a document $D1$ ≡
`appoint[appointee, role]:`
`(appointee=Charlie, role=BoardMember)`
`signed by President`
Charlie issues a document $D2$ ≡
`delegate[delegater, delegatee, role]:`
`(delegater=Charlie,delegatee=Dave,role=BoardMember)`
`signed by Charlie,`
Figure 6 illustrates this scenario. The document server grants access to Dave by verifying the evidence submitted by Dave.
Suppose that the institution has already defined the policy on delegation as
Delegate(*delegater*, *delegatee*, *role*) ≡
$\overleftarrow{E}$ Appoint(*delegater*, *role*) ∧
Role(*delegater*, *role*) ∧
$\overleftarrow{E}$(Delegatee(*delegatee*) $\dot{<}$ Signed(*delegater*)) ∧
$\overleftarrow{E}$(Delegatee(*role*) $\dot{<}$ Signed(*delegater*))
⟶ Role(*delegatee*, *role*).
Let us suppose that the institution allows access to a document d from a person that has role BoardMember, which is represented as the policy:

$$\forall X_p.(\text{Role}(X_p, \text{BoardMember}) \longrightarrow \text{AllowReadAccess}(X_p, d)).$$

To claim the privilege AllowReadAccess, document $D2$ is not enough. By attaching $D1$ to $D2$, that is, by using the document $D3$ ≡
`delegate[delegater, delegatee, role]:`
`((delegater=Charlie, delegatee=Dave,`
`role=BoardMember)`
`signed by Charlie, D1),`
with simple inference, $D3 \models \text{Role(Dave, BoardMember)}$ and

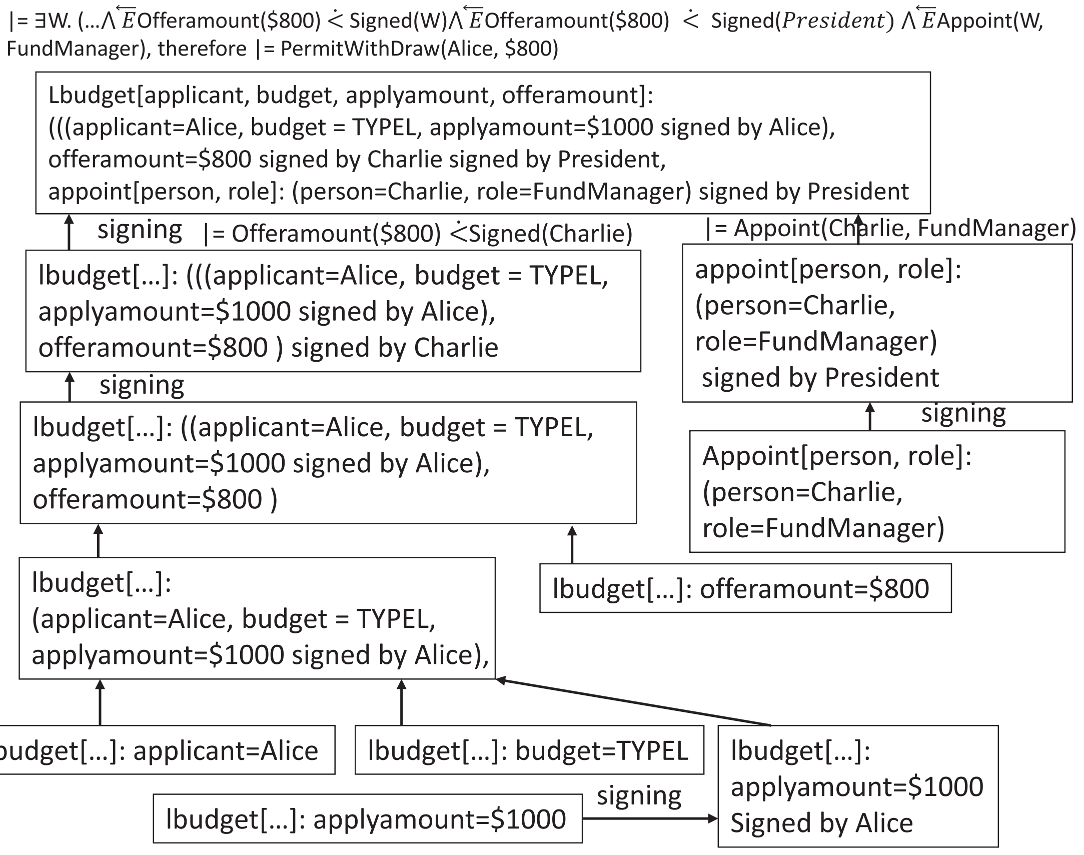
|= ∃W. (…∧ E Offeramount($800) ≺ Signed(W)∧ E Offeramount($800) ≺ Signed(President) ∧ E Appoint(W, FundManager), therefore |= PermitWithDraw(Alice, $800)
Lbudget[applicant, budget, applyamount, offeramount]: (((applicant=Alice, budget = TYPEL, applyamount=$1000 signed by Alice), offeramount=$800 signed by Charlie signed by President, appoint[person, role]: (person=Charlie, role=FundManager) signed by President
signing
|= Offeramount($800) ≺Signed(Charlie)
|= Appoint(Charlie, FundManager)
lbudget[…]: (((applicant=Alice, budget = TYPEL, applyamount=$1000 signed by Alice), offeramount=$800) signed by Charlie
appoint[person, role]: (person=Charlie, role=FundManager) signed by President
signing
signing
lbudget[…]: ((applicant=Alice, budget = TYPEL, applyamount=$1000 signed by Alice), offeramount=$800)
Appoint[person, role]: (person=Charlie, role=FundManager)
lbudget[…]: (applicant=Alice, budget = TYPEL, applyamount=$1000 signed by Alice),
lbudget[…]: offeramount=$800
lbudget[…]: applicant=Alice
lbudget[…]: budget=TYPEL
lbudget[…]: applyamount=$1000 Signed by Alice
signing
lbudget[…]: applyamount=$1000

Fig. 5 Inference of PermitWithDraw with document *DL*.

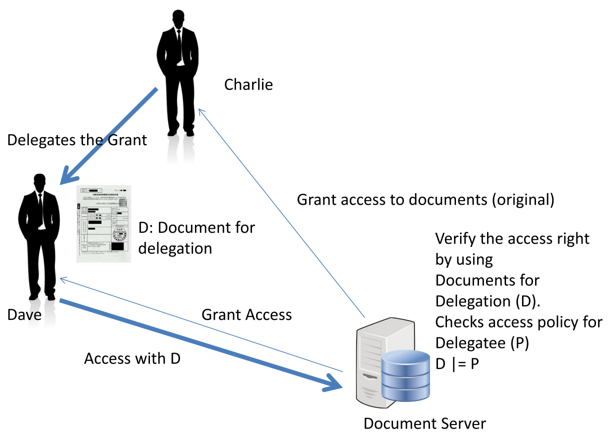
Charlie
Delegates the Grant
D: Document for delegation
Dave
Grant access to documents (original)
Verify the access right by using Documents for Delegation (D). Checks access policy for Delegatee (P) D |= P
Grant Access
Access with D
Document Server

Fig. 6 Workflow of delegation from Charlie to Dave.

hence $D3 \models$ AllowReadAccess(Dave, d).

Note that the delegation by using attribute certificates [12] can also be represented by this scheme. The privileges are delegated by the staff that the privileges are originally given to without centralized control. An institution can control delegation only by specifying the roles that can be delegated. The idea is very similar to issuing access tokens in OAuth(2), where once issued, an access token works as a delegation of privilege.

The point here is that an applicant collects evidence documents for one's authorization. The documents reflect the business process of request, approval and grant. The service side validates the evidence, and infers the privilege by using the service policies.

4.3 Sequencing of Signatures in Business Processes

In the definition of business processes, sequencing of transactions is one of the most significant processes.

Table 1 Activities of WS-BPEL.

Basics	`<invoke>`, `<assign>`, `<empty>`, `<sequence>`, `<pick>`, `<extensionActivity>`
Message Exchange	`<receive>`, `<reply>`
Flow Control	`<throw>`, `<exit>`, `<if>`, `<while>`, `<repeatUntil>`, `<forEach>`, `<flow>`, `<rethrow>`, `<wait>`
Others	`<scope>`, `<compensate>`, `<compensateScope>`, `<validate>`

There have been proposed a number of schemes for specifying business processes. Among those, we use the one used in WS-BPEL [23].

In WS-BPEL, three types of interactions between two processes are defined: sequencing, concurrent sequence, and synchronization across concurrent processes. WS-BPEL defines a business process as UML-like graph representation of component processes and message exchange between processes. The flow (`<flow>`) is defined, within which processes are invoked (`<invoke>`) and sequenced (`<sequence>` and `<wait>`). Flow control by using `<if>`, `<while>`, `<forEach>` is enabled. The message exchange is modeled as receive (`<receive>`) and reply(`<reply>`). We list in **Table 1** the collection of activities of processes in WS-BPEL.

In our model, document trees and DTL express this interactions as the sequencing of signatures and inclusion of documents. First, let us suppose that a signature is given when a process is completed.

sequencing The requirement that a process A with signature S_A must be followed by B with signature S_B is represented as

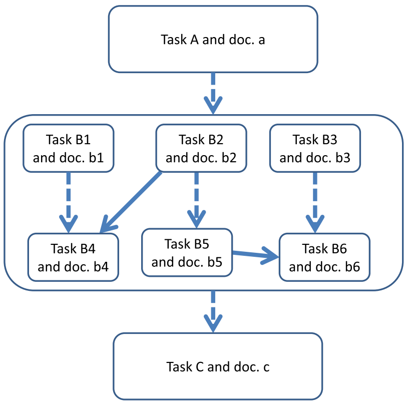

Fig. 7 A simple business process workflow in Ref. [23].

Signed(S_A) < Signed(S_B) in DTL. Documents satisfying this requirement show the sequencing of signatures from S_A to S_B.

concurrent sequencing There is no requirement in DTL. Documents satisfying this requirement show that all the subdocuments representing the target activities are included in any order.

synchronization The requirement that a process A with signature S_A must synchronize with a process B with signature S_B can be represented as the sequencing of signatures from S_A to S_B, that is Signed(S_A) < Signed(S_B). Moreover, the documents must be formed so that A is included in B. If a policy A is associated with a process A, this is represented as $\overleftarrow{E}\,(A < \text{Signed}(S_B) \wedge \text{Signed}(S_A) < \text{Signed}(S_B))$.

Note that the requirement on order of processes is expressed as a partial order. Because signatures in a document trees form a partial order, we can flexibly express the sequencing of business processes.

Example 3 Let us consider the case of **Fig. 7** that appears in the first example of a business process in Ref. [23]. When we suppose when each task x is completed, a document `x` is produced and signed by `X`. Both the documents $D1 \equiv$

```
((((a signed by A, ((b1 signed by B1, b2 signed
by B2, b4) signed by B4, b5 signed by B5), b3
signed by B3), b6 signed by B6))signed by B, c)
signed by C
```

and $D2 \equiv$

```
((((a signed by A, ((b1 signed by B1, b2 signed
by B2, b4) signed by B4, (b3 signed by B3, b5
signed by B5), b6 signed by B6))))signed by B, c)
signed by C
```

can be used as evidence for the formula that the task is completed according to Fig. 7.

In this case, the sequence of signatures represents the order of tasks. Furthermore, the concurrent sequencing allows freedom on topological sorting of the order of documents.

Example 4 Let us consider approval by document circular. The requirement is that all the related staff approves a given document. In the approval, the order of signatures does not matter. This process is expressed by using the partial order of signatures in DTL.

Let us assume that the staff at concern is specified by *Staff*. Then, the requirement that approval by document circular is successful for the policy D is expressed as DocCirc_D:

$\text{DocCirc}_D(\text{Staff}) \equiv$

$\overleftarrow{E}\,D \wedge \forall X \in \text{Staff}.\ \overleftarrow{E}\,(D < \text{Signed}(X))$,

meaning that there is a node (=component of document) that represents the policy D, and all the staff members gives their signature for the document. The order of signing does not matter here.

Let us suppose *Staff* = {A, B, C}. Because the order of signature does not matter, we see that for a document `d` for D, both `d1` ≡ `d signed by A signed by B signed by C` and `d2` ≡ `d signed by A signed by C signed by B` satisfy {d1 | d2} $\models \overleftarrow{E}\,D \wedge \forall X \in \text{Staff}.\ \overleftarrow{E}\,D < \text{Signed}(X)$.

4.4 Similarity to Blockchain

Although our scheme assumes institutions' predefined policies for processing documents, a transaction can be processed only by a claimant and server because the claimant submits evidence (document) to claim one's privilege, and the server has only to validate the submitted evidence, and does not have to wait for provisioning by a higher section. In this meaning, the transaction is decentralized within an institution. In the evidence, all the related transactions are recorded to be verified.

Blockchain [18] is another example of decentralized database. It also records all the transactions in the past so that a given chain of blocks is validated. The verification process is decentralized, and in this sense, similar to our scheme.

4.5 Authentication and Authorization

The model of submitting and verifying evidence, and giving grant works effectively in a variety of service frameworks. In general, servers provide service to a user with the verification of the claimant (authentication) and the verification of the privilege (authorization). This model is a solution to the authorization problem. The authentication must be accompanied to complete the service providing. The session of the authentication process must be managed separately by service providers.

This is similar to the motivation of the proposal of OpenID Connect [26] along with OAuth2[*1].

5. Related Work

There are several approaches in the type of logics. Modal logic and its variants are used to express the requirements of authentication and authorization [2], [10], [16]. Use of other type of logic includes linear logic (logic for resources) [9], [20], executable Horn-like logics [25], [32], [33], explicit time [6], and multi-valued logic [4].

This work is classified as a logical approach to access control [1], [5], [11], [17], [27]. In a logical approach, evidence is carried along with a claim on authorization. They carry proofs of a given claim as evidence, while our work carries documents as evidence. Receivers of evidence verify its correctness, and grants privileges according to a predefined policy.

[*1] http://www.thread-safe.com/2012/01/problem-with-oauth-for-authentication.html

There have been proposed several practical approaches for managing policies.

Publication of policy documents plays a central role in establishing trust in the Internet services. Its typical scenarios include trust frameworks. In a trust framework, service policies are defined, and parties participate in the trust framework by agreeing with the predefined policies. Control of identity trust frameworks by policy documents is proposed and documented [21]. Standards such as ISO 29115 and NIST SP 800-63-2 have also been defined for control of identity trust framework. Today, we can see several deployment of trust frameworks. US FICAM provides trust framework provider adoption process (TFPAP) [8] for a trust framework provider to operate a trust framework under the policies of US FICAM. GakuNin (http://www.gakunin.jp) is a Japanese nation-wide academic trust framework provider, partly participating in US FICAM via Kantara (https://kantarainitiative.org/). Furthermore, a number of nation-wide academic federations are operated as trust frameworks. Furthermore, the trust frameworks for IoT is studied [28] and practically discussed [24] in addition to the one for identities.

Deployment of policies on servers is another concern. In cloud environments, it is commonly seen that multiple stakeholders are enrolled in services. In such environments, provisioning is one of the most troublesome tasks in harmonization among stakeholders. Recent work concentrates on the incorporation among service providers [3], [7], [19]. Moreover, SCIM [15] has been specified as a standard of provisioning.

Another scenario is the CP/CPS publication in PKI. The template of CP/CPS has been defined in Ref. [13]. Privacy related policies are also important to make decisions on access to Web sites. P3P is defined in Ref. [31] to allow Web browsers to assess the policies of Web sites. What is significant is that the privacy policies are given in the form of XML so that its evaluation by a program is made possible. The approach of Ref. [29] generalizes this idea. Furthermore, the XML template of RFC 3647 is designed in Ref. [30].

There are a number of proposals for specifying business processes. WS-BPEL [23] gives a UML-like definition of business processes. It has some extensions such as BPEL4People to express real business scenarios.

OAuth 2.0 [14] is a protocol for delegating authorization. In this framework, once a client acquires an access token, it can access a specified resource of the resource server within the scope of the access token. Protocols used in access federations are intended so that multiple servers incorporate to complete a specific service such as authentication and authorization. OAuth is widely adopted by major companies such as Facebook (https://developers.facebook.com/docs/facebook-login/access-tokens), Twitter (https://dev.twitter.com/oauth), MS Azure (https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-protocols-oauth-code) and Google (https://developers.google.com/identity/protocols/OAuth2). SAML [22] and OpenID-Connect [26] are designed for authentication, while OAuth is designed for authorization.

Blockchain [18] is based on the idea that users can verify given data if the data contains the whole transaction trail with tamper-proof techniques such as digital signatures. There are proposed and implemented augmented logics on blockchain such as colordcoins (http://coloredcoins.org/).

6. Concluding Remarks

In this paper, we have proposed a new scheme "Authorization by Document." We have first analyzed the roles of documents. Next, we have given the formal definitions of documents and DTL. DTL is a simplified CTL* originally used in the verification of programs. In DTL, documents are used as evidence to a given policy written in DTL. We have shown their applications to typical business processes such as apply and grant, delegate, and approval by circular, and shown that they can be applied to wide variety of scenarios.

As the process of authorization grows in its complexity, we need a scheme that can represent such complexity. We have adopted the logic for program verification. This shows that our approach is extensible to more complicated processes. Extending our scheme to a wider scenario can be considered. In this paper, we assume that policies are managed and controlled by an institution that users belong to. If the policies are managed between institutions or managed as a contract between an institution and a consumer, scenarios in B2B or B2C are also feasible.

References

[1] Appel, A. and Felien, E.: Proof-carrying authentication, *Proc. 6th Int'l Conf. Computer and Communications Security*, pp.52–62 (1999).

[2] Bai, Y.: A modal logic for authorization specification and reasoning, *Proc. 2009 Int'l Conf. Intelligent Computing and Intelligent Systems*, Vol.1, pp.264–268 (2009).

[3] Bousquet, A., Briffaut, J., Caron, E., Dominguez, E., Franco, J., Lefray, A., Lopez, O., Ros, S., Rouzaud-Cornabas, J., Toinard, C. and Uriarte, M.: Enforcing Security and Assurance Properties in Cloud Environment, *Proc. Utility and Cloud Computing* (*UCC*), pp.271–280 (2015).

[4] Bruns, G. and Huth, M.: Access-Control Policies via Belnap Logic: Effective and Efficient Composition and Analysis, *Proc. 21st Computer Security Foundations Symposium*, pp.163–176 (2008).

[5] Chin, S.-K. and Older, S.: *Access Control, Security, and Trust: A Logical Approach*, CRC Press (2010).

[6] DeYoun, H., Garg, D. and Pfenning, F.: An Authorization Logic With Explicit Time, *Proc. 21st Computer Security Foundations Symposium*, pp.133–145 (2008).

[7] Ertl, B., Stevanovic, U., Hayrapetyan, A., Wegh, B. and Hardt, M.: Identity harmonization for federated HPC, grid and cloud services, *Proc. High Performance Computing & Simulation* (*HPCS*), pp.621–628 (2016).

[8] FICAM, Trust Framework Provider Adoption Process (TFPAP) for Levels of Assurance 1,2,3 and 4, V1.1.0 (2013).

[9] Garg, D., Bauer, L., Bowers, K.D., Pfenning, F. and Reiter, M.K.: A linear logic of authorization and knowledge, *Proc. 2006 European Symp. Research in Computer Security* (*ESORICS*), pp.297–312 (2006).

[10] Genovese, V., Rispoli, D., Gabbay, D.M. and van der Torre, L.: Modal Access Control Logic – Axiomatization, Semantics and FOL Theorem Proving, *Proc. 5th Starting AI Researchers' Symposium*, pp.114–126 (2010).

[11] Hirsch, A. and Clarkson, M.: Belief semantics of authorization logic, *Proc. 2013 Int'l Conf. Computer and Communications Security*, pp.561–572 (2013).

[12] IETF, An Internet Attribute Certificate Profile for Authorization, RFC 3281 (2002).

[13] IETF, Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework, RFC 3647 (2003).

[14] IETF, The OAuth 2.0 Authorization Framework, RFC 6749 (2012).

[15] IETF, System for Cross-domain Identity Management: Protocol, RFC 7644 (2015).

[16] Kosiyatrakul, T., Older, S. and Chin, S.-K.: A modal logic for role-based access control, *Proc. 2005 Mathematical Methods, Models, and*

Architectures for Computer Network Security (*MMM-ACNS*) (LNCS 3685), pp.179–193 (2005).

[17] Lesniewski-Laas, C., Ford, B., Strauss, J., Morris, R. and Kaashoek, M.F.: Alpaca: Extensible Authorization for Distributed Services, *Proc. 14th Conf. Computer and Communications Security*, pp.432–444 (2007).

[18] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008), available from ⟨https://bitcon.org/bitcon.pdf⟩.

[19] Ngo, C., Membrey, P., Demchenko, Y. and de Laat, C.: Policy and Context Management in Dynamically Provisioned Access Control Service for Virtualized Cloud Infrastructures, *Proc. Availability, Reliability and Security* (*ARES*), pp.343–349 (2012).

[20] Nigam, V.: On the Complexity of Linear Authorization Logics, *Proc. 27th Symposium on Logic in Computer Science*, pp.511–520 (2012).

[21] NIST, Developing Trust Frameworks to Support Identity Federations, NISTIR 8149 (draft) (2016).

[22] OASIS, Security Assertion Markup Language (SAML) V2.0 (2005).

[23] OASIS, Web Services Business Process Execution Language, Version 2.0 (2007).

[24] Online Trust Alliance: IoT Trust Framework (2017).

[25] Ruan, C. and Shahrestani, S.: Logic Based Authorization Program and Its Implementation, *Proc. 4th Int'l Conf. Security of Information and networks*, pp.87–94 (2011).

[26] Sakimura, N., Bradley, J., Jones, M.B., de Medeiros, B. and Mortimore, C.: OpenID Foundation: OpenID Connect Core 1.0 (2014).

[27] Sato, H.: Analyzing Semantics of Documents by Using a Program Analysis Method, *Proc. 2009 33rd Annual IEEE International Computer Software and Applications Conference* (*COMPSAC '09*), Vol.1, pp.373–382 (2009).

[28] Sato, H., Kanai, A., Tanimoto, S. and Kobaysshi, T.: Establishing Trust in the Emerging Era of IoT, *Proc. IEEE Int'l Conf. Service-Oriented System Engineering 2016*, pp.398–406 (2016).

[29] Sato, H., Tanimoto, S. and Kanai, A.: A Policy Consumption Architecture that enables Dynamic and Fine Policy Management, *Proc. 3rd ASE International Conf. CyberSecurity* (2014).

[30] Sato, H., Tanimoto, S. and Kanai, A.: XMLed Service Policy Documents for Access Control, Computer Security Symposium 2014, 1D4-1 (2014). (in Japanese).

[31] W3C, The Platform for Privacy Preferences 1.1 Specification (2006).

[32] Zhang, M., Chen, W., Wang, Y. and Zhang, M.: Flexible Authorizations with Logic Program, *Proc. 2009 Int'l Conf. Networks Security, Wireless Communications and Trusted Computing*, Vol.2, 259–262 (2009).

[33] Zhang, M., Wang, Y. and Ma, X.: Specifying Flexible Features in Authorization Using Logic Program, *Proc. Seconf WS on Education Technology and Computer Science*, Vol.1, pp.578–581 (2010).

Hiroyuki Sato Associate Professor in the University of Tokyo. Received B.Sc., M.Sc. and Ph.D. from the University of Tokyo in 1985, 1987, 1990, respectively. Majoring: Computer Science and Information Security. He is a member of ACM, IEEE, IPSJ and JSSST.