Notes.net

Iris Today


by
Chuck
Connell

**Level:** Intermediate
**Works with:** Domino 5.0
**Updated:** 12/04/2001

When you send e-mail from one Notes client to another via Domino servers, you probably don't think much about the security of that message. After all, Notes and Domino have always had security features to keep data safe from prying eyes. But what happens when you send e-mail to someone with a different e-mail client, such as Outlook, or via a different e-mail server? Can that message be as secure as Notes mail? As communication between disparate organizations proliferates around the world, the answer to that question becomes increasingly important. Part of that answer is S/MIME.

S/MIME stands for Secure Multipurpose Internet Mail Extension. S/MIME is a security-conscious e-mail standard based on an earlier nonsecure e-mail standard called MIME. (See the **What is MIME?** sidebar for more information.)

This article provides a broad introduction to S/MIME and how it is used within Notes and Domino. The article begins by looking at why S/MIME matters at all and the goals it is designed to accomplish. It presents some general background about cryptography, including public key cryptography, and explains how it can be used for secret and authenticated messages. The article applies these general principles to S/MIME and shows how S/MIME works, including an explanation of x.509 certificates and the role they play in verifying people's identities within S/MIME. Finally, it takes a look at where S/MIME fits into Domino and Notes messaging, with specific instructions for using S/MIME with Domino, Notes, and standard e-mail programs.

## Secrecy and authentication

Data security software is often complicated, with many confusing acronyms to remember and a new buzzword every month. So an important question to ask about every security method thrown your way is, "Why should I care about this?"—or to state it another way, "What significant capability does this technology give me that I didn't have before?"

What does S/MIME give us? It gives us two things vital to important e-mail traffic: secrecy and authentication.

*Secrecy* means that only the intended recipient of an e-mail message can read the message's content. It is the e-mail equivalent of putting a paper message into a thick envelope (so no one can hold it up to the light), sealing the envelope tightly, and delivering the message by trustworthy courier. *Authentication* means that the recipient of an e-mail message knows for certain the message really came from the person it appears to come from. It is the e-mail equivalent of a reader checking the signature at the bottom of a paper message and being confident they can detect a forged signature.

Closely related to authentication is the concept of message *integrity*. When you receive an e-mail with proof the message actually came from the sender, you also want to know that no one tampered with the message en route. Without integrity (or lack of tampering), authentication is nearly worthless. There is little point in knowing an e-mail certainly came from the

apparent sender, while being unsure whether someone changed the message—and possibly it's meaning—after it was sent.

Prior to S/MIME and similar technologies, Internet e-mail users did not have secrecy or authentication. E-mail messages were transmitted through the Internet in plain text format, so snoopy system managers or malicious eavesdroppers could read e-mail along its way from sender to recipient. Without S/MIME it also is simple to send e-mail that appears to be from anyone you want. (This is easy to see by experimenting with the Outlook Express settings in Tools - Accounts - Mail - Properties.)

## Cryptography primer

S/MIME accomplishes secrecy and authentication by using a combination of public key cryptography and more standard cryptographic techniques. Understanding these concepts will help you understand how S/MIME works.

The traditional method for creating a secret message, dating back thousands of years, is to scramble the message by using a *secret key*. The key can be almost anything at all, including words, sets of numbers, or text from a book. The key is merged into the message in such a way that the message is unreadable, unless the message is unscrambled using the same key. There are many ways to merge a key into a message, some quite simple and some extremely complex. In all cases though, the basic idea is the same. Two people can send an encrypted message to each other, and no one else can read the message as long as only the sender and receiver know the secret key. (Of course, this is the goal. Some symmetric key systems do it better than others and none are perfect.) This method is also knows as a *symmetric cipher*, because both the sender and receiver use the same key.

During the 1970s, a new kind of encryption scheme was invented that was radically different—*public key cryptography* (PKC). A more accurate name for PKC, however, would be public/private key cryptography because the method uses two keys. In this case, the keys are large numbers related to large prime numbers. One of the keys is freely distributed to anyone who wants it, and the other key is kept private to the person associated with this key pair. Either key can be used to encrypt (make secret) a message, and the other key is used to decrypt (recover) the message. Because of this fact (encrypt with one key, decrypt with the other key), PKC is also known as an *asymmetric cipher*. The two different keys give us the ability to use PKC for secrecy, or authentication, or both, as the following examples show.

**Public key example 1: Secrecy only**

Suppose I want to send a secret message to Katie. Suppose my message is "Let's have breakfast at Dunkin Donuts." I encrypt the message with Katie's public key (which I must have access to), yielding something unreadable like "4dRf7H4rt7dUfd3h58nGcFu7." I send the secret message to Katie by any convenient method. After receiving the message, Katie plugs the secret message and her private key into the decryption algorithm and recovers my original invitation. No one else can do this, because only the private key of Katie's key pair can decrypt the message. The message also cannot be decrypted with Katie's public key, even though that key was used to encrypt the message. (Again, this is the goal. No public key cipher is perfect.)

**Public key example 2: Authentication only**

Suppose I want to send the same message to Katie, but this time I am more concerned with authentication than privacy. I don't mind if someone else reads the message, but I want Katie to know for certain that the message really came from me. In this case, I encrypt the message with *my private* key, yielding an unreadable message. I send *both* messages to

Katie, the readable invitation and the unreadable version I made with my private key. When Katie receives the pair of messages, she uses my public key (which she must have access to) to decrypt the unreadable message. She then compares the decoded result to the readable invitation. If the two messages match, then Katie knows I must have sent her the invitation and no one tampered with either part of the pair. No one else has my private key, so no one else could have encrypted the invitation in such a way that it could be restored with my public key. If anyone tampered with either part of the message pair, the pair would not match at the end.

**Public key example 3: Secrecy and authentication**
Suppose I want to send the same message to Katie, but this time I want it to be both secret and authenticated. This is actually easy, given the techniques used above. I simply encrypt the message with my private key and pair it with the original invitation, just as in example 2. Before sending the message pair to Katie, however, I encrypt the whole package with Katie's public key. When Katie receives the package, she first decrypts the message pair with her private key, and then authenticates my message just as before. No one other than Katie can decrypt the pair of messages, and she then can verify no one other than me could have sent them.

**Additional concepts**
Another important cryptography concept is *hashing*. Hashing is a general term for making a shortened version of a message that is difficult to restore to its original form. For example, a hash of the message "Let's have breakfast at Dunkin Donuts" might be "dT5hDu." It is important there not be a simple method to recover the original message from the hash, otherwise the hashing algorithm is not good. Since hashing, in effect, creates a short form of a message, the hashed message is often known as a *message digest*.

Finally, let's look at the concept of a *public key certificate*. A public key certificate is a way to establish a reliable link between a person's name and a given public key. Without such a mechanism, anyone can send an authenticated e-mail and claim it came from someone else. I could create a public/private key pair for the name Yolanda Yodalacky, post the public key for all to use, sign an e-mail with the private key, and convince anyone that Yolanda is sending them mail. Public key certificates solve this problem (to a large extent; nothing is perfect). The most common format for public key certificates is x.509, and this format is used by S/MIME.

An x.509 certificate contains the following information about a person:
- Person's name
- Public key for this person
- Name of a certificate authority
- Digital signature of the certificate authority
- Expiration date
- Other assorted information

The *certificate authority* is an organization we trust to link names and public keys. When a certificate authority says a given public key goes with a given name, we believe it.

But there is still a problem. How do we know a particular x.509 certificate is actually from a trusted certificate authority and not just pretending to be? We check the validity of the x.509 certificate by decrypting the digital signature of the certificate authority. And how do we do this? With the certificate authority's public key, which is public information.

For x.509 certificates to work practically, the value of the certificate authorities' public keys must be both public and trusted. Netscape and Internet Explorer are delivered with a set of trusted certificate authority names and public keys. Users also can add other certificate authorities that

they trust to these built-in lists. Notes/Domino users and administrators declare trust in certificate authority public keys by creating "Internet cross certificates" and storing them in users' personal address books.

## How S/MIME works

S/MIME combines traditional symmetric ciphers, public key cryptography, hashing, and public key certificates. The reason these various methods are combined within S/MIME is efficiency. By using the strength of each method in the appropriate place, the result is a reasonably secure scheme that still runs quickly in practical use.

The most common symmetric ciphers used in S/MIME are RC2 and TripleDES. The usual public key method is RSA, and the hashing algorithm is SHA-1 or MD5. The mathematical details of these algorithms can be found in the links at the end of this article. The innards of each specific cryptography technique used within S/MIME, however, are not required to understand its general operation.

Here's how S/MIME puts these techniques together.

**S/MIME example 1: Secrecy only**

Suppose I want to send a message to Katie and be assured no one else can read it. My e-mail program and Katie's take the following steps:

1. My e-mail program creates a random key that will be used in the symmetric cipher. This key is known as the *session key*, since it is used just for this e-mail session.
2. My e-mail program encrypts the message with the symmetric cipher, using the session key.
3. My e-mail program encrypts the session key with public key cryptography, using Katie's public key.
4. My e-mail program creates a package of data that includes the encrypted message, the encrypted session key, my x.509 certificate, and identification of the encryption algorithms used.
5. The package of data is sent to Katie. This is an S/MIME e-mail message.
6. When Katie's e-mail program receives the message, it uses Katie's private key to decrypt the session key.
7. Using the session key (and the information about the symmetric cipher) Katie's e-mail program decrypts the message.

In S/MIME terminology, secret messages are sometimes known as *enveloped data*, or are said to have a *digital envelope*.

**S/MIME example 2: Authentication only**

Suppose I want to send a message to Katie and prove to her I am the sender. My e-mail program and Katie's take the following steps:

1. My e-mail program creates a digest of the message, using a hashing function.
2. My e-mail program encrypts the message digest with public key cryptography, using my private key.
3. My e-mail program creates a package of data that includes the original message, the encrypted message digest, my x.509 certificate, and identification of the encryption algorithms used.
4. The package of data is sent to Katie. This is an S/MIME e-mail message.
5. When Katie's e-mail program receives the message, it verifies that my x.509 certificate is valid and retrieves my public key from the certificate.
6. Katie's e-mail program uses my public key to decrypt the message digest.
7. Katie's e-mail program uses the information about the hashing function to independently compute the message digest of the original message.
8. Katie's e-mail program compares the decrypted message digest (from me) with the message digest it computed. If the two digests match,

Katie can trust the message was not tampered with.

In S/MIME terminology, authentication is sometimes known as *signing* or having a *digital signature*.

**S/MIME example 3: Secrecy and authentication**
To send a message that is both secret and authenticated, the S/MIME techniques shown above simply are nested. First the message is authenticated, then the authenticated package is made secret, and then the secret package is sent to the recipient. The recipient of the message unwraps the package by using his or her private key to decrypt the session key, then decrypts the rest of the package with the session key. After decrypting, the remaining data is a signed S/MIME message, which is authenticated as outlined above.

## S/MIME within Domino and Notes

After all this background about S/MIME, you might think S/MIME plays a key role in standard Notes e-mail. It does not. Standard Notes e-mail (between two users who have the Notes client software and are connected to a Domino server) does not need to use S/MIME for secrecy and authentication. Native features, built into the Notes client software and Notes ID files, already provide these options using techniques similar to S/MIME. For secrecy and authentication within native Notes e-mail, just open the Delivery Options dialog box while composing a Notes mail message. The Sign option provides authentication and the Encrypt option provides secrecy.

S/MIME is still crucial to Notes/Domino customers, however, because it is common for Notes users to send and receive e-mail with people who are not in a pure Notes environment. Many people, after all, use Outlook, Outlook Express, Netscape Messenger, Groupwise, or other programs for e-mail. (All four of these mentioned programs support S/MIME.) Also, without S/MIME, people in your own organization cannot use secrecy or authentication when they access their Domino mail from a non-Notes e-mail program.

There are generally three situations when Notes/Domino customers are not using pure Notes e-mail:

- When sending or receiving e-mail with Notes, through a Domino server, to someone who does not have Notes as his or her e-mail program.
- When sending or receiving e-mail with Notes, through a non-Domino e-mail server, to anyone else. In this case, you are using Notes as an Internet e-mail client program and ignoring the Notes e-mail capabilities. (You are using Notes as a POP3 or IMAP client and SMTP sender in this case.)
- Sending or receiving e-mail with a non-Notes e-mail program, through a Domino server, to anyone else. (You are using Domino as a POP3 or IMAP server and an SMTP relay in this case.)

## Using S/MIME

Up to this point in the article, I have focused on the background for S/MIME and how it works under the covers. Using S/MIME on a day-to-day basis is considerably easier than understanding it, however. (Good security mechanisms are designed this way.) The remainder of the article, therefore, puts away the theory and explains how to use S/MIME to send and receive e-mail that is secret and/or authenticated.

**Getting a digital ID**
The first step in using S/MIME is to get a *digital identification*. A digital identification is a public/private key pair, a name, and a certificate that attests to the validity of the public key for this name.

The two most popular vendors of digital identifications are **VeriSign** and **Thawte**. A VeriSign ID costs $15 per year and is simple to get and use. A Thawte ID is free but requires you to take additional steps to actually get your name attached to the ID. Business users who want to get up and running quickly probably will prefer VeriSign.

One thing to be aware of with both vendors: for security reasons, you must follow the online instructions carefully. Do exactly what they tell you to do, including doing all operations from one computer.

In addition, Domino administrators can use the Domino Certificate Authority application to automatically issue x.509 certificates to all existing Notes users. This process is transparent to end users within the domain and allows them to use S/MIME without acquiring digital IDs on their own. After setting up the Certificate Authority application, the Domino administrator selects Person records from the Domino Directory and chooses Actions - Add Internet Cert to Selected People. The administration process then issues an Internet certificate for each user based on the public key stored in the Person record. When the user next authenticates with their home server, the certificate is automatically added to the user's ID file.

Using Domino as its own certificate authority requires that e-mail recipients accept your Domino Certificate Authority as a trusted root, because it is not one of the external trusted roots that are pre-installed in their e-mail programs. For more information about the Domino Certificate Authority, see the *Iris Today* article "**Trust yourself: Become your own certification authority**."

**Using S/MIME with Domino**
Domino R5 handles MIME (and therefore S/MIME) message content natively. There is nothing you have to do to enable S/MIME messages to pass through a Domino server.

**Using S/MIME with general e-mail clients**
If you are using all-Microsoft software and acquire a digital ID on that computer, your private key and public key certificate are automatically installed correctly. They become integrated with Internet Explorer and Outlook/Outlook Express. After acquiring the digital ID, you can easily see it:

1. In Windows, click the Start menu and choose Settings - Control Panel.
2. Open Internet Options and click the Content tab.
3. Click the Certificates button to see the list.

To use your digital ID, just press the buttons for Sign (authenticate), or Encrypt (secret), or both, when composing an e-mail message. When you receive a signed message, you will see a symbol indicating this, near where the paperclip appears for attachments.

If you are using Netscape Messenger, Groupwise, or other e-mail software, the details for installing and using the digital ID may vary, but the general principles are the same.

**Using S/MIME with Notes**
For these instructions, I assume you already have installed a digital ID on a Windows computer using Internet Explorer and want to use that digital ID with Lotus Notes on the same computer. There are four general steps:

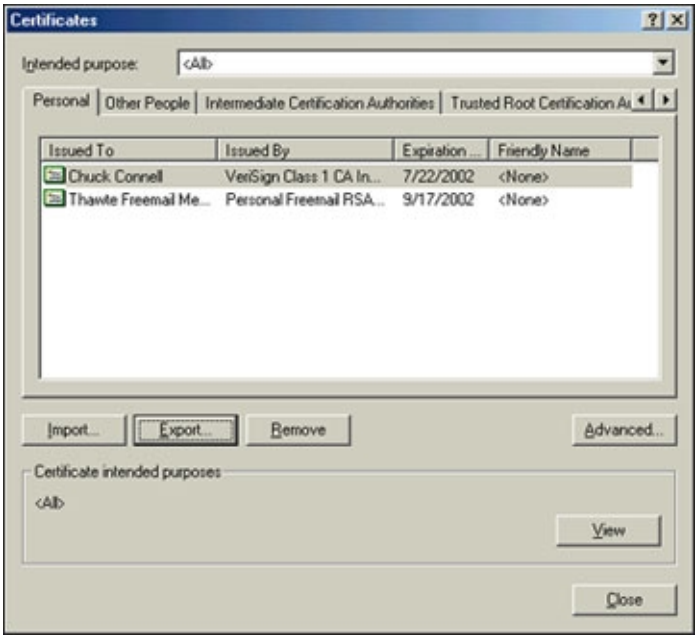
1. Export the digital ID from Windows.
2. Import the digital ID to your Notes ID file.
3. Make sure this certificate will be used for Internet mail from Notes.
4. Use the digital ID as you send and receive e-mail from Notes.

This is simpler than it sounds, since the first three steps only have to be done once.

If your situation is different—exporting on a non-Windows computer for example—the basic idea is still the same.

To export the digital ID from Windows:

1. In Windows, click the Start menu and choose Settings - Control Panel.
2. Open Internet Options and click the Content tab.
3. Click the Certificates button and select the certificate (digital ID) you want to export to Notes.

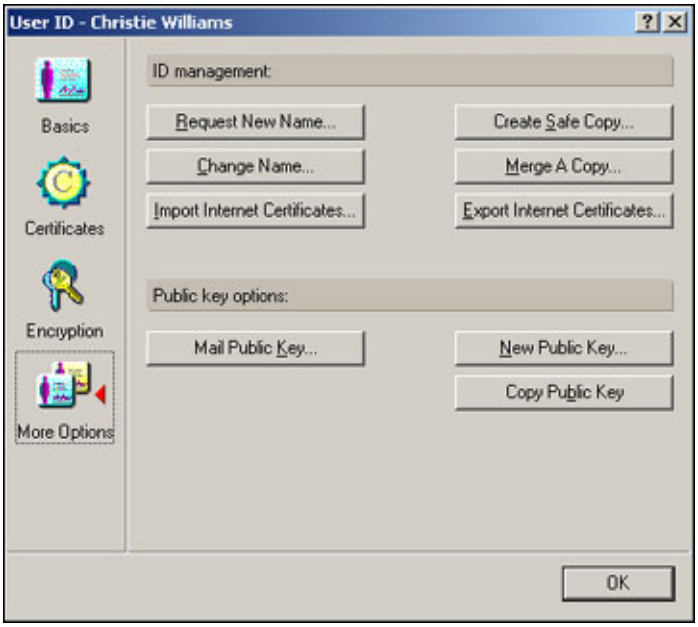



4. Click the Export button.
5. Click the Next button in the export wizard.
6. Select Yes to export the private key.
7. Select PKCS #12 as the export file format. Also select "Include all certificates" and "Enable strong protection" on this page.
8. Click the Next button and enter a password for the export file that will be created. Choose a good password, since the export file will contain your private key.
9. Enter a file name for the export file in the File name text field when requested. Something like c:\temp\mycert works fine. The .PFX extension will be added automatically.
10. Click the Next button and then confirm your choices by clicking the Finish button.
11. Click OK if you see a warning that your private key is being used.

To import the digital ID to your Notes ID file:

1. In Notes, choose File - Tools - User ID.
2. Enter your password when requested and click OK.
3. Go to the More Options panel of the User ID dialog box.



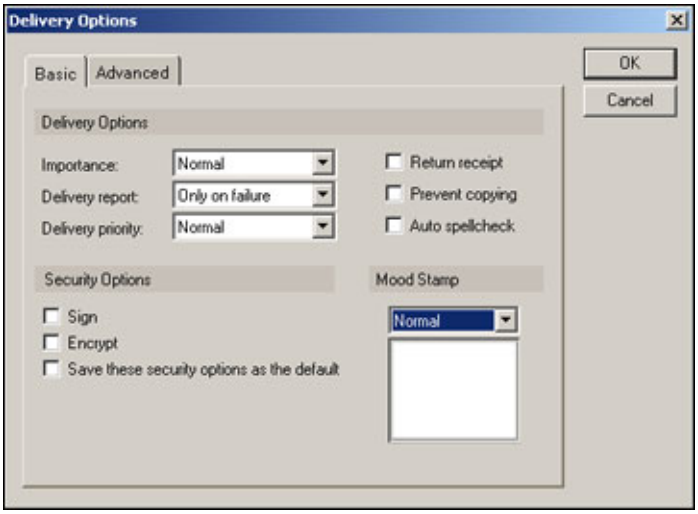

4. Click the Import Internet Certificates button.
5. In the Specify File Containing the Internet Certificates dialog box, browse to the file you exported above, select it, and click Open.
6. You will be asked for the password to the file. This is the password you chose above.
7. You will see a list of several certificates that are contained in the exported digital ID file. Click Accept All.

To make sure this certificate will be used for Internet mail from Notes:

1. Choose File - Tools - User ID.
2. Go to the Certificates panel of the User ID dialog box.
3. Scroll down in the Certificates Issued By list until you see the new certificates you just imported.
4. Select your public key certificate (not the certificate authority certificates). When you select the right certificate, your e-mail address will appear in the Certificates Issued To list.
5. Make sure that the "This is your default signing certificate" checkbox is selected.
6. Click OK.

To use the digital ID as you send and receive e-mail from Notes:

1. When composing an e-mail message, click the Delivery Options action button to open the Delivery Options dialog box.




2. On the Basics tab, select Sign (to authenticate the message) or Encrypt (to make the message secret), or both.
3. Click OK.

Keep in mind that to send an encrypted e-mail to someone who is not using Notes mail, you must have that person's public key certificate in your Domino Directory. The certificate, if present, is visible on the Certificates tab of the recipient's Person document under Internet Certificates. To get more detail about a particular certificate in a Person document:

1. Open the Person document in edit mode.
2. Click the Examine Internet Certificates action button.
3. Select the certificate you are interested in from the list of certificates in the Examine Internet Certificates dialog box. Details about the selected certificate will appear in the lower part of the dialog box.

If you want to send an encrypted message to someone using S/MIME and their Internet certificate is in their Person document in a Domino Directory to which you have access, no special steps are required. If you want to send an encrypted message to someone and you do not have their Internet certificate, ask that person to send you a signed e-mail message.
When you open the signed message, you will be prompted to cross certify. If you wish to establish trust with the certificate authority that issued their certificate in one simple step (in addition to trusting the user's certificate), you may select it from the Subject name list box. Confirmation that the message was signed will appear in the status bar. Then choose Tools - Add Sender to Address Book from the menu. The default action (on the Advanced tab) is to "Include x.509 certificates when encountered." When a Contact document is added to your personal address book, the sender's public key will be available to you and you will be able to encrypt messages to him or her.

## For further reading

Here are some additional resources:

- An excellent overview of cryptography and many popular encryption schemes can be found on the **RSA Laboratories** Web site, for example, **RSA Laboratories Frequently Asked Questions about Today's Cryptography**.
- *Cryptography and Network Security* by William Stallings, Prentice Hall, 1998, is one of the best general textbooks about these topics. The book covers mathematical theory as well as practical implementations, including lots of detail about S/MIME. The book is widely available, including online at **Amazon.com**.
- The Web site of the **S/MIME Working Group of the IETF** includes a

number of documents.

- Some information about the relationship between S/MIME and a similar method called PGP/MIME can be found on the Internet Mail Consortium's **S/MIME and OpenPGP** page.
- ***Lotus Notes and Domino R5 Security Infrastructure Revealed*** is the primary IBM Redbook about Domino R5 security. It covers a lot of material, including S/MIME.
- The *Iris Today* article "**Using field encryption in applications**" provides details about secret and public key encryption, and the article "**Security for Web-based mail: A case study**" examines one company's alternatives for securing Web-based mail.
- Also see the **Domino R5 Administration Help**. Click on the Index view, and then type "s/mime." If you read this book online, be aware that it is a big book so it might take a while to open.

**ABOUT THE AUTHOR**

Chuck Connell is president of CHC-3 Consulting and has 11 years of experience with Domino and Notes. Prior to CHC-3, Chuck worked at Lotus in several positions, including development manager for the Notes C API kit. Chuck runs the Web site **www.DominoSecurity.org**, teaches computer science at Boston University, and writes frequently on computer topics. He can be reached at **www.chc-3.com**.

]

## What is MIME?

If S/MIME is a secure version of MIME, what is MIME and what problems does it solve?

E-mail messages were originally limited to plain text. MIME was created to meet the needs of e-mail users who wanted to send information that was something other than plain text. While it is common now to send file attachments, programs, and pictures by e-mail, doing so was once considered a "wish list" item. The **Internet Engineering Task Force** (IETF) took up this problem in the early 1990s and devised methods to encode non-text e-mail content as a series of text characters. They used the existing text-based e-mail system to send non-text information, by making the information appear as if it were a text message.

For further information about MIME, here is a good **Mime Information Page**.