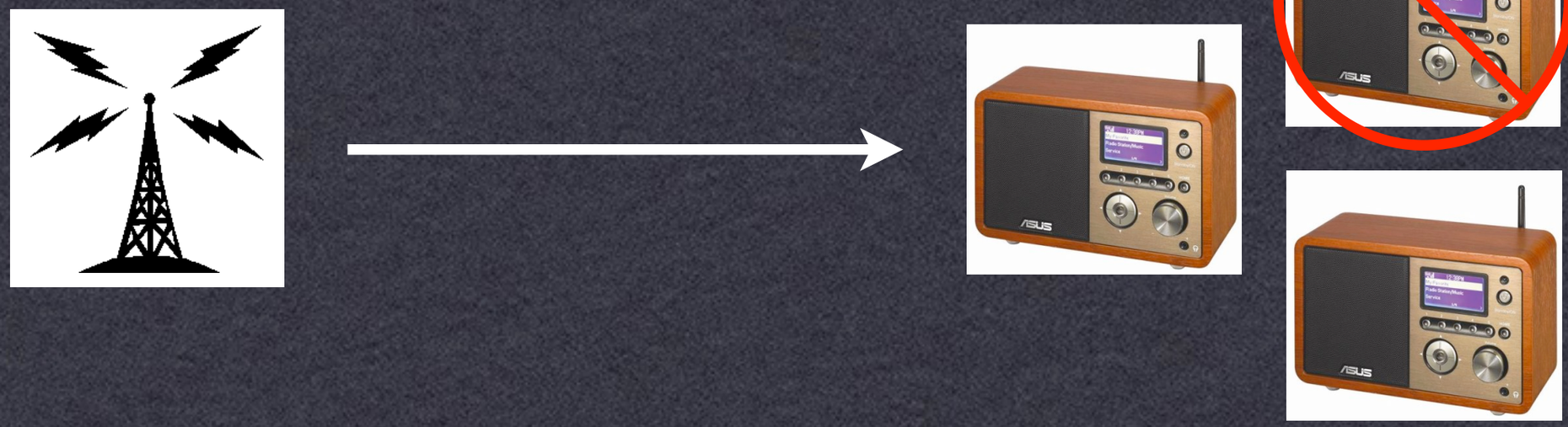
650.445: Practical Cryptographic Systems

Digital Rights Management II

Instructor: Matthew Green

Per-player keys

- **Broadcast Encryption**
  - **Single sender, many receivers**
  - **Only authorized receivers can decrypt**
  - **The set of authorized receivers can change (receivers may be revoked)**
- **Stateful & Stateless / Efficient**

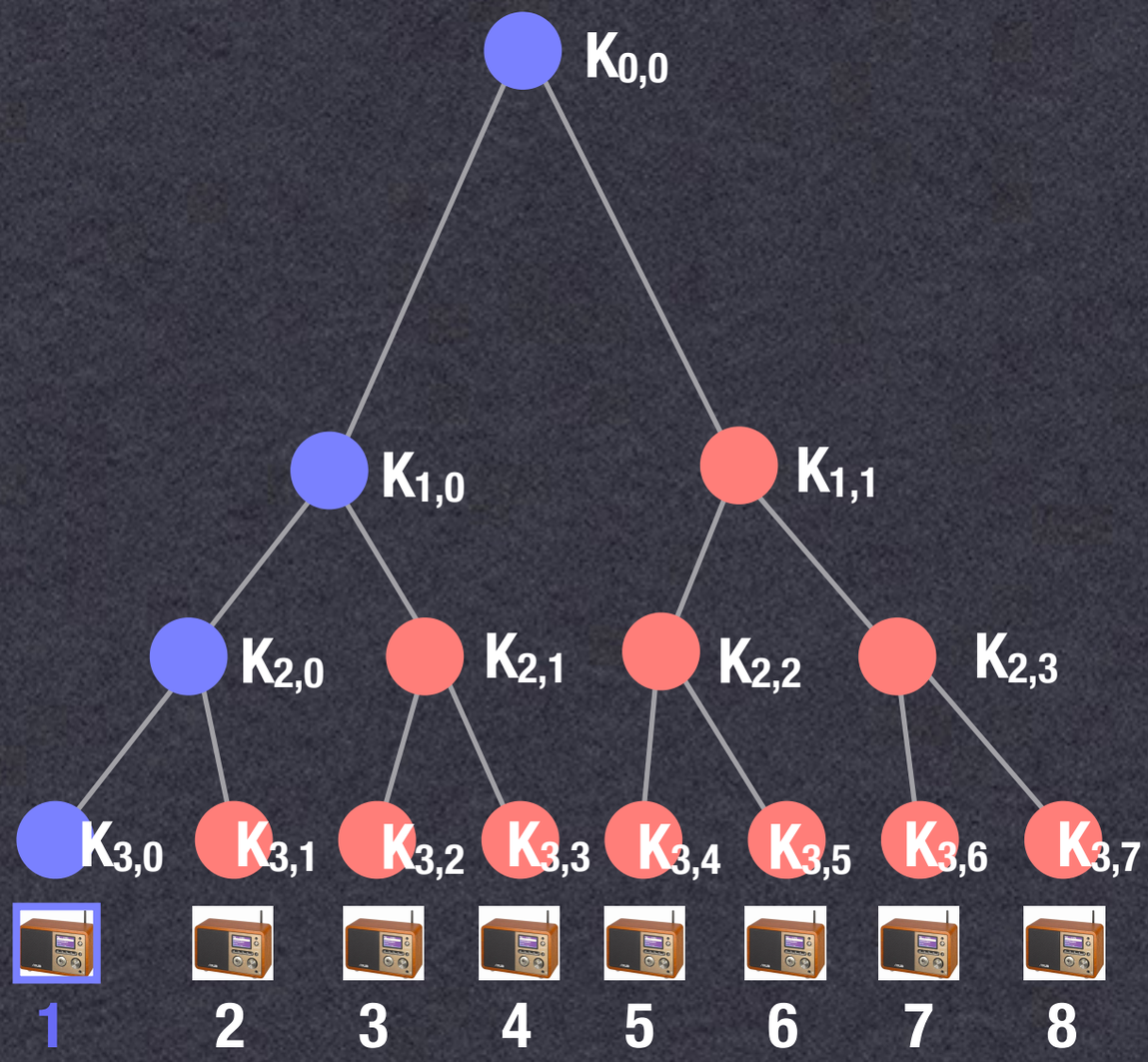


Review: NNL Subset Cover

$K_{0,0}$ $K_{1,0}$ $K_{2,0}$ $K_{3,0}$

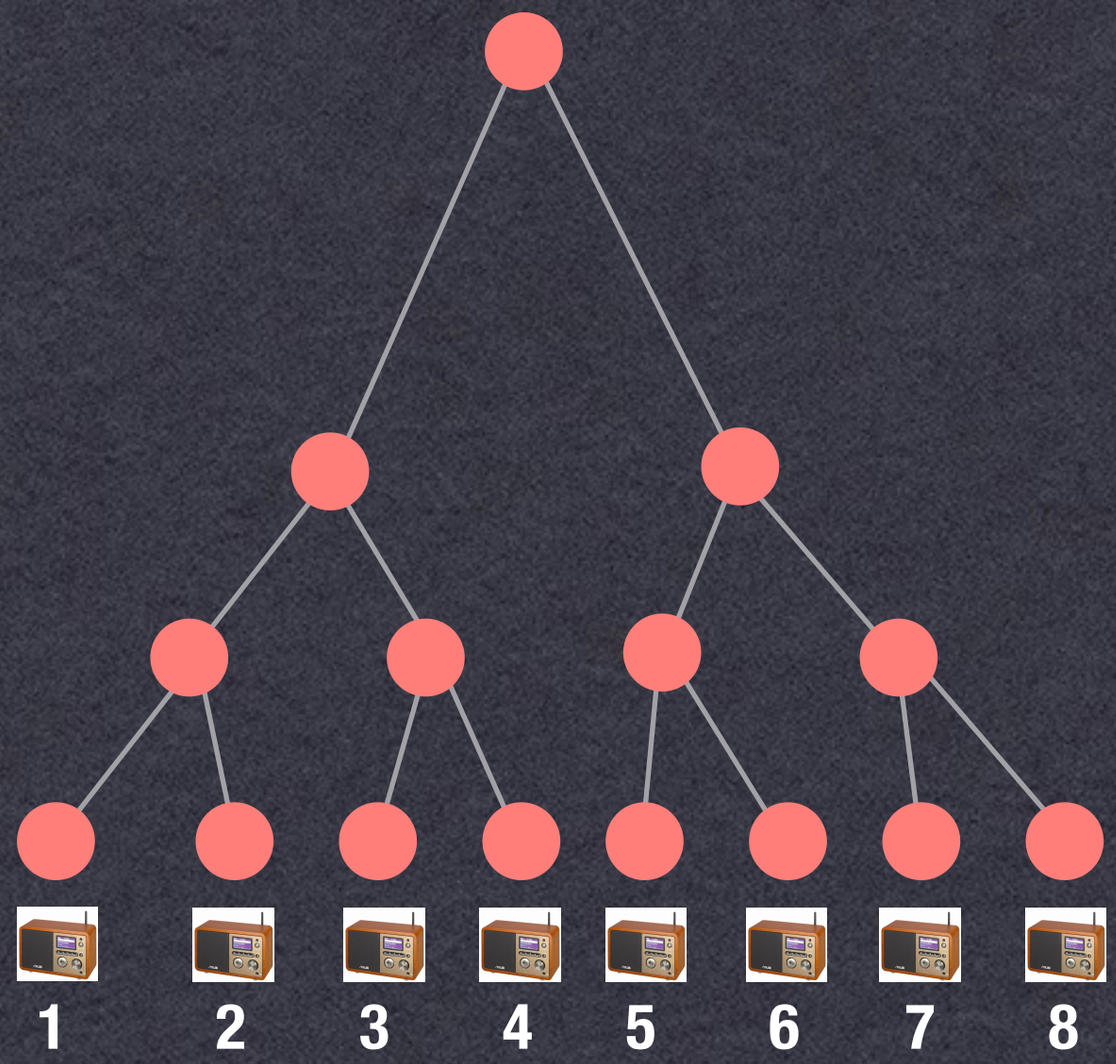
1

$$r \log(N/r)$$



NNL Subset Difference

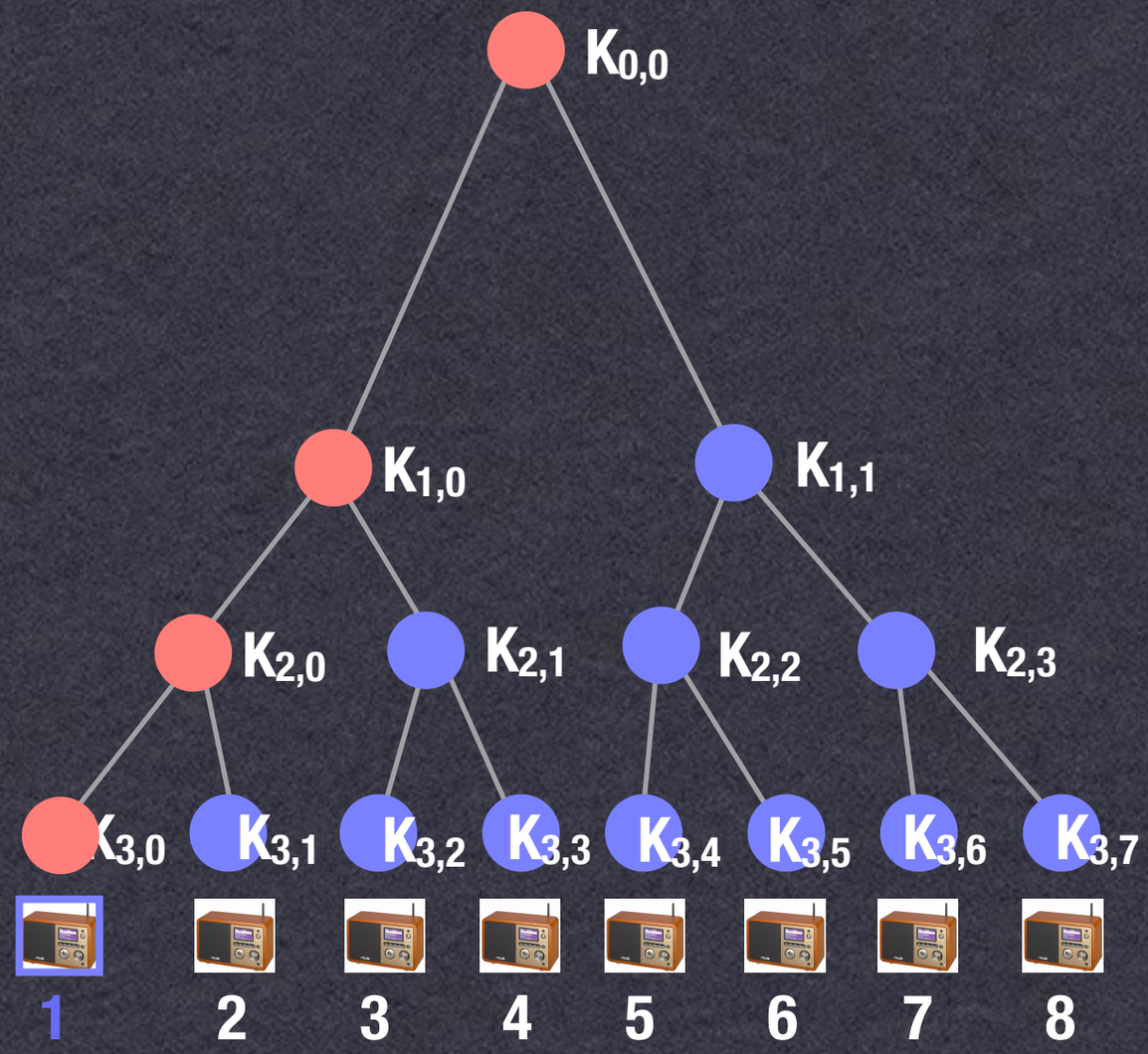
$2r - 1$



NNL Subset Difference

1

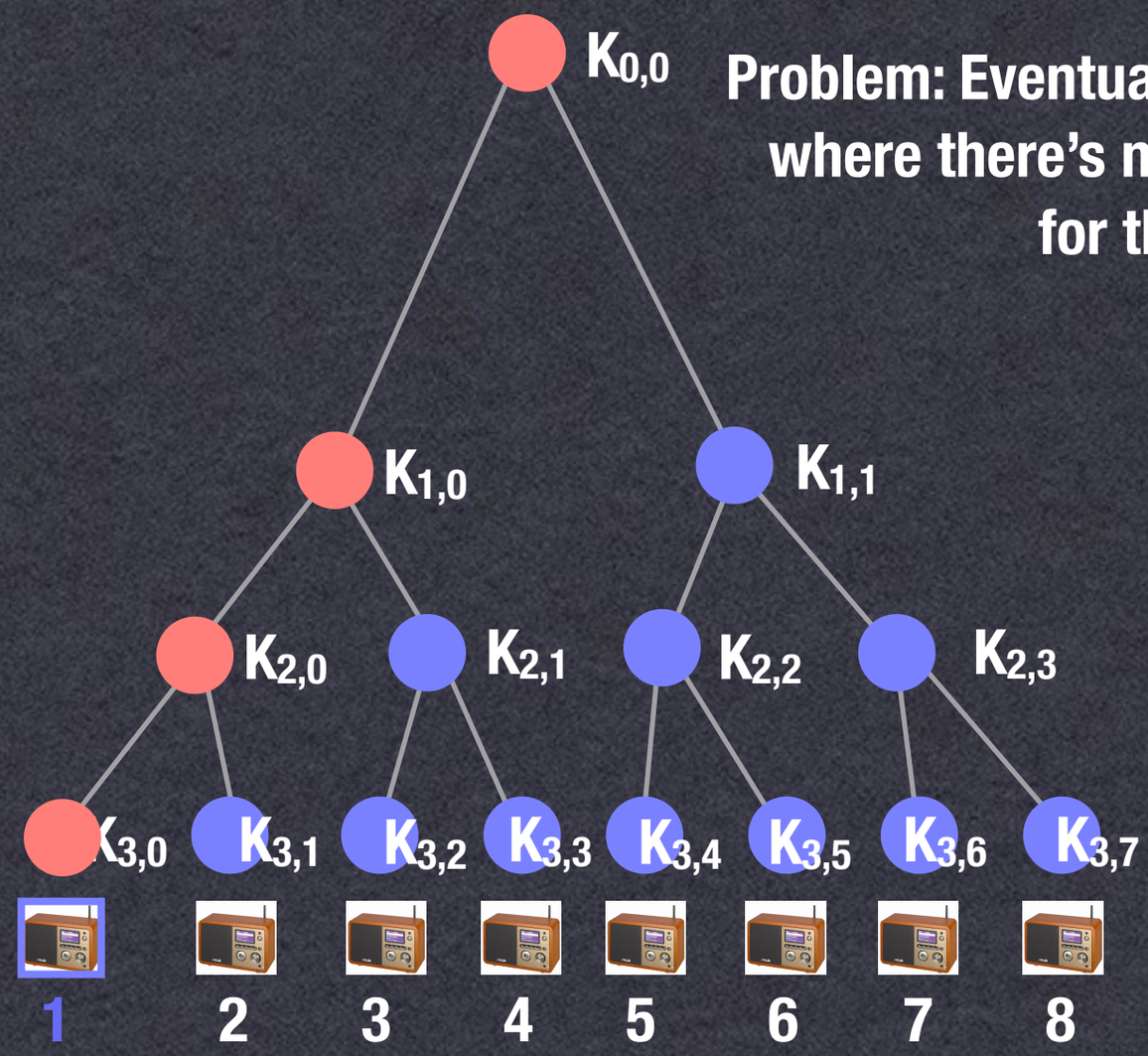
All keys <u>except:</u> $K_{0,0}$ $K_{1,0}$ $K_{2,0}$ $K_{3,0}$



NNL Subset Difference

1

All keys <u>except:</u> $K_{0,0}$ $K_{1,0}$ $K_{2,0}$ $K_{3,0}$

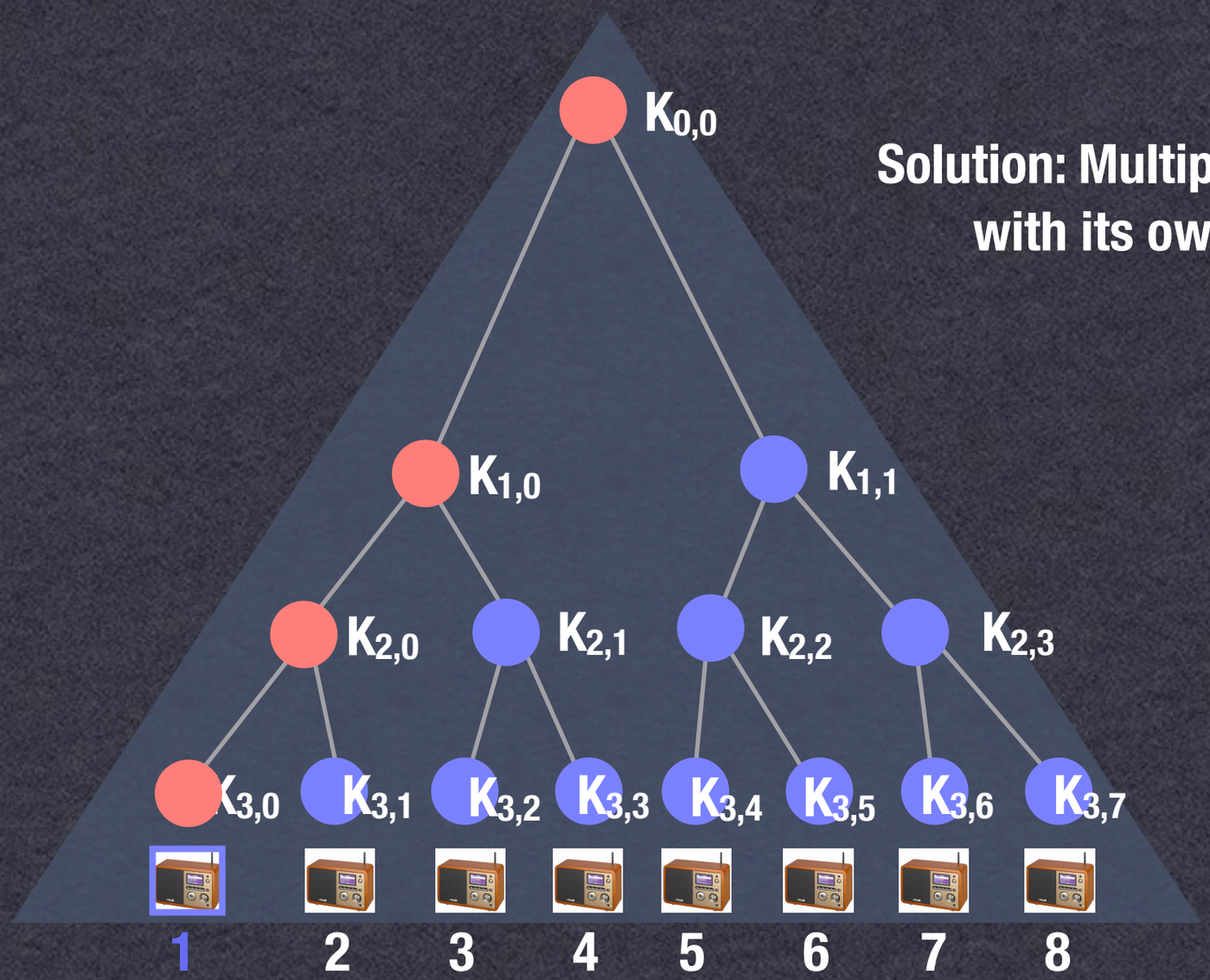


Problem: Eventually you run into situations where there's no set of keys that works for this given tree

NNL Subset Difference

1

All keys <u>except:</u> $K_{0,0}$ $K_{1,0}$ $K_{2,0}$ $K_{3,0}$

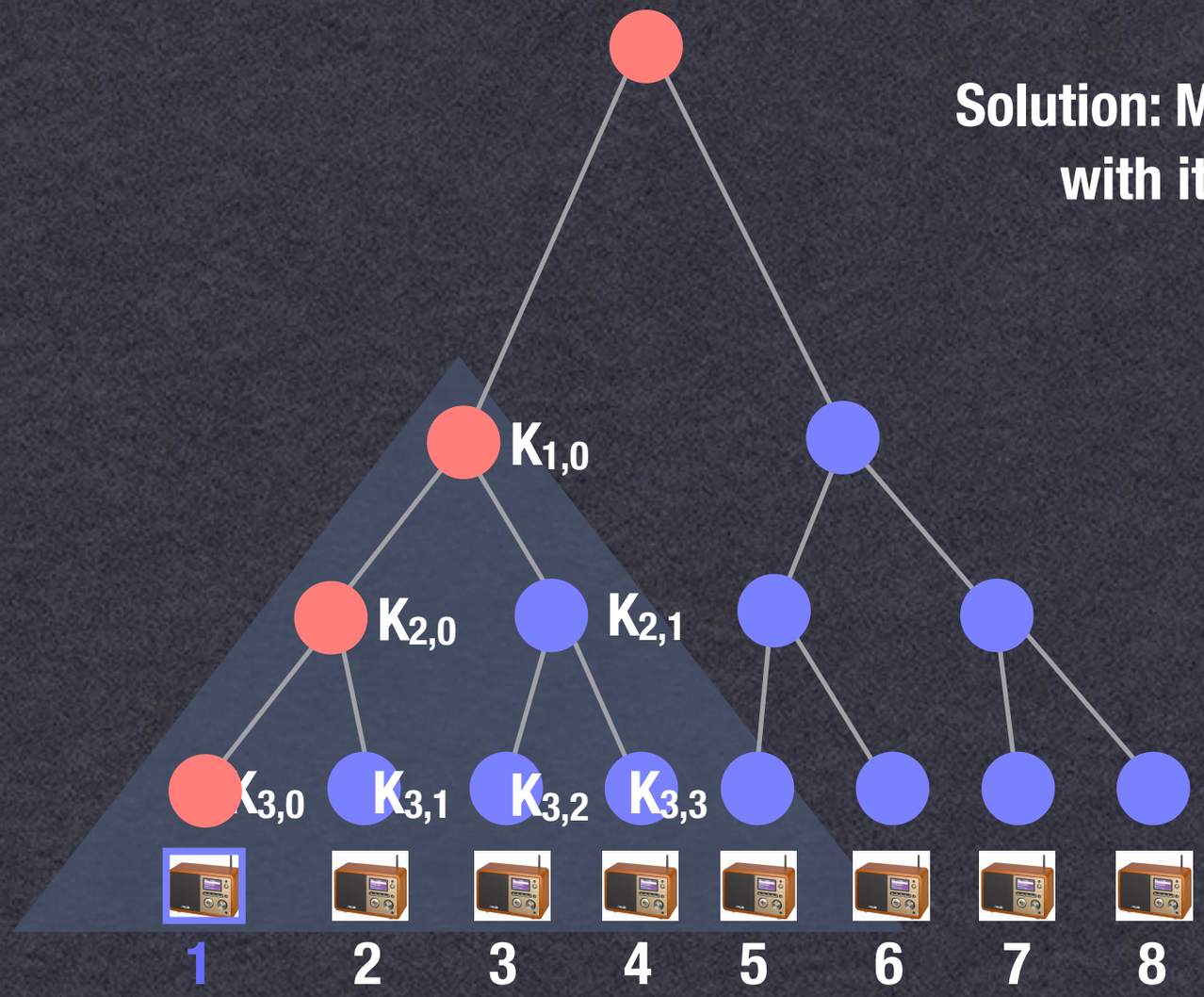


Solution: Multiple subtrees, each with its own set of keys

NNL Subset Difference

1

All keys <u>except:</u> $K_{0,0}$ $K_{1,0}$ $K_{2,0}$ $K_{3,0}$

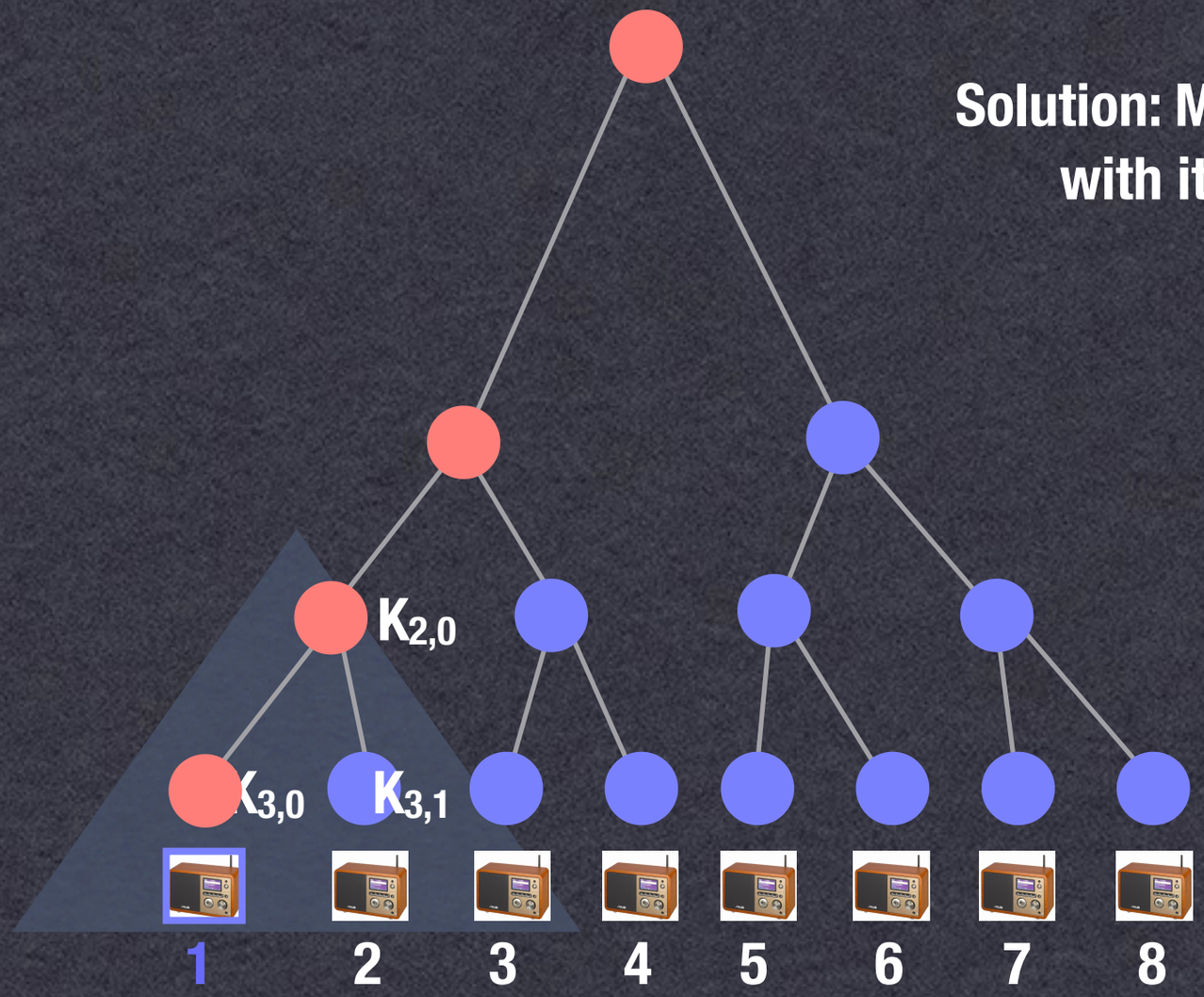


Solution: Multiple subtrees, each with its own set of keys

NNL Subset Difference

1

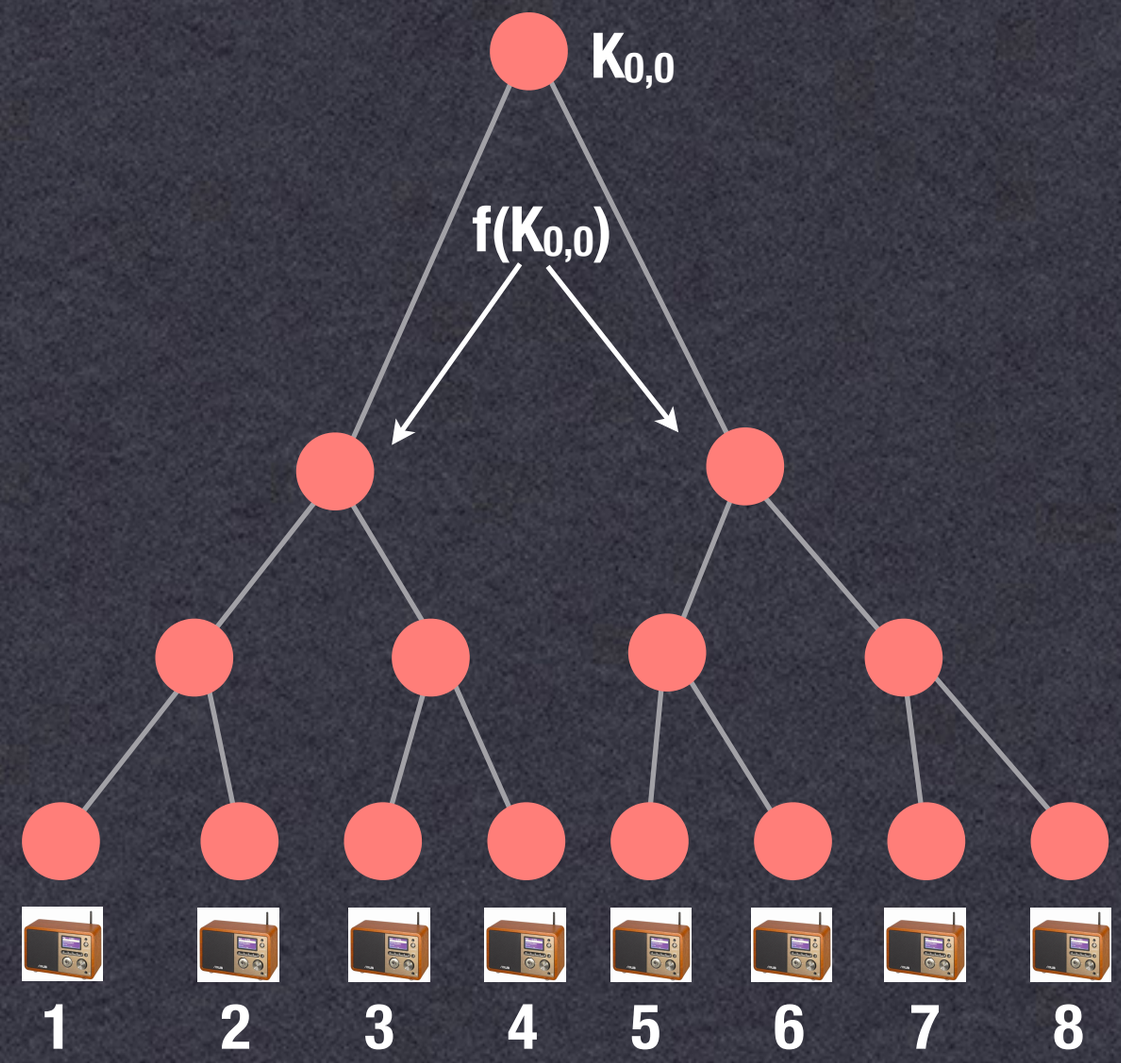
All keys <u>except:</u> $K_{0,0}$ $K_{1,0}$ $K_{2,0}$ $K_{3,0}$



Solution: Multiple subtrees, each with its own set of keys

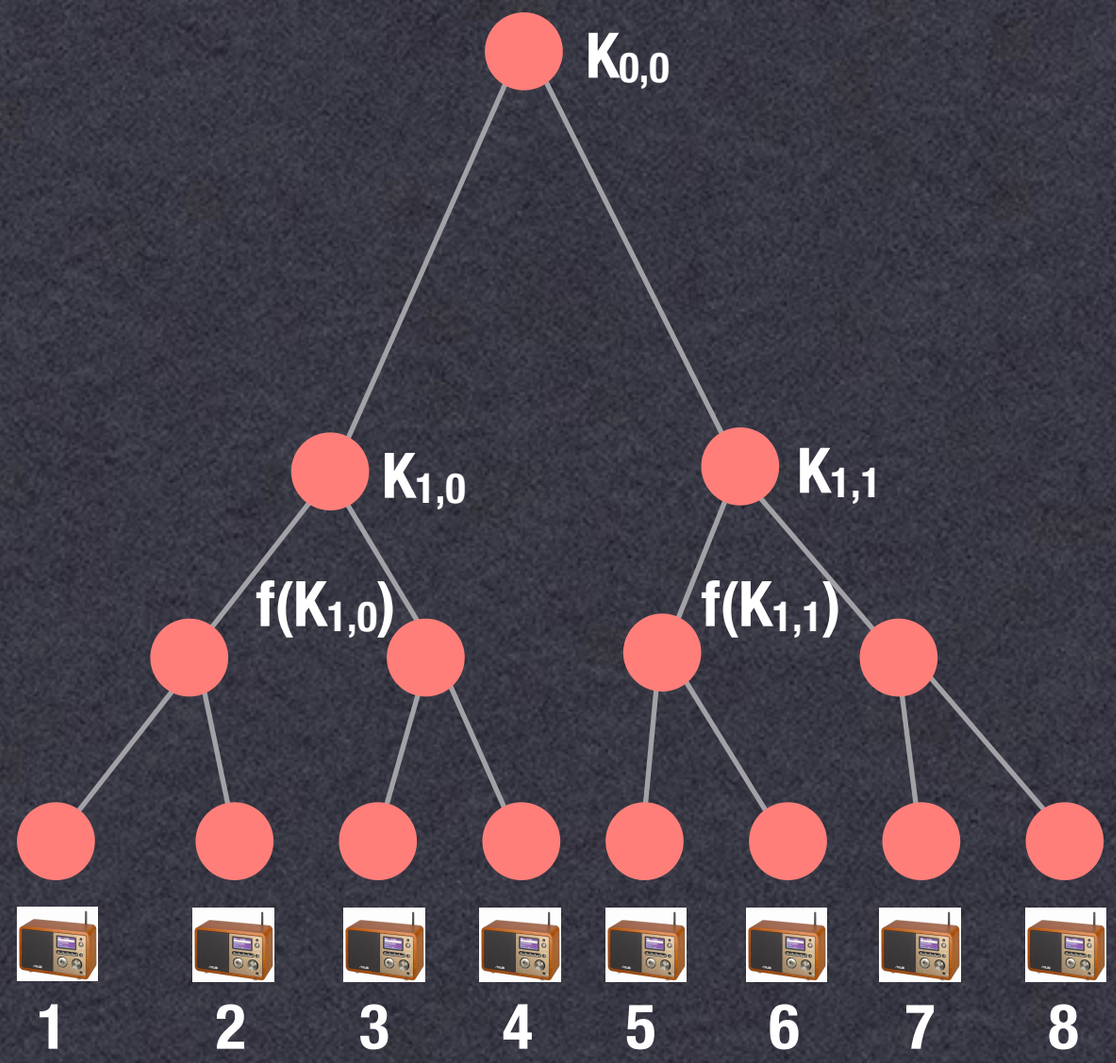
NNL Subset Difference

f(K) = (K1, K2): pseudo-random key derivation function

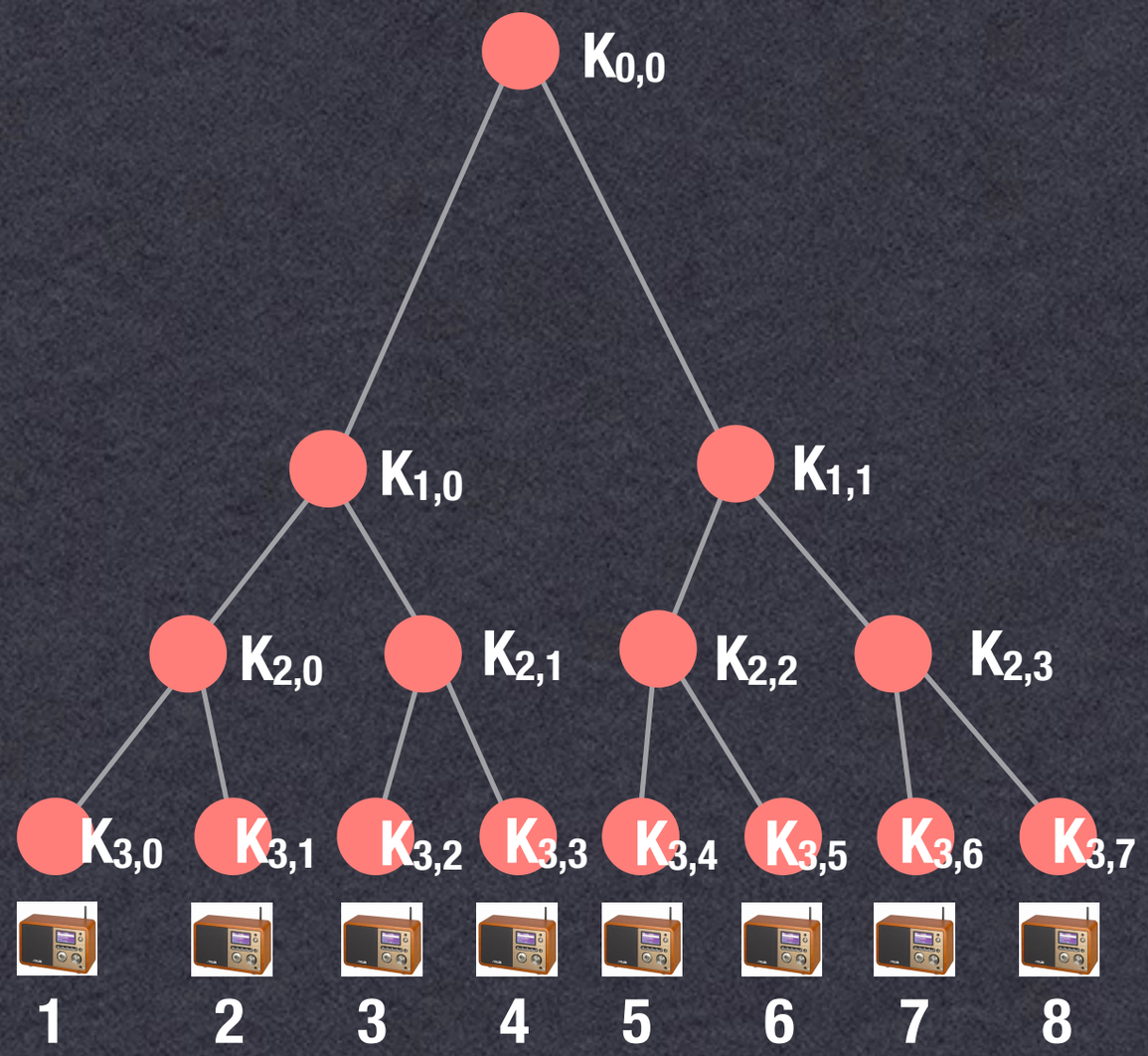


NNL Subset Difference

f(K) = (K1, K2): pseudo-random key derivation function



NNL Subset Difference



Content Authentication

- All AACS disks protected with EC-DSA signatures (controlled by AACS-LA):
  - However, there's a lot of content!
  - Can't verify the whole disk...

-At startup, verify a random sampling of blocks

-Use a Merkle Hash Tree to verify

Watermarking

- Steganographic marking of content
- Two purposes:
  - Content-oriented watermarks

-Used to identify copyrighted material (think YouTube), control playback

Availability of public verification oracle

  - User-oriented watermarks

-Embedded in the decoded content by the player

-Used to identify the player (think BitTorrent)

-Can also watermark keys

Watermarking

- AACS supports both:
  - Playback controls (audio watermark)

-Prevents ripping/recording of movies

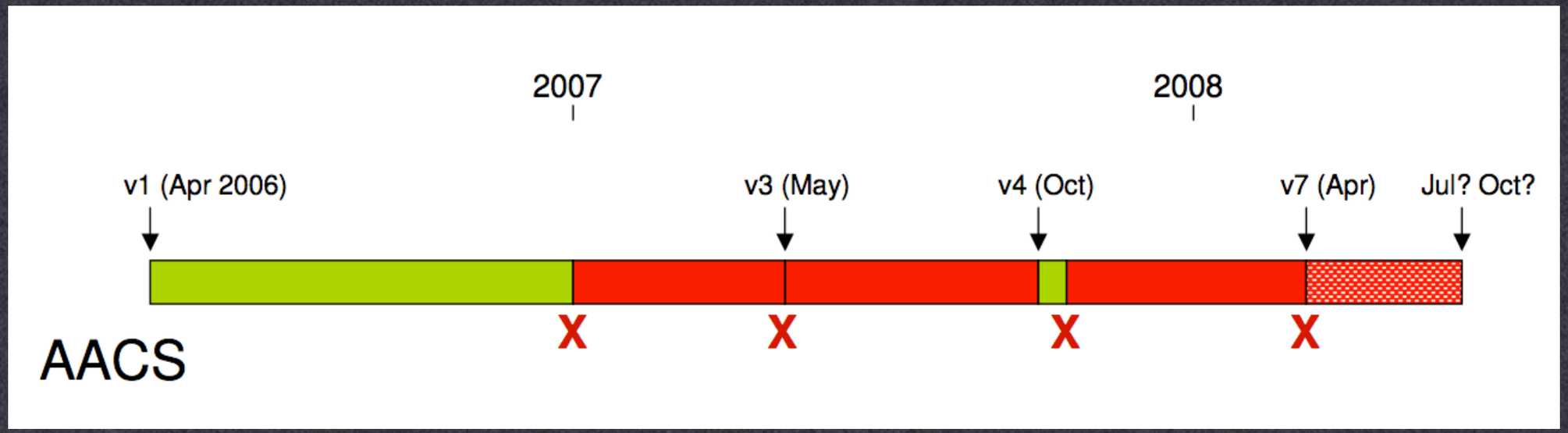
  - <u>Scene variants</u>:

-Multiple small chunks of content come in "variants". By inserting these into the material according to player ID, ID player that ripped content

-Problems?

Success?

- How has AACS stood up?



Source: Nate Lawson

arnezami
Registered User

Join Date: Sep 2006
Posts: 389

Wooow. I think I did it .

Processing Key found!!!

More info later.

To be sure I need to confirm my finding. I need the following (from anybody with a HD DVD disc):

1) - Movie Title (not King Kong please)

2) - The *Verify Media Key Record* in the MKBROM.AACS file. It starts with 81 00 00 14 followed by the 16 byte Record. In my cas

Code:

```
81 00 00 14 87 B8 A2 B7 C1 0B 9F AD F8 C4 36 1E 23 86 59 E5 7F 00 00 xx
```

3) - The first C-Value in the MKBROM.AACS file (also called Media Key Data). It starts with 05 00 20 14 (the 20 14 could be diffe first 16 byte C-Value. In my case this it at Offset 00004376h. Here is mine:

Code:

```
05 00 20 14 6D 02 CA C6 7B 1A 7E 95 C2 16 EF D4 C9 28 09 CF D3 CE 9A DC
```

If you react quickly I can check if the Processing Key is really valid (for multiple discs).

Yeah I'm happy...

arnezami

Success?

- Lessons learned:
  - Problem so far is software players

-Same as for CSS

-Only real protection is code obfuscation, anti-debugging

-But these tend to fail eventually

-Necessitates constant revocation, firmware updates

Success?

- Additional lessons:
  - Even if you have renewability, getting hacked is bad PR

The Solution?

- BD+
  - BluRay drives include an additional renewability mechanism
  - Based on SPDC spec, cryptography research
  - Owned by Macrovision
- Adds additional renewability through obfuscated code
  - Code is shipped on disk, runs in a VM
  - Validates player, can watermark, etc.

Code obfuscation

- A digression:
  - Is it possible to obfuscate code securely?
  - I.e., create code that can run in an untrusted environment

-Classic example is DRM

-Also useful protecting trade secrets

-Goal: protect the algorithm

Computing on Enc. Data

- Perform computation on ciphertexts
  - Computation never reveals underlying data
  - (Ciphertext in, ciphertext out)
  - Outsource computation w/o loss of privacy

Amazon Elastic Compute Cloud (Amazon EC2)

Google™

Code obfuscation

- In real life:
  - Anti-debugging, shuffle instructions, jump tables, disguised constants, etc. etc.
  - Example: most software DRM, proprietary algorithms, Hulu

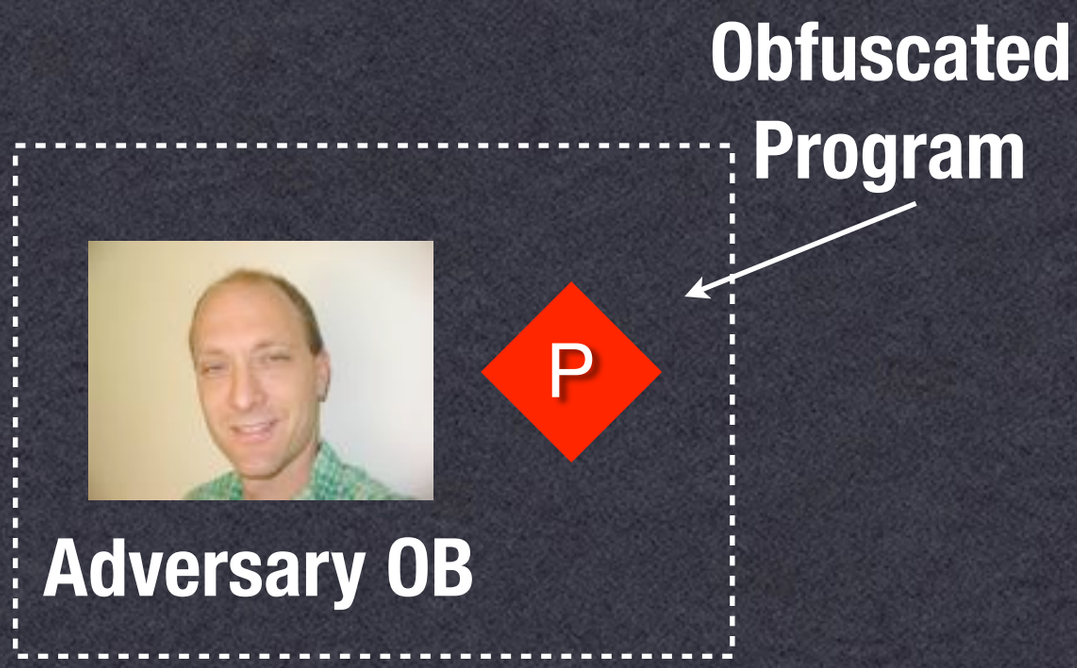
```
#include<stdio.h> #include<string.h>
main(){char*O,l[999]="’‘acgo\177~|xp .
-\OR^8)NJ6%K4O+A2M(*OID57$3G1FBL";
while(O=fgets(l+45,954,stdin)){*l=O[
strlen(O)[O-1]=0,strspn(O,l+11)];
while(*O)switch((*l&&isalnum(*O))-!*l)
{case-1:{char*I=(O+=strspn(O,l+12)
+1)-2,O=34;while(*I&3&&(O=(O-16<<1)+
*I---’-’)<80);putchar(O&93?*I
&8||!(  I=memchr( l , O , 44 ) ) ?’?’:
I-l+47:32); break; case 1: ;}*l=
(*O&31)[l-15+(*O>61)*32];while(putchar
(45+*l%2),(*l=*l+32>>1)>35); case 0:
putchar((++O ,32));}putchar(10);}}
```

Cryptographic obfuscation

- Cryptographers:
  All this code obfuscation stuff is a joke
- Can we do obfuscation securely?
  - What does that even mean?
    (Cryptographers care about definitions...)
  - Once we figure that out, is it possible?

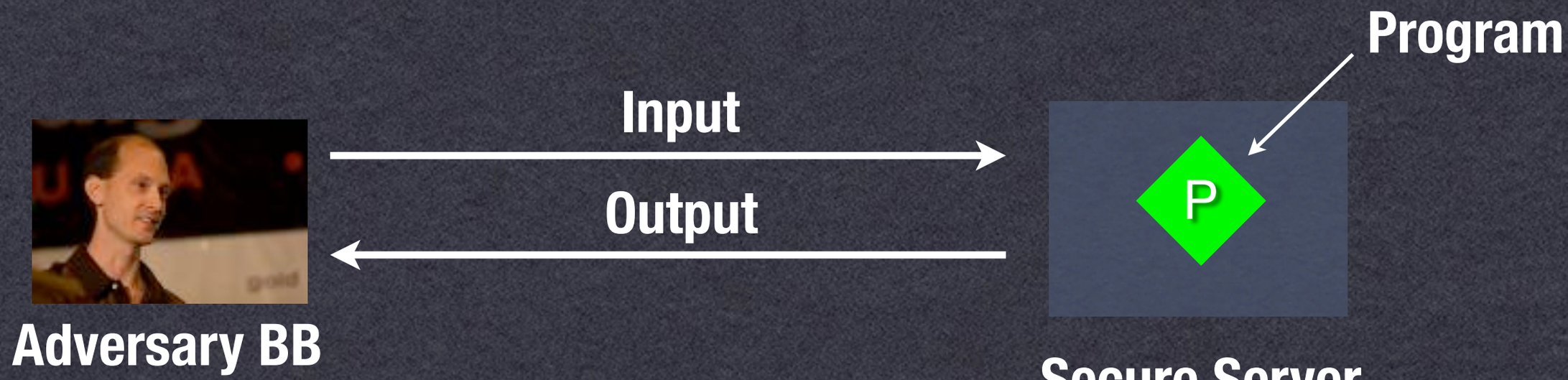
Black-box Obfuscation

- **Intuition:**
  - **With obfuscation, the attacker sees the (obfuscated) program**



Black-box Obfuscation

- In a perfect world:
  - We wouldn't have to give out the program
  - In other words, we'd run it in a secure server ("black box" access)



Black-box Obfuscation

- Therefore, for secure obfuscation:
  - We give Black Box access to an attacker (1)
  - We let him play with the Obfuscated Program (2)
  - He must learn <u>nothing more</u> in case (2) than in case (1)

Adversary BB

?

Adversary OB

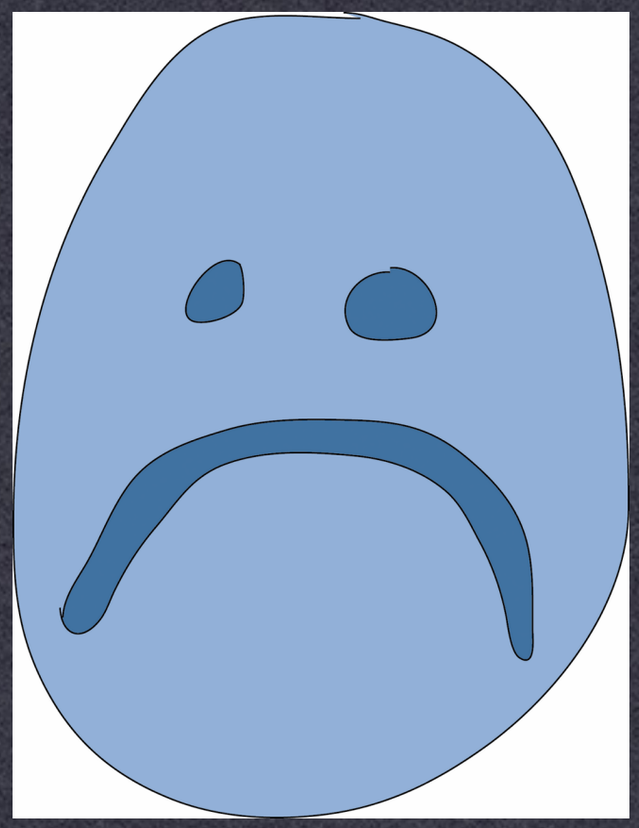
Black Box Obfuscation

- This definition has some nice points
  - Very powerful
  - Simple, easy to formalize
  - It doesn't need anything special from the program
    (some programs can be REd from just black box access, but that's not our prob

Example, online poker

Only one problem

- It's impossible to obfuscate many interesting programs
  - Under this definition, anyway
  - For some definition of "interesting"



Here's the problem

- In a nutshell:
  - The attacker who gets the obfuscated program <u>always</u> learns more than the black-box adversary

Adversary BB

Adversary OB

Here's the problem

- In a nutshell:
  - The attacker who gets the obfuscated program <u>always</u> learns more than the black-box adversary
  - He learns the obfuscated program

Adversary BB

Adversary OB

Ok...

- But by definition
  - Some programs are not “learnable”
  - You shouldn’t be able to “learn” a PRNG from observing its output
  - Ditto for signatures, MACs, and secure encryption

Adversary BB

Adversary OB

So...

- For those programs (PRNGs, MACs, etc.)
  - Obfuscation can never be black box
  - It's an impossibility
  - And that's too bad, cause these are the programs we care about

Adversary BB

Adversary OB

One sneaky exception

- One thought:
  - A program that outputs its own obfuscated code
  - The BB attacker <u>will</u> learn as much as the OB attacker
  - But what's the point of this??

Adversary BB

Adversary OB

The trick

- A small detail:
  - What if the BB program outputs a program that looks the same, but is actually garbage
  - Then the two adversaries learn “about” as much

Junk program

Adversary BB

Obfuscated Program

Adversary OB

The upshot

- To make this work:
  - We have to be able to make a junk program
  - That is <u>indistinguishable</u> from the real one
  - Example (board)

Junk program

Adversary BB

Obfuscated Program

Adversary OB

Open problems

- **Better definitions**
  - **Find weaker definitions that can be achieved**

Adversary BB

Adversary OB

Open problems

- Better definitions
  - Find weaker definitions that can be achieved
  - It's not totally hopeless:

-PKE can be considered "obfuscated" SE

-IBE is like obfuscated PKE (with tags)

Adversary BB

Adversary OB

Computing on Enc. Data

- Flip the application around
  - I want Google to compute something for me
  - (They may or may not know the program)
  - But they shouldn't know my input or outputs
  - We want to keep the data encrypted at all times

Computing on Enc. Data

- Can do a form of obfuscation here
  - Let the “program” be a VM
  - Let the “data” be a program to run on the VM
  - Note that the program only outputs encrypted data
- Hence this “obfuscation” is legal