A Flexible Authentication and Authorisation Mechanism for Securing Transactions in Digital Ecosystem

Ilung Pranata  
Faculty of Science and Information Technology  
University of Newcastle  
Callaghan, NSW  
Australia  
Ilung.Pranata@newcastle.edu.au

Geoff Skinner  
Faculty of Science and Information Technology  
University of Newcastle  
Callaghan, NSW  
Australia  
Geoff.Skinner@newcastle.edu.au

Rukshan Athauda  
Faculty of Science and Information Technology  
University of Newcastle  
Callaghan, NSW  
Australia  
Rukshan.Athauda@newcastle.edu.au

Abstract—This paper firstly introduces the newly emerged collaborative technology termed as Digital Ecosystem. The concept of Digital Ecosystem and the existing research in this area are discussed in detail. Next, the paper outlines several issues that need to be solved for a successful implementation of Digital Ecosystem environment. As the main focus of this paper is in the security area, we have identified several requirements to derive a secure Digital Ecosystem. Unfortunately, there is an apparent lack of research in Digital Ecosystem security mechanisms due to its newly emerged technology. Our review on the existing protection mechanisms from the established technology such as peer-to-peer, grid computing, and client-server appears not to meet the Digital Ecosystem requirements. Therefore, we propose our unique solution to enable a flexible authentication and authorisation mechanism for Digital Ecosystem with the aim to provide a rigorous protection in maintaining the confidentiality and integrity of the resources.

Keywords—Security, Authentication, Authorisation, Distributed, Digital Ecosystem

I. INTRODUCTION

Since its first introduction, a new emerging concept of Digital Ecosystem (DE) has grasped numerous attentions from researchers, businesses, ICT professionals and communities around the world. This concept aimed at achieving a set of predetermined goals of Lisbon summit in March 2000 which primarily focuses on dynamic formation of knowledge based economy [1]. Further, the knowledge based economy will lead to a creation of more jobs and a greater social inclusion in sustaining the world economic growth [2]. DE is a multi-dimensional concept that encompasses several current technology models such as collaborative environment [3], distributed system [4], and grid technology [5]. The combination of concepts from these models provides the ability for a DE environment to deliver an open, flexible and loosely coupled resource sharing environment. On the other hand, this combination also develops several complicated security issues which need to be addressed before the full implementation of a DE concept. Unfortunately, the evaluation on DE security dimensions from the current literature signifies a number of deficiencies in its security architecture particularly in protecting the enterprise resources and information. There is a need for a comprehensive resource protection solution that is able to provide a strong and rigorous mechanism to safeguard the critical resources and further to reduce the possibility of information leakage to the unauthorized parties.

A key challenge for enterprises who involved in a DE environment is to determine the right users who are able to access the services, resources, knowledge and information hosted by these enterprises. This challenge is occurred due to several reasons. First, the occurrences of multiple resources published and shared by each enterprise in a DE environment and second, the situation where various clients are able to access each individual resource. Due to these reasons, enterprises urgently need a mechanism that effectively manages their clients’ access control and authorization permissions with an aim to protect their resources. In this paper, we attempt to deliver a comprehensive framework allowing enterprises to protect their resources and information from any unauthorized use.

The remainder of this paper is organised as follow: Section II discusses the notion of Digital Ecosystem in detail. Section III lists the requirements for a secure Digital Ecosystem. This is followed by Section IV that proposes our unique solution to provide a rigorous protection for DE environment and section V that provides the conceptual validation of our solution. Section VI and VII respectively details the prototype implementation and security analysis of our solution. It then followed by section VIII that provide the performance and scalability testing of our prototype solution. Finally, section IX concludes our present work and provides several ideas for future work.

II. DIGITAL ECOSYSTEM ENVIRONMENT

In [6], we introduced the concept of Digital Ecosystem and its security requirements. The following sub-sections reiterate our vision on DE technology, and this is followed by a section that details its security requirements. We take two approaches in explaining the concept. First, we define Digital Ecosystem from research point of view. Second, we provide several real world examples that mimic its characteristics.

A. From Research Perspective

Since its first inception, the newly emerging concept of Digital Ecosystem has received increasing attention from researchers, businesses, IT professionals and communities around the world. A wide variety of researches and initiatives have been undertaken aimed at the realization and
implementation of a Digital Ecosystem concept. The enthusiastic efforts to realize the implementation of Digital Ecosystem. Although literature attempts to derive the notion of Digital Ecosystem encompasses some of the main characteristics of Digital Ecosystem [2, 8] are openness, interaction, balance, domain-clustered, loosely coupled, structural coupling, self-autonomous and evolutionary.

To date, the number of research has been progressed to realize the implementation of Digital Ecosystem environment. In Europe, the RЕgions for Digital Ecosystem Network (REDEN) [19] was established to provide the sustainable regional development of DE. Further, the Digital Ecosystems Network of regions for (4) DissEmination and Knowledge Deployment (DEN4DEK) [20] provides a thematic network to share experiences and disseminate all necessary knowledge that will allow the European regions to plan an effective deployment of Digital Ecosystems at all levels (economic, social, technical and political). A number of Professional and academic researchers have also focussed their ongoing research on Digital Ecosystem semantic and ontology [8, 21], technical infrastructure [22, 23], knowledge sharing [24], security [25, 26], etc. This ongoing research shows strong efforts to realize the implementation of Digital Ecosystem.

B. Analogy From Real World Example

To summarize the discussion from previous sub-section, we envisioned the concept of Digital Ecosystem as a distributed open network that allows data, information, knowledge, resources, etc., to be exchanged freely. Further, they can be interpreted to provide a meaning for their users. In Digital Ecosystem, the free exchange flow of data and the availability of users to interpret data drive the livelihood of its entire environment. Data can be ranging from financial transactions, music, pictures, videos, applications, etc. while the users can be persons or machines that transform and extrapolate data into something meaningful. A characteristic that differentiates Digital Ecosystem from existing technology concepts is its evolutionary mechanism. That is, when data is no longer appropriate or relevant or when the users are no longer available, they will slowly disappear from the ecosystem.

One real world example that mimics the characteristics of Digital Ecosystem is the Apple application store [27]. Apple Inc. has successfully created an ecosystem for its online store that allows the developers to develop applications and the users to buy and use their applications. This ecosystem removes the barrier for Apple’s customers to exchange their products in a low setup cost. Further, each of the application is autonomous while providing the opportunity to collaborate. It could further be inferred from Apple store example that the data is the exchanged applications, files, etc., while the users are the Apple customers that access these applications, files and etc. using Apple devices. Several examples of Digital Ecosystem can also be found in the social media sites such as Facebook, Twitter, etc. Those sites allow the frictionless flow of data that allows the users to interact and interpret the data.

Although one of main characteristics of Digital Ecosystem is free flow of data, the flow of data is much stronger within its own domain while less strong between multiple domains. This means the shared data in one domain may not be as relevant as it is in another domain, although a possibility that data is exchanged to other domains may occur. The real
world example for this case can be found in the community forum. Consider that each forum topic is a domain by itself and a registered user may post threads on multiple forum topics. Most information that is posted in a “movies” forum topic may not be relevant for “finance” forum topic. However, some information could still be shared between these two topics, e.g. revenue generated from a particular movie. Therefore, Digital Ecosystem supports multi-domains environment on which a stronger connection is achieved within its own domain. Figure 1 shows a graph that illustrates Digital Ecosystem environment and its flow of data.

Fig. 1 An illustration of a Digital Ecosystem Environment

Although several real world examples mimic the characteristic of Digital Ecosystem, its objective for stimulating business growth for SMEs (Small and Medium Enterprises) is still not achieved. The vision of Digital Ecosystem focuses at the development of structural frameworks, architectures, and mechanisms that allows SMEs to freely exchange their data, information and resources. This can be imagined as applying those aforementioned examples for the businesses to freely find the expertise that they need, to collaborate together, and to interact and sharing their data. For example: the manufacturing enterprise would be able to find and collaborate with the distributors and retailers to advance their businesses together, the marketing department would be able to use SEO (Search Engine Optimization) services provided by IT provider while getting market research from research company, etc.

III. REQUIREMENTS FOR A SECURE DIGITAL ECOSYSTEM

A successful adoption of Digital Ecosystem depends on a number of factors. They are ranging from flexible architecture that enables intense interactions, language translation and technology enablement for constructing a complex and evolutionary system, model for information and knowledge sharing, and effective security mechanism that strengthen data and information protection. It is important to note that each of these factors requires significant research to address its associated issues, and only after these issues are solved, the implementation of Digital Ecosystem can be successful. In this paper, we put our main focus to discuss several security issues that pertain to Digital Ecosystem. Moreover, we have identified the security requirements that need to be met before the successful adoption of a Digital Ecosystem environment can be realized.

A. Distributed Authentication Mechanism

Authentication has been long perceived as an enabler for any online transactions, and it is performed to restrict data access. Authors in [28] define authentication as “the process by which a system verifies that users are who or what they declare themselves to be”. It has been evidential that authentication is a de-facto requirement for securing any online data and information from the unauthorized users. The importance of authentication, in turn, also applies to Digital Ecosystem. With a high volume of data, information, knowledge, and resources being shared in a Digital Ecosystem environment, an effective and strong authentication mechanism is required to maintain the data confidentiality.

However, implementing a strong authentication mechanism in Digital Ecosystem is a challenging task. As an open, intensive sharing and evolutionary environment, enterprises in Digital Ecosystem face an extreme issue to protect their resources from malicious entities. This is greatly due to the ability of an entity to utilize a Digital Ecosystem environment without requiring a huge capital. Therefore, such situation provides an advantage for malicious entities to perform prohibited actions, such as attempting to access the critical information. This situation is worsening as only a limited control that could be put to this environment to govern its entities. In addition, as Digital Ecosystem community expands its size to incorporate more entities, the resource or information providers also face a challenge to identify the legal entities that are able to access their resources. A strong authentication mechanism is needed not only to allow the authorized entities to consume the resources, but also to maintain the openness and sharing nature of Digital Ecosystem.

We identify that the biggest challenge to implement authentication mechanism is the distributed nature of Digital Ecosystem. As have been pointed out in section II, Digital Ecosystem is defined by literature as an open distributed environment on which centralized structure and single point failure must be minimized. It is important to note that single point failure would possibly bring down the entire ecosystem where the entities are relied to perform their businesses. This, in fact, would yield a great devastation impact on the organization profits as well as the entire ecosystem economy.

B. Flexible Authorisation Model

While authentication ensures only the genuine entities are able to access the resources, authorization provides the mechanism to restrict entities’ access permissions on the resources. Both authentication and authorization mechanisms must be viewed in a holistic approach for its success implementation [29]. Not viewing these mechanisms as a unified approach may result failure in protecting online transactions. For example, an online business that establishes a unified approach may result failure in protecting online transactions. For example, an online business that establishes a unified approach may result failure in protecting online transactions.
the authorized entity, to access the resources of another entity would be advantageous.

We further identify and derive a list of requirements that we conceive as fundamental for an effective and efficient authentication and authorisation mechanisms for a Digital Ecosystem environment. These requirements are primarily based on the analysis of Digital Ecosystem architecture and the appropriateness of core criteria in deriving the effective and efficient authentication and authorisation mechanisms. The list of requirements in deriving a strong authentication and authorisation mechanism for Digital Ecosystem environment is outlined as follow:

- **Secure** – The mechanism must be computationally strong, unbreakable and reliable. The confidentiality, integrity and availability of exchanged information must always be preserved at all times. User information and security credentials must be protected in trusted repository.

- **Adaptive** – The mechanism must be easily adapted to the frequent change in Digital Ecosystem environment. The mechanism should be able to manage the identification and right assignments by its own. Further, it should allow various rights and permission assignments to multiple clients.

- **Distributed** – The mechanism should be in distributed approach without reliance to single server. Therefore, it limits the possibility of total failure in a highly interactive Digital Ecosystem environment.

- **Delegation** – The mechanism should allow permission delegation from one client to other clients. It must have the capacity to limit the depth of delegation and securely trust the delegated clients.

- **Community Interaction** – The mechanism should promote interaction between different entities in a Digital Ecosystem environment. It should encourage member entities to collaborate in protecting the entire ecosystem.

- **Cost-based** – The mechanism must be efficient and low-cost without a need of extensive resources for both computation and communication. It must align with the goal of Digital Ecosystem which is promoting knowledge economy that is driven by the interaction of SMEs.

### IV. DRPM: DISTRIBUTED RESOURCE PROTECTION MECHANISM

In the previous section, we have identified several requirements for protecting transactions in a Digital Ecosystem environment. We put our main focus in strong yet flexible authentication and authorisation mechanisms to allow only the genuine entities that have access to the resources. In this section, we propose a unique solution to address such identified requirements. This solution is termed as Distributed Resource Protection Mechanism (DRPM) [30, 31]. The following sub-sections detail all crucial elements in our proposed solution.

#### A. Identifying Entity through Client Profile

The present mechanism for service discovery in a DE environment requires a client to search for resources by utilizing a semantic discovery portal through its browsers or rich applications [32]. This discovery portal would search and list all resources which are provided by DE resource providers. Once the client finds the resource, it then contacts the resource provider and requests for that resource. At this stage, resource provider does not know any information about this client and its intended purpose on the resource. This may put the resource at risk as it may contain highly sensitive information which must be protected from any misuse and malicious act. Therefore, it is crucial for a resource provider to understand its client’s information before any access to the resource is granted. Taking this into consideration, we adopted a method of creation for a client profile that aims to capture all required, but voluntarily provided, information about a client. The information which is contained in a client profile provides necessary data about who the client is and about their intentions and purpose for using the requested resources. The aim of implementing a client profile is to ensure the resource provider that resources are not going to the wrong entities and further impose the confidentiality and integrity of the resources.

The use of client profiles also facilitates the auditing process on who is accessing a resource. For example, there may be a situation where a resource provider needs to trace back which client was delegated an access to the resource in case there was an incident involving a dispute or counterfeiting of the resource. In order to fully implement a client profile, it is necessary that a client registration portal is employed in DRPM. A client profile is generated through this registration portal. Further, resource providers are able to customize the registration portal to contain only the information which is important to them. New clients wishing to access a specific resource are initially redirected into this portal. If they wish to access the resource, they must continue to fill in all the necessary information required by the resource provider to produce a client profile. Once it is produced, the client profile is stored in the resource provider repository. Utilisation of this functional procedure and process provides an additional and enhanced method for determining who is accessing a particular resource at a particular time inside a DE environment.

#### B. Storing Permission in a Capability Token

It is always a challenge to enforce client access permissions on the available resources within a DE environment. This challenge is due to the occurrences of a high amount of entities that actively interact in a DE environment. Further, these entities could also make the same request for a particular resource either at the same or at different time. To solve the issue of managing multiple resource access permissions on a diverse range of DE clients, we utilize and further evolve the concept of capability introduced by CAS server [33] that is used in a Collaborative Environment. In CAS, capability is used to store all access rights of a user which are determined by a community policy. However, the implementation of the capability in our framework is slightly different to the capability implementation in a CAS server. In our framework, capability contains all 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 that hosts a particular resource. This capability would be used to grant the client access to the resources, and it further facilitates the authorization process for the clients.

Once a client profile is created, a list of client authorization permissions are assigned into the capability token. The client’s access permissions and policies are expressed in XML [34] due to its simplicity, wide usability and self-descriptive characteristics. Our basic design of a capability token contains the client profile identifier, resource provider
identifier, resource identifier and list of access permissions. A
time-stamp can be implemented in the capability token to
determine the validity period of a client when accessing the
resources. In the event where the trustworthiness of a new
client is equivocal, a short-life capability token can be issued.
Once the trustworthiness of this client 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. Once a capability token is created, it would be
disseminated to the requesting client. Every time a client
makes a request to the resource provider, the client sends
back its initial configured capability to the resource provider.
The resource provider then authenticates the client’s
capability token and grants the access permissions based on
the listed permissions obtained from the client’s capability.

C. Secure DRPM Workflows

Title we present a secure DRPM which provides a strong
authentication and authorization mechanism while upholding
the confidentiality, integrity and non-repudiation of resources.
The following notation will be used to mathematically define
the secure DRPM:

- CI: Client that request for the resources.
- RP: Resource Provider that host the resources.
- PKi: Public Key of i.
- SKi: Secret Key of i.
- Cl: Capability token of client.
- CK: Capability key of i.
- PKrp = Cl cp: The Public Key Certificate of the client.
- ATCl: Authentication Token of client.
- E[Sk(sj)Cl] = E[Si(Clcp)RP]: SHA

Algorithm is used to hash the capability token. This
process enhances the integrity of capability token over the
untrusted network.

1. A new client contacts the resource provider for requesting
a resource (CI → RP). Resource provider sends its WoT
endorsed public key to the client (CI ← RP:{PKrp}). 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
will build a unique client profile which identify the client,
and send this client profile to the repository server.

3. Resource provider then requests for client certificate and
stores the client public key on its repository (Cl:{PKCl} →
RP). 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 (SKrp + Cl cp = Sj(Cljpiky)). SHA
Algorithm is used to hash 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 (PKci +
Sj(Cljpiky) = E[Sj(Cljpiky)]ci) and send it to client end-point
(CI ← RP:{E[Sj(Cljpiky)]ci}).

7. Client uses his own private key to decrypt the encrypted
capability token (E[Sj(Cljpiky)]ci - PKci = Sj(Cljpiky)). This
process further ensures the confidentiality of capability
token. A capability token is breached if client cannot
decrypt the message.

8. Client then uses resource provider public key to generate
the capability token from the signed message (Sj(Cljpiky) -
Pkrp = Cl cp). This process further ensures that the client
receives the capability token from the genuine resource
provider unchanged.

Note that at the final step of registration process, client will
have his capability token and public key which were retrieved
from the resource provider. The capability token and resource
provider public key will then be stored in client repository for
future communication or resource access. 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.

2) Fine-Grained 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 granting the resource access. Resource provider utilizes client’s capability token to authenticate and authorize client 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 4.

The steps are as follow:

1. Client looks at his repository for his intended resource provider capability token. He then retrieves this capability token from client repository. The capability token contains the client access permissions and the resource URL. At this stage, the client also determines a symmetric pass key which will be shared with the resource provider and generate the Authentication Token which consists of symmetric pass key and capability token

\[ (CI_{rp} + S_K_{cl} = AT_{cl}) \]

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

\[ (SK_{cl} + AT_{cl} = S_k(AT_{cl})) \]

The signing process is essential to uphold the non-repudiation of capability token.

3. Client then encrypts the signed capability token using resource provider public key

\[ (PK_{rp} + S_k(AT_{cl}) = E[S_k(AT_{cl})]) \]

and he sends the encrypted message to the resource provider

\[ (CI : E[S_k(AT_{cl})]) \rightarrow (RP) \]

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

\[ E[S_k(AT_{cl})]_{cl} = PK_{rp} = S_k(AT_{cl}) \]

5. Resource provider then verifies the signature of capability token using the client public key

\[ (S_k(CT_{cl}) - PK_{cl} = AT_{cl}) \]

It then verifies the integrity of the capability token by generating the hash number from capability token using the SHA Algorithm.

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 the capability token authentication and authorization processes is valid. In an event where the capability token is stolen due to the man-in-middle attack, the unauthorized 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. If there is a security breach on which resource provider generates a new pair of public-private keys, client would not be able to decrypt using his current resource provider public key. Therefore, a request needs to be made to obtain a new public key.

PKI is extensively utilized during the DRPM resource workflow. The other party public key retained by both client and resource provider during the registration process is re-used to provide the confidentiality and integrity of capability token. PKI is primarily adopted during the initial handshake and capability token transfer. Due to the limitation of PKI which requires higher computation process, we suggest the utilization of symmetric key for transferring the data after the authentication and authorization process. The symmetric key can be incorporated into the capability token message before encrypting with the client’s private key. Resource exchange is then encrypted by this symmetric key over the untrusted network. Further, authorization permissions could be updated by generating new capability token for the client while updating the hash no of the new token.

3) Engaging Community to Protect Digital Ecosystem

Our analysis on several literature has shown that the majority of solutions heavily utilize a central CA or CP. These CA or CP acts as their third party entity that provides the digital certificate for authenticating and trusting the client and resource provider. A certificate is required to certify the authenticity of an entity which is vital before any transaction occurs. In order for a certificate to be accepted, CA must be trusted by both client and resource provider. Although the utilization of CA is essential for the entity verification, several detriments could possibly limit such implementation in a DE environment. First, the implementation of CA creates barriers to entry for the Small and Medium Enterprises (SMEs) due to the cost ineffectiveness and hectic audit process. In its practical implementation, an enterprise is required to pay a yearly subscription fee to the CA provider for each certificate that is issued for single enterprise service. In addition, CA requires a complex audit process to be conducted annually to verify the identity of an enterprise. Due to these reasons, the CA certificate is mainly utilized by a small number of large enterprises while hindering such implementation for the SMEs. Note that, this situation further deviate the core purpose of a DE concept which is to enable a greater involvement of SMEs by allowing SMEs to actively engage, collaborate, and gain advantage from an open socio economic environment [1, 35].

Second, there is a significant risk of single point failure and CA subversion. The availability of the entire DE environment is threatened when a failure occurs on CA. This failure will yield a substantial impact on situations such as when a joining entity requires a certificate or when certificate is expired. On another case of CA subversion, resource protection may be compromised if a hacker is able to obtain an authentic entity certificate from the CA. This was the case that was faced by the Microsoft and VeriSign in 2001 which the unauthorised entity posed as Microsoft employee and requested VeriSign to issue two digital certificates [36]. This
incident which allowed the attacker to post a virus has affected all Microsoft retail and business customers. The similar incident was also transpired recently in 2011 [37] where an attacker whose IP address originated from Iran forcefully requested several fraudulent certificates to be issued by Comodo certification services for the well-known websites, such as Google, Yahoo, Skype, Mozilla, and several others. Therefore, in order to address the previously mentioned concerns, we propose an idea to integrate the community trust services, such as Web of Trust (WoT) into DRPM workflow, particularly in the 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 identification 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) [38], which is used extensively 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.

V. DRPM CONCEPTUAL VALIDATION

Our main objective in securing the DRPM is to ensure the protection of any information and resources, such as capability token that are being transferred over in the network communication channel. This objective further leads to upholding the confidentiality, integrity, and non-repudiation of the transferred information and resources. Our theoretical validation on the aforementioned proposal proves that the objective is achieved. Given that

\[ PK_i \sim SK_i \]

it means that any private key is only work with its corresponding public key and vice versa. Therefore, any resources or information that is being either encrypted or signed by the public or private key of i can only be decrypted or verified by the corresponding public or private key of i only, e.g. on the step 5 and 8 in the registration workflow of the previous section:

\[ SK_{BP} + C_i = S_i(C_{i,RP}) \]; therefore
\[ S_i(C_{i,RP}) - PK_{CI} = C_i \]
\[ S_i(C_{i,RP}) - PK_{RP} = C_{i,RP} \]

It means that the signed capability token using resource provider private key can only be verified using the resource provider public key. The similar principle is also implemented on the encrypted capability token that is shown in step 6 and 7 in the registration workflow of the previous section:

\[ PK_{CI} + S_i(C_{i,RP}) = E[S_i(C_{i,RP})/C_i] \]; therefore
\[ E[S_i(C_{i,RP})/C_i] - PK_{BP} = S_i(C_{i,RP})_{BP} \]

The steps explained in the resource access workflow of the previous section also follow the similar principle to protect their exchanged authentication token.

Hash algorithm is implemented in resource provider end-point to check the integrity of the retrieved capability token from the client during access request. If

\[ HashNo(C_{i,RP})_{BP} \neq HashNo(C_{i,RP})_{Original} \]

resource provider will identify that the original capability token has been modified. It then up to the resource provider policy whether to allow or disallow the resource access. The symmetric key passphrase that is implemented after the resource access is granted is used to encrypt or decrypt the resources:

only \[ E[resource]_{SP} - SK_i \]

As only the resource provider and client that know the symmetric key passphrase, the confidentiality of resource is upheld.

Given that the private key is securely stored in both client and resource provider end-point and sharing of private key is proscribed, any exchange of capability token is practically secured from any unauthorized entities. The only approach to decrypt the capability token in both registration and resource access workflows is by obtaining the private key (SKi). This approach requires the unauthorized entities to hack into client or resource provider server on which we assume the server is secured for this paper. The security protection of servers is not in the scope of this paper. Further, the encryption technique with 128 bits key size which is implemented for both asymmetric and symmetric encryption leads to nearly impossible for the unauthorized entities to perform brute force attack. 128 bits represents 2 to the 128th power, or 3.4 x 1038 which requires 1016 years to break the encryption [39]. Therefore, the confidentiality and non-repudiation of the exchanged capability token is advocated

VI. DRPM IMPLEMENTATION

Our DRPM prototype [40] 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.

A. Resource Provider Architecture

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 5 explains the activity workflow that reflects full functionalities of this listener server component.

![Listener server activity workflow](image)

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.

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 EncryptAndSignToken method). Figure 6 demonstrates the capability token that was taken before and after the encryption and signing process.

![Original and encrypted-sigend capability token](image)

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 4. 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.

B. Client Architecture

1) Key admin component
Similar to the KeyManager module of resource provider listener component, this component allows clients to generate, manage, and distribute their public and private keys. Any created keys would be stored in keys repository which is kept in client workstation. This component was proved to be very useful for client administrator to manage his own keys.

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 the 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 7 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 8 explains the entire activity workflow during resource registration from the client node point of view.

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 4.

VII. 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 6 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.

- **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 [41]. 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.

![Fig. 9 Access disallowed as failure to verify token signature](image)

![Fig. 10 Access disallowed as failure to verify the hash no](image)

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.

**VIII. PERFORMANCE AND SCALABILITY ANALYSIS**

**A. 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 1.86Ghz Intel Centrino with 1Gb of RAM while our provider server machine run on a 2.26Ghz Intel Core2 Duo processor with 2Gb 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 below:

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Steps on Workflow</th>
<th>Processing Type</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Overhead Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registration:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>initial request</td>
<td>Server (Step 1)</td>
<td>Loading Page</td>
<td>226.18</td>
<td>89.08</td>
<td>N/A</td>
</tr>
<tr>
<td>Registration:</td>
<td>Server (Steps 2-4)</td>
<td>Creating Token</td>
<td>6.84</td>
<td>6.89</td>
<td>5.08 %</td>
</tr>
<tr>
<td>processing</td>
<td></td>
<td>Encrypt and Sign</td>
<td>44.46</td>
<td>6.84</td>
<td>33.04 %</td>
</tr>
<tr>
<td>Client (Steps 7-8)</td>
<td>Total Server Time</td>
<td>134.56</td>
<td>48.17</td>
<td>10.78</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Decrypt</td>
<td>375.78</td>
<td>46.19</td>
<td>81.92 %</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Verify Signature</td>
<td>51.86</td>
<td>46.77</td>
<td>11.31 %</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total Time</td>
<td>458.68</td>
<td>55.32</td>
<td>N/A</td>
</tr>
</tbody>
</table>

The result of our performance testing on DRPM prototype is shown below:
The average total time take for initial registration request was 226.18 milliseconds. This timing accounted for loading registration page and processing provider certificate. 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 the token creation process 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.

**TABLE 2. RESOURCE ACCESS WORKFLOW OVERHEAD COST (IN MILLISECONDS)**

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Steps on Workflow</th>
<th>Processing Type</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Overhead Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server (Steps 4-6)</strong></td>
<td>Decrypt</td>
<td>0.42.48</td>
<td>4.04</td>
<td>77.75%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verify Signature</td>
<td>5.08</td>
<td>1.90</td>
<td>9.08%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verify Hash</td>
<td>0.52</td>
<td>0.71</td>
<td>0.93%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Retrieve Permissions From token</td>
<td>1.5</td>
<td>0.50</td>
<td>2.68%</td>
<td></td>
</tr>
<tr>
<td><strong>Total Server Time</strong></td>
<td></td>
<td></td>
<td>55.92</td>
<td>5.15</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Client (Steps 1-3)</strong></td>
<td>Decrypt &amp; Sign</td>
<td>435.84</td>
<td>55.75</td>
<td>67.53%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total Client Time</td>
<td>649.36</td>
<td>70.15</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

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 1Kb 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 5Kb. 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.

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 11 shows the overhead percentage of each process.

**B. Scalability Testing**

We tested the scalability of the listener server component to handle multiple HttpWebRequest requests. This test was conducted by utilizing the Apache JMeter 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.
IX. RELATED WORK

A. Review on the Existing Literature

In this sub-section, we provide a review on the existing authentication and authorisation solutions that has been provided by several literatures. This review further draws the novelty and originality of DRPM in providing a secure protection for Digital Ecosystem. 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. The most prominent mechanism to manage the client credentials is through the implementation of Identity Provider (IdP) or Credential Provider [42, 43]. IdP mainly focuses at storing and providing the client credential to any resource providers for their client authentication process. On every authentication process, resource provider will request the client credential from its trusted IdP where it receives any access request from a client. In latter development, several technology standards such as SAML [44] and Liberty Alliance [45] are adopted in this mechanism to provide the federation mechanism of multiple entities for Single Sign On (SSO) services. Similarly, the Credential Server (CRES) [46] and the Grid Security Infrastructure (GSI) MyProxy [47] utilizes the IdP concept, and they further leverage its concept for large number of dispersed servers over a wide geographical area. Both mechanisms store the clients’ credentials in the local server; however, authentication of a remote client can be facilitated by requesting his credential from the trusted remote server.

In both MyProxy and CRES mechanisms, the resource provider requests the client credential from the local server on every authentication process. The local server then creates a certificate token which contains client information. Subsequently, the certificate token is sent to the resource provider as the acknowledgement of the authentic client. When the resource provider receives the token, it allows the client to access the resources based on the trust established with the publisher of token. Although these approaches could be deployed well in a DE environment, 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, our literature review identifies that several internet authorisation mechanisms take similar approach as its authentication mechanisms. The most prominent authorisation mechanisms, such as CAS [33], Akenti [48], and PMI [49], utilize a central server to assign the multiple access permissions to the clients individually although their implementation are differ between each other. These mechanisms also inherit several issues pertinent to the central management of authorisation permissions. 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. Although it is possible to replicate the central server, the replication process will bring abundance administrative issues, 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 access 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. Fourth, DE consists of multiple entities that form the domains based on their suitability. Literature in DE has strongly highlighted that these entities have the freedom to move between the domains to find its suitable place. Therefore, the implementation of the central point for each domain would significantly limit the freedom of entities to move. This is due to the requirement of the entity to be registered to the central point of each domain and trust must be established between the central point and the entity.

B. Review on Industry Standard: SSL

The SSL/TLS technology [50] has been extensively used in e-commerce transactions for secure authentication and communication. This technology is designed with a highly reliance on the Certificate Authority (CA) to ensure the legitimacy of an entity. As have been discussed in section IV sub-section C3, CA presents a centralized credential management and prone to several significant risks. In this sub-section, we focus on the comparative analysis between SSL and our proposed solution: DRPM. However, such analysis does not discuss the conspicuous single point failure that SSL presents as it has been discussed in the previous sub-section.

Although SSL proves the identity of resource provider and authenticate it to the client, client authentication on resource provider endpoint is limited in SSL. To authenticate client in SSL, client needs to create the public key and request CA to sign its public key. This is essentially a similar process with provider authentication, and it requires client to pay annual cost to the CA to maintain its certificate signature. To avoid such cost for the clients, most resource providers implement other means for authenticating its clients. The most popular solution is to generate a pair of username and password for the clients to remember. Client authentication is then performed by verifying the username and password during each login. The username and password are encrypted by symmetric key that was generated earlier during SSL handshake. DRPM takes different approach in authenticating the client by issuing the capability token and by verifying client signature. A client is authenticated if the presented capability token is genuine and it contains client signature. Figure 13 shows the comparative analysis on processes between SSL and DRPM. In addition, DRPM grants access to the client based the permissions that are contained in the capability token. As this functionality is not performed in SSL, it requires provider to implement other authorisation method such as Access Control List (ACL), Active Directory (AD) and etc.

In SSL, preliminary trust must be established between client and the CA before provider’s certificate is verified. This method further limits the applicability of SSL in multi-
Fig. 13 Comparative analysis between SSL and DRPM.
domains environment such as Digital Ecosystem. If a client would like to consume services from multiple domains, he needs to find the CA from each domain and manually register and trust them. This would increase the overhead when consuming the services and possibly introduce vulnerability. If a rogue CA exists in the domain, client would not be able to know as he has only limited knowledge on that domain. In addition, SSL does not measure the trustworthiness and Quality of Service (QoS) of the provider as its focus is to prove provider’s identity. We propose TIDE [51] to measure the trust and QoS of an entity in a DE environment. TIDE can be adopted to DRPM to provide more comprehensive security protection.

X. CONCLUSION

This paper has introduced the notion of Digital Ecosystem (DE). DE is a loosely-coupled and interactive environment that allows its member entities to share their information and resources in a secure and open manner. It is expected that the implementation of DE would bring a significant advantages for Small and Medium Enterprises (SMEs) which further contributes to the world economy. However, there are a number of issues that need to be solved before a successful implementation of DE, in particular in the security area. In this paper, we have highlighted the requirements for a secure Digital Ecosystem. After a thorough review on the literature, we found a number of deficiencies of the current protection mechanisms to meet DE requirements. Therefore, we propose the Distributed Resource Protection Mechanism (DRPM) for DE to provide a comprehensive resource protection.

DRPM can be classified as a new approach to facilitate the authorization process for enterprises that request for specific resources or information. DRPM emphasizes on the decentralized authorization mechanism that is performed by each resource provider. It is achieved by utilizing the client profile and capability token for its authentication and authorization permissions. Our proposal also incorporates the Web of Trust (WoT) to actively engage the community to protect the resources. The paper also outlined the prototype implementation of our solution together with its security analysis and performance/overhead cost.

Several further works exists to further enhance our proposed solution. The most prominent future work for DRPM is trust. Trust is critical in DRPM to build the confidence of the entities to interact and sharing their resources. Incorporating trust in DRPM would promote extra security to the interacting entities in DE environment. This is particularly important for the resource consumer entity that does not have any past interaction with the resource provider entity. Other future work in DRPM is to expand the capability token for allowing the partners of entity so that they are able consume the resources provided by the resource provider. These areas further become our major motivation to provide a full fledged security protection for DE environment.

REFERENCES


Ilung Pranata is currently a Ph.D. candidate at The University of Newcastle, Australia where he received the prestigious EIPRS scholarship from the Australian Government. His research interests include Information Security and Trust Management System in the enterprise collaborative environments such as Internet and Digital Ecosystem. He received his Bachelor of Information Technology (Hons) degree with First Class Honours and University Medal from The University of Newcastle, Australia in 2009. Prior to Ph.D. studies, he worked full time as a technology associate in one of the largest financial industry in Singapore. Besides doing his research, he also enjoys teaching several tertiary courses at the University. He hold several industry certifications such as Microsoft Certified Professional (MCP) and CompTia Security +.

Dr. Rukshan Athauda is currently a Lecturer in IT at The University of Newcastle, Australia. He received his Bachelor of Science (Computer Science), Master of Science (Computer Science) and PhD from Florida International University (USA) in 1996, 1998 and 2000 respectively. Soon after his Doctorate graduation, he joined Microsoft Corp. as Software Development Engineer in Test with the SQL Server 5th IEEE International Database management system team. He has taught IT courses in a number of Universities in Australia, Sri Lanka and Singapore. His research interests are in integration and query of heterogeneous data sources, decision support system, ICT education, and network security. Previously, he was actively involved in High-Performance Database Research Centre (HPDRC) at Florida International University and the Priority Research Centre for Reproductive Science in developing a decision support system for the Mothers and Babies Research Centre at The University of Newcastle, Australia.

Dr. Geoff Skinner is currently a Lecturer in IT at The University of Newcastle, Australia. He received his PhD from Curtin University in 2007 and Bachelor of Engineering (Computer Engineering) (Hons) from University of Newcastle in 1999. He has worked for a number of leading enterprises in IT area both in US and Australia. From this work experience, He brought extensive industry knowledge to the teaching and research at the university. He is currently an active researcher in the area of Information Privacy, Data Security, and Knowledge Discovery. He also holds a Convener position for Master of Information Technology (MIT) at University of Newcastle, Australia. He has taught a wide number of courses ranging from E-Business, Knowledge Management and Data Mining, Business Intelligence Systems, and Application Development.