This article presents a primer on data encryption technology and looks at the ways it is used in Notes to protect databases, documents, field contents, mail, and traffic on a port.

Security in one form or another touches nearly every aspect of our life. We lock our homes and cars for protection. We lock documents in filing cabinets for privacy. We use standard "corded" phones as opposed to "cordless" phones to prevent eavesdropping. Some of us even use high-tech identification cards for access to our workplace. We have many ways to secure a variety of things for a multitude of reasons; however, can the same be said for our computing environment?

For most people, security in their computing environment translates to a user name and a password. While this manifestation of security in computing is critical, a user name and password is only one piece of a truly secure computing environment.

The digital world that nearly all of us participate in is similar to our real world. We have servers that need protection. We use databases and read documents that must be kept private. We transmit data, such as electronic mail, across private and public networks that must be kept safe from eavesdropping.

While padlocks are relatively useless in our computing environment, we do have a digital equivalent: encryption. Notes has provided the digital padlock of encryption ever since its first release. Through each release, Notes has continued to enhance and augment its encrypting features. This article is a guided tour through the world of Notes encryption, and it demonstrates how Notes is second-to-none when it comes to protecting data and ensuring privacy.

The ABCs of encryption

Before plunging into where and how Notes uses encryption, we need to nail down the fundamentals of encryption (a.k.a. cryptography -- the art and science of keeping information secure). Let's start with a glossary of terms:
**Plaintext**: Data ranging from a simple text message to more complex binary information, such as a Lotus 1-2-3 spreadsheet or a WAV sound file.

**Ciphertext**: The scrambled, unintelligible, version of *plaintext*.

**Encryption Algorithm**: The algorithm that transforms *plaintext* into *ciphertext*. This algorithm relies upon a *key* to uniquely scramble *plaintext*.

**Decryption Algorithm**: The algorithm that transforms *ciphertext* back into *plaintext*. This algorithm relies upon a *key* to help decrypt information. Sometimes this *key* is the same one used during encryption, and other times, this *key* is different.

**Key**: A long number used with an encryption and decryption algorithm. The key is what makes *ciphertext* a uniquely scrambled piece of information. Also, the key is what gives encryption its strength: Generally, the longer the key, the stronger the encryption.

**RC4**: A stream encryption method. Each character, or symbol, is individually and dynamically transformed into *ciphertext* (RC is believed to stand for Rivest Cipher -- The name of the guy who created this encryption method). This encryption method is often used for communication port encryption.

**RC2**: A block encryption method. Data is encrypted one block at a time versus one character at a time (RC4). RC2 is slower than RC4 but can be used in some cases where for technical reasons RC4 cannot. You'll often find this encryption method used to encrypt e-mail.

**RSA**: A public-key, block-encryption method. Data is encrypted one block at a time and decrypted with a different key (see public-key algorithm below). RSA is much slower than RC2 and RC4.

**BSAFE**: A product from RSA Data Security, Inc., which provides the cyptosystem methods and algorithms Notes uses for encryption. This product includes RC2, RC4, and RSA encryption technologies in its encryption algorithms.

Now that the terminology is out of the way, let's move on and define the two most popular, key-based encryption algorithms:

- **Symmetric algorithm**: This algorithm, also referred to as a single-key or secret-key algorithm, relies on using the same key for encryption and decryption. Plaintext is encrypted using a single key. The result is ciphertext, which is ready for transmission or storage. To transform ciphertext back to plaintext, the ciphertext is decrypted using the same single key that the encryption process used. The security strength for this process rests in the key. So long as the key is kept secret, ciphertext is secure. If, however, the key is lost or stolen, the ciphertext is no longer secure. The security achieved with this algorithm is similar to locking your car. A single key is used to lock your car, and the same key is required to unlock your car. Without this key, the car is inaccessible (of course, this statement assumes a perfect world).
Figure 1. The secret key encryption/decryption process

- **Public-key algorithm**: The process surrounding this algorithm is similar to the symmetric algorithm process, except that the key used for encryption is different than the key used for decryption. This dual-key process is possible because these keys are mathematically related. This relationship allows one key to encrypt information, and the second key to decrypt the information. With dual keys, one key cannot perform both operations of encryption and decryption. Therefore, we can safely place one key in a publicly accessible location and keep the second key private. Now, anyone can encrypt something specifically for me using my public key, and since I'm the only person with access to my private key, I'm the only person that can decrypt the information.

Figure 2. The public/private key encryption/decryption process

The power behind the symmetric and public-key algorithms is the length of the key. The longer the key, the stronger the encryption. Because of this simple principle, you might assume that Notes uses the longest key known to man. In theory, this assumption sounds nice. In practice, however, the mother-of-all-keys is impractical for two reasons: encrypt process time and export laws.

Encryption and decryption algorithms perform a tremendous amount of mathematical calculations. These calculations are extremely CPU intensive. The length of the key directly affects the time it takes to execute the calculations associated with these algorithms. Therefore, the longer the key, the longer the encryption process takes. If the process takes too long, no one will encrypt anything. On the other hand, a short key enables quick encryption, but also lowers security. Therefore, the length of the key is limited appropriately to achieve a balance between encryption process time and security.

**The many editions of Notes**

Aside from encryption process time, U.S. government export laws limit encryption key length. These laws are the driving force behind the three major editions of Notes: North American, International, and French. Despite the different names, the product functionality is exactly the same. The difference, however, lies in the length of the keys used for encryption.

The North American edition uses encryption keys that are 64-bits long. The U.S. Government, for reasons of national security, limits the length of encryption keys for export to 40 bits. To
comply with these restrictions, we have the International edition. When we generate a 64-bit key for the International edition, the top 24 bits are encrypted using the U.S. Government's public key and stored in what is called the Workfactor Reduction Field (WRF). Splitting the key in this manner results in a key that's 40 bits for the U.S. Government and 64 bits for everyone else. This approach maintains a high level of security worldwide without violating the export laws of the U.S. Government.

Most countries are content with the way the International edition complies with U.S. encryption key export laws. The government of France, however, found the International edition unacceptable. To comply with French law, we created the French edition, which uses a plain 40-bit encryption key and can therefore be "broken" by attackers willing to apply considerable computing power (presumably, including the French government).

Once people understand the three different implementations of encryption, they all ask the same question, "Can I send encrypted mail from the U.S. to a colleague in Japan (or any other country for that matter)?" Yes, provided you have access to the recipient's public key. Normally, you can access public keys in your domain's Public Address Book, or some other directory in your search path. In the worst case, however, you can always copy a recipient's information, including the public key, into your Personal Address Book. Also, keep in mind that when you send encrypted messages between different editions of Notes, the weaker of the two endpoint's encryption schemes is used.

**Note:** The real interoperability issue people encounter isn't sending encrypted mail, but rather, mixing and matching user IDs with different editions of the Notes client. Keep in mind that user IDs are upward compatible. Therefore, a user ID registered as an International license can use a North American copy of the Notes client. A user ID registered as a North American license, however, cannot use an International copy of the Notes client. Also, there are no "French" IDs. The French edition of Notes uses an International licensed ID, which uses weaker crytography when coupled with a French edition.

**The digital locks within Notes**

Now that we have a clear understanding of the fundamentals of encryption, we can take a look at where Notes allows us to use the digital locks of encryption.

**Database encryption**

The largest data group you can encrypt is an entire database. This option is ideally suited for mobile users that typically travel with sensitive or classified information on their laptops. If the laptop is lost or stolen, the databases are inaccessible unless the thief, or the lucky laptop finder, can successfully guess the user's ID password.

**Note:** Locally stored databases are not effectively protected by Access Control List (ACL) settings. Even the advanced setting of maintaining a consistent ACL across all replicas is not a foolproof means of defense. (For details about the ACL, check out the *Iris Today* article, "The ABC's of using the ACL.")
Because encryption and decryption are CPU-intensive operations, database encryption comes with three strength options: strong, medium, and simple. The stronger the encryption, the longer it takes to open the database. So long as you're not using disk compression, in most cases, the best option is medium encryption because of its balance between security strength and speedy access; however, if you're using disk compression, select simple encryption so that the disk compression works effectively.

To encrypt any local database:

1. Select the database.
2. Choose File - Database - Properties.
3. In the Properties dialog box, click the Encryption button. The following dialog box appears:

   **Figure 3. The Encryption dialog box**

4. Select "Locally encrypt this database using:" and then select the desired strength (Strong, Medium, Simple).

Database encryption uses the public-key algorithm. This means that encryption is performed with a user's public key and decrypted with his private key. By default, databases are encrypted using the current user's public key; however, you can specify a different user. A word of caution, selecting someone other than the default user encrypts the database using that person's public key. Therefore, the current user can no longer access this database. Before selecting another person, make sure you have a backup of this database.

### Document encryption

Database encryption requires no forethought or planning -- a couple of simple menu options, and the database is encrypted. Document encryption, however, is only possible when all of the fields on the document's *form* have encryption enabled.

During database design, the developer sets each field's security option to "Enable encryption for this field" ("None" is the default setting). Once this option is set, you can encrypt this document in one of three ways:

- **Document Properties**: The security (key) tab of the document properties dialog box allows you to encrypt a document using a public or secret key. If you click the icon next to the Public Encryption keys field, the Public Address Book dialog box opens. You can select one or more users who can view the encrypted document (make sure to add your own name; otherwise,
you won't be able to view the encrypted information even though you are the author). Secret key encryption presents a dropdown list of available keys (see the Secret Key Details sidebar for details).

- **Form Attribute:** Once you enable encryption on any of the fields, you can set the security property of the form and specify which secret key to use for encryption. The same encryption key is used for each document created with this form. The creator of the encryption key manages the distribution of the key. Without this secret key, no one can read the encrypted fields.
- **Reserved field, SecretEncryptionKeys:** Once you enable encryption on all the fields, you can add a reserved field named SecretEncryptionKeys. You can make this field a dropdown list of secret key choices, or leave it blank and let the document author specify which key to use.

**Note:** Encrypted fields won't appear in a view. Therefore, encrypt fields with caution, or at least design views that don't rely on encrypted fields.

### Field encryption

At the finest level of encryption, we have the *field*. This type of encryption works the same as document encryption, except only select fields have encryption enabled. For example, a document that contains employee information might include encrypted fields for confidential data, such as salary, social security number, and age. Less personal information such as name, work phone number, and title would remain in plaintext format.

### Mail encryption

The most popular type of information to encrypt is e-mail. For many people, e-mail is often a substitute for a real conversation. Therefore, similar to the real world where you can shut the door and have a private conversation, e-mail correspondence needs the same capability. To accommodate this need, Notes provides the ability to encrypt mail before you send it, after you receive it, or before you save it.

If you encrypt a mail message before you send it, Notes uses the recipient's public key to transform the plaintext of the message body to ciphertext. Once received, the recipient's private key decrypts the message. The private key is part of the user's ID file. Therefore, without the right ID file, the message body looks like gibberish. Of course, messages aren't safe if that ID file is lost or stolen; however, if the ID owner has applied a password (one that is difficult to guess, such as a string of random alpha-numeric characters) to his ID file, encrypted messages remain safe.

**Note:** Notes only encrypts the Body field of a mail message. The Subject, Sender, and Date fields are not affected. Therefore, when sending encrypted mail, refrain from disclosing sensitive information in the Subject line.

You can encrypt mail before you send it through a manual option, or automatically by selecting the encryption setting in your user preferences. To use the manual setting:

1. Open the message you want to send in edit mode.
2. Choose Actions - Delivery Options.
3. Select the Encrypt option.

**Figure 4. The Delivery Options dialog box**

![Delivery Options dialog box](image)

4. Click OK.

The message is encrypted. This option applies to the current message; future messages are not encrypted unless this delivery option is selected. If, however, you want all outgoing messages encrypted automatically, make this user preference selection:

2. Select Mail in the User Preferences dialog box.
3. Select Encrypt sent mail.

**Figure 5. User Preferences dialog box**

![User Preferences dialog box](image)

4. Click OK.

Once this option is set, Notes encrypts all outgoing messages. If the recipient's public key is not available, which is often the case for Internet mail, a warning message appears. This message allows you to continue sending this message or cancel it.

Since Notes administrators have direct access to the Notes Mail servers, they could, if improperly motivated, read all of your mail. If you suffer from television-induced paranoia, like myself, you might want to encrypt all incoming e-mail. This option is found in your Person document in the Public Address Book. To encrypt all incoming mail:
1. Open the Public Address Book.
2. Open your Person document.
3. Click the Edit person button.
4. In the "Encrypt incoming mail" field, select Yes.

**Figure 6. A Person document**

Enabling this option automatically encrypts all received mail with your public key. Decryption relies on your private key, which is part of your user ID file. Keep in mind, however, this setting does not automatically encrypt outgoing mail or copies of outgoing mail that you save. Therefore, if you've encrypted all received mail, you might as well encrypt all saved mail. This option is part of your user preference settings. To encrypt all the mail you save:

2. Select Mail in the user preferences dialog box.
3. Select Encrypt saved mail.
4. Click OK.

**Port encryption**

Encrypting databases, documents, fields, and e-mail makes for a highly secure computing environment; however, if you want to take privacy protection to the next level, encrypt the port used for data transmission.

Information travels across a network in the form of packets. These packets are little electronic impulses with no value to our human senses. Network sniffers, however, have the ability to analyze these packets and reveal their contents to the human eye. In order to prevent this type of high-tech eavesdropping, enable port encryption. To enable port encryption:

2. Select Ports in the user preferences dialog box.
3. Select the port you want to encrypt from the Communication Ports list box (for example, TCPIP).
4. Select Encrypt network data.
When you enable port encryption, a client server session starts by exchanging a secret session key. The server generates a random encryption key for the session and encrypts it with the client’s public key. This secret session key is sent to the client, where it’s decrypted using the client’s private key (which is part of the user ID file). Now that the client and server have the same secret key, communication between the client and the server is encrypted with this key. When the session is closed, the secret session key is deleted. Future encrypted communication sessions take place with a new, and different, secret session key.

**Note:** This type of added security does come with its drawbacks. Port encryption decreases communication performance by approximately 10% on a LAN, and slightly more if applied to a modem port that uses compression.

**What the future holds**

The developers of Notes understood the importance of encryption from the beginning. Release 1.0 was introduced with mail and port encryption. The use of encryption was extended in Release 2.0 with document encryption and secret keys for field-level encryption. Database and database design encryption were added in Release 4.0, and Release 4.5 brought us document encryption with public keys. Now, with Release 5.0 around the corner, what new implementations of encryption can we expect?

You’ll find most of the encryption enhancements in Release 5.0 have to do with Internet-related security, such as SSL (to learn more about SSL, check out the Iris Today articles, "SSL: it’s not just for commerce anymore" and "SSL client authentication: It’s a matter of trust"). When it comes to Notes-specific encryption enhancements, however, Release 5.0 is looking to use the latest and greatest BSAFE from RSA Data Security Inc., hopefully increasing the size of encryption keys (if the U.S. Government is willing), and implementing S/MIME (secure e-mail that interoperates with Microsoft and Netscape). Even though encryption enhancements are in the works, that doesn’t mean the current state of encryption is weak. Notes provides one the most secure computing environments.
environments in the industry, and what's new in Release 5.0 is only there to help carry on this
tradition.

**I live in a safe neighborhood -- I don't need encryption**

If you're the kind of person that routinely doesn't lock their car or house, encryption probably
doesn't mean much to you. Perhaps you rationalize your behavior by saying, "I live in a safe
neighborhood" or "My car isn't worth stealing." These statements might apply to our real world. In
cyberspace, however, our neighborhood is global, and people judge the value of information once
they read it and not by glancing at it from across the street. Like it or not, encryption is something
for everyone.
Related topics

- The ABC's of using the ACL
- SSL: it's not just for commerce anymore
- Secret Key Details sidebar
- SSL client authentication: It's a matter of trust

© Copyright IBM Corporation 1998
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)