# Two SSO Architectures with a Single Set of Credentials

## Abstract

Single sign-on (SSO) is a widely used mechanism that uses a single action of authentication and authority to permit an authorized user to access all computers and systems. There are several SSO architectures. Some use a single set of credentials and some use multiple credentials. Each has advantages and disadvantages. SSO architectures with a single set of credentials are generally either Token-based or Public Key Infrastructure-based. This report explores these two flavors in depth, discusses their similarities and differences, and compares their strengths and weaknesses.

## Outline

1. What is SSO
    1.1 Introduction
    1.2 SSO terminologies
    1.3 SSO architectures
2. SSO architectures with a single set of credentials
    2.1 Token-based SSO system
        2.1.1 Kerberos authentication protocol
        2.1.2 Token-based SSO in HTTP environment
    2.2 PKI-based SSO system
        2.2.1 Integrating Public Key cryptography with Kerberos
3. Conclusions
4. References

# 1. What is SSO?

## 1.1 Introduction

Consider how many logons do you need each day to get your works done? If the number of different applications you use is not large, this may not be a big issue, but as the growing use of web-enable applications, the amount of time you spend on typing various passwords to login might be unacceptable. Imagine that you are a accountant that works in a big company, the first thing you usually do when you come to your office each day is to logon to a number of, we say, 10 applications which connect to databases that may locate in different companies. “A study conducted by the Network Applications Consortium (http://www.netapps.org) in large enterprises showed that users spend an average of up to 44 hours per year to perform logon tasks to access a set of 4 applications. The same study measured the content of the calls to companies’ helpdesk: 70 per cent of the calls were password reset requests.”[1]

The administration cost is also very high, no need to mention that managing multiple credentials for each user are much more difficult than just managing one single credentials. Furthermore, as the number of credentials for each user increase, so does the possibility of losing or forgetting them.

Single Sign-on is the technology that aims to solve the above problems, one single logon to the main authentication centre would subsequently enable user to get access to all other resources available. Multiple logons are no longer needed.

## 1.2 SSO terminologies

Below are the most commonly used SSO terminologies.

Authentication authorities: authentication authorities are the entities that are trusted by users to perform SSO functions [1].

Authentication servers: authentication servers are physical machines that perform SSO functions [1].

Application servers: application servers are physical machines which serve client machines.

Domain: a domain is a set of resource dominated by an authentication authority.

Credentials: There are various forms for those credentials: it might be, for example, a simple username/password or certificate, but other methods can be implemented. The SSO server performs a validation of the users' credentials using the underlying security infrastructure

Server key: server keys are keys shared by authentication servers and application servers.

Session key: Session keys are keys shared by client machines and application servers, a client first need to registers at an authentication authority, the authentication centre generate a session key and distribute it to both the client machine and the application server.

## 1.3 SSO architectures:

In Jan De Clercq’s Single Sing-on architectures [1], SSO systems are divided into two basic architectures: single SSO architecture and complex SSO architecture. Simple SSO architecture is when there is only one authentication authority available, so there is also just one set of credentials for each user. According to operating system vendors like Novell and Microsoft, this architecture is could be easily implemented in homogeneous LAN and intranet environment. Complex SSO architecture is when there are multiple platforms and multiple credentials managed by multiple authentication authorities. The situation for complex SSO architecture is much more complex and harder to be accomplished. Complex SSO architecture could be further distinguished to two basic schemes: complex SSO dealing with a single set of credentials and complex SSO dealing with many different credentials. There are two ways to accomplish complex SSO system dealing with a single set of credentials: Token-based and Public Key-based. There are also a couple of ways to handle complex SSO system dealing with many different credentials, but that is not the concern of this report. This report aims to discuss and contrast Token-based and PKI-based complex SSO systems.

## 2. SSO architectures with a single set of credentials

### 2.1 Token-based SSO system

The basic idea of Token-based SSO system is as follow: when a user submit the credentials to the taken-based authentication authority, the authentication authority check its credential database, if the credentials match, the user is then returned with a token. This token is temporarily cached in the user's computer until the user needs to access another application server governed by another authentication authority, this token is delivered to the secondary authentication authority to regain a ticket to the application server, the successful of this process rely on the trust that the other authentication authorities have on the primary authentication authority.

#### 2.1.1 Kerberos authentication protocol

Kerberos is a distributed authentication system that was developed on mid 80's as part of MIT's project Athena [2]. It is a classical implementation of Token-based authentication protocol.

The paper "Kerberos: An Authentication Service for Computer Networks" [2] has given a simplified version about how Kerberos protocol works. In this paper, the whole process is divided into two parts: Authentication Request and Response, Application Request and Response.

Authentication Request and Response: The client sends its request for the service of a particular application server to the authentication server; the request includes the client's identity, the name of the application server, a requested expiration time for the ticket and a random number which will be used to match the response. The authentication server sends a response if this is a valid request. The response includes the session key shared by the authentication server and the application server, the expiration time, the name of application server and a ticket, all of these are encrypted by the secret key shared by the client and authentication server.

Application Request and Response: as soon as the client receives the response, it extracts the session key and other information with its secret key, and forwards the ticket and an authenticator to the application server to prove its validity, the authenticator include the current time, a checksum and an optional encryption key, all encrypted with the session key. The application server decrypts the request with the session key it owns, if the process goes smoothly, the application server assume that the client is a valid user and return the data requested by the client. If a mutual authentication is needed, the application server needs to extract the client's time from the authenticator, and then encrypt it with some other information and return it to the client.

The simplified version of Kerberos authentication protocol ignores some important parts of the authentication process, the author did give a graph of the complete scheme but the explanation is not detailed enough. The complete scheme and detailed explanation can be found in Ian Downnard's paper Public-key cryptography extensions into Kerberos [3].

If a client has to submit its credentials each time it calls for the service of a new application server, the process would be cumbersome and perhaps more risky, fortunately, this is not the case. In real Kerberos systems, the client initially submits its credentials to the Key Distribution Centre (KDC), KDC consists of two parts:

Authentication Server (AS) and Ticket-Granting Server (TGS). AS validate the credentials and return a Ticket Granting Ticket (TGT), TGT then permit the client to request tickets to application servers.

### 2.1.2 Token-based SSO in HTTP environment

In paper “Single Sign-on architectures” [1], the author mentioned the token-based SSO could be implemented by using cookies in HTTP environment when introducing Token-based SSO architecture but no details are given. The paper “Single Sing-On using Cookies for Web Applications” extends this architecture [4].

A cookie is a set of information given to the web browser by the web server and store in the client machine, the server could then retrieve the information and provide customized service for the client.

Kerberos system provide basis for constructing secure SSO in network environment, however, it needs client side infrastructure and configuration. In HTTP-enabled environment, cookies could be used to construct SSO system and no extra installation or configuration is necessary.

The biggest difference between Kerberos system and Cookies-enabled SSO system is that the former use Remote Procedure Calls to transport authentication tickets, while the latter use cookies to play the role of tokens in Token-based SSO system.

## 2.2 PKI-based SSO system

There are two popular cryptographic methods: symmetric-key cryptography and asymmetric-key cryptography. Symmetric-key cryptography uses the same key (secret key) to encrypt and decrypt information. While for asymmetric-key cryptography, the key used to encrypt (public key) and the key used to decrypt (private key) are different. And the secret key could not be deduced from the public key.

Public Key cryptography is usually used to perform two main functions: (1). Encryption/Decryption: a user could make his/her public key public so everybody could send message encrypted with the public key to him/her, but the message could only be decrypted with the corresponding private key kept by the user. (2) Digital signature: a user could send a message encrypted with the private key to a receiver, the receiver could then determine if the message came from the user by decrypting the message with public key and check the originality of the message.

In conventional Token-based SSO system, symmetric-key cryptography is used to encrypt the tickets, the authentication server must have application server’s secret key stored in the database. In Kerberos system, Key Distribution Centre is responsible for storing registered application server’s secret keys. Token-based SSO system works well but it can be improved by adopting asymmetric-key cryptography in two aspects: Firstly, registration is easier, because there are no needs to transfer the secrete key to the authentication server, public key could be securely transported over the network without worrying being exposed. Secondly, since there are no copies of secret keys stored in the authentication centre, attackers could not compromise the system by hacking and retrieving keys from authentication centre.

Strictly saying, PKI-based SSO system is still a kind of Token-based SSO system because they differ in cryptographic methods but tokens are still used to permit the client to get access to application servers.

### 2.2.1 Integrating Public Key cryptography with Kerberos

PKI-based SSO system is mentioned in paper “Single Sign-on Architectures” [1], the basic structure is presented but little details are available. The paper “Public-key cryptography extensions into Kerberos” [3] aims to describe the schemes to extend Kerberos system with public key cryptography.

Although integrating Public Key with Token-based SSO like Kerberos could enhance the security and scalability of the SSO system, the performance of Public Key cryptography is the bottleneck. Poor performance of Public Key cryptography reflects in two ways: (1). the size of Public Key is much larger than the size of Secret Key, according to data provided by Ian Downnard [3], the Public Keys are usually 10 to 20 times larger than the Secret Keys. (2). the calculation process of encryption or decryption with Public Keys takes much longer than the process with Secret Keys.

In this paper, author introduced three efforts made by researchers to extend Kerberos system with Public Key infrastructure: Public Key cryptography for Initial Authentication in Kerberos (PKINIT), Public Key cryptography for Cross Realm Authentication in Kerberos (PKCROSS) and Public Key based Kerberos for Distributed Authentication (PKDA).

## PKINIT

Contrast to transitional Kerberos system, PKINIT use Public Key cryptography to perform the initial authentication, if the authentication process is successful, the remaining transaction will use secret keys that stored in KDC database.

In PKINIT system, a client send a request for TGT to the KDC along with the credentials and digital signature, the KDC check the credentials and digital signature to see if the request is valid. If the request is valid, KDC send the TGT encrypted with the client's public key back to the client. The message is encrypted with KDC's private key so the client could check the signature to make sure the message is sent by a valid KDC. This is called mutual authentication.

## PKCROSS

Public Key cryptography for Cross Realm Authentication in Kerberos, as its name suggest, is designed to build cross-realm Public Key SSO system.

The Public Key cryptography of the system takes place only between local KDC and remote KDC. “The messages exchanged between the two key distributions centers closely follow the PKINIT specification, with the local key distribution center acting as the client.”[3]

Since PKINIT use Public Key cryptography between the client and the local KDC and PKCROSS use Public Key cryptography between local KDC and remote KDC, the author suggests that the Public Key cryptography can be applied to the whole Kerberos framework by integrating PKINIT with PKCROSS [3].

**PKDA**

PKDA is a created to enhance privacy of the clients and improve scalability and security of Kerberos system. It is a totally different approach compared to PKINIT and PKCROSS. “Client-side privacy is increased by simply ‘moving the client identity fields from unencrypted to encrypted portions’ of the authentication messages. Increased scalability and security is attempted by moving authentication procedures away from centralized key distribution centers to between the individual clients and application servers on a network.”[3]

KDC is totally made redundant in PKDA, clients do not rely on KDC for initial authentication, and all services are between clients and application center supported by Public Key cryptography.

PKDA has been expected to perform better than PKINI and PKCROSS because it avoids the bottleneck of KDC; however, the practice shows that PKDA may not necessarily faster than PKINI or PKCROSS because it needs Public Key cryptography each time a client require service from a application server.

## 3. Conclusions

There are two main solutions for SSO systems dealing with a single set of credentials: Token-based SSO and PKI-based SSO, Token-based SSO has inherent drawbacks using Secret Key cryptography for transactions. PKI-based SSO is designed to improve security and scalability of Token-based SSO by extending Public Key cryptography in the process of validating tokens.

Kerberos authentication protocol is a classical example of Token-based SSO system, Kerberos system provide secure means of authentication by adopting Secret Key encryption and mutual authentication. Token-based SSO in HTTP environment could also be achieved by using HHTP cookies without the needs for client side instillation and configurations.

Researches have been trying to develop mature Kerberos system with Public Key cryptography support. PKINT, PKCROSS and PKDA are results of these efforts. PKINI use Public Key cryptography to perform an initial registration and use Secret Key to perform subsequent services, PKCROSS use Public Key cryptography to perform cross realm transactions. PKDA made a fundamental change to Kerberos standard by removing KDC and perform mutual authentication between clients and applications servers.

## 4. References

[1] J. De Clercq, “Single Sign-On Architectures” In G. Davida et al. (eds.): InfraSec 2002, LNCS 2437, 2002. Pages: 40-58,

[2] Neuman B.C. Ts'o T, “Kerberos: An Authentication Service for Computer Networks” Communications Magazine, IEEE, Volume: 32, Issue: 9, Sept. 1994. Pages: 33 - 38

[3] Downnard I, “Public-key cryptography extensions into Kerberos” Potentials, IEEE, Volume: 21, Issue: 5, Dec. 2002-Jan. 2003. Pages: 30 - 34

[4] Sama V, “Single Sign-on using Cookies for Web applications” Enabling Technologies: Infrastructure for Collaborative Enterprises, 1999. (WET ICE '99) Proceedings. IEEE 8th International Workshops on, 16-18 June 1999. Pages: 158 - 163