SETECS
Secure Transactions and Electronic Commerce Systems

Tel: (240) 535–2095 ◆ **http:**//security.setecs.com ◆ **E-mail:** info@setecs.com

SETECS® Security System

Secure Protocols and Applications for Cloud Computing Environments

White Paper – February 17, 2011

Executive Summary

Cloud computing is becoming more and more popular computing paradigm. Using remotely hosted applications and on–demand, with even all user resources stored at remote servers ("virtual servers") has many advantages compared to the standard, client–server environment.

However, in such environments, security has much more important role than in classical network, client–server, environments. Not only that the same, standard, security services are needed (authentication, authorization, confidentially, integrity, authorization, etc.), but their provision must be offered to clients transparently and in an environment comprising distributed components and delegated authorities. Cloud computing makes security not only much more important, but also much more difficult to organize and manage, due to the transparent nature of cloud resources, components, and services.

In the associated white paper we describe security infrastructure, components and various security services required for cloud computing environments. In this white paper we describe the details of security protocols and examples of some secure applications, adjusted to function in a secure cloud computing environments.

1. Introduction

Design of security protocols and applications described in this white paper is based on modular approach and each module is implemented using the concept of generic security objects. Security protocols are initial user authentication (login into user workstation) using FIPS 201 (PIV) smart cards, remote user authentication using FIPS 196 based strong authentication protocol, single-sign-on protocol, establishing and using secure sessions between clients and cloud servers, SAML authorization protocol, and key distribution protocol. In our system we use security protocols to provide various security services to users of cloud computing services.

2. Authentication Protocols

2.1 Local User Authentication Protocol – Login to Workstation

Local user authentication protocol is designed as a generic login module. This means that it supports local authentication of users (login into user workstation) base don variety of protocols. All are supported by PIV–compliant smart cards, but security administrators may configure different local authentication protocols. Sometimes, users are registered in a local Windows system with their username and password (standard Windows local authentication). Even that protocol is supported by our PIV smart cards.

It is known that standard PIV cards do not support authentication based on username and password. In the data model of the standard PIV applet, there are no username and password attributes. To support even standard Windows login, we have created extended PIV smart card. That is standard PIV card with standard PIV applet, but with an additional applet loaded in the card – Security applet. Its data objects include security attributes additional to standard PIV attributes, such as username, password, SAML ticket, user PIV role, several secret session keys, etc.

Based on such extended PIV card, our system supports user authentication based on userName/password, IDMS-based authentication, and certificates-based authentication. Upon starting user workstation (Windows PC), the workstation automatically checks installation environment and its configuration and based on that selects the appropriate local authentication protocol. If local authentication is configured to use userName/password, our system displays login panel asking user for a PIN and/or PIN plus fingerprint in order to activate smart card. When user provides those parameters and inserts smart card in the reader, our security client fetches userName and password from the Security applet and presents them to the login module of the native Windows operating system. Login module consults standard Windows local user accounts database for authentication. If the user is registered there, he/she will be successfully authentication and the workstation will continue boot process.

The main problem with this type of authentication is that userName and password stored in the smart card must be mapped to the operating system's user accounts database. The change of that password requires administrative privileges at the workstation and requires also the change of the password in a smart card. To solve this problem, instead of user password, we store special symmetric-key in the Security applet of our extended PIV smart card and at the same time we keep user password in the database of our IDMS server, encrypted with the symmetric key stored in the smart card. Therefore, the purpose of the symmetric-key is to encrypt a password before storing it in the IDMS database. When a user performs IDMS-based authentication, our local login module fetches the encrypted password from the IDMS and decrypts it using symmetric key from the smart card. After that, it presents userName and password to the standard login module of Windows.

2.2 Remote User Authentication Protocol – Login to Security Access Point

In our system remote user authentication may be performed either as standard Windows remote authentication (authentication into Windows domain) or as our extended authentication (authentication into the cloud). In both cases, our generic user authentication module supports certificate-based authentication. With authentication into the Windows domain, our login module fetches PIV authentication certificate from a FIPS-201 (PIV) based smart card and presents it to the Windows login module for domain level authentication. As it is well–known, the prerequisite for such authentication is that PIV certificate must be issued by Microsoft (or compliant) CA server, the certificate must have strictly required values of the `keyUsage` attribute, and it must be stored in Microsoft's Active Directory.

Another remote authentication protocol that our system supports is authentication with the cloud's Security Access Point using mutual strong authentication protocol. Our protocol is an extension of the FIPS-196 strong authentication protocol. Its extended security functions are verification of certificates by the Local Certificate Authority (LCA) Server and verification of identities by the IDMS Server. Our mutual strong authentication protocol also uses PIV credentials and smart card-based cryptographic functions.

In our system client initiates mutual strong authentication protocol with the Security Access Point (SAP) server in the cloud and sends PIV authentication certificate to the SAP Server instead of the *Hello* message, as specified in the FIPS 196 standard:

$$Client \rightarrow SAP\ Server : Cert_{PIV-a}$$

SAP Server receives the certificate and verifies it by sending it to the LCA Server. In addition, it also verifies the Distinguished Name of the user using IDMS Server. Upon successful verification, SAP Server generates random number R_s and sends it back to the client. Otherwise, if verification fails, it informs the client and stops conversation with the client.

$$Server \rightarrow Client : R_s$$

Client receives R_s and signs it using private key stored in the PIV card corresponding to the PIV authentication certificate. The following cryptographic functions are used to generate signature of the R_s:

$$h = H(R_s) \quad \ldots \quad (1)$$
$$S(R_s) = E(h, \text{private key}) \quad \ldots \quad (2)$$

In these equations *H* is a hash function and *h* is the result of the hash function. *E* is an encryption function, which encrypts *h* using private key corresponding to the PIV authentication certificate. In the next step, client generates a random number R_c and returns it with $S(R_s)$ to the SA Server:

$$\text{Client} \rightarrow \text{Server} : \{S(R_s),\ R_c\}$$

SA Server receives the message and verifies client's signature using the following cryptographic functions:

$$h = H(R_s)$$
$$h` = D(S(R_s), \text{public key}) \quad \ldots \quad (3)$$

In equation (3), SAP Server uses public key, extracted from the PIV authentication certificate of the client, for verification of the signed challenge *(S(Rs))*.

If *h* is equal to *h`*, SA Server sends digitally signed R_c and its digital signature certificate to the client. Cryptographic functions are the same as explained in Equations (1) and (2):

$$\text{Server} \rightarrow \text{Client} : \{S(R_c),\ Cert_{sa}\}$$

Client receives signed random number and verifies its digital signature using Equation (1) and Equation (3). But, in this case it uses public key extracted from the digital signature certificate of the SAP Server. In addition, it also verifies digital signature certificate from the LCA Server and the identity of the SAP Server using IDMS Server.

$$\text{Client} \rightarrow \text{Server} : S(R_c),\ Cert_{sa}$$

Upon successful authentication, SAP Server creates connection with the XACML Policy Decision Point (PDP) Server and sends the identity of the client (Distinguished Name) requesting SAML Ticket. PDP Server validates client's identity using IDMS Server and generates SAML ticket. SAML ticket contains ticket-id, identity of the client, timestamp, and IP address of the PDP Server. PDP Server also digitally signs SAML ticket (ST) using its own private key corresponding to its digital signature certificate. It sends signed ST to the SAP Server which then sends it back to the client. Client receives ST and stores it in the Security applet in a smart card.

2.3 Single-Sign-On Protocol – Login to The Cloud

When client wants to send request to some secure Application Server in the cloud, single sign-on protocol is initiated. This protocol is slightly different in a cloud environment, compared to the standard client-server environment. Namely, in the client-server environment, client access application server directly. Therefore, each application server must be extended with the associated Policy Enforcement Point (PEP) module.

But, in cloud computing environment, the situation is different, since user does not access application servers directly. Each request is first received by the SAP Server. Therefore, single sign-on protocol is performed by the SAP Server. That Server initiates single-sign-on protocol. Upon receiving the initiation message, client fetches ST from a smart card and digitally signs it using private key corresponding to the digital signature certificate. It sends ST to the Policy Enforcement Point (proxy associated with the application server) along with digital signature certificate:

$$\text{Client} \rightarrow \text{SAP} : \text{Request}\ (ST_s,\ Cert_{DSc}) \ldots \quad (4)$$

SAP also signs ST and concatenates to it multi-party signature ST_{ss}. SAP encapsulates ST_{ss} in the `SAMLAuthenticationRequest` message and sends it to the PDP Server for validation:

$$SAP \rightarrow PDP : SAMLAuthenticationRequest\ (ST, ST_{ss}, Cert_{DSc}) \quad \ldots (5)$$

PAP Server verifies both signatures. Successful verification of signatures proves that the SAML ticket was received from the PEP and presented by the owner of the SAML Ticket, which provides source authentication. After this, PDP Server consults SAML-Ticket database, in order to validate ST. If *OK*, it sends SAMLAuthenticationResponse message to the SAP Server, as shown in Equation (6), which contains PDP's authentication decision:

$$PDP \rightarrow SAP : SAMLAuthenticationResponse\ (Permit/Deny) \ldots (6)$$

If the decision is *Deny*, SAP informs the client that specific request has not been approved and terminates the connection without any further correspondence. If the decision is *Permit*, SAP component establishes secure session with the client.

3. Secure Cloud Application Services

In our implementation of the secure cloud computing environment, all servers located in the cloud are strongly protected and communications between them are also strongly protected. Security of internal cloud servers and their mutual protocols is described in additional white paper. Under those conditions, establishment of end–to–end secure sessions between users and application servers is in fact two-step security: the first step is external secure session established between clients at user workstations and SAP Server and the second step is secure sessions internal in the cloud established between SAP Server and application servers.

This arrangement for two-step end–to–end security is one of essential differences between cloud security and client-server security. In a client-server environment, clients access application servers directly, perform authentication or single sign-on with those servers and in that process shared session keys can be established. In a cloud computing environment, clients do not access application servers directly. Their requests are received by Application Access Point (AAP) Server that analyses them and based on different parameters and conditions directs those requests to appropriate application servers located in the cloud. Thus, in principle, at the time when accessing the cloud, users and SAP Server do not know which application servers will be accessed later, during user's session.

This means that during initial client authentication to the cloud, secure session can be established only between the client and SAP Server. Later, fro secure access to application servers two approaches are possible:

- Combined secure session, comprising of one portion between the client and SAP Server and the other portion between SAP Server and the application server selected to serve particular request; or
- Single secure session between the client and selected application server, in case of multiple repetitive requests.

The first option introduces performance overheads due to decryption/encryption that has to be performed for each request and response by the SAP Server. The second option is more efficient, but requires an intermediary step – "tunneling" protocol between selected application server and the client. The essence of that protocol is the following: during remote authentication to the cloud, client sends its certificate to the SAP Server. After receiving client's first request and selecting appropriate application server to service that request, SAP server appends also client's certificate to the client's request and sends it to the application server. Application server generates shared secret key and using client's certificate, envelops secret key, and appends it to its response. Thus, the client receives shared session key from an application server.

3.1 Secure Sessions

In the current version of our system secure session is established after single-sign-on protocol is successfully completed. SAP Server requests `KeyExchange` certificate from a client, as shown in (7):

$$SAP \rightarrow Client: Request(KeyExchangeCert_c) \quad \ldots\ldots\ldots (7)$$

The purpose of the `KeyExchange` certificate is to securely exchange session-key and session-id between a client and the SAP Server. To manage secure sessions' attributes by the SAP Server, the server creates an active session object for the specific client in a session's container. Each object in the session container contains the identity of the authenticated client, session key, and session ID.

Upon receiving certificate request, client fetches `KeyExchange` certificate from a smart card and sends it back to the SAP Server, as shown in (8):

$$Client \rightarrow SAP\ Server : Response(KeyExchange\ Cert_c) \quad \ldots\ldots\ldots \quad (8)$$

Since single-sign-on protocol is capable to authenticate clients in a distributed environment, there is still a possibility that the attacker may launch replay or impersonation attacks by presenting valid SAML ticket. To counter such attacks, SAP Server receives `KeyExchange` certificate and compares its Distinguished Name with the identity stored in the session container. In addition, SAP Server also verifies the certificate chain. Upon successful verification, SAP generates a session-symmetric-key and session id which is digitally signed by using private key corresponding to its own digital signature certificate and enveloped using public key corresponding to the `KeyExchange` certificate of the client. It then sends session key exchange message to the client, as shown in (9):

$$SAP\ Server \rightarrow Client: P(SK, SID), KeyExchange\ Cert_{as} \quad \ldots\ldots\ldots (9)$$

Client receives the message and verifies the signature. Upon successful verification, it opens the envelope using private key corresponding to the `KeyExchange` certificate in order to extract session-symmetric-key and session id. Client stores both session attributes in the Security applet of the smart card. Otherwise, it stores them in a key-file. Client uses session-symmetric-key and smart card-based cryptographic functions to create secure messages in the standard format – *PKCS*#7`SignedAndEnvelopedData`. The purpose of session-id is to enable later application servers to perform secure asynchronous communication with the client.

3.2 Authorization Protocol

Authorization policies in OneCLOUD™ system are based on the XACML standard. It supports Role-Based Access Control model, so an authorized person (for example Security Administrator (SA_d)), creates a group based on roles and assigns roles to users. Then Security Administrator creates rules by specifying access privileges for each group (role) along with permitted actions. SA_d then collects different rules into a Policy Token (Policy Set), which includes *Target* objects used to identify the role of each user in a group. *Target* contains the role, the name of a resource, and actions permitted to be performed by a group member with the specified resource. In addition, SA_d can also specify *Policy* and *Rules* objects, if needed. SA_d saves newly created policy in an XACML policy file.

When an authenticated user requests an access to a specific resource in the cloud, its client fetches SAML ticket from a smart card and sends it to the SAP Server, along with the name of the requested resource. SAP Server creates `SAMLAuthorizationRequest` message and sends it to the PDP Server. PDP Server consults XACML policy file and generates `SAMLAuthorizationResponse` message, which contains authorization decision. `SAMLAuthorizationResponse` is sent back to the SAP Server in order to dispatch request to an appropriate application server in the cloud, if the decision was Permit or to reject the request otherwise and return notification to the user.

4. Security Management Protocols

4.1 Group Key Management Protocol

Some of secure applications in the cloud may operate in a collaborative environment and therefore need key exchange protocol to exchange group-keys between the members of the group. For this purpose, OneCLOUD™ system supports Group Key Distribution (GKD) Server compliant with the GSAKMP standard. GKD Server performs key-related functions in a group of users, like group key creation, group key distribution, and rekeying. GKD supports both Push and Pull-based operations to distribute shared group keys. This Server is associated with various application servers whose resources and actions are shared by a group of users. It also uses SAP Server for enforcement of authorization policies for shared group keys.

When a user who is the member of a group requests shared group-key, he/she first performs single-sign-on protocol and establishes secure session with the SAP Server, as described in Sections 2.3 and 3.1. After that, group member sends request for a group key to be used by the particular service. As before, SAP Server requests SAML ticket fro the service of distribution of group keys. Client fetches SAML ticket from a smart card and sends it back to the SAP Server. SAP Server enforces authorization policies based on procedure described in Section 3.2. After successful authorization, the request is forwarded to the GKD Server, which returns group key to the authorized user using secure communication channel described in section 3.1.

4.2 Certificates Management Protocol

The service to generate and distribute secret, shared session keys is described in section 3.1, while distribution of shared, session group keys is described in section 4.1. The only remaining type of keys is public keys, distributed in the form of certificates.

Management of certificates may also be interpreted as the service of the cloud. In this case Certification Authority (CA) Server is located in the cloud. The cloud may contain several Issuing CA Servers, but most probably their use will not be transparent to users. The reason is that individual users must be certified under the specific Certification Policy. But, in both cases, either with requests for certificate services sent directly to the CA Server selected by the user or by treating such request as services of the cloud, certificate requests may still be interpreted and handled as cloud services.

Certificate request, in the form of PKCS#10 data object may be send to the AAP Server in the cloud. In case of multiple Issuing CA Servers, AAP Server will select one of them and direct the request. Regardless of one of more Issuing CA Servers in the cloud, all issued certificates will anyhow be stored in the IDMS Server.

4.3 Smart Cards Management Protocol

FIPS 201 standard specifies procedures and components to create request, issue and manage PIV cards. The components of the architecture comprises various types of PIV stations and PIV Card Management System (CMS) Server. CMS Server may be used in two modes: as managed server or as the deployed server. Managed server is located in the facility of some service provider and deployed version is located in premises of the agency.

The concept of cloud computing is particularly suitable for mode of the CMS Server. In this case, the server may be located in the cloud and PIV card requests may be treated as request to the cloud for any other type of services.