University of Arizona

CSc 620

Security Through Obscurity

Christian Collberg

January 29, 2002

Steganography

Steganography

- Steganography is the art of hiding a secret message inside another *host* message.
- Historical tricks: invisible ink, hidden tatoos, microdots, etc.
- Steganography is uded to hide a message, but, more importantly, it hides the fact that the message *exists*.
- There are military applications, of course, but watermarking and fingerprinting are modern commercial applications.

- Ross Anderson, Fabien Petitcolas, *On The Limits of Steganography*:

While classical cryptography is about concealing the content of messages, steganography is about concealing their existence. It goes back to antiquity: Herodotus relates how the Greeks received warning of Xerxes' hostile intentions from a message underneath the wax of a writing tablet, and describes a trick of dotting successive letters in a covertext with secret ink, due to Aeneas the Tactician. Kahn tells of a classical Chinese practice of embedding a code ideogram at a prearranged place in a dispatch; the same idea arose in medieval Europe with grille systems, in which a paper or wooden template would be placed over a seemingly innocuous text, making a secret message visible.

... The "Prisoners' Problem".... In this scenario, Alice and Bob are in jail, and wish to hatch an escape plan; all their communications pass through the warden, Willie...

...if Willie detects any encrypted messages, he will frustrate their plan by throwing them into solitary confinement. So they must find some way of hiding their ciphertext in an innocuous looking covertext. As in the related field of cryptography, we assume that the mechanism in use is known to the warden, and so the security must depend solely on a secret key that Alice and Bob have somehow managed to share.

Apparently, during the 1980's, Margaret Thatcher became so irritated at press leaks of cabinet documents that she had the word processors programmed to encode their identity in the word spacing of documents, so that disloyal ministers could be traced.

Simmons' formulation of the Prisoners' Problem was itself an instance of information hiding. It was a ruse to get the academic community to pay some attention to a number of issues that had arisen in a critical but at that time classified application – the verification of nuclear arms control treaties....

. . . The US and the USSR wanted to place sensors in each others' nuclear facilities that would transmit certain information (such as the number of missiles) but not reveal other kinds of information (such as their location). This forced a careful study of the ways in which one country's equipment might smuggle forbidden information past the other country's monitoring facilities.

Steganography must not be confused with cryptography, where we transform the message so as to make its meaning obscure to a person who intercepts it. Such protection is often not enough. The detection of enciphered message traffic between a soldier and a hostile government, or between a known drug-smuggler and someone not yet under suspicion, has obvious implications; and recently, a UK police force concerned about criminal monitoring of police radios has discovered that it is not enough to simply encipher the traffic, as criminals detect, and react to, the presence of encrypted communications nearb.

Slide 10–4

Watermarking & Fingerprinting

Watermark: a secret message embedded into a cover message.

- Image, audio, video, text,. . .
- Visible or invisible marks.
- Watermarking
  1. discourages theft,
  2. allows us to prove theft.
- Fingerprinting
  3. allows us to trace violators.

Slide 10–5

- What is watermarking? The idea is simple:
  1. Start with a *cover message* of some kind. It can be an image, a sound-file, a video.
  2. Now, take a secret message and embed it inside the cover message. Usually, this secret message is some sort of copyright notice.
- In this example, the cover message is an image and the secret message is a the word "copyright", a date, and my name.
- If someone were to steal this image and use it in a book or magazine, then I could take them to court for breach of copyright, and I would argue that this is indeed my image because it has my copyright notice in it.

Slide 10–6

- The fact that this copyright notice is visible does take away some of the value of the image, so, in most cases we want the watermark to be imperceptible. On the other hand, if the watermark is visible, then that will discourage someone from stealing the image in the first place.
- Fingerprinting is a variant of watermarking. When we watermark an image we store the same copyright notice in every copy. When we fingerprint an image we store a unique customer-identification-number in every image we sell. That way, if lots of illegal copies of the image should start to appear, then we may be able to trace the customer who bought the original copy, and then bring them to court.

Slide 10–7

Slide 10–8

Media Steganography – Restrictions

1. The quality of host object should not be too degraded.
2. We should not be able to detect (view/hear) the presence of the embedded data.
3. The data should be embedded directly in the media, not in headers, etc., since these are removed when converting to other formats.
4. The data should survive: channel noice, filtering, resampling, cropping, encoding, lossy compression, printing, scanning, D/A conversion, A/D conversion, etc.

Slide 10–9

Media Steganography – Applications

Tamper-proofing: We would like to find out if a particular image has been modified. For example, one could embed a checksum of the image in the image itself.

Feature tagging: The names of the people in a photograph could be invisible embedded into the picture, at the location where those people occur.

Embedded captions: The caption that describes a picture could be embedded in the picture itself.

Slide 10–10

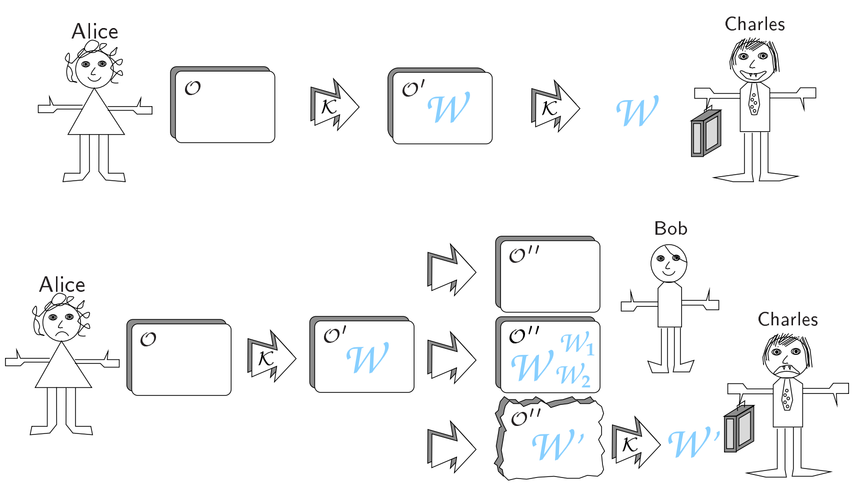
Media Watermarking

- Bender says:

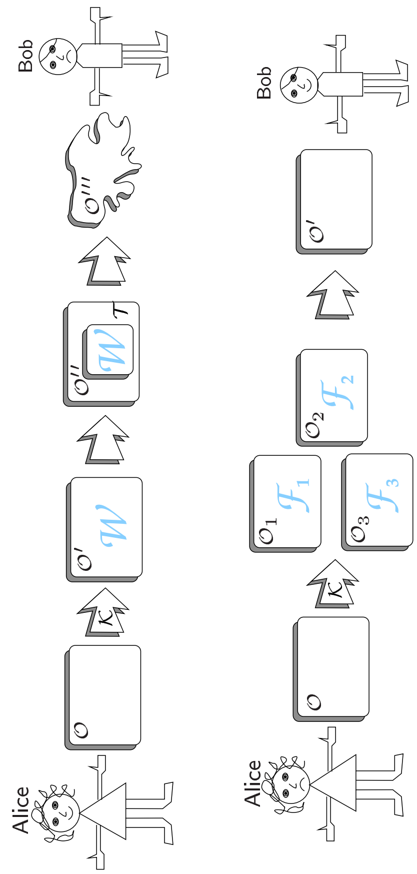
  The key to successful data hiding is the finding of holes that are not suitable for for exploitation by compression algorithms. A further challenge is to fill these holes with data in a way that remains invariant to a large class of host signal transformations.

Slide 10–11

Watermarking Overview I



Watermarking Overview II



Slide 10–12

- To build a good watermarking system these problems have to be solved:
  1. The watermarkshould be stealthy, i.e., it should be difficult for an adversary to locate.
  2. It should have a high data-rate, i.e., we should be able to store a large secret message in a small cover message.
  3. It should be resilient to attack, i.e. it should be hard to remove the watermark from the cover message.
- Scenario: Alice, has an object (an image, a program, whatever) that she wants to sell. Bob wants to steal this object to sell it on to a third party.
- First, Alice adds a watermark to her object. She uses a secret key to make sure that she is the only one who can extract the watermark.

Slide 10–13

- Then Bob steals a copy of Alice's watermarked object. Bob sells a stolen copy to Charles who is Alice's lawyer. Charles extracts the watermark using the secret key, and turns Bob over to the authorities.
- If Bob can find the location of $\mathcal{W}$, he may try to *crop* it out of the object, without destroying too much of the object iteself. We call this a *subtractive* attack.
- An even simpler idea is an *additive* attack, where Bob adds new watermarks to the object to make it hard for Charles to prove that Alice's watermark is the original one.

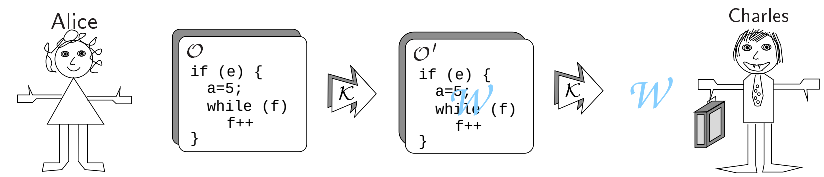
Slide 10–14

- Or, Bob could launch a *distortive* attack, where he applies a sequence of transformations to the object. Ideally, he will add just the right level of distortion to the object so that the watermark will be useless, but the object itself still has some value to Bob.
- Alice can add *tamperproofing* to her object such that if Bob tries to remove the watermark the resulting object is completely useless.
- If Alice fingerprints her object she leaves herself open to *collusive* attacks. Bob can steal several copies of the object – each with its own fingerprint – and by comparing their differences he is able to extract the original object.

Slide 10–15

Slide 10–16

Software Watermarking



data rate: ≤ 1000 bits?

cover program: source code/object code? typed/untyped? architecture-neutral/native binary?

threat-model: semantics-preserving transformations (translation, optimization, obfuscation)?

logistics: generation, distribution, bug-reports?

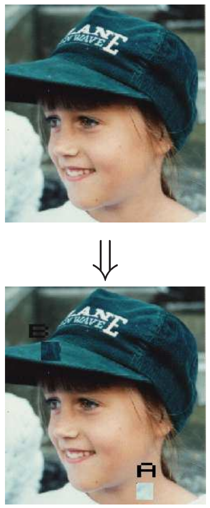
Slide 10–17

Attacks on Media Watermarks

- We trade-off between
  1. *stealth* (we want imperceptible marks),
  2. *bit-rate* (we want to embed much data),
  3. *resilience* (we want the mark to withstand attacks).
- Attacks: compression, scaling, cropping, blurring, rotation...

Slide 10–18

Image Watermarking – Patchwork



Embed: `Init_RND`($\mathcal{K}$);
  `repeat` $n \approx 10000$ `times`
    $i \leftarrow$ `RND`(); $j \leftarrow$ `RND`();
    `fix brightness` : $a_i \leftarrow a_i + \delta$; $b_j \leftarrow b_j - \delta$;

Extract: `Init_RND`($\mathcal{K}$); $S' \leftarrow 0$;
  `repeat` $n \approx 10000$ `times`
    $i \leftarrow$ `RND`(); $j \leftarrow$ `RND`();
    `sum brightness` : $S' \leftarrow S' + a_i - b_j$;

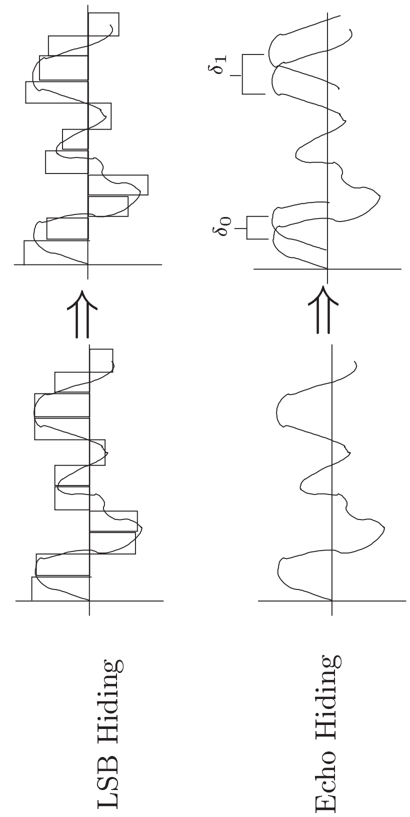
- $\sum_i^n (a_i - b_i) \gg 0 \Rightarrow$ watermark!
- Bit-rate: 1 bit per image.

Slide 10–19

- Might some media watermarking techniques carry over to software watermarking?
- The most well-known image watermarking algorithm is called patchwork. The idea is to store the watermark in the brightness of a random sequence of pixels in the image.
  1. To embed the watermark we start by initializing a random number generator with our secret key.
  2. Using the random number generator we then pick a sequence of pairs of pixels. We then adjust the brightness of one of the pixels up by a small amount, and the brightness of the other pixel, down by the same amount.

- Notice that the overall brightness of the image hasn't changed.
- 1. To extract the watermark we visit the same pseudo-random sequence of pixels, this time summing up the difference of their brightness.
  2. For a completely random sequence of pixels we'd expect this sum to be zero. If it's not, we've detected a watermark.
- This method is stealthy but the bit-rate is low, just one bit per image.
- We expect $S \approx 0$ as n increases.
- δ would be 1–5 parts in 256 bits.
- Each patch can contain more than one bit.

Audio Watermarking



- Many media watermarking algorithms take advantage of the fact that our human sensory systems aren't perfect.
- To watermark a sound-clip we can flip the the least significant bits of each sample. This may introduce some noice, but if the sample is noisy to begin with we will not be able to detect it.
- Humans also are not very good at detecting short echos. We can use this fact to encode a secret message by introducing short echos in the sample. The length of these echos would encode the watermark.
- Again, we can use a pseudo-random number generator to pick out the places where the watermarking bits are encoded.

Text Watermarking

Hard-Copy	12pt { `I saw the best minds` `of my generation,` 14pt { `starving hysterical naked`

White-Space	`I␣␣␣saw␣␣the␣best␣␣␣minds`
	 `of␣␣␣␣␣␣my␣␣␣generation,`
	 `starving␣hysterical␣naked`

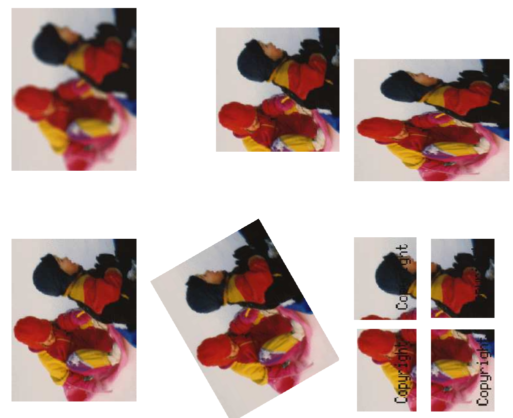
Syntactic	`It was the best minds`
	 `of my generation that I saw,`
	 `starving hysterical naked`

Semantic	`I observed the choice intellects`
	 `of my generation,`
	 `starving hysterical nude`

- We can encode watermarks in English text.
- In a formatted hard-copy document we can encode a mark in the word or line spacing.
- In an soft-copy document we can encode the mark in white-space: One space means 0, two spaces means 1.
- If every line has to be justified we can't encode a bit between every pair of words. We can use "Manchester-encoding": $01 \equiv 1$, $10 \equiv 0$, $11 \equiv$ `null`, $00 \equiv$ `null`.
- We can also encode a mark in the syntactic structure of an English text. Here we've applied a syntactic transformation to make a "Cleft sentence."
- Finally, we can store information in the choice of words.

Slide 10–24

Attacks on Media Watermarks



- We trade-off between
  1. *stealth* (we want imperceptible marks),
  2. *bit-rate* (we want to embed much data),
  3. *resilience* (we want the mark to withstand attacks).
- Attacks: compression, scaling, cropping, blurring, rotation...

Slide 10–25

- There are three things we want from our watermarking algorithm:
  1. Marks should be hard to find (they should be *stealthy*);
  2. The *bit-rate* should to be high (we want large watermarks);
  3. Marks should be *resilient* (they should withstand as many kinds of attacks as possible).
- Many simple image transforms will obliterate a watermark. Lossy compression will remove imperceptible bits. If watermarks are stored in the least significant bits of the media then we can randomly flip all such bits.
- There will always be a trade-off between stealth, bit-rate, and resilience. You can increase resilience by including the watermark more than once, but then the bit-rate goes down.

Slide 10–26

Stego tools – Outguess

- Outguess, `http://www.outguess.org`, hides messages in jpg images. The command

  `outguess -k "key" -d hidden.txt deer.jpg out.jpg`

  hides the text in `hidden.txt` in the image `deer.jpg` using the key *secret-key*. The command

  `outguess -k "key" -r out.jpg message.txt`

  retrieves the message.

Slide 10–27

Slide 10–28

Stego tools – StirMark

- StirMark, `http://www.cl.cam.ac.uk/~fapp2/watermarking/stirmark/`, Applies transformations to images in order to destroy any messages hidden within. The command

  ```
  stirmark out.jpg >! out-stir.jpg
  ```

  tries to destroy the message in the image we marked with `outguess`. However,

  ```
  outguess -k "key" -r  out.jpg message.txt
  ```

  still retrieves the message.

Slide 10–29

Stego tools – Snow

- Snow, `http://www.darkside.com.au/snow/index.html`:

  > The program snow is used to conceal messages in ASCII text by appending whitespace to the end of lines. Because spaces and tabs are generally not visible in text viewers, the message is effectively hidden from casual observers. And if the built-in encryption is used, the message cannot be read even if it is detected.

- Snow can also be run as an applet: `http://www.darkside.com.au/snow/jsnowapp.html`

Slide 10–30

Stego tools – NiceText

- NiceText, `http://www.ctgi.net/nicetext/`

  > NICETEXT is a package that converts any file into pseudo-natural-language text. It also has the ability to recover the original file from the text! The expandable set of tools can:

  - Create custom dictionaries from a variety of sources.
  - Simulate many different writing styles by example!
  - Alternatively use Context-Free-Grammars to control writing style.

- I'm unable to download this.

Slide 10–31

Stego tools – texto

- `ftp://ftp.ntua.gr/pub/crypt/mirrors/idea.sec.dsi.unimi.it/cypherpunks/steganography/texto.tar.z`

  Texto is a rudimentary text steganography program which transforms uuencoded or pgp ascii-armoured ascii data into English sentences. This program's output is hopefully close enough to normal English text that it will slip by any kind of automated scanning. Texto text files look like something between mad libs and bad poetry. Texto works just like a simple substitution cipher, each of the 64 ascii symbols used by pgp ascii armour or uuencode is replaced by an english word. Not all of the words in the resulting English are significant, only those nouns, verbs, adjectives, and adverbs used to fill in the preset sentence structures. Punctuation and "connecting" words (or any other words not in the dictionary) are ignored.

Slide 10–32

```
> pgp -kg
Pick your DSS/DH key size:
Choose 1, 2, 3, or enter desired number of bits: 1
Enter a user ID for your public key: 620 <620@cs.arizona.edu>

Enter pass phrase: go 2 hell
Key generation completed.

> pgp -kv
Type bits/keyID    Date       User ID
sec  1024/65F29028 2002/01/29 620 <620@cs.arizona.edu>

> cat msgfile
Attack at dawn
> pgp -esa msgfile
Enter the Recipient's user ID: 620
Enter pass phrase: go 2 hell
```

Slide 10–33

```
> cat msgfile.asc
-----BEGIN PGP MESSAGE-----
Version: PGP 6.5.8
qANQR1DBwE4DQWQxphsdu78.............

> texto msgfile.asc > engfile
> wc engfile
     93    1009    5866 engfile
> cat engfile
The case finally severs to the wooden canyon.  I eat quick
caps near the bright lazy cliff.  Sometimes, shirts vend
behind untouched arenas, unless they're dim.  Never wash.....

> texto -p engfile > pgpfile.out
```

Slide 10–34

```
> cat pgpfile.out
-----BEGIN PGP MESSAGE-----
Version: 2.?
qANQR1DBwE4DQWQxphs.........

> pgp pgpfile.out
Key for user ID: 620 <620@cs.arizona.edu>
Enter pass phrase: go 2 hell
> cat pgpfile
Attack at dawn
```

Slide 10–35

- `texto` uses a simple database of sentence patterns and a dictionary of common words:

```
The _THING _ADVERB _VERBs to the _ADJECTIVE _PLACE.
Have a _ADJECTIVE _THING.
The _ADJECTIVE _THING rarely _VERBs.
He will _VERB _ADVERB if the _THING isn't _ADJECTIVE.
# Things
frog pen egg
# Verbs
run eat smell
# Places
hill room hallway
# Adverbs
slowly quickly lazily
# Adjectives
blue ajar new
```

Slide 10–36

md5sum

- `md5sum` generates or checks MD5 message digests.
- MD5 is a 128-bit "summary" of an arbitrary text. Any changes to the text will generate a different checksum.

```
> md5sum lecture.tex
64059d4d61308fd0db2e1b19e0e727a7  lecture.tex
> cat >> lecture.tex
.
> md5sum lecture.tex
731bab65b3151bb4f0e6ae9f5de0e58c  lecture.tex
```

Slide 10–37

Readings and References

- Bender, Gruhl, Morimoto, Lu, *Techniques for Data Hiding*, IBM Systems Journal, Vol. 35, Nos 3&4, 1996, `http://www.research.ibm.com/journal/sj/mit/sectiona/bender.html`.
- `http://www.stegoarchive.com/` has links to many steganography tools.
- `http://www.cosy.sbg.ac.at/~pmeerw/Watermarking/source/` has source for many watermarking algorithms.
- The CYBERPUNK's mailing list: `http://www.csua.berkeley.edu/cypherpunks/Home.html`.