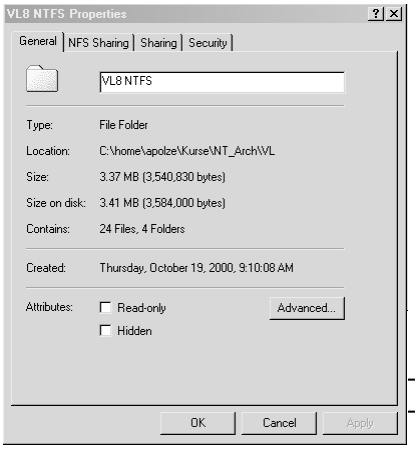
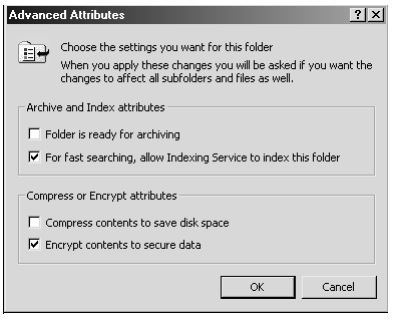
Unit 8: File System

8.3. Encrypting File System Security in Windows 2000

Encrypting File System Security

- EFS relies on Windows 2000 cryptography support
  - Transparent encryption through Windows Explorer or cipher-utility





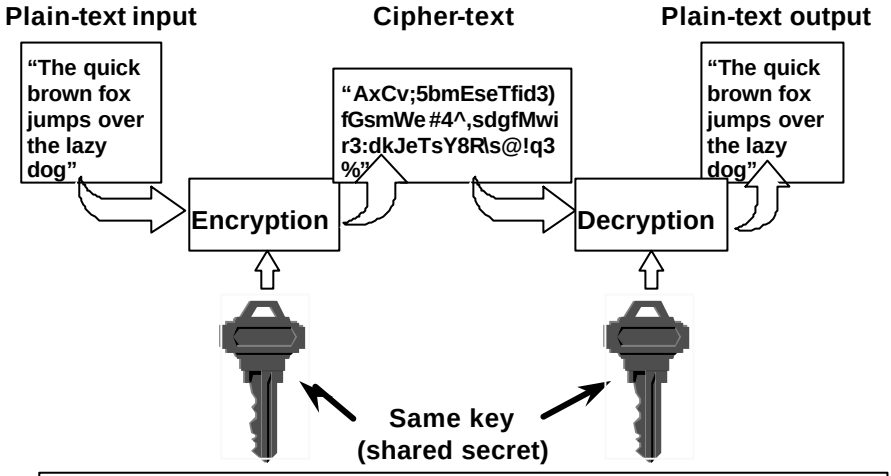
EFS operation

- When a file is encrypted...
  - EFS generates random File Encryption Key (FEK) to encrypt file content
  - Stronger variant of Data Encryption Standard (U.S.: 128/intl.: 56 bit) (symmetric DESX-algorithm) to encrypt file content (fast, shared secret)
  - File‘s FEK is stored with file and encrypted using the file creator‘s RSA public key (slow)
- File can be decrypted...
  - only with the user‘s private RSA key
  - What about lost keys?
- FEK can be stored in multiple encryptions...
  - Users can share an encrypted file
  - Can store a recovery key to allow recovery agents access to files
- Secure public/private key pairs are essential
  - Stored on computer harddisk... (but soon on smartcards)

Basic Terminology

- Plaintext
  - The stuff you want to secure, typically readable by humans (email) or computers (software, order)
- Ciphertext
  - Unreadable, secure data that must be decrypted before it can be used
- Key
  - You must have it to encrypt or decrypt (or do both)
- Cryptoanalysis
  - Hacking it by using science
- Complexity Theory
  - How hard is it and how long will it take to run a program

Symmetric Key Cryptography



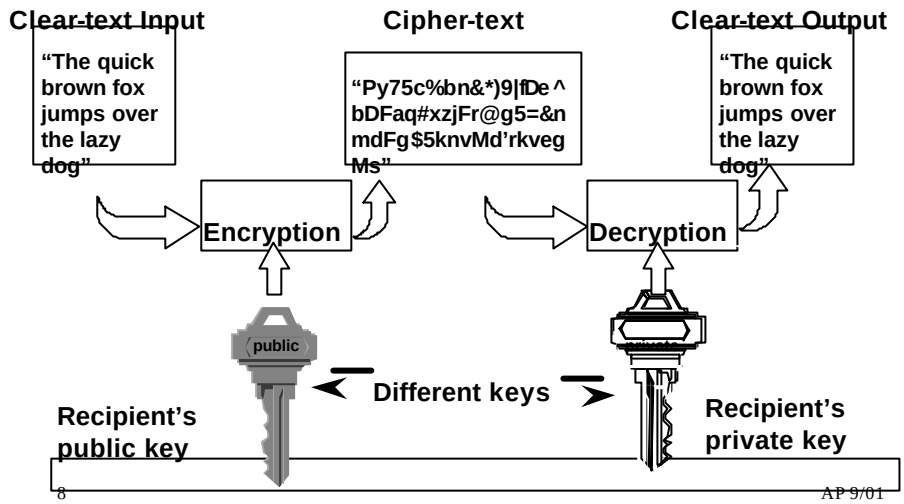
Symmetric Pros and Cons

- Weakness:
  - Agree the key beforehand
  - Securely pass the key to the other party
- Strength:
  - Simple and really very fast (order of 1000 to 10000 faster than asymmetric mechanisms)
    - Super-fast if done in hardware (DES)
    - Hardware is more secure than software, so DES makes it really hard to be done in software, as a prevention

Public Key Cryptography

- Knowledge of the encryption key doesn't give you knowledge of the decryption key
- Receiver of information generates a pair of keys
  - Publish the public key in directory
- Then anyone can send him messages that only she can read

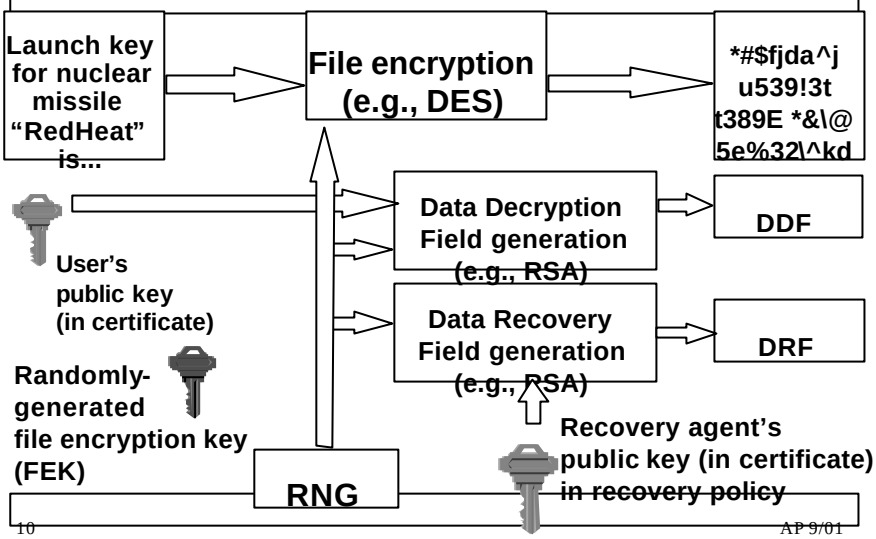
Public Key Encryption



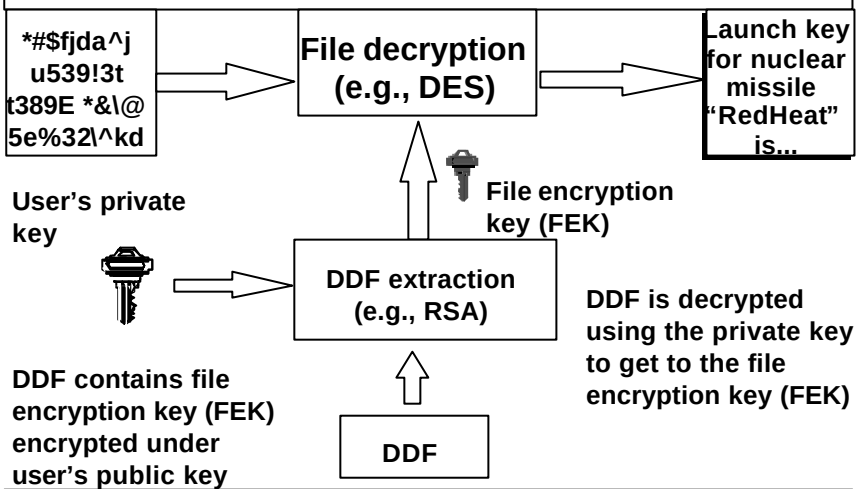
Problem of Key Recovery

- What if you lose the private key? ☺
- Data recovery by authorized agents
  - Integrated key management
- Windows 2000:
  - Flexible recovery policy
    - Enterprise, domain, or per machine
  - Encrypted backup and restore
    - Integrated with Windows NT backup
- Potential weakness but you can opt not to use it!

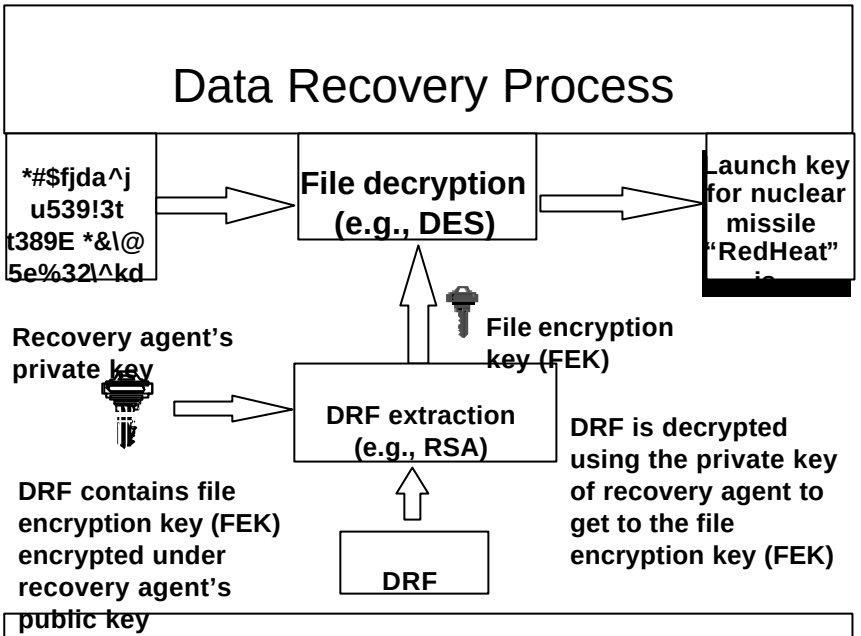
Data Encryption Process



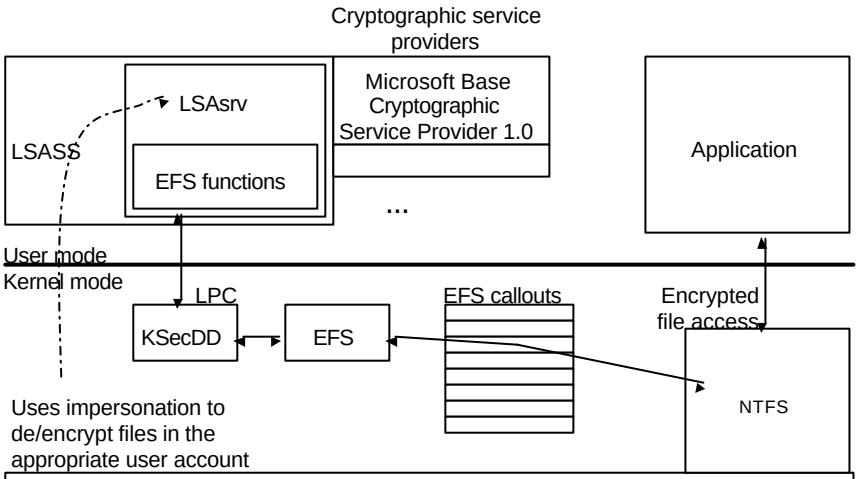
Data Decryption Process



Data Recovery Process



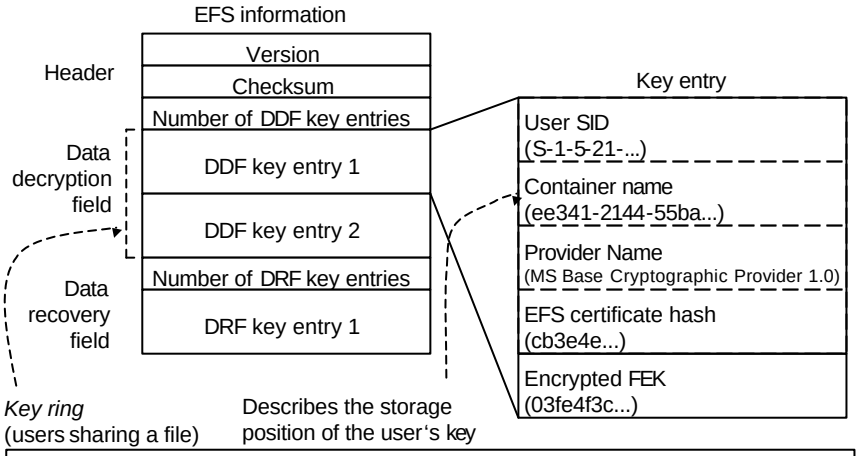
Windows 2000 EFS Architecture



EFS Components

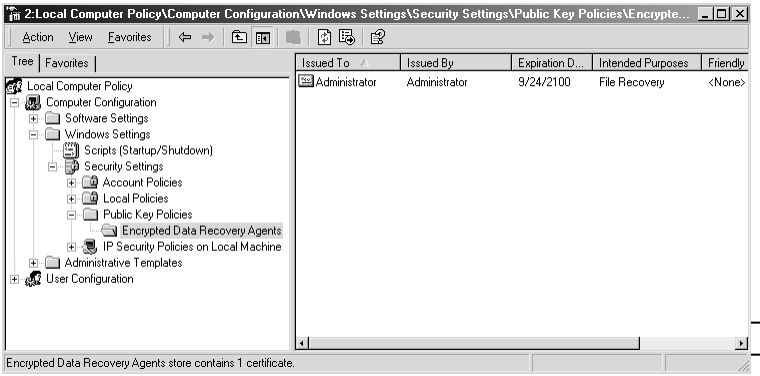
- Local Security Authority Subsystem
  - LSASS (\Winnt\System32\Lsass.exe) manages logon sessions
  - EFS obtains FEKs from LSASS
- KSecDD device driver implements comm. with LSASS
- LSAsrv listens for LPC comm.
  - Passes requests to EFS functions
  - Uses functions in MS CryptoAPI (CAPI) to decrypt FEK for EFS
- Crypto API ...
  - is implemented by Cryptographic Service Provider (CSP) DLLs
  - Details of encryption/key protection are abstracted away
- NTFS does not require EFS driver (Efs.sys)
  - But encrypted file will not be accessible without presence of Efs driver

Format of EFS information and key entries for a file

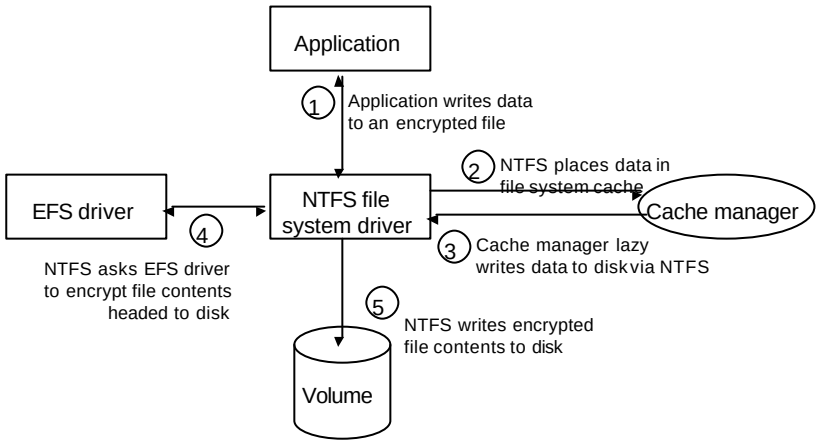


Encrypted Data Recovery Agents group policy

- Use Group Policy MMC snap-in to configure recovery agents (...list may be empty)



Flow of EFS



Encryption Process Details

1. User profile is loaded if necessary
2. A log file Efs*x*.log is created
   - In system volume info dir; x is unique number
3. Base Cryptographic Provider 1.0 generates random 128-bit FEK
4. User EFS private/public key pair is generated or obtained
   - HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion \EFS\CurrentKeys\CertificateHash identifies the user's key pairs
5. A DDF key ring is created for the file with an entry for the user
   - Entry contains copy of FEK encrypted with user's public key
6. A DRF key ring is created for the file
   - Has an entry for each recovery agent on the system
   - Entries contain copies of FEK encrypted with agents' public keys

Encryption Process Details (contd.)

7. A backup file is created (Efs0.tmp)
   - Same directory as original file
8. DDF and DRF rings are added to a header
   - EFS attributes - $LOGGED_UTILITY_STREAM
9. Backup file is marked encrypted, original file is copied to backup
10. Original file's contents are destroyed
    - Backup is copied to original
    - This results in encrypting the file contents
11. The backup file is deleted
12. The log file is deleted
13. The user profile is unloaded (if it was loaded in step 1)

In case of system crash, either original file or backup contain valid copy of the file content.

Backing Up Encrypted Files

- Data is never available in unencrypted form
  - Except to applications that access file via encryption facility
- EFS provides a facility for backup programs:
  - New EFS API: *OpenEncryptedFileRaw(), ReadEncryptedFileRaw(), WriteEncryptedFileRaw(), CloseEncryptedFileRaw()*
  - Implemented in Advapi32.dll, use LPC to invoke function in LSAsrv
  - LSAsrv calls *EfsReadFileRaw()* to obtain file's EFS attribute and the encrypted contents from NTFS driver
  - Similarly, *EfsWriteFileRaw()* is invoked to restore file's contents