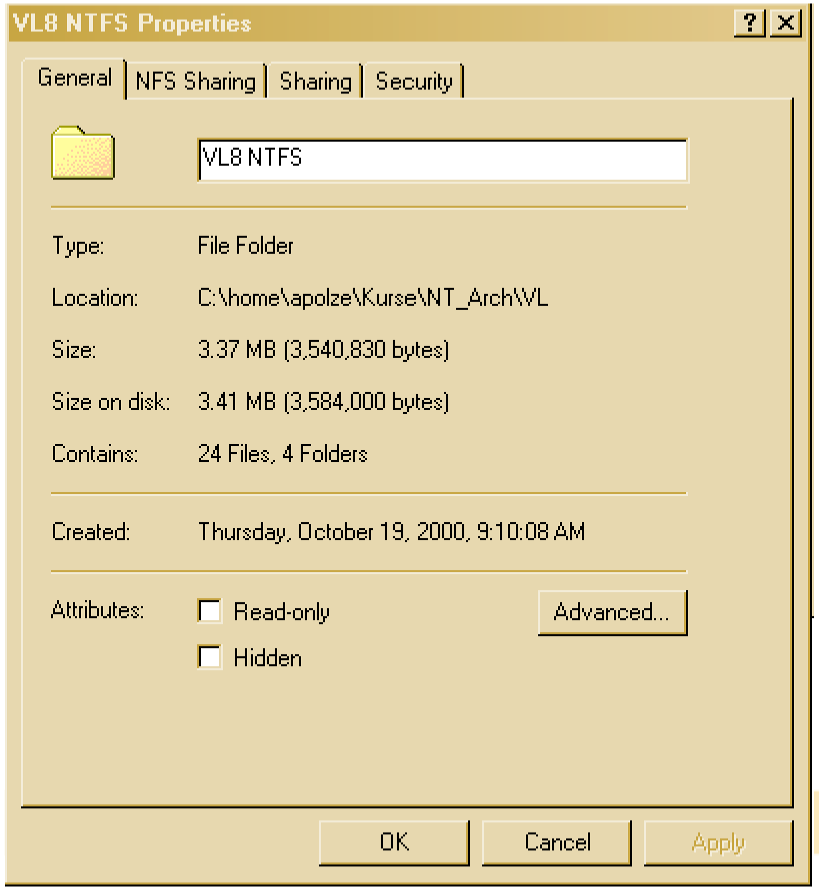
Unit 8: File System

8.3. Encrypting File System Security in Windows 2000

Encrypting File System Security

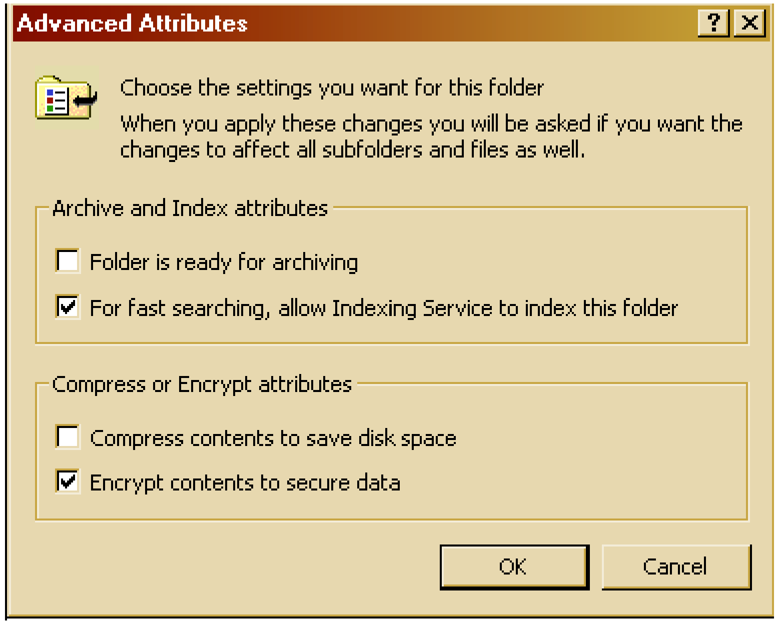
- EFS relies on Windows 2000 cryptography support
  - Transparent encryption through Windows Explorer or cipher-utility



VL8 NTFS Properties

VL8 NTFS

Type:	File Folder
Location:	C:\home\apolze\Kurse\NT_Arch\VL
Size:	3.37 MB (3,540,830 bytes)
Size on disk:	3.41 MB (3,584,000 bytes)
Contains:	24 Files, 4 Folders
Created:	Thursday, October 19, 2000, 9:10:08 AM
Attributes:	☐ Read-only
	☐ Hidden



Advanced Attributes

Choose the settings you want for this folder

When you apply these changes you will be asked if you want the changes to affect all subfolders and files as well.

Archive and Index attributes

☐ Folder is ready for archiving

☑ For fast searching, allow Indexing Service to index this folder

Compress or Encrypt attributes

☐ Compress contents to save disk space

☑ Encrypt contents to secure data

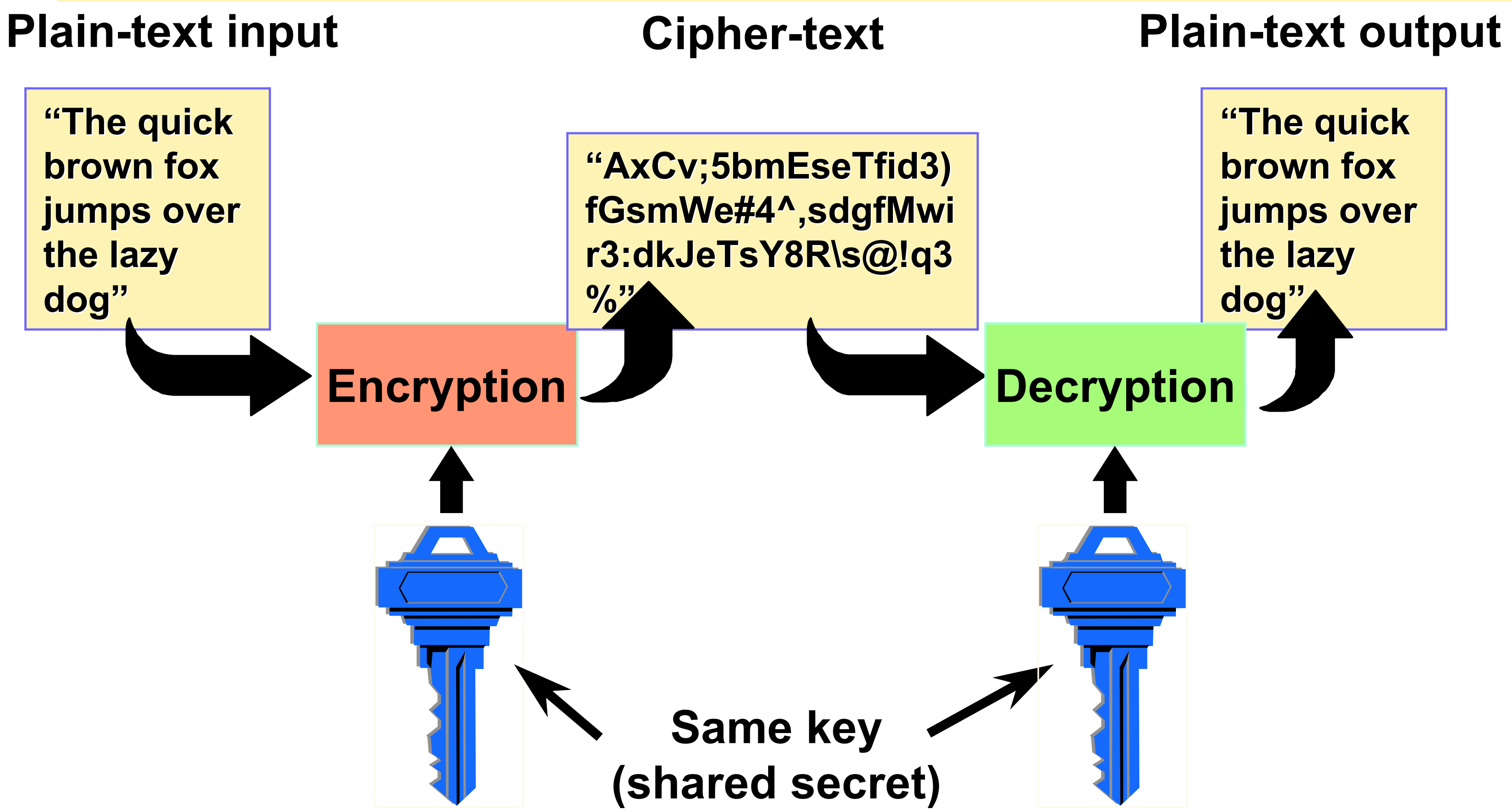
EFS operation

- 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


Plain-text input
Cipher-text
Plain-text output
“The quick brown fox jumps over the lazy dog”
“AxCv;5bmEseTfid3)fGsmWe#4^,sdgfMwir3:dkJeTsY8R\s@!q3%”
“The quick brown fox jumps over the lazy dog”
Encryption
Decryption
Same key (shared secret)

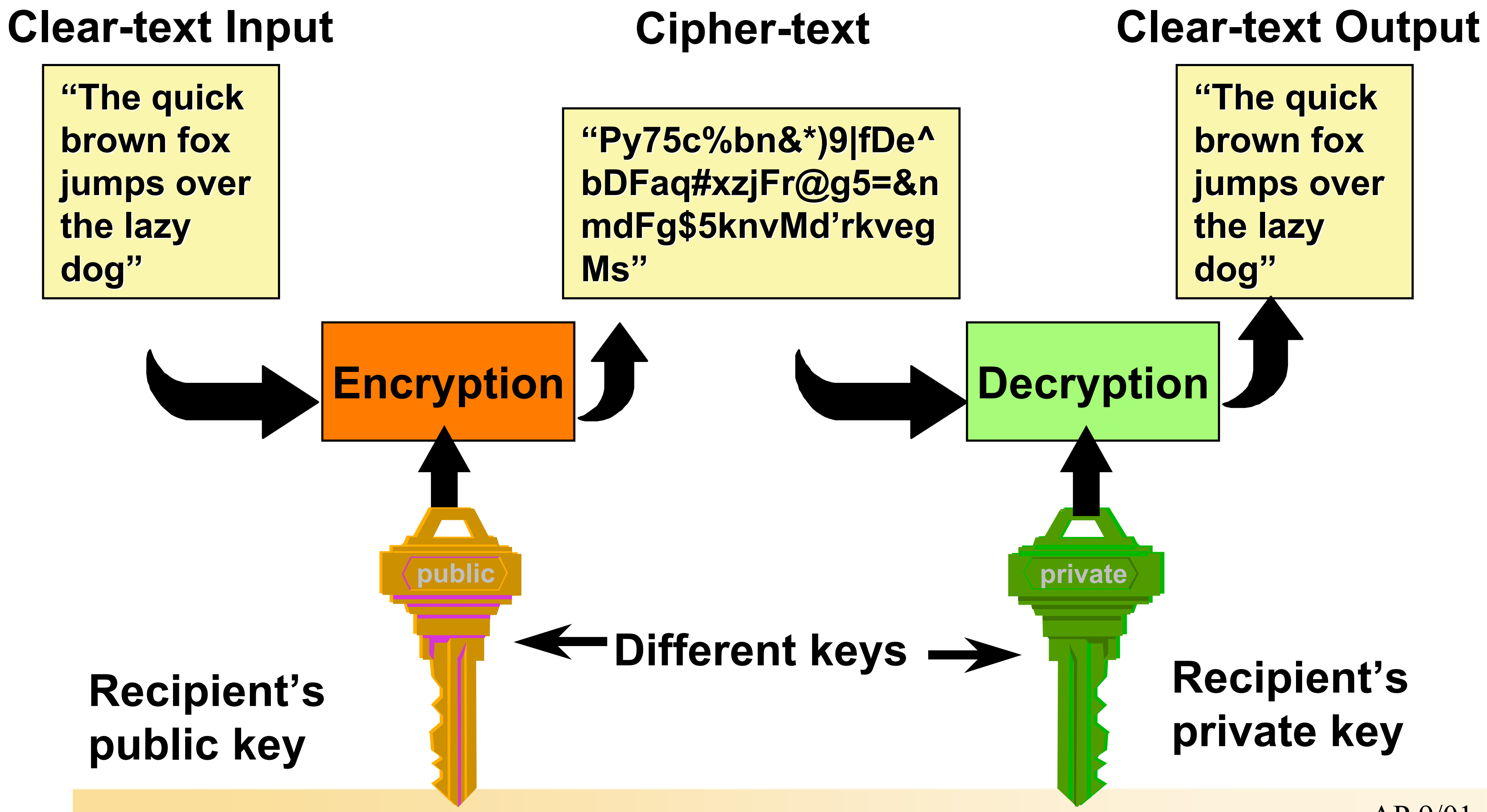
Symmetric Pros and Cons

- 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

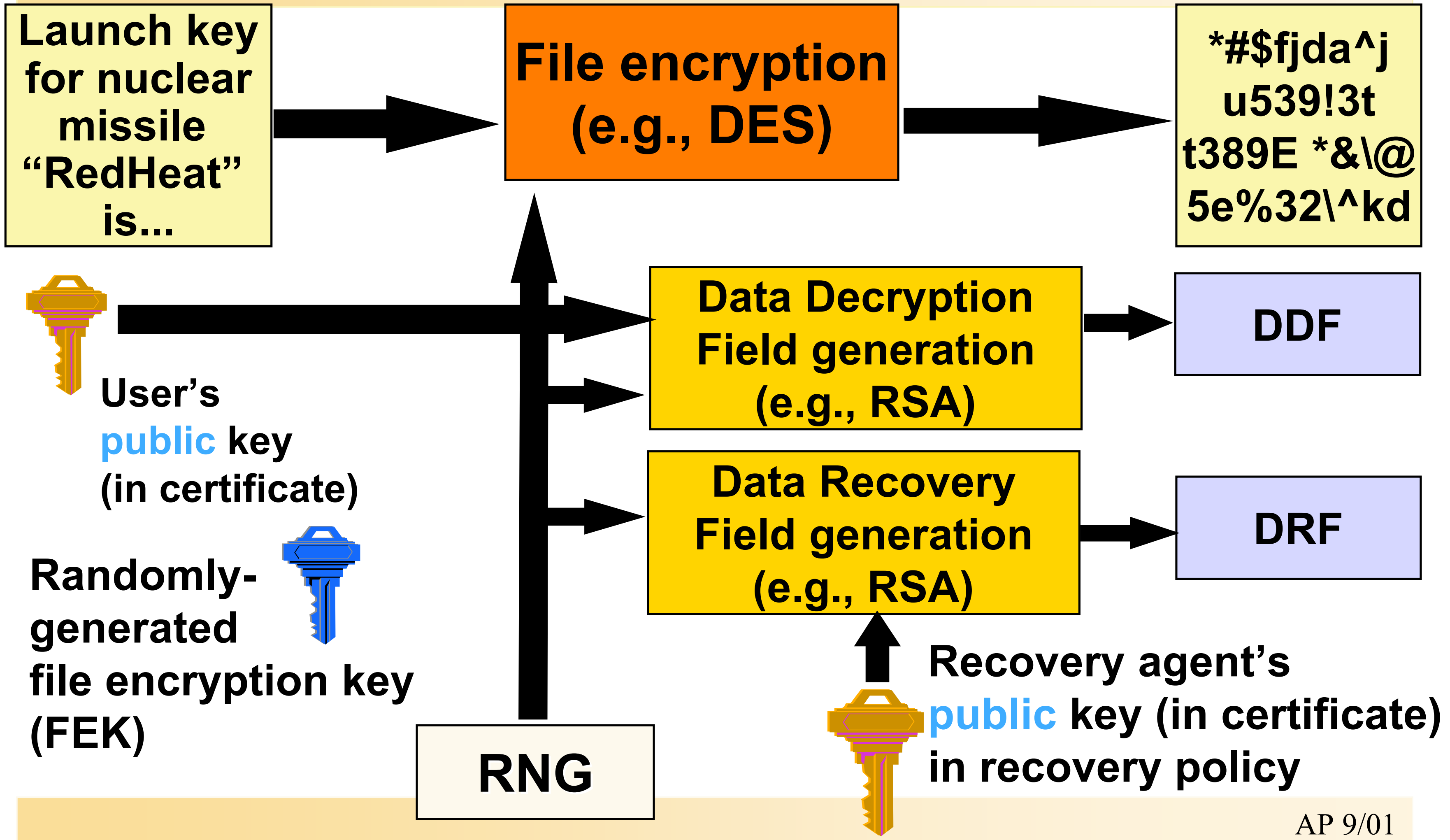
Public Key Encryption


Clear-text Input
"The quick brown fox jumps over the lazy dog"
Cipher-text
"Py75c%bn&*)9|fDe^bDFaq#xzjFr@g5=&nmdFg$5knvMd'rkvegMs"
Clear-text Output
"The quick brown fox jumps over the lazy dog"
Encryption
Decryption
public
private
Different keys
Recipient's public key
Recipient's private key

Problem of Key Recovery

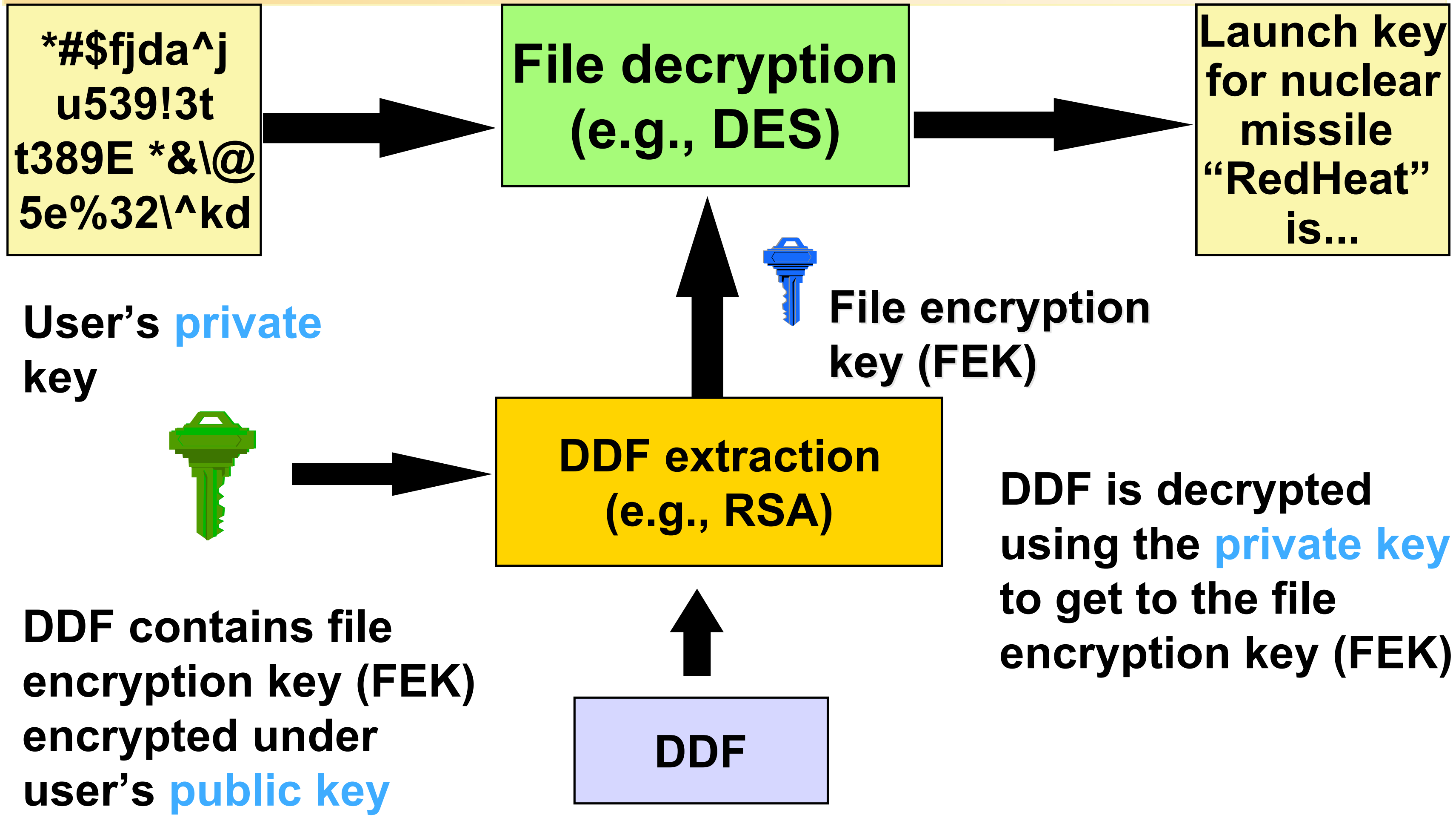
- What if you lose the private key? ☺
- Data recovery by authorized agents
  - Integrated key management
- Windows 2000:
  - Flexible recovery policy
    - Enterprise, domain, or per machine
  - Encrypted backup and restore
    - Integrated with Windows NT backup
- Potential weakness but you can opt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

Data Encryption Process



Launch key for nuclear missile "RedHeat" is...
File encryption (e.g., DES)
*#$fjda^j u539!3t t389E *&\@ 5e%32\^kd
User's public key (in certificate)
Data Decryption Field generation (e.g., RSA)
DDF
Data Recovery Field generation (e.g., RSA)
DRF
Randomly-generated file encryption key (FEK)
RNG
Recovery agent's public key (in certificate) in recovery policy

Data Decryption Process


*#$fjda^j u539!3t t389E *&\@ 5e%32\^kd
File decryption (e.g., DES)
Launch key for nuclear missile “RedHeat” is...
DDF extraction (e.g., RSA)
DDF

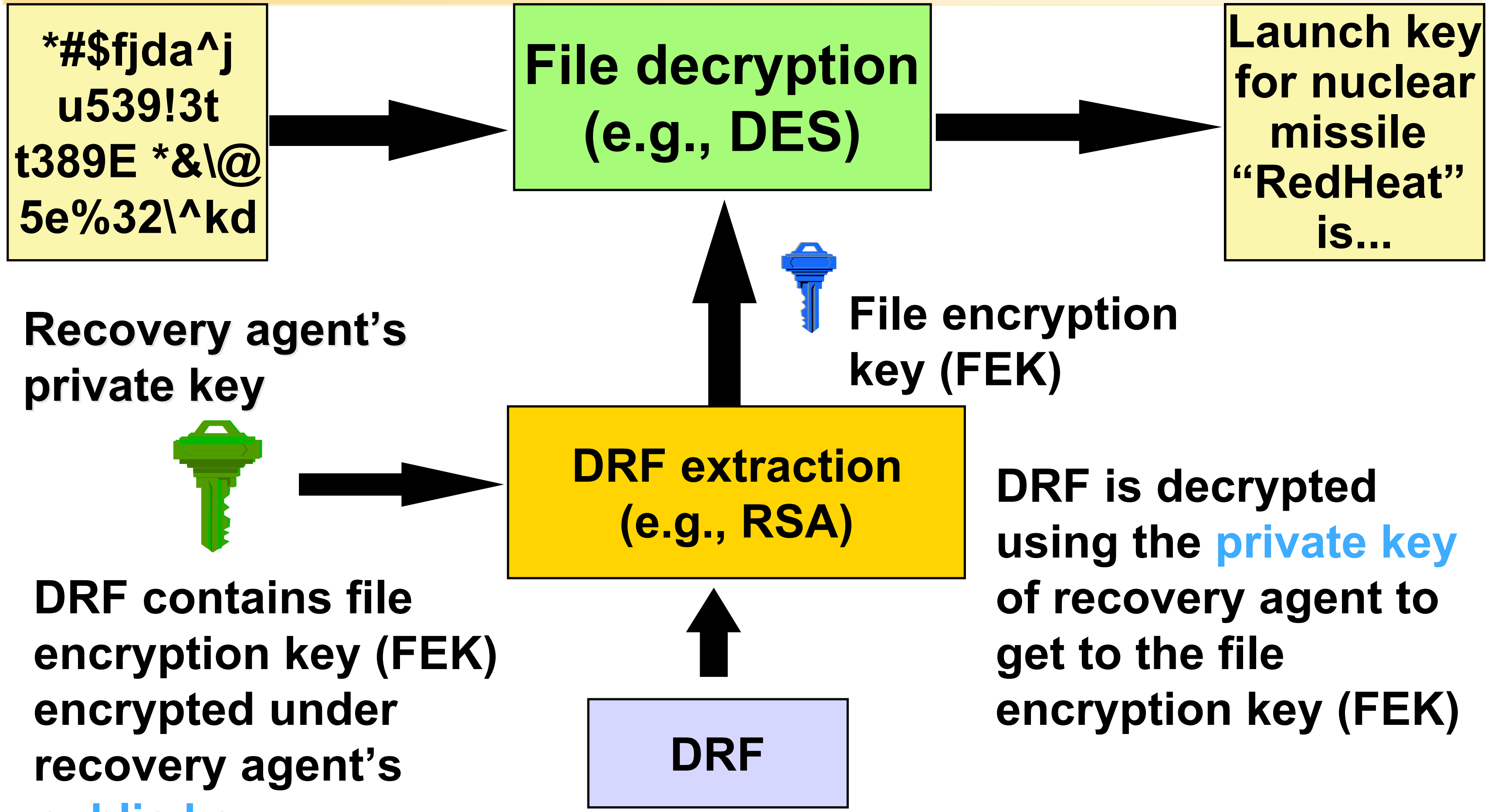
User’s private key

File encryption key (FEK)

DDF is decrypted using the private key to get to the file encryption key (FEK)

DDF contains file encryption key (FEK) encrypted under user’s public key

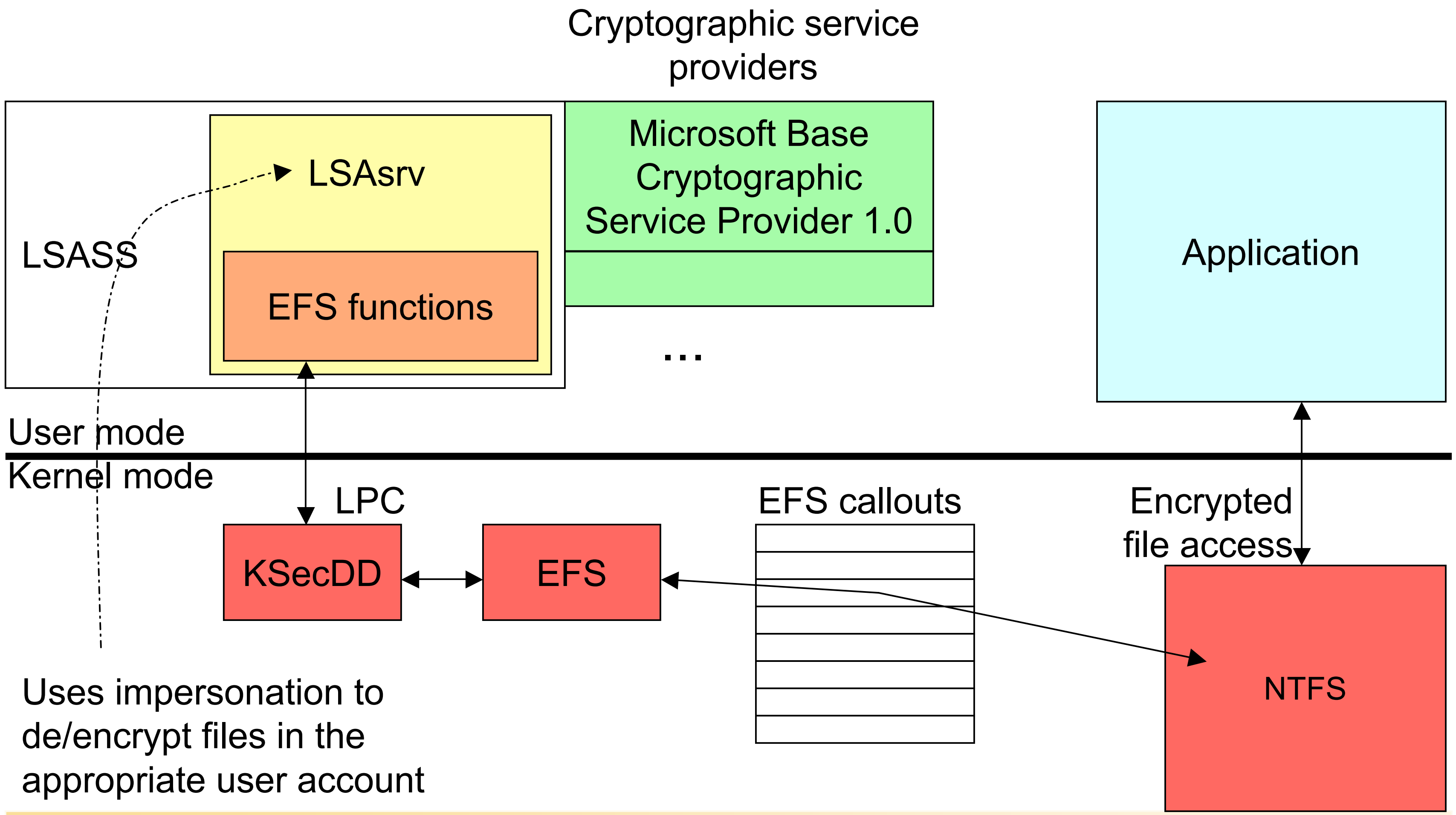
Data Recovery Process


*#$fjda^j u539!3t t389E *&\@ 5e%32\^kd
File decryption (e.g., DES)
Launch key for nuclear missile “RedHeat” is...
File encryption key (FEK)
Recovery agent’s private key
DRF extraction (e.g., RSA)
DRF

DRF contains file encryption key (FEK) encrypted under recovery agent’s public key

DRF is decrypted using the private key of recovery agent to get to the file encryption key (FEK)

Windows 2000 EFS Architecture

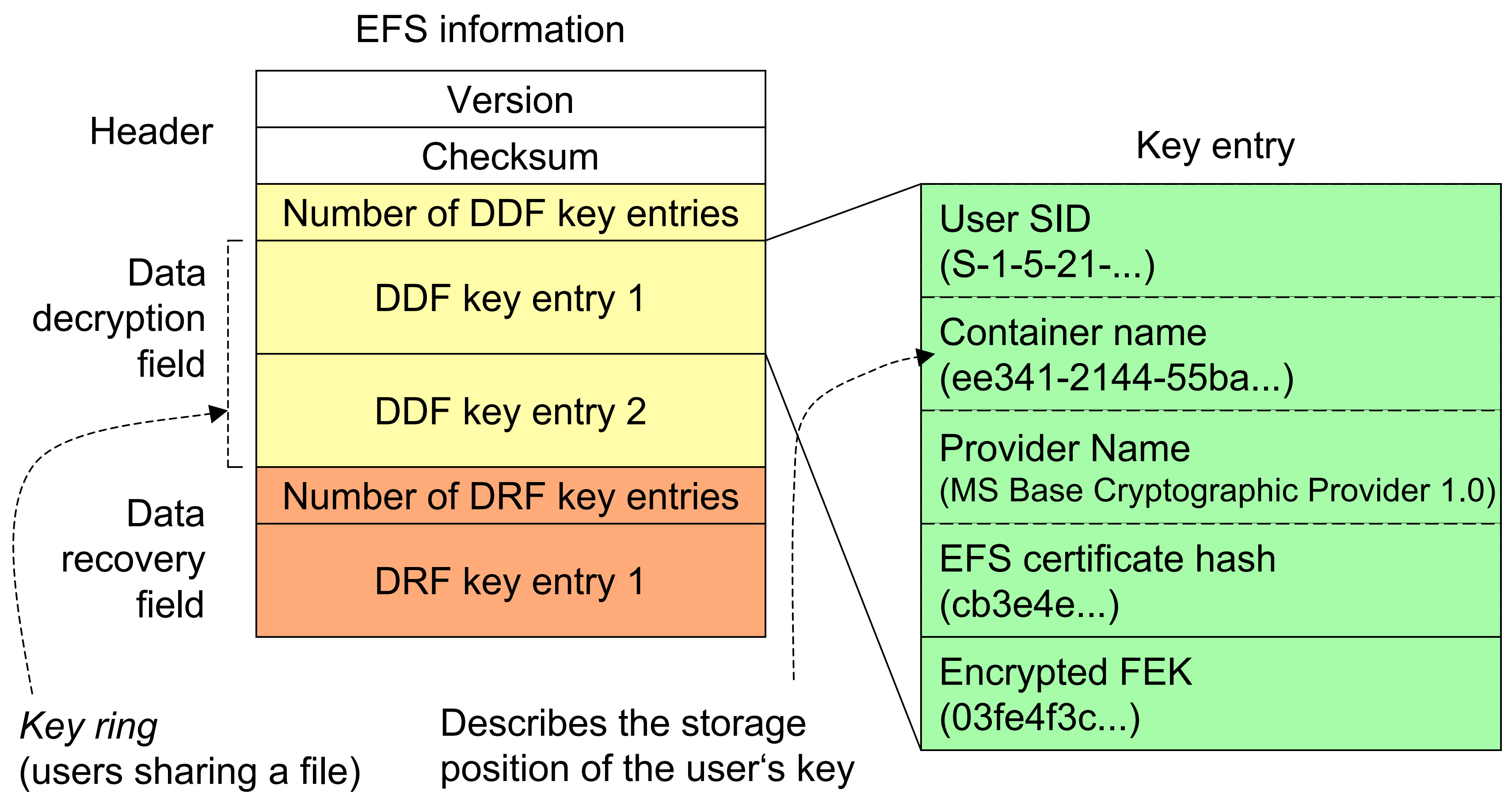


Cryptographic service providers
Microsoft Base Cryptographic Service Provider 1.0
…
LSASS
LSAsrv
EFS functions
Application
User mode
Kernel mode
LPC
KSecDD
EFS
EFS callouts
Encrypted file access
NTFS
Uses impersonation to de/encrypt files in the appropriate user account

EFS Components

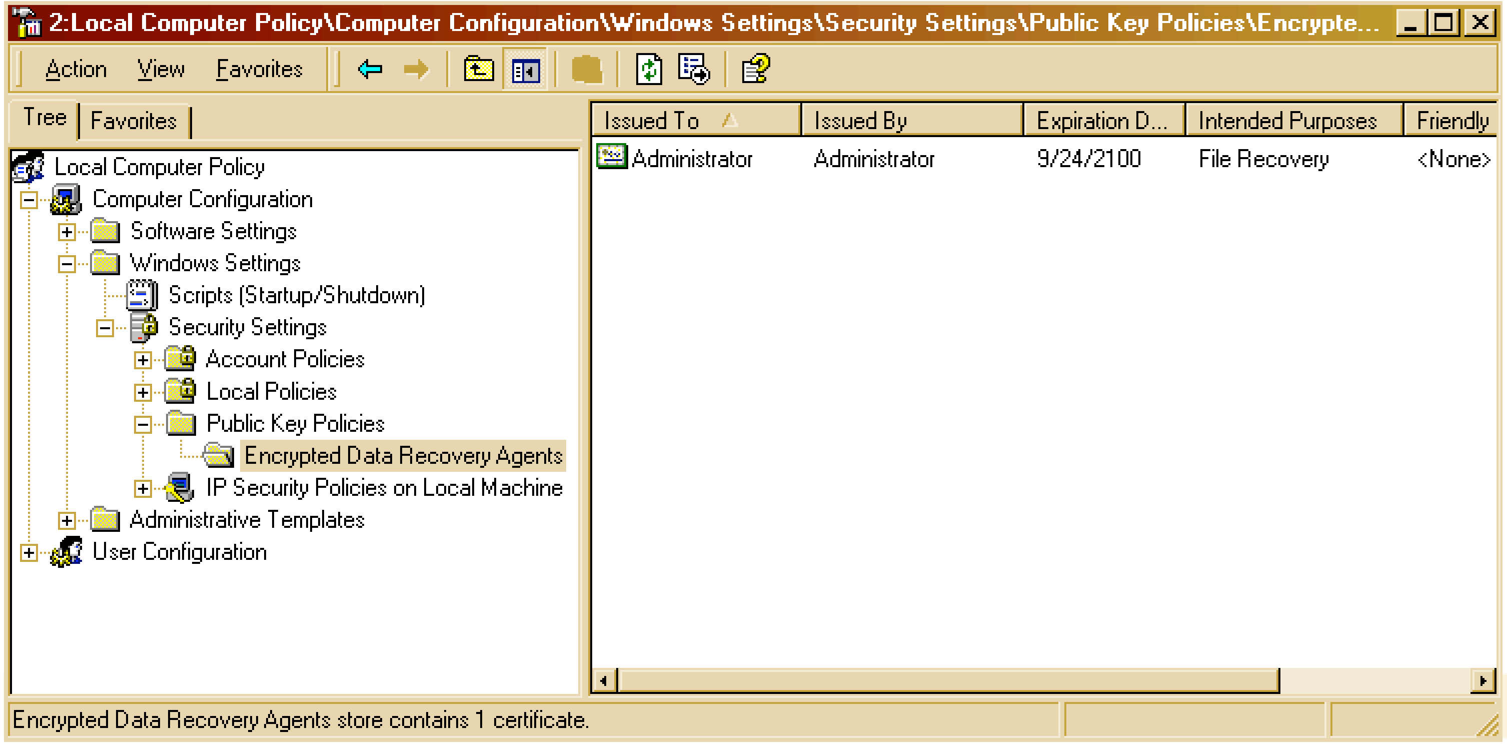
- Local Security Authority Subsystem
  - LSASS (\Winnt\System32\Lsass.exe) manages logon sessions
  - EFS obtains FEKs from LSASS
- KSecDD device driver implements comm. with LSASS
- LSAsrv listens for LPC comm.
  - Passes requests to EFS functions
  - Uses functions in MS CryptoAPI (CAPI) to decrypt 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


EFS information
Header
Version
Checksum
Number of DDF key entries
Data decryption field
DDF key entry 1
DDF key entry 2
Number of DRF key entries
Data recovery field
DRF key entry 1
Key entry
User SID (S-1-5-21-...)
Container name (ee341-2144-55ba...)
Provider Name (MS Base Cryptographic Provider 1.0)
EFS certificate hash (cb3e4e...)
Encrypted FEK (03fe4f3c...)
Key ring (users sharing a file)
Describes the storage position of the user‘s key

Encrypted Data Recovery Agents group policy

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

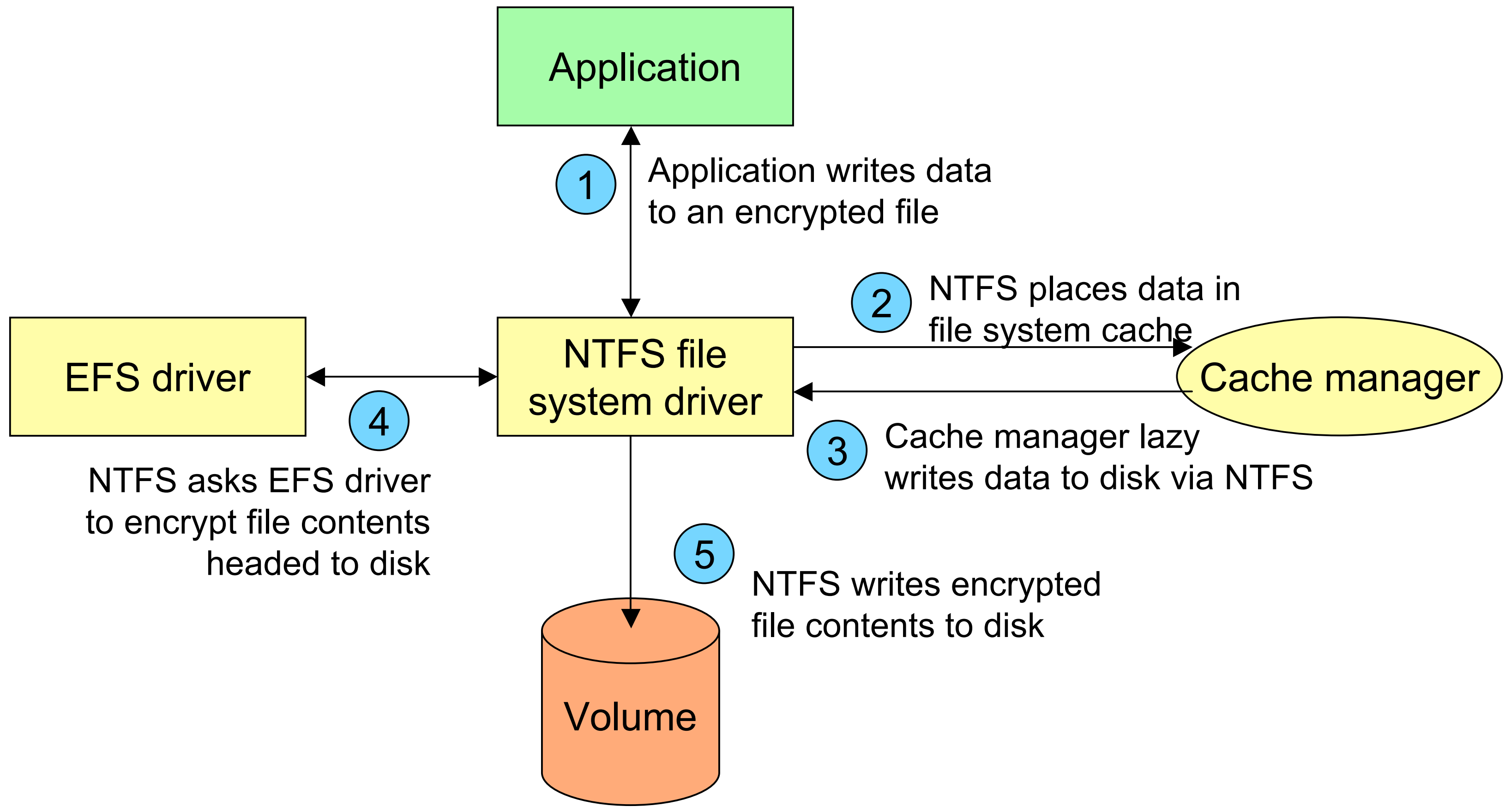


2:Local Computer Policy\Computer Configuration\Windows Settings\Security Settings\Public Key Policies\Encrypte...

Issued To	Issued By	Expiration D...	Intended Purposes	Friendly
Administrator	Administrator	9/24/2100	File Recovery	<None>

Encrypted Data Recovery Agents store contains 1 certificate.

Flow of EFS


Application
1
Application writes data to an encrypted file
2
NTFS places data in file system cache
EFS driver
NTFS file system driver
Cache manager
4
3
Cache manager lazy writes data to disk via NTFS
NTFS asks EFS driver to encrypt file contents headed to disk
5
NTFS writes encrypted file contents to disk
Volume

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 thta 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