# EFS - Encrypting File System. Encrypted Files and Folders (NTFS5 only)

## EFS Internals

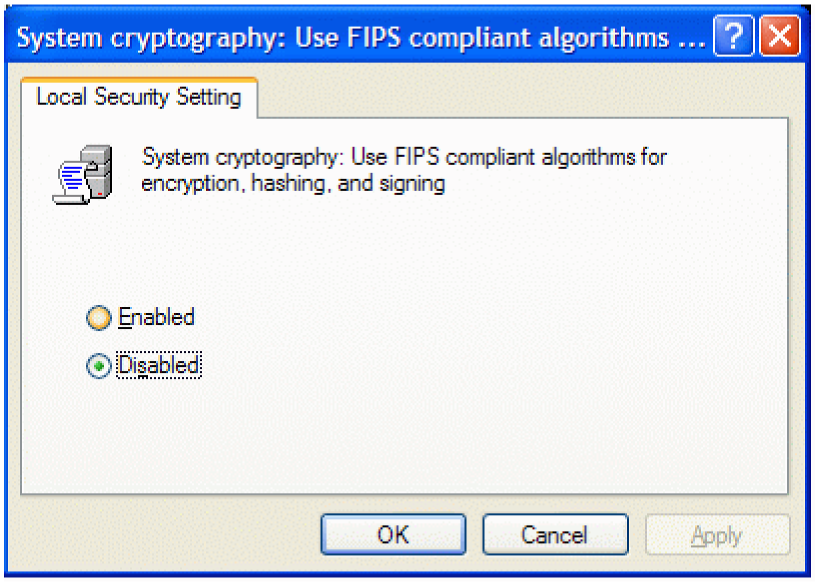
**EFS uses symmetric key encryption in combination with public key technology to protect files.** File data is being encrypted with symmetric algorithm (DESX). The key, used in symmetric encryption is called *File Encryption Key* (FEK). The FEK in its own turn is encrypted with a public/private key algorithm (RSA) and stored along with the file. The reason why two different algorithms are used is the speed of encryption. The performance burden of asymmetric algorithms is too much to use them for encrypting a large amount of data. Symmetric algorithms are about 1000 times faster making their suitable for encrypting of large amounts of data.

As a first setp to encrypt file, NTFS creates a log file called Efs0.log in System Volume Information folder on the same drive, as encrypted file. Then EFS aquires access CryptoAPI context. It uses Microsoft Base Cryptographic Provider 1.0 as cryptographic provider. Having the crypto context open, EFS generate File Encryption Key (FEK).

The next step is to get public/private key pair; if it does not exist at this stage (the case when EFS invoked first time), EFS generate a new pair. EFS uses 1024-bit RSA algorithm to encrypt FEK.

Then, EFS creates Data Decryption Field (DDF) for the current user, where it places FEK and encrypts it with public key. If recovery agent is defined by system policy, EFS creates also Data Recovery Field (DRF) and places there FEK encrypted with public key of recover agent. A separate DRA is created for every recovery agent defined. Please note, that on Windows XP not included into domain, there's no recovery agent is defined, so this step is omitted.

Now a temporary file Efs0.tmp is created in the same folder as the file being encrypted. The contents of original file (plain text) is copied into temporary file, after that the original is overwritten with encrypted data. By default, EFS uses DESX algorithm with 128-bit key to encrypt file data, but Windows could be also configured to use stronger 3DES algorithm with 168-bit key. In that case FIPS compliant algorithms usage must be turned on in LSA policy (it is disabled by default):



System cryptography: Use FIPS compliant algorithms ...

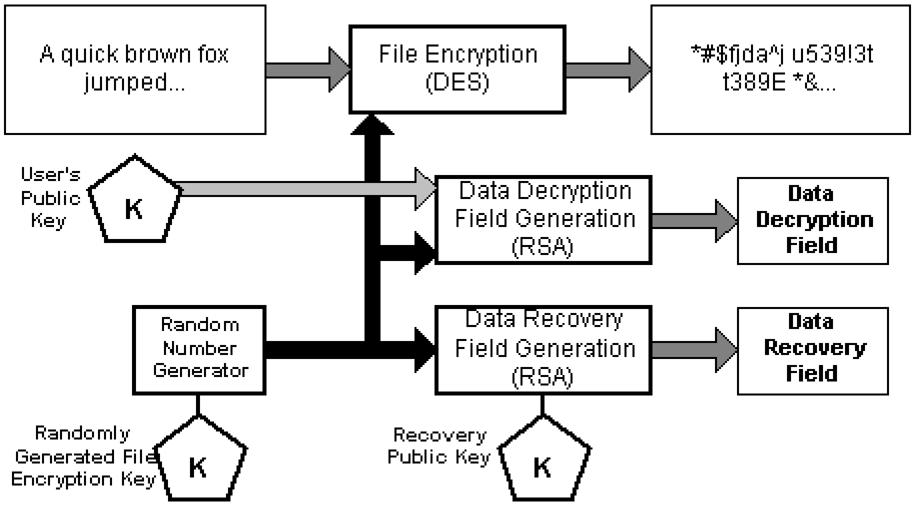
Local Security Setting

System cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing

Enabled [✓ Disabled]

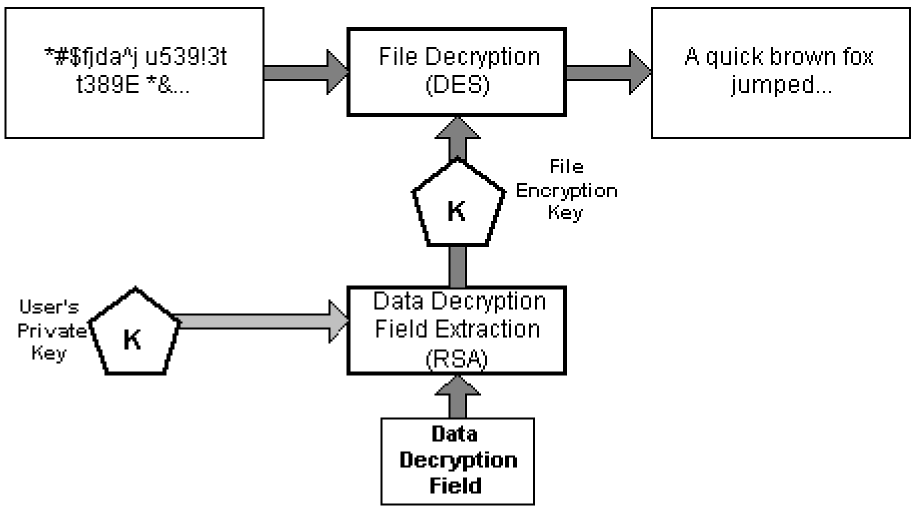
EFS uses the registry to determine if it will use DESX or 3DES. If HKLM\SYSTEM\CurrentControlSet\Control\LSA\FipsAlgorithmPolicy = 1, then 3DES will be used. If not, then EFS checks HKLM\Software\Microsoft\Windows NT\CurrentVersion\EFS\AlgorithmID (this value may not be present); if present, it will have ID CALG_3DES or CALG_DESX, otherwise, DESX should be used.

After encryption is done, temporary and log files are deleted.


A quick brown fox jumped...
File Encryption (DES)
*#$fjda^j u539!3t t389E *&...
User's Public Key
K
Data Decryption Field Generation (RSA)
Data Decryption Field
Random Number Generator
Data Recovery Field Generation (RSA)
Data Recovery Field
Randomly Generated File Encryption Key
K
Recovery Public Key
K

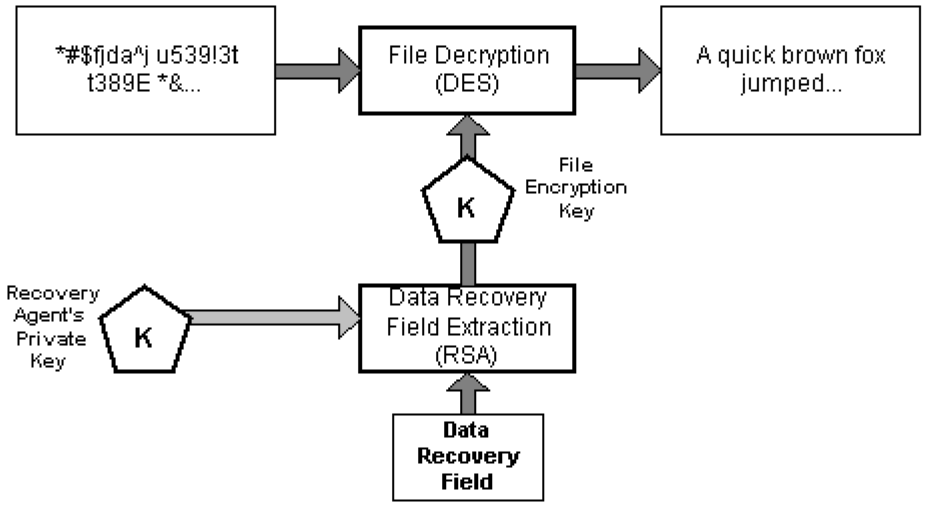
After file is encrypted, only users who has correspondent DDF or DRF can access the file. This mechanism is separate from common security meaning that beside rights to access file, the file must have its FEK encrypted with user's public key. Only user who can decrypt FEK with his own private key, can access the file. The consequence is, that user, who has access to the file, can encrypt it thus preventing the owner to access his own file. Initially only one DDF is created for user who encrypts the file, but later he can add extra users to key ring. In this case EFS simply decrypts FEK with private key of user who wants to give access to the file to another user, and encrypts FEK with public key of target user, thus creating a new DDF which is stored along with the first one.

The decryption process is opposite to encryption:

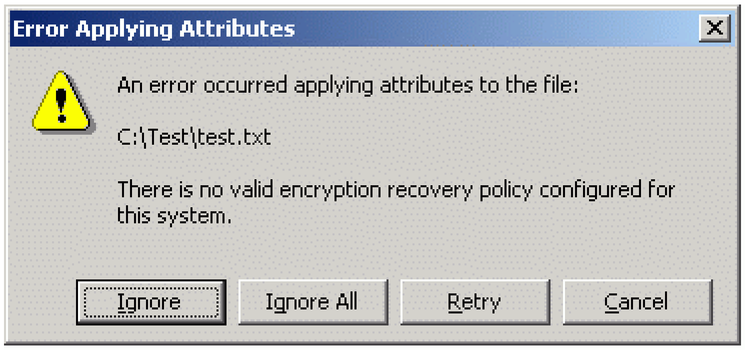

*#$fjda^j u539!3t t389E *&...
File Decryption (DES)
A quick brown fox jumped...
K
File Encryption Key
User's Private Key
K
Data Decryption Field Extraction (RSA)
Data Decryption Field

First, system checks if user has a private key used by EFS. If yes, it reads EFS attributes and walk through the DDF ring looking for DDF for current user. If DDF is found, user's private key is used to decrypt FEK extracted from DDF. Using decrypted FEK, EFS decrypts file data. It should be noticed that file never decrypted in whole but rather by sectors when upper level module requests particular sector.

Recovery process is similar to decryption, except that it uses the recovery agent's private key to decrypt the FEK in the DRF, not in DDF:


*#$fjda^j u539!3t t389E *&...
File Decryption (DES)
A quick brown fox jumped...
K
File Encryption Key
Recovery Agent's Private Key
K
Data Recovery Field Extraction (RSA)
Data Recovery Field

DRA policy is implemented differently for Windows 2000 and Windows XP. In Windows 2000 by default on computers, not included into domain, local Administrator is added to Public Key Policy as Encrypted Data Recovery Agent. So, when user encrypts file, both DDF and DRF fields are created. If the last DRA is deleted, the whole EFS functionality is turned off and it is not possible to encrypt file anymore.


Error Applying Attributes

An error occurred applying attributes to the file:

C:\Test\test.txt

There is no valid encryption recovery policy configured for this system.

In Windows XP the situation is different. Since majority of home users working standalone do not need anybody else to be able to decrypt file except themselves, there's no need in data recovery agents, so there's no DRA included into Public Key Policy and EFS works without DRA. In this case only DDF field is created for encrypted file.

previous | contents | next