Safe Areas of Computation for Secure Computing with Insecure Applications

André L. M. dos Santos and Richard A. Kemmerer
Reliable Software Group
Computer Science Department
University of California
Santa Barbara, CA 93106 USA
{andre, kemm}@cs.ucsb.edu

Abstract

Currently the computer systems and software used by the average user offer virtually no security. Because of this many attacks, both simulated and real, have been described by the security community and have appeared in the popular press. This paper presents an approach to increase the level of security provided to users when interacting with otherwise unsafe applications and computing systems. The general approach, called Safe Areas of Computation (SAC), uses trusted devices, such as smart cards, to provide an area of secure processing and storage.

This paper describes preliminary results of using the Safe Areas of Computation approach to protect specific browsing applications. The intent is for protected browsers to be used to interact with institutions that have requirements for high security, such as financial institutions that enable users to perform sensitive operations for electronic commerce or online banking.

1 Introduction

The Safe Area of Computation (SAC) approach uses a collection of trusted devices that enforces the protection of users from the insecurity of specific applications. Each of these devices is called a Safe Area of Computation. The goal of such devices is to provide islands of security that interact with an ocean of insecurity.

The main goal of the SAC approach is to provide security for client-server applications. The approach can be used to protect stand-alone applications too. However, due to space limitations this paper only describes its use for protecting client-server applications. In a client-server configuration Safe Areas of Computation provide:

a) *Strong authentication*. A client SAC exchanges cryptographic messages with a server SAC in order to perform mutual authentication. At the end of the exchange the client SAC and the server SAC will have agreed on a secret key.

b) *Secure channels*. Both the client and the server SAC will use the secret key that was agreed on in the authentication step to encrypt and decrypt messages that are exchanged, thus providing a secure channel.

c) *Access control*. Every client SAC has an access control list that specifies for each application being secured, the types of data that the user can access. In addition, if the type is hierarchical, the access list entry also specifies the clearance level of the user for that type. In order to access data the user's access control list must contain an entry of the type specified by the data's security label and the user's clearance level must dominate the security label of the data.

An important goal of SAC is to provide strong security even for users that are not concerned with security. The price paid for this security is that the SAC approach requires the user to deal with a hardware token, which is the client SAC. Although this requires an initial adaptation to its use, the benefits of the additional security provided outweigh the initial discomfort. The only requirement for a user is to insert the client SAC in a reader before performing a secure transaction, in the same way that ATM cards are currently used for interactions with bank ATM machines.

The SAC approach provides access control in a generic way. The approach uses generic hierarchical security labels and access control lists without predefining what the labels and clearance levels are. This enables contextual interpretation and implementation of more than one security policy concurrently. A bank can, for example, use security labels to specify what banking transactions a user can perform, while a digital library can use labels to specify what type of documents a user can access and if the documents have clearance levels what level is required. Although this paper uses documents to demonstrate the use of security labels, many different access policies can be modeled in the same way.

The SAC approach can be better understood with an example. Therefore, the next section gives a high level example of how the SAC approach can be used to protect data distribution over the Internet. Section 3 describes the details of the data structures used in the SAC approach and the overall architecture of the system. A prototype implementation of the SAC approach is described in section 4. Finally, conclusions and future work are presented in the last section.

2 Overview of the Use of SAC for Secure Internet Transactions

Currently, many Internet transactions are claimed to be secure because they use Secure Socket Layer (SSL) to establish an encrypted link between the client and the server. Although the SSL protocol is likely to be secure, it is generally misused. The result is that the transactions are not secure. Some of the problems are:

a) *Users don't check certificates*. The usage of SSL does not guarantee that a user is interacting with the site that he/she wants to interact with. It only guarantees that the site implements SSL. A user needs to check the site certificate in order to verify the name under which the site is certified. This is usually neglected even by users that are aware of the existence of certificates, and it is unknown to the rest. Another problem is that some sites do not get a certificate using the same name by which they may be known to the user.
b) *Anybody can have a certificate*. Anybody can apply for a certificate and receive one.
c) *User credentials are stored in an unsafe place*. Most of the sites that use SSL do not require a user to have a credential to authenticate his/herself to the site. However, even when a site requires a credential and the browser supports the use of credentials, this alone does not guarantee that the user identified is the one interacting with the site. For example, credentials that are stored in an unsafe place, such as the user's computer, can be stolen. These credentials can then be used later by an attacker.
d) *The loss of a credential may not be noticed by the user*. Credentials that are stored in unsafe areas can be compromised without the user ever noticing. In contrast, if a token is used for safe transactions, the stealing or loss of a user's token is easily noticed by the user.

Commercial products that use smart cards to store certificates in order to prevent attacks against these certificates already exist. These products represent a move in the right direction and provide some protection against attacks to a user's credentials. They, however, use the smart card primarily for certificate storage, while still making most of the decisions in an unsafe area.

The goal of the SAC approach is to protect specific servers. Thus, it does not use the general approach of certificates. Figure 1 shows a high level view of the SAC approach to protecting Internet transactions. Each client has a client SAC, which holds its access control list and other security relevant information, and each server has a server SAC. There are two different types of server sites: central authority sites and service provider sites. Initially there is a single central authority site with which the client SAC can communicate. A central authority site is the only site that can add access to other sites to a client SAC. However, once a service provider site has been added to a client site, Internet transactions are performed between the service provider site and the user without intervention from a central authority. A service provider can also modify the portion of the client SAC's access control list that refers to that particular service provider, and it can remove all of the user's access to the site. It cannot, however, modify the user's access list for any other site.

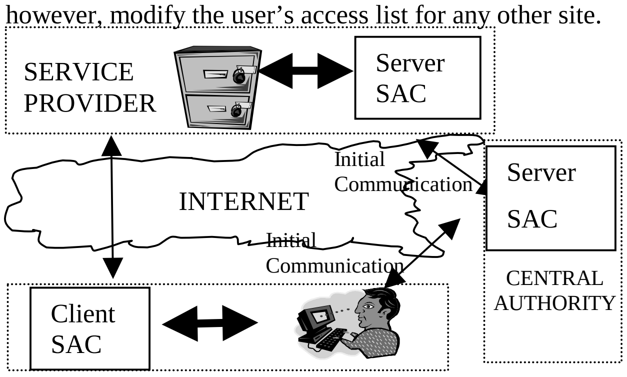

Figure 1: SAC approach for Internet Transactions

The use of the SAC approach for Internet transactions has many desirable features. The protocol used by the SAC approach provides strong authentication and prevents spoofing attacks against a user in a user-friendly manner. A private identification number (PIN) is used to authenticate a user to the SAC. This number can be any easy to remember number. To protect the data in the SAC, in case the SAC is lost or stolen, the client SAC has a lock out procedure that locks itself after a specified number of wrong attempts. Because of this, any dictionary attack against the client SAC has a high probability of being unsuccessful. In addition, because the SAC is tamper resistant, the data cannot be copied for repeating the attack. After entering the PIN, a user does not need to perform any additional authentication. The PIN is the only secret information that a user has, and it is only useful in the presence of the correct client SAC. In addition, any spoofing attack will not be successful since the protocol does not reveal any secret in the clear and the user can at most be tricked into revealing his/her PIN. A critical element of the SAC protocol is that the SAC exchanges a secret key in the authentication step. Thus, an attacker who tries to perform a man-in-the-middle attack, during authentication, will be unsuccessful, because either the user exchanges the secret key with the attacker, which does not authenticate the user to the server, or with the server, which does not reveal the secret key to the attacker. The use of the server public key by the user and the user's public key by the server for encryption guarantees that only the end parties can decrypt and retrieve the secret key.

Another desirable feature of the SAC approach is that the use of access lists and security levels enables the personalization of a user's services without overloading the server. Personalized data that is security related is safely stored in the client SAC. This data is used by the SAC to determine if the user can access the data requested, and only pre-authorized access is transmitted. Thus, any access that would not be authorized will not even be transmitted to the server, which reduces the load on the server. Although only pre-authorized requests are transmitted to the server, there is a back up check at the server to prevent any malicious requests.

A user can only be impersonated if his/her PIN is compromised and his/her SAC is lost or stolen. If this is the case, the user can easily notice the loss and the particular SAC can be disabled the first time that it is used. Furthermore, in the event that the SAC is not disabled, it can only be used to the extent of the user's clearance. For example, in the case where a bank is the service provider, a user can choose not to have access that allows money transfers over the Internet, which the SAC will enforce. Then if this client SAC is compromised it will still only have limited access, such as the ability to check the user's balance.

The client SAC must be a device that is either integrated into a computing base or that is easy to carry and can interact with a computing base. Smart cards are plastic cards that resemble magnetic strip cards, which most users are already familiar with and that most users associate with secure operations. Smart cards are an appropriate choice for the client SAC requirements. Current production versions of smart cards can have a cryptographic co-processor, 16 Kbytes of EEPROM, 2 Kbytes of RAM and 32 Kbytes of ROM. This is sufficient to implement a SAC client, although it still places some restrictions on the number of service providers that can be accessed by one card. In addition, smart cards and their readers are dropping in price, their memory size is increasing, and computer operating systems already support smart card readers.

3 Details of the SAC Approach

In a client-server configuration a client SAC interacts with a server SAC through an insecure medium. In this configuration, the user is not protected when interacting with either a client or server SAC alone. For example, in order to safely browse the web a user must interact with a server through a client site, both of which have safe areas of computation. If either the client or server site does not have the SAC, that particular interaction will not be protected.

The SAC, however, does not need to enforce the security policy of the computing system as a whole. The purpose of the SAC is to enforce a security policy tailored to the interactions of specific applications that a user needs to accomplish securely. The use of generic trusted areas that implement security critical functions to protect particular applications makes this approach different from approaches that use specific hardware for integrity checks of the computing system [9].

A security perimeter is defined as the imaginary boundary of a trusted computing base (TCB) within which all security-related functions are executed [8]. The trusted devices that compose a SAC define the SAC's security perimeter. The SAC can be as simple as a standalone smart card or secure coprocessor or it can be a collection of trusted processors, keyboards, displays, interfaces, etc.. The idea is to increase the number of safe operations that can be performed by adding trusted devices. As each new trusted device is added the security perimeter is extended.

3.1 SAC Data Structures

In the SAC approach data is organized in multi-level security containers. The container model used is a variation of the container model described in [5], which was proposed for military multi-level security documents, where each container is an abstraction for a set of data that has some attribute in common.

The SAC approach guarantees that only data to which a user has clearance is released outside the SAC. However, it is a function of the application software to present these containers and a description of the data inside the containers to the user for browsing and possible selection, after the SAC has approved access to them. To achieve this the SAC approach represents information by metadata, which has three fields: description, security level, and pointer. The metadata can be related to containers or to data. For containers the pointer field in the metadata information is a pointer to the container header, and for data this field is a pointer to actual data. The security level field represents the level a user must be cleared to in order to receive the data or header pointed to. A user who does not have the necessary clearance will not even know about the existence of the data or header to which the pointer points. The description field is used by the software that the SAC is protecting. It contains all information necessary to the software to interpret the metadata and to present it to the user for browsing and selection (e.g., MIME type information).

A header is a set of zero or more metadata entries. The possibility of a header having zero entries enables a pointer to a container header to have any security level without releasing additional covert information about the existence of classified data. In particular, this enables an unclassified pointer to a classified container header. This fact is used by the SAC approach to enable users to start their sessions with unclassified pointers to (possibly empty) container headers. An analysis of these pointers does not reveal the existence or the number of containers with classified data since the pointers can, and some will, point to empty container headers.

Consider a user making a query to request images of Santa Barbara, California. An example return value from the query could be the information shown in Table1. In practice there would be many entries, each with metadata that satisfies the query parameters. Since the security level of the example container "Satellite images of Santa Barbara from July 14th, 1998" is unclassified, any user would be able to request the container header. If a user requests the container header, the information in Table 2 will be sent in a secure way to the client SAC. This information represents the metadata that can be used to request further data or containers. Using containers inside other containers, any level of indirection, with possibly different clearance requirements, can be supported using this approach.

The user is shown representations of only the metadata from Table 2 to which he/she has clearance. For instance, if the user does not have secret clearance, the "Satellite Image 4 (infra-red)" metadata will not be shown. That is, the decrypted metadata about "Satellite Image 4 (infra-red)" will never leave the client SAC. Based on the information shown the user may then choose and request further metadata of interest.

DESCRIPTION	SECURITY LEVEL	POINTER
Container: "Satellite images of Santa Barbara. 07/14/ 98"	Unclassified	ContainerServer ::DB0::satellite: :SB071498
:	:	:

Table 1: Pointer to a container with satellite images

DESCRIPTION	SECURITY LEVEL	POINTER
Visible Container: "Container of classified Satellite Images"	Confidential	ConfidentialSer ver::DB0::sat::S Bvis071498
Infra-Red Image: "Satellite Image 3" (infra-red)	Unclassified	Mainserver::DB 1::sat::SBir0714 98
"Satellite Image 4" (infra-red)	Secret	SecretServer::D B0::sat::SBir07 1498

Table 2: Header of the container in Table 1

3.2 The Client-Server SAC Architecture

When used in a client-server configuration, the SAC architecture is as shown in Figure 2. The following subsections describe the client SAC and server SAC in more detail.

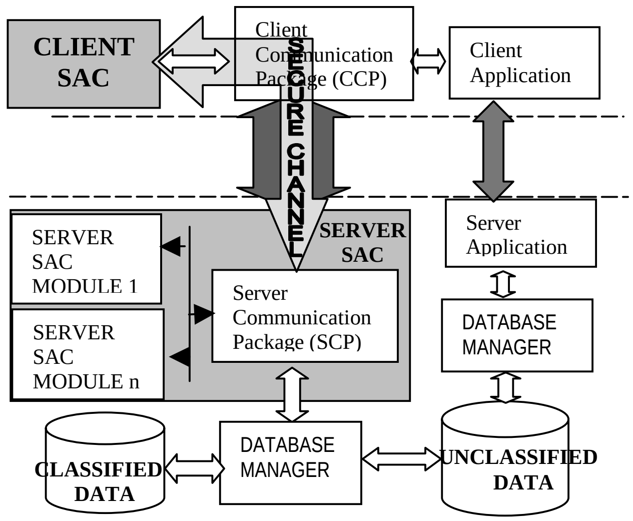

Figure 2: SAC in client-server configuration

3.2.1 The Client SAC

The client SAC and Client Communication Package (CCP) together are used to provide security without interpreting or presenting the data to users for browsing and selection. After the client SAC receives encrypted metadata from the server SAC, it decrypts it and uses the security level fields to check if a specific user has the appropriate clearances to receive the metadata. If the user is allowed to access the data, then the client SAC hands the decrypted metadata to the particular application via the CCP. The application then presents this metadata to the user to browse and query. The application software primarily uses the description field of the metadata to accomplish this, but in some cases may additionally use the other fields.

The CCP is not inside the security perimeter and is not considered a safe area. Because of this it does not deal with clear text data until the data has been authorized for access. The application software interacts directly with the CCP using generic functions, it does not interact with the client SAC. The generic functions provided deal with user and server authentication, requests for headers, and requests for data. Each of these is discussed in the next subsections.

The initial queries of the server application do not involve the SAC's; that is, the initial query of the server is made directly from the client application to the server application over an unprotected communication channel. The metadata returned by this query is a set of unclassified descriptions of the data or container in addition to their pointers. The disadvantage of this approach is that a user may be returned a pointer to a container that has no data. This, however, is not believed to be critical since the user will immediately find that he/she is following a link that will result in no data after requesting the container header with no data. The advantage of this approach is that if a user wishes to browse only unclassified data, then the SAC components will not be involved and some may not even be present.

3.2.2 User and Server Authentication

In order to start a session with a secure server the user must be authenticated to the client SAC and the client SAC and server SAC must mutually authenticate themselves. If this is not done or if the authentication fails, then any additional requests from the application software to the CCP will be denied.

The application software interacts with the user to request a pin code. After this pin code has been entered, the application software calls a CCP function "boolean AskSACAuthentication(char *pin)". This function takes as input a 4 digit pin code, represented as an array of characters. The choice of a 4 digit pin code is used due to an embedded functionality in most smart cards, which uses a 4 digit pin code to authenticate a user to the card. The card does not allow any transaction with it without the correct pin code. Although four is a small number of digits, which could suggest a brute force attack, the card only allows three wrong pin codes to be presented in a sequence. If a fourth wrong pin code is presented, the card will lock itself and will not work anymore. In order to unlock a card and return it to service, a special sequence of commands must be invoked on the locked card along with the use of a secret master key.

After the user has been authenticated to the client SAC the client SAC uses the Feige-Fiat-Shamir identification scheme [3] to identify itself to the server SAC. It then uses the Diffie-Hellman key-exchange algorithm [2] to agree on a session key. To prevent a man-in-the-middle attack, the client SAC and the server SAC use digital signatures based on the RSA algorithm [7] to sign the key-exchange. All information exchanged by the SACs will then be protected against eavesdropping by DES [6] using the session key. This provides a virtual secure channel between the client SAC and the server SAC.

3.2.3 Request for Headers

From the client SAC perspective, the starting point of any transaction request is an unclassified metadata. This metadata is the result of a query run without the use of any SAC component. If the metadata pointer points to a container, then a header describing this container can be requested. The headers are all encrypted and are assumed to be small, so they can be efficiently decrypted inside the security perimeter of the client SAC.

In order to request a header, the application software calls the function "int RequestHeader(DataPtr req, DataPtr **ans)". This function uses the struct DataPtr, which is defined as:

```
typedef struct _DataPtr{
        char description[100];
        char pointer[50];
        unsigned char securityLevel;
} DataPtr;
```

When requesting a header, the security level field of DataPtr is filled in by untrusted software (including CCP), which resides outside the client SAC. For this reason, the SAC approach does not rely solely on this information to grant or deny access to data. Thus, the SAC approach guarantees that even if a user provides a wrong or malicious security level to the client SAC, the server will not be misled into sending data that the user is not allowed to access. To assure this the server SAC will perform a backup check between the claimed security label and the security label that is linked to the header, which is stored in the server database.

When the function returns, the application software will receive an array of DataPtr in the ans argument. The number of elements in this array is specified by the integer returned by the function. The elements returned are only those elements of the container to which this user has access.

3.2.4 Request for Data

A request for data can be made if the user has a DataPtr structure that points to data. The request usually happens after a header is received and the available data is shown to the user. The user chooses the data of interest and the application software will make a request on behalf of the user. It is the function of the application software to arrange the data to be shown to the user and to handle his/her selections.

When the application software wants data, it calls the function "boolean RequestData(DataPtr req, char filename[8])". The requested data is specified in the argument req, in the same way that requests for headers are specified. The function returns true if the data has been received by the client computer, and the filename argument contains a pointer to the file that has the requested data in clear text. Currently it is the responsibility of the application software to sanitize the file after using it.

3.3 The Server SAC

The individual Server SAC modules and Server Communication Package (SCP) are sensitive software that must be protected. The protection of these modules is centralized in the server that uses the SAC approach for security. The administrator of the server must ensure the protection of this server, which is usually easier than ensuring the protection of the distributed clients. In addition, it is expected that the system administrator of this server is more security aware than the individual clients. For these reasons, the tamper resistant area where these modules run may be implemented using existing hardware on the server, such as implementing it in kernel space.

The Server SAC has a database, which contains client SAC public keys to identify and interact with each user. Every time a user initiates an authentication with the server the SCP spawns a new server SAC module with properly initiated parameters, including the user ID, the user's public key, and the server private key. The SCP also checks if the specific user needs to have any of his/her accesses updated, such as for adding a new service provider. To accomplish this, the SCP uses a database of access control list (ACL) modifications. The modification, if there is any, is requested by the server SAC through the secure channel, immediately after the channel is established.

The Safe Area of Computation in the server is composed of the dynamic set of server SAC modules, the databases that store keys and ACL modifications, and the Server Communication Package. This area, which must be well protected, is shown in Figure 3

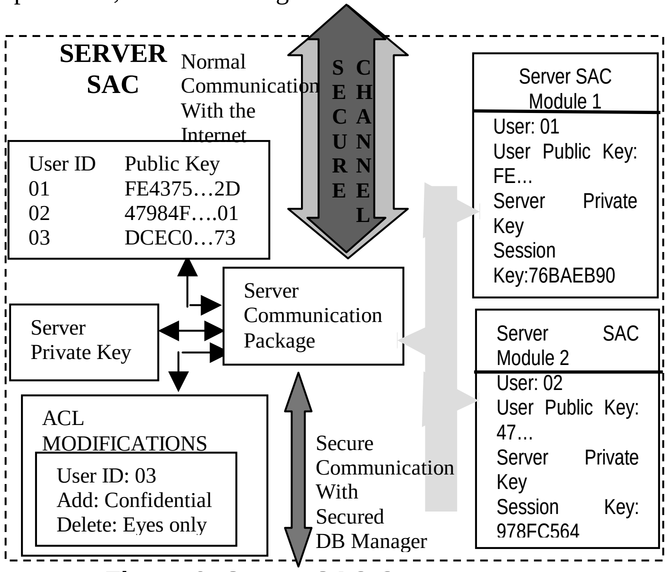

Figure 3: Server SAC Components.

The following subsections explain the communication of the server with the Internet and with the secured database and discuss the security of the server.

3.3.1 Server SAC Communication with the Internet

Communication with the Internet is handled by the Server Communication Package (SCP). Every time a client SAC issues a request to the server through the Client Communication Package, the SCP sends the request to a specific server SAC module. After this server SAC module processes the request it sends an answer to the SCP, which sends it back through the secure connection used by the client to issue the request.

Every communication out to the Internet and from the Internet is encrypted when the secure channel is used. Every pair of server SAC module/client SAC have specific cryptographic keys to encrypt and decrypt their transactions. This means that although only one secure channel is shown in Figure 3, there will be one distinct secure channel for each client. Thus, any adversary in the Internet cannot maliciously eavesdrop on a transaction that is conducted through the secure channel. In addition, since the keys are unique, malicious impersonation is not possible.

3.3.2 Server SAC Communication with the Secured Database Manager

The Server Communication Package also manages communication with a secured database manager, which handles all accesses to classified data. Since the database manager sends sensitive unencrypted data in answer to requests, the line of communication between the SCP and the secured database manager must be secure.

The request that is sent to the secured database manager is the data pointer sent from the client SAC and decrypted by the server SAC. Neither the server SAC nor the SCP interprets the pointer. As a result, the SAC approach can accommodate any database format without the need for any change, assuming the pointers are consistent with the secured database manager format. The requirement is that the database manager exports a function call, so that the SCP can send the pointer and receive back the security level of this data in addition to the data requested.

3.3.3 Security of the Server

A proper implementation of the server is an implementation where the modules shown in Figure 3 are kept in a safe area. In addition, this implementation trusts the data received from the secure communication with the secured database. This is a blind trust and nothing is done to guarantee the integrity of the data.

There is a possibility of denial of service attacks due to the fact that in every transaction with a client the server first needs to identify which user is requesting the transaction. This verification is accomplished by appending the user ID encrypted with the server's public key to the transaction. A malicious attacker cannot benefit from this since he/she will not be able to complete an authentication nor be able to get the session key used by a legitimate user interacting with the server. The attacker can, however, flood the server with transactions from possibly existing user IDs. This will require the server to process all these requests and may force legitimate users to restart sessions because of time-outs.

4 A Prototype SAC for Internet Transactions

As a proof of concept, the SAC approach was implemented for secure Internet transactions. It uses the configuration shown in Figure 4 to protect the transactions. The data used in this implementation is generic MIME data. The intent was to explore the ability of a browser to directly handle many types of MIME data securely. The transactions use the Netscape Navigator browser, and Plug in technology is used to communicate with the Client Communication Package (CCP).

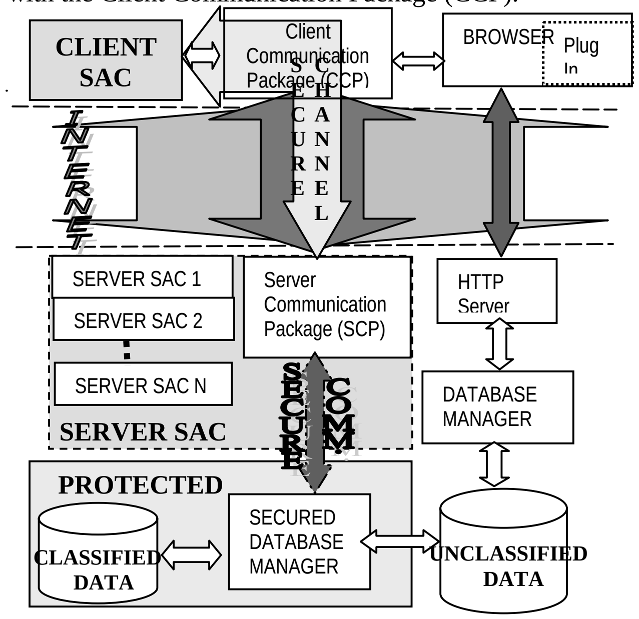

Figure 4: Safe Internet transactions using the SAC approach

When a client visits a secure server, the first (main) page in the secure server loads the Plug in, which loads the CCP and the client SAC. The client SAC is presently implemented by a program in the user's computer, but is currently being implemented using smart cards. The Plug in pops up a window requesting the pin code from the user. After the user enters this information, it calls the authentication function.

If the user is authenticated, one of the frames in the main page allows the user to request a search from the secure server. The answer to this search is a set of unclassified pointers to collections.

The following subsections detail the components in Figure 4 and describe the steps of a simple transaction.

4.1 Components of the Internet Prototype

In this section each of the components is described.

4.1.1 Plug In

The current version of the implementation assumes a user interacting with the secure server using the Netscape Navigator browser. The Plug in technology was chosen to implement the active interaction between the browser and the Client Communication Package (CCP). The same design may be used with ActiveX controls for interactions using the Microsoft Internet Explorer browser and Java applets with special capabilities.

By implementing this program as a Plug in, all the visual interfaces are provided to the user. The user interacts with the server using these visual interfaces, and the Plug in program interacts with the CCP when it needs classified data.

4.1.2 Client Communication Package

While the Plug in has a specific design developed to satisfy a particular application and server, the Client Communication Package (CCP) is designed to be generic and to support the functionality of the SAC approach. The CCP does not require a safe area and never deals with sensitive plain text data that has not been first approved for release by the Client SAC.

The CCP implements all functionality that can run in the unprotected area. This includes manipulation of sensitive data that has been cleared by the Client SAC. In addition, it implements the Internet communication line, which is used to exchange encrypted data between the client SAC and the server SAC and in which the virtual secure channel is established.

4.1.3 Client SAC

The Client SAC is assumed to have limited memory and low processing power because it will eventually be housed in a smart card. The code that provides security, as well the cryptographic keys, are stored inside its security perimeter.

Because the client SAC has low processing power, it would take a long time to decrypt large files. Therefore, file decryption is performed by the CCP, outside of the client SAC. The server SAC encrypts each file with a unique one-time key and sends the encrypted result to the CCP using a standard Internet connection. The key is then sent through the secure channel to the client SAC. The client SAC can then hand the CCP this key to decrypt sensitive data file. Thus, the client SAC always controls the decryption of sensitive data. This sensitive data is either used directly by the CCP, such as metadata in the clear, or is a key that can be used by the CCP to decrypt other sensitive data. The sensitive data or key will only leave the security perimeter if the access control list (ACL) inside the client SAC certifies that this user is cleared to access the data.

4.1.4 HTTP Server
The SAC approach does not make any assumption about the http server. That is, the generic nature of the approach does not require any particular http server. Although not currently implemented, the http server could transmit and receive encrypted sensitive data that is sent from the CCP to the SCP or vice-versa without compromising the security of the data. This would be accomplished by adding a communication channel between the http server and the SCP using a technology that extends the http server, e.g. CGI and servlets.

4.1.5 Secured Database Manager
The secured database manager handles sensitive data and should be kept in a protected area satisfying only requests from trusted sources. No sensitive data will be leaked if the SCP runs in a safe area and its communication with the secured database manager is secure.

4.1.6 Server Communication Package
The Server Communication Package (SCP), unlike the Client Communication Package, needs to run in a protected area in the server. The reason for this is that it interacts directly with the secured database manager to request sensitive data.

The function of the SCP is to interact with different clients and to dynamically allocate a specific Server SAC for each client. Each server SAC establishes a separate secure channel with the corresponding client SAC. A spawned server SAC uses the SCP as an auxiliary module to communicate with its respective client SAC in a secure way. In addition, the SCP is used to communicate and request data from the secured database manager.

4.1.7 Server SAC Module
One Server SAC module is spawned for each user that is successfully authenticated to the server. The Server SAC module has the necessary keys to interact with a particular client SAC.

The server SAC modules are kept inside the server SAC, while interacting with the corresponding client. When the client SAC ends a session with the server, the corresponding server SAC module is deleted. However, a session between a client and a server SAC module often has transactions that cannot be anticipated. Therefore, it may be desirable to keep that specific server SAC module around. Time-outs are used in the current implementation to solve this problem and to determine if a session has finished. This can cause some users to restart a session after a time-out.

4.1.8 Database Manager
In order to lower the workload when dealing with unclassified data, the server may run a second database manager that does not require a protected area. The system is responsible for assuring that this database manager can only access non-sensitive data. Using this approach the http server can request non-sensitive data directly from the unclassified database manager. This unclassified data can be used later to fetch classified data through the Safe Areas of Computation.

4.2 Example Session
The following sections describe in detail each step of a session. They discuss what tests and functions are executed in order to provide security.

4.2.1 Authentication
The Plug in interacts with the user, with the browser, and with the Client Communication Package providing the particular functionality needed by a specific application. The Plug in, Client Communication Package, and Client SAC, are implemented as programs that run in the user's computer and that must be installed before any transaction can be accomplished. The Client SAC, which is a very sensitive set of modules, is currently being migrated to a smart card.

When accessing the protected server, the user displays a page that resides in this server. This page will load the Plug in, which starts the client SAC and the Client Communication Package programs. A window pops up requesting the user to enter a pin code in order to authenticate the user to the client SAC.

After the user enters a pin code and clicks the OK button the process of authentication starts. If the user enters the correct pin code, he/she will be authenticated to the client SAC. The client SAC will then authenticate itself to the server SAC, using an Internet connection opened by the CCP. The result of the successful authentication process is a session key that provides a secure channel over the standard Internet connection.

4.2.2 Request for Container Pointers
After the authentication and secure channel establishment takes place, the user can request containers by making queries. The browser issues a query, through an insecure Internet connection to the http server. The http server queries the database manager, which will run the query on the unclassified data and return unclassified information that points to container headers. This information is composed of text information to be displayed to the user, an internal pointer to be used when requesting the container, and the security level of each container, as was shown in Table 1.

The list of available containers is returned to the user's browser through the Internet connection. Since this information is unclassified it is displayed to the user as a list without restriction.

4.2.3 Request for Container
The user selects a container from the list. When the user clicks a “Get Container” button, requesting the container,

the Plug in sends this request to the Client Communication Package (CCP), which forwards the request to the client SAC, specifying the pointer to the container header and its security level. If the user has the clearance necessary to request the container header, as specified by the security level of the container and the access control list, then the client SAC uses the secure channel to request the container header. Since the data used in the request is sent through the secure channel, neither the CCP nor the Internet can tamper with the request.

The request includes the security level of the container to which the clearance was checked and a header with the user ID. The security level will be used to prevent software outside the client SAC from maliciously faking a request. The user ID is used in order to identify which spawned server SAC module has established a secure channel with the particular user.

4.2.4 Server Processing of Container Requests

The appropriated server SAC module, from the possibly many spawned by the Server Communication Package (SCP), receives a request for a container from the client SAC that is associated with it. The server SAC uses the SCP to request the container specified by the pointer sent by the client SAC. The SCP requests the container from the Secured Database Manager using this pointer.

The Secured Database Manager fetches the requested data (a container header) and returns it to the SCP, together with the security level associated with this data. The SCP returns the container with this security level to the appropriated server SAC module. The result is tested against the security level sent by the client SAC when requesting the specific container. If both security levels are not the same the operation fails and an error is sent to the client SAC. This is the second time that the security level of the requested data is checked and will catch malicious requests, where the security level has been changed before it is sent to the client SAC. In the case where the security levels agree, the server SAC will send the container and the security level received from the SCP back to the client SAC through the secure channel.

4.2.5 Requests for Data

The client SAC receives the container header through the secure channel and processes it. Before processing any of the data inside the header the client SAC uses the access control list and the security level of the header received from the server SAC, to check if the user has access to this header. This is the third time that the clearance is checked and should never fail. A failure at this point is unexpected and should be logged for further studies.

The client SAC checks the user clearance for each data or container embedded in the container, as represented in the container header. It sends the client communication package only the information for the data and containers to which the user has clearance. The CCP will send this information to the Plug in, which shows it to the user for browsing and selection.

After browsing the information about the metadata inside the container, the user can select one of interest by clicking the “Get Data” button. The client SAC makes the request in the same way that it requests container headers.

4.2.6 Server Processing of Data Requests

The process of directing a request for data to the correct server SAC module is the same as the process of directing a request for a container. The correct server SAC module again uses the Server Communication Package (SCP) to request the data from the secured database manager. When the server SAC receives the potentially large data back, it generates a one-time key to encrypt this data. The server SAC will only continue operations, as in the request for a container header, if the security level of the data reported by the secured database manager agrees with the security level reported by the client SAC when requesting the data.

The server SAC encrypts the data using DES and the generated one-time key, and sends it to the SCP, which sends it to the Client Communication Package (CCP) via the standard Internet connection between them (outside the secure channel). Additionally, the server SAC uses the secure channel to send the key used for the encryption and the security level of the data to the client SAC. The client SAC will release the key to the CCP only if the user is cleared to access this data. This is the third check of clearance, analogous to the procedure used when requesting container headers.

4.2.7 Results of Requesting Data

The Client Communication Package receives the encrypted data from the Server Communication Package and the key for the decryption of this data from the client SAC. It uses this key to decrypt the data and to write it to a local file, which is returned to the Plug in. The Plug in shows the data to the user as appropriate to the particular application.

The Plug in must interpret the data in a way appropriate for the application. The SAC approach only delivers the data to the Plug in without interpreting it. A specific application can use the description field to specify possible formats. In the prototype, the description field includes the MIME type of the data, which is easily interpreted by the browser.

4.2.8 Access Control List Updates

The steps above describe a simple session. Additionally, the current implementation supports dynamic modification of the client SAC access control list (ACL). For this the server SAC may send a request to the client SAC to update its ACL when the client SAC authenticates itself. The request is sent through the secure channel just after it is established. This request is a list of accesses to add and delete. It is not possible to tamper with this data since it uses the established secure channel, which uses a different session key each time.

5 Conclusions and Future Work

The technologies currently used for interactions between users and sites on the World Wide Web have many security weaknesses. This paper presented the SAC approach, which uses trusted devices to improve the security of user interactions with a site. An advantage of the SAC approach is that standard applications only need to be modified to use the generic functions provided by the SAC Client Communication Package to get secure access to sensitive data. The prototype Internet SAC showed how this could be easily implemented using Plug in technology. This initial implementation was tested exchanging MIME data, particularly images and html pages. Images were classified with different security levels to demonstrate the container model.

The use of safe areas of computation for protecting user transactions is intended to make these transactions more secure. An advanced test bed for the current system is being implemented to protect Internet transactions of a user with the Alexandria digital library [4]. For this test bed two smart card operating systems are currently being used: CardOS M3 from Siemens and MultOS from MAOSCO. A test bed for protecting a financial institution is also planned.

The SAC approach uses access control lists implemented on the client SAC. Employing a user-specific, or device-specific access control list enables the personalization of the accesses and restrictions a user will have (e.g. only allowing the checking of a bank account balance and not allowing withdrawals). This personalization has the advantage of providing access to only the services that a user signs up for, which limits the possibility of impersonation of the user to only these services.

Another advantage of storing the access control lists on the client SAC is that when the server receives a request for a service, the client SAC has already authorized this request at the client host. The result is that the server does not need to again verify that the specific client has the necessary access rights for the requested service. However, if this redundant verification is desirable, it could be done for a small set of critical services. The security provided would be in addition to that already provided by the client SAC alone.

The implementation of the protected browser demonstrated that much of the burden of providing a user-friendly interface is placed on the application specific non-secure software. It is not a function of the SAC components. In addition, the SAC approach does not impose additional requirements that would make the application software less user-friendly.

The grouping of users and/or data providers is one way to lower the memory requirements for the client SAC. This means, however, that private keys will be shared inside a group, which represents a security weakness. The grouping used and the feasibility of grouping is an area for further investigation.

It is possible to charge for information using electronic commerce techniques and electronic cash. There are already some standards for these transactions. The best way to do this and the best technology available is another area to be investigated.

Currently, traffic analysis prevention is not considered to be a requirement; however, in future implementations it may be required. Thus, the interactions between the user's computer and the secure server for the first search will need to be protected. This could be accomplished using the protection of SSL. However, even the use of SSL will not prevent a man-in-the-middle attack whose goal is traffic analysis [1]. Because of this, functions for protecting queries will need to be implemented by the SAC for use if traffic analysis protection is desirable.

6 Acknowledgements

This research was partially supported by PulsePoint Communications and the University of California through a Micro Grant. It was also partially supported by Siemens AG and Mondex International.

References

1. F. De Paoli, A.L. dos Santos, and R. A. Kemmerer, “Web Browsers and Security”, Mobile Agents and Security, LNCS 1419, pp. 235-256, Springer-Verlag, 1998.
2. W. Diffie and M. E. Hellman, “New directions in cryptography”, IEEE Transactions on Information Theory, v. IT-22, n. 6, Nov. 1976.
3. U. Feige, A. Fiat, and A. Shamir, “Zero knowledge proofs of identity”, Proceedings of the 19th annual ACM symposium on the theory of computing, 1987.
4. J. Frew, et. al., “The Alexandria Digital Library Architecture”, Proc. of the Second European Conference on Research and Advance Technology for Digital Libraries, September 1998.
5. C. E. Landwehr, C. L. Heitmeyer, and J. McLean, “A Security Model for Military Message Systems,” ACM Trans. on Computer Systems Vol. 9, No. 3 (Aug. 1984), pp. 198-222.
6. NBS FIPS PUB 46, “Data Encryption Standard”, National Bureau of Standards, U.S. Department of Commerce, 1977.
7. R. L. Rivest, A. Shamir, and L. M. Adleman,”A method for obtaining digital signatures and public-key cryptosystems”, Communications of the ACM, v. 21, n.2, Feb 1978.
8. D. Russel and G. T. Gangemi Sr., "Computer Security Basics", O'Reilly & Associates, Inc., July 1991.
9. B. S. Yee. “Using Secure Coprocessors”. PhD dissertation, Carnegie Mellon University, 1994.