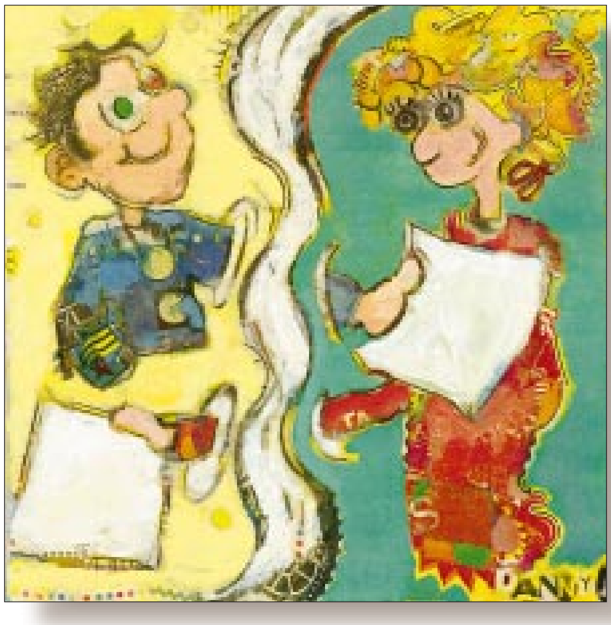
by Jim DeRoest



Certificate-Based Authentication

Jim DeRoest *has been involved (for better or worse) with IBM UNIX offerings from the IX/370 days, through PC/IX, AIX RT, AIX PS/2, AIX/370, PAIX, AIX/ESA and AIX V3. He is employed as an assistant director supporting academic and research computing at the University of Washington, and is the author of* AIX for RS/6000–System and Administration Guide *(McGraw-Hill). He plays a mean set of drums for the country gospel band Return. Email:* `deroest@cac.washington.edu`.

In last month's column ("Internet White Pages," Page 82), I discussed how Lightweight Directory Access Protocol (LDAP) might address the problem of locating users on the Internet by providing an Internet White Pages service. Let's assume that the network software vendors and service providers have actually standardized on a directory protocol like LDAP and that we can now use almost any Web browser to look up user directory information. I know this is a real stretch of the imagination, but bear with me for a page or two.

You use this directory service to locate an email address of a business associate, and now you're ready to start up an electronic conversation. The content of this conversation is somewhat confidential, so you want to make certain that you will be connecting with the intended recipient. Unfortunately, you've never met this person face-to-face and you don't have any other means for contacting them out of band. How are you going to authenticate the recipient? For that matter, how are they going to verify that you're who you say you are?

Many of us play this game each time we enter a credit card number into a form on a Web page. How do you know the link is secure? Do you always verify that the blue bar or key is displayed on your browser before typing in those critical account numbers? Do you read all the connection and certificate warnings that are displayed, or do you click "OK" without a second thought? Even if you're certain that you've contacted the correct site, are you sure there isn't some rascal sitting between you and the service provider listening in or even modifying the transaction? It's another case of technology taken for granted. Hey, those Netscape and Microsoft guys are real sharp characters, right? We can trust them with our money and reputations. I don't know about you, but I like to know exactly how the bits are being shuffled before I let someone tap into my meager bank account.

Certificates

The most common form of trusted authentication between parties in the wide world of Web commerce is the exchange of certificates. A certificate is a digital document that at a minimum includes a

Distinguished Name (DN) and an associated public key. The certificate is digitally signed by a trusted third party known as the Certificate Authority (CA). The CA vouches for the authenticity of the certificate holder. Each principal in the transaction presents a certificate as its credentials. The recipient then validates the certificate's signature against its cache of known and trusted CA certificates. A "personal certificate" identifies an end user in a transaction; a "server certificate" identifies the service provider.

Generally, certificate formats follow the X.509 standard. X.509 is part of the Open Systems Interconnect (OSI) X.500 specification discussed in last month's column on LDAP. Fields within an X.509 certificate are defined using Abstract Syntax Notation 1 (ASN.1). Architecture independence is facilitated by encoding a certificate using Direct Encoding Rules (DER) for binary representation and Base64 for ASCII text.

The certificate system seems simple enough. However, there is a great deal of infrastructure that makes certificate-based authentication possible. Certificates must be unique and unalterable.

The certificate system seems simple enough. However, there is a great deal of infrastructure that makes certificate-based authentication possible. Besides the elements listed above (DN, public key, CA signature, common format and encoding), certificates must be unique and unalterable. The signature on which the validity of a certificate is made manifest must be backed by policies that cannot be repudiated. Let's take a closer look at the technologies that make up this infrastructure and some of the difficulties that arise as the technology is scaled to large and diverse populations.

Public Keys

Key-based encryption is fundamental to the digital signature requirements for certificates. Essentially, there are two basic key encryption schemes, symmetric and asymmetric. Symmetric algorithms use the same key to encrypt and decrypt messages. The main drawback with symmetric key systems is in securely passing a copy of the key to associates so that they may encrypt and decrypt messages when corresponding with you.

With an asymmetric system, two different keys that are mathematically related are used to encrypt and decrypt messages. A message encrypted with one key may only be decrypted by the other. The public key algorithm developed by Whitfield Diffie and Martin Hellman of Stanford University is an asymmetric system that can be used in conjunction with other cypher methodologies to digitally sign electronic documents. One of the keys, the private key, is kept secret by the owner. The second, the public key, is made available to anyone who wishes to electronically converse with the owner in a secure manner using encryption.

Using a public key system, a user may encrypt a portion of a document using his or her private key. This message will later be decrypted by the recipient using the sender's public key. Assuming that the sender's private key is secure, the recipient can assume basic authenticity of the encrypted portion of the document in that only the sender holds the private key used to encrypt the message. Thus, the encrypted portion of the message is said to have been "digitally signed" by the sender.

Digests

The digital signature system may still fall short in ensuring authenticity. A malicious third party might begin sending documents using a public/private key set that he has distributed claiming to be you. The way to defeat this type of attack is to use a function called a digest. A digest produces a hash of a random text string that is difficult to reverse.

Basically, the procedure works like this. The sender transmits a random string in plain text to the recipient, accompanied by a digest of the random string that is signed using the sender's private key. The receiver runs the same digest function on the random string producing the hashed message, and decrypts the signed digest using the sender's public key. The two digests are then compared. This procedure is simplistic but describes how digests can be used to provide additional levels of verification.

As I mentioned earlier, the job of a CA is to act as a trusted third party who will verify that the DN and public key contained in a certificate are unique and bind the certificate to a subject. Much of this trust comes from the policy statement published by the CA that identifies its realm of responsibility and the extent to which it can assure accuracy of the certificates it issues. Along with granting certificates, a CA may also revoke certificates. The CA distributes its own certificates, just as individuals do. A CA's certificate and public key are used as an authenticator to validate its signature against personal and server certificates.

Scaling Problems

Certificate-based authentication systems are not without a few problems when it comes to scaling the infrastructure. First, if you are distributing certificates to a large population of users, what mechanism will be used to authenticate each user at the time a certificate is obtained from a CA? If you've downloaded a copy of the Microsoft Corp. Internet Explorer, then it's likely that you have opted for the free personal certificate offer from VeriSign Inc. Other than prompting for DN information and a pass phrase, no attempt is made to validate who is entering data into the VeriSign Web form. Would you accept one of these personal certificates as proof of identity? I think not!

Example Digest Procedure

Sender:
random string
signed digest = private key[digest[random string]]

Receiver:
sender digest = public key [signed digest]
compare digest[random string] to sender digest

Table 1. Public Key and Certificate Information

Generalized Certificates
`http://www.clark.net/pub/cme/html/cert.html`

Netscape Certificate and SSL Information
`http://home.netscape.com/comprod/server_central/support/faq/certificate_faq.html`
`http://home.netscape.com/newsref/ssl/3-SPEC.html`

Usenix Electronic Commerce Proceedings
`http://www.usenix.org/publications/library/proceedings/`

Thawte Digital Certificate FAQ
`http://www.thawte.com/faq/certs.html`

On a small scale, it's easy enough to work individually with a community of users. You can use some out-of-band system such as face-to-face verification before granting a certificate. Larger environments with existing authentication systems such as Kerberos might require you to authenticate to Kerberos before obtaining a personal certificate.

The next problem is how to design and manage a networkwide CA hierarchy. One can certainly imagine a large-scale CA such as VeriSign or the U.S. Postal Service handling corporate- and organization-level certificate distribution. However, the costs and administrative issues of working with a large-scale CA for certificates within the enterprise are too extreme. Most organizations want to act as their own CA for local business activities. But what happens when company certificates are distributed beyond the local organization? Who will trust them? How will a hierarchy of CA levels be administered? And how will the chain of trust be negotiated and maintained?

How about anonymity and privacy in Web commerce? It's not clear that a certificate must contain a DN that includes the identity of an individual user. For example, a certificate used to authenticate purchases of commodities over the network might only need to identify an account number and a bank. The bank is then responsible for the mapping of the account number to an individual. The merchant is primarily concerned that the certificate will validate a draw on a real bank account. Thus, the certificate and CA must vouch for a particular attribute or mechanism rather than an identity. There's no reason that privacy cannot be implemented within the existing certificate infrastructure. If you're interested in generalized certificate systems, take a look at the Internet draft document "Simple Public Key Certificate" by Carl Ellison, Bill Frantz and Brian Thomas (see Table 1, "Generalized Certificates").

One of the assumptions made in the existing certificate system is that the end user always works from a particular workstation. The user's certificates are assumed to be located in a browser cache on that workstation. Unfortunately, for those of us who move around between computers or work for universities and libraries with public workstations, this just won't do. How do you carry your certificates along with you as you move from system to system?

What happens when company certificates are distributed beyond the local organization? Who will trust them? And how will the chain of trust be negotiated and maintained?

Developers are already talking about storing public key data in directory servers like LDAP. How about adding support for storing certificates as well? Certificates could be encrypted, and an authentication mechanism like Kerberos could be used to authenticate access. Once authenticated, the certificates could be downloaded to the local workstation and decrypted for use. This architecture would require that Web browser developers support merging or replacing the certificate cache on a workstation. The cache might also need to be wiped clean once a browser session has ended.

A certificate escrow infrastructure might also solve the problem of automating certificate revocation by removing certificates from the directory. Certificates that have long life spans will occasionally need to be revoked. One can always argue that you should limit the life span of a certificate such that revocation is not an issue. I'm not convinced that this argument extends into real-world use. Models that might include certificate authentication for things such as electronic mail lend themselves toward long life spans. It's difficult to globally revoke a certificate whose only home is an individual's workstation. Certificate Revocation Lists (CRLs) have to be distributed to every service that may need to validate the certificate. This reminds me of the old checking account lists that department store clerks had to examine before accepting a check.

It seems like there are a large number of issues that must be resolved, yet here we are using certificates for authentication in day-to-day Web commerce. Demand never seems to wait for elegant technical solutions.