A Secure Authentication System for Distributed Computing Environment Based on PKI Biometric Verification and Kerberos

Chih-Chen Yen
Department of Electronic Engineering
National Tsing Hua University
HsinChu,Taiwan 300, R.O.C.
yan@alumni.ee.nthu.edu.tw

Wen-Hsing Hsu
Department of Electronic Engineering
National Tsing Hua University
HsingChu, Taiwan 300, R.O.C.
Institute of Information Science
Academia Sinica, Taipei, Taiwan 115, R.O.C.
whhsu@pc05.ee.nthu.edu.tw

Abstract

Today, the most widely-used authentication protocol is Kerberos. But Kerberos has potential weaknesses which result from its password based architecture, purely symmetric cryptography, and the assumption that client can securely protect users' verification documents. In this paper, we propose a secure authentication system based on Kerberos, biometric verification, and public-key technology. Our system is expected to achieve five goals: user convenience, storage security, robust authentication, administrator management, and attack resistance.

Keyword— Kerberos, Biometric Verification, Smart Card, Public Key Infrastructure (PKI), Distribution System

1. Introduction

Authentication is a mechanism for one of the communicating parties to verify the identities of the other(s). Currently, most authentication systems are based on password verification, such as Kerberos [1], SESAME [2]. These systems suffer from password-guessing attack. Most users prefer short and meaningful word sequences rather than long and random passwords. This leaves hackers an opportunity to crack users' passwords.

Another kind of authentication system is based on biometric verification. Biometric verification benefits from the uniqueness and the mobility of biometric features. Besides, biometric features are usually longer and more random than password. Isobe *et al.* [4] proposed an authentication protocol based on biometric verification. Their system relies on the verification on smart card. However, smart card is a terminal device. Administrators may have a suspicion about the impartialness of verification on smart card.

Except for security issue, efficiency is also important. Symmetric cryptosystems can achieve good performance. But it has weaknesses that result form inevitable shared keys. On the other hand, asymmetric cryptosystems are securer but are more computing complex.

In our system, a server is responsible for biometric verification and later log-in procedures. The result of central verification is more impartial than that proposed by Isobe *et al.* Besides, we adopt both symmetric and asymmetric cryptography [5] to achieve a balance between security and efficiency.

This paper is organized as follows. In Section 1, we give an overview of authentication systems. In Section 2, we review two widely-used authentication systems and list their features and disadvantages. While in Section 3, we introduce the biometric verification technology used in our system. Then, we propose our authentication system in Section 4. In the following Section 5, we analyze the security of our system. Finally, we give the conclusions in Section 6.

2. Authentication System

2.1. Kerberos [1]

2.1.1. Authentication Flowchart of Kerberos

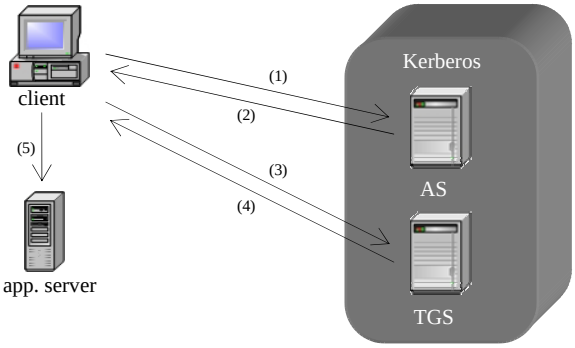


Figure 1. Kerberos authentication flowchart

Kerberos is developed by Massachusetts Institute of Technology (MIT) in middle 1980 and it is based on the early architecture proposed by Needham and Schroeder [6].

In Kerberos, a trusted third-party named *Authentication Server* (AS) is responsible for the verification of one's identity. In order to detect replay attack [7], Kerberos adopts timestamp mechanism. Before providing services, one must store his/her password in both client machine(s) and AS (apply symmetric cryptography).

In Figure 1, we illustrate the authentication flow of Kerberos. First, one has to request AS for an identity credential. Once he/she receives the respondent document, he/she can request *Ticket-Granting Server* (TGS) for a ticket to application server.

2.1.2. Kerberos Analysis

In Kerberos, user can obtain an identity credential after his/her verification in AS. This credential can be used to request TGS for tickets to multi-application servers. Therefore, Kerberos supports the functionality of "single sign-on". However, Kerberos exists some problems [8, 9, 10]:

1. System assumption: Kerberos is one part of MIT's Athena project. The designers of Kerberos assumed that all terminal machines are secure. In a word, Kerberos was designed to provide secure context over insecure network. The secure storage of authentication data is left to users. However, this assumption is not suited to current network environment due to the rampancy of attackers. Therefore, in the proposed system, we consider the security in both terminal-ends and transaction messages.
2. Adopt symmetric cryptography: In Kerberos, AS must maintain a password table for the authentications of users. Cracking AS will not only impact administrator(s) but all users. Another choice is asymmetric cryptosystem, such as SESAME in the next subsection. But this kind of authentication systems suffers from their poor performance. Therefore, we adopt a mixed cryptography [5] in our system (the detail is available in Section 4).
3. Password based authentication: As what we have mentioned above, password based authentication is vulnerable against to dictionary attack. Although some password selection strategies [11] can be used in users' registers, password based authentication always has contradictory between security and convenience. In our system, users' biometric features rather than password are used to provide an evidence of one's identity. We will discuss the benefits of biometric verification in the later sections.

2.2. SESAME [2, 3]

2.2.1. Authentication Flowchart of SESAME

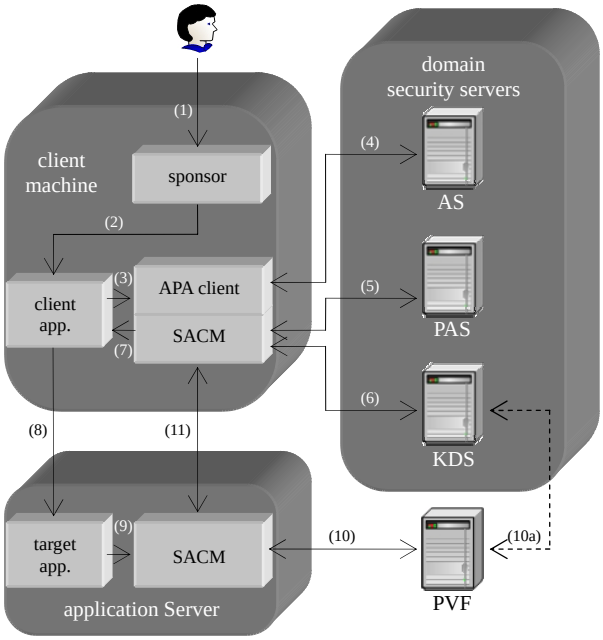


Figure 2. SESAME authentication flowchart

SESAME is an authentication scheme extends Kerberos. Its development is mainly driven by European computer manufacturers (BULL, ICL, and Siemens). SESAME was expected to implement not only a secure authentication protocol but also access control architecture. The default cryptosystem of SESAME is asymmetric cryptosystem.

In SESAME, there exist three kinds of servers for authentication. The *Authentication server* (AS) is responsible for the verification of user's identity.

The *Privilege Attribute Server* (PAS) is in charge of the issuance of *Privilege Attribute Certificate* (PAC), which is used in later access control. The *Key Distribution Server* (KDS) takes care of the management of tickets to application servers. Moreover, the *PAC Validation Factory* (PVF) exists for the validation of received PAC(s) and the establishment of basic keys, which is used to construct secure context between the communicating *Secure Association Context Managements* (SACMs).

Now, we illustrate the authentication flow of SESAME in Figure 2 and the following walk-through:

(1) User logins the client machine. A *Sponsor* will be the agent of current user after his/her log-in.
(2) User requests the remote service by invoking a specific client application.
(3) *Authentication and Privilege Attribute Client* (APA Client) is responsible for the request of credential. It is implemented as a library routine for hiding from Sponsor the details of accesses to AS and PAS.
(4) APA Client requests AS for authentication by using either Kerberos-formed or asymmetric cryptographic authentication message. After examining, AS returns a credential and a PAS ticket with the corresponding format.
(5) APA Client caches PAS ticket and requests PAS for a PAC by invoking SACM. A PAC with KDS ticket will be returned after PAS's check.
(6) Now, the module of SACM requests KDS for the key information, which is used to generate a basic key.
(7) The encrypted documents and the key information are sealed in GSS token. Then the GSS token is sent to client application.
(8) The client application sends GSS token to the target application.
(9) The target application bypasses the GSS token to the SACM of the application server.
(10) The SACM of target application server extracts and passes security information to the PVF. If the examination in the PVF[1] is positive, the validation result, PAC, and the basic key will be returned to the SACM of target application server the SACM caches the basic key for later communication with the SACM in the client-end.
(11) An optional GSS token is sent to the SACM of the client machine if mutual authentication is requested.

[1] The step of (10a) is necessary if a cross-domain access is met. In the condition, the received documents must be decrypted by an inter-domain key in the local KDS.

2.2.2. SESAME Analysis

The advantages of SESAME include: (1) support single sign-on, authentication, mutual authentication, *Role Base Access Control* (RBAC), and delegation of user's privilege, (2) administrators can develop their authentication system easily due to the modulo-components of SESAME [13], and (3) the designers of SESAME well consider the situation of cross-domain operations [13] and try to minimize the steps of inter-domain transaction [14]. However, the default cryptosystem of SESAME is asymmetric algorithms, so SESAME has poorer performance than Kerberos.

As Kerberos, SESAME relies on password verification. As a result, it is also weak against password-guessing attack. And client-storage is vulnerable to Trojan horse. Besides, the accesses of a user are limited to some specific machines.

3. Biometric Verification

Biometric authentication is based on two characteristics of biometric features: **uniqueness** and **mobility**. Nobody is completely equal to the others. Therefore, biometric features provide better proof of our participation than password verification.

Among all methods, we believe that fingerprint verification is currently the optimal method because the following discussion. When comparing with the other approaches, we find that fingerprint verification, hand geometry verification, and retina verification have relative lower *False Acceptation Rate* (FAR) and *False Rejection Rate* (FRR) [16]. However, fingerprint capturer is cheaper and requires lesser memory to store one's fingerprint features. Besides, retina scanner may hurt eye balls. Therefore, we choose fingerprint verification to construct our authentication system.

In the research of Isobe *et al.* [15], there are three storage tactics of one's verification template. When examining the three categories, we find that a portable device is more suitable to store one's verification data. Besides, user will not suffer from the Trojan horse seeded in the client machine. In our system, we adopt the smart card to securely store one's fingerprint template.

4. The Proposed System

We make the following three assumptions about our system: (1) we assume that the client machine is a public device (i.e. anyone can access the protected resources via any available machine)., (2) the clocks of all on-lined nodes (a node may be a client or a server) are synchronous because timestamp check is adopted in our system to detect replay, (3) the administrators must take care of *Deny of Service* (DoS), which means that a particular server overloads because of the explosive flow from a lot of embedded Trojan horses.

In our system, user identification is based on the fingerprint verification. And a Java card exists for the storage of the user's fingerprint features. With the Java card, the accesses will not be limited to some specific machines. Moreover, the crack of the client machine will not result in the steal of user's fingerprint features.

Table 1. The symbols used in our system

Symbol	Description
c	client or Java card
v	target application server
TS	timestamp
RD	Random number
FP	fingerprint features
ID_A	the identity of user A
AD_A	the network address of user A
$doc1 \| doc2$	$doc1$ is merged with $doc2$
K_A	a secret key of user A
K_{A_B}	a secret key between user A and B
$K_{pub,A}$	the public key of user A
$K_{priv,A}$	the private key of user A
$E_K\{doc\}$	encrypt doc with the secret key K
$Hash\{doc\}$	the hashed value of doc

4.1. The Servers of the Proposed System

In our system, there are the following servers for user authentication (Figure 3):

1. ***Application Server***: an application server contains some valuable resources and these resources are accessible via some authentication processes.
2. ***Certificate Authority*** (CA): the CA is responsible for the management of the public key certificates (X.509 [18]) and the publication of a *Certificate Revocation List* (CRL) [17].
3. ***Register Authority*** (RA): the RA of each domain is the agent of a specific CA.
4. ***Authentication Server*** (AS): the AS is in charge of the issuance and the update of identity credential—*Ticket-Granting Ticket* (TGT), which can be used to request the tickets to several application servers.
5. ***Ticket-Granting Server*** (TGS): by presenting the TGT, the user will obtain a ticket to his/her target application server from the TGS.
6. ***Biometric Secure Policy Server*** (BSPS): the BSPS is the "doorkeeper" of our authentication system. A user must be verified with his/her fingerprint features before presenting a ticket to the target application server. In other words, the BSPS is trusted by the application server(s), and it manages the log-in procedures of users. This mechanism lightens the load of the application servers and the security risks which result from the different log-in policies.

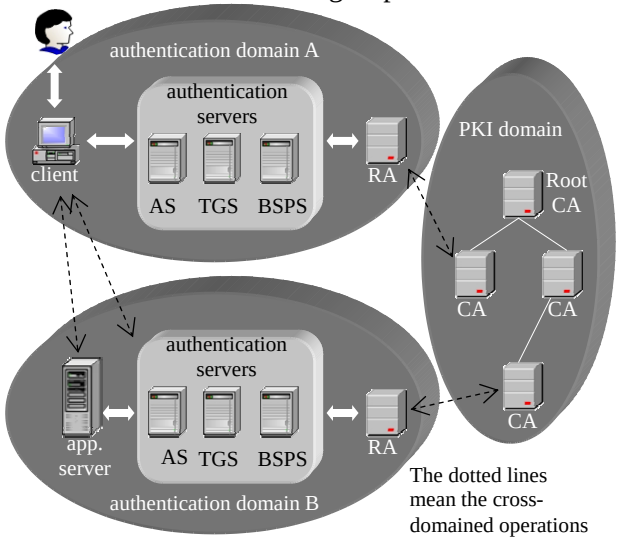


Figure 3. PKI domain vs. authentication domain

As what we see in Figure 3, the PKI domain is constructed of a number of hierarchical CAs. Each CA is responsible for the issuance and the revocation of public key certificate in a specific application. And a root CA exists for the management of all CAs. On the other hand, the authentication domain contains a single AS, one or some TGSs, one or several BSPSs, a few of client machines, and a number of application servers. The user must be verified by the authentication servers before starting his/her accesses.

The different between the PKI domain and the authentication domain is that the CA in the PKI domain services not only our system but also other applications. In other words, the PKI domain must be system independent. Our system focuses on the establishment of secure authentication in the authentication domain.

4.2. The Authentication Flowchart of the Proposed Authentication System

4.2.1. Intra-Domain Authentication

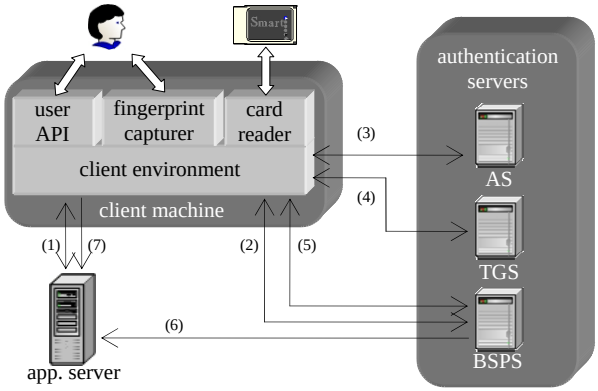


Figure 4. Access request in the intra domain

We illustrate the authentication flow of Figure 4 in the following walk-through.

(1) The user tries to access the protected service(s) in the target application server. The application server requests this user for a one-time session key. If the key is unavailable, the application server asks the user to be first verified by the trusted BSPS.

(2) The client sends a LOG_IN request to the BSPS. If this client machine has been verified by the BSPS. A pre-negotiated key between the smart card (not the client machine) and the BSPS will be used to encrypt the login ticket to the target application server before its delivery to the BSPS. If this client machine is a new node, the BSPS requests the client machine for the negotiation of a shared key. Then, the client machine asks the user to plug his/her smart card in the card reader. The client machine and the BSPS must apply mutual authentication before the transmission of the fingerprint features on the smart card. After the mutual authentication, the smart card and the BSPS use DH algorithm [18] to negotiate a secret key. Besides, the user must capture his/her fingerprint via the fingerprint capturer and pass it to the BSPS (if the fingerprint capturer supports encryption, the captured fingerprint can be encrypted before its transmission). Once the BSPS receives both fingerprint templates, it can verify the identity of the card holder. An AUTHENT request is responded to the LOG_IN request after the successful verification of the BSPS.

(3) The client sends its network address, the value of current time, and the public key of AS to the smart card (the public key of AS is obtained form the local RA). The card generates two session keys via two functions—$K_{au} = Hash\{FP \| TS\}$ and $K_c = Hash\{K_{au}\}$. Then $E_{K_{pub,as}}\{E_{K_{priv,c}}\{K_{au} \| TS\} \| K_c \| AD_c \| Flags\}$, which is an authentication request, is delivered to the AS via the transmission of the client machine ($Flags$ define some characteristics which this user needs in these requests of TGT and server-granting ticket). The AS decrypting $E_{K_{priv,c}}\{K_{au} \| TS\}$ to verifies the user's identity (note that TS is used to detect replay). Then the AS generates the same K_c from the hash function. By now, the smart card and the AS have known a shared key. This key can be used to encrypt and update the TGT (we will introduce the update of TGT in Subsection 4.2.3). $E_{K_{as\_tgs}}\{K_{c\_tgs} \| ID_c \| AD_c \| Flags \| lifetime\}$, which is a TGT, is delivered to the user. Next, the AS renews the K_c via $K_c = Hash\{K_c\}$. Then, $E_{K_c}\{TGT \| K_{c\_tgs} \| ID_{tgs} \| AD_{tgs} \| Flags \| TS\}$ is responded to the user's AUTHENT request. After the decryption with the updated K_c, the smart card can get the TGT and a secret key between itself and the TGS.

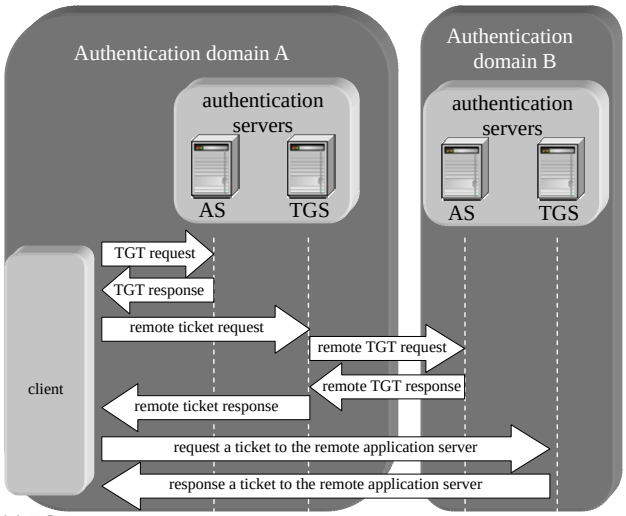
(4) $E_{K_{c\_tgs}}\{ID_v \| TGT \| ID_c \| AD_c \| Flags \| TS\}$, which is a TICKET request generated by the smart card, is sent to the TGS. A $Ticket_v$ with the format of $E_{K_{v\_tgs}}\{K_{c\_bsps} \| ID_c \| AD_c \| Flags \| lifetime\}$ is sealed in $E_{K_{c\_tgs}}\{Ticket_v \| K_{c\_bsps} \| ID_c \| AD_c \| Flags \| TS\}$ after the successful examination of the TGS. Once again, only the correct smart card can obtain the $Ticket_v$ and the K_{c_bsps} from the decryption of the TGS's response.

(5) The smart card generates and sends $E_{K_{c\_bsps}}\{ID_v \| Ticket_v \| ID_c \| AD_c \| Flags \| TS\}$ to the BSPS. After checking the $Ticket_v$, an encrypted one-time session key ($E_{K_{c\_bsps}}\{K_s \| TS\}$) will be delivered to the smart card.

(6) The BSPS must also tell the application server about the existence of the authenticated user by sending $E_{K_{v\_bsps}}\{K_s \| TS\}$.

(7) By now, the user can start his/her accesses with the K_s.

4.2.2. Inter-Domain Authentication



(1) TGT request:

$$E_{K_{pub,l-as}}\{E_{K_{priv,c}}\{K_{au} \| TS\} \| K_c \| AD_c \| Flags\}$$

$$K_{au} = Hash\{FP \| TS\}, \quad K_c = Hash\{K_{au}\}$$

(2) TGT response:

$$E_{K_c}\{TGT_{local} \| K_{c\_l-tgs} \| ID_{l-tgs} \| AD_{l-tgs} \| Flags \| TS\}$$

$$K_c = Hash\{K_c\},$$

$$TGT_{local} = E_{K_{l-as\_l-tgs}}\{K_{c\_l-tgs} \| ID_c \| AD_c \| Flags \| lifetime\}$$

(3) remote ticket request:

$$E_{K_{c\_l-tgs}}\{ID_v \| TGT_{local} \| ID_c \| AD_c \| Flags \| TS\}$$

(4) remote TGT request:

$$E_{K_{pub,r-as}}\{ID_v \| AD_c \| ID_{l-tgs} \| AD_{l-tgs} \| Flags \| TS \| RD\}$$

(5) remote TGT response:

$$E_{K_{pub,l-tgs}}\{TGT_{remote} \| document_{r-tgs}\}$$

$$TGT_{remote} = E_{K_{r-as\_r-tgs}}\{K_{c\_r-tgs} \| ID_c \| AD_c \| Flags \| lifetime\}$$

$$document_{r-tgs} = K_{c\_r-tgs} \| ID_{r-tgs} \| AD_{r-tgs} \| Flags \| TS$$

(6) remote ticket response:

$$E_{K_{c\_r-tgs}}\{TGT_{remote} \| K_{c,r-tgs} \| ID_c \| AD_c \| Flags \| TS\}$$

(7)request a ticket to the remote application server:

$$E_{K_{c\_r-tgs}}\{ID_v \| TGT_{remote} \| ID_c \| AD_c \| Flags \| TS\}$$

(8)response a ticket to the remote application server:

$$E_{K_{c\_r-tgs}}\{Ticket_v \| K_{c\_bsps} \| ID_c \| AD_c \| Flags \| TS\}$$

$$Ticket_v = E_{K_{v\ tas}}\{K_{c\_bsps} \| ID_c \| AD_c \| Flags \| lifetime\}$$

Figure 5. Ticket request of the inter-domain operation

If the client machine and the target application server are sited in different domains, the user must request the remote BSPS for a log-in key. The procedures of biometric verification are the same as the step (1) and (2) of the intra-domain operation. The TGT and the cross-domain ticket requests are similar to that in PKCROSS [19, 20]. In Figure 5, we show the flow of requesting a ticket for a remote service. First, the user requests the local AS for a TGT. Second, he/she presents the TGT to the local TGS for a ticket to the remote TGS. Thirdly, the TGS acts as an agent of this user—it requests the remote AS for a remote TGT (the local TGS and the remote AS apply mutual authentication to authentication the identity of each other). Fourthly, this user can request the remote TGS for a ticket to the target application server just like that of the intra-domain operation.

4.2.3. Credential Update

A long-lived user must update his/her TGT when it has expired. In our system, when a new user is requesting AS for a TGT, we assume that there is no pre-negotiated secret between the communicating parties. Therefore, public-key cryptography must be adopted in this case. Besides, A share key ($K_c = Hash\{FP \| TS\}$) will be established between Java card and AS by using a timestamp and the fingerprint features stored in the smart card. Once the TGT is overdue, the user first updates the K_c via $K_c = Hash\{K_c\}$. Then a TGT-update request ($E_{K_c}\{TGT \| ID_c \| AD_c \| Flags \| TS\}$), which contains the overdue TGT, is sent to the AS for another TGT. After the authentication in the AS, the user can get an updated TGT.

4.2.4. Ticket Update

When the user is accessing a long-lived service, his/her ticket may be invalid (the lifetime has expired). Therefore, ticket update is also an important functionality of authentication system. In our system, the user can ask an updated ticket by sealing the overdue ticket in the TICKET-UPDATE request ($E_{K_{c\_tgs}}\{ID_v \| Ticket_v \| ID_c \| AD_c \| Flags \| TS\}$) to the TGS. After the examination of the TGS, this user can obtain another ticket with a valid lifetime. Then, the user can resume his/her unfinished access with the updated ticket.

5. Security Analysis

Except IC card based authentication systems, most authentication systems rely on the client machines to store user's data. In these kinds of authentication systems, user must first log-in some particular client terminals when he needs the protected services in the network. This is inconvenient and insure because the user must register himself with a new client every time he uses an unfamiliar end-terminal. In the proposed system, client serves as a message transmitter. The user's data is stored in a smart card. He/She can access the protected services from any client machine which support our system.

Performance is what a user always concerns. On the other hand, security is always the first thing of administrators. In the symmetric cryptography based system, such as Kerberos, the trusted authentication server must maintain a password table, which is a public target of attackers. In our system, the initial authentication adopts public key cryptosystem. A secret key, which is derived from the timestamp and the fingerprint features stored in the smart card, is sealed in the credential request. If the user has to update his/her identity credential, this key is used to encrypt the transmitted message(s). Once the AS is cracked, the user does not re-enrol his/her fingerprint but renews this shared key by simply re-plugging his/her smart card. This mechanism brings users both security and convenience.

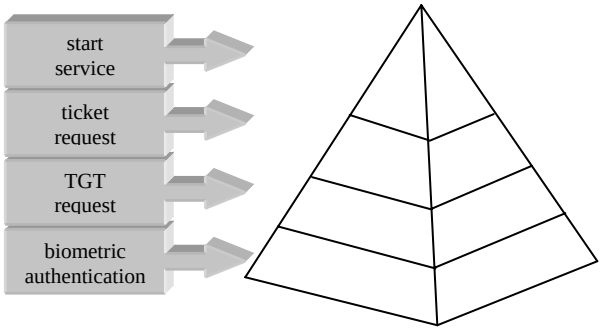


Figure 6. The authentication pyramid of the proposed system

In our system, the application servers can focus on their service providing. The procedures of log-in management are left to the BSPS. This brings two benefits: (1) lighten the load of application servers and (2) avoid security perplexity results from different authentication procedures and trusted linkages. The log-in examinations of BSPS include the biometric verification and ticket check. A user must prove himself/herself before presenting his/her ticket. With this mechanism, the hackers can not fool our authentication system if they can not get the fingerprint features from the crack of the smart card or the transmitted message between the smart card and the BSPS. Therefore, the authentication scheme of our system forms a pyramid authentication structure, which is showed in Figure 6. The basis of our system is biometric verification. In a word, we deem that the authentication is meaningful just one can propose the proof of his/her participation.

6. Conclusions

In this paper, we proposed a network authentication system to achieve on five considerations: **user convenience**, **storage security**, **robust authentication**, **administrator management**, and **attack resistance**.

In our system, users adopt the smart card to store their fingerprint features, which is a strong evidence of one's identity. In addition, smart card is securer than the client machine. In other words, our system brings users both **convenience** and **storage security**.

A new server named Biometric Secure Policy Server (BSPS) is proposed in our system to meet three goals—robust authentication, separated ticket management, and attack resistance. For a user, BSPS serves as the doorkeeper of application servers. Anyone must be verified by this server. From the viewpoint of application servers, they do not need to manage tickets. This task is left to BSPS. These application servers only have to take care of their services. This mechanism fits in with the concept of **task separation**.

Attack resistance plays an important role in our system designation. Because the client is insecure, we apply the smart card as the agent of the real user. The fingerprint features and other important messages of Java card must be encrypted before their transmission in the client machine. For intruders and Trojan horses, in order to obtain the communicating messages, they must derive the key between the smart card and the communicating parity from the stolen messages. But, a cracked key will be useless in the next session because the generating procedures of key involve timestamp. On the other hand, we take the attacks of servers into consideration. There are four kinds of servers in our system: Authentication Server (AS), Ticket-granting Server (TGS), BSPS, and application server. Once AS or TGS is cracked by a hacker, he can obtain the credentials and the tickets of all users. However, this hacker is not able to fool our system because he must first be verified via the biometric verification procedures before using these stolen documents.

Besides, we also consider the trade-off between security and performance in the construction of our system. The main problem of symmetric-key based authentication systems comes form the inevitable share secrets between servers and users. Other kinds of authentication systems are based on asymmetric-key cryptography. The secret communications between servers and users rely on public-key algorithms. Nevertheless, all public-key algorithms are more computing complex than symmetric-key cryptography. Thus, asymmetric-key based authentication systems are usually more secure but show poorer performance. In our

show poorer performance. In our system, the credential request of a new user is encrypted with AS's public key. At the same time, a secret key is sealed in this request. Once the credential has expired, this user can request AS for a renew credential by using the secret key. With this mechanism, we can reduce the use of asymmetric-key cryptography.

According to the above discussions, we can conclude that our authentication architecture can provide higher security and better efficiency than other authentication systems. Hence, our proposed system is appropriate for log-in authentication in distributed computing environment.

References

[1] John Kohl and B. Clifford Neuman. "The Kerberos Network Authentication Service (Version 5)," Internet Request for Comments RFC-1510, September 1993.

[2] Tom Packer and Denis Pinkas "SESAME-V4 Overview," Bull SA (Bull), International Computers Ltd (ICL), Siemens Nixdorf Informationssymteme (SNI), December 1995.

[3] Ashley, P. and Broom, B. "An Implementation of the SESAME Security Architecture for Linux," Australian UNIX and Open Systems Group Technical Conference, 1997.

[4] Isobe, Y., Saeto, Y., and Kataoka, M. "Development of Personal Authentication System Using Fingerprint with Digital Signature Technologies," *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, pp. 4039~4047, 2001.

[5] Ju-Chen Hsueh "Design of Authentication Systems with IC Cards," (NCS99) *Nation Computer Symposium*, 1999.

[6] Roger M. Needham and Michael D. Schroeder "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM*, vol. 21, no. 12, December 1978.

[7] V. L. Voydock and S. T. Kent, "Security Mechanisms in High-Level Network Protocols," *Computing Surveys of the ACM*, vol. 15, no. 2, June, 1983.

[8] John T., Kohl, B., Clifford Neuman, and Theodore Y. T'so "The Evolution of the Kerberos Authentication System," *IEEE Computer Society Press on Distributed Open Systems*, pp. 78~94, 1994.

[9] S. M. Bellovin and M. Merritt. "Limitations of the Kerberos Authentication System," *Computer Communication Review*, vol. 20, no. 5, pp. 119~132, October 1990.

[10] Ganesan, R. "Yaksha: Augmenting Kerberos with Public Key Cryptography," *Proceedings of the Symposium on Network and Distributed System Security*, pp. 132~143, 1995.

[11] Alvare, A. "How Crackers Crack Passwords or What Passwords to Avoid," *Proceedings of UNIX Security Workshop II*, August 1990.

[12] Ashley, P., Rutherford, M., Vandenwauver, M., and Boving, S. "Using SESAME's GSS-API to Add Security to UNIX Applications," (WET ICE '98) *Proceedings of the 7th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 359~364, 1998.

[13] Ashley, P. and Broom, B. "An Implementation of the SESAME Security Architecture for Linux," Australian UNIX and Open Systems Group Technical Conference, 1997.

[14] Vandenwauver, M., Govaerts, R., and Vandewalle J. "Overview of Authentication Protocols," *Proceedings of the 31st Annual IEEE Carnahan Conference on Security Technology*, pp. 108~113, 1997.

[15] Isobe, Y., Seto, Y., and Kataoka, M. "Development of Personal Authentication System Using Fingerprint with Digital Signature Technologies," *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, pp. 4039~4047, 2001.

[16] ISO/IEC JTC 1/SC 21, Technical Corrigendum 2 to ISO/IEC 9594-8: 1990 &1993 (1995:E), July 1995.

[17] "Public Key Infrastructure Specification," The specification issued by Object Management Group (OMG), February 2001, available at http://www.omg.org/issues/

[18] Diffie, W. and Hellman, M. "New Directions in Cryptography," *IEEE Transactions on Information Theory*, November 1976.

[19] Brian Tung, Tatyana Ryutov, and Clifford Neuman "Public Key Cryptography for Cross-Realm Authentication in Kerberos," internet draft working documents of IETF, May 8 2001, available at http://search.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-cross-08.txt

[20] Harbitter, A.H. and Menasce, D.A. "Performance of Public-Key-Enabled Kerberos Authentication in Large Networks," *Proceedings of 2001 IEEE Symposium on Security and Privacy*, pp. 170~183, 2001.