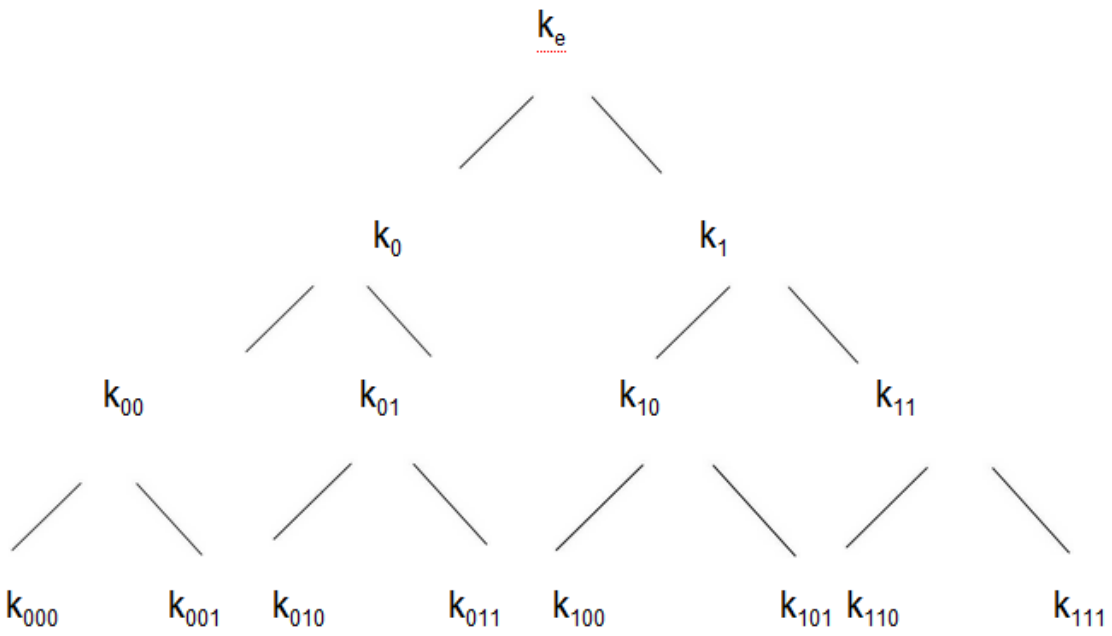
CS261 Scribe Notes
Privacy - 11/28/2011
Saung Li

Untrusted Platforms Continued:

Continuing off of the discussion at the end of last class about defenses for BluRay players, we would like each BluRay player to have its own set of cryptographic keys, and the players must decrypt discs in order to read them. If an attacker manages to extract those keys from a player and publishes them to the public, we need a way to be able to revoke those keys so that the player cannot read new discs.

AACS

One technique used is the Advanced Access Content System (AACS), which is used for content distribution and digital rights management for restricting copying of discs. We'll explain a simplified version of the process below.



The scheme can be represented as a binary tree, where each leaf is a BluRay player, each node is a key, and the keys are all unrelated to each other. An example is shown above, where in this case there are 8 players. Each player is given all the keys of its direct ancestors up to the root node. For example, the player with k_{101} will have the keys k_{101}, k_{10}, k_1, and k_e.

We have a broadcast encryption problem, in which we have n devices (the number of BluRay players, which is huge), and r (the number of revoked devices, which is small relative to n). Since n is massive we want an algorithm that scales with r and not n.

Lets say we have a message M, which is a separate key that can decrypt contents of a disc.

The encryption scheme is if there are no players revoked, then we can efficiently encrypt M using $E_{ke}(M)$. All players share this key and can easily decrypt using it.
If we revoke the device with k_{001}, then we cannot use keys k_{001}, k_{00}, k_0 k_e because that player has all of these keys so the attacker has access to them. We still need M to be readable by the other 7 players, so we choose to encrypt M with these keys:

$$E_{k1}(M), E_{k01}(M), E_{k00}(M)$$

All of the 7 players can still read M whereas the revoked device cannot. By choosing these keys we also cover all 7 players with the least number of keys.

Performance of this scheme:
Each player has log n keys.
Space per disc required to store the metadata for this scheme: O(r log n)
Keys per player: O(log n)

The actual AACS scheme is a little more complicated and has better performance:
Space: 2r
Keys per player: $O(\log^2 n)$

Problems:
There's a rogue company that extracts keys and publishes them and offers BluRay rippers. This company is located in some offshore island so we cannot shut it down. This company probably extracts keys from the players, and when the keys are revoked they break into another one and extract those keys. Also, since each player has log n keys, the attackers get access to log n keys that they can publish one at a time, so that log n keys have to be revoked in order to revoke the device. Hence, revokation needs to be done quickly in order to be effective.

Traitor Tracing
The rogue company has a service that can decrypt messages using compromised keys. Hollywood can send some metadata to the service and see if the service can decrypt it. This allows us to trace back to which keys the company has compromised, and we can proceed to revoke those keys.

Marking
We can add contents to the disc so that we can trace if a player has been broken into.

BD+
BluRay players have virtual machines that execute code, and the discs can provide code to be executed. We can add code to discs to make them update the software of the BluRay player, and we can change the encryption schemes to throw off attackers.

Concluding thoughts for copy protection
When we think about computer security, we tend to think that we are making an indestructible defense to prevent all attacks. This is not so for copy protection, as there is no way to come up with a solid defense that works against all attacks. It is more like an arms race where there is no lasting victory. In order to succeed, the defenders need to react faster than the attackers and need to be able to make software updates fast.

One example of such arms race is PayTV/smartcards, where when an attacker broke something, defender releases an update that barely fixes that attack. Then the attacker breaks something else, and the defender releases another update. This cycle repeats, and it only trained attackers to become better attackers. In this case, it may be better to invest in a stronger defense upfront than to reactively upgrade it.

Privacy

Lots of technical research has been done on privacy, but few techniques have been adopted because of the economics of privacy. Many studies on the view of privacy are based on self-reporting surveys, which are problematic because many people say they care about privacy but do not do much to protect it. Basically they are saying things that don't match their actual behavior.

Some statistics from a study done by Westin, which clustered the population into 3 groups:

1. Pragmatists (50% of population) - have some concern for privacy, but are willing to give it up for some benefits (e.g., frequent fliers, grocery cards, etc.)
2. Privacy fundamentalists (20%) - very concerned about privacy
3. Don't care (30%) - not worried about privacy

Incentives for intruding on privacy

On the Internet, services like search, email, picture repositories, etc., need to make money in order to support these services. The first business model used was micropayments, where services would charge people for doing things like reading an article or clicking a link. People would have their credit cards tied to the service, and the charges were on the order of cents. There was a move towards anonymous payments to protect users' privacy. However, this business model failed, as the thought of having to pay for every service used is psychologically unattractive, even if the payments are very small.

Services started moving to advertising and make money when a user looks at an ad on the sites. This proved to be successful, and this business model compels companies to learn more about its users so that it can target them with more specific ads to generate more revenue. Hence, users' privacy is declining as companies gather more data from them.

Case studies

Library RFID tags

Some libraries put RFID tags in books so that people can self-check them out and also make sure you don't walk out with a book without checking it out. This reduces the number of librarians needed and saves money. The problem is that someone could read the tags and find out what books the patron reads.

FasTrak

The FasTrak device has a unique ID tied to the users billing account. A patrol can interrogate

the device to do things like charging the user for a toll, tracking average speed, and tracking where users pay tolls. There is nothing to stop malicious attackers from reading the information provided by FasTrak if they have a reader and some access to the database storing the information. There was actually a divorce lawsuit in which a lawyer used FasTrak data to prove that someone was having an affair. There is not much technical progress for privacy protection here; one could imagine using public key cryptography to encrypt the data provided by FasTrak.

Geolocation

Companies who make revenues from advertising have strong incentives to profile users for better advertising targeting. One way could be to choose advertisements based on your location, so they would like your location data to be revealed. Also, many apps can do things based on your location, such as Google Maps, and phones use GPS to obtain such information.

One interesting attack is when a user first turns on her laptop or mobile phone, it queries for a WiFi connection, sending probe packets with SSIDs around to networks. Someone could use a WiFi sniffer to see which of these probe packets go to what the user designates as her “home” network, possibly finding the user’s home.

Tor, an anonymization system

Suppose Alice wants to anonymously send a message m to Bob. Alice could use a third party to relay an encrypted message for her to Bob, as shown below:

Alice → Talia: {Talia, please send to Bob; $\{m\}_{kbob}\}_{ktalia}$
Talia → Bob: $\{m\}_{kbob}$
Using this scheme, Bob would not know the message came from Alice if Talia is trusted. However, an eavesdropper could possibly figure this out if he observes that Alice sends a message to Talia and Talia immediately relays that message to Bob. One way to prevent this is for Talia to wait a while to receive more messages, then shuffle all the messages and send them out at once.

There is still the problem of relying on Talia to be trusted. We can add more trusted parties into the scheme to make it stronger. For example, in onion routing, suppose we add another trusted party, Steve. The protocol would be like this:

A → S: {Steve, please send to Talia; {Talia, please send to Bob; $\{m\}_{kbob}\}_{ktalia}\}_{ksteve}$
S → T: {Talia, please send to Bob; $\{m\}_{kbob}\}_{ktalia}$
T → B: $\{m\}_{kbob}$

At least 1 of S or T needs to be honest for A to get privacy. We can add as many layers as we’d like to strengthen privacy. However, Tor only protects privacy for this one aspect of communication, and it needs to be supplemented by other systems to ensure full privacy.