A Framework of Trust and Authorization for ECommerce Service.

George S Oreku
[1]Department of Computer Science and Engineering
Harbin Institute of Technology, Foreign Student Building
A13 Room 601, P.O.Box 773, 92 Xi Dazhi Street,
Nangang District, Harbin 150001 China
gsoreku@yahoo.com

Jianzhong Li
Harbin Institute of Technology, Heilongjiang Province, China
lijzh@hit.edu.cn

Fredrick J. Mtenzi
Dublin Institute of Technology, Dublin 8, Ireland
fred.mtenzi@dit.ie

Abstract

Changes in consumer behavior, advances in broadband technology, the industry's maturity and entropy are converging to usher in a new era of eCommerce security. In particular, traditional access control a.k.a authorization for eCommerce systems is not flexible and efficient enough to combat these challenges i.e. more interactive user experience delivered through rich internet applications e.g. supply chain, disaggregated service that provide only a portion of the ecommerce experience and so on and public remains concerned about the security of online transactions.[14]. Framework proposed provides trust transformation rules which have associated conditions for authorizations to control access.

Our proposed framework aims to build the architecture for trust and authorization within an eCommerce service system. The architecture will help to build a secure and privacy protection eCommerce service system. The underlying framework will not only inform researchers of a better design for secure eCommerce service, but also assist eCommerce systems developers in the understanding of intricate constructions within trust and authorization. This includes protecting the privacy of transactions records of customers in terms of information privacy and access.

Key words: Authorization, Access control logic, eCommerce services, Intermediates
General terms: Security, Design.

1. Introduction

Trusted computing means predictability the systems should operate as the designer or manufacturers, software providers, and users intended which is essentials for both security and privacy. In a proposed framework trust is all about access control; you preserve trust by denying access to operations that might violet it. Congruent with formal security models, our framework applies intermediate as man in the middle analysis and seeks to deny the snooper access to eCommerce system operations by taking steps to achieve *Confidentiality , integrity and authorizations* .Confidentiality means snoopers can't observe the intended operations or at least some critical aspects of it. Integrity means that they can't perhaps even blindly, alter the intended operation without being detected. Finally, authorizations means snoopers can't successfully represent themselves as legitimate parties to the peripheral communication because of authorizations checks. In this frame work, successful

attacks against trusted computing arise because confidentiality, integrity or authorizations fails. This is true even with so called "social engineering" or "insider" attacks because legitimate user can blame an important specification or security policy about who has authorization.

The proposed architecture for trust and authorizations for eCommerce services system is a practical architecture presented *figure (1)* and new solution for access control within eCommerce service systems since presently there is no such solution that focuses on the authorization and trust simultaneously and sequentially for the access control within eCommerce service system as one of the applications in real-world scenario. Our proposed frame work suggests that it's efficient to put a versatile access control system into every intermediate.

Our approach of Trust Access Control intermediates for Authorization to eCommerce service application is an idea that has been explored in other settings namely previously authorization frame work based ABLP logic. Most closely related in this regard is proof carrying authorizations (PCA)[15,16] a framework for specifying and enforcing webpage access policies (though the logic used there is not ABLP, but an application specific variant).However that system comprise a general framework for webpage access control so expressible policies are potentially more complicated than approach we propose for eCommerce service. Other authorizations system founded in ABLP logic include that used in the Taos operating system [17], essentially a direct implementations of a subset of ABLP logic. Also Wallah et al have formalized the "security- passing style" of the java stack inspections mechanism in a subset of ABLP [18,7] which has served as a foundation for the SAFKASI programming language-based security architecture[19] The SDSI/SPKI architecture [20,21] is another authorization system for distributed communication. Their security model is similar to ABLP, but is based on a system of local names and emphasizes delegation. Delegation logic [22.] and RT [23] are more recently proposed, logically well-founded authorization frameworks, based on data log. No formulations of SDSI/SPKI, delegation logic, or RT currently comprise trust access control intermediates for authorizations In [24], a web services authorization system is defined, allowing specification of security policies in temporal logic, which are translated into reference monitors embedded in applications software. However, their approach is focused on complex policies for usage patterns similar to [25], and they make no online/offline checking phase distinction.

We establish a formal setting for our eCommerce service authorization framework in the Calculus of Access Control [9]. Proposed architecture for authorizations on eCommerce environment involves six intermediates (Card holder, Issuer, Merchants, Acquirer, Payment gateway, Certificates Authority) as shown in figure 1. Figure 1 takes our framework and shows how we can implement access control for any intermediate. The use of access control logic for framework specification is fundamental to our proposal, and we argue that this approach promotes unambiguous specification languages, reliability, and verifiability. The contribution of this paper is to introduce the trust intermediates framework for authorizations (access control) in eCommerce service environments, and formalize the framework conditions within logical perspective. By characterizing the intermediates within authorization logic, we also propose rigorous logical foundation for authorization in eCommerce service environments.

2. Access Control Logic

Our proposed architecture consists of six participants: the ways users gain their security credentials; and how these credentials are used to access eCommerce data. Three types of digital certificates are used: identity certificates for authentication; attribute certificates for authorization; and access-rule certificates for propagation of

access control. Once a user is identified and authorized, subsequent access decisions are based on trust transformations rules which convert complicated access pattern to simpler ones. In our implementation approach we propose the use of digital certificates, SSL, SET to be used in conjunction with access control intermediates. We place an access control on every intermediate using simple logic argument and we argue that if can be successfully and securely managed can be used to control access in eCommerce applications.

We will model the intermediates using a logical language ABLP [3, 11, 2]. Logic concept allows us to reason about what is to be true given the state of eCommerce environment and a set of axioms. It has been used to describe authentication and authorization in distributed systems such as Taos [3] and appears to be a good match for describing access control within eCommerce transactions. We use a subset of the full [11] logic, which we will describe in sections 3. Readers who want a full description and a more formal development of the logic should see [3, 2] or appendix. The logic is based on a few simple concepts: principals, conjunctions of principals, targets, statements, quotation, and authority.

Example 1: Trust and Authorization Procedure within eCommerce Service System

Consider there are cardholder, *Merchant*, Issuer, Acquirer, and a number of clients, who will first register with an eCommerce service system. Suppose the *Merchant* wants to access an electronic eCommerce record within the eCommerce service system.

(1) *Merchant* will send an access request by access request agent.
(2) The access request agent then sends the access request token to the administrative agent.
(3) The administrative agent then refers to the authorization trusted intermediate of the system and then sends the preliminary authorization reference for the *Merchant*.
(4) The *Merchant* will be authenticated based on the biometric information and the digital PIN. If authentication is successful, the *Merchant* will be granted a formal authorization reference.
(5) By the formal authorization reference, the *Merchant* will be verified whether he has the privilege of reading/writing into the eCommerce service system.
(6) By the role-and-privilege based authorization intermediates, if the *Merchant's* status satisfies the requirements of the syntax principals, he will granted to access to the targeted client electronic eCommerce record i.e. bank accounts

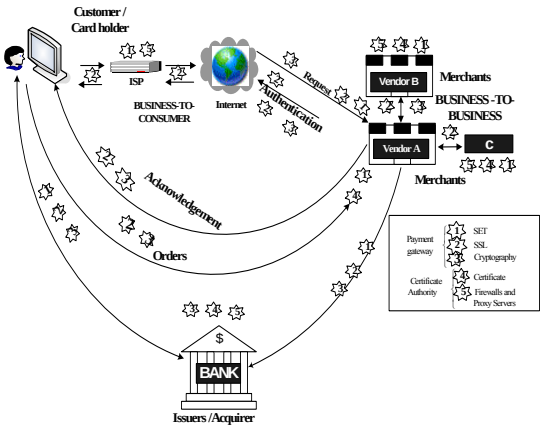

Figure 1. Architecture for Trust and Authorization within eCommerce service system

3. Syntax of Principals and Formulae

The syntax of ABLP principals, constituting identities in distributed communications, and formulae, representing statements and beliefs, is defined in equations. 1. Regarding principles, we will mostly be concerned with atomic principles A and principles $P \mid Q$, pronounced "P quoting Q". Statements P *says* s generally represent that an assertion s has originated with a principal P. The relation $P \Rightarrow Q$, pronounced "P speaks for Q" denotes that statements uttered by P can also be attributed to Q; this is clarified in Section 3.2, which describes the derivation rules for the authorizations.

$A, B, C, \ldots \in Atom$ *atomic principals*

$P ::= \quad A \mid P \wedge P \mid \vee P \mid \vee P \mid P \mid P$

principals

$p \in Prop$ *primitive propositions*

$s ::= p \mid \neg s \mid s \wedge s \mid p \Rightarrow p \mid p \text{ says } s$

formulae

Equations 1: ABLP Syntax

3.1 Abbreviations and Notational Conventions

We assume that all principals can perform digital signatures. We let *K* range over public keys as principals, and always write K_A to denote the public keys of *A*, and $K^{-1}{}_A$ the matching private key. The formula *K says s* represents the formula *s* encrypted under *K*. A number of useful abbreviations are defined in Equations 2.and appendix along with macros for standard logical connectives, these include *A as R*, denoting the principal obtained when *A* assumes the role *R*, and *A controls s*, denoting that *A* is directly authorized for *s*. for a complete explanation of these abbreviations see [2]

$$s_1 \vee s_2 \triangleq \neg (\neg s_1 \wedge \neg s_2) \quad .$$

$$s_1 \supset s_2 \triangleq \neg s_1 \vee (s_1 \wedge s_2)$$

$$A \text{ as } R \triangleq A \mid R \; A \text{ controls } s \triangleq (A \text{ says } S) \supset S$$

Equations 2: ABLP Formulae Abbreviations

3.2 Proof Theory

We write $\vdash s$ to denote that a formulae *s* is logically derivable, on the basis of the axioms and inference rules of the theory. A selection of the ABLP inference rules, connecting the calculus of principals to the underlying propositional logic [3], is adapted to this approach specifically those which will be relevant to our presentation. We note that the rule names given are of our own devise, for easy reference in the remaining presentation. Also for convenience, we write $s \vdash s'$ iff s' is derivable given assumption *s*. Here is an example showing how the logic can be used to model and reason about statements signed by digital signatures associated with particular principals.

Example: 2. We trust that private keys remain indeed private, so that messages signed with K_J *carry the authority of J:*

$$K_J \Rightarrow J$$

Thus, if any statement s is ever signed with Js private key:

$$K_J \text{ says } s$$

By rule Speaks for:

$$(K_J \Rightarrow J) \supset (K_J \text{ says } s \supset J \text{ says } s)$$

hence by two applications of valid argument form $\dfrac{\vdash s' \quad \vdash s' \supset s}{\vdash s}$ *we have:*

$$J \text{ says } s$$

That is, any signed message can be taken as a statement of the owner of the signature key.

4. Mapping Trust Authorizations to ABLP

Authorizations refer to enforcing access control to ensure *confidentiality* and *integrity*. One of the key issues in e-business is legitimate use. Legitimate use has two components: Identifications and authentication. Traditional access control mechanisms make authorization decisions based on the identity of the requester. However, in decentralized or multi centric environments i.e. eCommerce environment, the resource owner and the requester often are unknown to one another, and access control based on identity may be ineffective. Trust authorization proposed is an approach to distributed access control and authorization, in which access control decisions are based on *ABLP* made by multiple principals.

4.1 Principals

In eCommerce service authorizations certificates are digitally signed with a private key, and then shipped to the virtual machine where it will run. If K_{Signer} is the public key of *Signer*, the public-key infrastructure can generate a proof formula (3) of the statement

$$K_{Signer} \Rightarrow Signer. \qquad (1)$$

Signer's digital signature on the code *Code* is interpreted as K_{Signer} **says** $Code \Rightarrow K_{Signer}$ (2)

Using formula (1) and (26) appendix, this implies that $Code \Rightarrow Signer$ (3)

When *Code* is invoked, it generates an *entity Intermediate*.

The authorizations principals assumes that the entity speaks for the code it is accessing:

$Entity \Rightarrow Code$ (4)

The transitivity of $\Rightarrow$ (which can be derived from formula (25) appendix) then implies

$Entity \Rightarrow Signer.$ (5)

We define Φ to be the set of all such valid $Entity \Rightarrow Signer$ statements. We call Φ the *Entity credentials*.

Note also that code can be signed by more than one principal. In this case, the code and its entity intermediates speak for all of the signers. To simplify the discussion, all of our examples will use single signers, but the theory supports multiple signers without any extra difficulty.

4.2 Targets

Recall that the resources we wish to protect are called *targets*. For each target, we create a dummy principal whose name is identical to that of the target. These dummy principals do not make any statements themselves, but various principals may speak for them. For each target *T*, the statement *Ok*(*T*) means that access to *T* should be allowed in the present context.

The axiom:

$\forall\ T \in Targets, (\ T\ \textbf{says}\ Ok(T)) \supset Ok(T)$ (6)

says that *T* can allow access to itself.

Many targets are defined in relation to services offered by eCommerce service underlying the Secures electronic Transactions (SET). From the eCommerce's point of view, the SET is a single process and all system calls coming from the SET are performed under the authority of the SET's principal (often the user running the SET). The SET's responsibility, then, is to allow authorizations only when there is justification for issuing that system call under the SET's authority. Our proposal will support this intuition by requiring the SET to prove in ABLP logic that each request has been authorized by a suitable principal.

4.3 Access Control Lists

Access control lists (ACLs) are fundamental to access control systems on eCommerce, providing an explicit association of principals with the privileges for which they're authorized. Setting up a new access controls on intermediates is also a firmware resource itself and thus has access controls on it. In the original presentation of ABLP [2], ACLs are conjunctions of statements of the form *P controls s,* where *s* is some privilege. We adapt this approach, letting *A* range over ACLs. Furthermore, we designate a subset of *Prop* as the set of *privileges* in eCommerce environment, letting **priv** range over this set. Hence, ACLs are conjunctions of statements *P controls* **priv**. Our justification for designating atomic propositions as privileges, and our use thereof, is discussed in Section 5 and proofs for more detail.

4.5 Authorization Contexts and Decisions

E-commerce services authorization is based on requests for the service made by invokers. Here we describe our proposed structure for these requests, and for the authorization decision predicated on them. A request is an ABLP assertion *s* uttered by the invoker of an eCommerce service, which the invoker intends to be used in architecture presented in figure 1 for authorization of its use. In addition to the request, a web service may possess other facts and beliefs, e.g. ACLs and role certifications that affect the authorization judgment; we assume that these facts and beliefs are expressed as ABLP formulae. The conjunction of these components constitutes an authorization *context*; authorization for an eCommerce service is granted upon a particular invocation if the context of the invocation allows the privilege required for use of the eCommerce service to be derived in the ABLP proof theory.

*Example: 3 Suppose an eCommerce service eCS requires **priv** to be accessed, and a trusted principal D makes an access request:*

$$A \triangleq D \text{ controls } \mathbf{priv} \wedge A' \text{ ACL defined by eCS}$$

$$\frac{s \vdash \text{D says priv} \quad \vdash \text{D says priv} \supset \text{priv}}{s \vdash \text{priv}}$$

*Since **priv** is provable in this context, authorization succeeds.*

As mentioned in Section 3, we posit a set of atomic formulae **priv**, each of which represent the privilege required to access a particular eCommerce service. Authorization for **priv** in a context s is effected by checking validity of $s \vdash$ **priv** . Thus, our method is inspired by access control mechanisms such as stack inspection [4], which are specialized for program procedure calls, rather than challenge/response systems such as [5], which are adapted to human usage (i.e. web browsing) patterns. Our justification for this is that eCommerce service invocation bears a strong similarity to RPC/RMI, as observed in [9, 12], with chains of web service invocations resembling call stacks. Here is a brief example illustrating the concepts of the framework described thus far:

*Example: 4 Suppose some eCommerce service **eCS** requires the privilege **priv** to be used, and the web service **ACL** A grants this privilege to a principal **D**, i.e.* $A \equiv A' \wedge D$ *controls **priv** for some* A' *. Suppose also that **D** invokes **eCS** on its own behalf, making the request **D** says **priv**. Thus, the authorization context is **D** says* $\mathbf{priv} \wedge A$ *. Clearly, D says* $\mathbf{priv} \wedge A \vdash \mathbf{Priv}$ *, since the context implies both* $\vdash \mathbf{D}$ *says **priv** and* $\vdash \mathbf{D}$ *controls **priv**, which implies* $\vdash \mathbf{Priv}$ *by valid argument form* $\dfrac{\vdash s' \quad \vdash s' \supset s}{\vdash s}$

Naturally, it is desirable for authorization judgments to be decidable. Although ABLP logic is undecidable in general, various presentations have described non-trivial, decidable access control mechanisms [7, 11].

Condition 1. *Let s be an authorization context; then validity of* $s \vdash \mathbf{Priv}$ *is decidable.*

This condition requires any verifications implementation to provide a decision procedure for validity of authorization judgments, and also implicitly requires the form of authorization contexts to be well-defined. As with all the framework conditions, the formal statement of the condition allows correctness of a decision procedure to be provable, i.e. implementations can (and should) be accompanied with proofs of their adherence to framework conditions.

We assert that an authorization for trusted contexts is decidable (and efficient) satisfying conditions 1:

Lemma 1.let's be an extrapolated context; then for all **priv,** *validity of* $[\![s]\!] \vdash$ **priv** is decidable

Theorem 1 (Soundness) *If the decision procedure returns* true *when invoked in entity* F*, then there exists a proof in ABLP logic that* $E_F \supset Ok(T)$.

Proof: By assumption, there is a path connections $(A, v_1, v_2, ..., v_k, B)$ in the speaks-for graph of E_F. In order for this path connections to exist, we know that the statements $A \Rightarrow v_1$, $v_i \Rightarrow v_{i+1}$ for all $i \in [\![1, k-1]\!]$, and $v_k \Rightarrow B$ are all members of E_F. Since $\Rightarrow$ is transitive, this implies that $E_F \supset A \Rightarrow B$. where E_F is the entity environment

Proof of Theorem 1: There are two cases in which the decision procedure can return *true*.

1. The decision procedure returns *true* while it is iterating over the Class 1 statements (section 5.1). This occurs when the decision procedure finds the statement $Ok(T) \in E_F$. In this case, $Ok(T)$ follows trivially from E_F .

2. The decision procedure returns *true* while it is iterating over the Class 2

statements section 5.1. In this case we know that the decision procedure found a Class 2 statement section 5.1 of the form $P_1 \mid P_2 \mid \ldots \mid P_k$ ***says*** $Ok(T)$. where for all $i \in [\![1,k]\!]$ there is path connections from P_i to T in the speaks-for of E_F. It follows from Lemma 1 that for all $i \in [\![1,k]\!], P_i \Rightarrow T.$ It follows that $E_F \supset (T \mid T \mid \ldots \mid T$ ***says*** $Ok(T))$ (7) Applying formula (6) repeatedly, we can directly derive $E_F \supset Ok(T)$.

4.6 Trust Transformations

A trust transformation is a function from ABLP formulae to ABLP formulae. Any trust transformation's domain is formulae in *extrapolated* form, which take into account all components of access control, down to every detail verified during offline checking. The range of any trust transformation is formulae in *trusted* form, which are the "watered down" formulae that exclude restrictions the system takes for granted during online checking.

Example:5.Suppose that an eCommerce service eCS requires **priv** *to be accessed, and: Any access request must be accompanied by a signed certificate authenticating the request, ACL A defined by eCS comprises entries* $(P \wedge K_P)$ *controls* **priv***, For the sake of efficiency, signed certificates are not actively included (suppressed) in online authorization.*

$$[\![P \text{ says } s \wedge K_P \text{ says } s \wedge A]\!] = P \text{ says } s \wedge [\![A]\!]$$

$$[\![(P \wedge K_B) \text{ controls priv} \wedge A]\!] = P \text{ controls priv} \wedge [\![A]\!]$$

The distinguishing characteristic of our proposal is the separation of online and offline checking phases, where in the online phase certain elements of authorization are taken for granted, or trusted to hold. This yields a simpler authorization decision, which can be verified more rigorously during the offline phase. However, with security at stake, vague accounts of the relation between these phases does not suffice, rather we desire a formal relationship, so that offline *verification* of online authorization is meaningful. We embody this notion in the trust transformation, which specifies what elements of online authorization are to be taken for granted, by specifying how to transform untrusted requests into trusted ones. Thus, we make the following simplifications for more efficient online checking

1. Individual are trusted to make valid claims about role members
2. Authenticity of request is assumed for any intermediary; for example if request R_1 *says* R_2 *says* s is received, and then we trust that R_1 truly said $"R_2 \text{ says } s"$ and R_2 truly said *"s"*

In practice these assumptions are too simple, some justifications for that claim should be made , for example key signature on the "top level" of the request so that instead of $R_1 says\ R_2 says\ s$ the signed request K_{R_1} *says* R_2 *says* s would be communicated

Lemma 2.Suppose s is a trusted context, $s \vdash$ ***Priv*** *is a valid, and authorizations (s) returns* s'; then $s' \vdash$ ***Priv*** *is valid*

The trust transformation mapping rigorously defines the relation between extrapolated and trusted forms. Since notions of trust can vary depending on the system, we specify the type and necessary preconditions of trust transformations, but the definition of the function itself is left up to a particular service requested. For any extrapolated formula s, we denote its trust transformation as $[\![s]\!]$. Any trust implementation must define extrapolated and trusted authorization forms, and the trust transformation between them, with the requirement that it be a total function on the set of extrapolated statements. Also, we specify that the authorization contexts mentioned in Condition 1 are in trusted form.

Example 6: Suppose access control for an eCommerce service ***eCS*** *is based on*

requests made in both signed and unsigned form, so that all requests are of the form: $B \text{ says } s \wedge K_B \text{ says } s$

Suppose further that in the online component, players are trusted to communicate messages faithfully, and signatures are not checked. All authorization contexts include an ACL A, which is left unchanged by the trust transformation. Thus, for all B and s, the trust transformation is defined as: $[\![B \text{ says } s \wedge K_B \text{ says } s \wedge A]\!] = B \text{ says } s \wedge A$. *Note that this transformation is total for the extrapolated form of requests in this example*

5. Specifying Trust through Authorizations

While online checking takes trust into account, the purpose of offline checking is to verify that this trust is warranted. As Trust transformations injects trust into Authorizations, Offline checking inverts the trust transformations, to verify online trust in the offline phase-that is, given a trust request *s,* offline checking searches for an extrapolated request of which *s* is the trust transform. We call this all process *Verifications*

First, as mentioned above, Authorizations returns the extrapolated form of trusted authorization contexts. Since the trust transformation has been defined formally, we can precisely characterize this condition as follows:

Conditions 2.*Let s be a trusted context; then if Verifications (s) succeeds, verifications(s)* $\vdash s'$ *such that* $[\![s']\!] = s$

Note that this condition allows a certain degree of flexibility, in that authorization must return a statement that is at least as strong as *an* extrapolated form of the input, not necessarily an extrapolated form per se.

Furthermore, we say an extrapolated form, since it is possible that any given trust transformation is many to-one. Significantly, it is not even necessary that an extrapolated form of an authorized statement be authorized. However, since Authorizing seeks to verify trust implicit in an online check, we require that authorizing not only return an extrapolated form of input statements, but one that is also authorized for the privilege in question; otherwise, authorizations fails. This motivates the third condition of our framework:

Condition 3. *Let s be a trusted context and* **priv** *be a privilege. If* $s \vdash$ ***Priv*** *holds, then so does Verifications(s)* $\vdash$ **priv** .

It is important to note that this condition does not necessarily require theorem proving on extrapolated forms, but rather provability should follow by adherence to this condition generally.

5.1 Checking Privileges

To make sure that the requested operation is authorized. *Check Privilege* (*T*) returns true if the statement *Ok*(*T*) can be derived from Φ, A_{eCE}, and B_F (the belief set of the entity which called *check Privilege*). We define *eCE(F)* to be the eCommerce environment in which a given entity *F* is running. Next, we can define

$$E_F \equiv (\Phi \cup A_{eCE(F)} \cup B_F) \qquad (8)$$

We call E_F the *environment* of the entity *F*. The goal of *check Privilege* (*T*) is to determine, for the entity *F* invoking it, whether $E_F \supset Ok(T)$.

The decision procedure used by *check Privilege* takes, as arguments, an environment E_F and a target *T*. The decision procedure examines the statements in E_F and divides them into three classes.

- *Class 1 statements* have the form *Ok*(*U*), where *U* is a target.
- *Class 2 statements* have the form $P \Rightarrow Q$, where *P* and *Q* are atomic principals.
- *Class 3 statements* have the form $F_1 \mid F_2 \mid \ldots \mid F_k$ **says** *Ok(U)*

where F_i is an atomic principal for all $i, k \geq 1$, and *U* is a target.

Theorem 2: (Termination) *The decision procedure always terminates.*

Proof: The result follows directly from the fact that E_F has bounded cardinality. This

implies that each intermediate in the eCommerce environment has a bounded number of iterations; and the amount of work done in each iteration is clearly bounded.

5.2 Intermediate Authority

In any access control statement ***P*** controls **priv**, it is not necessary for ***P*** to be atomic, allowing a fine-grained approach to access control. For example, if it is not desirable to grant a role ***R*** direct access to **priv**, but only on behalf of a principal ***D***, the ACL can specify $R \mid D \text{ controls } \mathbf{priv}$, disallowing ***R*** direct access to **priv**.

However, we're concerned with access control decisions in the presence of multiple intermediaries-that is, several intervening nodes may transport an authorization request from source to *target*. The above scheme would require separate entries for every possible chain of intermediaries; for example, given statements:

$$R_1 \text{ says } R_2 \text{ says } D \text{ says } \mathbf{priv} \quad (9)$$

$$R_2 \text{ says } R_1 \text{ says } D \text{ says } \mathbf{priv} \quad (10)$$

authorization for both would require both of the following statements to be present in the relevant ACL:

$$R_1 \mid R_2 \text{ } D \text{ controls } \mathbf{priv} \quad (11)$$

$$R_2 \mid R_1 \text{ } D \text{ controls } \mathbf{priv} \quad (12)$$

Either that or it would require access statements:

$$R_1 \mid D \text{ controls } \mathbf{priv} \quad (13)$$

$$R_2 \mid D \text{ controls } \mathbf{priv} \quad (14)$$

to be present, along with known relations $R_1 \Rightarrow R_2$ and $R_2 \Rightarrow R_1$. The former solution (9) is clearly cumbersome, unrealistically so given the number of possible intermediaries on the eCommerce architecture above in figure 1. The later is better (10), but is restrictive, requiring every intermediary to adopt the same role as, or a more powerful role than, its predecessor (although this problem could be alleviated by adapting the "speaks for regarding" relation proposed in [6] as an extension to ABLP). Since the only privilege at issue is **priv**, it is intuitively sufficient for each intermediary to have some sort of authorization for **priv**, as in e.g. stack inspection [4].

However, unlike stack inspection, eCommerce service intermediaries should often not be granted direct access to privileges-for example, an eCommerce service should not be granted direct access to withdraw cash from an individual's bank account, but only on behalf of that individual. To maintain this property, and to overcome the drawbacks of the approaches described in the previous paragraph, we introduce the notion of an *intermediator authority*. Intuitively, an intermediator authority is the authority to *carry* an authorization request for a particular principal, but not the authority to make the request itself. Formally:

$$R \text{ intermediator } \mathbf{priv} \text{ for } D \triangleq \left(R \mid D \text{ says } \mathbf{priv}\right) \supset \left(D \text{ says } \mathbf{priv}\right) \quad (15)$$

In the above example (15), access requires the intermediator authorities' R_1 *carries* **priv** for *D* and R_2 *carries* **priv** for *D*, as well as the direct authority *D controls* priv. In general, *intermediator* authority allows a fine-grained and flexible approach to authorization in the context of eCommerce services. We call conjunction of *intermediator* authority statements *intermediator control lists* (ICLs), which we denote *I* in our proposal.

6. Conclusion and Future works.

On the theoretical front, we also plan to embed our authorization framework within a richer formalism, such as that proposed in [10], allowing the semantics of authorization to be expressed and verified in a realistic threat model. In addition to stand alone eCommerce services, authorization for composite web services [8, 13] also promises to be an interesting topic for future work; access control in that setting presents a number of unique issues for investigation.

We have used ABLP logic to establish a formal setting for framework design, and

specified the conditions that any trust intermediator authorizations implementation must satisfy. The central ideas we have presented are the separation of online and offline authorization phases, the notion of a trust transformation that establishes a meaningful relation between these phases, and a characterization for offline verification of online checking.

However this paper presented a framework for trust and authorization for eCommerce services system. The proposed architecture integrated the role-based and privilege-based access control into the trust and authentication services. This will better suit to the eCommerce service system in terms of identity management. The authors hope that the proposed framework would help the developing and administration of eCommerce service systems in a secure, efficient and flexible way. In the next step of this research, we will design and implement the authorization policy and the role-and-privilege based authentication for e-health service systems.

Our proposal has made some suggestions which can be used for implementations in general, including the use of role key certifications and intermediator authority for flexible authorization schemes in an open eCommerce environment. We showed that access control decision based on trust intermediator corresponds to the constructions of a proof in the logic and we have presented an efficient decisions procedure for generating these proof.

References:

[1] R.L. Rivest, A. Shamir, and L.A. Adleman, method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM 21*, 2 (Feb. 1978), 120–126.

[2] B. Lampson, M. Abadi, M. Burrows, and E. Wobber, Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems 10*, 4 (Nov. 1992), 265–310.

[3] M. Abadi, M. Burrows, B. Lampson, And G. D. Plotkin, A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems 15*, 4 (Sept. 1993), 706–734.

[4] D. S. Wallach and E. W. Felten. Understanding java stack inspection. In *Proceedings of 1998 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1998.

[5] L. Bauer, A. W. Appel, and E. W. Felten. Mechanisms for secure modular programming in java.Technical Report TR-603-99, Princeton University,Computer Science Department, July 1999.

[6] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI certificate theory. RFC 2693, Sept. 1999.

[7] D. S. Wallach, A. W. Appel, and E. W. Felten. SAFKASI: a security mechanism for language-based systems. *ACM Trans. Softw. Eng. Methodol.,*9(4):341-378, 2000.

[8] V. Christophides, R. Hull, G. Karvounarakis, A. Kumar, G. Tong, and M. Xiong. Beyond discrete e-services: Composing session-oriented services in telecommunications. In *Proceedings of the Workshop on Technologies for E-Services (TES)*, Rome, Italy,2001.

[9] A. D. Gordon and R. Pucella. Validating a web service security abstraction by typing. In *Proceedings of the 2002 ACM workshop on XML security,* pages 18_29.ACM Press, 2002.

[10] A. Gordon, K. Bhargavan, C. Fournet, and R. Pucella. Tulufale: A security tool for web services. In Springer, editor, *Formal Methods for Components and Objects,* LNCS, 2003.

[11] C.Skalka and X.Sean Wang ,Tust But Veryfy: Authorizations for Web Service: *ACM Workshop on Secure Web Services,* October 29, 2004, Fairfax VA,USA.

[12] K. Bhargavan, C. Fournet, and A. D. Gordon. A semantics for web services authentication. In *Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages,* pages 198-209. ACM Press, 2004.

[13] R. Hull and J. Su. Tools for design of composite web services. In *ACM SIGMOD*, pages 958-961, 2004.

[14] Security Breaches Survey Technical Report (URN 06/803).This is available from 25 April 2006 and can be downloaded from www.security-survey.gov.uk

[15] A. W. Appel and E. W. Felten. Proof-carrying authentication. In G. Tsudik,

editor,Proceedings of the 6th Conference on Computer and Communications Security, Singapore, Nov. 1999. ACM Press

[16] L. Bauer. Access Control for the Web via Proof-carrying Authorization. PhD thesis, Princeton University, 2003.

[17] E. Wobber, M. Abadi, M. Burrows, and B. Lampson. Authentication in the Taos operating system.Technical Report 117, DEC Systems Research Center, 130 Lytton Avenue, Palo Alto, Ca 94301, December1993.

[18] D. S. Wallach. A New Approach to Mobile CodeSecurity. PhD thesis, Princeton University, 1999.

[19] D. S. Wallach, A. W. Appel, and E. W. Felten.SAFKASI: a security mechanism for language-based systems. ACM Trans. Softw. Eng. Methodol.9(4):341_378, 2000.

[20] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI certi_cate theory. RFC 2693, Sept. 1999

[21] R. Rivest and B. Lampson. SDSI _ a simple distributed security infrastructure, 1996.

[22] N. Li, B. N. Grosof, and J. Feigenbaum. Delegation Logic: A logic-based approach to distributed authorization. ACM Transaction on Information and System Security (TISSEC), Feb. 2003. To appear

[23] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust management framework. In Proceedings of the 2002 IEEE Symposium on Security and Privacy, pages 114_130. IEEE Computer Society Press, May 2002.

[24] E. G. Sirer and K. Wang. An access control language for web services. In Proceedings of the seventh ACM symposium on Access control models and technologies, pages 23_30. ACM Press, 2002.

[25] A. W. Appel and E. W. Felten. Proof-carrying authentication. In G. Tsudik, editor, Proceedings of the 6th Conference on Computer and Communications Security, Singapore, Nov. 1999. ACM Press.

Appendix

A. ABLP Logic

Here is a list of the subset of axioms in ABLP logic used in this paper. We omit axioms for delegation, roles, and exceptions because they are not necessary to discuss trust authorizations logic applications.

Axioms about Statements

If s is an instance of a theorem of propositional logic then s is true in ABLP (16)

If s *and* $s \supset s'$ *then* s'. (17)

$(A \textbf{ says } s \wedge A \textbf{ says } (s \supset s')) \supset A \textbf{ says } s'$. (19)

Axioms about Principals

$(A \wedge B) \textbf{ says } s \equiv (A \textbf{ says } s) \wedge (B \textbf{ says } s)$ (20)

$(A \mid B) \textbf{ says } s \equiv A \textbf{ says } B \textbf{ says } s$ (21)

$A = B \supset (A \textbf{ says } s \equiv B \textbf{ says } s)$ (22)

$i \mid$ is associative. (23)

$i \mid$ distributes over $\wedge$ in both arguments. (24)

$((A \Rightarrow B) \equiv (A = A \wedge B))$ (25)

$((A \textbf{ says } (B \Rightarrow A)) \supset (B \Rightarrow A)$ (26)

if s is an instance of a propositional-logic tautology

then $\vdash s$ (27)

if $\vdash s$ and $\vdash (s \supset s')$ then $\vdash s'$ (28)

$If\ A\ says\ (s \supset s') \supset (A \textbf{ says } s \supset A \textbf{ says } s')$ (29)

if $\vdash s$ then $\vdash A \textbf{ says } s$, for every A (30)

Syntax: The formulas are defined inductively, as follows:

a countable supply of primitive propositions $p_0, p_1, p_2, \ldots$ are formulas;

if s and s' are formulas then so are $\neg s$ and $s \wedge s'$;

if A and B are principal expressions then $A \Rightarrow B$ is a formula;

if A is a principal expression and s is a formula then $A \textbf{ says } s$ is a formula.

We use the usual abbreviations for boolean connectives, such as $\supset$ and we also treat equality between principals (=) as an abbreviation. In addition, *A controls s* stands for $(A \textbf{ says } s) \supset s$.