Kerberos Plus RSA for World Wide Web Security

Don Davis*

August 3, 1995

Abstract

We show how to use Kerberos to enable its clients to interact securely with non-Kerberized World Wide Web servers. That is, our protocol does not require that the Web server be a member of a Kerberos realm, and also does not rely on time-synchronization between the participants. In our protocol, the Kerberos client uses the Web server's public-key certificate to gain cryptographic credentials that conform to public-key authentication standards, and to SHTTP. The client does not perform any public-key encryptions. Further, the client is well-protected from a man-in-the-middle attack that weakens SSL. Our protocol conforms to the current specifications for the Kerberos protocol and for the Secure Hypertext Transfer Protocol.

1 Introduction

The effort to secure the World-Wide Web for electronic commerce brings to a sharp focus the difficulties that have prevented the Internet from adopting a single coherent security model. On the one hand, like the Internet, the Web as a whole is too big to bring into a centrally-administered key-distribution system like Kerberos. [5, 13, 8] Indeed, the whole point of the Web is to bring together clients and servers across great distances, so it seems that public-key security is the only natural model, and the only security model that can be made to work. On the other hand, though, it's unrealistic to suppose that all Web clients will have public-key certificates any time soon. Public-key cryptography costs more money and more cycles than many home-computer users want to spend on an invisible benefit.

Netscape's Secure Socket Layer protocol [7] attempts to finesse this problem by waiving the client's authentication to the server, so that only the server needs a public-key certificate. This approach prevents the Web server from detecting credit-card fraud, which puts all credit-card holders at risk. CommerceNet's proposed standard, S-HTTP [10], provides for full mutual authentication, and supports several varieties of public-key and private-key cryptography. However, S-HTTP cannot do anything to bring these several varieties into cooperative communion; it enables public-key clients to shop at public-key-authenticated Web pages, and brings Kerberos-equipped clients and servers together, but cannot bring the two groups together.

What Web commerce needs now is a lot of Web users who have cryptographic credentials of some sort, and some way to bring the clients' and servers' disparate kinds of credentials together. We believe that the natural way to authenticate most home-computer users is with trusted third-party key-distribution, such as Kerberos provides. Internet access providers have to do password and account administration anyway, and some of them are beginning to install Kerberos. We also believe that public-key cryptography is more natural for authenticating the Web servers. We propose as a solution a merger or marriage between these competing styles of cryptography, with Kerberos taking responsibility for bridging the differences. Our protocol combines Kerberos' performance advantages with public-key's terrific geographic reach.

Other mergers between Kerberos and RSA have been proposed. MIT's Schiller and Atkins have built a Kerberized service that certifies PGP public keys [15] for Project Athena's users [12]. Neuman et al. have preliminarily proposed in an Internet Draft that Kerberos can help its clients manage their RSA keys [9]. Finally, the present paper's protocol is derived from a protocol that we presented in [3].

2 Problem Statement

We assume that each Web user will share a password with at least one Kerberos service, which typically will be administered by the user's commercial

*Affiliations: Independent Consultant, 1318 Comm. Ave #16 Allston, MA 02134; *don@mit.edu*

Internet-access provider. Some users might get their net access and credentials via an employer or a university network. We don't assume that these Kerberos servers are linked in interrealm trust relationships. We don't assume that the Kerberos clients are capable of public-key encryption operations. Finally, we don't assume that the Kerberos clients' clocks are synchronized, because of a recent finding that Kerberos' requirement for synchronized clocks can be relaxed [1].

We assume that each commercial Web server will gain an RSA public-key certificate [11], that the server will support S-HTTP, and that the server can perform a few styles of symmetric-key encryption, including DES, 3DES, and RC4. We do not assume that the Web servers can contact a Kerberos server, and we don't assume that the Web servers have any administrative relationship with a Kerberos service.

With these assumptions, we want to extend Kerberos and S-HTTP as little as possible, so as to enable any Kerberos-authenticated client to mutually authenticate with any public-key-bearing Web server. Thus, the client and server need to share a symmetric session-key. It turns out to be surprisingly easy to issue such a session-key.

3 Protocol Notation and Review

To clarify our cryptographic terms and notation, this section will quickly review the Kerberos protocol. [1] Because S-HTTP supports Kerberos as an authentication option, we don't refer to S-HTTP details, and don't need to review the S-HTTP protocol itself.

Before proceeding, we must clear up a jargon collision: the Kerberos and RSA communities use the term "private key" in different ways. To replace "private key," we will use "secret inverse key" for the non-public RSA keys, and we'll use "password key" for a Kerberos client's identifying DES key. The other types of key we'll talk about are RSA public keys and symmetric session keys. These last may be RC4, IDEA, DES, or other keys. We'll denote public keys by P_a, secret inverses by P_a^{-1}, password keys by K_a, and session keys by $K_{a,b}$.

Now, suppose Alice is a client of a Kerberos server KRB, so that she shares a password key K_a with the server. When Alice logs in, KRB can use her password key to securely issue a session-key for her to use later:

$$A \rightarrow KRB \quad : \quad A,\ KRB \tag{1}$$

$$KRB \rightarrow A \quad : \quad \{KRB,\ L,\ K_{a,krb}\}^{K_a},\ \{A,\ L,\ K_{a,krb}\}^{K_{krb}} \tag{2}$$

Alice's request says that she wants a ticket for the Kerberos service, The Kerberos server replies with two similar encrypted messages, which identify Alice and KRB to each other as the only bearers of a new symmetric session-key $K_{a,krb}$. Each message also limits the key's lifetime to a fixed span L. The second message is called Alice's Ticket-Granting Ticket. We denote it $T_{a,krb}$, or TGT informally.

Alice has not yet proven her identity to anyone; her request was entirely in plaintext. As we will see below, her TGT enables her to share session-keys with other services, and she'll use those keys to authenticate herself to them.

4 Our Proposed Solution

Our approach to Web security is to equip each Kerberos server with the ability to RSA-encrypt its key-certifying tickets. For this to work, each Kerberos server will need a public-key pair $P_{krb},\ P_{krb}^{-1}$ of its own. To prepare a symmetric session-key for a Web server, a KDC will encrypt with its own secret-inverse key, and again with the Web server's public key.

So, suppose Alice contacts Bob's commercial Web server; when she decides to order something from Bob, she requests his public-key certificate:

$$A \rightarrow B \quad : \quad \text{"}cert.\ request\text{"} \tag{3}$$

$$B \rightarrow A \quad : \quad C_b,\ C_{ca} \tag{4}$$

Here, C_b is Bob's public-key certificate $\{B,\ P_b\}^{P_{ca}^{-1}}$, which asserts that P_b is Bob's public key, and which is signed with the secret-inverse key P_{ca}^{-1} of Bob's Certification Authority CA. Similarly, C_{ca} is CA's certificate; in practice, Bob may send Alice a chain of CA certificates, and not just one [4].

Now, Alice sends Bob's credentials and her own to her Kerberos server KRB, in a request for a new session key and ticket:

$$A \rightarrow KRB \quad : \quad B,\ C_b,\ C_{ca},\ T_{a,krb}, \{time\}^{K_{a,krb}} \tag{5}$$

$$KRB \rightarrow A \quad : \quad \{B,\ L,\ K_{a,b}\}^{K_{a,krb}},\ \{\{A,\ L,\ K_{a,b}\}^{P_b}\}^{P_{krb}^{-1}}, C_{krb} \tag{6}$$

Alice's request is essentially a standard user-to-user ticket request [2, 8], except that instead of sending a TGT $T_{b,krb}$ for Bob, she sends his certificate-chain.

[1] For clarity, we avoid mentioning the Ticket-Granting Service [8] by name, and we omit the details of how to avoid synchronizing clocks [1].

The last part of her request is an encrypted timestamp. To service Alice's request, the Kerberos server first validates Bob's public key P_b with his Certification Authority's key P_{ca}. Kerberos' response is also standard, except that Alice's ticket is RSA-encrypted twice, instead of being DES-encrypted as is usual. [2] Kerberos uses P_b to encrypt the new session key $K_{a,b}$, and finally uses its own RSA secret-inverse key P_{krb}^{-1} to sign the encrypted key.

Alice uses her daily Kerberos session-key $K_{a,krb}$ to decrypt the first part of Kerberos' reply, gaining her new session-key $K_{a,b}$. This enables her to mutually authenticate with Bob, and to send her credit-card number N_{cc} to him securely:

$$A \rightarrow B \quad : \quad \{\{A,\ L,\ K_{a,b}\}^{P_b}\ \}^{P_{krb}^{-1}}, C_{krb}, \{A,\ N_{cc},\ time'\ \}^{K_{a,b}} \qquad (7)$$

$$B \rightarrow A \quad : \quad \{B,\ time' + 1\ \}^{K_{a,b}} \qquad (8)$$

When Bob receives the doubly-encrypted session key from Alice, Kerberos' outer signature assures him that it was Kerberos who performed the inner encryption, so Bob believes that only Alice has seen the key $K_{a,b}$. Bob then uses $K_{a,b}$ to decrypt Alice's timestamped charge-number, and returns the timestamp to her.

Bob trusts Alice's Kerberos server KRB to identify her truthfully. For this arrangement to work, each Kerberos server's certificate-chain will have to include a signed authorization certificate, identifying the server as a legitimate issuer of session-keys. Bob checks this certificate for authenticity when he checks the rest of KRB's certificate-chain.

The Kerberos service KRB does not access the key database when it answers Alice's request for Web tickets. Thus, KRB can actually be an ancillary, diskless key-server, which would be located close to Alice. For example, her online-access provider would probably put such a Web key-service in each of its local dialin offices. This gives Alice better response-time, and relieves her central Kerberos server of a substantial work-load. Alice would get her ticket $T_{a,krb}$ and session-key $K_{a,krb}$ from Kerberos in the usual way.

5 Discussion

By relieving clients of the need for public-key certificates, this protocol lowers users' entry-level costs for Web commerce. Since most home customers will probably use their online-access providers' browsers, this really means that the OA providers won't have to pass per-client fees for RSA certificates and licensing on to their customers.

Our protocol concentrates all RSA encryption operations at the servers. The Web-ticket servers are easily replicated, so this concentration does not create a bandwidth bottleneck. This means that Alice's CPU waits idly for her server to perform these slow encryptions. She sees about the same real-time delay as she would have if she had done the RSA operations herself, but her CPU is free for other tasks, such as rendering. This makes better use of everyone's computational resources, especially if the servers run with RSA encryption hardware.

Another advantage of our proposal is that it is the Kerberos server who verifies Bob's certificate with the top-level CA's public key. This is better than having Alice check the certificate, because Alice receives her copy of P_{ca} in her browser's executable. With casual freeware distribution of browsers, Alice is unlikely to know whether her browser's P_{ca} has been altered by an attacker. If it has, Alice is vulnerable to a "man-in-the-middle" attack that collects credit-card numbers. (When Alice requests Bob's certificate, the attacker intercepts it, and uses it to set up a secure connection with Bob, in Alice's name. Meanwhile, the attacker sends Alice a forged certificate C_b for Bob, signed with the fraudulent CA secret-inverse key P_{ca}^{-1}. Thus, when Alice sets up her connection, she believes that she's corresponding securely with Bob. The attacker forwards all of Alice's and Bob's messages faithfully, but records Alice's credit-card number for later use.) We suggest that it is natural for users to rely on a trusted third-party like Kerberos to manage the top-level CA key, so as to block this attack. Similarly, the Kerberos server (or the Web-ticket server) can manage the public-key Certificate Revocation List for Alice.

Our proposal does present a minor difficulty: MIT's current Kerberos version 5.5 β source-distribution [6] is not conveniently able to express the encryption-type of Alice's "hybrid" ticket. The MIT code currently assumes that a ticket has one encryption-type, which represents both the algorithm for decrypting the ticket, and the algorithm for which the ticket's session-key is intended. This is not a major problem, though, because this portion of the spec is too vague for other reasons, and needs to be reworked [14].

6 Conclusion

Our proposal judiciously uses the strengths of symmetric-key and public-key cryptography to solve the problems for which they are best suited. Most

[2] This hybrid use of Kerberos with RSA and a symmetric-key algorithm was proposed in [3].

Web users will get their Internet access from large corporate networks, so a natural corollary of their centralized account-admin is centralized key-distribution such as Kerberos provides. Web providers will be spread more thinly, so it makes sense that they should use public-key credentials. The proposal concentrates public-key encryptions at the servers, relieving clients of this substantial performance burden, and also relieving them of the responsibility for verifying certificates. Our proposal is compatible with the current specifications of the Kerberos system and the S-HTTP protocol.

7 Acknowledgements

Dan Geer and Jon Kamens at OpenVision, Win Treese at OpenMarket, Ted Ts'o at MIT, and the USENIX referees contributed helpful comments. Thank you all.

References

[1] D. Davis and D. Geer, "Kerberos Security With Clocks Adrift," *Proc. 5^{th} USENIX Security Symposium* (summer 1995).

[2] D. Davis and R. Swick, "Workstation Services and Kerberos Authentication at Project Athena," *LCS Technical Memorandum* **TM-424**, MIT Lab. for Comp. Sci. (February 1990).

[3] D. Davis and R. Swick, "Network Security via Private-Key Certificates," *USENIX 3^{rd} Security Symposium Proceedings*, (Baltimore; Sept. '92). Also in *ACM Operating Systems Review*, v.24, **4** (Oct. 1990).

[4] International Telegraph and Telephone Consultative Committee (CCITT). Recommendation X.509: The Directory - Authentication Framework. In *Data Communications Network Directory, Recommendations X.500-X.521*, pp. 48-81. Vol. 8, Fascicle 8.8 of *CCITT Blue Book*. Geneva: International Telecommunication Union, 1989.

[5] S.P. Miller, B.C. Neuman, J.I. Schiller, and J.H. Saltzer, *Project Athena Technical Plan*, Sec. E.2.1: "Kerberos Authentication and Authorization System," (Cambridge, Mass.) M.I.T. Project Athena internal document, Dec. 21, 1987.

[6] M.I.T. Information Systems, anonymous ftp distibution site for Kerberos software [athena-dist.mit.edu:pub/kerberos].

[7] Netscape Communications, "Secure Socket Layer Reference Document," Unofficial Internet Draft.

[8] C. Neuman and J. Kohl, *The Kerberos Network Authentication Service (V5)*, Internet RFC 1510, September 1993.

[9] C. Neuman, B. Tung, and J. Wray, "Public Key Cryptography for Initial Authentication in Kerberos," Internet Draft RFC, updates RFC 1510 (expires Sept. 1995).

[10] E. Rescorla and A. Schiffman, "Secure Hypertext Transfer Protocol," Internet Draft RFC (May '95).

[11] R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM*, v. 21, **2**, Feb. '78, pp. 120-126.

[12] J.I. Schiller, D. Atkins, "Scaling the Web of Trust: Combining Kerberos and PGP to Provide Large Scale Authentication," *USENIX Winter Conference Proceedings*, January 1995.

[13] J.G. Steiner, C. Neuman, and J.I. Schiller, "Kerberos: An Authentication Service for Open Network Systems", *USENIX Winter Conference Proceedings*, February 1988. [athena-dist.mit.edu:pub/kerberos/doc/usenix.PS]

[14] Ted Ts'o of MIT Information Systems, personal communication.

[15] P. Zimmermann, *Official PGP User's Guide*, Cambridge, Mass.:MIT Press, 1995.