Journal of Information Security, **2012, 3, 25-38**
http://dx.doi.org/10.4236/jis.2012.31004 Published Online January 2012 (http://www.SciRP.org/journal/jis)

A Distributed Secure Mechanism for Resource Protection in a Digital Ecosystem Environment

Ilung Pranata, Geoff Skinner, Rukshan Athauda
Faculty of Science and IT, University of Newcastle, Callaghan, Australia
Email: {Ilung.Pranata, Geoff.Skinner, Rukshan.Athauda}@newcastle.edu.au

Received October 2, 2011; revised November 9, 2011; accepted November 27, 2011

ABSTRACT

The dynamic interaction and collaboration of the loosely coupled entities play a pivotal role for the successful implementation of a Digital Ecosystem environment. However, such interaction and collaboration can only be promoted when information and resources are effortlessly shared, accessed, and utilized by the interacting entities. A major requirement to promote an intensive sharing of resources is the ability to secure and uphold the confidentiality, integrity and non-repudiation of resources. This requirement is extremely important in particular when interactions with the unfamiliar entities occur frequently. In this paper, we present a distributed mechanism for improving resource protection in a Digital Ecosystem environment. This mechanism can be used not only for any secure and reliable transaction, but also for encouraging the collaborative efforts by the Digital Ecosystem community members to play a major role in securing the environment. Public Key Infrastructure is also employed to provide a strong protection for its access workflows.

Keywords: Digital Ecosystem; Authentication; Authorisation; Distributed Mechanism

1. Introduction

Information and resource protection is a de-facto requirement that must be advocated by every enterprise, organisation, and government entity. The importance of this requirement is further escalated when the entities are performing transactions in an online environment. Information security has been long considered as a crucial factor for e-commerce transactions. It is important to note that lack of sufficient security protection may limit the expansion of e-commerce technology [1]. However, although several e-commerce security mechanisms have been proposed and debated over a number of years, current internet technology still poses a number of incidents pertinent to the loss of information, unauthorized use of resources, and information hacking. These incidents generate an excruciating cost for the enterprises, ranging from the loss of revenue to the damage of their reputation. A recent survey [2] shows that the average cost resulted from the worst incident at about £280k - £690k per incident for a large organisation and £27.5k - £55k per incident for a small and medium organisation.

Similarly, Digital Ecosystem (DE) faces the identical issues due to its open environment where information and resources are exchanged over the network. With possibly thousands of Small and Medium Enterprises (SMEs) that form series of communities in a DE environment [3], protecting enterprise resources and acknowledging which entities are trusted to access the resources become extensive tasks for each enterprise. While ensuring security protection is all about upholding the confidentiality, integrity, availability and non-repudiation of information, it is evident that the most consistent and effective way to ensure the preservation of these security properties is through the implementation of authentication, authorisation, encryption, and access control mechanisms [1,4]. Additionally, the provision of an efficient mechanism to measure the trustworthiness of entities will further strengthen the information and resource protection [5]. While authentication ensures only the right entities that are allowed to consume the resources, authorisation restricts the access over multiple hosted resources based on each entity's privileges. Nevertheless, current research in these areas for a DE environment is still very much limited or not attempted. This research gap further becomes our main motivation to focus our work in.

The remainder of this paper is structured as follows: Section 2 provides an introduction of Digital Ecosystem and its security challenges. Section 3 provides an overview of our proposed solution. Section 4 provides an implementation of our proposed solution. Section 5 presents an security analysis on the proposed solution. This is followed by Section 6 which shows the results of performance and scalability testing on our solution. To conclude the paper, Section 7 summarizes our present work

and demonstrates several future works.

2. Digital Ecosystem & Its Security Challenges

Since its first inception, the newly emerging concept of Digital Ecosystem (DE) has received increasing attentions from researchers, businesses, IT professionals and communities around the world. A wide variety of researches and initiatives have been undertaken that aimed at the realization and implementation of a Digital Ecosystem concept. The enthusiasms were revealed inside numerous projects funded by European Commission under FP6 framework programme followed by FP7 framework programme as well as in numerous pilot regional workplans in Aragon, Tampere, Piedmont and West Midland [6,7]. The derived objective of DE primarily focuses on dynamic formation of a knowledge based economy [8]. Further, it was proposed that a knowledge based economy will lead to a creation of more jobs and a greater social inclusion in sustaining the world economic growth [6]. To realize this objective, it is critical to form an open framework infrastructure in promoting a wide use of Information and Communication Technology (ICT), as well as to solve the digital divide issues of internet and e-business adoption by small and medium enterprises [9].

The term "ecosystem" used inside DE notion is a fundamental biological science terminology to represent a dynamic interrelation between organisms and species that actively interact to conserve the environment [10]. Therefore, the similarity in concept and an analogy can be drawn between Digital Ecosystem and natural ecology community [9,11]. In a fundamental perspective, Digital Ecosystem is described as a digital environment and infrastructure where multiple digital components form a synergic correlation and collaboration with an evolutionary ability to adapt with its local circumstances [12]. Such digital components, or digital species used in other literatures, encompass various applications, services, frameworks, ontologies, knowledge, laws, taxonomies, reputation, training modules, trust relationships, and business models [9]. In a more technical term, Briscoe & Wilde [13] clearly define DE as the Multi-Agent Systems [14] that utilize distributed evolutionary computing [15,16] to combine the suitable agents in meeting user requests for applications. Further, the connectivity between the agents must be defined by the geography or spatial proximity unlike peer to peer [19] on which its peers' connectivity is based on bandwidth and information content.

In a DE environment where multiple interacting entities exist, the required efforts to enforce a strong authentication and authorisation mechanism are extensive. We identify three core issues that appear to be the challenging tasks to enforce such mechanisms. First, as the DE community expands its size to incorporate more entities, the resource providers face a challenge to identify the legal entities that are able to access their resources. Second, the fact that each entity would have different set of access permissions to access multiple resources further complicates the implementation of an efficient mechanism. Third, it is probable that each resource provider would host multiple resources and services in a DE environment. This situation, in turn, creates a great issue to authorize the right entities to the right resources with the right permissions. The failure to assign the right permissions to the authorized entities would compromise the usage of resources which would bring negative impact to the resource provider. Therefore, it is apparent that enforcing strong and efficient authentication and authorisation mechanisms in a DE environment needs in-depth solutions on the core issues.

However, the current internet mechanisms are still far from adequate to provide a reliable authentication and authorisation processes for a DE environment. This view is reflected from our literature analysis over a number of internet mechanisms. Several prominent authentication mechanisms such as Identity Provider (IdP) or Credential Provider [20], Credential Server (CRES) [21], Grid Security Infrastructure (GSI) MyProxy [22] utilizes a centralised approach for creating user credential although their implementation differs between one another. These mechanisms could be implemented in DE, however the conspicuous issue of single server failure must be carefully considered. In an event where the credential provider server is down, there possibly a chaos in a DE community due to the unavailability of credential services for client authentication. Apparently, several authorisation mechanisms such as CAS [23], Akenti [24], and PMI [25] also take a similar centralised approach.

These mechanisms inherit several issues pertinent to the centralised management. First, the central management would face real issue with the bottleneck and failure on its servers. Security breach would occur if the central servers fail to perform their authorisation processes over the clients. This situation exposes the resources to the malicious attacks as there is no other authorisation mechanisms are in place. Although it is possible to replicate the central server, the replication process will bring abundance administrative issues and higher chances for compromising the resources, considering a huge amount of data that needs to be replicated. Second, challenges occur when the central server attempt to assign the access permissions to the DE member entities. As a large number of resource providers that host one or more resources, the central server needs to register each resource and its access permissions individually. Further, this situation becomes even more challenging as a single resource could be associated with multiple different access permissions, and each client may have different ac-

cess permissions assigned to him. Therefore, the central management is not practical when there is huge number of entities in a DE environment. Third, serious administration issues would occur as a DE environment grows in size and diversity due to the great benefits that they can achieve. A central server will be experiencing huge burden to manage all client and resource providers' accounts and permissions even with the use of super computers or grid collections of computers.

Several DE literatures in [6,10,13] clearly reveal that DE is characterized as an open environment on which a centralized structure is minimized. DE must be engineered to provide a high resilience infrastructure while avoiding single point of control and failure. Therefore, a completely distributed control mechanism is required that immune to the super control failure. It is evident that the aforementioned internet mechanisms are inappropriate to be implemented in a DE environment due to its centralized management. We have dedicated a paper in [26] that derive the analogy of DE concept and further discuss its security challenges and requirements in detail. In this paper, we focus at the implementation and validation of our proposed DRPM solution [27,28] to solve the identified issues.

3. An Overview of DRPM

In this section, we provide a brief explanation of the important elements and workflows in in DRPM. Full discussion on these workflows could be found in [27,28].

3.1. Important Elements of DRPM

Two important elements of DRPM are client profile and capability token. A client profile is created in registration workflow when a client registers himself to access the resources. Its main function is to allow resource provider to capture all required, but voluntarily provided, client's information before any access to the resources is granted. The data that is contained in a client profile provides necessary information about who the client is and about their intentions for using the requested resources. Therefore, it ensures the resource provider that resources are not going to the wrong entities which would further impose the confidentiality and integrity of the resources. The use of client profiles also facilitates auditing process for the clients that are accessing a resource. For example, there may be a situation where a resource provider needs to make a trace back to determine which client was delegated access to the resource in case there was an incident involving a dispute or counterfeiting of the resource in question. Such implementation would reduce the risk of stolen data by the unauthorized entities.

At the end of registration process, an entity that is deemed eligible to access the resources would be issued a capability token. The purpose of this token is to simplify the management of access permissions that appear to be the challenge in collaborative environment, as have been pointed out by the number of literature from the previous section. A capability token functions as the authentication token for any subsequent access requests as well as for granting the access permissions. It contains the necessary right permissions for each client to perform a set of operations on a particular resource. This capability is produced by the resource provider on which a particular resource is hosted. On subsequent access request, this token is used by the resource provider to grant the client access to the resources and further provide the authorization process for the clients. Our basic design of capability token, as shown in **Figure 1**, contains the client profile identifier, resource provider identifier, resource identifier and list of access permissions, and it is expressed in XML [43] due to its simplicity, wide usability and self-descriptive characteristics. A time-stamp can be implemented in the capability token to determine the validity period of a user in accessing the resources. In the event where the trustworthiness of a new user is equivocal, a short-life capability token can be issued. Once the trustworthiness of the user gradually increases, resource provider can replace the short-life token with longer time-stamp validity. Additionally, the Uniform Resource Locator (URL) of resources is embedded in the token to provide an automatic and seamless connection to resource servers.

3.2. A Secure Registration Workflow

The DRPM registration portal is used to generate a client profile during the initial resource provisioning. The registration process comprises of three main stages: client registration, public key exchanges, and secure transfer of capability token. The resource provider endorsed certificate is utilized to identify the authentic resource provider

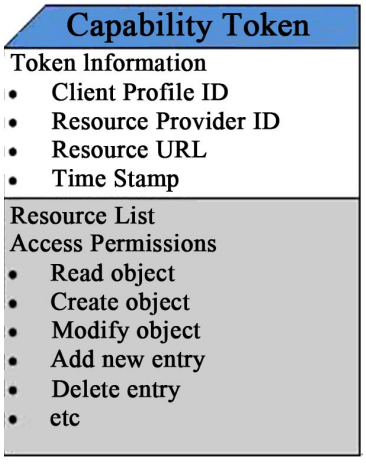


Figure 1. The structure of capability token.

based on its community endorsed public key certificate, which will be discussed in the next sub-section. The Public Key Infrastructure (PKI) is used to provide a secure communication between the client and resource provider. **Figure 2** shows the principal workflow for securing three stages of registration process.

The registration steps are detailed below:

1) A new client contacts the resource provider for requesting a resource. Resource provider sends its WoT endorsed public key to the client. Once the client determines and accepts the trustworthiness of the public key, he stores the resource provider trusted public keys and fills his information on the registration portal.

2) After the client information is filled, the registration portal creates a unique client and save this client profile.

3) Resource provider then requests for client certificate and stores the client public key on its repository. If required, WoT verification could be performed on client certificate to ensure the trustworthiness of the client.

4) The resource provider generates a client capability token based on client's allowed permissions.

5) Resource provider uses its own private key to sign the capability token. This process enhances the integrity of capability token over the untrusted network.

6) Resource provider then uses client's public key, received from step 3, to encrypt the signed message and send it to client end-point.

7) Client uses his own private key to decrypt the encrypted capability token.

8) Client then uses resource provider public key to generate the capability token from the signed message.

Note that at the final step of registration process, client will have his capability token and public key which was retrieved from the resource provider. The capability token and resource provider public key will be stored in client repository for subsequent requests. On another end-point, the resource provider stores the client's public key in its own repository. We trust that the combination of both encryption and hashing mechanisms further uphold the confidentiality, integrity and non-repudiation of capability token during its transfer in the communication channel.

3.3. A Secure Resource Access Workflow

Once a client has been successfully registered with the resource provider, client will present his capability token to the resource provider on every access request. The capability token which contains client assertions and authorization permissions is primarily used as a base by the resource provider for authenticating the client and granting the resource access. Three foremost protection requirements for the resource access are the identification of resource provider, secured transfer of capability token, and authentication of a requesting client. A detailed workflow that ensures security protection on each resource access is provided in **Figure 3**.

The steps are as follow:

1) Client retrieves the resource provider capability token. The capability token contains the client access permissions and the resource URL. At this stage, the client

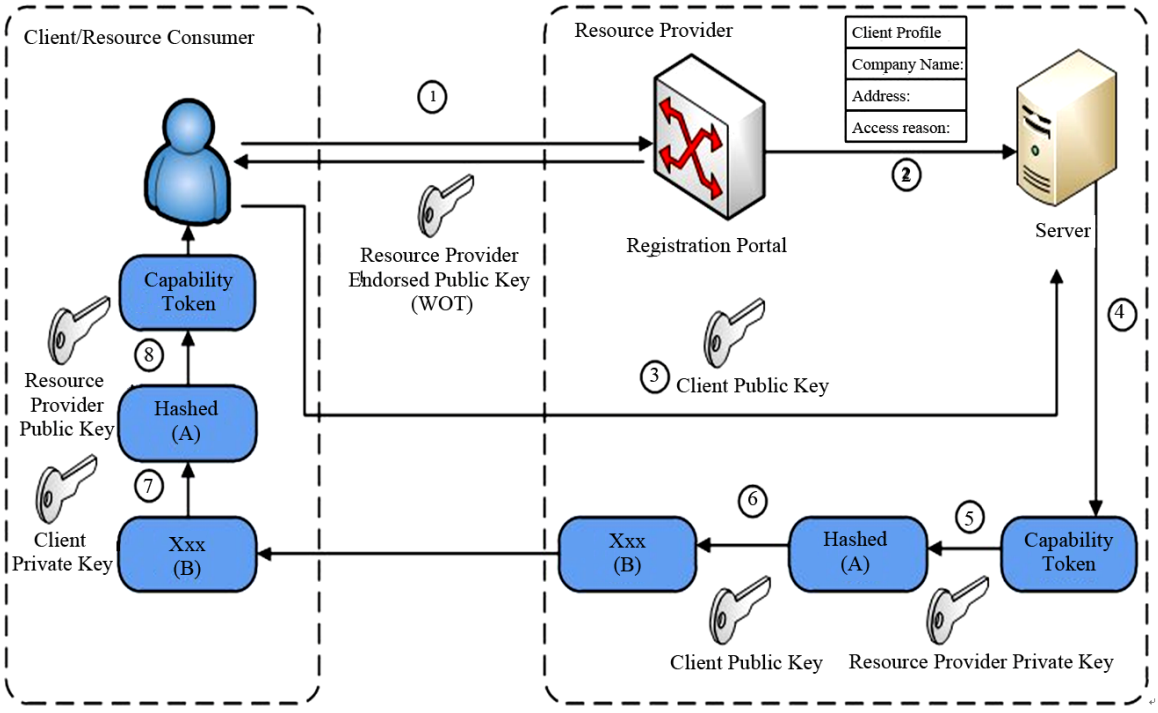


Figure 2. DRPM secure registration workflow.

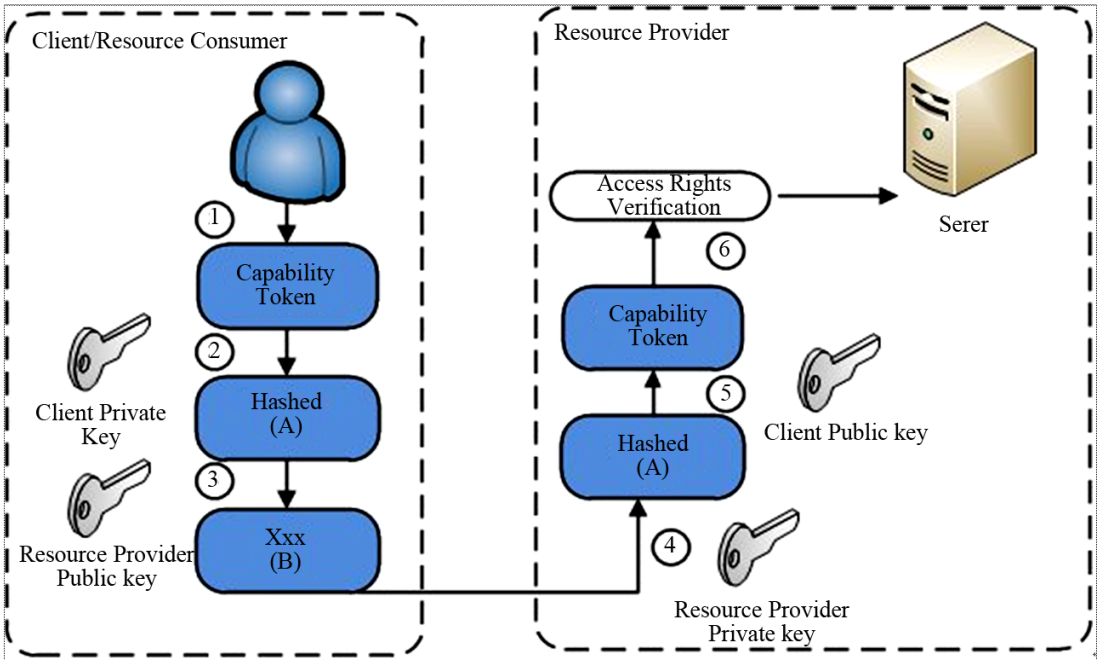


Figure 3. DRPM resource access protection.

also determines a symmetric pass key which will be shared with the resource provider and generate an Authentication Token which consists both symmetric pass key and capability token.

2) Client uses his private key to sign the capability token.

3) Client then encrypts the signed capability token using resource provider public key and he sends the encrypted message to the resource provider.

4) When resource provider received the encrypted message, it uses its own private key to de-crypt the message and retrieve the signed capability token.

5) Resource provider then verifies the signature of capability token using client public key. It then verifies the integrity of the capability token by generating the hash number from capability token.

6) Resource provider retrieves the access permissions listed in capability token.

Note that, on the step 1 of the workflow the client determines a symmetric pass key. This pass key will be utilized to generate a symmetric key for further communication after client is authenticated. In an event where the capability token is stolen due to man-in-middle attack, the unauthorized entity will still not be able to access the resource due as the symmetric key passphrase is shared between the legitimate client and resource provider only. This symmetric key would be used to secure communication after the authentication process. This is primarily due to the limitation of PKI which requires higher computation process. Further, a request for updating the resource provider public key could be made if the resource provider generates a new pair of public-private keys due to an unforeseeable security breach.

3.4. Engaging Community Protection

As discussed in Section 2, DE must limit its centralized structure and promotes the involvement of Small and Medium Enterprises (SMEs) for its successful implementation. It is evident that centralized approach using either a Certificate Authority (CA) or Credential Provider (CP) must be minimized. For DRPM, we propose an idea to integrate the community trust services, such as Web of Trust (WoT) into DRPM workflow, particularly in client registration process. We present this idea as an alternative approach for protecting resources in a DE environment. Further, the implementation of WoT in DRPM encourages active participation of DE member entities to protect their environment.

Web of Trust (WoT) is a community endorsed certificate which provides a decentralized trust management in a digital community. In WoT, there is no central authority (such as CA) that every entity trust, instead each entity is able to sign others certificates or public keys to build an interconnected web of public keys. The identifycation of an entity is provided primarily by his public key which is digitally signed by any number of “introducers”. Three degree of trustworthiness is introduced to reveal the reliability of the entity public key certificate: undefined, marginal and complete. Final decision for trusting the entity is rely on the user after examining the degree of trustworthiness. The prominent application of WoT is in Pretty Good Privacy (PGP) [30], which is used exten-

sively to secure emails. The implementation of WoT in DRPM and the mechanism to ensure the trustworthiness of WoT entities is not within the scope of this paper. This issue becomes an inspiration for our future work.

4. DRPM Implementation

Our DRPM prototype is divided into two major applications: the resource provider application and the client application. The resource provider application consists of three main system components: listener component, registration component and resource component. The respective tasks of these components are to listen for any incoming connection from the client, to automatically create client profile and capability token, to securely exchange and host multiple resources. In contrast, client application is primarily utilized by resource consumer to securely register and access the hosted resources. Further explanation of each of the components that builds up our prototype architecture is provided.

4.1. Resource Provider Architecture

4.1.1. Listener Server Component

The main functionality of listener server component is to accept any incoming HTTP requests from DE client members. Three main client requests on which this component handles are client registration, provider key signing and provider key retrieval. Upon receiving a client request, this listener component analyses the header of the incoming HTTP connection. This process has an objective to determine the nature of client request. Our client component, which would be discussed in detail in the following sub-section, was able to create unique HTTP headers to identify the objective of each request. **Figure 4** explains the activity workflow that reflects full functionalities of this listener server component.

In a case where an incoming HTTP request contains a registration header, the listener component constructs a certificate object that contains resource provider information and its public key. This certificate object is then sent to the client together with the registration page URL for redirection purposes. When client receives a token, he may verify the certificate to ensure the trustworthiness of resource provider and subsequently, he is redirected to the registration page. In a case where an incoming HTTP request contains a signed key header, the listener component sends provider public key to the client for signing process. We implemented a transaction lock on which other clients are not able to retrieve provider key during the signing process. Further, a configurable timeout of 5 minutes were adopted for each signing process. If the timeout is reached and client has not returned the signed key, the transaction lock will be released. In the last case where the client request is to retrieve provider key, this listener component would response back with provider public key.

When starting the listener server component, the system administrator would need to configure three URL addresses: URL address where the listener server is located, URL address of registration page component, and URL address of resource login component. These URLs are used by the clients to connect to the resource provider server and its resources, and for the listener server to redirect the client to the registration components. Another functionality of the listener server component is to generate both public and private keys. The KeyManager module of this component presents the existing resource provider public and private keys. It has a functionality to re-create both public and private keys. If the new keys are created, these keys will be stored securely in the key repository on the resource provider server.

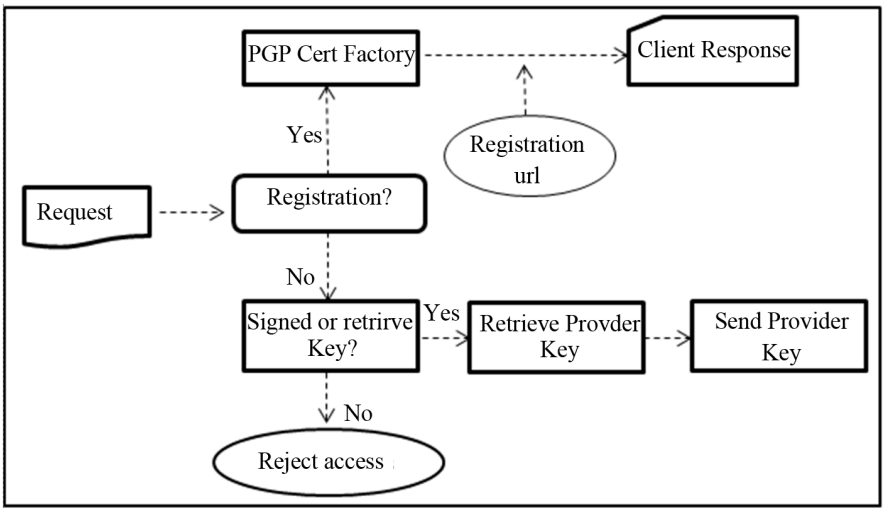


Figure 4. Listener server activity workflow.

4.1.2. Registration Page Component

The registration page component contains a set of minimum information which needs to be filled in by the client. For our testing, we uploaded 4 resources on which the access permissions of each resource could be requested. In real implementation, types of resources and their access permissions are highly dependent on the configuration that is set by each resource provider. During the registration process, client public key is obtained and stored by resource provider in its server repository. The obtained public key will be used in future access requests. That is to verify the signature of the presented capability token. When a client submits his information and indicates which resources that he intends to consume the registration page component then creates a client profile based on the supplied information. This client profile will be stored in resource provider's database. After the creation of client profile, this component constructs a capability token and generates a hash no of this token. The hash no is then stored in the database for subsequent resource access verification. The newly created capability token goes through encryption and signing process (by calling the *EncrypAndSignToken* method). **Figure 5** demonstrates the capability token that was taken before and after the encryption and signing process.

4.1.3. Resource Page Component

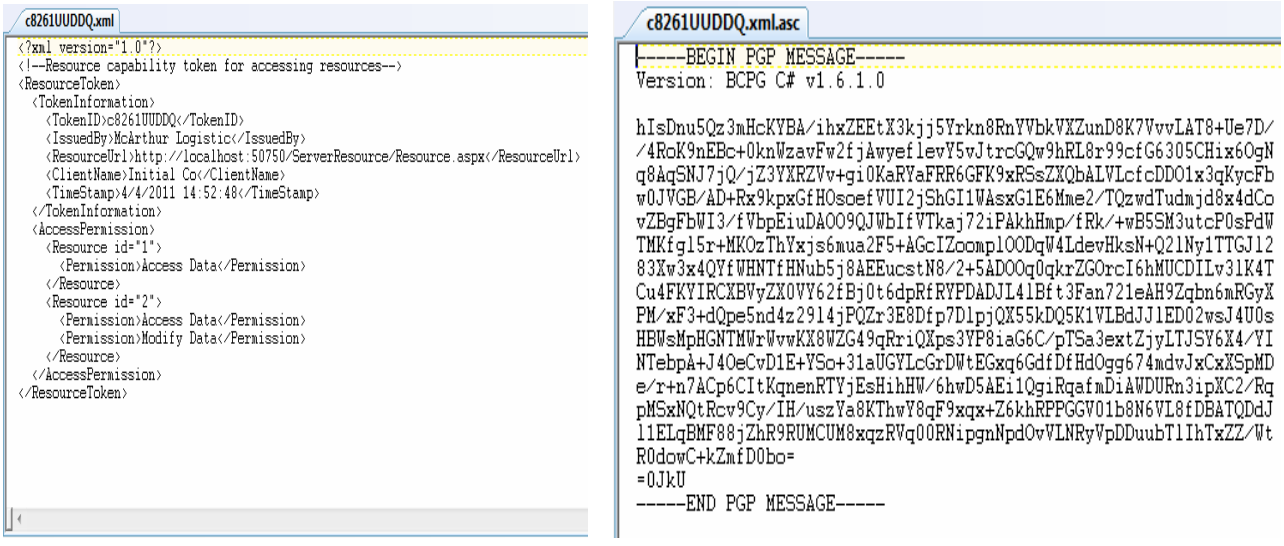
This is the critical component where the resources and critical information are hosted. Therefore, a considerable amount of security must be implemented. When this component is requested by client, a series of validation checks are conducted to determine the availability of capability token, its originality and integrity. This component contains a login module that performs these checks. When resource component receives a HTTP request with access header, login module checks the availability of capability token in the request. It obtains client capability token from the incoming request. The module then calls the decryptFile and the verifySignature method to decrypt and verify the signature as showed in **Figure 3**. If any of these processes fails, access request will be rejected as a failed authentication process, otherwise the client will be redirected to the resource page component where he will be given access based on the permissions that is contained in his capability token.

Note that, when a client is able to access the resource page component, it means that he has been authenticated by the resource provider as a genuine client. However, at this stage the access permissions which are presented by the client in his capability token have not been authorised. A hash no verification of capability token is performed for this purpose (through createVerifyHash module). The hash no will be compared with the hash no that was obtained during the registration process. If hash no verification succeeds, the resource page component retrieves all access permissions from client's capability token, and it further granted the resource access based on these listed permissions. If the verification fails, a notification would be presented to the client and access to the resource would be disallowed.

4.2. Client Architecture

4.2.1. Key Admin Component

Similar to the KeyManager module of resource provider listener component, this component allow clients to generate, manage and distribute his public and private keys. Any created keys would be stored in keys repository



(a) (b)

Figure 5. The original (a) and the encrypted-signed (b) capability token.

which is kept in client workstation. This component was proved to be very useful for client administrator to manage his own keys.

4.2.2. Client Registration Component

This component is a critical module for a client to Register for new resources. A client provides his intended resource provider listener server URL and its port for a secure communication. When a client registers for resource, a new HttpWebRequest will be created and sent to the resource provider listener server. This HttpWebRequest serves as a request message from client to listener component to process and return the provider's certificate and its registration page URL. Upon receiving the response, the component processes and de-serializes this response. in order to obtain the certificate object. The requested client would then be able to view the WoT certificate, and simultaneously he is redirected to the resource provider registration page. **Figure 6** shows our implementation of client registration component.

We built this secure component in a windows application with a simple web browser interface. This component is equipped with a capability to generate a HTTP protocol request for its initial communication. We set a temporary listener which is started when the client clicks on "register" button. This objective of temporary listener is to receive the encrypted token and further decrypts the token with resource provider public key which is obtained from its certificate object. The signature verification process follows after the decryption process. **Figure 7** explains the entire activity workflow during resource registration from the client node point of view.

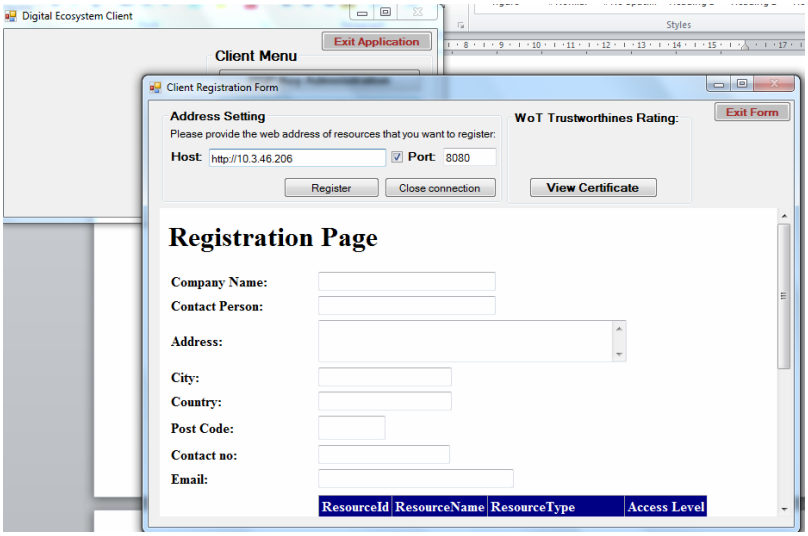


Figure 6. Client registration component.

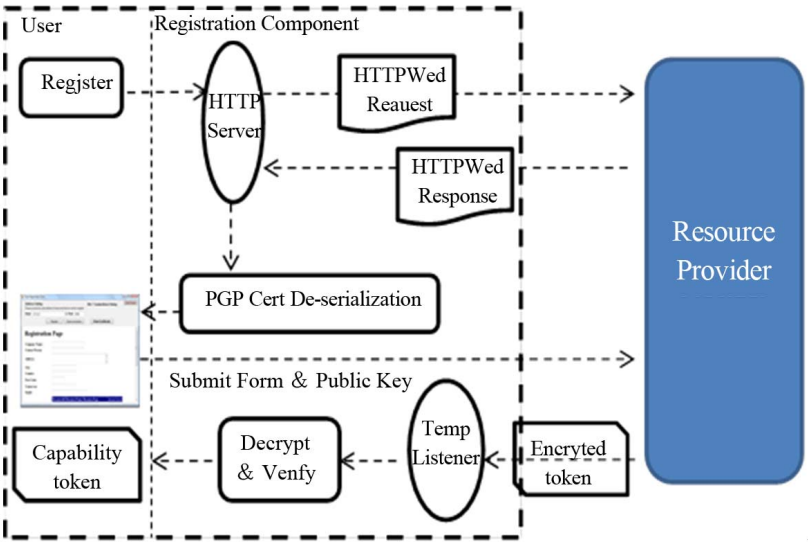


Figure 7. Activity workflow in client component.

4.2.3. Resource Access Component

DE client uses this component to access resources that are hosted in resource provider server. This component establishes communication and exchange processes of capability token and resources. A client is able to select which resource provider that he wants to access the resources based on his retrieved capability tokens. The component also lists all resources and their respective granted permissions. The lists of resource providers, resources, and permission types are retrieved from all capability tokens that are stored in client repository. When a client submits an access request, the capability token of its resource provider is retrieved and a symmetric key passphrase is generated. A temporary object will then be created to encapsulate this capability token and symmetric key passphrase. This object is further encrypted and signed as discussed previously in **Figure 3**.

5. Security Validation & Analysis

In order to validate our proposed solution, we developed several scenarios that showed various unauthorised attempts to access the resources. In each scenario, we focussed at how the existing prototype could mitigate any threats that were attempting to access the resources. The test outcomes of these scenarios would further attest the ability of our proposed solution to uphold the confidentiality, integrity and availability of resources and information while strengthening the authentication and authorisation mechanisms in a DE environment. These scenarios are detailed below:

- **Scenario 1:** If a capability token is stolen, would the unauthorised entity be able to read its content?

In our solution, a capability token is always encrypted and signed before it is being transferred in the network. The strongest encryption algorithm for e-commerce transactions (RSA algorithm) is utilized to prevent the unauthorised entity to read the content of capability token. Moreover, the Public Key Infrastructure (PKI) is fully utilized in our solution. With these measures in place, the confidentiality of capability token is always enforced. The unauthorised entity is only able to read the scramble data unless it has the corresponding private key to decrypt it. **Figure 7** above has showed the encrypted capability token before it is being sent in the network.

- **Scenario 2:** If an unauthorised entity attempts to use the stolen capability token to camouflage himself as a legitimate client, would he be able to access the resources?

In this situation, the unauthorised entity would not be allowed to access the resource for two reasons:

1) Resource provider always verifies the signature of capability token with the client public key on every access requests. Unless the unauthorised entity successfully obtained the client private key, the verification process of this capability token will be failed, and access will not be granted. **Figure 8** shows the failed verification processes of capability token signature.

2) In an event when an unauthorised entity performs the man-in-middle or relay attack during an active communication between client and resource provider, he would be able to obtain the capability token. However, the unauthorised entity will still not be able to access the resource due to the symmetric key passphrase that is shared between the legitimate client and resource provider only.

- **Scenario 3:** If an unauthorised entity or client modifies the capability token by adding or removing the resources or permission types, would he be able to consume the resources?

We implement a hashing mechanism for the resource provider to check the integrity of capability token. If there is any change, either minor or major change, in the capability token, the hash verification process will fail and the resource access will not be granted, as shown in **Figure 9**. This verification process further upholds the integrity of a capability token before any access to the resources is given.

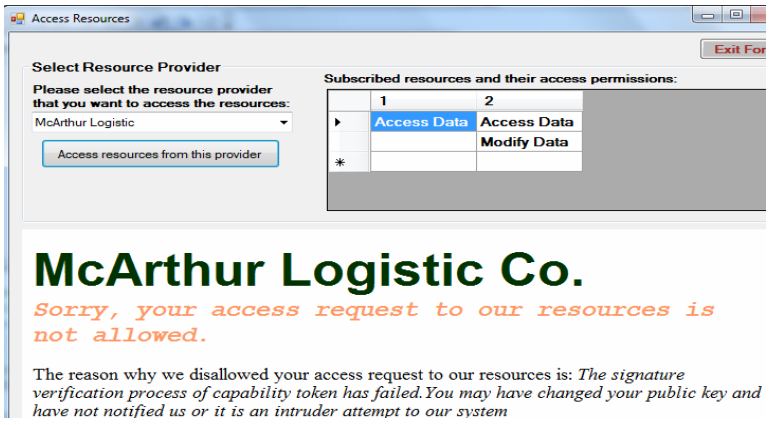


Figure 8. Access disallowed as failure to verify token signature.

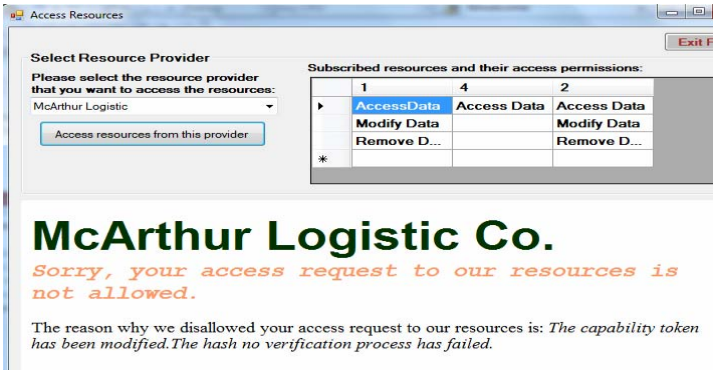


Figure 9. Access disallowed as failure to verify the hash no.

- **Scenario 4:** It is possible that a client is redirected to the unauthorized sites that claim to be the legitimate resource provider, how would DRPM prevent such situation?

This situation is known as phishing attack on which the client is redirected to the fraudulent link where he enters the personal details to the fraudster without any knowledge that he is the victim of the crime [31]. Our proposed capability token contains a ResourceURL field which functions to automatically redirect the client to the legitimate resource provider website. Further, the encryption method that is implemented before any token transfer is performed further upholds the confidentiality of a capability token during the transfer process. Therefore, our solution further reduces the possibility of phishing attack in a DE environment.

The verification and evaluation of DRPM proved to be successful, and it further resolved our main research issue of "authenticating the genuine client and managing multiple authorisation permissions over various resources". Our prototype implementation and testing has proved to solve several security concerns which were derived in section V as well as various threat scenarios that are likely to occur in a DE environment. Finally, the unique authentication and authorisation mechanisms offered by DRPM in conjunction with its secure architecture provide a complete and full fledge security solutions for a DE environment.

6. Performance and Scalability Analysis

6.1. Performance Testing

In this section, we briefly review our analysis on the computational cost of DRPM prototype. To measure the performance impact of DRPM, three different scenarios from the workflows are identified. They are: the overhead cost during initial registration process where provider's public key and its registration page are sent to the client, the overhead cost in processing client's registration that includes generating capability token and securing communication, and the overhead cost during resource access workflow. In addition, we compare the overhead cost of cryptography between two sets of machine.

All performance tests were conducted in two machines, and these machines respectively act as client and server. Our client machine run on a 1.86 GHz Intel Centrino with 1 Gb of RAM while our provider server machine run on a 2.26 GHz Intel Core2 Duo processor with 2 Gb of RAM and IIS 7 web server. Both machines were using NET framework 2.0 and MSSQL Server 2008 for the database. It is important to note that both client and provider server machines are below today's average of client and server computer standards. This configuration was taken to measure the performance of using DRPM prototype on the low end machines. As both client and provider server were connected via local high speed Ethernet LAN with 100 Mbps, we assume that the network latency during testing was insignificant. Therefore, it was not considered on the final performance results.

We conducted 52 repeated experiments for each scenario to obtain various performance results of DRPM. The result of the first two results was omitted due to the need for the application prototype to be compiled and the process workers to be started by the Just in Time (JIT) compiler in Net Framework 2.0. Therefore, only the results of subsequent 50 experiments were obtained. Due to the length limitation of the paper, we only present the mean and standard deviation of the experiment results. Further, the overhead costs derived from each scenario are calculated by applying the mean scores to the formula below:

$$\text{Overhead Cost}: \frac{\text{Processing Time}}{\text{Total Time Taken}} \times 100\%$$

The result of our performance testing on DRPM prototype is shown in **Table 1**.

The average total time take for initial registration reest was 226.18 milliseconds. This timing accounted for loading registration page and processing provider cer-

ficate. Furthermore, the average total time for client registration in our test was 593.24 milliseconds. This timing accounted for 134.56 milliseconds of server process and 458.68 milliseconds of client process. Our findings showed that token creation process accounted was accounted for the lowest time needed compare to other server processes such as encryption and signing process. The highest overhead cost (61.88%) was needed by other server process such as running complex stored procedures in database, forming response to client, and etc. In client process, the highest overhead cost was token decryption process. It was followed by signature verification process which was almost 88% faster than the former process. The rest client process accounted for 31.05 milliseconds (6.77%) was for processing response, storing token to repository, and etc.

As shown in **Table 2**, the overall time needed for accessing resource in our test was 701.28 milliseconds. This timing was account for 55.92 milliseconds for server login process and 645.36 milliseconds for client process and resource page loading. The result for server login process showed that the decryption process took the majority of server login time. This was followed by signature verification which was almost 88% faster than the decryption process aligned with the registration workflow decryption-signature process. Overall time take to retrieve the access permissions from token was 1.46 milliseconds for 1 Kb token size. As the capability token is intended for storing the access permissions in lightweight XML format, we expect that it only contains limited information; therefore its file size should not over than 5 Kb. Hash verification performance was a surprising result in our DRPM prototype testing. This was due to the process of hash verification method which involves generating hash no from the token (using SHA-1 algorithm) and comparing it with the original hash that is obtained from the database. The rest 5.34 milliseconds (9.54%) was needed for other server login processes such as reading private keys from key ring, forming HttpWebResponse message, writing logs, creating sessions and etc.

Table 1. Registration workflow overhead cost (in milliseconds).

Scenario	Steps on Workflow	Processing Type	Mean	Standard Deviation	Overhead Cost
Registration: initial request	Server (Step 1)	Loading Page	226.18	89.08	N/A
Registration: processing	Server (Steps 2-6)	Creating Token	6.84	6.89	5.08%
		Encrypt and Sign	44.46	6.84	33.04%
		Total Server Time	134.56	48.17	N/A
	Client (Steps 7-8)	Decrypt	375.78	46.19	81.92%
		Verify Signature	51.86	46.77	11.31%
		Total Time	458.68	55.32	N/A

Table 2. Resource access workflow overhead cost (in milliseconds).

Scenario	**Steps on Workflow**	**Processing Type**	**Mean**	**Standard Deviation**	**Overhead Cost**
Resource Access	Server (Steps 4-6)	Decrypt	43.48	4.04	77.75%
		Verify Signature	5.08	1.90	9.08%
		Verify Hash	0.52	0.71	0.93%
		Retrieve Permissions From token	1.5	0.50	2.68%
		Total Server Time	55.92	5.15	N/A
	Client (Steps 1-3)	Encrypt & Sign	435.84	55.75	67.53%
		Total Client Time	645.36	70.15	N/A

Token encryption and signature process accounts for the majority of total client process time. The rest 209.52 milliseconds (32.47%) was needed for other client processes including the activity to form HttpWebRequest, to retrieve token from repository, to process HttpWebResponse that was obtained from server, to redirect to resource page, to load resource page from server side and etc. Another finding from our testing was the overhead cost of cryptography processes which include decryption, encryption, sign and verification was reduced by almost 89% by doubling up the machine specification. This could be found by comparing any cryptography process between the client machine and server machine. **Figure 10** shows the overhead percentage of each process.

6.2. Scalability Testing

We tested the scalability of the listener server component to handle multiple HttpWebRequest requests as shown **Figure 11**. This test was conducted by utilizing the Apache JMeter [32] tool that specializes on the web scalability and performance testing. The purpose of this test was to review the scalability of the listener component to handle multiple client registration requests.

In our test bed, 1000 users were generated to access the listener component concurrently. Our test shows that the average elapsed time of this set of results was 162 milliseconds with the aggregate highest elapsed time of 327 milliseconds and the aggregate lowest elapsed time of 5 milliseconds. The scalability testing on other components such as registration, login, resource page, etc. were not able to be conducted by JMeter. This is primarily due to the need to utilize the prototype to create the request stream that holds the encrypted and signed capability token. We tested the scalability of other components only with a small number of machines (5 or less) and only a small number of clients (4 or less). Therefore, scalability of these components was not addressed in the experiment.

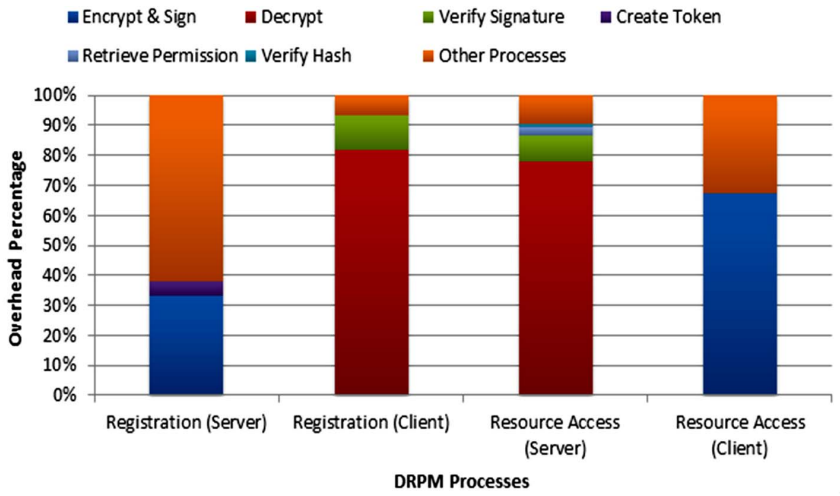


Figure 10. Overhead cost on DRPM prototype.

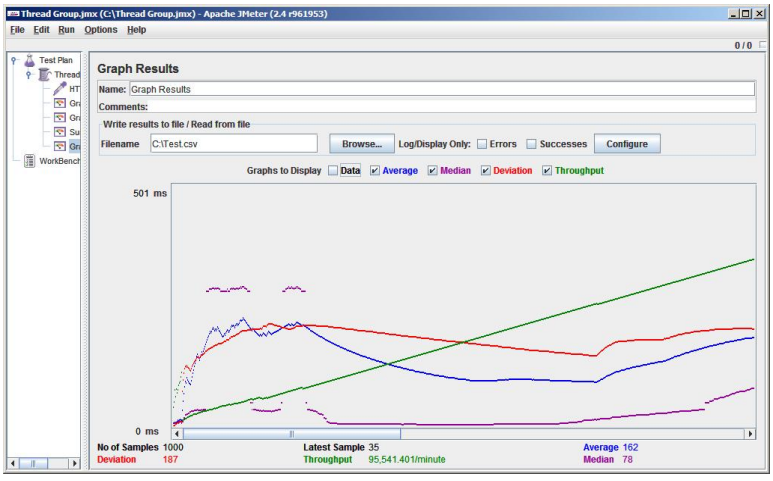


Figure 11. DRPM listener component scalability testing.

7. Conclusions

In this paper, we introduced the concept of Digital Ecosystem (DE), and we also outlined the challenges that are faced by DE, in particular the authentication and authorisation mechanism. We have also highlighted the requirement of DE to minimize the centralized structure and promoting the engagement of Small and Medium Enterprises (SMEs) for its successful adoption. We further propose a unique solution the Distributed Resource Protection Mechanism (DRPM) that attempt to solve the identified challenges. DRPM focuses on the creation of client profile and capability token to protect the critical resources. Moreover, we propose a secure DRPM architecture with adoption of Public Key Infrastructure (PKI). This further allows DRPM functions as a complete and full fledge security solution for Digital Ecosystem.

This paper also provides our implementation of secure DRPM which consists of two major system architectures: resource provider system and client system. Several scenarios of security threats are provided in this paper to analyse and evaluate the ability of our secure DRPM implementation to mitigate these threats. The analysis shows that the secure DRPM proposal is able to uphold the confidentiality, integrity, and non-repudiation of the resources. In addition, the performance and scalability analysis of DRPM are presented. Future works include the investigation of an effective trust mechanism in DRPM. As pointed in this paper, the utilisation of Web of Trust (WoT) would engage the interacting entities to actively protect their DE environment. At this point, however, DRPM does not include any trust mechanism that could allow the interacting entities to determine which resource providers are honest and which are not. The inclusivity of trust mechanism would significantly improve the overall security protection in DE environment. Therefore, investigation on the effective trust mechanism to improve the overall DRPM security is needed.

REFERENCES

[1] D. Boughaci and H. Drias, "A Secure E-Transaction model for E-Commerce," *IEEE GCC Conference* (*GCC*), Manama, 20-22 March 2006. doi:10.1109/IEEEGCC.2006.5686242

[2] C. Potter and A. Beard, "Information Security Breaches Survey 2010," *Technical Report*, PricewaterhouseCoopers, 2010.

[3] P. ltner and T. Grechenig, "A Joint Infrastructure of 'Digital Corporate Organisms' as Facilitator for a Virtual Digital Retail Ecosystem," 4*th IEEE International Conference on Digital Ecosystems and Technologies* (*DEST*), Dubai, 12-15 April 2010.

[4] Y.-L. Fang B. Han and Y.-B. Li, "Research and Implementation of Key Technology Based on Internet Encryption and Authentication," *International Conference on Networking and Digital Society* (*ICNDS*'09), Guiyang, 30-31 May 2009.

[5] X. Tian and W. Dai, "Study on Information Management and Security of E-commerce System," *International Symposium on Intelligence Information Processing and Trusted Computing* (*IPTC*), Huanggang, 18-20 November 2010.

[6] P. Dini, M. Darking, N. Rathbone, M. Vidal, P. Hernandez, P. Ferronato, G. Briscoe and S. Hendryx, "The Digital Ecosystems Research Vision: 2010 and Beyond," 2011. www.digital-ecosystems.org/events/2005.05/de_position_paper_vf.pdf

[7] EComm, "Technologies for Digital Ecosystems," 2005. www.digital-ecosystems.org/doc/flyer-de-sector.doc.

[8] F. Nachira, P. Dini and A. Nicolai, "A Network of Digital Business Ecosystems for Europe: Roots, Processes and Perspectives," 2011. http://www.digital-ecosystems.org/book/DBE-2007.pdf

[9] F. Nachira, E. Chiozza, H. Ihonen, M. Manzoni and F. Cunningham, "Towards a Network of Digital Business Ecosystems Fostering the Local Development," *Discussion Paper*, Bruxelles, 2002.

[10] H. Boley and E. Chang, "Digital Ecosystem: Principles and Semantics," *Inaugural IEEE International Conference on Digital Ecosystems and Technologies* (*IEEE DEST* 2007), Cairns, 12-23 February 2007.

[11] M. Hadzic, E. Chang and T. Dillon, "Methodology Framework for the Design of Digital Ecosystem," *ISIC IEEE International Conference Systems, Man and Cybernetics*, Montreal, 7-10 October 2007.

[12] DigitalEcosystem.org, "Digital Ecosystems: The New Global Commons for SMEs and Local Growth," 2011. http://www.digital-ecosystems.org/de/refs/ref_books.html

[13] G. Briscoe and P. Wilde, "Digital Ecosystems: Evolving Service-Oriented Architectures," 1*st International Conference on Bio Inspired Models of Network, Information and Computing Systems*, New York, June 2006.

[14] M. Wooldridge, "Introduction to MultiAgent Systems," Wiley, New York, 2002

[15] E. Cantu-Paz, "A Survey of Parallel Genetic Algorithms," *Reseaux et Systemes Repartis, Calculateurs Paralleles*, Vol. 10, 1998, pp. 141-171.

[16] J. Stender, "Parallel Genetic Algorithms: Theory and Applications," IOS Press, Amsterdam, 1993.

[17] A. Berson, "Client/Server Architecture," 2nd Edition, McGraw-Hill Companies, Upper Saddle River, 1996.

[18] W3C, "Web Services Architecture," 2011. http://www.w3.org/TR/ws-arch/

[19] R. Schollmeier, "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications," *Proceedings of* 1*st IEEE International Conference on Peer to Peer Computing*, Linkoping, 27-29 August 2001.

[20] H. Koshutanski, *et al*., "Distributed Identity Management Model for Digital Ecosystems," *International Conference*

on Emerging Security Information, Systems and Technologies (*Securware*'07), Valencia, 14-20 October 2007.

[21] J. M. Seigneur, "Demonstration of Security through Collaborative in Digital Business Ecosystem," *Proceedings of the IEEE SECOVAL Workshop*, Athens, September 2005.

[22] J. Novotny, "An Online Credential Repository for the Grid: MyProxy," *Proceedings of the IEEE* 10*th International Symposium on High Performance Distributed Computing* (*HPDC*-10), San Fransisco, 7-9 September 2001.

[23] L. Pearlman, *et al*., "A Community Authorization Service for Group Collaboration," *Proceedings of the* 3*rd International Workshop on Policies for Distributed Systems and Networks*, Monterey, 5-7 June 2002. doi:10.1109/POLICY.2002.1011293

[24] M. Thompson, *et al*., "Certificate-Based Access Control for Widely Distributed Resources," *Proceedings of the* 8*th Conference on USENIX Security Symposium*, Washington DC, 23-26 August 1999.

[25] J. Weise, "Public Key Infrastructure Overview," Sun Microsystem, Sun BluePrints Online 2001.

[26] I. Pranata, G. Skinner and R. Athauda, "A Survey on the Security Requirements for a Successful Adoption of Digital Ecosystem Environment," 2*nd International Conference on Information Technology Security* (*ITS* 2011), Singapore, 24-25 November 2011.

[27] I. Pranata, G. Skinner and R. Athauda, "Community Based Authentication and Authorisation Mechanism for Digital Ecosystem," 5*th IEEE International Conference on Digital Ecosystems and Technologies* (*DEST*'11), Seoul, 31 May-3 June 2011.

[28] I. Pranata, G. Skinner and R. Athauda, "Distributed Mechanism for Protecting Resources in a Newly Emerged Digital Ecosystem," 11*th International Conference on Algorithms and Architectures for Parallel Processing* (*ICA3PP* 2011), Melbourne, 24-26 October 2011.

[29] W3C, "Extensible Markup Language (XML)," 2011. http://www.w3.org/XML/

[30] P. R. Zimmermann, "The Official PGP User's Guide," MIT Press, Cambridge, 1995.

[31] FTSC, "FSTC Counter-Phising Initiative," Financial Services Technology Consortium, New York, 2004.

[32] Apache JMeter, 2011. http://jakarta.apache.org/ jmeter/

