A Smartcard Fault-tolerant Authentication Server

Laurent Blain[*]
LAAS-CNRS

Yves Deswarte
LAAS-CNRS & INRIA

7, Avenue du Colonel Roche
31077 Toulouse Cedex (France)
Telephone: +33 / 61 33 62 88
Fax: +33 / 61 33 64 11
E-mail: deswarte@laas.fr

Abstract: In this paper we present a fault-tolerant and intrusion-tolerant authentication server. This server is composed of several sites, each one managed by a different security administrator. We describe the registration and authentication protocols based on smartcards. Those protocols have been implemented by using Bull CP8 MP smartcards. We show how to make them secure in spite of possible failures on a minority of authentication sites. Then we describe a possible implementation based on public-key smartcards achieving the same properties of fault tolerance and intrusion tolerance.

Keywords: Authentication protocol, fault tolerance, intrusion tolerance.

I. Problem statement

User authentication, i.e., user identification and identity verification, is a necessary step to security: in most cases, access to secure data or services is granted or denied according to the identity of the user requesting the access, and when investigating security failures, it is important to know which users have performed which operations. Moreover, authentication reliability is critical for security since, in case of failure, legitimate access would be denied to authorized users and/or illicit access would be

[*] Currently with GERII - INSA/DGE
31077 Toulouse, France, tel: +33 / 61 55 98 41, fax: +33 / 61 55 98 08
E-mail: blain@insage.insa-tlse.fr

granted to unauthorized users, enabling them to perform unauthorized actions with total impunity.

In distributed systems, user authentication can be realized by the user terminal (or workstation), by the remote application server, or by a dedicated security server. Terminal-based authentication is usually unreliable since it is relatively easy for a user to control or modify his terminal or workstation to impersonate another user. On the other hand, in large distributed systems, it is generally impracticable to implement authentication within application servers because such servers can be too numerous, making too complex the security management: for instance to register a new user or to remove a former user, the user identity (and corresponding authentication information) has to be added or removed consistently on every application server. A more efficient solution consists in specializing a server for the authentication; this authentication server has to be trusted by the application servers as being able to guarantee user identities. With this solution, the user has to be authenticated only once whatever servers he wishes to access.

For the authentication server to be trusted, it has to be strongly protected by physical components and personal clearances. But none of these barriers can ensure a perfect security or reliability, as an accidental fault or an intrusion may always occur. Moreover, administrators and operators of the authentication server have to be trusted since they can easily register illegitimate users, remove or modify legitimate users information, or help illegitimate users to impersonate legitimate users. New techniques should then be implemented in order to tolerate possible intrusions (including by administrators) and accidental faults (including operator mistakes). Accidental faults can be easily addressed by well known techniques (e.g., back-up servers), but intrusions, while seldom envisaged, are an increasing threat in distributed systems and require novel techniques.

In this paper we present an authentication server with the corresponding protocols able to tolerate as well accidental faults as intrusions. Authentication is based on smartcards and the protocols described here carry out registration (i.e., generation of a user's smartcard), authentication and session keys generation. The system has been designed so as to avoid any single point of failure, whether it would be a person, a smartcard or a machine. The protocols have been studied for both public key and secret key cryptographic systems, but only the latter has been implemented on a prototype.

Our solution is based on the separation of duties between different security administrators, different smartcards, and different machines. The authentication server is composed of several sites, each one being managed by a different security administrator who owns his own master smartcard (a *mothercard*). Each site has knowledge of only a part of the secret data and this part (or a set of these parts) is not sufficient to succeed in performing an intrusion. The user's smartcard is divided into several areas. Each one is readable and writable by one site's mothercard only. An agreement of a majority of the sites (i.e., the administrators and the mothercards) is necessary to achieve the registration and the authentication. By this way, the system tolerates failures (due to accidental or intentional faults) of a minority of sites, smartcards or security administrators.

All the involved protocols have been implemented on a prototype in the framework on the ESPRIT project Delta-4 [Powell 91] using the Bull CP8 MP smartcards. In Section IV.2, we describe how similar protocols with the same properties can be implemented by using public key smartcards.

II. Comparison with previous work

A typical example of an authentication server is Kerberos [Steiner 88] whose authentication protocol is based on password verification using a Needham-Schroeder's authentication protocol [Needham 78]. The Kerberos server maintains a database containing all users' passwords and all servers' permanent cryptographic keys. When a user logs in a workstation, the user password present in the database is used by Kerberos to generate a session ticket including a session key. The user is authenticated by deciphering the ticket by means of the password typed on the workstation. Afterwards, whenever the user workstation wishes to establish a connection to a remote server, it presents the session ticket to the Kerberos server to get a new ticket based on the remote server's permanent key.

The authentication server is a single point of failure and thus must be strongly protected against different kind of accidental or intentional faults. Kerberos is built so as to tolerate at least some accidental faults by means of replication: on-line slave back-up servers enable users to be authenticated even if the master Kerberos server is out of service (but the database can be updated only on the master server). But if an intruder succeeds in gaining administrator's privileges on the Kerberos server or if an administrator, by mistake or by malice, performs illegitimate operations, the whole distributed system security is compromised. In particular, the confidentiality of the user password database is very critical.

This point can be improved by using smartcard based authentication protocols. Let us give an example of such protocols: when the user logs in a workstation, a challenge is issued from the server; the response to this challenge is computed by the user smartcard according to a secret key, specific of this user smartcard, registered within the protected memory of the smartcard; this response is sent back to the server which compares it to the response computed by the server smartcard, according to the server smartcard key and the user smartcard identity. The two responses are to be identical because the user smartcard key has been created by the server smartcard (called master smartcard or *mothercard*), from its own secret key and the user smartcard identity during the user smartcard personalization phase. Such mothercards limit the possibilities of external intrusions from people who do not possess them, but the administrator remains a single point of failure, as well as his mother smartcard.

In conclusion, the centralized approach for an authentication server leads to the existence of a single point of failure, the machine where the server runs. An intrusion or accidental fault on this server is not tolerated and, worse than that, the security administrator can perform illicit actions. The smartcards improve the security with respect to external intruders but introduce new components, the master smartcards which become critical. In the rest of the paper, we will describe how to build a distributed fault tolerant and intrusion tolerant authentication server. We will show how to use smartcards so that no single master smartcard can be a single point of failure.

III. A fault tolerant and intrusion tolerant server

III.1. Several authentication sites

As we said above a centralized authentication server is a single point of failure with respect to accidental and intentional faults. A single site has to be trusted by all the users of the system and it must cope with all possible faults. This is a realistic objective for external intrusions if strong physical and logical protections are implemented, but unrealistic with respect to administrator's illegitimate operations.

In order to address this problem, the authentication server is composed of several sites called authentication sites. All the authentication sites have the same functions and the same software, they differ only by the data they own. There exist two kinds of data:

- the public data which are replicated on the different sites,
- the private data which are secret data, different on each site: for instance diversified authentication keys (see §IV.1.2), different session keys (see §IV.1.4) or shadows of a threshold scheme [Blain 92].

The sites are all running at the same time, all playing the same role: there is no master/backup sites. From the viewpoint of the user's workstation, the link established with the server is composed of *N* independent communications with the *N* sites. All the operations are repeated *N* times and the user must keep *N* private data necessary to communicate with the *N* sites.

The authentication server, composed of the *N* authentication sites, is a trusted part of the system. Its role is to authenticate users in order to permit them to access remote servers. Therefore, it has to be trusted by both user sites and remote servers. The latter receive from the authentication server information enabling to check the claimed identity of a requesting user (in our case, this information is sent during authorization phase and not during authentication, and it will not be described here).

However, while the server is trusted, this is not necessary the case of each of its sites. Indeed, from the viewpoint of another site (user site, remote server or even an other authentication site), a single authentication site is not trusted: but the majority of them are. It means that a site does not take into account only one or a minority of identical messages coming from authentication sites but only a majority ((N+1)/2). If the site receives a majority of identical messages (identical stands for the same meaning, not for the same value), it knows it can consider the message as correct. For instance, only if the authentication is accepted by a majority of authentication sites, the user site considers itself as authenticated. On the contrary, if the authentication is denied by a majority the user site considers it is not authenticated. The main assumption is that at most a minority of authentication sites will fail and that a majority will always be safe and secure.

The general protocol can be described as follows (Fig. 1):

① the user site broadcasts its authentication request to all the authentication sites,

② each authentication site performs independently the authentication by using the secret data,

③ each authentication site broadcasts its decision to the other authentication sites,

④ each one votes on the decisions,

⑤ each one sends the final result to the user site,

⑥ the user site votes on the replies to be sure it has been successfully authenticated by a majority of authentication sites; if so, it can send other types of requests to the authorization server in order to access remote servers.

The steps 3, 4, and 5 permit to avoid problems due to the network, the user site or the authentication sites which would lead to an inconsistency of the states of the sites - i.e., a minority of the sites has taken a decision different of the majority's. Indeed as we increase the number of involved sites we increase as well the probability of faults, it is thus necessary to add some mechanisms to tolerate them. After this agreement, all the correct authentication sites are sure of the identity of the requesting user, and each one can reply OK or not OK - i.e., the result of the vote - to the user site.

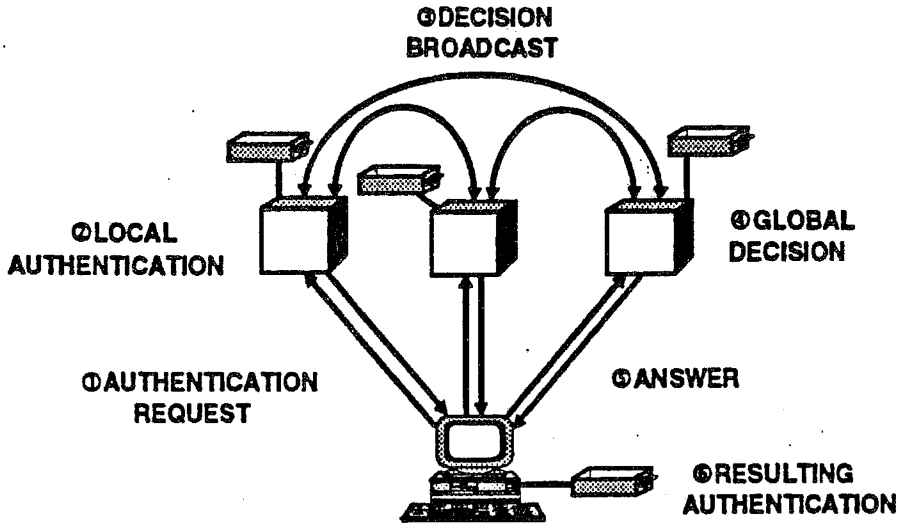

Figure 1: Authentication protocol between a user site and the authentication sites.

This architecture and the corresponding protocol are intended to provide a safe and secure service in spite of possible (*N-1*)/2 failed authentication sites. It deals with failures produced by accidental faults, external intrusions or user attempts to extend their rights. We have however still to solve the problem of the administrator.

III.2. Several security administrators

To address the problem of the security administrator, each authentication site is managed by a different security administrator. A security administrator possesses all the privileges on his site and thus on the data stored on it, but he has no rights on the other sites. That means he does not know the values of the remote private items and he cannot modify or destroy the public items which are not located on his site. The consequence of this scheme is that an illicit operation carried out by the administrators can be successful only if at least a majority of them collude. *N* authentication sites and *N* security administrators tolerate about *N*/2 intrusions of whatever origin: external, user or security administrator.

All the protocols provided by the authentication server, registration, authentication and session key generation, are studied to take into account this separation of duties between the administrators and the authentication sites. This has essentially three consequences:

① a confidential item stored on one authentication site may not be disclosed to others during the protocols,

② the access to public or private items related to an authentication site is strictly limited to this site under the responsibility of its administrator,

③ each message issued from an authentication site cannot be issued from others (the other sites must be sure of the origin of a message).

This means that a security administrator does not only own all privileges on data managed by his site but also on those issued from it and nothing on those managed or issued by the others. This is essential when using smartcards for authentication.

III.3. Several mothercards

In our approach we have adopted smartcards for the authentication. Unfortunately, the current implementation of smartcards is not very suitable for the fault-tolerance techniques. Indeed, as we have said above, there generally exists one master card able to read and write the users' cards. This master card is thus a single point of failure since a malicious user or administrator, with this card, can create false user cards and then impersonate correct registered users. To address this problem, we have developed a new technique compatible with existing smartcards.

We use *N* different mothercards, one for each site, and one grand-mother card which generates all the other cards. Each security administrator is given a mothercard which is able to authenticate all the user cards on one authentication site[1]. The creation of a new user card requests the collaboration of the security administrators, and at the end of this phase none of the mothercards or of the grand-mothercard has full access rights on the user's card. Like for the security administrators, a collusion of a majority of mothercards is necessary to succeed an intrusion. The grand-mothercard which is unique is necessary to create new mothercards or user cards. But it is not a single point of failure. Indeed, the only malicious action that the owner of this card can do is to make incorrect cards and thus to make the registration service unavailable. The error can be easily detected and the problem can be quickly fixed. In order to prevent the complete unavailability of the service, the grand-mothercard can be replicated and one copy given to each of the security administrators. Since the power of this card on the others is temporary (between the creation of the user card and the registration by the administrator's cards), it does not matter who possesses effectively it (assuming this is someone considered as relatively trusted).

IV. Smartcard protocols based on separation of duties

In this part, we shall describe the different protocols for two cryptographic techniques. The first one is based on secret shared keys and has been implemented on Bull CP8 MP (MP stands for Multi Purpose) smartcards. The second is based on public keys and was studied and formally described, but not implemented yet.

[1] Several mothercards can be generated for each authentication sites, one for each person who can be administrator of this site. Of course, the same person cannot be administrator of several sites.

IV.1. Protocols for shared secret keys

The Bull CP8 MP smartcards present two characteristics we were interested in:

① an authentication protocol based on DES is provided; this is a classical challenge-response protocol between a user smartcard and a mothercard from which the secret key was issued,

② the card is divided into several areas; each area is protected by a different key. This last feature makes the implementation of fault and intrusion tolerant protocols possible.

Each card has a different identity, an ID, created by the manufacturing process, which cannot be changed and which is absolutely unique. There exist three kinds of cards:

- the grand-mothercard which can define the different areas of the other cards and which initialise the access-control information,
- the mothercards which can modify the content and the access-control information of the areas they are authorized to access,
- the user smartcards which cannot modify either their own structure or other smartcards: the only operations allowed are those authorized by the access-control mechanism.

A Bull CP8 smartcard is composed of several areas, each one possesses its own access control mechanism which is based on an access key and rights. The key is necessary to enter the area, and the rights define the operations authorized for the different parts of the area. Each area contains two sorts of zones:

- a write-only confidential zone, where are stored the access keys and data which can be processed by the card but not exported outside of it,
- a read-write transaction zone where are stored data which can be exported outside the card.

We use two functions provided by the CP8 MP smartcards: the diversification and the certificate. The diversification is used by the grand-mothercard and the mothercard to issue a new access key on another card from an access key they possess. This new key is the result of a one-way function (D) based on DES with the known key and the ID of the card as inputs. This new diversified key will be then specific for the card (the ID is unique) and can be regenerated at anytime by the mothercard or the grand-mothercard. But someone who does not possess the initial access key is unable to create the diversified key and thus to enter the protected area. The diversification function is also applied to other keys such as the authentication keys. The diversification function permits to a mothercard to generate many cards (as many as different IDs) without storing all the secret keys stored on the users smartcards: one key (the generator of all others) is sufficient. Obviously, the function must be carefully built, first to prevent the possibility to find the generator from a diversified key, second to avoid collisions - i.e., two different IDs producing the same keys.

The other function is the certification (C) which is also a one-way function but with different inputs and with a different result. The certification is intended to check that a value is stored in the transaction area. The inputs are an external random number, a local authentication key and the number to be certified. The result is a certificate which can be exported or not outside the card. It is impossible to compute the authentication key or the number from the certificate without breaking DES. Moreover, each mothercard provides a random number generator which permits to compute certificates and to generate new messages for each new authentication and so to avoid the replay attacks.

The security of all these cards is ensured by their physical protection against unauthorized read or write access, by a PIN the card user must enter to validate the subsequent access to the card and by the access control mechanisms of the different areas.

IV.1.1. Creation of mothercards

The grand-mothercard is initialized with *N* areas, *N* being the number of authentication sites. In each area, a key (*GMi*) is written. These keys are not confidential for the security administrators. They only enable the initialization of the mothercards and of the users smartcards and they do not permit to access personalized cards nor to create false cards. Its use if however limited to people knowing the PIN and an access key.

The mothercards are initialized by the grand-mothercard by the creation of two areas (Fig. 2). The first area contains an access key (*GMi*) specific to the mothercard necessary to read and write in one area of the user's smartcard (see below). The second area contains nothing at this moment. In this state, everyone who knows *GMi* can write into the mothercards: it means essentially all the administrators without distinction. But, the mothercard at this moment is unable to do anything, registration or authentication.

Then each security administrator gets his card and changes the second area by protecting it with a new key (M_i) he is alone to know. He writes into the confidential part of this area the authentication key (*AKi*) and the session key generator (*SKi*). They are both not exportable outside the card. Another data item (*Ti*), used by the certification function, is stored in the transaction zone. In the confidential zone of the other area, he puts the access key (*Mi*) necessary to be diversified and to protect the users cards (see below). *Mi*, *AKi*, *SKi* and *Ti* are secret values, known only by the administrator of site *i*. The grand-mothercard can thus only create a wrong key, or can generate an incorrect user card that the administrator can easily detect during the registration phase.

At the end of this phase, each security administrator owns a smartcard he is alone to be able to use since he has got two secrets, the PIN and the access key (*Mi*)[2]. The card contains all the confidential items necessary to register a new user and to authenticate him later. The mothercard is ready to use and fully protected since even the grandmother-card cannot read or write in the second area.

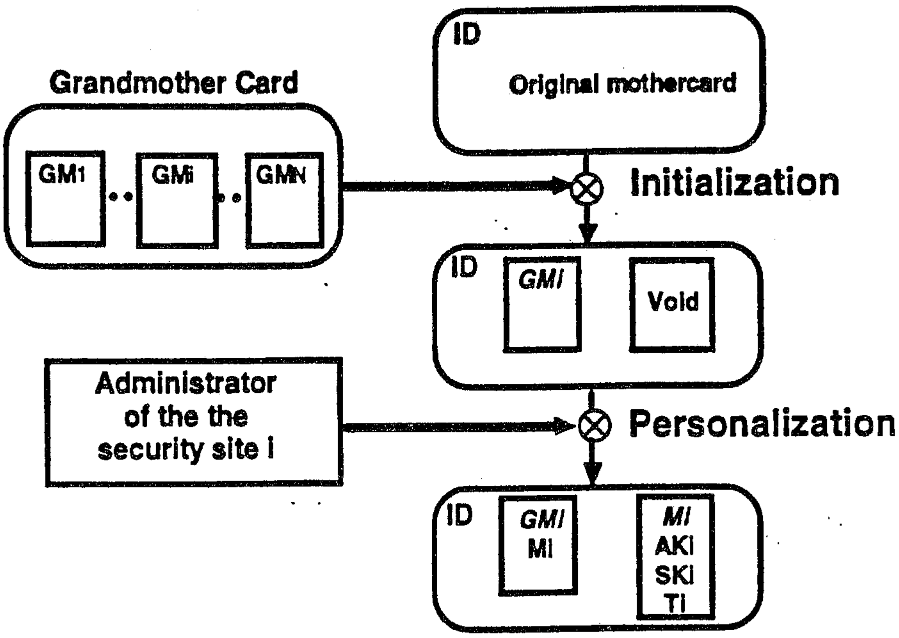

Figure 2: Personalization of a mothercard (the italics variables are access keys).

IV.1.2. Registration

The registration consists in two phases (Fig. 3):

- the initialization of the user's smartcards by the grand-mothercard with N areas, corresponding to the N mothercards: each area is initiated with the access key D(GM_i,ID), ID being the unique identity of the card (created during manufacturing),
- the actual registration performed by each of the N mothercards which store the secret keys in its own area: to do so, the mothercard must compute the same value D(GM_i,ID) (from its GM_i field) and use this key to access the area i , then it uses the same function to compute the diversified authentication (D(*AKi*,ID)) and generation keys (D(*SKi*,ID)) which will then be stored in the user's smartcard. Another secret value (*Ti*) which will be used by the certification function is stored in the transaction area. At last, it changes the access key of the area as D(M_i,ID).

[2] If several persons are to administrate the same authentication site, each one will be given a mothercard with the same values *Mi*, *AKi*, *SKi* and *Ti* but with a different PIN.

In this second step, an administrator could prevent a correct registration by writing in more than one areas. If the administrator of site *i* can get the grand-mothercard (and its PIN and access key), he can obtain the *GMi* of the other mothercards; and then he can initialize the other areas of the user card. However, such an intrusion would be detected easily as well as the responsible.

After this step, only the mothercard *i* is able to re-compute the values D(*Mi*, ID) D(*AKi*, ID) and D(*SKi*, ID) with the secret keys it owns and the ID of the card.

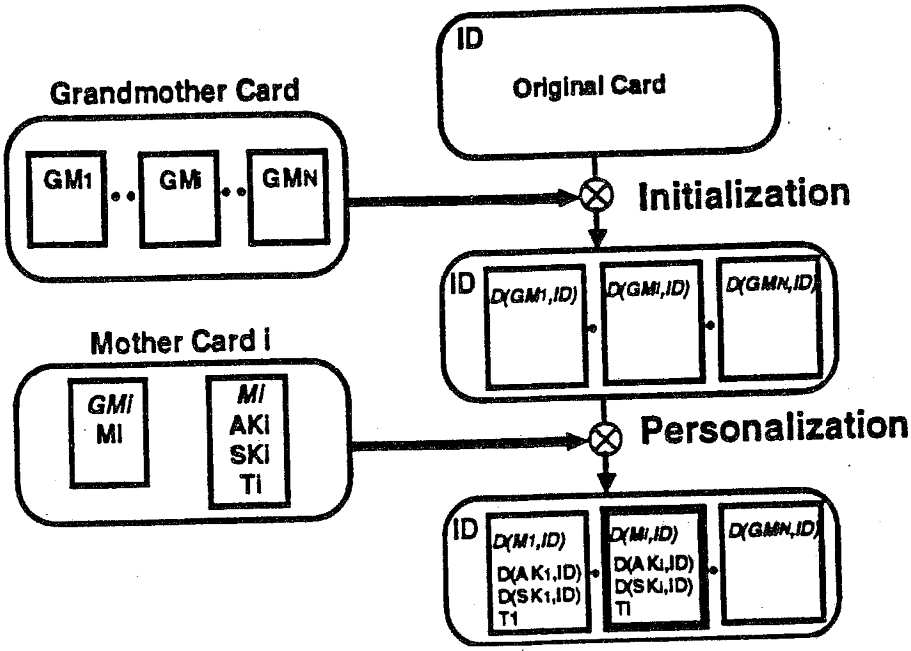

Figure 3: User registration.

This registration protocol permits to avoid that one card becomes a single point of failure:

- the grand-mothercard cannot access the personalized mothercards or the registered users cards since it does not know the *Mi* keys, nor it cannot create a false user's card which would not be identified false by the mothercards,
- a mothercard can only read and write the corresponding area on the user cards and cannot access the other areas protected by access keys it does not know: thus it cannot impersonate a user nor create a new one,
- the secret authentication key is split into *N* different parts (the *AKi*), nobody knowing the whole,
- a registered card cannot guess the secret keys of others since all depend on the unique ID of the card and the generators *AKi*, *SKi* it does not know,
- the availability of the system is ensured since if stolen or destroyed, a new card can be easily regenerated - a mothercard can be regenerated with the secret information (*Mi*, *AKi*, *SKi*) only known by the administrator and a user's card with the mothercards -,

Even if a mothercard was stolen as well as its PIN, then it would be only necessary to change on the user cards the contents of the corresponding area without any modification on the others, and the authentication service would still be available with *N*-1 authentication sites before the changes were completely done.

In order to make this recovery easy, it would be useful that the registration of a user's card could be done remotely. The current implementation implies that the user puts his card into the different smartcard readers connected to the authentication sites to be registered, but the CP8 smartcards functionality enables the remote registration.

To summarize the registration, we can say that, on the contrary of traditional approaches of authentication, it does not exist a person with enough privileges to prevent a smartcard's user to be registered or to impersonate an already registered user.

IV.1.3. Authentication

The global authentication consists in *N* different authentications between the user's smartcard and the mothercards which know the secret keys. Each mothercard can authenticate one and only one secret key in the corresponding area. At the end of this step, each mothercard has taken a decision. But before replying, the authentication sites agree on a global answer which is just a majority vote on the *N* different decisions (i.e., authentication accepted or rejected). After the agreement, all the correct authentication sites reply OK or not OK to the user site.

The detailed authentication protocol is as follows:

① the user site broadcasts an authentication request to the authentication sites containing the card's ID,

② each mothercard generates a random number *X* (a challenge) which is then sent to the user site with the address of the authentication key D(*AKi*,ID) within the confidential area,

③ each mothercard computes an authenticator, first by computing D(*KSi*,ID) then the certificate *Ci* = C(D(*AKi*,ID),*X*,*Ti*)

④ the user's card computes the *N* certificates *Ci'* = C(D(*AKi*,ID),*X*,*Ti*) then sends them to the authentication sites,

⑤ each mothercard compares the two certificates *Ci* and *Ci'* and sends the result to the authentication site: if they are identical the security site considers that the user's smartcard has been successfully authenticated,

⑥ each authentication site broadcasts its decision to others: the messages are signed with a private key, ⑦ each authentication site checks the signatures of the received messages with the public keys of the other sites, then it votes on all the decisions: if a majority accepts the authentication then the authentication is locally granted, else it is denied.

The independence of the authentication process and the splitting of the whole authentication value into N secret keys make the protocol fault and intrusion tolerant whatever happens on a minority of authentication sites:

- a legitimate user will be successfully authenticated by at least the majority of the authentication sites,
- an intruder, whoever he is, can be successfully authenticated by only a minority of faulty authentication sites.

The security of the protocol itself is based on the security of the certification function provided by the card and on the efficiency of the random number generator also provided by the card.

IV.1.4. Session key generation

The final step consists in generating the session keys which will be used to communicate securely between the user's site and the authentication sites. In our case, the authentication server acts also as an authorization server [Deswarte 91]. A session key is thus necessary to ensure both the confidentiality of private exchanges and the authenticity (by way of a signature) of the messages.

The session key generation consists in 3 phases:

① each authentication site, when it has agreed to authenticate the user, generates a random number Y and then sends it with the address of D(SKi,ID) in the area i of the user's card,

② each mothercard computes D(SKi,ID) then the certificate C(D(SKi,ID),Y,Ti) which will be used as a session key,

③ the user's card computes the N certificates C(D(SKi,ID),Y,Ti) and gets the N session keys identical to those generated on the authentication sites.

An authentication site (or its mothercard) cannot guess the value of the other sites' keys since they are generated from data known only by one mothercard. There is no exchange of information between them during this phase.

At the end of the protocol the user possesses N keys. Each authentication site possesses one key and has no information on the other keys. The user site can then use these keys to communicate with the different authentication sites. Each message if it is ciphered can be deciphered by only one authentication site which owns the key. And the user site is sure of the origin of a message signed with a session key.

These session keys have a limited lifetime necessary to ensure that an intruder who performs successfully an intrusion on the user site can only carry out a limited number of operations. When the lifetime is elapsed, the intruder cannot get a new session key or be authenticated again (except if he has got the user's smartcard and knows his PIN). The session keys are temporary because the user site does not tolerate faults or intrusions and is only protected by classical security means.

All these protocols have been implemented on a prototype which is running on Bull computers and workstations. The authentication server is composed of 5 sites, each one connected to a smartcard reader wherein the administrator mothercard is inserted. Each user workstation is itself connected to a smartcard reader which the user puts his card into. The authentication sites play as well the role of authorization sites controlling the access to an intrusion and fault tolerant file server [Deswarte 91].

The following table summarizes the security means to ensure the security of the authentication server with respect to intrusions, illegitimate operations or accidental faults.

	Intrusions and illegitimate operations		**Accidental faults**
	User site	*Authentication site*	*Authentication site*
OFF LINE (out of session)	Smartcards protections Multi-areas scheme PIN	Mothercards protections Splitting of secret keys	Multiple authentication sites
ON LINE (during a session)	Protection of the user site Temporary session key	Different session keys	Majority voting protocol

IV.2. Protocols for public keys

Public key protocols present several advantages over secret key schemes: the public key algorithms are generally stronger than DES and public keys can be replicated without compromising the secret keys. Moreover, the protocols are much easier to implement. Indeed, in the public key technique the component which authenticates the other cards has not to know the private keys. It knows only the corresponding public keys. The private key is stored only within the user's smartcard and is never exported out the smartcard.

However these cards present a main drawback which is how the key is generated. Indeed, these public key cards are usually based on an identity-based algorithms [Shamir 84] that we cannot use because such protocols use an external key generator which creates the keys from the card identity. This generator knows how to create false cards and how to guess the secret key of a given card. This generator is then the single point of failure we want to avoid.

Moreover, if a public key scheme presents some advantages over the shared secret key scheme, we have studied and chosen the latter for the implementation because at time the public keys cards were rare and difficult to use and to connect to our system.

Nevertheless, we have formally studied the authentication protocol to apply it to public key smartcards. In this case, one private key is sufficient. The main problem still remains to register the user securely. We cannot use identity-based cryptographic algorithms and thus we do need another system. The solution consists in requesting the user's card to initiate itself by creating a couple public key - private key and then to export the public key.

Moreover, the authentication sites do not need to own a particular smartcard[3]: the site itself is able to verify the identity of the user - i.e., the validity of his private key. The mothercards disappear in this scheme, only the grand-mothercard still remains to create the users cards. But in many cases this card exists only at the factory where the cards are made and has not to be taken into account in our system.

The registration process would be as follows:

- first, the initial user card generates both the public and the private keys: the private key is stored in a confidential area from where it cannot be exported, the public key is stored in a transaction area from where it can be read,
- second, each authentication site reads the public key and stores it in a database with the identity of the user (the ID of the card has not to be stored on the authentication site).

For each user, there is so only one public key which is replicated on all the sites and this information does not provide any way to impersonate a user, nor to create a false user. Moreover, the replication scheme permits to recover lost data much more easily than the secret key scheme. One needs only to copy data stored the others authentication sites and it is not necessary to register again the users.

[3] Public key smartcards can be used by the authentication sites to sign exchanges, in particular during agreement protocol.

The authentication and the session keys distribution protocols are then easy to define. We can use well-known algorithms such as [Guillou 88], [Fait 86], etc. We only need to add an agreement protocol between the authentication sites. The protocol looks like this:

① the user site broadcasts an authentication request to all the authentication sites,

② each authentication site using the public key establishes a protocol with the user site and authenticates or not the card,

③ each authentication site broadcasts its decision to others (the messages are signed with a private key),

④ each authentication site checks the signature of the received messages with the public keys it owns, then it votes on all the decisions: if a majority accepts the authentication then the authentication is locally granted, else it is denied,

⑤ each authentication site generates a random session key and sends it ciphered with the user's public key to the user site,

⑥ the user's card deciphers the message and gets *N* different session keys[4].

V. Conclusion

This paper describes a new approach to build an authentication server able to tolerate an arbitrary number of faults and intrusions, without any single point of failure. The server is made of several sites, each one managed by a different security administrator. Registration and authentication protocols were studied and implemented on Bull CP8 MP smartcards in order to guarantee the fault tolerance properties. The cryptographic functions of these cards are DES based. We split the secret authentication value between the different mothercards of the administrators. We have used the multi-area protection scheme of these cards to separate the privileges of the different administrators, and a three level hierarchy to ensure a full security and reliability at each step of the protocols. But it has also been shown how to implement the same server with public keys smartcards.

Acknowledgements: This work has been partially supported by the European ESPRIT programme, project n° 2252: Delta-4 (Definition and Design of an open Dependable Distributed architecture). The authors are grateful to Renaud Gandara and Jean-Michel Pons for their contribution to the implementation of our prototype, and Michel Dinghin of Bull-CP8 for facilitating the use of CP8 smartcards. The intrusion tolerance techniques have been defined and developed with the help of Jean-Claude Laprie, David Powell and Jean-Charles Fabre.

[4] These session keys are used because symmetric ciphers are faster than asymmetric ones.

References

[Blain 92]

L. Blain, "La tolérance aux fautes pour la gestion de la sécurité dans les systèmes répartis", *Thèse de doctorat de l'INPT*, LAAS Report n° 92 011, 152 p. (*in French*).

[Deswarte 91]

Y. Deswarte, L. Blain, J.-C. Fabre, "Intrusion Tolerance in Distributed Computing Systems", in *Proc. of Symposium on Research in Security and Privacy*, IEEE Computer Society Press, pp. 110-121, Oakland (CA), May 1991.

[Fiat 86]

A. Fiat, A. Shamir, "How to Prove Yourself: Practical Solutions of Identification and Signature Problems", in *Proc. of CRYPTO'86*, Santa Barbara (CA), August 1986, (A.M. Odlyzko, Eds), Springer-Verlag, Lecture Notes in Computer Science, Vol. 263, pp. 186-194, ISBN 3-540-18047-8.

[Guillou 88]

L.C. Guillou, J.J. Quisquater, "A Practical Zero-knowledge Protocol Fitted to Security Microprocessor Minimizing both Transmission and Memory", in *Proc. of EUROCRYPT'88*, Davos (Switzerland), May 1988, (C.G. Günther, Eds), Springer-Verlag, Lecture Notes in Computer Science, Vol. 330, pp. 123-128, ISBN 3-540-50251-1-3.

[Needham 78]

R.M. Needham, M.D. Schroeder, "Using encryption for authentication in large networks of computers", *Communications of the ACM*, vol.21, n°12 (December 1978), pp. 993-999.

[Powell 91]

Delta4 - A Generic Architecture for Dependable Distributed Computing, Springer-Verlag, (D. Powell, Ed.), 484 p., ISBN 3-540-54985-4, 1991.

[Shamir 84]

A. Shamir, "Identity-Based Cryptosystems and Signature Schemes", in *Proc. of CRYPTO 84*, Santa Barbara (CA), August 1984, (G. R. Blakley, D; Chaum, Eds.), Springer-Verlag, Lecture Notes in Computer Science, Vol. 196, pp. 47-53, ISBN 3-540-15685-5.

[Steiner 88]

J.G. Steiner, C. Neuman, J.J. Schiller, "Kerberos: an Authentication Service for Open Network Systems", in *Proc. of USENIX Winter Conference*, Dallas (Texas), February 9-12, 1988, pp. 191-202.