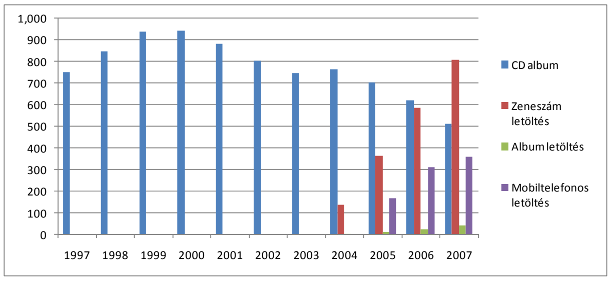
Digital Rights Management

Introduction

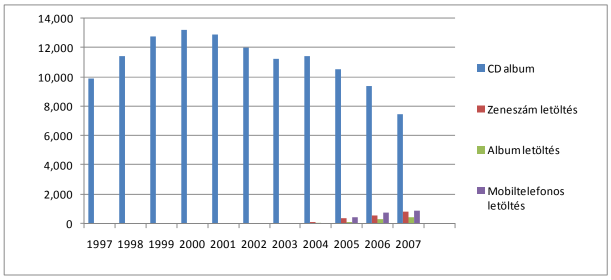
- today music, video, images, books, etc. (shortly: content) are stored and distributed in digital form
  – examples: CD, DVD, media streaming, e-books, …
  – enablers: advances in technologies such as compression, streaming, and communications (high speed Internet access)
- advantages of digital storage and distribution:
  – high quality (digital encoding, error correction)
  – instant access (through communication networks)
  – global distribution at reduced cost (via the Internet)
- the problem (for content owners and distributors):
  – digital content is easy to copy without quality degradation and at large scale
  – illegal copying and distribution leads to substantial loss in revenues

RIAA (Recording Industry Assoc. of America) statistics

- number of items sold (million units)



- revenue (million dollar)

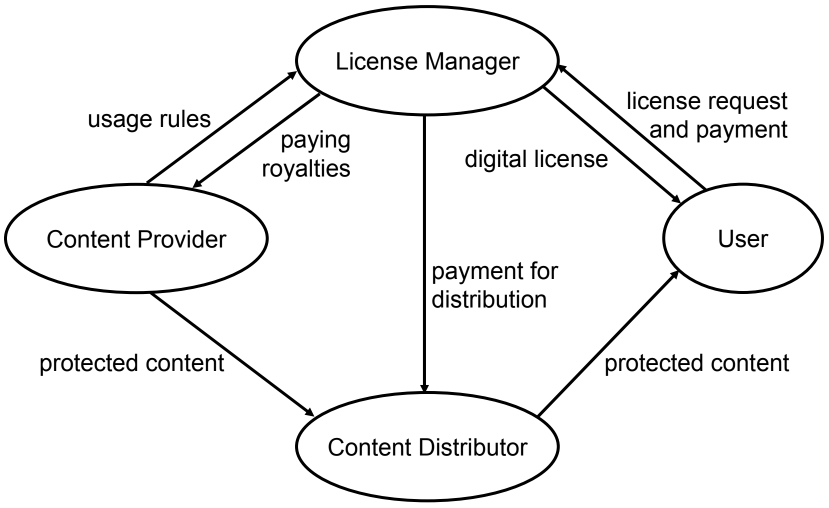


source: Fehér, Polyák, Oláh: DRM technológiák, Híradástechnika, 2008 november

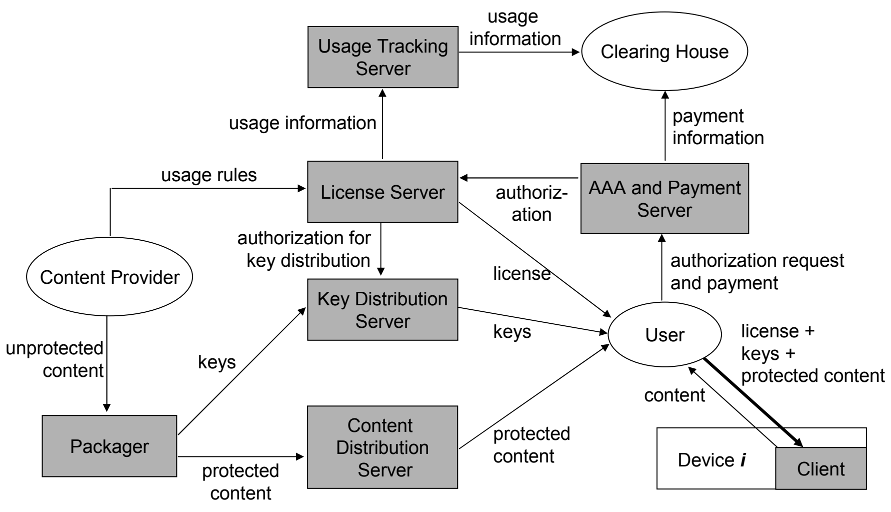
Digital Rights Management (DRM)

- objective: control of distribution and usage of digital content
  - protection against unauthorized access
- core concept: digital license
  - a digital data file that specifies certain usage rules (such as frequency of access, expiration date, restriction of transfer to other devices, copy permission, etc.) for the digital content
  - usage rules can be combined to enforce certain business models (e.g., rental or subscription, try-before-buy, pay-per-use, and a lot more)
- user downloads protected (encrypted) content and buys an appropriate license
  - content cannot be accessed without a license
  - content and licenses may be stored separately → protected content can be freely distributed among users and license requests can take place later (flexibility)

The typical DRM model

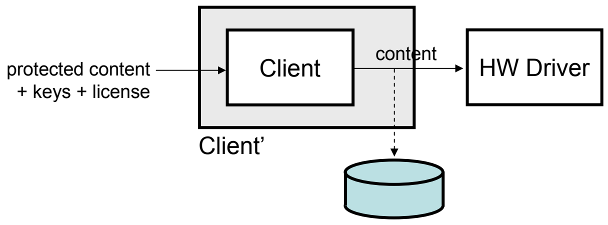


A general DRM architecture



Client-side DRM plug-ins

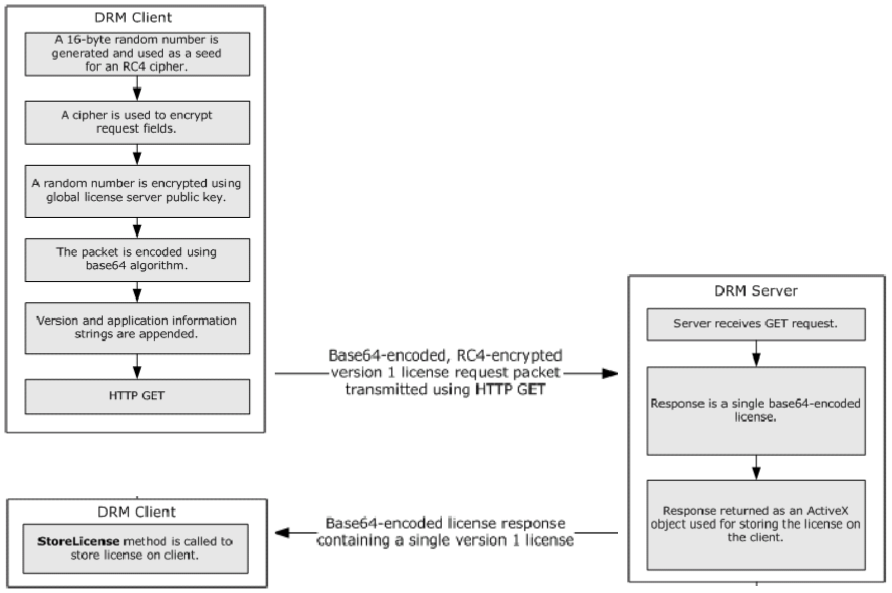
- client applications have an important role in enforcing the usage rules
  - the enforcement module is typically implemented as a DRM plug-in that can be added to existing applications
- a modified client can override the usage rules, or at worst, it can output the unprotected digital content which can then be played or viewed without any restrictions
- ensuring the tamper resistance (evidence?) of clients is a very hard (almost impossible) problem in a purely software based environment (when no trusted hardware is available on the client host)



MS Windows Media DRM (WMDRM)

- a DRM service for the Windows Media platform
- tries to provide secure delivery of audio and/or video content over an IP network to a PC or other playback device
- components:
  - **Windows Media Rights Manager (WMRM) SDK:** packaging content and issuing licenses
  - **Windows Media Format (WMF) SDK:** building Windows applications which support DRM and the Windows Media format
  - **Windows Media DRM for Portable Devices (WMDRM-PD):** supporting offline playback on portable devices (Janus)
  - **Windows Media DRM for Network Devices (WMDRM-ND):** streaming protected content to devices attached to a home network (Cardea)
- license protocol versions:
  - **v1:** license request is sent to a license server as an HTTP GET request; the server returns the license embedded within an HTML page;
  - **v7:** license request is encoded in XML format and sent using an HTTP POST request; the server responds with an XML packet containing any number and combination of version 1 and version 7 licenses;
  - **v11:** functionally equivalent to the version 7 protocol, with the addition of a few more XML fields in the license request challenge body

WMDRM License Protocol – Overview



DRM Client
A 16-byte random number is generated and used as a seed for an RC4 cipher.
A cipher is used to encrypt request fields.
A random number is encrypted using global license server public key.
The packet is encoded using base64 algorithm.
Version and application information strings are appended.
HTTP GET
Base64-encoded, RC4-encrypted version 1 license request packet transmitted using HTTP GET
DRM Server
Server receives GET request.
Response is a single base64-encoded license.
Response returned as an ActiveX object used for storing the license on the client.
Base64-encoded license response containing a single version 1 license
DRM Client
StoreLicense method is called to store license on client.

WMDRM License Protocol – Algorithms and keys

- public key encryption: ECC
- digital signature: ECDSA
- symmetric key encryption: RC4
- server keys:
  - server encryption key pair (KS_{pub}, KS_{priv})
  - server signature key pair (KS'_{pub}, KS'_{priv})
  - server certificates (CS, CS')
- client keys:
  - client encryption key pair (KM_{pub}, KM_{priv})
  - client certificate (CM)

WMDRM License Protocol – License Request

Version

EncRandNum: a random number encrypted with ECC using KS_{pub}

PKCert: certificate of the client computer (CM) encrypted with RC4 using a key KR generated from EncRandNum

KeyID: content key identifier encrypted with RC4 using KR; the content key ID is generated by the server and stored in the header of the protected content

Rights: rights requested by the client encrypted with RC4 using KR

– e.g., Play content on the client computer, Burn content to a CD, Transfer content to a portable device, Permit backup of the license, Allow the license to be restored from another location

AppSec: security level of the application that makes the request encrypted with RC4 using KR

WMDRM License Protocol – License Format

LicenseVersion

DataLen: size of the LicenseData field in bytes

Signature: signature of the LicenseData field created with ECDSA using the server's private key KS'_{priv}

LicenseData:

KeyID: content key ID

Key: content key ($K_{content}$) encrypted with ECC using KM_{pub}

Rights: rights for the licensed content

AppSec: minimum application security level required to play content associated with this license

ExpiryDate: date on which the license expires (FFFFFFFF = never expires)

(LicenseResponse containing the License is encrypted with RC4 using KR which can be computed by the server from the LicenseRequest)

Individualization

- content providers may require their digital content to be played only on a player that has been individualized
- the Microsoft Individualization Service generates a unique DLL that is bound to the client computer using its hardware ID
- a public/private key pair is stored in the DLL file
- the public key is used as the player’s identifier when requesting a license and the license server will encrypt the content key using this key
- the DLL will not work on another computer, so players on other computers will be unable to decrypt the encrypted content key and play the content
- individualization degrades user experience, because the user cannot transfer purchased content among his own devices

Why is DRM a bad idea?

- DRM systems are usually broken in minutes (sometimes days, rarely months)
  - all DRM systems share a common vulnerability: they provide their attackers with the ciphertext, the cipher, and the key
- “But DRM doesn't have to be proof against smart attackers, only average individuals!”
- this is wrong for two reasons:
  - one does not need to be smart to defeat DRM protection, he only needs to know how to search Google, or any of the other search tools for the unprotected content that someone smarter has already extracted
  - at the end of the day, DRM is capable only to keep honest user honest; but this makes no sense…

Why DRM is a bad idea?

- DRM will encourage honest users too to download content from file sharing systems instead of buying CDs and DVDs
  - it is difficult to make usable copies of a DVD on a VHS tape
  - it is difficult to play content purchased on-line by multiple devices
  - it is difficult to play a DVD bought in Europe on a player bought in the US
  - …
- DRM prevents innovation, and hence, is bad for business
  - it used to be illegal to plug anything that didn't come from AT&T into the phone network in the US; when this ban was struck down, it created a billion dollar market for third party phone equipment (phones, answering machines, cordless handsets, …)
  - DRM is the software equivalent of those closed hardware interfaces: e.g., DVD is a format where the party who makes the records gets to design the record players (anti-competition)

The response from industry …

- "Apple today announced that EMI Music's entire digital catalog of music will be available for purchase DRM-free (without digital rights management) from the iTunes Store worldwide in May." – Apple, Press Info, April 2007
- "Following digital music pioneer Apple Inc.'s lead, Microsoft Corp. said it will soon sell digital music online without digital rights management (DRM) protection." – ComputerWorld, April 2007
- "Amazon.com plans to launch a digital music store later this year, featuring music downloads without copyright restrictions." – CNET News, May 16, 2007
- "Apple has finally struck deals with all the major music labels, making songs sold via the iTunes Store free of digital rights management." -- CNET News, January 2009

What’s next?

“Microsoft has won a patent for a digital-watermarking technology that could be used to protect the rights of content owners even when digital music is distributed without DRM protection…

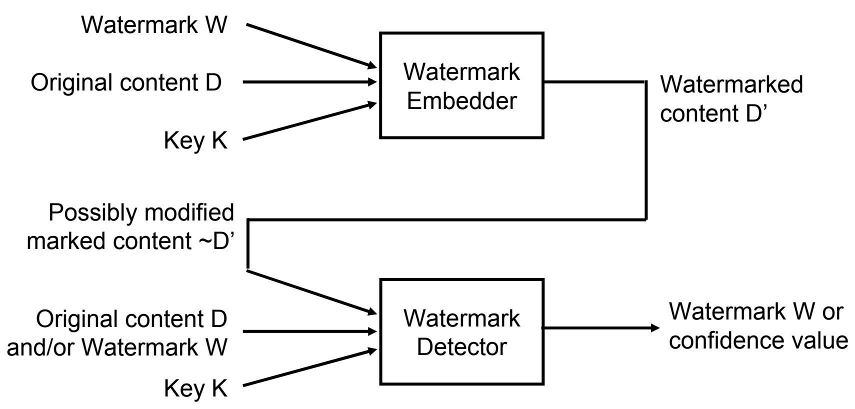
The technology, called "stealthy audio watermarking," inserts and detects watermarks in audio signals that can identify the content producer, "providing a signature that is embedded in the audio signal and cannot be removed,” …

Forensic digital watermarking technology like the technology Microsoft has patented … can be used to prove who owns the content of the digital file by encoding a file with a unique digital signature. That means illegally traded songs could be tracked back to the original purchaser, allowing authorities to identify illegal sharers and serving as a deterrent.” – InfoWorld, September 2007.

Digital fingerprinting

- idea:
  - deliver digital content with unique identification information of the user embedded in the content itself in a robust (un-removable) way
  - a pirate copy would reveal the identity of the user who shared the original content (forensics)
  - if prosecution of traitors can be enforced, then users would be discouraged to share fingerprinted content
- terminology and relation to watermarking:
  - in general, watermarking refers to embedding the same information (e.g., a copyright notice) in each copy of the content delivered to the different users
  - fingerprinting can be viewed as a specific application of watermarking where the embedded mark is user specific
  - watermarks can be fragile (manipulation destroys the mark) and robust (un-removable mark), while fingerprinting always needs robust watermarking techniques

General model of watermarking


Watermark W
Original content D
Key K
Watermark Embedder
Watermarked content D’
Possibly modified marked content ~D’
Original content D and/or Watermark W
Key K
Watermark Detector
Watermark W or confidence value

Variants

- private marking
  - detector requires at least the original content D
  - type I systems: extract the watermark W from the possibly distorted marked content ~D’ using the original content D as a hint to find W
  - type II systems: also require a copy of the embedded watermark W, and just yield a yes/no response (with confidence value) to the question: Does ~D’ contain W?
- semi-private marking
  - similar to type II private marking systems but the detector does not use the original content D
- public marking
  - detector uses neither the original content D nor the watermark W
  - it only takes the (possibly modified) marked content ~D’ and the key K, and tries to extract the watermark W

Properties of interest

- invisibility
  - often the watermark should be embedded into the host media in a way such that it remains invisible (inaudible) to humans
  - there are applications where this is not so important
- robustness
  - often the embedded watermark should be resistant against various attacks and processing techniques

    e.g., for digital image watermarking, a good watermarking algorithm should be robust against filtering, noise addition, geometric transformations such as rotation, scaling and translation, and lossy compression such as JPEG compression
  - there are applications with exactly the opposite requirement (fragile watermarks)
- capacity
  - the maximum amount of information the embedded watermark can carry and the amount of information that can be detected reliably (relative to a given attacker model)
- a good watermarking algorithm should achieve a good (application dependent) trade-off among these requirements

Watermark application examples

- content fingerprinting
  - needs invisibility and robustness
- copyright protection
  - embed information about the copyright owner into the content to prevent parties from claiming to be the rightful owners of the content
  - needs robustness, but may be visible
- content authentication
  - to be able to authenticate the content, any change to or tampering with the content should be detected
  - this can be achieved through the use of fragile watermarks, which have low robustness to modifications of the marked content
- content description
  - the watermark can contain some descriptive information of the content, such as labeling and captioning
  - the capacity of the watermark should be relatively large
  - the mark should usually be invisible, and there is usually no strict requirement for robustness

Image watermarking techniques

- spatial domain watermarks vs. transform domain watermarks
  - spatial domain: embed mark by manipulating the image (e.g., its pixels) directly
  - transform domain: transform the image into the spectral domain, embed watermark in the "spectrum" of the image, and transform the marked spectrum back in the spatial domain
- **spatial domain watermarks**
  - hide information in the LSB (Least Significant Bit) of the pixels
  - provides invisibility
  - selection of pixels to be modified and the modifications can be controlled by a key
  - does not provide robustness: easy to destroy the mark by simple filtering (without knowledge of the key)
  - can be used for image authentication, but not appropriate for copyright protection

Image watermarking techniques

- attacks aiming at destroying watermarks can be viewed as noise → techniques developed for reliable communications in a noisy environment can be useful
- **transform domain watermarks** based on principles of spread spectrum communications
  - use the Fourier transform on the image to compute its spectrum
  - select a subset V of the magnitude components in a pseudo-random manner
  - modify V as $V' = V + \alpha W$, where W is the watermark and α is some scaling parameter used to determine the embedding strength (larger α means higher robustness but also less invisibility)
  - convert the spectrum back in the spatial domain
  - for detection, extract the watermark W' from the spectrum using the same pseudo-random sequence
  - compute the cross-correlation between W and W'
  - larger correlation means higher confidence of the detection
  - experimental results show that this method resists JPEG compression with a quality factor down to 5%, and many other types of transformations

Perceptual models

- a perceptual model of the human visual system (HVS) is useful in adjusting the value of the scaling factor α in order to achieve an optimal trade-off between robustness and invisibility
- frequency
  - a large smooth area of the image corresponds to low frequency components, while a heavily textured area corresponds to high frequencies
  - a watermark embedded in the high frequency can be easily removed by attacks such as low pass filtering and JPEG compression
  - embedding the watermark in the low frequency will affect the visual quality of the host image dramatically
  - → in practice, the watermark should be embedded into the middle frequency components
- brightness
  - human eyes are less sensitive to the bright area of the image
  - the watermark can be embedded with different strengths according to a luminance function (however, this can be used only in spatial domain watermarking)
- color
  - human eyes are less sensitive to changes in the blue channel
  - in color images, use the blue color component for watermark embedding

Software protection

- software is also easy to copy and re-distribute illegally
- traditional DRM and watermarking approaches don't apply directly
  - DRM approach: needs a trusted execution environment that would enforce certain usage rules (similar to a trusted player in case of multimedia content)
  - watermarking approach: watermarked programs should preserve the functionality of the original program
- many products use hardware assisted software protection
  - hardware keys in the form of secure coprocessors, smart cards, hardware dongles, … that are needed for the software to run
  - they are cumbersome and intrusive for users
- software based approaches have been proposed based on software fingerprinting, tamper proofing, and obfuscation
- some theoretical results suggest that complete protection of software is unattainable (On the (im)possibility of obfuscating programs, Crypto'01), but *some* degree of protection can still be achieved

Attacks on software

- software piracy
  - one makes copies of a software he bought legally, and sells those copies illegally

  → software fingerprinting
- reverse engineering
  - one decompiles and reverse engineers a software and uses some of its modules in his own software

  → obfuscation
- tampering
  - one modifies the behavior of a software to his own benefit (e.g., bypassing license checking or other security controls)

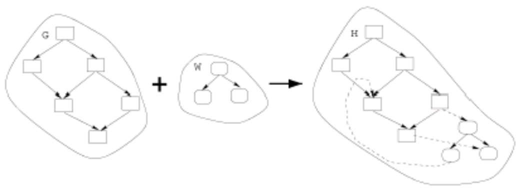
  → tamper-proofing ???

The software fingerprinting problem

- embed a structure (the possibly unique watermark W) into a program P such that:
  - W can be reliably located and extracted from P if certain secret information is available, but otherwise, it is difficult to find and remove W from P (e.g., embedding W in P does not change any statistical properties of P)
  - W has a mathematical property that allows us to argue that its presence in P is the result of deliberate actions (e.g., W can be the product of two large primes, and only the owner of the software knows how to factor it)
  - embedding W in P does not adversely affect the functionality and performance of P
  - W should withstand different distortion attacks such as various types of semantics preserving code transformations, translations (compilation and de-compilation), optimization and obfuscation
- no fingerprinting scheme will withstand a determined manual attack (where the software is inspected by a human reverse engineer for an extensive period of time) → defense against automated attacks is still an interesting objective

Static watermarking of software

- static watermarks are stored in the program code itself
- examples:
  - include an watermarked image in the static data section of the program (any image watermarking can be reused, but location and removal are very simple)
  - represent the source code as a control flow graph and embed a watermark graph in it resulting in a new control flow graph (does not really resist against optimization that could completely restructure the control flow graph of a program)



- unfortunately, all currently known static watermarks are susceptible to simple distortive and code obfuscation based dewatermarking attacks

Dynamic watermarking of software

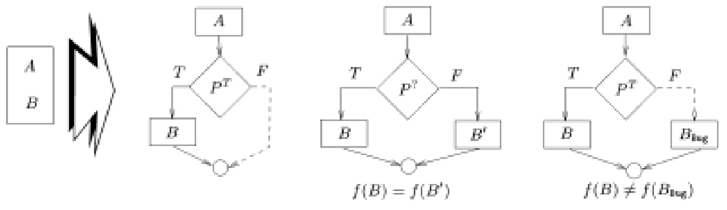
- dynamic watermarks are constructed at runtime and stored in the dynamic state of the program
- the watermark can be recognized by running the program with some special input and observing the result, which itself is the watermark
- examples:
  - easter eggs:
    - on special input they produce some special output or behavior (e.g., about:mozilla in Firefox)
    - usually easy to locate in the program code
  - execution trace watermarks:
    - the watermark is the execution trace produced by the program on the special input
    - does not resist to obfuscation and optimization
  - data structure watermarks:
    - watermark is the state of some data structure (e.g., heap, stack) produced by running the program with the special input
    - also vulnerable to obfuscation
- basically, all currently known dynamic watermarks are susceptible to code obfuscation or optimization attacks

The code obfuscation problem

- given a set of transformations T={T1, T2, …} and a program P=(S1, S2, …) find a new program P’=(…, Ti(Sj), …) such that
  - P’ has the same observable behavior as P (semantics preservation)
  - understanding and reverse engineering P’ is strictly more difficult than understanding and reverse engineering P (effectiveness)
  - it is difficult for an automated tool to identify and undo the obfuscating transformations applied in P’ (resilience)
  - the statistical properties of P’ are similar to those of P
  - the performance of P’ is similar to that of P
- note: code obfuscation is similar to code optimization but in addition to minimize execution time, code obfuscation also aims at maximizing obscurity

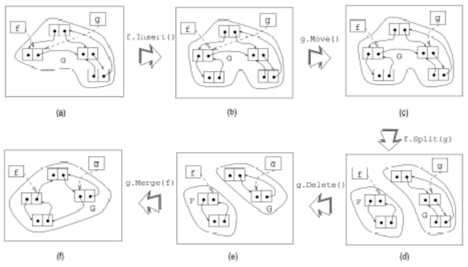
Code obfuscations techniques

- simple lexical transformations (e.g., scrambling identifiers)
- control transformations based on opaque predicates
  - an opaque predicate is a predicate R whose outcome is known to the obfuscator at obfuscation time, but it is difficult for the de-obfuscator to deduce it
  - given such predicates, the control flow of the program can be modified without changing the semantics of the program



How to manufacture strong opaque predicates?

- most de-obfuscators will employ static analysis techniques → build opaque predicates on problems that are hard to tackle with static analysis, e.g., alias analysis in pointer structures
  - extend the program with code that builds a set of complex dynamic structures, and occasionally update those structures, but maintain certain invariants (e.g., pointer p and q will never refer to the same location)
  - use these invariants to construct opaque predicates
  - de-obfuscator cannot rely on static analysis, but he essentially needs to emulate the execution of the program in order to evaluate these predicates

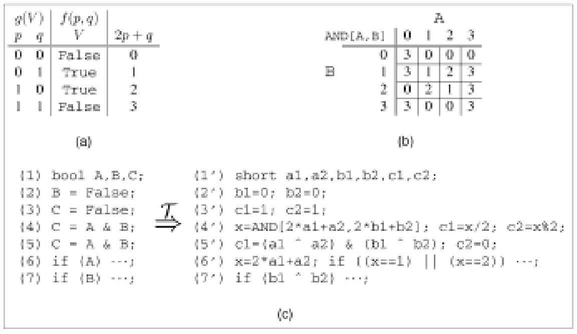


$(f == g)^{?}$

$(f == g)^{False}$

Code obfuscations techniques

- data transformations
  - e.g., variable splitting



g(V) p	q	f(p,q) V	2p + q
0	0	False	0
0	1	True	1
1	0	True	2
1	1	False	3

(a)

AND[A,B]	A: 0	1	2	3
B 0	3	0	0	0
1	3	1	2	3
2	0	2	1	3
3	3	0	0	3

(b)

```
(1) bool A,B,C;              (1') short a1,a2,b1,b2,c1,c2;
(2) B = False;               (2') b1=0; b2=0;
(3) C = False;        T      (3') c1=1; c2=1;
(4) C = A & B;       ==>     (4') x=AND[2*a1+a2,2*b1+b2]; c1=x/2; c2=x%2;
(5) C = A & B;               (5') c1=(a1 ^ a2) & (b1 ^ b2); c2=0;
(6) if (A) ···;              (6') x=2*a1+a2; if ((x==1) || (x==2)) ···;
(7) if (B) ···;              (7') if (b1 ^ b2) ···;
```

(c)

note that there are multiple ways to compute the same expression!