Segurança de Sistemas e dados (MSI 2019/2020)

Aula 10

Rolando Martins

DCC – FCUP

Slides Adaptados do Prof. Manuel Eduardo Correia

Digital Rights Management

Digital Rights Management

- DRM is a good example of limitations of doing security in software
- We'll discuss
  - What is DRM?
  - A PDF document protection system
  - **DRM for streaming media**
  - **DRM in P2P application**
  - DRM within an enterprise

➔Iris

What is DRM?

- "Remote control" problem
  - Distribute digital content
  - Retain some control on its use, **after delivery**
- **Digital book** example
  - Digital book sold online could have huge market
  - But might only sell 1 copy!
  - Trivial to make perfect digital copies
  - A fundamental change from pre-digital era
- Similar comments for digital music, video, etc.
  - EME (Encrypted Media Extensions) for example is the standard that allows DRM in HTML5 video that is used by Netflix.
    - https://en.wikipedia.org/wiki/Encrypted_Media_Extensions
    - https://en.wikipedia.org/wiki/Widevine

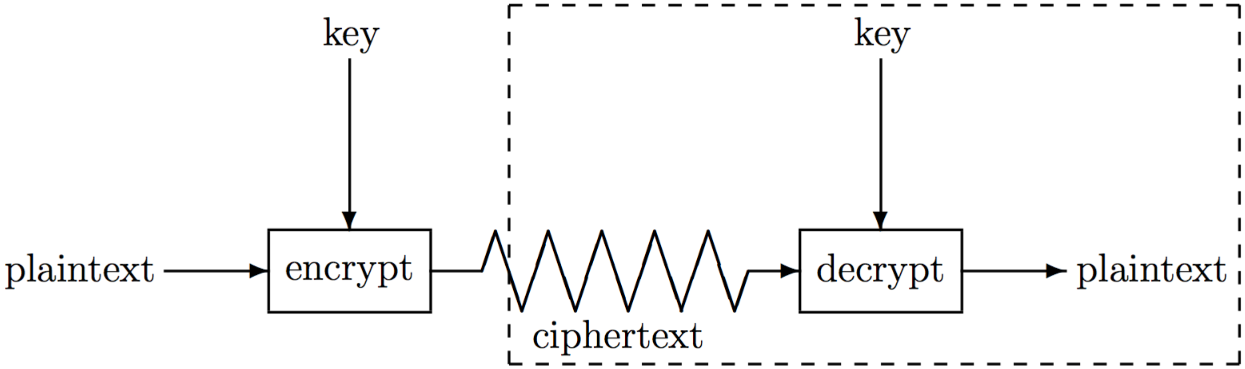
Persistent Protection

- "Persistent protection" is the fundamental problem in DRM
  - How to enforce restrictions on use of content **after** delivery?
- Examples of such restrictions
  - No copying
  - Limited number of reads/plays
  - Time limits
  - No forwarding, etc.

What Can be Done?

- The honor system?
  - Example: Stephen King's, *The Plant*
- Give up?
  - Internet sales? Regulatory compliance? etc.
- Lame software-based DRM?
  - The standard DRM system today
- Better software-based DRM?
  - MediaSnap's goal
- Tamper-resistant hardware?
  - Closed systems: Game Cube, etc.
  - Open systems: TCG/NGSCB for PCs
  - Intel SGX ➔ André Brandão's MSc Work

Is Crypto the Answer?



- Attacker's goal is to recover the **key**
- In standard crypto scenario, attacker has
  - Ciphertext, some plaintext, side-channel info, etc.
- In DRM scenario, attacker has
  - Everything in the box (at least)
- Crypto was not designed for this problem!

Is Crypto the Answer?

- But crypto is necessary
  - To securely deliver the bits
  - To prevent trivial attacks
- Then attacker will not try to directly attack crypto
- Attacker will try to find keys in software
  - DRM is "hide and seek" with keys in software!

Current State of DRM

- At best, **security by obscurity**
  - A derogatory term in security
- Secret designs
  - In violation of **Kerckhoffs Principle**
- Over-reliance on crypto
  - "Whoever thinks his problem can be solved using cryptography, doesn't understand his problem and doesn't understand cryptography." — Attributed by Roger Needham and Butler Lampson to each other

DRM Limitations

- The **analog hole**
  - When content is rendered, it can be captured in analog form
  - DRM **cannot** prevent such an attack
- **Human nature** matters
  - Absolute DRM security is impossible
  - Want something that “works” in practice
  - What works depends on context
- DRM is not strictly a technical problem!

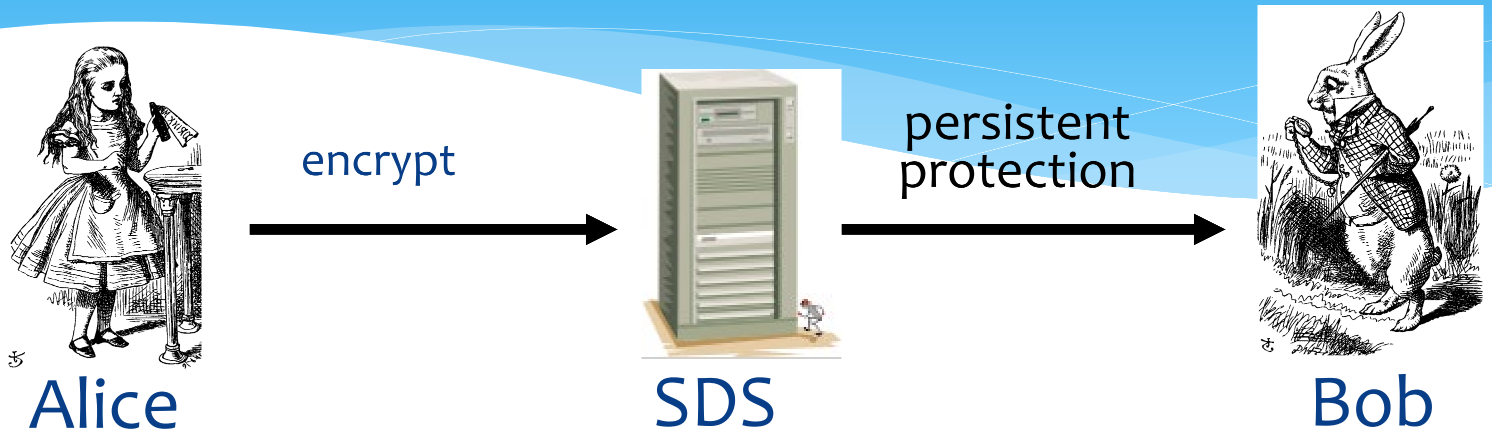
Software-based DRM

- Strong software-based DRM is impossible
- Why?
  - We can’t really hide a secret in software
  - We cannot prevent SRE
  - User with full admin privilege can eventually break any anti-SRE protection
- Bottom line: **The** killer attack on software-based DRM is SRE

DRM for PDF Documents

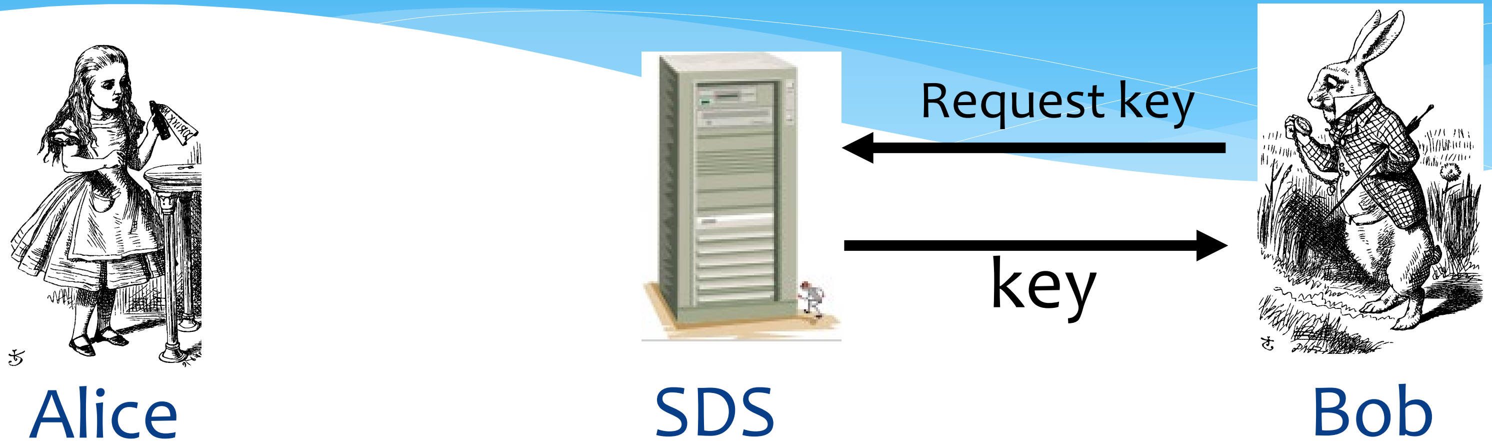
- Based on design of MediaSnap, Inc., a small Silicon Valley startup company
- Developed a DRM system
  - Designed to protect PDF documents
- Two parts to the system
  - Server — Secure Document Server (SDS)
  - Client — PDF Reader “plugin” software

Protecting a Document



- Alice creates PDF document
- Document encrypted and sent to SDS
- SDS applies desired "persistent protection"
- Document sent to Bob

Accessing a Document

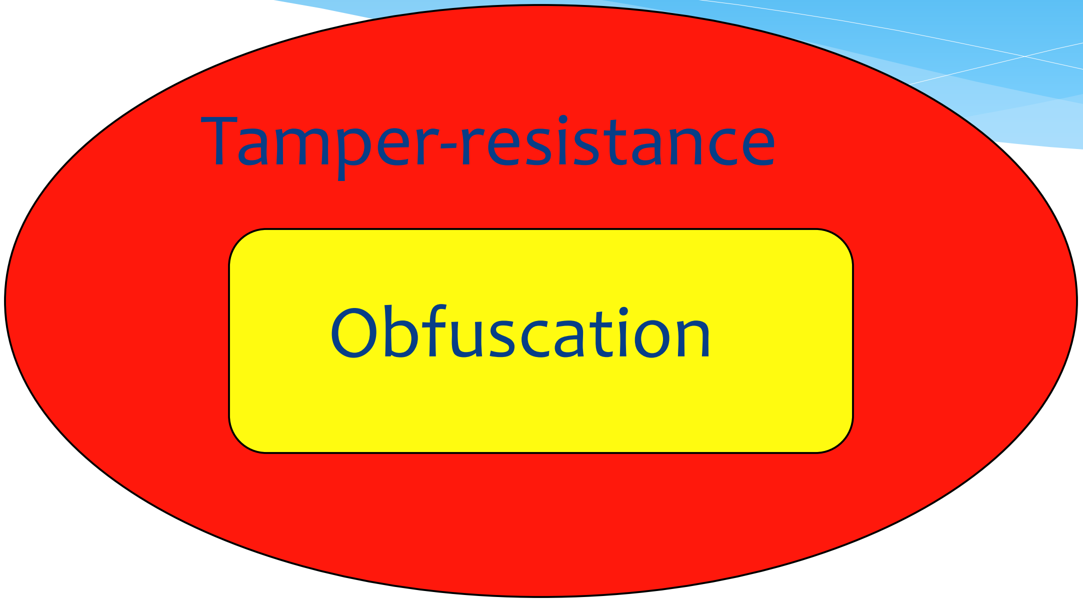


- Bob authenticates to SDS
- Bob requests key from SDS
- Bob can then access document, but only thru special DRM software

Security Issues

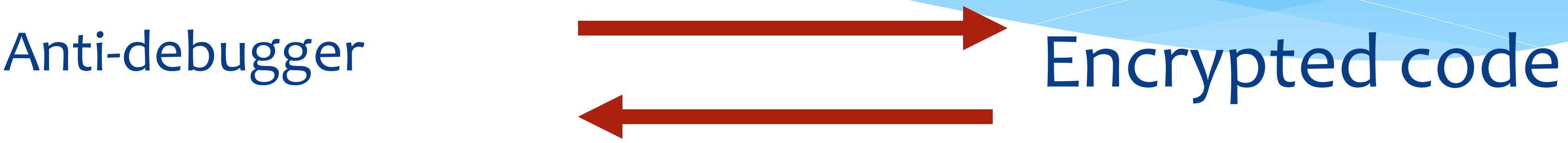
- Server side (SDS)
  - Protect keys, authentication data, etc.
  - Apply persistent protection
- Client side (PDF plugin)
  - Protect keys, authenticate user, etc.
  - Enforce persistent protection
- Remaining discussion concerns **client**

Security Overview



- A tamper-resistant outer layer
- Software obfuscation applied within

Tamper-Resistance



Anti-debugger

Encrypted code

- ❑ Encrypted code will prevent static analysis of PDF plugin software
- ❑ Anti-debugging to prevent dynamic analysis of PDF plugin software
- ❑ These two designed to protect each other
- ❑ But the persistent attacker will get thru!

Obfuscation

- Obfuscation can be used for
  - Key management
  - Authentication
  - Caching (keys and authentication info)
  - Encryption and “scrambling”
  - Key parts (data and/or code)
  - Multiple keys/key parts
- Obfuscation can only slow the attacker
- The persistent attacker still wins!

Other Security Features

- Code tamper checking (hashing)
  - To validate all code executing on system
- Anti-screen capture
  - To prevent obvious attack on digital documents
- Watermarking
  - In theory, can trace stolen content
  - In practice, of limited value
- Metamorphism (or individualization)
  - For BOBE-resistance

Security Not Implemented

- More general code obfuscation
- Code “fragilization”
  - Code that hash checks itself
  - Tampering should cause code to break
- OS cannot be trusted
  - How to protect against “bad” OS?
  - Not an easy problem!

DRM for Streaming Media

- Stream digital content over Internet
  - Usually audio or video
  - Viewed in real time
- Want to charge money for the content
- Can we protect content from capture?
  - So content can't be redistributed
  - We want to make money!

Attacks on Streaming Media

- Spoof the stream between endpoints
- Man in the middle
- Replay and/or redistribute data
- **Capture the plaintext**
  - This is the threat we are concerned with
  - Must prevent malicious software from capturing plaintext stream at client end

Design Features

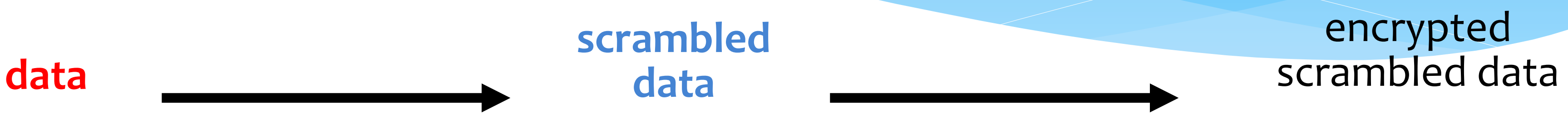
- Scrambling algorithms
  - Encryption-like algorithms
  - Many distinct algorithms available
  - A strong form of metamorphism!
- Negotiation of scrambling algorithm
  - Server and client must both know the algorithm
- Decryption at receiver end
  - To remove the strong encryption
- De-scrambling in device driver
  - De-scramble just prior to rendering

Scrambling Algorithms

- Server has a large set of scrambling algorithms
  - Suppose N of these numbered 1 thru N
- Each client has a subset of algorithms
  - For example: LIST = {12,45,2,37,23,31}
- The LIST is stored on client, encrypted with server's key: $E(\text{LIST},K_{\text{server}})$

Server-side Scrambling

- On server side



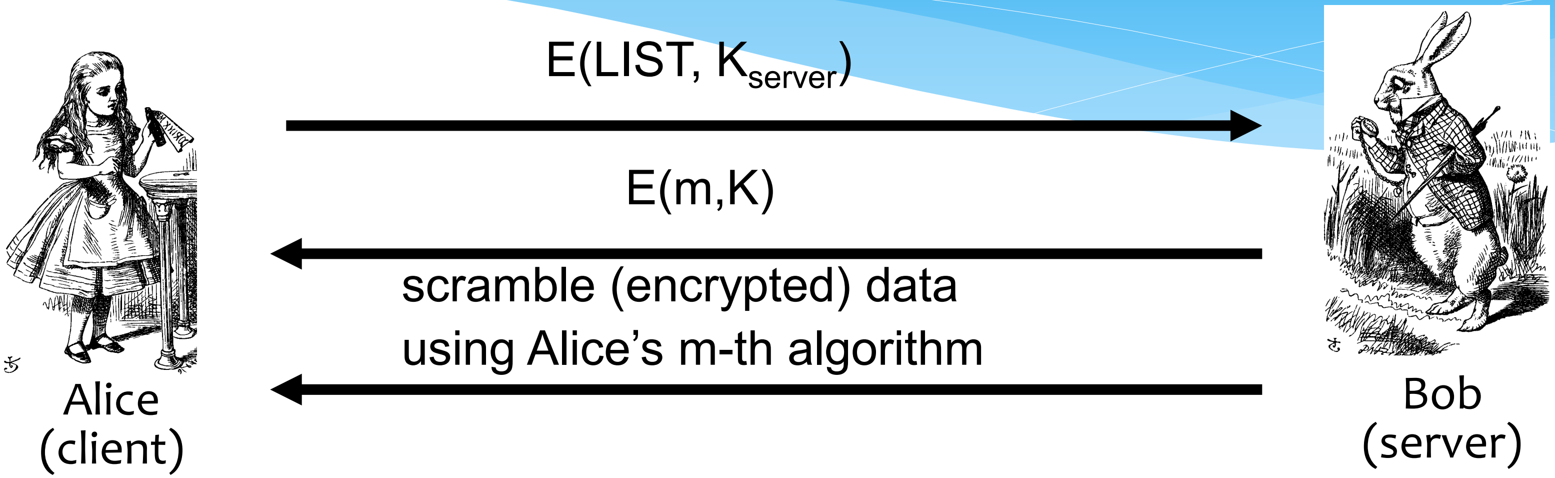
data

scrambled data

encrypted scrambled data

- □ Server must scramble data with an algorithm the client supports
- □ Client must send server list of algorithms it supports
- □ Server must securely communicate algorithm choice to client

Select Scrambling Algorithm



E(LIST, K_{server})

E(m,K)

scramble (encrypted) data
using Alice’s m-th algorithm

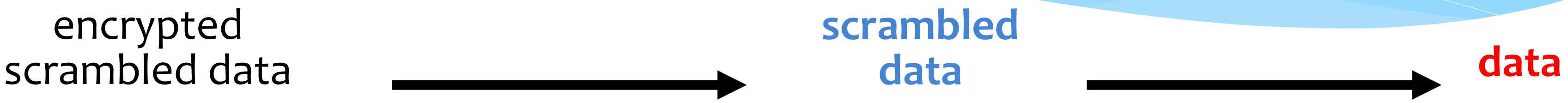
Alice
(client)

Bob
(server)

- The key K is a session key
- The LIST is unreadable by client

Client-side De-scrambling

- On client side



encrypted scrambled data

scrambled data

data

- Try to keep plaintext away from potential attacker
- "Proprietary" device driver
  - Scrambling algorithms "baked in"
  - Able to de-scramble at last moment

Why Scrambling?

- **Metamorphism** deeply embedded in system
- If a scrambling algorithm is known to be broken, server will not choose it
- If client has too many broken algorithms, server can force software upgrade
- Proprietary algorithm harder for SRE
- We cannot trust crypto strength of proprietary algorithms, so we also encrypt

Why Metamorphism?

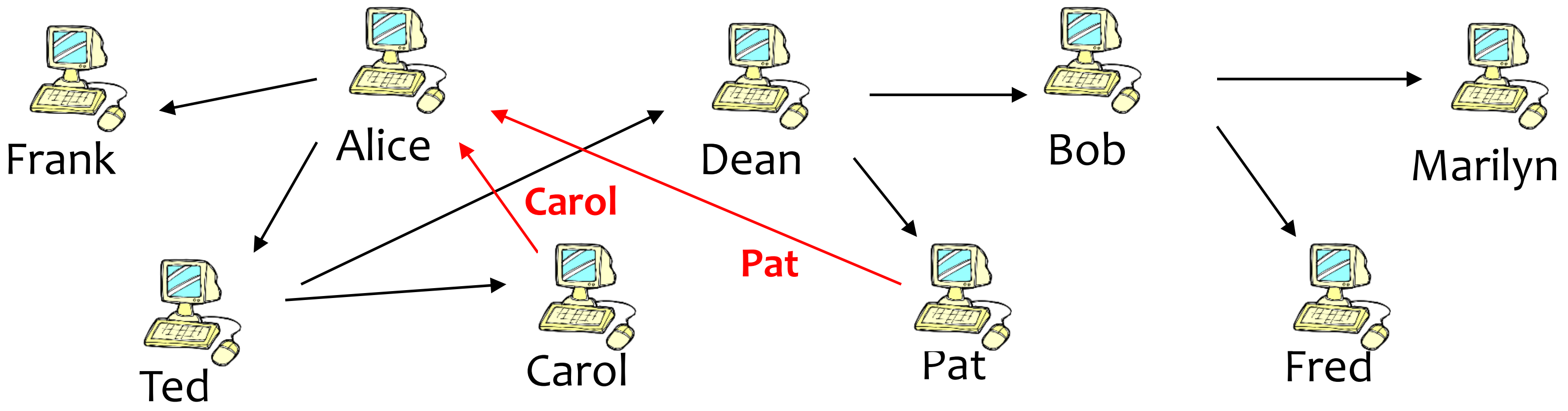
- The most serious threat is **SRE**
- Attacker does not need to reverse engineer any standard crypto algorithm
  - Attacker only needs to find the key
- Reverse engineering a scrambling algorithm may be difficult
- This is just **security by obscurity**
- But appears to help with BOBE-resistance

DRM for a P2P Application

- Today, much digital content is delivered via peer-to-peer (P2P) networks
    - P2P networks contain lots of pirated music
- Is it possible to get people to pay for digital content on such P2P networks?
- How can this possibly work?
- A peer offering service (POS) is one idea

P2P File Sharing: Query

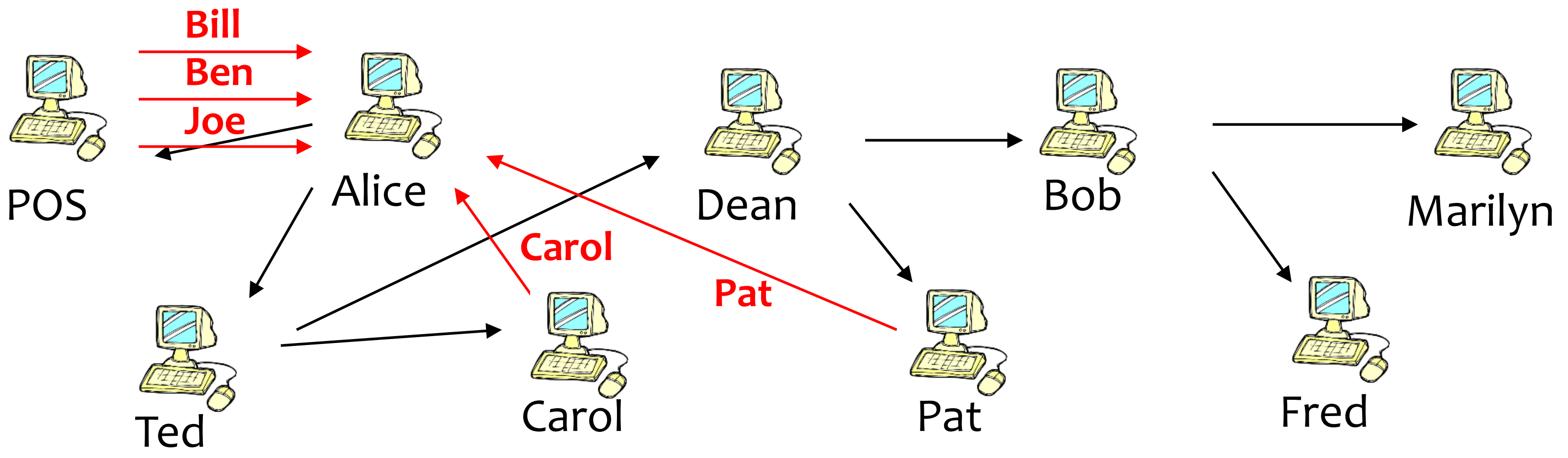
- Suppose Alice requests "Hey Jude"
- **Black** arrows: query flooding
- **Red** arrows: positive responses



□ Alice can select from: **Carol**, **Pat**

P2P File Sharing with POS

- Suppose Alice requests "Hey Jude"
- **Black** arrow: query
- **Red** arrow: positive response



- ❑ Alice selects from: **Bill**, **Ben**, **Carol**, **Joe**, **Pat**
- ❑ **Bill**, **Ben**, and **Joe** have legal content!

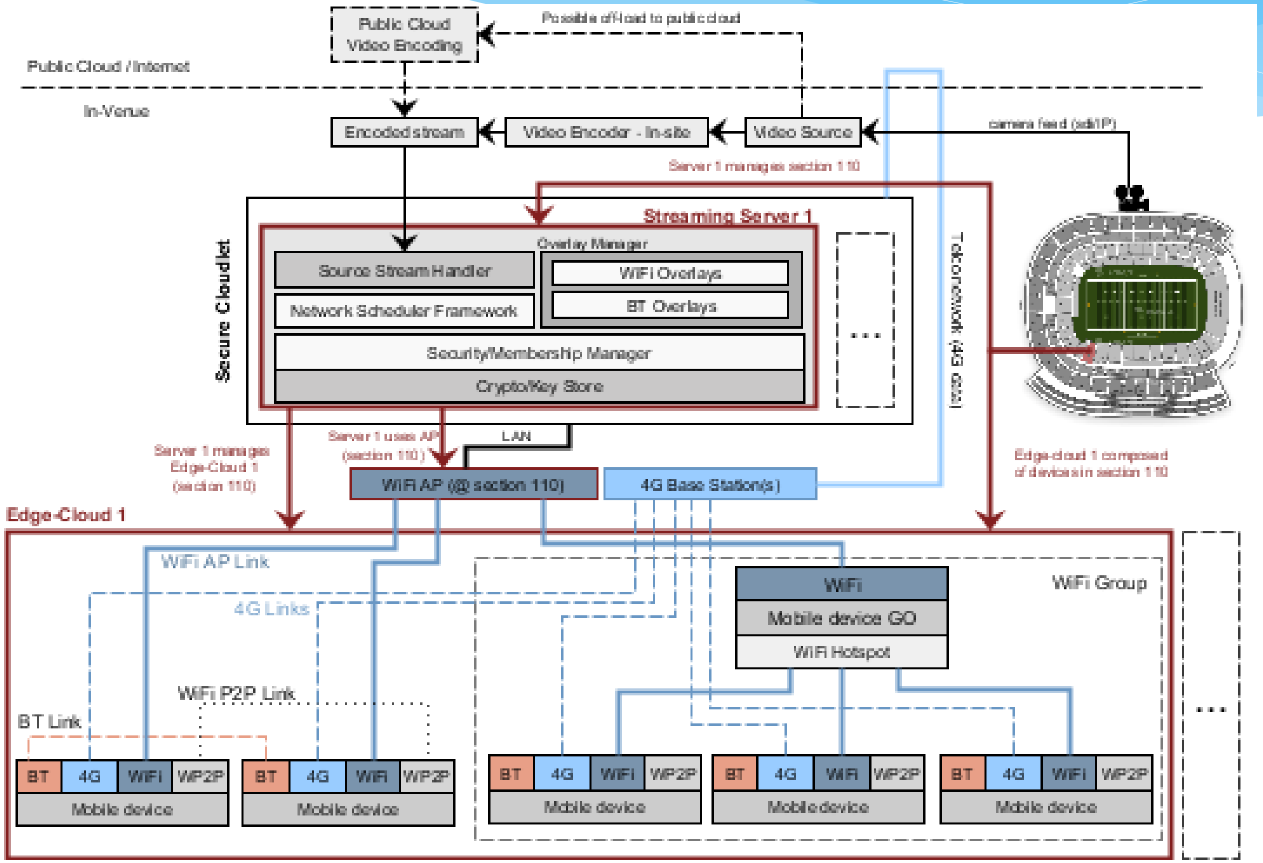
POS

- Bill, Ben and Joe must appear normal to Alice
- If “victim” (Alice) clicks POS response
  - DRM protected (legal) content downloaded
  - **Then** small payment required to play
- Alice can choose not to pay
  - But then she must download again
  - Is it worth the hassle to avoid paying small fee?
  - POS content can also offer extras

POS Conclusions

- A very clever idea!
- Piggybacking on existing P2P networks
- Weak DRM works very well here
  - Pirated content already exists
  - DRM only needs to be more hassle to break than the hassle of clicking and waiting
- Current state of POS?
  - Very little interest from the music industry
  - Considerable interest from the “adult” industry

Iris


Public Cloud
Video Encoding
Possible off-load to public cloud
Public Cloud / Internet
In-Venue
Encoded stream
Video Encoder - In-site
Video Source
camera feed (sdi/IP)
Server 1 manages section 110
Streaming Server 1
Secure Cloudlet
Overlay Manager
Source Stream Handler
WiFi Overlays
Network Scheduler Framework
BT Overlays
Security/Membership Manager
Crypto/Key Store
. . .
Telco network (4G data)
Server 1 manages
Edge-Cloud 1
(section 110)
Server 1 uses AP
(section 110)
LAN
WiFi AP (@ section 110)
4G Base Station(s)
Edge-cloud 1 composed
of devices in section 110
Edge-Cloud 1
WiFi AP Link
4G Links
WiFi P2P Link
BT Link
WiFi Group
WiFi
Mobile device GO
WiFi Hotspot
BT
4G
WiFi
WP2P
Mobile device
BT
4G
WiFi
WP2P
Mobile device
BT
4G
WiFi
WP2P
Mobile device
BT
4G
WiFi
WP2P
Mobile device
BT
4G
WiFi
WP2P
Mobile device
. . .

DRM in the Enterprise

- Why enterpise DRM?
- Health Insurance Portability and Accountability Act (HIPAA)
  - Medical records must be protected
  - Fines of up to $10,000 “per incident”
  - GDPR in effect on 25th May 2018 – Even heavier fines.
- Sarbanes-Oxley Act (SOA)
  - Must preserve documents of interest to SEC
- DRM-like protections needed by corporations for **regulatory compliance**

What’s Different in Enterprise DRM?

- Technically, similar to e-commerce
- But motivation for DRM is different
  - Regulatory compliance
  - To satisfy a legal requirement
  - Not to make money — to avoid losing money!
- Human dimension is completely different
  - Legal threats are far more plausible

Enterprise DRM

- Moderate DRM security is sufficient
- **Policy management issues**
  - Easy to set policies for groups, roles, etc.
  - Yet policies must be flexible
- **Authentication issues**
  - Must interface with existing system
  - Must prevent network authentication spoofing (authenticate the authentication server)
- Enterprise DRM is a solvable problem!

DRM Failures

- Many examples of DRM failures
  - One system defeated by a felt-tip pen
    - http://www.berkeleydailyplanet.com/issue/2002-05-31/article/12308?headline=Sony-s-CD-protection-method-foiled-with-a-felt-tip-pen&status=301
  - One defeated my holding down shift key
  - Secure Digital Music Initiative (SDMI) completely broken before it was finished
  - Adobe eBooks
  - Microsoft MS-DRM (version 2)
  - Many, many others!

DRM Conclusions

- DRM nicely illustrates **limitations of doing security in software**
- Software in a hostile environment is extremely vulnerable to attack
- Protection options are very limited
- Attacker has enormous advantage
- Tamper-resistant hardware and a trusted OS can make a difference
  - We'll discuss this more later: TCG/NGSCB

Secure Software Development

Penetrate and Patch

- Usual approach to software development
  - Develop product as quickly as possible
  - Release it without adequate testing
  - Patch the code as flaws are discovered
- In security, this is "penetrate and patch"
  - A **bad** approach to software development
  - A **horrible** approach to secure software!

Why Penetrate and Patch?

- First to market advantage
  - First to market likely to become market leader
  - Market leader has huge advantage in software
  - Users find it safer to "follow the leader"
  - Boss won't complain if your system has a flaw, as long as everybody else has the same flaw
  - User can ask more people for support, etc.
- Sometimes called "network economics"

Why Penetrate and Patch?

- Secure software development is hard
  - Costly and time consuming development
  - Costly and time consuming testing
  - Easier to let customers do the work!
- No serious economic disincentive
  - Even if software flaw causes major losses, the software vendor is not liable
  - Is any other product sold this way?
  - Would it matter if vendors were legally liable?

Penetrate and Patch Fallacy

- **Fallacy:** If you keep patching software, eventually it will be secure
- Why is this a fallacy?
  - Empirical evidence to the contrary
  - Patches often add new flaws
  - Software is a moving target due to new versions, features, changing environment, new uses, etc.