

Authentication for Clone Cyber-entities in Bio-net Computing Environments

Abstract

For mobile agent systems, some mechanisms have been proposed for authenticating mobile agents and execution environments. But, most current mobile agent systems don't have the Bio-net concepts such as autonomous replication or reproduction of the mobile agents or cyber-entities. In the Bio-net environments many cyber-entities can be created dynamically by the replication or reproduction mechanisms, and for security sensitive applications, the cyber-entities should be registered by on-line mechanisms for authentication afterwards. Also, there should be a peer-peer authentication mechanism for cyber-entities in the system that can be performed efficiently. For the authentication of cyber-entities in the Bio-net platform, we will propose necessary components to be included in security architecture and some mechanisms for authenticating them.

1. Introduction

Recently, there has been much research on mobile agent technologies. Mobile agent is an important programming paradigm for our increasingly networked world. It provides a flexible way to structure cooperative computation in distributed systems [1]. Already, some mobile agent technologies such as Java applets are popularly in use for constructing dynamic applications based on web-based distributed systems.

A new architecture, called Bio-net architecture, is proposed for providing scalability, adaptability, survivability, and availability for future network applications. The Bio-net architecture, inspired by key principles and mechanisms of biological systems, is a paradigm as well as a middleware for the design and implementation of scalable, adaptive and survivable/available network applications [2]. In the Bio-net architecture, a collection of autonomous mobile agents, called cyber-entities, are used to implement an application. The desirable characteristics of an application such as scalability, adaptability, survivability, and availability emerge from the collective actions and interactions of its constituent cyber-entities.

Cyber-entities in Bio-net architecture have important behaviors such as migration, replication, reproduction, pheromone emission, protection, energy exchange, and social networking. Among these behaviors, replication and reproduction behaviors enable cyber-entities to create another cyber-entities sexually or asexually. In the Bio-net platform, cyber-entities are usually designed to favor replication or reproduction in response to higher levels of stored energy, and in this way, there can be so many cyber-entities created dynamically from their parent cyber-entities due to the abundance of stored energy.

To design a security-sensitive application based on the Bio-net architecture it is necessary for cyber-entities to authenticate each other. For authentication, each cyber-entity should be

registered into the CA(Certificate Authority) and the registration should be performed on-line and efficiently because there can be so many cyber-entities created dynamically.

This paper proposes an authentication mechanism for cyber-entities of the security-sensitive application in Bio-net environments, and especially focuses on the authentication between two cyber-entities, not between a cyber-entity and a host.

We will propose two mechanisms in this paper. The first is the on-line registration mechanism for clone cyber-entities, and the second is the peer-peer authentication mechanism between cyber-entities. The registration mechanism is necessary when a cyber-entity creates a clone cyber-entity and gives him an id and private/public key pair. Two different cases, that is, single-CA environment and multiple-CA environment will be discussed. For authentication mechanism, the peer-peer authentication mechanism between two cyber-entities will be discussed with the assumption that they are already registered into some CAs.

We have some assumptions for our schemes in this paper. First, we assume that cyber-entities do not carry security sensitive information such as keys and registration states, and all security sensitive information are managed by the hosts or CAs. Second, as we mentioned earlier, we deal with on-line registration mechanism for clone cyber-entities for effectiveness because so many cyber-entities can be created dynamically by the cyber-entity replication or reproduction mechanism in the Bio-net environment. To control cyber-entity creation, we let cyber-entity launchers be able to register additional cloning policy of their cyber-entities in their home site if necessary. Third, we assume that each host has a security facility that can manage its private and public keys and also can provide security services for the cyber-entities in the Bio-net platform. We also assume that this security facility is secure in the viewpoint that it can be protected from the attacks by malicious cyber-entities or malicious hosts. It may be possible by constructing the security facility with tamper-proof hardware.

The balance of this paper is organized as follows. Section 2 presents some previous work on mobile agent systems. In section 3, we describe an overview of the Bio-net concepts and its architecture. Our system model and security architecture for authentication are described in section 4. We propose registration schemes of cyber-entities for both single-domain environment and multiple-domain environment in section 5, and peer-peer authentication scheme in section 6. Finally, section 7 concludes with a discussion of future work.

2. Previous Research on Mobile Agent Security

Key elements of any mobile agent based distributed system are the security mechanisms (1) to protect the execution environments against potentially malicious mobile agent under execution, (2) to protect the mobile agent against potentially malicious hosts or execution environments, and (3) to authenticate the mobile agent or host to ensure that the counterpart is the one who he wants.

Generally speaking, there is no complete routine that can decide for every possible mobile code segment whether it contains malicious code or not. So, the problem of protecting execution environment has been addressed in alternative ways. Current protection mechanisms for execution environment can be classified as *sand-boxing*, *digital shrink-wraps*, and *proof-carrying code* [3,4,5].

Protecting mobile agent is much harder problem than protecting execution environment because execution environment has full access to mobile code and data. The resulting approaches can be classified as prevention mechanisms and detection mechanisms. Prevention mechanisms, which aim at preventing any possible attack on mobile agents, include *limited blackbox security* [6], *computing with encrypted functions* [7,8,9,10], and *tamper-proof devices* [11,12]. Detection mechanisms, which aim at detecting any possible illegal modification of agent code, state, and execution flow, include *cryptographic traces* [13] and *state appraisal* mechanisms [14,15].

A fundamental concern in building a secure mobile agent environment is authentication of the entities in the system. In some point of view, the problem of authentication in mobile agent systems is similar to the problem of authentication in classical distributed systems. But, in mobile agent systems, there are some more difficulties in solving the authentication problem. First, supposing that the authentication mechanism is based on the cryptosystem, no strong and general key distribution mechanisms are known until now. Moreover, even if we have the key-distribution mechanism, there is still a problem of determining the correct semantics of the signature because there are a number of different signatures that might be affixed to a mobile agent and a number of different parties that might have signed it.

Woo and Lam proposed an authentication framework that can be used for designing secure distributed systems, including some specific protocols [16]. Their work proposed the secure bootstrapping, user-host authentication, and peer-peer authentication protocols and all the protocols are based on the client-server based distributed systems.

Lampson et. al. described a theory of authentication and proposed a system that implemented it [17]. Their theory is based on the notion of principal and a '*speaks for*' relationship between principals. The theory can be used for explaining many existing and proposed security mechanisms, especially for authentication mechanisms for distributed environments and mobile agent environments.

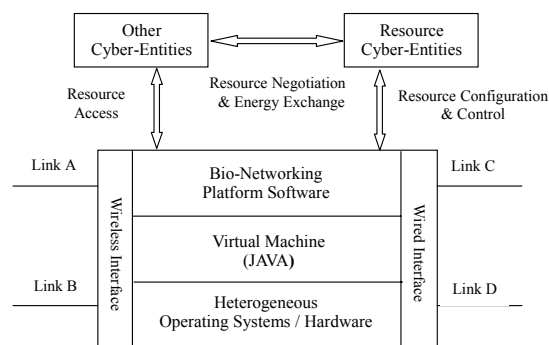
Berkovits et. al. proposed a security architecture that can perform authentication based on four distinct trust relationships between the principals of mobile agent systems [15]. They presented and proved conditions necessary to establish each trust relation and then created an architecture that established the conditions. They clarified the architecture and showed that it met its objectives using the distributed authentication theory developed by Lampson et. al. [17].

But, they focused mainly on the migration of mobile agents and described authentication mechanisms that are necessary when a mobile agent migrates.

3. Bio-net Architecture

The Bio-net architecture is a computing paradigm as well as a middleware that enables the construction and deployment of scalable, adaptive, and survivable/available applications. The paradigm is inspired by the principles and mechanisms of the biological systems such as the immune system, the bee colony, and the ant colony [2].

The paradigm consists of two major components, cyber-entities and Bio-net platforms. Cyber-entities are autonomous mobile agents, which are used to construct network applications. Bio-net platforms provide an execution environment for the cyber-entities. The Bio-net node with these two components is shown in Figure 1.



(Figure 1) Bio-net Node Architecture

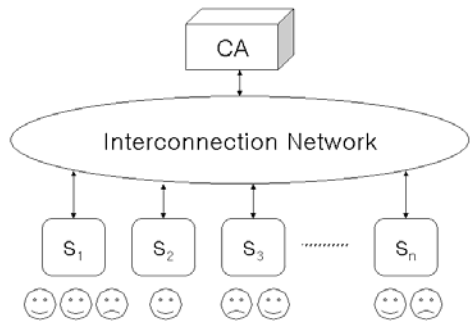
A Bio-net platform is any networked hardware device that runs a JVM(Java Virtual Machine) and the Bio-net platform software. All resources on a Bio-net platform, such as CPU time, memory, disk space, and network bandwidth, must be purchased with energy units. When a cyber-entity is created, it is given energy units by the system administrator who launched it or by its parent cyber-entities. Cyber-entities use their energy to buy resources from Bio-net platforms. The energy level of each cyber-entity can be stored and managed by the cyber-entity itself, the platform in which the cyber-entity resides, or a centralized trusted node [18]. The platform software also provides system level services that cyber-entities cannot perform directly, such as migration and reproduction. Since these services consume CPU and network resources, cyber-entities must pay energy to the platform to receive these services.

Cyber-entities run on Bio-net platform and follow biological principles and contain biological mechanisms. They collect energy from human users or other cyber-entities, and they also give energy to Bio-net platforms to use some resources. If a cyber-entity exhausts its energy, it cannot be allowed to run on Bio-net platform and it dies of starvation. On the other hand, if a cyber-entity gets large amounts of energy, it will have more opportunities to replicate or

reproduce. So, the cyber-entity that adapt to user demand or network environments can get more energy and can give birth to more cyber-entities similar to itself. Moreover, a cyber-entity may be explicitly coded to create its child at a specific site and at a specific time. In that case, the cyber-entity can create its child if it does not violate the Bio-net principles.

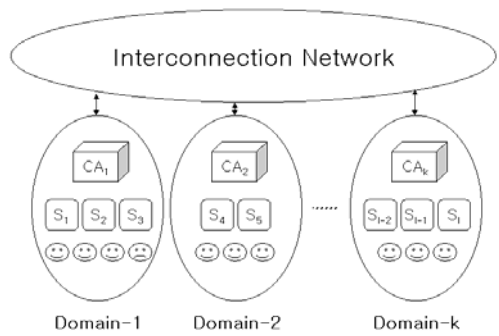
4. System Model and Architecture

We propose the cyber-entity registration mechanisms for two network environments, single-domain environment and multiple-domain environment. In single domain environment, there are several sites and cyber-entities, with a single CA. In this case, all cyber-entities should be registered in the CA. Figure 2 shows the single-domain environment.



(Figure 2) Single-domain Environment

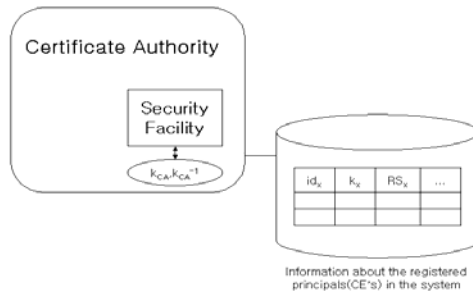
In multiple-domain environment, the entire network is partitioned into several domains and there is a CA in each domain. For authentication, each cyber-entity should be registered in the home CA or in the CA of the domain that he is currently located. Figure 3 shows a multiple-domain environment.



(Figure 3) Multiple-domain Environment

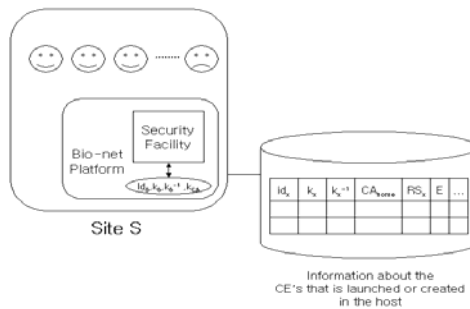
We assume that each CA in the network environment contains a *Security Facility* that stores its public/private key pair. Each CA also has a database that stores and manages some information

such as id, public key, and registration state for each of the cyber-entities that are registered into the CA. The architecture of the CA is shown in Figure 4.



(Figure 4) Architecture of the Certificate Authority

Each site S contains a security facility that stores its id, its public/private key pair, and the public key of the CA of the S's domain. Each site also has a DB that contains some information about the cyber-entities that are launched or created in the site. For each cyber-entity, the DB contains the id of the cyber-entity, public/private keys of the cyber-entity, home CA of the cyber-entity, registration state of the cyber-entity, energy level of the cyber-entity, and so on. Figure 5 shows the security components of each site in Bio-net environment.



(Figure 5) Security Components of Each Site in Bio-net Environment

5. Registration Schemes

Now, based on the security architecture discussed above, we will introduce the registration schemes for two different network environments. When a cyber-entity is created, whether it is due to energy level or due to explicit coding of the cyber-entity, the id and the public key should be registered into the CA, which is a trusted third party.

5.1 Registration Scheme for Single-domain Environments

In the single-domain environment, there is only one CA and registration of the clone cyber-entity is simple. When a parent cyber-entity P creates a child cyber-entity C, P sends a creation request to the Bio-net platform where P resides. Then, the Bio-net platform, with its security

facility, sends the registration request to the CA instead of P and C. Finally, the CA replies with the result of the registration, and the host activates the child cyber-entity when the registration is successful.

Detailed registration procedure is as follows. Suppose now that a cyber-entity P wants to create a child cyber-entity C. Cyber-entity P requests to create a child cyber-entity to the Bio-net platform S that he resides, and the Bio-net platform S assigns a new id, id_C , to the child cyber-entity. Also, the security facility of S creates the public/private key pair, k_C and k_C^{-1} for C and stores them in its DB. Now, S sends a following registration request message M1 to the CA.

$$M1 = \{S, CA, n_S, t_S, id_P, id_C, k_C\}_{k_S^{-1}}$$

Here, S is the source of the message and CA is the destination of the message. The third field n_S and the fourth field t_S are the nonce and timestamp, respectively and they are used to prevent replay attacks. The fifth field id_P is the id of the parent cyber-entity P and the sixth field id_C is the id of the child cyber-entity C. The last field k_C is the public key of the child cyber-entity C that is to be registered into the CA. This message M1 requests to register the id and public key of the cyber-entity C to the CA, and is signed with the private key of S before transmission.

CA, when it received the message M1, checks and verifies the message with the public key of S. After verification, CA makes a decision on whether to accept the request or not according to the predefined policy of the cyber-entity launchers if any. When it decided to accept the request, it registers id_C and k_C into its DB, and sends a reply message M2 that notices acceptance.

$$M2 = \{CA, S, n_S, D(M1), Mssg\}_{k_{CA}^{-1}}$$

In the message M2, the first two fields are the source and destination of the message. The third field is the nonce that is received with the request message. The fourth field is the message digest of M1 and is produced by some one-way function that both S and CA know. The last field notices that the request is accepted or not.

When site S receives the message M2, it verifies the message with the public key of the CA and checks whether the request is accepted or rejected. When the request is accepted, it activates the child cyber-entity C. When rejected, it deletes the entry of C from its DB and notifies P that the request is rejected.

5.2 Registration Scheme for Multiple-domain Environments

In multiple-domain environment, registration of the clone cyber-entity is relatively complicated. As in the case of single-domain environment, when a parent cyber-entity P creates a child cyber-entity C, P sends a creation request to the Bio-net platform where P resides. Then, the Bio-net platform, with its security facility, sends the registration request to the local CA of the

domain that it currently resides. This local CA processes the request and decides whether to accept the request or not. When it decides to accept the request, the child cyber-entity C is registered temporarily in the local CA. It is just a temporary registration because the local CA is not the CA in the domain that launched original application.

After temporary registration, the child cyber-entity is activated and can do its job with some constraints that depends on its application. Due to the temporary registration, cyber-entities can start his job early without the necessity of waiting full registration. But, the child cyber-entity should have some constraints because it is not fully registered. For example, it may retrieve some information from a DB but cannot update the information.

After temporary registration, the local CA immediately starts the full registration procedure by contacting the CA of the home domain of the cyber-entity. After full registration, the child cyber-entity can do its job without any constraints.

Detailed registration procedure is as follows. Suppose now that a cyber-entity P in a domain wants to create a child cyber-entity C. Cyber-entity P requests to create a child cyber-entity to the Bio-net platform that he resides. The Bio-net platform S assigns a new id, id_C , to the child cyber-entity, and the security facility of S creates the public/private key pair, k_C and k_C^{-1} for C and stores them in its DB. Now, S sends the following registration request message M1 to the local CA.

$$M1 = \{S, CA_{cur}, n_S, t_S, CA_{home}, id_P, \{id_C, k_C\}_K\}_{k_S^{-1}}$$

Here, the first 4 fields are the same as in the case of single-domain environment. The fifth field CA_{home} is the id of the CA in the home domain of the parent cyber-entity, and it is included for full registration that is to be done later. The sixth field id_P is the id of the parent cyber-entity P and the seventh field $\{id_C, k_C\}_K$ is the id and public key of the child cyber-entity C to be registered and it is encrypted with the symmetric key K that is generated by the security facility of the Bio-net platform S. This encryption is necessary because the local CA is not the CA in the home domain of the cyber-entity and so it cannot be fully trusted by the cyber-entity P. This message M1 requests to register the id and public key of the cyber-entity C to the CA, and is signed with the private key of S before transmission.

The local CA, when it received the message M1, checks and verifies the message with the public key of S. After verification, the CA makes a decision on whether to temporarily accept the request or not. When it decides to accept the request, it sends a reply message M2 that notices acceptance.

$$M2 = \{CA_{cur}, S, n_S, D(M1), M_{ssg}\}_{k_{CA_{cur}}^{-1}}$$

In the message M2, the last field M_{ssg} notices that the request is temporarily accepted or not.

In case of acceptance, the message M2 is also an implicit request for the key K.

When site S receives the message M2, it verifies the message with the public key of the local CA and checks whether the request is temporarily accepted or rejected. When the request is temporarily accepted, it sends the message M3 that contains the symmetric key K encrypted with the public key of the local CA.

$$M3 = \{S, CA_{cur}, n_S, \{K\}_{k_{CA_{cur}}}\}_{k_S^{-1}}$$

Now, the local CA checks the validity of the message M3 and obtains the symmetric key K from the message. With the key K, it decrypts the id and public key of the cyber-entity to be registered and register them into its DB. The local CA replies with an acknowledgement message M4 saying that the temporary registration is successful.

$$M4 = \{CA_{cur}, S, n_S, D(M3)\}_{k_{CA_{cur}}^{-1}}$$

Up to now, the child cyber-entity C is temporarily registered to the local CA. Now, the child cyber-entity C can be activated and can do its job with some constraints that is predefined by the application or the user.

At the same time the local CA starts the full registration procedure by contacting the CA in the home domain of the cyber-entity, CA_{home} . In other words, the full registration protocol is accomplished by the CA_{cur} and the CA_{home} only, and the cyber-entity C is not involved in the protocol so that the cyber-entity C can only do its application specific job. For full registration, the local CA sends the full registration request message M5 to CA_{home} .

$$M5 = \{CA_{cur}, CA_{home}, n_S, t_S, id_P, \{id_C, k_C\}_{k_{CA_{home}}}\}_{k_{CA_{cur}}^{-1}}$$

In this message, the id and public key of the cyber-entity C are encrypted with the public key of CA_{home} . This encryption is necessary for the confidentiality and integrity of the information. In other words, if the id and the public key are not encrypted the information may be modified by some malicious sites during transmission. Also, the entire message is signed with the private key of the local CA, and so, CA_{home} can be sure that the message was sent by CA_{cur} .

Now, CA_{home} checks and verifies the message M5 and decides whether to accept the request or not according to the cloning policy pre-declared by the application or the user that launched the application. When the cloning policy represents 'unlimited cloning', CA_{home} may always accept the registration request. But, when the cloning policy limits the cloning in any way, CA_{home} should check the cloning policy and make a decision based on the cloning policy.

The cloning policy, if any, can be registered when the application is launched and the root cyber-entity is activated, and the registration may be performed into the home site only or

additionally into the home CA also. When CA_{home} decides to accept the request, it extracts the id and public key from the message M5, and registers the information into its DB. CA_{home} sends the reply message M6 signed with its private key to the local CA that the cyber-entity C currently resides. In the message M6, the last field Mssg notices that the request is accepted or not.

$$M6 = \{CA_{home}, CA_{cur}, n_S, D(M5), Mssg\}_{k_{CA_{home}}^{-1}}$$

Now, the local CA checks and verifies the message with the public key of the CA_{home} . When the mssg field represents 'accepted', the local CA updates the registration state of the cyber-entity C from 'temporarily registered' to 'fully registered' and sends the notice(message M7) of full registration to the site S that created the cyber-entity C.

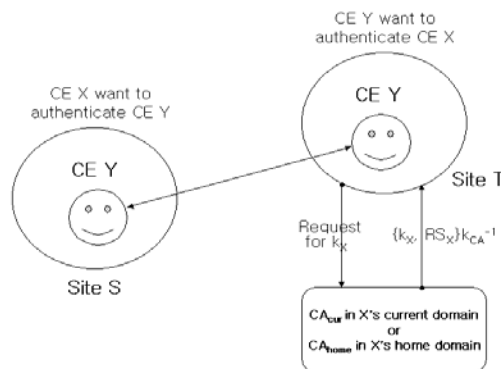
$$M7 = \{CA_{cur}, S, n_S, id_C, \text{"fully registered"}\}_{k_{CA_{cur}}^{-1}}$$

Site S, after receiving and verifying the message M7, also changes the registration state of the cyber-entity C and forwards this notice to other sites if necessary. The forwarding of this notice may be necessary when the cyber-entity C was moved to other site already. Eventually, the cyber-entity will have its state fully registered and can do its job without any constraints.

6. Authentication between Cyber-entities

When a security-sensitive application is launched in the Bio-net platform, where the related cyber-entities may interact frequently for synchronizing an event or for exchanging information, the interacting cyber-entities should authenticate each other to be sure that the counterpart is the one that he wants.

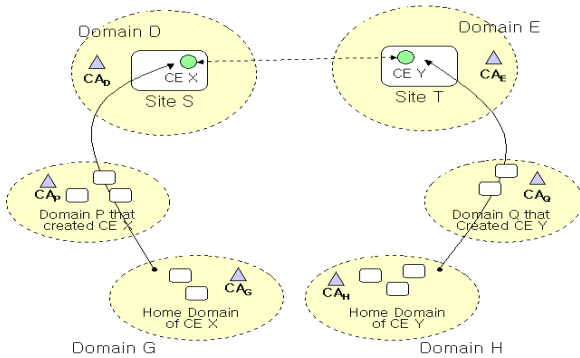
In Figure 6, a cyber-entity X in site S wants to interact another cyber-entity Y in site T. In this case, the cyber-entity X should authenticate the cyber-entity Y and the cyber-entity Y should authenticate the cyber-entity X. The information for authenticating the cyber-entities X and Y can be obtained from some CAs in the Bio-net environments because, as explained in the previous section, every cyber-entity is registered in some CAs when it is created.



(Figure 6) Authentication of Cyber-entities

In addition to authentication, this protocol also accomplishes the negotiation of the registration state of each cyber-entity and a new session key for future communication between the cyber-entities. The new session key is a symmetric key and can be used between the two cyber-entities for secure communication with low overhead. The registration states will affect the cyber-entities in such a way that limit the cyber-entities' capabilities. For example, temporarily registered cyber-entities can only request to retrieve some information but cannot request to update any information.

Figure 7 shows a scenario for a cyber-entity X at site S in the domain D contacts another cyber-entity Y at site T in the domain E. In this scenario, the cyber-entity X is created at a site in the domain P and has domain G as its home domain. In other words, the application or the root cyber-entity of cyber-entity X was originally launched at a site in the domain G. Similarly, the cyber-entity Y is created at a site in the domain Q and has domain H as its home domain.



(Figure 7) Authentication Scenario

So, for cyber-entities X and Y to mutually authenticate each other, one of them should first contact the CA in its domain and let the CA to get the information necessary for authenticating them. When the cyber-entity X initiated the interaction with the cyber-entity Y, the CA_E in the current domain of cyber-entity Y contacts the CAs in the domains that created the cyber-entities or the CAs of their home domains. After CA_E obtains the necessary information, it is delivered to the cyber-entity Y and again forwarded to the cyber-entity X. With the information obtained from the CA_E the two cyber-entities can authenticate each other. The authentication may be *partial* when the authenticated cyber-entity is temporarily registered and it may be *complete* when the authenticated cyber-entity is fully registered.

6.1 Peer-peer Authentication Protocol

Because every cyber-entity is registered into some CAs, the detailed peer-peer authentication procedure can be constructed in a similar way as that of the distributed systems [16]. To explain this procedure, we assume that a cyber-entity X in domain CA_D wants to contact

another cyber-entity in domain CA_E and their home domains and the domains that created them is as shown in the Figure 7.

When the cyber-entity X wants to interact another cyber-entity Y , it first creates a nonce n_x and sends a *connection establishment request* message $M1$ signed with the private key of X to the cyber-entity Y .

$$M1 = \{X, Y, n_x\}_{k_X^{-1}}$$

Upon receiving the request message $M1$, the cyber-entity Y creates its nonce n_y and constructs an *authentication request* message $M2$ containing the message $M1$ from X and the nonce n_y . The cyber-entity Y signs the message with its private key and sends the message to the CA in its local domain.

$$M2 = \{Y, CA_E, \{X, Y, n_x\}_{k_X^{-1}}, n_y\}_{k_Y^{-1}}$$

Now, CA_E obtains the public keys k_x and k_y and registration states RS_x and RS_y from the CAs in the home domain of the cyber-entities X and Y or the CAs of the domain in which the cyber-entities are created. Here, searching for CAs to obtain the public key of a cyber-entity is necessary. But, that is not in the scope of this paper and only the procedure for exchanging the public keys between the CAs will be discussed later in section 6.2.

Now, CA_E decrypts the message $M2$ with the public keys k_x and k_y just obtained and knows that the two cyber-entities X and Y wants to authenticate and interact each other. After generating a new session key k that is to be used for secure communication between the two cyber-entities X and Y , CA_E sends a message $M3$ containing the key k to Y .

$$M3 = \{CA_E, Y, k_X, \{X, n_x, RS_x, Y, n_y, RS_y, k\}_{k_{CA_E}^{-1}}\}_{k_Y}$$

In this message $M3$, the nonces and the registration states of the cyber-entities X and Y are included also so that the cyber-entities X and Y can confirm that the message is for them and it is for their recent authentication request.

After decrypting and verifying that the nonces are returned, the cyber-entity Y obtains the session key k and checks the registration state of the cyber-entity X . Y sends the message $M4$ containing the CA_E -signed portion of the message $M3$ and the authenticator $\{n_x, n_y\}_k$ to X . This authenticator let the cyber-entity X be sure that the counterpart is the cyber-entity Y because only cyber-entity Y knows the session key k .

$$M4 = \{Y, X, \{X, n_x, RS_x, Y, n_y, RS_y, k\}_{k_{CA_E}^{-1}}\}_{k_X}, \{n_x, n_y\}_k$$

Now, after verifying the message M4, cyber-entity X has authenticated the cyber-entity Y and creates a security association to Y with the session key k. Of course, this security association may be restrictive according to the registration state of the cyber-entity Y. The cyber-entity X sends an acknowledgement message M5 including the nonce n_Y to Y. This message M5 is encrypted with the session key k.

$$M5 = \{X, Y, n_Y\}_k$$

The cyber-entity Y decrypts the message M5 with the key k and verifies it. Finally, cyber-entity Y has also authenticated the cyber-entity X and creates a security association to X with the session key k according to the registration state of X. Up to now, the mutual peer-peer authentication between cyber-entities is finished. The authenticated cyber-entities can now communicate using the secure channel.

6.2 Obtaining Authentication Information

For authenticating a cyber-entity X, the cyber-entity Y that received the connection establishment request message from X requests some information such as public keys and registration states to its local CA. This local CA, CA_E , if he does not have the necessary information, should contact other CA, CA_L , to obtain the information. The CA_L may be the CA in the domain where the cyber-entity is created, the CA in the home domain of the cyber-entity, or the CA in any other domain. It may depend on the search algorithm that should be constructed, which is not in the scope of this paper. Here, we introduce the procedure that a $CA(CA_E)$ contacts another $CA(CA_L)$ and gets some necessary information about the cyber-entity.

After finding the CA, CA_L , that maintains the necessary information for the cyber-entity X, CA_E sends CA_L the message M1 that requests information about the cyber-entity X. This message contains the id of the cyber-entity X and a nonce, and is signed with the private key of CA_E .

$$M1 = \{CA_E, CA_L, id_X, n_E\}_{k_{CA_E}^{-1}}$$

Upon receiving the message M1, CA_L checks and verifies the message, and retrieves the information on the cyber-entity X from its DB. CA_L sends the retrieved information such as the public key and registration state of the cyber-entity X to CA_E , together with the nonce that he received from CA_E .

$$M2 = \{\{CA_L, CA_E, n_E, id_X, k_X, RS_X\}_{k_{CA_L}^{-1}}\}_{k_{CA_E}}$$

This message M2 is signed with the private key of the CA_L , and again encrypted with the public key of CA_E to guarantee the secrecy and integrity.

7. Conclusion and Future Work

Bio-net architecture, inspired by key principles and mechanisms of biological systems, is a computing paradigm as well as a middleware for the design and implementation of scalable, adaptive and survivable/available network applications. It is proposed for providing scalability, adaptability, survivability, and availability for future network applications. In the Bio-net architecture, a collection of autonomous mobile agents, called cyber-entities, are used to implement an application. The desirable characteristics of an application such as scalability, adaptability, survivability, and availability emerge from the collective actions and interactions of its constituent cyber-entities.

In Bio-net platforms, cyber-entities are usually designed to replicate or reproduce their child cyber-entities in response to higher levels of stored energy. So, so many cyber-entities can be created dynamically from their parent cyber-entities due to the abundance of stored energy.

For security-sensitive applications in Bio-net architecture, it is necessary to have a mechanism for cyber-entities to authenticate each other. For authentication, each cyber-entity should be registered into some CAs by efficient on-line mechanism because so many cyber-entities can be created dynamically.

This paper proposed mechanisms for authentication of cyber-entities for security-sensitive applications in Bio-net environments. Especially we proposed two mechanisms in this paper, the on-line registration mechanism for cyber-entities created dynamically and the peer-peer authentication mechanism between cyber-entities. We focused on the efficiency of the peer-peer authentication mechanism because peer-peer authentication is much more frequent than cyber-entity registration.

Although our network model has several domains and each domain has a CA in our security architecture, each cyber-entities created dynamically should be registered to some CAs and it may cause bottleneck in the CAs. Also, a heavy congestion can be caused around the CAs, limiting the scalability of our schemes. Later, it may be necessary to develop authentication schemes that don't need any centralized entities and fit exactly to the Bio-net concepts.

[References]

- [1] V. A. Pham and A. Karmouch, "Mobile Software Agents: An Overview," IEEE Communications Magazine, Jul. 1998.
- [2] M. Wang and T. Suda, "The Bio-Networking Architecture: A Biologically Inspired Approach to the Design of Scalable, Adaptive, and Survivable/Available Network Applications," TR 00-03, Dept. of Info. and Comp. Sci., UC-Irvine, Feb. 2000.
- [3] R. Oppliger, "Security Issues Related to Mobile Code and Agent-based Systems," Computer Communications, Vol. 22, 1999.
- [4] S. Oaks, Java Security, O'Reilly and Associates, Sebastopol, CA, 1998.

- [5] G. C. Necula and P. Lee, "Safe, Untrusted Agents Using Proof-Carrying Code," in *Mobile Agent and Security*, Lecture Notes in Computer Science 1419, Springer-Verlag, Berlin, 1998.
- [6] F. Hohl, "Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts," in *Mobile Agent and Security*, Lecture Notes in Computer Science 1419, Springer-Verlag, Berlin, 1998.
- [7] T. Sander and C. F. Tschudin, "Protecting Mobile Agents against Malicious Hosts," in *Mobile Agent and Security*, Lecture Notes in Computer Science 1419, Springer-Verlag, Berlin, 1998.
- [8] T. Sander, A. Young, and M. Yung, "Non-Interactive CryptoComputing for NC¹," in *Proc. 40th IEEE Symposium on Foundations of Computer Science*, 1999.
- [9] C. Cachin, J. Camenisch, J. Kilian, and J. Muller, "One-Round Secure Computation and Secure Autonomous Mobile Agents," in *Proc. 27th International Colloquium on Automata, Languages, and Programming*, Lecture Notes in Computer Science 1853, Springer-Verlag, Jul. 2000.
- [10] J. Algesheimer, et.al., "Cryptographic Security for Mobile Code," in *Proc. 2001 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001.
- [11] B. Yee, "A Sanctuary for Mobile Agents," in *Secure Internet Programming*, Lecture Notes in Computer Science 1603, Springer-Verlag, 1999.
- [12] U. G. Wilhelm, S. Staamann, and L. Buttyan, "Introducing Trusted Third Parties to the Mobile Agent Paradigm," in *Secure Internet Programming*, Lecture Notes in Computer Science 1603, Springer-Verlag, 1999.
- [13] G. Vigna, "Cryptographic Traces for Mobile Agents," in *Mobile Agent and Security*, Lecture Notes in Computer Science 1419, Springer-Verlag, Berlin, 1998.
- [14] W. M. Farmer, J. D. Guttman, and V. Swarup, "Security for Mobile Agents: Authentication and State Appraisal," in *Proc. of the 4th European Symposium on Research in Computer Security*, 1996.
- [15] S. Berkovits, J. D. Guttman, and V. Swarup, "Authentication for Mobile Agents," in *Mobile Agent and Security*, Lecture Notes in Computer Science 1419, Springer-Verlag, Berlin, 1998.
- [16] T. Y. C. Woo and S. S. Lam, "Authentication for Distributed Systems," in *Internet Besieged: Countering Cyberspace Scofflaws*, ACM Press and Addison-Wesley, 1997.
- [17] B. Lampson, M. Abadi, M. Burrows, and E. Wobber, "Authentication in Distributed Systems: Theory and Practice," *ACM Trans. on Comp. Sys.*, Vol. 10, No. 4, Nov. 1992.
- [18] S. Song and T. Suda, "Security on Energy Level in the Bio-networking Architecture," in *Proc. of the 3rd International Conference on Advanced Communications Technology*, 2001.