

Design of information security transmission scheme and related authentication services based on Kerberos

Puyuan Du^{1*,†}, Xinlong Yan^{2,†}, Peiyao Feng^{3,†}

¹ Department of computer science and technology Jilin University Changchun, China

² Department of computer science and technology The University of Melbourne Beijing China

³ Department of computer science and technology Century College of Beijing University of Posts and Telecommunications Beijing, China

* Corresponding Author Email: dpy2120@mails.jlu.edu.cn

†These authors contributed equally.

Abstract. A plethora of cloud service companies have sprung up in response to the growing amount of data. Data security protection solutions have been proposed by service providers. As a result, a rising number of people are using the cloud to store their own data. However, as technology advances and hackers' capabilities increase, service providers' protection mechanisms become less effective, and data security cannot be guaranteed. We review the logic of the Kerberos algorithm and discuss the work of each part in this paper. This document also includes and discusses difficulties with Kerberos, as well as improvement strategies. The main focus will be on strengthening the security of the Kerberos secure communication process, which will serve as a reference for Kerberos security transmission and authentication research. It solves the problem of the symmetric encryption algorithm's relatively low security in the authentication process by replacing it with the RSA one-time encryption algorithm; in addition, the Kerberos authentication protocol is added between the client and the cloud service provider to secure the data transmission process. Furthermore, this paper explains some basic authentication principles.

Keywords: Online security; identity authentication; kerberos; encryption algorithm; kerberos authentication configuration.

1. Introduction

Data on the transmission, storage, and deletion stages of the transportation life cycle will be vulnerable to security risks. The process of sending data from a client to a cloud server is the initial stage in the life cycle of cloud data, and data security is critical. As a result, data requires high security measures at this time to avoid assaults by the unauthorized users [1].

With the advancement of network society, an increasing number of people are turning to cloud resources to store their personal data. However, as technology advances, the safety of data provided by cloud services is becoming less effective in comparison to the methods used by hackers. By using inappropriate methods, unlawful users are able to crack and steal the data of many users. As a result, it is difficult for consumers to rely on cloud service providers' data security guarantees. In this work, the Kerberos authentication protocol is upgraded, and an improved data security protection mechanism is discussed, based on the security problem of data onto the transmission process and targeting at the shortcomings of Kerberos. To a large extent, the one-time encryption technique is introduced to this scheme to assure data security [2].

2. Kerberos

The Kerberos authentication protocol is a network security authentication mechanism that verifies the network's security. The protocol authenticates between the client and cloud service apps by transferring sensitive information about the user [3]. At the same time, the agreement allows for the prevention of illegal eavesdropping and the response to illegal user attacks. The Kerberos authentication protocol ensures data confidentiality and integrity while also maintaining security

authentication capabilities [4]. The key distribution center (KDC) of the Kerberos authentication protocol is known as the Kerberos authentication protocol's key distribution center, and it is primarily used to handle security authentication for users and services in trusted third-party platforms. Authentication server (AS), ticket grant server (TGS), and real server S are the three servers that make up the Kerberos authentication system [5]. Figure 1 depicts the procedure.

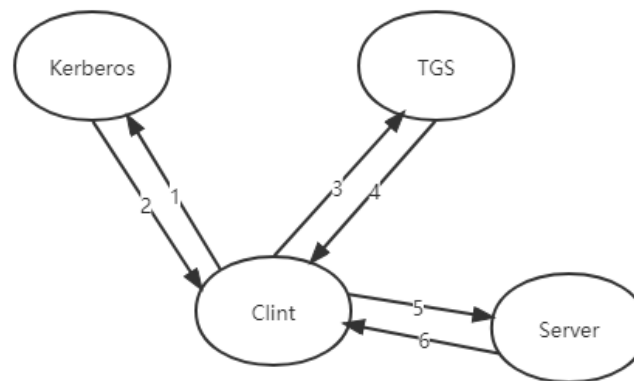


Figure 1. Kerberos authentication process.

There are multiple sub-protocols in the Kerberos protocol (or exchanges). A client can request credentials from a Kerberos server using one of two techniques. The client sends a clear text request to the AS for a ticket for the desired server in the first approach. The client's secret key is used to encrypt the response. This is typically a request for a ticket-granting ticket (TGT), which can be utilized with the ticket-granting server later (TGS). The client submits a request for the TGS in the second way. The client utilizes the TGT to authenticate itself to the TGS in the same way that it would any other Kerberos-enabled application server. The reply is encrypted in the TGT's session key. Though the AS and TGS are described as separate servers in the protocol specification, they are implemented as various protocol entry points within a single Kerberos server in practice.

Credentials can be used to validate the identity of the principals in a transaction, maintain the integrity of messages sent between them, or protect the privacy of the messages once they've been accessed. The application is free to select any level of protection it requires.

The client sends the ticket to the application server to authenticate the identities of the principals in a transaction. Because the ticket is sent "in the clear" (parts of it are encrypted, but the encryption does not prevent replay) and could be intercepted and reused by an attacker. The additional information is sent to show that the message came from the principal to whom the ticket was issued. The session key encrypts this information (called the authenticator) and provides a timestamp. The timestamp establishes that the message was created recently and is not a replay. The fact that the authenticator in the session key is encrypted. It demonstrates that it was generated by the party that owns the session key. Since the session key is only known by the requesting principal and the server (it is never communicated over the network in clear text), the client's identity is guaranteed.

The session key can also be used to ensure the integrity of messages sent to principals (passed in the ticket and contained in the credentials). Both replay and message stream alteration attacks can be detected using this method. It's done by creating and sending a collision-proof checksum (also known as a hash or digest function) of the client's message that's keyed with the session key. Encrypting the data to be passed using the session key contained in the ticket or the sub-session key found in the authenticator can ensure the privacy and integrity of communications exchanged between principals.

The aforementioned authentication exchanges necessitate read-only access to the Kerberos database. However, there are instances the database records need to be changed, such as when adding new principals or updating a principal's key. This is accomplished through a protocol between a client and the Kerberos Administration Server, a third Kerberos server (KADM). A mechanism for

preserving multiple copies of the Kerberos database exists as well. This document does not cover either of these protocols.

3. Disadvantages of the Kerberos certification protocol

The Kerberos certification protocol has a number of drawbacks, the most significant of which are as follows:

Kerberos is only partially compatible with most Red Hat Enterprise Linux servers' Pluggable Authentication Modules (PAM) architecture.

Kerberos assumes that each user is trustworthy, but that they are connecting to an untrustworthy host on an untrustworthy network. Its main purpose is to stop unencrypted passwords from being sent across the network. The entire Kerberos authentication system is at danger if anyone other than the proper user has access to the one server that provides tickets used for authentication, known as the key distribution center (KDC).

To use Kerberos, an application's source code must be changed to make the necessary calls to the Kerberos libraries. Kerberos-aware applications, or kerberized applications, are those that have been modified in this fashion. Due to the size or design of some applications, this can be rather difficult. Changes to the server and client's communication must be made for other incompatible applications. This may necessitate substantial programming. Closed-source apps that don't support Kerberos by default are frequently the most troublesome.

Kerberos is a one-size-fits-all solution. Any unencrypted credentials communicated to a non-Kerberos aware service are at danger if Kerberos is utilized on the network. As a result, Kerberos provides no benefit to the network. To secure a Kerberos network, you either utilize Kerberos-aware versions of all client/server apps that send passwords unencrypted, or you must not use any client/server applications at all.

4. Improvement based on a one-secret Kerberos protocol

The Kerberos authentication protocol establishes a connection between the user and the cloud server, allowing data to be transferred from the client to the cloud server [6]. The main focus of this paper is on the Kerberos certification protocol's flaws. The information provided back to the client by the server is easily cracked. Figure 2 depicts the procedure. The first is to gradually replace the RSA one-secret encryption technique in this protocol with the symmetric encryption algorithm.

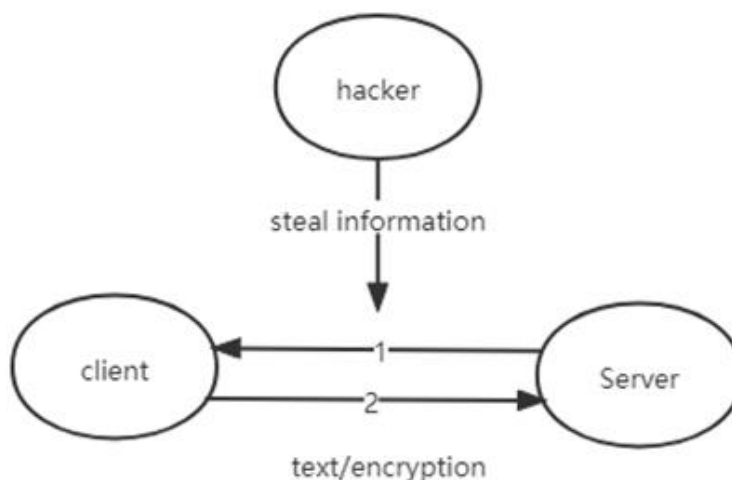


Figure 2. Information stealing process.

The RSA public key cryptosystem employs a variety of encryption and decryption keys. Deducing the decryption key from the known encryption key is computationally impossible.

The encryption key (i. e. public key) PK is public information in the public key cryptosystem, whereas the decryption key (i. e. secret key) sk must be kept hidden. Both encryption and decryption

algorithms E and D are open to the public. Although the public key PK determines the decryption key SK, SK cannot be calculated using the PK.

The famous RSA algorithm was created in 1978 based on this principle. It's commonly a pair of RSA keys. One of which being the secret key that the user saves; The other is a public key, which can be shared with the general public or even stored on a network server. The RSA key should be at least 500 bits large to improve security, and 1024 bits is commonly recommended. This increases the size of the encryption computation. When sending information, a combination of traditional encryption methods and public key encryption methods is frequently utilized, with the material being encrypted with an upgraded des or idea discussion key. The conversation key and information summary are encrypted with an RSA key. The second party decrypts the information with the separate keys and checks the information summary after getting it.

The RSA algorithm is the most commonly researched public key algorithm. It's been nearly three decades since it was first proposed. It has been put to the test by a variety of attacks and is gradually being accepted by the public. It is widely regarded as one of the greatest public key schemes available today. In 1983, MIT filed a patent application in the United States for the RSA algorithm. The RSA algorithm's premise is depicted in Figure 3.

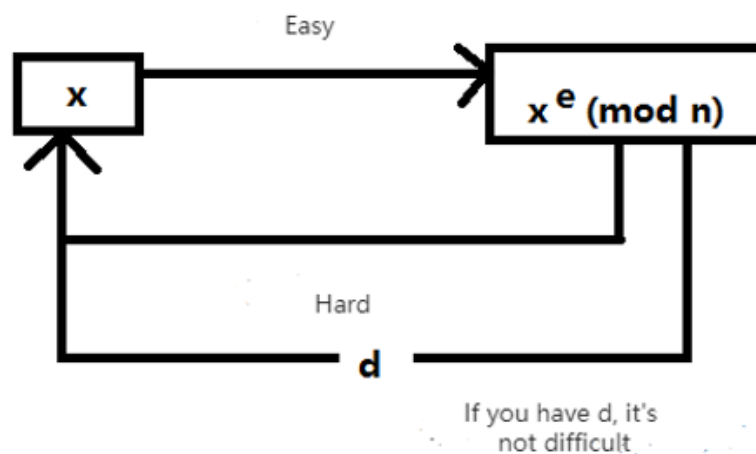


Figure 3. Security Foundation.

The specific methods are presented as below:

4.1. Client to KDC

The client must first create a service application and send a service request to the KDC before the data can be uploaded. The KDC issues a TGT in order to be eligible to connect with the TGS via the bill. The client's application request must include the user's own ID information as well as a TGT that can be proven to connect with the TGS. This approach is identical to Kerberos [7].

4.2. KDC to Client

The key distribution center initially confirms whether the received ID is in the database according to the information after receiving the service request submitted by the client. If the DC finds a key TC in the database, it will construct a key TC for communication between the client and TGS, and then encrypt TC with the public key PKC generated by the RSA encryption technique. It can generate authentication information AC to prove that the client can communicate with TGS, encrypt AC with the public key PK generated by TGS via RSA [8], and return all user information.

4.3. Client to TGS

The client gets the information returned by the KDC, confirms the information, assesses whether the call can be obtained, and then sends the bill Tc and the service application for the session to the TGS, which includes the server ID, timestamp, and session key Tc [9].

4.4. TGS to Client

TGS checks the information given by the verification client and sends a service note to the user client.

4.5. Client to Server

The client transmits the server TC and the authentication code AC after getting the service ticket. Sid and pkt are included in the data. After verifying the user's identification through comparison, the server provides the service.

4.6. Server to Client

The server will send a message to the client if it supports relevant services. The public key stamps and encrypts the communication in real time. The client decrypts the message and compares the time stamps after receiving it. If the time stamps are the same, the session key supplied back by the server is correct, indicating that the server's identification is correct. Finally, the connection will be created, and data from the client to the server will be exchanged.

5. Certification Service

The user's public key certificate is the most important aspect of X.509. The certificate is created by a trusted certificate issuing authority CA, and the CA or the user then places it in the directory for other users to see. The directory server is not responsible for creating public key certificates for users; rather, it is responsible for making public key certificates accessible to users [10]. The certificate format is shown in Fig.4.

```
Certificate ::= SEQUENCE {
    tbsCertificate      TBSCertificate,
    signatureAlgorithm  AlgorithmIdentifier,
    signatureValue      BIT STRING }

TBSCertificate ::= SEQUENCE {
    version             [0] EXPLICIT Version DEFAULT v1,
    serialNumber        CertificateSerialNumber,
    signature            AlgorithmIdentifier,
    issuer              Name,
    validity            Validity,
    subject             Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID      [1] IMPLICIT UniqueIdentifier OPTIONAL,
                        -- If present, version MUST be v2 or v3

    subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL,
                        -- If present, version MUST be v2 or v3
    extensions          [3] EXPLICIT Extensions OPTIONAL
                        -- If present, version MUST be v3
}
```

Figure 4. Certificate format.

The following information is included in the certificate [11]:

Version number.

The number of the sequence.

Identifier for the signature algorithm.

The name of the company that issued the certificate.

Period of validity.

Name of the subject.

- Information about the subject's public key.
- The issuer's unique identifier.
- The subject's unique identifier.
- Extended domain.
- Signature.

5.1. Obtaining a certificate

The following are typical properties of the certificate generated by the CA for the user:

Any other user, as long as he receives the CA's public key, can obtain the public key signed by the CA for the user.

No one else, except the CA, has the ability to modify the certificate's content in an undetectable manner. If user A obtained his public key certificate from X1 and user B obtained his certificate from X2, the following scenario applies. Even if A can read B's certificate, he cannot validate X2's signature without access to X2's public key. A is unable to use B's certificate as a result.

If CAX1 and X2 have securely exchanged public keys, however, A can obtain B's public key by following the steps below:

A acquires the certificate of X2 signed by X1 from the directory since A has access to X1's public key and can thus verify X2 and obtain X2's public key.

A then gets B's certificate from the directory, verifies it with X2's public key, and gets B's public key from it [12].

5.2. Revocation of certificate

Each certificate has a specific validity period. Because the user's secret key may have been disclosed or the user is no longer authenticated by the CA, certain certificates will be revoked by the CA before the expiration date. As a result, each CA must also maintain a certificate revocation list (CRL), which keeps track of all certificates that haven't yet expired but have already been revoked. Before saving the CRL in the directory for others to check, the CA must sign it.

The data field of the CRL contains the name of the issuing CA. The date the CRL was established. The date for the next CRL is expected to be announced. The data field for each revoked certificate.

6. Certification process

6.1. One-way authentication

User A sends a message to user B to prove A's identity, the message's origin, the message's intended recipient, as well as the message's integrity and freshness.

To provide one-way authentication, the message sent by A to B should contain multiple data items signed by A's key. The data item should include the timestamp identifier t_A , as well as the one-time random integers r_A and B . The timestamp includes the message's generation, the expiration times, and a one-time random number r_A to prevent replay attacks, in order to account for unanticipated delays in the message transmission process. Before the deadline, this message should be the only one with a random number of r_A . As a result, B can always store r_A before the deadline of this message in order to reject other messages with the same r_A . If the message sent by A to B is only for authentication, it might be used as a certificate to indicate the submission from A to B. If the data item signed by A with its own secret key isn't just for authentication, it can also include other information signal data to ensure the information's validity and integrity [14].

6.2. Two-way authentication

The foundation for two-way authentication is the previously discussed one-way authentication. The identity of B, the identity of the response message, the identity of the expected recipient of the response, and the completeness and freshness of the response message are all established by B's

answer to A. The response message contains the one-time random number r_A supplied by A (to make the response message valid), the time stamps t_B , and the one-time random number r_B created by B. Like one-way authentication, the answer message may include other messages as well as the session key, which is encrypted by A's public key.

6.3. Three-way authentication

Following the two-way authentication described above, A signs a random number B and sends it to B, resulting in three authentications. The purpose of three-way authentication is for each party to return the one-time random number sent by the other. As a result, neither party needs to check the timestamp; instead, both sides must examine the other party's one-time random number to see if a replay attack has occurred. This strategy can be used when the two communication parties are unable to achieve clock synchronization [15].

7. Conclusion

Security of transactions has become a major issue. Researchers and experts have been paying close attention to the security issues that arise during data transmission. The Kerberos authentication protocol is a data transmission security technique based on RSA one-time encryption technology that is examined in this paper. Kerberos authentication protocol and asymmetric encryption technology constitute the scheme, which is a data transmission channel. Kerberos is a client-server authentication system that uses secret key exchange messages to authenticate clients and servers. Furthermore, the Kerberos authentication protocol can prevent attacks and eavesdropping, increase data security, and alleviate data transmission security issues.

References

- [1] Bhatia, A., Xiong, C., Zhang, F., Kumar, N., & Cai, Y. (2019). U.S. Patent No. 10,403,372. Washington, DC: U.S. Patent and Trademark Office.
- [2] Neuman, B. C., & Ts'o, T. (1994). Kerberos: An authentication service for computer networks. *IEEE Communications magazine*, 32(9), 33-38.
- [3] Malar, M. B. A., & Prabhu, J. (2020). Trust based authentication scheme (tbas) for cloud computing environment with Kerberos protocol using distributed controller and prevention attack. *International Journal of Pervasive Computing and Communications*.
- [4] Saltzer, J. H. (2021). On the Origin of Kerberos. *IEEE Annals of the History of Computing*, 43(1), 89-91.
- [5] Anita Narwal and Sunita Tomar. (2015). Kerberos Protocol: A Review. *International Journal of Engineering Research and*, V4(04).
- [6] Fatima, S., & Ahmad, S. (2021). Quantum Key Distribution Approach for Secure Authentication of Cloud Servers. *International Journal of Cloud Applications and Computing (IJCAC)*, 11(3), 19-32.
- [7] Bhagwat, P., Tapkir, V., Andure, K., & Sorte, K. Single Sign on for Secure Authentication of Web Services using Kerberos.
- [8] Lu, Y., Yu, K., & Lv, X. (2021). Image encryption with one-time password mechanism and pseudo-features. *Multimedia Tools and Applications*, 80(10), 15041-15055.
- [9] Ashok Kumar J and Gopinath Ganapathy. (2019). Kerberos Authorization with Hybrid Access Control Model in Public Cloud. *International Journal of Recent Technology and Engineering (IJRTE)*, 8(4), pp. 7267-7271.
- [10] Uahhabi, Z. E., & Bakkali, H. E. (2018). Trust assessment of X. 509 certificate based on certificate authority trustworthiness and its certificate policy. *International Journal of Internet Technology and Secured Transactions*, 8(1), 103-136.
- [11] Kent, J. J. (2018). U.S. Patent No. 10,063,536. Washington, DC: U.S. Patent and Trademark Office.
- [12] Lippert, M., Karatsiolis, E. G., Wiesmaier, A., & Buchmann, J. (2005, May). Directory Based Registration in Public Key Infrastructures. In *IWAP* (pp. 17-32).

- [13] Adja, Y. C. E., Hammi, B., Serhrouchni, A., & Zeadally, S. (2021). A blockchain-based certificate revocation management and status verification system. *Computers & Security*, 104, 102209.
- [14] Yu, L., Yang, Z. Y., & Zhuo, Z. P. (2021). Extension of PCL Theory and Its Application in Improved CCITT X. 509 Analysis. *International Journal of Network Security*, 23(2), 305-313.
- [15] Ceccanti, A., Vianello, E., & Giacomini, F. (2020). Beyond X. 509: token-based authentication and authorization in practice. In *EPJ Web of Conferences* (Vol. 245, p. 03021). EDP Sciences.