

Cryptographic File System Using Smart Card

AMIT GUD (4691)

Department of Computer
D. Y. Patil College of Engg.
Friday, September 11, 2003

Agenda

- **Security Issues (Need for encryption)**
- **Encryption**
 - Encryption System wish list
 - Encryption Levels
 - Hardware Level
 - Application Level
 - System Level
- **Cryptographic File System**
 - Functionality
 - User Interface
 - Encryption Scheme
 - DES Modes
 - Problems with CFS
- **Smart Card**
 - Introduction
 - Why Smart Card?
- **SC-CFS**
 - Key Management
 - Caching
 - Performance Evaluation
- **SC-CFS & CFS: A comparative study**
 - Sniffing & stealing
 - Storage theft & dictionary attacks
- **Conclusion**

Security Issues (Need for Encryption)

- **Data Secrecy**
 - Keep data encrypted
 - Keep metadata encrypted
- **Safe Data Transfers**
 - Secure sharing of files
 - Safe data access from the network
- **Intrusion / Attacks**
 - Network attacks
 - Insider attacks

Encryption



Encryption system wish list

- **Under user control**
- **Infrequent typing of keys**
- **Transparent access semantics (compatibility)**
- **Should work with all applications**
- **Transparent performance**
- **Granular enough**
- **Integrity of file contents**
- **Concurrent access**
- **Network sharable**

Hardware Level Encryption

- **Pros:**

- + **End-to-end encryption**
- + **Error free**
- + **Very fast**
- + **Application compatibility**
- + **No frequent entering of keys**

- **Cons:**

- **Portability not supported**
- **Not flexible**
- **No user intervention**

User Level Encryption

- **Pros:**

- + **Totally under user controlled**
- + **Granularity achieved**

- **Cons:**

- **User can make mistake**
- **Incompatible with applications**
- **Frequent entering of keys**

System Level Encryption

- **Pros:**

- + **Applications become compatible**
- + **User mistakes can be overcome**
- + **Applications need not be re-written**

- **Cons:**

- **Granularity is lost**
- **Very large processing overhead**
- **No user intervention**

Cryptographic File System



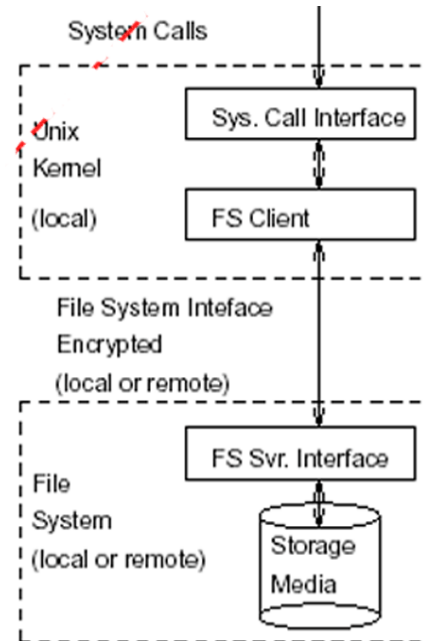
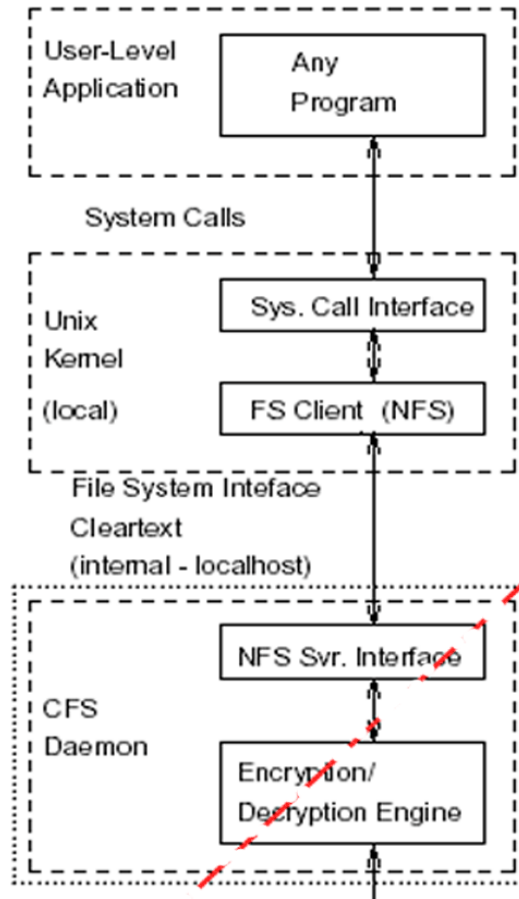
CFS - Functionality

- **Is a trade off between user level (error prone) and system level (extra overhead) encryption**
- **Provides a virtual file system on the “client’s machine”**
- **Typically mounted in `/crypt`, users access encrypted files through this directory**
- **“User’s give an attach command” to create an entry in `/crypt`**

CFS - Functionality

- **The underlying encrypted directories can reside on any file system including remote file systems**
- **Keys are supplied by the user in the form of ASCII passphrases (at least 16 characters long)**
- **Contents in `/crypt` appear in clear text to the owner of the file**

CFS - Functionality



CFS – User Interface

- **To create an encrypted directory /usr/mab/hide**
 - \$ `cmkdir /usr/mab/hide`
 - \$ **Key:** *(enter passphrase, which does not echo)*
 - \$ **Again:** *(same phrase entered again)*
- **To “use” an encrypted directory, attach the encrypted directory to a local virtual directory**
- **For example, to attach the directory created above to the name /crypt/matt perform the following,**
 - \$ `cattach /usr/mab/hide matt`
 - \$ **Key:** *(same key used in the cmkdir command)*
- **If the key is supplied correctly, the user “sees” /crypt/matt as a normal directory; all standard operations work as expected**

CFS – User Interface

- **The actual files are stored encrypted under /usr/mab/hidden, which are not used directly**
- **To create a single encrypted file**

```
$ ls -l /crypt
total 1
drwx----- 2 mab 512 Apr 1 15:56 matt
$ echo "murder" > /crypt/matt/crimes
$ ls -l /crypt/matt
total 1
-rw-rw-r-- 1 mab 7 Apr 1 15:57 crimes
$ cat /crypt/matt/crimes
murder
```

CFS – User Interface

- **List contents of /usr/mab/hide**

```
$ ls -l /usr/mab/hide
total 1
-rw-rw-r-- 1 mab 15 Apr 1 15:57
8b06e85b87091124
$ cat -v /usr/mab/hide/8b06e85b87091124
M-Z,k^ ]^B^VM-VM-6A~uM-LM-_M-DM-^ [
```

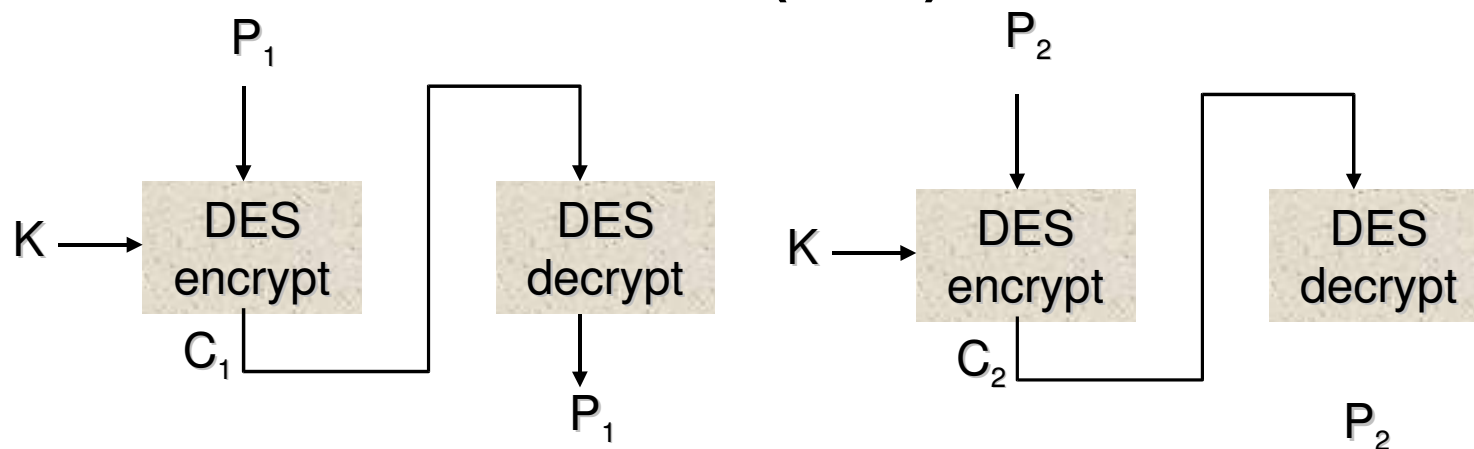
- **Only the user who issued the `cattach` command can see the clear text file**
- **After `cattach` the user does not need to provide the key, access is based on the *uid***

CFS – Encryption Scheme

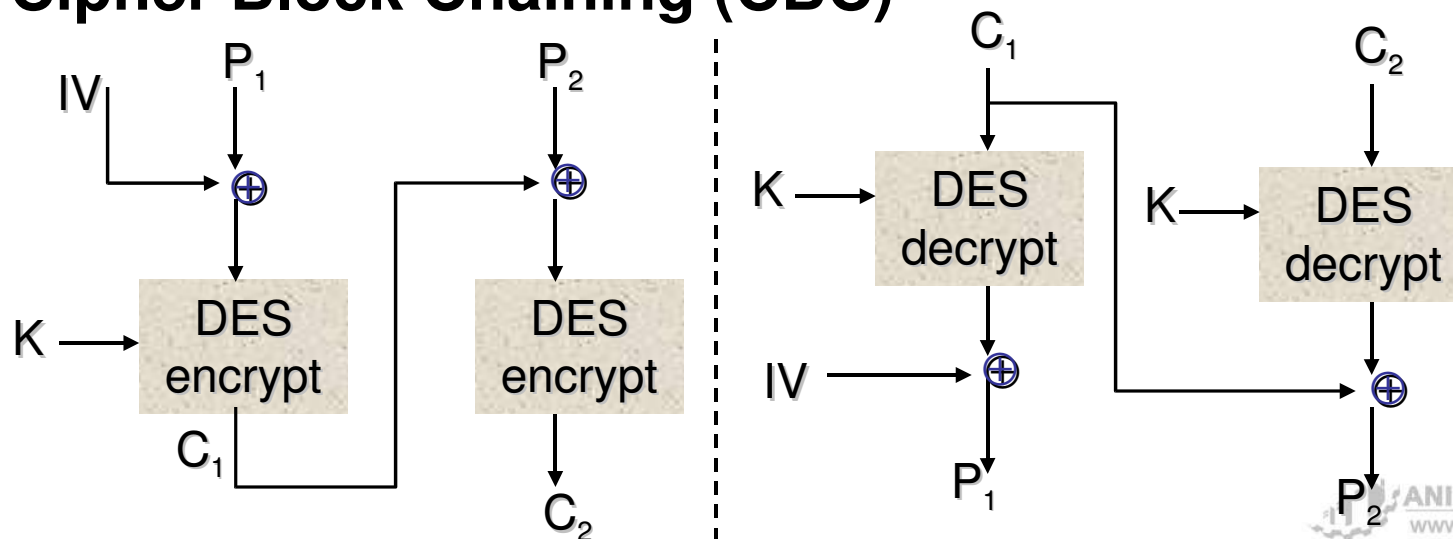
- **CFS uses DES in ECB+CBC mode**
- **Why ECB+CBC?**
 - **Only ECB mode**
 - **A given block of plaintext encrypts to the same cipher text**
 - **Chaining modes**
 - **A write to a middle of a file could require reading and re-encrypting the preceding cleartext**
 - **And then rewriting the following data**
 - **Difficult for random read or write**
 - **DES with 56-bit key encryption is vulnerable to exhaustive search of the key space**

DES Modes

- **Electronic Code Block (ECB)**



- **Cipher Block Chaining (CBC)**



Problems with CFS

- **If the user's account is compromised, the attacker can see all the attached encrypted files**
- ***User Key:* CFS relies on the user chosen key**
- ***Sharing of files***
- ***Cfsd Size:* cfsd daemon grows very fast with each attached encrypted directory.**
- ***Memory Stealing:* CFS does not talk about the key stealing from the memory.**

Smart Card



Introduction

- A credit card sized plastic card having an integrated circuit chip embedded in it and conforming to ISO – 7816 is called as Smart Card



Why Smart Card?

- **On-board microprocessor**
- **Adherence to international standards**
- **Established track record**
- **Durability and long expected life span**
- **COS that support multiple applications and secure independent data storage on one single card**

SC - CFS



Key Management

- **PIN for smart card**
- **Master key (mkey), a random number is stored in smart card**
- **Unique file key (fkey) derived from mkey is used to encrypt / decrypt the files**
 - **A hash of {inode#, timestamp, mkey} is fkey**
 - **Timestamp is the last modified time of the file**
- **Fkey changes with the associated file**

Caching

- **Two copies of cache is stored every time a file is accessed**
 - One in the CFS daemon
 - One in the local file system
- **The CFS cache copy is in clear form while the local file system cache copy is in encrypted form**

Performance Evaluation

- Using ABM standard following results were obtained

	Local FS	CFS	SC-CFS
MakeDir	0	0.2	0.2
Copy	0.6	1.0	21.8
ScanDir	1.2	1.6	1.0
ReadAll	2.0	3.0	22.6
Make	5.0	7.8	29.6

- Figures are in second with average of 5 runs

SC-CFS & CFS: A Comparative Study



Sniffing & stealing

CASE: Network sniffing or memory stealing

- **CFS**

- If user password is revealed, all the files are accessible
- In memory stealing currently opened files are accessible and remain accessible unless the key is changed explicitly

- **SC-CFS**

- If user password is revealed, without smart card no files are accessible
- In memory stealing currently opened files are accessible only until the file is closed

Storage theft & dictionary attacks

CASE: Storage theft & dictionary attacks

- **CFS**
 - CFS keys are susceptible to dictionary attacks
- **SC-CFS**
 - SC-CFS stores a random mkey in smart card, which is not susceptible to dictionary attacks
 - The hash function used to calculate fkey is not known

Conclusion

- **SC-CFS improves security (smart card + PIN)**
- **SC-CFS reduces user dependence**
- **Key theft and super user's theft is minimized**

Thank You!



Questions...

