IMPLEMENTATION OF A SECURITY SERVICE PROVIDER FOR INTRANETS

Hamid M. Ali Wameedh N. Flayih

ABSTRACT

Among the many branches of security, authentication and confidentiality are very important to be provided. This work studies authentication focusing on the authentication systems supported by Windows 2000 family, especially Kerberos. As a result of this study, some unconvincing points are found along with others that are considered as weaknesses, such as being subject to offline dictionary attacks and the lack of perfect forward secrecy. Hence, some protocols (for authentication and key agreement) are chosen to build an authentication system that takes into consideration the observations on Windows 2000 systems. Based on this system, a security service provider is developed. The proposed provider isolates the developer from the complexity of the underlying system.

KEYWORDS

:

security, security service provider, authentication, Kerberos, dictionary attacks, certificates.

الخلاصة

يعتبر التصديق و الخصوصية من اهم فروع الامن التي يجب توفيرها. يقوم هذا البحث بدراسة التصديق مركزاً على انظمة التصديق المعتمدة في مجموعة انظمة التشغيل Windows 2000، وخصوصاً Kerberos . كنتيجة لهذه الدراسة وجدت بعض النقاط غير المقنعة و بعض مواطن الضعف، كالتعرض لoffline dictionary attacks و عدم توفير perfect forward secrecy . لذلك تم اختيار بعض البروتوكولات (للتصديق و الاتفاق على المفتاح) لبناء نظام تصديق ياخذ بنظر الاعتبار الملاحظات الماخوذة على الانظمة في Windows 2000 . و بالاعتماد على هذا النظام قد طور مزود خدمة امن. هذا المزود المقترح يعزل المطور عن تعقيدات النظام التحتي.

INTRODUCTION

Security is a must in today networks. For this reason, the software developers are facing a challenge in this field, because they must provide security for their applications, especially authentication and confidentiality. This is the role of security service providers (SSPs). They provide security services to the applications isolating them from the details. This leads to the fact that the security provided is dependent on the security protocols used by the SSP.

Passwords are the most common way to authenticate users. But they are considered weak keys; this means they cannot be used directly as encryption keys, because human users cannot remember long random numbers. This introduced the threat of dictionary attacks. An attacker carries ***offline dictionary attacks*** without direct interaction with the server. He eavesdrops and records messages exchanged between an authentic user and the server. These messages are constructed using functions that take the password as input. Therefore, he applies the same functions on the entries in his dictionary and compares the result with the recorded messages, until they match. To slow down this attack, it is desirable to make the functions relatively slow such that the search operation takes more time, but in such a way not to annoy the users [MOV96]. A possible solution is the use of Diffie-Hellman based algorithms that reveal no information about the password, such as the Authentication via Memorable Passwords (AMP) protocol [Kwo01].

Another type of offline dictionary attacks is that against stored password file. It is common to store verifiers (hashes) of the passwords in a file on the server and not the passwords themselves. This prevents the compromise of passwords in case the file is stolen. But if it is stolen then the attacker can do the same previous procedure, but against the whole list, and not a specific user record. To slow down this attack, the password is salted before being hashed. Each password is attached with a random string (salt) and then hashed, so the hash and the salt are stored for each user.

In the evaluation of a key establishment protocol, the effect of compromise of long term keys should be considered. This is what ***perfect forward secrecy*** describes. *Perfect forward secrecy* is provided by a protocol, if compromise of long term keys does not compromise past session keys [MOV96]. For this reason, it is preferred to be provided by a protocol. All the transport protocols do not provide *perfect forward secrecy*. Also, symmetric techniques cannot provide this feature. The key agreement protocols based on public-key techniques (Diffie-Hellman) achieve this property.

IMPORTANCE OF THE WORK

In this work the authentication systems available in Windows 2000 through the Security Support Provider Interface (SSPI) are studied, taking into consideration the two concepts previously presented: *offline dictionary attacks* and *perfect forward secrecy*. This gives some unconvincing or weak points that must be noticed.

Then a Security Service Provider (SSP) is proposed taking into consideration the noticed points on Windows 2000 authentication, withstanding offline dictionary attacks and providing perfect forward secrecy.

SECURITY SUPPORT PROVIDER INTERFACE (SSPI)

Microsoft has given software developers the ability to develop software that makes use of some cryptographic functions and the authentication systems available in Windows 2000. This can be achieved through the use of Cryptographic Application Programming Interface (CryproAPI) and the Security Support Provider Interface (SSPI).

The CryptoAPI provides the basic cryptographic functions (encryption, hash, and public key signature). So it is suitable if the developer needs little cryptography for his applications, but as the requirements are extended to authentication systems then it is not a good choice.

On the other hand, SSPI provides a mechanism by which a distributed application can call one of several security providers to obtain an authenticated connection without knowledge of the details of the security protocol. This facilitates the way developers build their applications on the security of Windows 2000. A security provider is a Dynamic Link Library (DLL) that implements the SSPI and makes one or more security packages available to applications. A security package maps the SSPI functions to an implementation of the security protocol specific to that package, such as New Technology Local Area Network Manager (NTLM), Kerberos, or Secure Socket Layer (SSL). Security packages are sometimes referred to as Security Support Providers (SSPs) [Mic99a]. The security strength provided by the SSPI is dependent on the three underlying authentication systems. As a result, a study on the authentication systems available in Windows 2000 is necessary.

WINDOWS 2000 AUTHENTICATION

Windows 2000 supports several protocols for verifying the identities of users, including protocols for authenticating dial-up connections and protocols for authenticating external users who access the network over the Internet. But there are only two choices for network authentication: Kerberos Version 5 and NTLM [Mic99b][TJ01].

SSL provides end-to-end encryption, integrity protection, and server authentication for the Web. It is also provided for applications through the SSPI. It makes use of certificates containing the public keys of the two parties. These certificates should be signed by a trusted Certificate Authority (CA). As a result, the next sections present NTLM, Kerberos, and certificates.

NTLM

The NTLM protocol was the default for network authentication in the Windows NT 4.0 operating system. It is retained in Windows 2000 for compatibility with down level clients and servers. NTLM is also used to authenticate logons to standalone (not domain members) computers with Windows 2000. It provides unilateral authentication. NTLM authentication was designed for a network environment in which servers were assumed to be genuine [Mic99b]. For this reason Kerberos v5 was used as the default system in Windows 2000.

Kerberos

Kerberos was developed at Massachusetts Institute of Technology (MIT) as a part of Project Athena in 1988 [MNS+87]. Symmetric cryptography and a trusted third-party are the basis of this authentication system. It is considered an authentication and key transport protocol. There have been two versions of the protocol in public use, namely Kerberos v4 and v5. The latter was chosen by Microsoft to be the default authentication system in Windows 2000 family [Mic99b]. The following points are noticed:

- It is subject to the offline dictionary attack [Wu99][KT03][BM91]. Since the client A sends the timestamp encrypted using the password hash, and the Authentication Server (AS) sends in the second message the timestamp encrypted using the A’s password hash. This means that an eavesdropper which captures these messages can try to decrypt them using a dictionary of common passwords.

- The AS does not store a clear-text copy of the passwords. Instead, the password is encrypted using RSA MD4 algorithm, which takes a variable-length password and encrypts it using a secret key to produce a fixed-length result called the *message digest*. The result is the hash of the password using a secret key. Domestic versions of Windows 2000 use a 128-bit message digest, considered unbreakable by anyone but the National Security Administration (NSA), and only then with a great deal of expensive supercomputer time. Export versions of Windows 2000 are limited to a 40-bit message digest, considered easily crackable [Bos00].

- The Kerberos protocol does not provide *perfect forward secrecy*. This means that if long term secret of the client (password) is compromised then all session keys used by the client are compromised. This introduces a security risk, but depending on the requirements of the system it may be serious or not.

- The session key is controlled by the AS; this means that not only the two involved parties know the secret key, but the AS too. It is preferable that the secret be known to only the involved parties.

- An overhead introduced is that the client A needs to get a different ticket for each different service (application server). This requires from the client to contact the AS to get a new ticket. This also means an overhead in the storage of secret keys. A needs to store K_{AB}, K_{AC}, and K_{AD}, which represent the keys corresponding to tickets used to contact B, C, and D.

- In Windows 2000, Kerberos is the authentication system, but NTLM is used to authenticate previous versions of Windows. So, this weakens the security, because it is not totally based on Kerberos. To prevent this, all clients must be upgraded to Windows 2000, and then NTLM should be prevented.

CERTIFICATES

A certificate is a digital document (i.e. a formatted file) that binds a public key to its owner. A trusted Certificate Authority (CA) creates the certificate and digitally signs it using the CA private key. Using the CA public key, applications verify the issuing CA digital signature, and hence, the integrity of the contents of the certificate (most importantly, the public key and the identity of the owner).

Microsoft Certificate Services, included with Windows 2000 server family, provide a means for an enterprise to easily establish CAs. Windows 2000 supports the installation of stand-alone CAs or enterprise CAs.

Stand-alone certification authority does not require the use of Active Directory. By default, all certificate requests sent to the stand-alone CA are set to *pending* until the administrator of the stand-alone CA verifies the identity of the requester and Okays the request. This means there is no automated procedure in certificate granting.

Enterprise certification authority requires the use of Active Directory, which means there is a need to implement a domain. Here the requests are automatically serviced, and according to the user information in Active Directory, the certificate is issued or not [Mic00][TJ01].

The following points are noticed:

- In Windows 2000, the case of stand-alone CA is not practical because it puts the decision on the administrator. For this reason, the enterprise CA is considered the best choice. But this type will depend on the user information in the Active Directory. This means that when a user requests a certificate, then he is authenticated to the CA using the network authentication system available, which may be NTLM or Kerberos [Mic00][TJ01].

- Certificate Revocation List (CRL) may add overhead in the communication and management, and the need to protect the list from unauthorized change.

- Another very important point is that the certificate contains the public key, but the private key must be kept secret to the owner. So, in case the owner is a PC, there is no problem. But if the owner is a user, who needs to get access from different PCs, then he needs to store the private key on each PC, which is a security problem.

- Windows 2000 stores the private keys of users using the Cryptographic Service Providers (CSPs). This means the keys are stored in the PC at which the user requested the certificate.

So, for PCs that has several users, the CSP stores the keys according to the identity of the user. Each user is given access to keys corresponding to his identity. This makes us conclude that CSPs depend on the (username, password) pair to protect the keys. This gives (in theory) the attacker a chance to launch dictionary attack against the stored keys, assuming the attacker has access to the location where the keys are secured and knows how they are secured. In other words, dictionary attacks are possible if the only thing missing is the password. This may compose a serious threat because the keys of certificates are kept for long time, which gives the attacker the required time.

PROPOSED AUTHENTICATION SYSTEM

The proposed system will be based on a central authentication server (AS) hosting the clients' information. The system can be viewed to have two different participants: the authentication server (AS) and the clients (including application servers). Each client should contact the AS to authenticate itself, and the AS will grant it a ticket (certificate) that proves the client's identity, and binds it to the public key in the ticket. When a client wants to contact another client (usually an application server), the tickets are exchanged and the two parties agree on a session key. Some important objectives are that the system should be secure against offline dictionary attacks and stolen verifier file, and should provide perfect forward secrecy.

The authentication protocol used when the client contacts the AS is based on the AMP protocol. This will achieve security against dictionary attacks and stolen verifier file. The handshake protocol between two parties is based on Diffie-Hellman, providing perfect forward secrecy. RSA is used for digitally sign the tickets. Advanced Encryption Standard (AES) encryption is used as the symmetrical encryption algorithm. SHA-1 is the hash algorithm used. The ticket lifetime should be in hours or minutes.

SYSTEM PROTOCOLS

The details of the authentication protocol and ticket renewal protocol are presented next. Some important parameters used in the protocols are:

(Any arithmetic operation not specifying the modulus, means 'mod p')

π: password
hw: hardware id of the PC (for example: LAN card physical address or hard disk serial number).
v= h(userid, π)
V: password verifier = g^v
x , g^x: private/public key pair of the PC.
y , g^y: private/public key pair of the AS.
V_{AS}: digital signature verification key of the AS.
p , q: large prime numbers; as in AMP protocol.

The AS stores PC information (pcid, g^x, g^{hw}). It also stores the password verifiers in a file; each record is encrypted symmetrically using a secret key ***S*** and a salt $\boldsymbol{\tau}$. So, each record in the file is of the form :(userid, τ, $E_{S+\tau}$(V)). This makes the file secure against offline dictionary attacks when stolen.

A. AUTHENTICATION PROTOCOL

Fig. 1 shows the authentication protocol, assuming the user and PC are registered on the AS. The client A will be authenticated through H_1, while the AS will be authenticated to the client through G_3. At the end of the protocol the AS sends its digital signature verification key (V_{AS})

and its public exponential (g^y). Other values are also sent, such as the session duration and the session parameters (α and *n*). The public exponential will be used by clients later to communicate with the AS. g^{x1} will be used as the public key in the ticket.

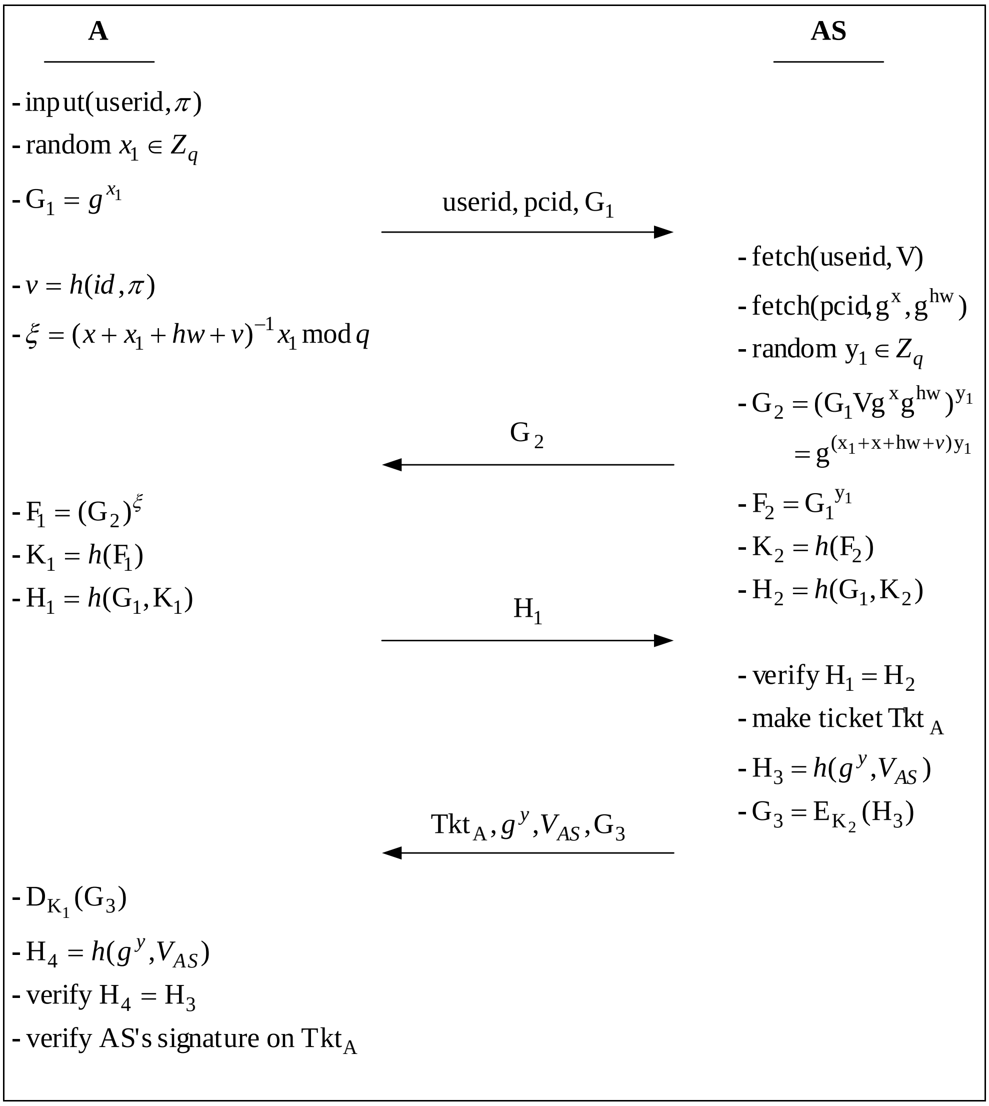

A
AS
$-\text{input}(\text{userid}, \pi)$
$-\text{random } x_1 \in Z_q$
$-G_1 = g^{x_1}$
$\text{userid, pcid, } G_1$
$-\text{fetch}(\text{userid}, V)$
$-\text{fetch}(\text{pcid}, g^x, g^{hw})$
$-v = h(id, \pi)$
$-\xi = (x + x_1 + hw + v)^{-1} x_1 \bmod q$
$-\text{random } y_1 \in Z_q$
$-G_2 = (G_1 V g^x g^{hw})^{y_1} = g^{(x_1 + x + hw + v) y_1}$
G_2
$-F_1 = (G_2)^{\xi}$
$-K_1 = h(F_1)$
$-H_1 = h(G_1, K_1)$
$-F_2 = G_1^{y_1}$
$-K_2 = h(F_2)$
$-H_2 = h(G_1, K_2)$
H_1
$-\text{verify } H_1 = H_2$
$-\text{make ticket Tkt}_A$
$-H_3 = h(g^y, V_{AS})$
$-G_3 = E_{K_2}(H_3)$
$\text{Tkt}_A, g^y, V_{AS}, G_3$
$-D_{K_1}(G_3)$
$-H_4 = h(g^y, V_{AS})$
$-\text{verify } H_4 = H_3$
$-\text{verify AS's signature on Tkt}_A$

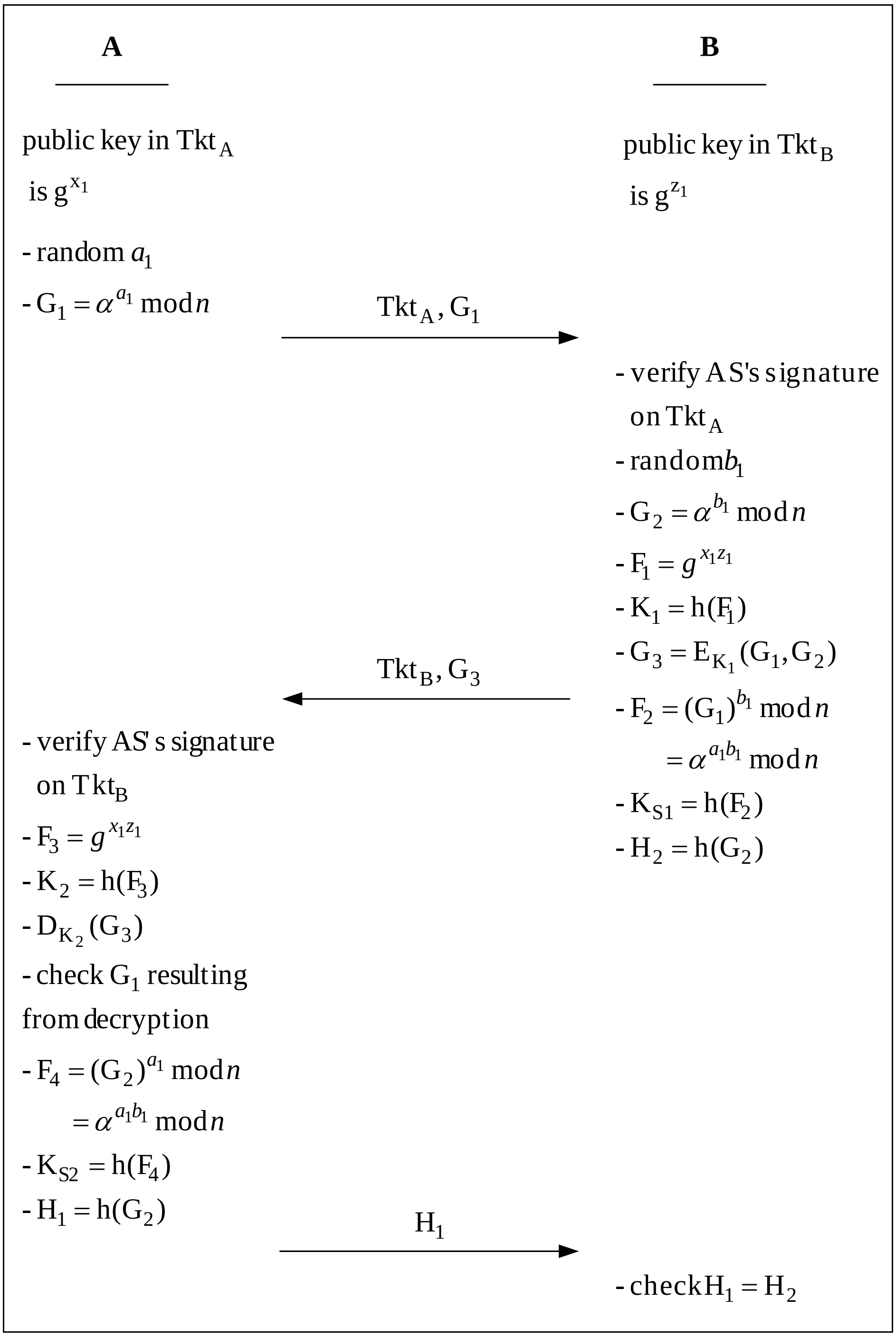
Fig. 1: Authentication Protocol

B. KEY AGREEMENT PROTOCOL

After the clients have been granted the tickets at the authentication phase, they can now communicate with each other. To communicate securely they must agree on a session key. The public exponentials included in the tickets are used to carry out a Diffie-Hellman key agreement. So, each client verifies the signature of the AS on the ticket of the other, and its validity. The details are shown in Fig. 2.

When A receives G_3 in step 2 it decrypts it using K_1 and checks α^{a1}, if it is correct then A is sure that the other party is really B. At step 3 A sends the hash of B's exponential (G_2). B checks it, and if it is correct then it is sure of the identity of A, because A is the only one that knows K_1 and so can decrypt G_3.

It is known that Diffie-Hellman states that: knowing the public exponentials only; it is hard to find the key α^{a1b1}. But here the case is more difficult, because α^{b1} is not public. It is encrypted using K_1. Hence, the attacker must solve two such problems: one to find K_1 and then to find the session key K_2. For this reason the size of the number (n) can be reduced, this can improve the performance. So, 768 bits can be used in stead of 1024 bits used for (p).



A

public key in Tkt_A is g^{x_1}

- random a_1
- $G_1 = \alpha^{a_1} \bmod n$

Tkt_A, G_1 →

B

public key in Tkt_B is g^{z_1}

- verify A S's signature on Tkt_A
- random b_1
- $G_2 = \alpha^{b_1} \bmod n$
- $F_1 = g^{x_1 z_1}$
- $K_1 = h(F_1)$
- $G_3 = E_{K_1}(G_1, G_2)$
- $F_2 = (G_1)^{b_1} \bmod n = \alpha^{a_1 b_1} \bmod n$
- $K_{S1} = h(F_2)$
- $H_2 = h(G_2)$

← Tkt_B, G_3

- verify AS' s signature on Tkt_B
- $F_3 = g^{x_1 z_1}$
- $K_2 = h(F_3)$
- $D_{K_2}(G_3)$
- check G_1 resulting from decryption
- $F_4 = (G_2)^{a_1} \bmod n = \alpha^{a_1 b_1} \bmod n$
- $K_{S2} = h(F_4)$
- $H_1 = h(G_2)$

H_1 →

- check $H_1 = H_2$

Fig. 2: Key Agreement Protocol

System Architecture

The system may be viewed to be composed of two parts: the Authentication Server (AS) and the Client’s Security Service Provider (CSSP) as shown in Figure 3. The blocks representing the applications have discontinuous lines. This is to indicate that they are not part of the system, but the system will be used to provide services to these applications.

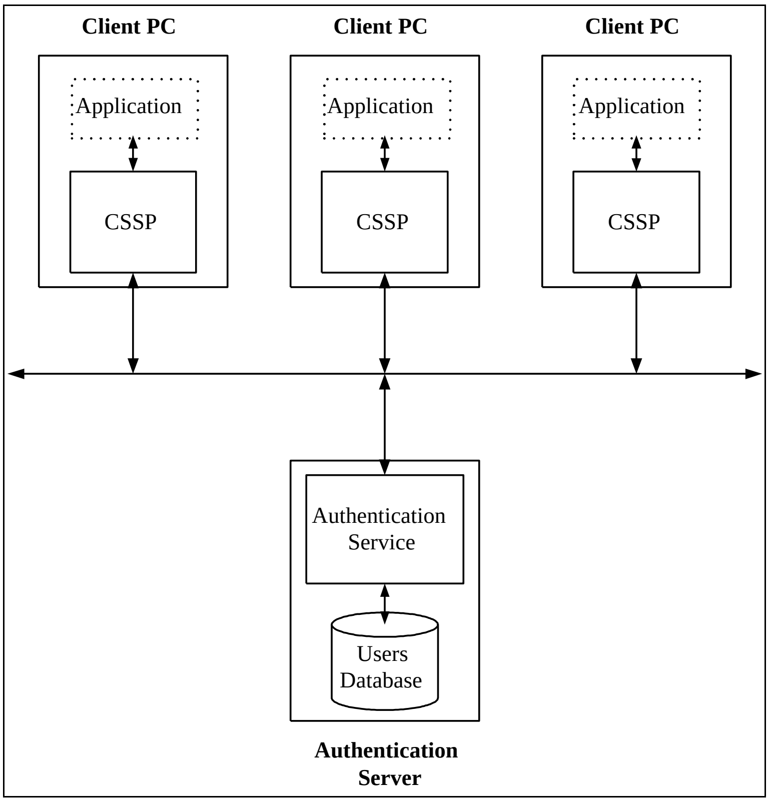


Fig. 3: System Components

A. Authentication Server Architecture

It is the part of the system hosting the tables containing information about the users, PCs, groups, and tickets granted. The AS is responsible for the authentication of clients, granting tickets, renewal of tickets, and consequently the administration of the system occurs here. The architecture is shown in Fig. 4.

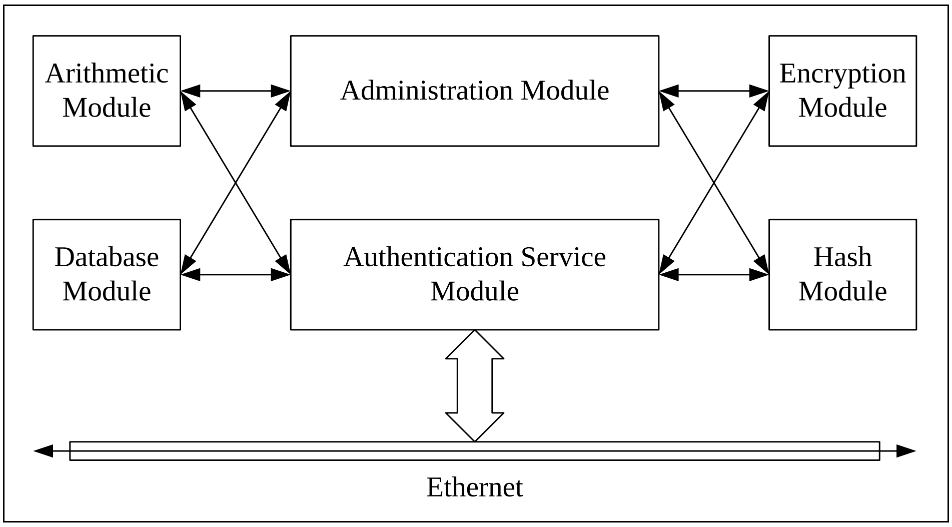


Fig. 4: The AS Architecture

B. Client's Security Service Provider

It represents the part of the system residing at the client PCs. It is considered as a gateway for the application built over it. Through it the application connects to the AS and to other instances of the applications on other PCs. The CSSP is not involved in the details of the information sent between the applications. It just provides the secure communication channel between them. The CSSP is involved in the authentication, key agreement, and ticket renewal protocols. These protocols require several functionalities, and according to them the architecture of the CSSP will be build, as shown in Fig. 5.

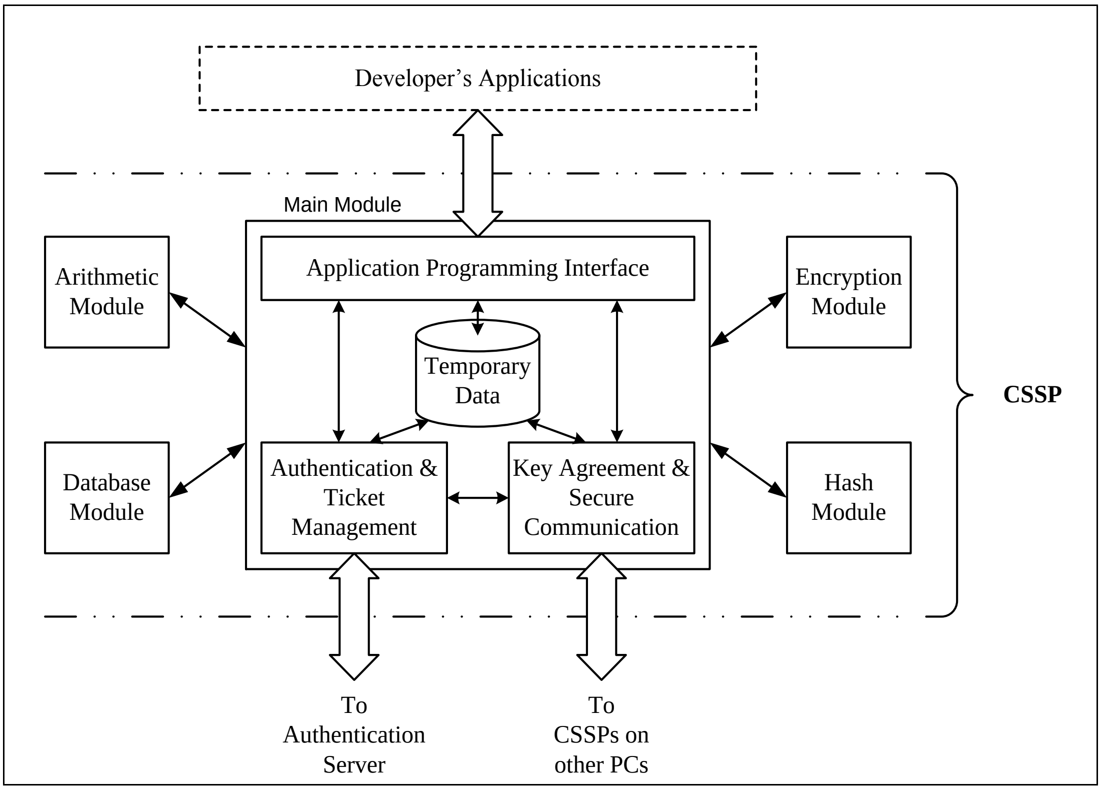


Fig. 5: CSSP Architecture

1. ***Application Programming Interface (API):*** it is the set of functions that the developer must deal with to link his application with the system. This part isolates the details of the system, providing the developer simple functions to interact with.
2. ***Authentication and Ticket Management:*** it includes the authentication protocol and ticket renewal protocol. This part communicates with the AS. So, it contains functions that provide network connectivity to the AS.
3. ***Key Agreement and Secure Communication:*** it communicates with other CSSPs (specifically with the Key Agreement and Secure Communication part). This part includes the key agreement protocol and provides the secure transfer of data between applications.

The steps carried out to establish a secure communication channel between two applications App_{1A} and App_{1B} are clarified through Fig. 6. App_{1B} acts as an application server that listens for client requests, and App_{1A} acts as a client to this application server.

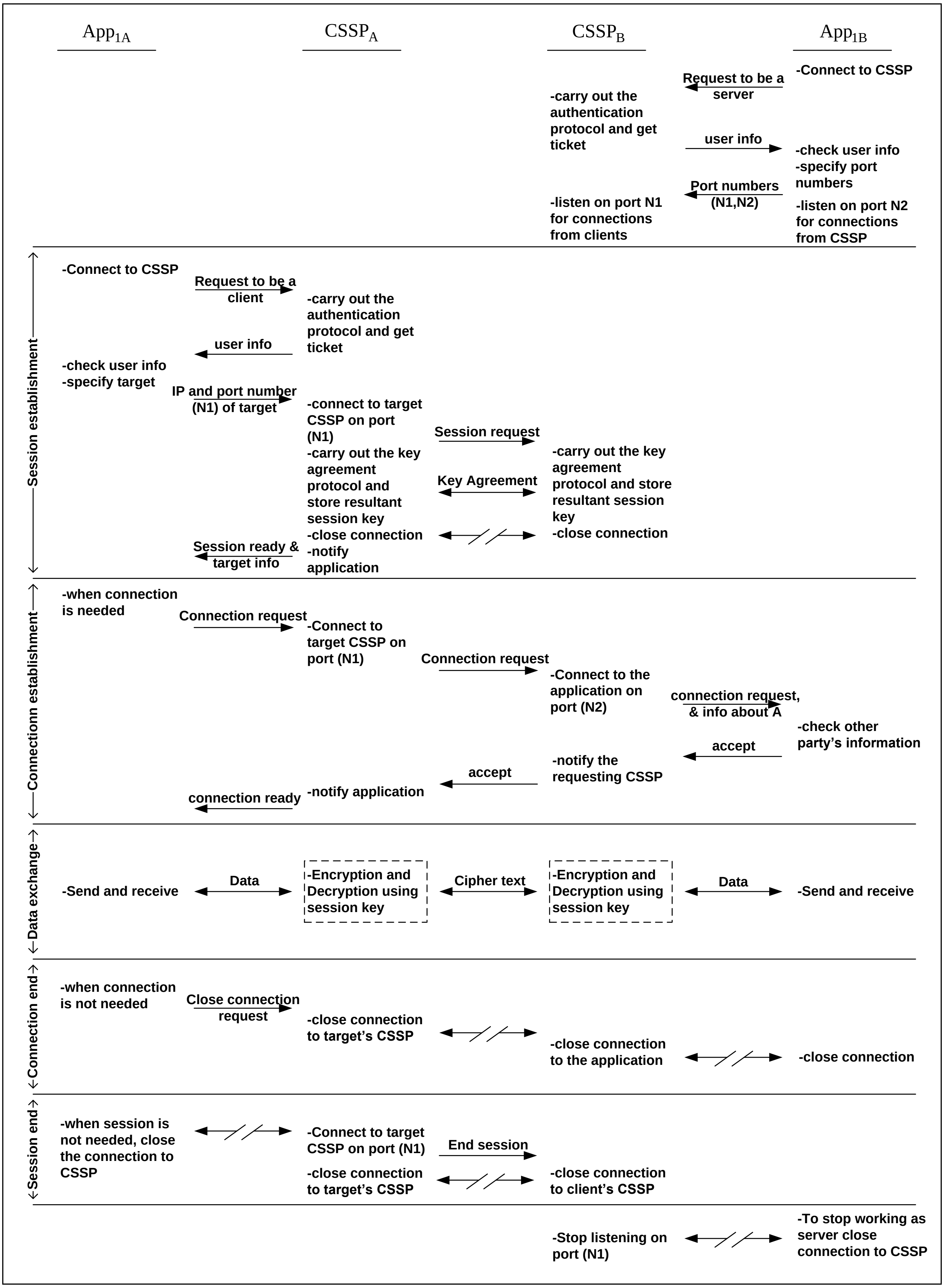

App1A
CSSPA
CSSPB
App1B
-Connect to CSSP
Request to be a server
-carry out the authentication protocol and get ticket
user info
-check user info
-specify port numbers
Port numbers (N1,N2)
-listen on port N1 for connections from clients
-listen on port N2 for connections from CSSP
Session establishment
-Connect to CSSP
Request to be a client
-carry out the authentication protocol and get ticket
user info
-check user info
-specify target
IP and port number (N1) of target
-connect to target CSSP on port (N1)
Session request
-carry out the key agreement protocol and store resultant session key
Key Agreement
-carry out the key agreement protocol and store resultant session key
-close connection
-notify application
-close connection
Session ready & target info
Connectionn establishment
-when connection is needed
Connection request
-Connect to target CSSP on port (N1)
Connection request
-Connect to the application on port (N2)
connection request, & info about A
-check other party's information
accept
-notify the requesting CSSP
accept
-notify application
connection ready
Data exchange
-Send and receive
Data
-Encryption and Decryption using session key
Cipher text
-Encryption and Decryption using session key
Data
-Send and receive
Connection end
-when connection is not needed
Close connection request
-close connection to target's CSSP
-close connection to the application
-close connection
Session end
-when session is not needed, close the connection to CSSP
-Connect to target CSSP on port (N1)
End session
-close connection to target's CSSP
-close connection to client's CSSP
-Stop listening on port (N1)
-To stop working as server close connection to CSSP

Fig. 6: CSSP Operation

IMPLEMENTATION AND PERFORMANCE

Visual C++ was the programming language used. Each module is a separate library (DLL) of functions. However, the parts of the main module of the CSSP are all integrated in one library (one DLL). The separation of the modules allows the modification of each module without affecting the others. The RSA operations are implemented using the CryptoAPI library. The connections between the applications and the CSSP will be using sockets programming (based on the *CAsyncSocket* class). The system works on Windows 2000 family and next versions. It is preferred to use NT File System (NTFS) to give access control to the files.

The system was tested on Pentium III with Windows 2000 Server operating system and 100 Mbps Fast Ethernet PCI Adapter. The authentication protocol starts and ends at the client side. Its average time has been found to be (**131.2 msecs**). This protocol starts at the CSSP acting as client, and ends at the other side (CSSP acting as server). Thus, the time will be taken for each side apart (i.e. the time each side participates in the protocol). The average time for the client side was (**91.6 msecs**) while for the server side was (**78.5 msecs**). Also, the effect of the protocols on the processor and network traffic was monitored using Windows 2000 Performance Monitor, and there was no high load.

CONCLUSIONS AND FUTURE WORK

The most important point is that Kerberos is subject to *offline dictionary attacks* and it does not provide *perfect forward secrecy* which may be considered important points, in addition to other notes. Also, it was shown that SSL supported by the SSPI depends on certificates, which are granted to clients through the network authentication system (Kerberos). This means that they suffer from the same weaknesses. This means that the SSPI provides a good level of security through Kerberos and SSL if their weaknesses are not considered serious. The seriousness of the weaknesses depends on the network environment in which the system is implemented.

In case the level of security provided by the SSPI is not considered enough, then the proposed protocols presented in section 5.1 solve the unconvincing points and weaknesses in Kerberos. This was done using the AMP authentication protocol and the Diffie-Hellamn key exchange. But this gave some disadvantages, because the proposed authentication system is slower than Kerberos because the latter is based on symmetric encryption techniques, while the AMP protocol is based on public key. Also, it adds overhead as compared to SSL, because the proposed system requires online trusted third part, while SSL requires offline trusted third party.

A very important point to be stated is that the proposed provider was implemented above the transport layer. This provides implementation flexibility and ensures that only the applications needing security use the system. But a disadvantage is that these applications must be modified (i.e. transparency is not provided). Transparency is provided if the security layer is implemented at the lower layers, network or data link layers, but at the cost of additional overhead and implementation complexity. The overhead results because not all the applications require the provided security.

A suggestion for the future is to extend the design to provide cross domain trust, such that a client under the control of AS1 can contact another client under the control of AS2 in case there is a trust relationship between AS1 and AS2. Also, it is possible to integrate the system with the SSPI, taking the advantages and disadvantages into consideration.

REFERENCES

[BM91] S. M. Bellovin and M. Merritt, *"Limitations of the Kerberos Authentication System"*, Proceedings of the Winter 1991 Usenix Conference, pp. 253-267, Dallas, January, 1991.

[Bos00] W. Boswell, *"Inside Windows 2000 Server"*, New Riders, 2000.

[KT03] K. Kasslin and A. Tikkanen, *"Attacks on Kerberos V in a Windows 2000 Environment"*, Research project for Helsinki University of Technology, 2003.

[Kwo01] T. Kwon, "*Authentication and key agreement via memorable passwords"*, Proceedings Network and Distributed System Security Symposium, San Diego, California, February 7-9, 2001.

[MOV96] A. Menezes, P. van Oorschot, and S. Vanstone, *"Handbook of Applied Cryptography"*, CRC Press, 1996.

[Mic99a] Microsoft Corporation, *"The Security Support Provider Interface"*, Windows 2000 White Paper, March, 1999.

[Mic99b] Microsoft Corporation, *"Windows 2000 Kerberos Authentication"*, Windows 2000 White Paper, 1999.

[Mic00] Microsoft Corporation, *"Windows 2000 Certificate Services"*, Windows 2000 White Paper, 2000.

[MNS+87] S. Miller, C. Neuman, J. Schiller, and J. Saltzer, *"Kerberos Authentication and Authorization System"*, M.I.T. Project Athena, Cambridge, Massachusetts, December 21, 1987.

[TJ01] C. Todd and N. L. Johnson, *"Hack Proofing Windows 2000 Server"*, Syngress Publishing, 2001.

[Wu99] T. Wu, *"A Real-World Analysis of Kerberos Password Security"*, Proceedings of the 1999 Internet Society Network and Distributed System Security Symposium, San Diego, CA, February, 1999.

ABREVIATIONS

AES	Advanced Encryption Standard
AMP	Authentication via Memorable Passwords
API	Application Programming Interface
AS	Authentication Server
CA	Certificate Authority
CRL	Certificate Revocation List
CryptoAPI	Cryptographic Application Programming Interface
CSP	Cryptographic Service Provider
CSSP	Client's Security Service Provider
DLL	Dynamic Link Library
MIT	Massachusetts Institute of Technology
NSA	National Security Administration

NT	New Technology
NTFS	NT File System
NTLM	New Technology Local Area Network Manager
SSL	Secure Socket Layer
SSP	Security Service Provider
SSPI	Security Support Provider Interface