A Knowledge-Based Approach to Internet Authorization Using PKI

Abing Lin
Trusted E-Services Laboratory
HP Laboratories Bristol
HPL-2000-133
October 13th, 2000*

E-mail: alin@hpb.hplhp.com

policy-driven management, security management, authorization

In this paper, a knowledge-based approach to Internet authorizations is proposed by using Public-Key Infrastructure (PKI) based digital certificates, trust models, Role-Based Access Control (RBAC), and intelligent back tracking. Security policies are expressed as the rules in a knowledge base. An inference engine is utilised to evaluate policies, dynamically assign roles to Internet users, and redo roles assignment automatically.

* Internal Accession Date Only
Approved for External Publication

A Knowledge-Based Approach to Internet Authorizations Using PKI

Along Lin

Trusted E-Services Laboratory
Hewlett-Packard Laboratories
Filton Road, Stoke Gifford
Bristol BS34 8QZ, U.K.
Email: alin@hplb.hpl.hp.com

Abstract

In this paper, a knowledge-based approach to Internet authorizations is proposed by using Public-Key Infrastructure (PKI) based digital certificates, trust models, Role-Based Access Control (RBAC), and intelligent backtracking. Security policies are expressed as the rules in a knowledge base. An inference engine is utilised to evaluate policies, dynamically assign roles to Internet users, and redo roles assignment automatically.

Keywords: Policy-Driven Management, Security Management, Authorization

1 Introduction

With more electronic services being deployed for access over the Internet, security is becoming an increasingly critical issue. Existing access control mechanisms do not provide a flexible management solution to Internet authorizations. Discretionary Access Control (DAC) prevents unauthorized users from performing operations on resources by administering the relevant Access Control List (ACL), which is not scalable for Internet environments. Mandatory Access Control (MAC) eases the security management by attaching security labels to subjects and objects, and enforces a specific security policy that allows only one-directional information flow between those subjects and objects. However, it does not provide any support for general Internet authorizations. Although several RBAC systems have been developed, they still have the following drawbacks: manual user-role assignment, business logic coupled privileges, and the assumption that users are known to resource providers in advance. Attribute and authorization certificates contain accessing rights within them and seem to be a promising solution [5].

Attribute Certificates (ACs) bind attributes to users Distinguished Names (DNs). They can be used with identity certificates to achieve the mapping: attributes ? DN ? public key. The motivation for introducing ACs in PKI technology was that X.509 public key certificates were used for identity purpose by establishing a secure certified relationship between issued public keys and their owners. Although attribute information can be

included in X.509 v3 certificates as extensions to satisfy application-specific needs, this is not worthy from the perspectives of interoperability, jurisdiction, and revocation. First, private attributes extensions make it hard for disparate systems to interoperate with each other. Second, it is difficult for an organization to issue a certificate containing both the identity information and attributes that may be accessing rights. Finally, attributes are more volatile than identity information, and therefore need revoking more frequently.

However, having authorizations in an AC has drawbacks too. First, an attribute certificate authority must issue ACs to potential Internet users before they can access the controlled resources, which is inconvenient to users and sometimes can be disastrous to security managers from certificates management perspective. Second, for those ACs that have a long period of validity, revoking them can be much of a burden to the issuing CA when the accessing rights in them have to be updated due to the business policy changes.

In this paper, it is assumed that ACs contain only users comparatively stable attributes without any accessing rights. A user may have several ACs issued by different trusted CAs that have intimate knowledge of the user. E.g., an accredited university or institute issues its graduates normal ACs containing such attributes as degrees, qualifications, majors, graduation dates, and the like. A bank issues each of its account owners an AC that contains account number, account type, branch name, opening date, and so on. A trusted driving license agency may issue a qualified driver an AC that includes driving license number, car categories, validity dates, address, and the like. A user's ACs will then be used by various resource providers to make authorization decisions based on their business-specific security policies.

In the following, a knowledge-based RBAC is discussed first. Second, an automated user-role assignment based on their digital certificates and intelligent backtracking is described. Third, a knowledge-based approach to Internet authorizations is proposed by using PKI, trust models, RBAC, automated user-role assignment, and intelligent backtracking.

2 Knowledge-Based RBAC

In RBAC [1, 2, 3], permissions are only associated with roles and users get permissions by being members of appropriate business roles. This greatly simplifies a system's security management. Security officers create business roles and assign those roles to users based on their qualifications and responsibilities in organizations. Users can be easily reassigned from one role to another. Roles may be granted new permissions as new services are deployed, and permissions can be revoked from roles as needed.

However, traditional RBAC systems assume that a user's role is known when it requests particular resources, which requires the user to register with the resource providers in advance. This is neither convenient to Internet users nor cost-effective from system

management perspective. In Internet situations, it may be helpful to allow authorized users to access resources without asking them to register with resource providers in advance. An automated user-role assignment solution has been suggested [4], but it did not discuss how digital certificates are exchanged between users and resource providers.

In Knowledge-Based RBAC (KBRBAC), application-specific user-role assignment policy, trust models, and business logic on roles privileges are provided as a knowledge base. Any changes to it will dynamically drive the changes in an organization's security policies on its business processes, thus significantly alleviating the security management in a system. An independent knowledge base also gives security managers a lot of flexibility. When the knowledge base for configuring an RBAC system is empty, roles must be assigned to users manually and there is no extra business logic on roles privileges; otherwise, if the user-role assignment policies are enforceable and described in the knowledge base, those roles may be assigned to users automatically at run-time. For other roles that must be manually assigned to users, their role-assignment policy attributes are 'null'. To be more flexible, each role privilege is associated with a predicate used as the business logic on it. A predicate is defined by a set of rules. A privilege consists of a set of operations and objects to be performed on by role members. The role definition and an example are given in appendixes A, B, respectively.

Even though the knowledge base can be built by the security manager of an RBAC system with the help of an authoring tool on the resource controller's side, it still requires some information about the potential Internet users. What kind of information about a resource requestor should or can be sent to resource providers? How can the trust relationship be established between the unknown Internet users and service providers? These issues will be discussed in the following sections.

3 Automated Role-Assignment Using PKI and Intelligent Backtracking

An approach has been proposed to dynamically map a user to predefined business roles based on the user's digital certificates issued by trusted Certificate Authorities (CAs) and role-assigning policies pre-set by resource providers [4]. Because the user does not know anything about a resource provider's authorization policy, various digital certificates may be required to present to the resource provider for accessing a particular resource [5]. An intelligent backtracking mechanism is therefore necessary for supporting the non-deterministic automated user-role assignment, which means there will be a lot of traffic between a resource requestor and a resource provider at run-time. Furthermore, in order to validate Internet users digital certificates, the trust relationships between resource providers and CAs have to be defined in the trust models of a knowledge base.

Although a user may be assigned several business roles based on its digital certificates and pre-set role-assigning policies, only some of them will authorize it to access the requested resource. For those roles that have privileges matching the user's current access

request, the policy evaluator will be called to evaluate the policies enforced on those privileges. If the policy evaluates true, the user's current request is authorized and the relevant actions can then be performed on the resource; otherwise, alternative roles will be tried. For each assigned role, if either it does not have a matching privilege for the user's current request or the matching privilege's associated authorization policy evaluates false, the inference engine will redo the previous role assignment automatically. If all roles have been tried, the user's request will be denied. Because the inference engine can support backtracking and policy evaluation, it can be combined into one on the resource provider side.

To avoid assigning a user to the roles that do not have the required privilege, an efficient indexing mechanism is needed to support the retrieval of the candidate assignable roles. Only those roles having the required privilege will be tried assigning to the user for current request. However, whether a role will be assigned to a user still depends on the user's presented digital certificates, the service provider's role-assignment policy and trust models, which are expressed as rules in a knowledge base. The service provider may not trust some Certificate Authority (CA), and asks the user to present other digital certificates issued by particular CAs. When a user accesses a resource, the resource provider will first get a set of candidate roles whose privileges match the user's current request to improve the role-assignment efficiency. Then, it will try to assign those roles to the user based on the presented digital certificates and the knowledge base.

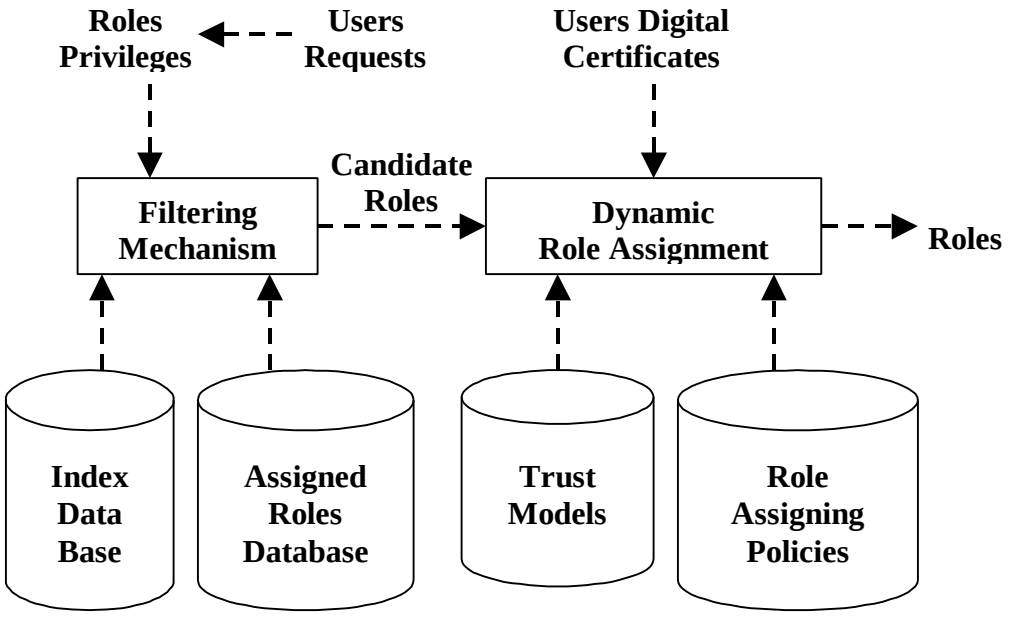


Figure 1. Automated Role Assignment Model

Even though it is the security manager's responsibility to enforce the principle of *separation of duties* when role-assigning policies are set, an automated mechanism for guaranteeing this is very helpful so that two conflicting roles will not be assigned to a

user at run-time. This can be achieved by not satisfying the policies on two conflicting roles assignment simultaneously or storing users assigned roles information in a database persistently. A user's candidate assignable roles must not be in conflict with the roles that have already been assigned to the user (see Fig. 1). If such conflicts occur, the security manager will be informed of them and take necessary measures as required.

Our automated role-assignment solution based on PKI and intelligent backtracking has a unique advantage over the traditional RBAC and others [4] in that it enables users to specify various domain-specific security policies in a knowledge base. It allows Internet users to access resources more conveniently and security policies can be managed flexibly. Furthermore, because much of the user-role assignment management has been automated, the security manager's work will be reduced to a minimum by using a configurable knowledge base.

4 A Knowledge-Based Approach to Internet Authorizations Using PKI

Now that we have discussed how to authorize a user's request based on its digital certificates, a knowledge base, and intelligent backtracking, we need to consider what information needs to be exchanged between the user and the resource provider without disclosing the server side security policies to the user. It is not reasonable to ask the user to submit all its digital certificates, most of which may be irrelevant to its current request, and the user may not be willing to do that. Therefore, during the authorization decision-making, users digital certificates may be requested. For simplicity, we assume that the user has already been authenticated and we will only focus on the authorizations below.

There are three steps in requesting a user's digital certificates. First, the Server Security Agent (SSA) needs to decide what digital certificates are required for the current user's request. Based on the request and the security policies in the knowledge base, the SSA can automatically find the potential set of digital certificates using a logical reasoning. Second, the SSA checks if some of those digital certificates have already been cached in a directory and are still valid, and then sends a request for unavailable digital certificates to the user. Finally, the user sends the requested digital certificates back to the SSA, which will then cache them in its directory for future use using Lightweight Directory Access Protocol (LDAP).

However, sending digital certificates is only part of the Internet authorization (illustrated in Fig. 2). If any of the following conditions holds, the SSA will have to redo its previous role assignment and find another set of digital certificates to try other alternative roles for the user's current request, and re-evaluate the associated authorization policy.

- Some digital certificates presented have already been revoked by their CAs using Certificate Revocation Lists (CRLs) or the Online Certificate Status Protocol (OCSP).

- Some digital certificates are neither valid nor trusted by the SSA according to the trust models described in the knowledge base.
- The presented digital certificates cannot assign the user a role that authorizes the current request based on the resource provider's role-assigning policy expressed in the knowledge base.
- The authorization policy on the assigned role's matching privilege evaluates false using the system models in the current context.

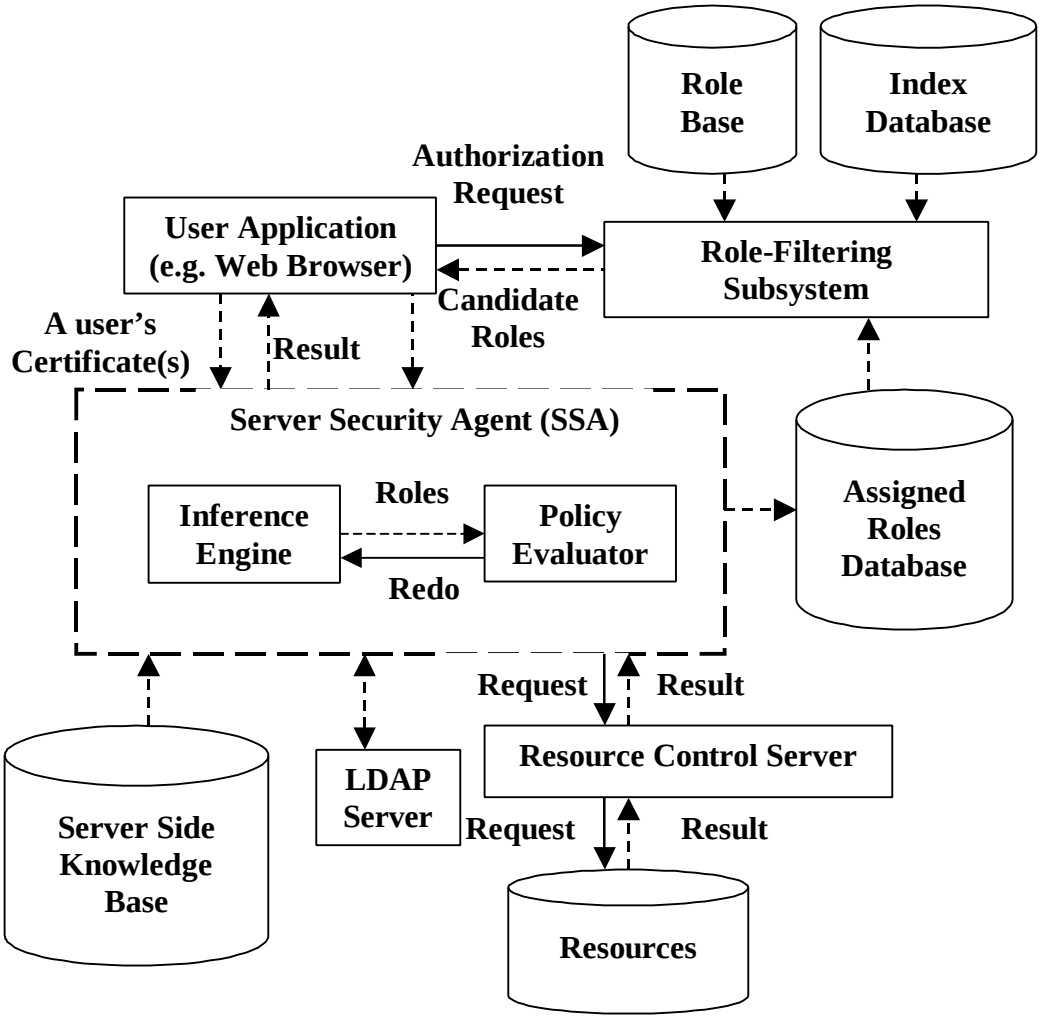


Fig. 2. Architecture for Knowledge-Based Internet Authorization Using PKI

The client's digital certificates can be sent to the SSA by using a trusted applet digitally signed by the resource provider or a plug-in for the client's Web browser. The communication between the user and the resource provider is based on Secure Socket Layer (SSL). The client can optionally configure the certificate-sending policy on what digital certificates can be sent automatically without being informed in advance. If there is no policy configured for sending a particular certificate, a user will be informed of the

certificate request by default, and will decide whether to send it to the SSA for the user's current request. If there is a certificate-sending policy and it evaluates false, the requested digital certificate will not be sent to the SSA; otherwise, it will be sent to the SSA automatically without the user's intervention. In Business-to-Business situations, client side authorization for sending digital certificates to the server side also needs a knowledge base that describes the client's certificate-sending policy and trust models (as shown in Fig. 3).

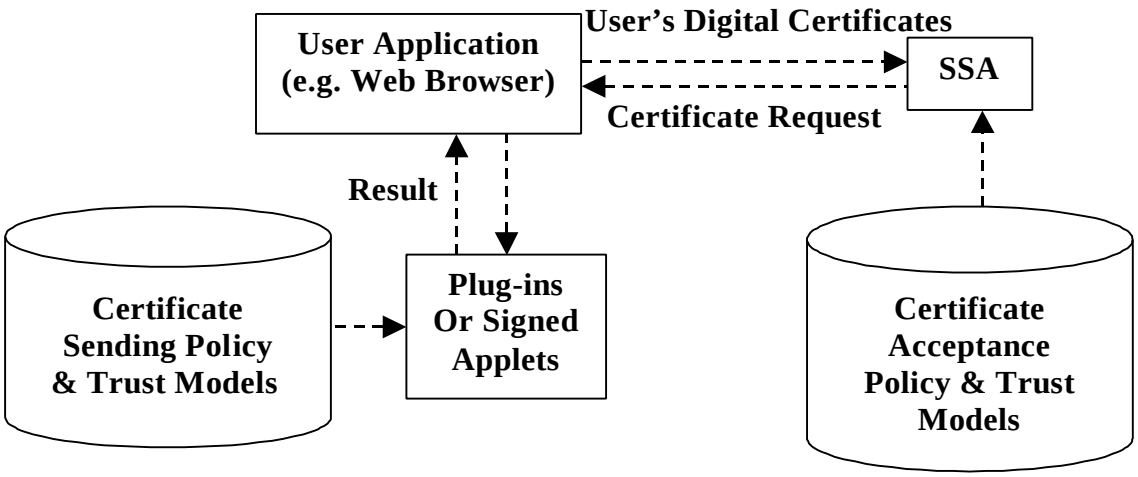


Figure 3. Knowledge-Based Certificate Sending

Regarding the client's trust in an applet, there are two primary types of models. One is hierarchical and the other is based on the web of trust. The former assumes that the user's chain of digital certificates is rooted by a CA trusted by its receivers. The latter allows any of the digital certificates in a chain can be cross-certified by more than one CAs. Because trust policies are always business related and application-specific, they must be configurable by means of an independent knowledge base so that the SSA, a Web browser or a trusted applet can use it to make trust decisions.

As pointed out in [6,7,8], a logic-based knowledge representation is expressive enough and has several advantages over other approaches. PROLOG (PROgramming in LOGic) is based on Horn-clauses, which is a subset of first order logic, and has a solid theoretical foundation. What is more, both policy evaluation and policy conflict detection can be easily done within the logic framework as well. Therefore, it is adopted as the policy representation language in this paper.

5 Conclusions

In this paper, a knowledge-based approach to Internet authorizations is proposed by using PKI-based digital certificates, trust models, policy-based roles privileges, automated user-role assignment, and intelligent backtracking. Security policies are expressed as the rules in an independent knowledge base. An inference engine is utilized to evaluate various policies, dynamically assign roles to Internet users, and redo roles assignment automatically. The innovative approach makes the administration of Internet users accesses to resources less of a burden to security managers. The proposed architecture for Internet authorizations is capable of dealing with both unknown Internet users and automatic user-role administration. It overcomes some of the drawbacks of existing RBAC systems, which are neither flexible for access control management nor convenient to Internet users.

A policy-authoring tool is strongly recommended to be used to alleviate the administration task of security managers. The tool should be capable of mapping a user's access request to a role's privilege and even refining a high-level security policy into executable rules, so that security managers can focus on the specification of authorization policies, trust models, and role-assigning policies in the knowledge base.

References

[1] Ferraiolo D F, Barkley J F, Kuhn D R, A Role-Based Access Control Model and Reference Implementation Within a Corporate Intranet, ACM Transactions on Information and System Security, Vol. 2, No. 1, (1999) 34-64.

[2] Ferraiolo D F, Cugini J A, Kuhn D R, Role-Based Access Control (RBAC): Features and Motivations, In Proc. of 11th Annual Computer Security Applications Conf., (1995) 241-248, http://hissa.ncsl.nist.gov/rbac/newpaper/rbac.html.

[3] Giuri L and Iglio P, A formal model for role based access control with constraints, In Proc. of the Computer Security Foundations Workshop, (1996) 136-145.

[4] Herzberg A, Mass Y, Mihaeli J, Naor D, and Ravid Y, Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers, http://www.hrl.il.ibm.com/TrustEstablishment/paper.htm.

[5] Johnston W, Mudumbai S, and Thompson M, Authorization and Attribute Certificates for Widely Distributed Access Control, In Proc. of the IEEE 7th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE-98), (1998) 340-345.

[6] Li N, Feigenbaum J and Grosof B, A Logic-based Knowledge Representation for Authorization with Delegation, In Proc. of the 12th IEEE Computer Security Foundations Workshop, (1999) 162-174.

[7] Massacci F, Reasoning about Security: a Logic and a Decision Method for Role-Based Access Control, In Proc. of the International Joint Conference on Qualitative and Quantitative Practical Reasoning (ECSQARU/FAPR-97), Lecture Notes in Artificial Intelligence, Vol. 1244 (1997) 421-435.

[8] Seamons K E, Winsborough W and Winslett M, Internet Credential Acceptance Policies, In Proc. of the Workshop on Logic Programming for Internet Applications, July 1997, http://www.transarc.com/~winsboro/papers/CAP.html.

Appendix A – the role syntax

```
<Role>          ::= 'Name:'                    <Role Name>'.'
                    'Role-Assigning Policy:'  {<Predicate> | 'null'}'.'
                    'Authorizations:'          <Privileges>
                    '.'
<Role Name> ::= <Symbolic Atom>
<Privileges>  ::=  <Privilege>*
<Privilege>    ::=  <Policy> ',' {<Method>}+'.'
```

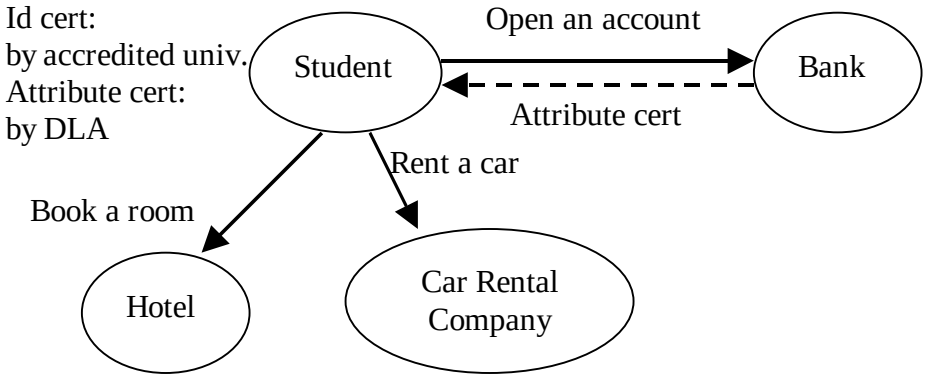
Where, <Policy> is a predicate and is described by a set of PROLOG clauses defined as below, and <Method> is defined as an external function in models. A special predicate is symbolic atom 'true'.

```
<Clause>       :: = <Fact> | <Rule>
<Fact>          ::= <Predicate>'.'
<Rule>          ::=  <Predicate> ':-' <Conditions>'.'
<Conditions>  ::= '(' <Conditions> ')'
                     | <Expression> | '!' <Expression>
                     | <Expression> ',' <Conditions>
<Expression> ::=  <Argument><RelOp><Argument> | <Predicate>
<RelOp>       ::= '==' | '<' | '>' | '<=' | '>=' | '!='
<Symbolic Atom> ::= ['a'-'z']{['a'-'z'] | ['A'-'Z'] | ['0'-'9'] | '_'}*
<Argument>   ::= <Symbolic Atom> | <String> | <Number> | <Logical Variable>
                    | <Functor><Parameters>
<Functor>      ::= <Symbolic Atom> | '.' | <Arithmetic Operators>
<Parameters>  ::= '('<Argument>{','<Argument>}*')'
<Logical Variable> ::= '_' | ['A'-'Z']{['a'-'z'] | ['A'-'Z'] | ['0'-'9'] | '_'}*
<Predicate>    ::=  <Symbolic Atom>[<Parameters>]
<Policy>        ::= <Predicate>
```

Appendix B – A Bank Example

We use a simple example to demonstrate the ideas contained in the paper. It is assumed that a full-time student has already been issued an identity certificate by an accredited university, and has got a digital driving license issued by the Driving License Agency (DLA) after having passed both the theory and practical tests. The student wants to get

the service of car rental or hotel reservation on the Internet, which requires the student to open an account in a recognized bank first to get digital credit certificates.



The bank provides a set of E-services, including new account creation, normal bank account transactions, and interest enquiries. In the following, various kinds of policies are modeled by PROLOG rules and the methods of privileges are left undefined. Some predicates are simply defined and even omitted here.

```
Name: bank_account_owners.
Role-Assigning Policy: bank_account_attribute_certs( Account_ids, Certificates ).
Authorizations:
        transfer_accounts_policy( Request, Account_ids, [From, To, Amount] ),
                transfer_money( From, To, Amount ).
        account_access_policy( Request, Account_id, Account_ids ),
                get_balance( Account_id, Balance ). // get the balance of the bank account.
        account_access_policy( Request, Account_id, Account_ids ),
                deposit_money( Account_id, Amount ).
        withdraw_policy( Account_id, Amount),
                withdraw_money( Account_id, Amount ).
        .
Name: bank_account_creators.
Role-Assigning Policy: certificates_for_bank_account_creation( Request, Identity ).
Authorizations:
        true, issue_account_attribute_certificate( Identity, Request ).
        .
Name: default.
Role-Assigning Policy: true.
Authorizations:
                get_interest_rate_policy( Request, Type ),
                        report_interest_rate( Type ). // return the interest rate to the user.
        .
// If the request is to create a new bank account and the user has valid and acceptable
// identity certificates, collect them into Accepted_certificates; Otherwise, return false.
```

```
certificates_for_bank_account_creation( Request, Identity_cert ) :-
        creating_bank_account( Request ),
        request_certificates( Certificates ), // Its definition is omitted
        valid_and_accepted_certificates( Certificates, Accepted_Certificates ),
        has_acceptable_identity_certificate( Accepted_certificates, Identity_cert ).
// Test if there is a valid and acceptable identity certificate in the given set of certificates

withdraw_policy( Account_id, Amount ) :-
        get_balance(Account_id, Balance),
        get_credit_limit( Account_id, Credit_limit),
        Amount <= Balance + Credit_limit.

transfer_accounts_policy( Request, Account_ids, [From, To, Amount] ) :-
        get_transfer_parameters( Request, From, To, Amount ),
        member( From, Accounts_ids ),
        member( To, Accounts_ids ),
        withdraw_policy( From, Amount ).

// Request attribute certificates issued by the bank to the user, and collect bank account
// numbers into Account_numbers. If the user doesn't have valid and acceptable attribute
// certificate issued by the bank, it returns false.
bank_account_attribute_certs( Account_numbers, Certificates ) :-
        request_certificates( Certificates ), // Its definition is omitted
valid_and_accepted_certificates( Certificates, Accepted_Certificates ),
        get_account_numbers( Accepted_Certificates, Accounts_numbers ),
        Accounts_numbers \= [].

// Get the bank account numbers of valid and accepted bank account attribute certificates
get_account_numbers( [Certificate|Rest], [No|Account_Nos] ) :-
        get_account_identifier( Certificate, No ),
        get_account_numbers( Rest, Account_Nos ).
get_account_numbers( [_|Rest], Account_Nos ) :-
        get_account_numbers( Rest, Account_Nos ).
get_account_numbers( [], [] ).

// Get the bank account number of a valid and accepted bank account attribute certificate
get_account_identifier( Certificate, Id ) :-
        valid_certificate( Certificate ),
        this_bank_service_accepted_certificate( Certificate ),
        bank_account_no( Certificate, Id ).

account_access_policy( Request, Account_id, Account_ids ) :-
        get_account_id_from_request( Request, Account_id ),
        // get the account number from the user's request
```

```
        member( Account_id, Account_ids ).

valid_and_accepted_certificates( [Valid|Rest], [Valid|Certificates] ) :-
        valid_and_accepted_certifiacte(Valid ),
        // test if the certificate is valid and accepted by the bank service provider
        valid_and_accepted_certificates ( Rest, Certificates ).
valid_and_accepted_certificates ( [_|Rest], Certificates ) :-
        valid_and_accepted_certificates ( Rest, Certificates ).
valid_and_accepted_certificates( [], [] ).
```