Network Security via Private-Key Certificates

Don Davis and Ralph Swick

MIT Project Athena

Abstract

We present some practical security protocols that use private-key encryption in the public-key style. Our system combines a new notion of *private-key certificates*, a simple key-translation protocol, and key-distribution. These certificates can be administered and used much as public-key certificates are, so that users can communicate securely while sharing neither an encryption key nor a network connection.

Suppose as usual that Alice and Bob want to communicate securely. Conventional private-key authentication requires that they share a secret key, but if instead each shares a key with a translator Tom, Alice and Bob can avoid sharing directly by using Tom as an intermediary [9, 11, 5, 2]. Bob writes a message for Alice, but encrypts it for Tom's eyes only; when Alice wants to read this message, she asks Tom to translate its encryption into her key. Tom is trusted not only to keep the message secret, but also to sign the message as Bob's proxy:

1. $B \longrightarrow A : \{A, msg\}_{K_{b,t}}$
2. $A \longrightarrow T : \{A, msg\}_{K_{b,t}}, B$ (1)
3. $T \longrightarrow A : \{msg, B\}_{K_{a,t}}$

Here, the key $K_{b,t}$ is known only to Bob and Tom. Tom receives and decrypts a message addressed to Alice; before re-encrypting this with Alice's key $K_{a,t}$, he replaces Alice's name as the addressee with Bob's name. Alice will read this as proof of Bob's authorship. Alice and Bob can use an encrypted timestamp to protect against replay.

We now describe an economical way of scaling up such a key-translation service. So far, we haven't described how the translator Tom knows his clients' keys, but we've implicitly assumed that he keeps them in a database. This is impractical in the large, because it's difficult and risky to replicate the database for duplicate translators. So, we disperse the database by publishing the keys under a master-key's encrypted protection. It now falls to Alice to provide Bob's key and her own to Tom:

1. $B \longrightarrow A$: $\{A, msg\}_{K_{b,t}}$
2. $A \longrightarrow T$: $\{A, msg\}_{K_{b,t}}$, $\{K_{b,t}, B, L_b\}_{K_t}$, $\{K_{a,t}, A, L_a\}_{K_t}$ (2)
3. $T \longrightarrow A$: $\{msg, B\}_{K_{a,t}}$

The key K_t is Tom's master-key, and is known only to him and his clones. The encrypted key $\{K_{b,t}, B, L_b\}_{K_t}$ is Bob's *private-key certificate* [4, 1]; it is a published message from Tom to himself, reminding him that $K_{b,t}$ is Bob's key during the certificate's lifetime L_b.

It isn't actually necessary for Alice, the receiver, to request the translation; Bob or any third party with access to the certificate directory can make the request. The translator just refuses to do the translation if the message isn't addressed to the target certificate's owner. The protocol that we've presented here is preferable, though, if Bob addresses the message to a list of recipients. Indeed, if Bob addresses the message to "Public," Tom can relay Bob's signature without enforcing secrecy.

Clearly, each of these two protocols affords Bob a chosen-plaintext attack on Alice's long-term key $K_{a,t}$. To block this, Alice can use a short-lived key $K'_{a,t}$ to request the message's translation. Similarly, Bob will want to avoid the cryptographic exposure of using his long-lived key $K_{b,t}$ in bulky encryptions. If Bob wants to use a single-use key $K'_{b,t}$ to encrypt his message, he should request a key-lifetime L'_b that survives the span between his encryption and Alice's translation-request.

Key Distribution

Private-key certificates support a natural key-distribution protocol similar to that used in the Kerberos Authentication System [10, 12]:

1. $B \longrightarrow T$: $\{K_{b,t}, B, L_b\}_{K_t}$ (3)
2. $T \longrightarrow B$: $\{K'_{b,t}, B, L'_b\}_{K_t}$, $\{K'_{b,t}, \mathit{Tom}, L'_b, \mathit{checksum}\}_{K_{b,t}}$

Along with a new certificate containing a fresh key, Bob receives a separate copy of the same key and a checksum of the new certificate, encrypted in his old key $K_{b,t}$. By computing the same checksum himself and comparing, Bob can ensure that it was Tom who encrypted the certificate. Even without the checksum, Tom would detect a substituted certificate later anyway.

Sally, the system administrator, uses a variant of this protocol to give a new user his first key and certificate:

1. $S \longrightarrow T$: $\{K_{s,t}, S, L_s\}_{K_t}$, B
2. $T \longrightarrow S$: $\{K_{b,t}, B, L_b\}_{K_t}$, $\{K_{b,t}, B, L_b, \mathit{checksum}\}_{K_{s,t}}$ (4)
3. $S \longrightarrow B$: $\{K_{b,t}, B, L_b\}_{K_t}$, $\{K_{b,t}, \mathit{Tom}, L_b, \mathit{checksum}\}_{K_b^0}$

In Tom's response, Sally checks that the new key is addressed to Bob, and replaces Bob's name with Tom's. Then, Sally re-encrypts the key under Bob's initial password K_b^0, which he must provide personally (this is the only out-of-band communication that the system needs). When Bob gets his certificate and key, he checks Tom's timestamp and checksum, publishes the certificate in the public directory, and reencrypts the key with a new password.

Still another variant of the key-service protocol allows Alice and Bob to share a key [4, 8]:

1. $A \longrightarrow T$: $\{K_{a,t}, A, L_a\}_{K_t}$, $\{K_{b,t}, B, L_b\}_{K_t}$
2. $T \longrightarrow A$: $\{K_{a,b}, A, L_{a,b}\}_{K_{b,t}}$, $\{K_{a,b}, B, L_{a,b}, \mathit{checksum}\}_{K_{a,t}}$ (5)
3. $A \longrightarrow B$: $\{\mathit{msg}, A\}_{K_{a,b}}$, $\{K_{a,b}, A, L_{a,b}\}_{K_{b,t}}$

As before, Alice should timestamp her message. This protocol readily generalizes to allow Alice to share a key with Bob, Carl, ... , and Zack; she presents Tom with N private-key certificates, and receives N copies of the new key $K_{a,\ldots,z}$.

In all three key-service protocols, just as in the translation protocols, Tom doesn't care who presents the certificates in a request. Alice, Bob, and Sally's privacy is protected by Tom's use of their keys, so it's profitless to replay their requests.

Communication Between Translators

Suppose Alice and Bob are distant pen-pals, and that Tina is Alice's translator. If Bob wants to send a message to Alice, he needs a certificate that Tina can read; we present here two varieties of hierarchical key-distribution.

If it's necessary to avoid public-key encryption, a higher-level server Cathy can issue private-key certificates for the translators Tina and Tom. Before Bob communicates with Alice, he gets a new certificate in Tina's master-key:

1. $B \longrightarrow C$: $\{K_{b,tom}, B, L_b\}_{K_{tom}}$, $\{K_{tom}, Tom, L_{tom}\}_{K_c}$, $\{K_{tina}, Tina, L_{tina}\}_{K_c}$ (6)
2. $C \longrightarrow B$: $\{K_{b,tina}, B, L'_b\}_{K_{tina}}$, $\{K_{b,tina}, Tina, L'_b, checksum\}_{K_{b,tom}}$

If Tom were willing for Tina to share Bob's key $K_{b,tom}$, Cathy could instead translate Bob's certificate from Tom's key to Tina's, returning to Bob the certificate $\{K_{b,tom}, B, L_b\}_{K_{tina}}$. This has the advantage of simulating the hierarchy of certificate signatures specified by the CCITT's X.509 proposal [7], though X.509's elegance is admittedly lost.

If we do use public-key encryption to connect translators, then Tina holds a public key P_{tina} and its secret inverse P_{tina}^{-1}, in addition to her master-key K_{tina}, and Tom has such a public-key pair, too. Bob can use Tina's public-key certificate to request Tom's key-service, and Tom can give Bob a private-key certificate for use in Tina's domain:

1. $B \longrightarrow Tom$: $\{K_{b,tom}, B, L_b\}_{K_{tom}}$, $\{P_{tina}, Tina, L_{tina}\}_{P_{ca}^{-1}}$ (7)
2. $Tom \longrightarrow B$: $\{\{K_{b,tina}, B, L'_b\}_{P_{tom}^{-1}}\}_{P_{tina}}$, $\{K_{b,tina}, Tina, L'_b, checksum\}_{K_{b,tom}}$

For simplicity, we assume that Tom and Tina share the public-key certification authority CA. Whether the hierarchy uses public keys or private, Bob can now talk to a remote Alice via either a shared key or Tina's translation service.

Once Tom can handle public-key certificates, he can also translate between private-key and public-key messages, so that Bob and Alice can communicate with any public-key user X, and not just with translators:

1. $B \longrightarrow Tom$: $\{X, msg\}_{K_{b,tom}}$, $\{K_{b,tom}, B, L_b\}_{K_{tom}}$, $\{P_x, X, L_x\}_{P_{ca}^{-1}}$
2. $Tom \longrightarrow B$: $\{\{msg, B\}_{P_{tom}^{-1}}\}_{P_x}$ (8)
3. $B \longrightarrow X$: $\{\{msg, B\}_{P_{tom}^{-1}}\}_{P_x}$, $\{P_{tom}, Tom, L_{tom}\}_{P_{ca}^{-1}}$, $\{$"Tom speaks for Bob", $L\}_{P_{ca}^{-1}}$

Similarly, Tom can translate public-key messages from X as well. Because we have Tom encrypt his signature under X's public key, rather than sign the encryption, the public-key message $\{\{msg\}_{P_{tom}^{-1}}\}_{P_x}$ does *not* comply with X.509. However, X.509's "exposed" signatures have been shown to be insecure as specified [3], and hidden signatures offer the simplest fix. A secure exposed signature would work here, too.

Thus, both protocols, translation and key-service, extend to allow Alice or Bob to communicate with anyone, anywhere, who holds a public- or private-key certificate.

Conclusion

Private-key certificates support a full-function authentication system. The translation protocol gives us a group-access digital signature and a nice congruence with public-key protocols, including X.509. Besides server replication, the system allows certificate revocation and centralized encryption hardware. These benefits do come at the cost of weakening public-key encryption's absolute privacy, and a stolen master-key does compromise more traffic in our system than in a public-key system. More important than these considerations, though, is that private-key certificates are compatible with many encryption algorithms; public-key *protocols* no longer rely on public-key *encryption* [6].

Acknowledgements

We thank Mark Lillibridge, Jeff Schiller, Mike Burrows, Martín Abadi, Jon Rochlis, Dan Geer, Ed Guzovsky, and Roger Needham, for their helpful suggestions and critiques.

References

[1] This use of private-key certificates was proposed to us by Martín Abadi and Mike Burrows of Digital's Systems Research Center, and by Butler Lampson of Digital's Cambridge Research Laboratory, in a personal communication.

[2] Selim G. Akl, "Digital Signatures: A Tutorial Survey," *Computer* Vol. 16(2) pp. 15-24 (Feb. 1983).

[3] Michael Burrows, Martín Abadi, and Roger Needham, "A Logic of Authentication," *Proc. R. Soc. Lond. A* 426(1989) pp. 233-271.

[4] Don Davis and Ralph Swick, "Kerberos Authentication and Workstation Services at Project Athena," *MIT Laboratory for Computer Science Technical Memorandum* 424 (Feb. 1990).

[5] Dorothy E. R. Denning, *Cryptography and Data Security.* Reading, MA: Addison-Wesley, 1983. pp. 14-15.

[6] Colin I'Anson and Chris Mitchell, "Security Defects in CCITT Recommendation X.509 - The Directory Authentication Framework," *ACM Computer Communication Review*, 20(2) pp. 30-34 (April 1990).

[7] 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.

[8] John Kohl, Clifford Neuman, and Jennifer Steiner, "Kerberos Version 5 Request for Comments," (in preparation at Project Athena, MIT bldg. E40, Cambridge MA 02139).

[9] Ralph C. Merkle, "Protocols for Public-Key Cryptosystems," pp. 122-133 in *Proc. 1980 Symp. on Security and Privacy*, IEEE Computer Society (April 1980).

[10] Roger M. Needham and Michael D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *CACM* 21(12) pp. 993-999 (Dec. 1978).

[11] G. J. Popek and C.S. Kline, "Encryption and Secure Computer Networks," *Computing Surveys* Vol. 11(4) pp. 331-356 (Dec. 1979).

[12] Jennifer Steiner, Clifford Neuman, and Jeffrey Schiller, "Kerberos: An Authentication Service for Open Network Systems," USENIX Winter Conference Proceedings, February 1988.