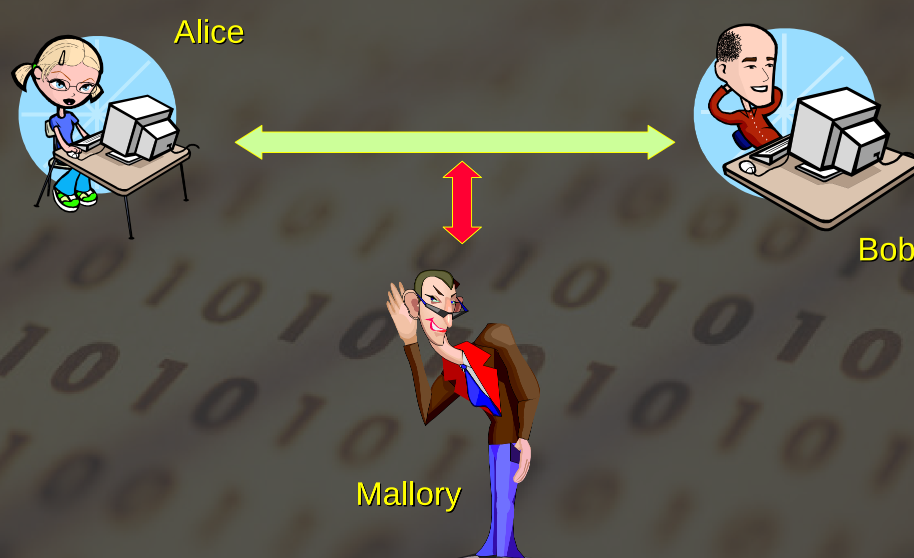
Fragile watermarks for LZ-77

Stefano Lonardi

University of California, Riverside

joint work with M. Atallah (CERIAS & CS, Purdue U.)

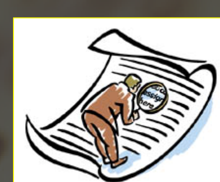


Problem

- Alice sends a document *T* to Bob
- She wants to make sure that what Bob receive is
  – Authentic
  – Integral
- Mallory monitors the communication and he will attempt to modify *T* and impersonate Alice

Signatures

- Signature requirements
  – Authentic/Unforgeable
  – Not reusable
  – Cannot be repudiated



- The signed document should be unalterable *(integrity)*
- Typical solution involves PKC

Information Hiding

- Steganography
- Watermarking

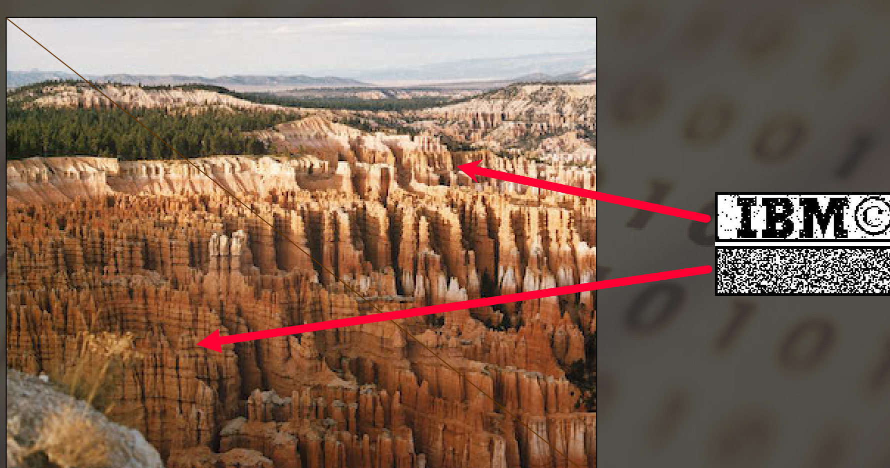
Steganography

- The art/science of hiding a secret message within another one, in such a way that the adversary cannot discern the presence or the content of the hidden message

(Robust) Watermarking

- The art/science of hiding a secret message within another one, in such a way that the adversary cannot remove the hidden message (watermark) without destroying the cover

Example of watermarked image



From research.ibm.com

Image Watermarking

- Some methods have been proved remarkably resilient to
  - Lossy compression/Filtering
  - Cropping/Resizing
  - Scanning and printing
  - Repeated photocopying

(see, e.g., Cox *et al.*, IEEE TIP 97)

Watermarking

- So far, most of the research has been focused on
  - Images
  - Movies
  - Audio
  - Source Code
- Little has been done for textual data

Information hiding in textual data

- It is believed that

  "… text is in many ways the most difficult to hide data … due largely to the lack of redundant information in a text as compared with a picture or a sound file …"

Information hiding in textual data

- Methods range from changing slightly the fonts or the spacing between words/lines, to rewriting some words/phrases of the text without changing the semantics

- Hiding information in textual data is a challenging problem

Motivation

- Lossless compression is very common nowadays
  - gzip, (win)zip, (win)rar, compress, bzip2, etc.
- Since we are sending the document over the network and it is likely that we are going to compress it anyway, why not watermark the compressed file?

Fragile watermarks

- A *fragile watermark* is a watermark designed to break as soon as the content of the document is changed
- An alternative way to authenticate a document and ensure that it reaches the destination in a integral state

Notation

- *T:* document, |*T*|=*n*
- *k:* secret key
- *W:* (fragile) watermark
- *T':* watermarked & compressed document

Specifications

- *T=T'* (or semantically equivalent)
- Unless *k* is known
  - it is very hard to retrieve *W* from *T'*
  - it is very hard to add *W* to another text and pretend to be Alice
- The presence of *W* in *T'* would hold up in court (false positives are extremely rare)
- The security of the process should be based solely on the secrecy of the key *(Kerckhoffs' principle)*

Approach

- We propose a method that hides *W* (the digest of *T*) directly in the compressed file as a fragile watermark, and therefore
  - is transparent to the casual observer
  - does not require to send separately the signature
- It also satisfies all the previous requirements

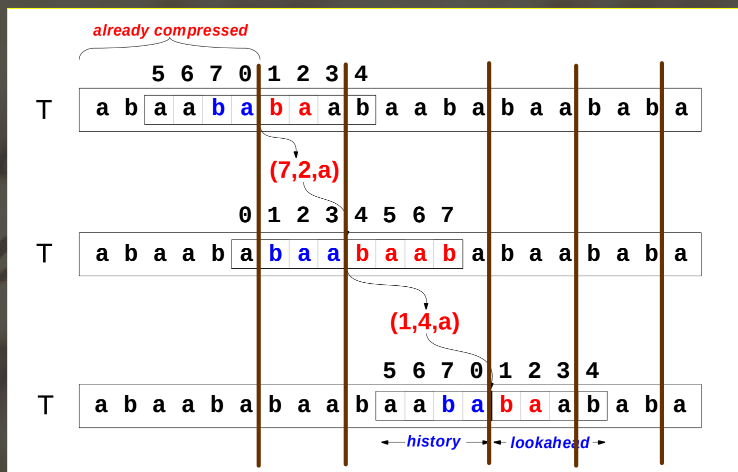
Which format?

- We choose Lempel-Ziv '77 because …

… is very popular and widespread

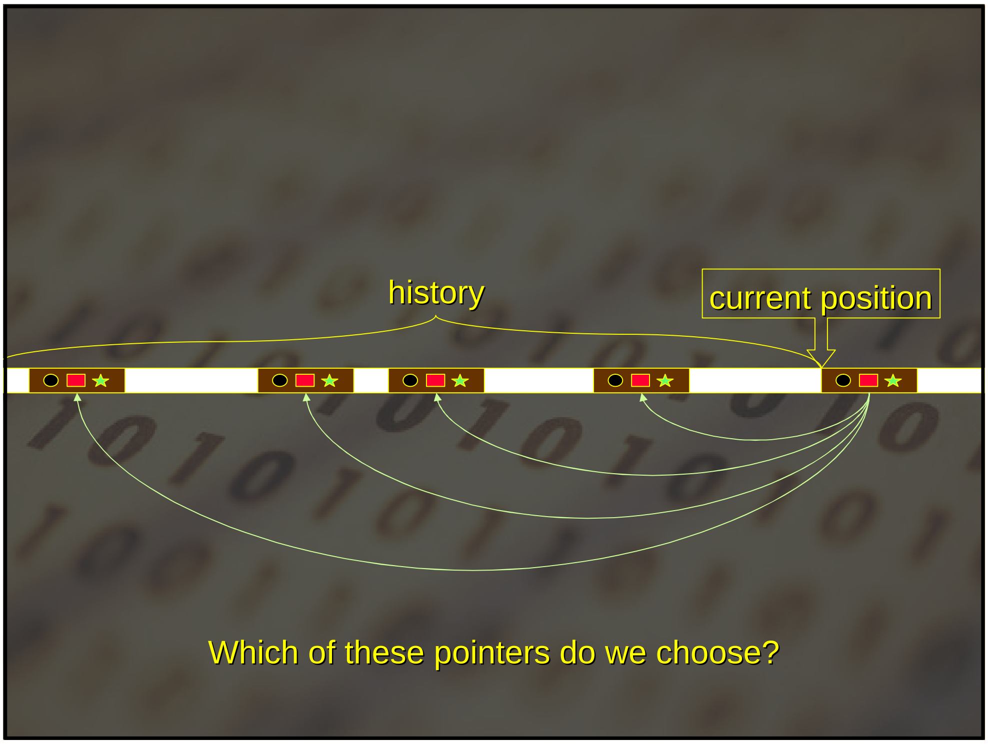
… hiding data turns out to be very elegant

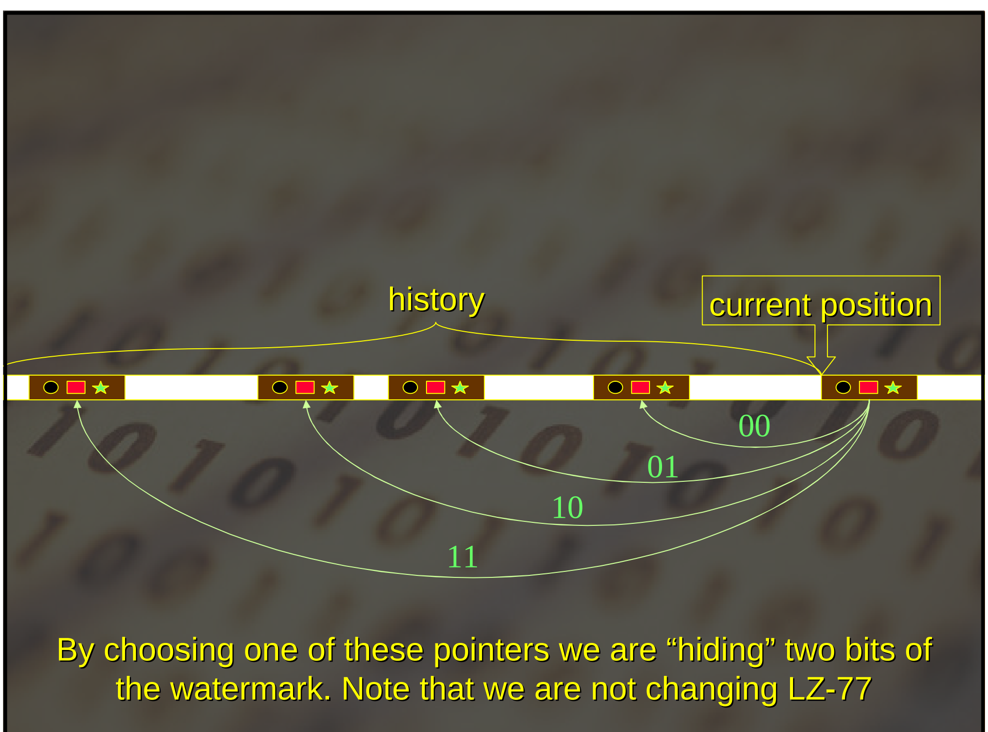
Lempel-Ziv 77 (gzip)

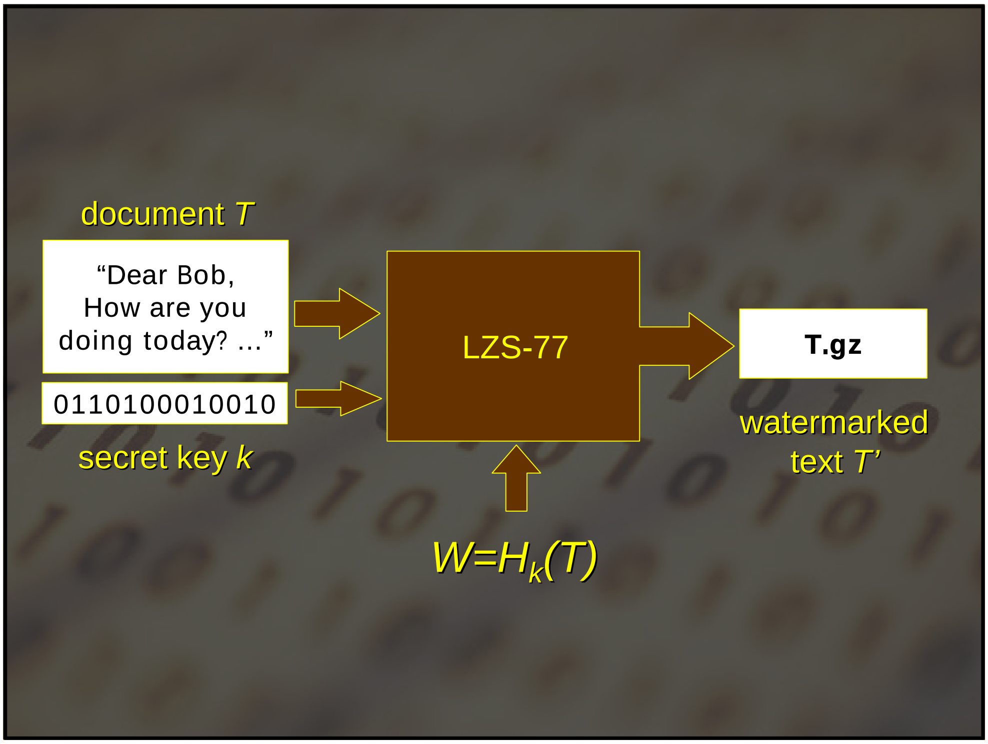


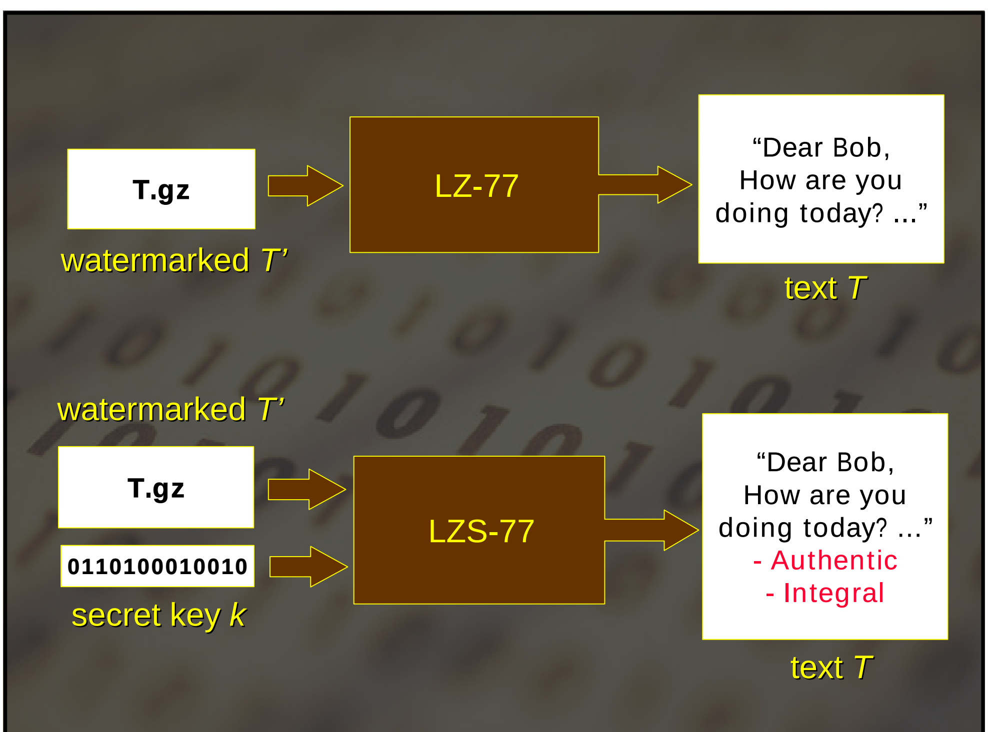
The LZ processing induces a parsing of *T* into *phrases*

Idea


history
current position
Which of these pointers do we choose?


history
current position
00
01
10
11
By choosing one of these pointers we are "hiding" two bits of the watermark. Note that we are not changing LZ-77


document T
"Dear Bob,
How are you
doing today? …"
0110100010010
secret key k
LZS-77
T.gz
watermarked
text T'
W=H_k(T)


T.gz
watermarked T'
LZ-77
"Dear Bob,
How are you
doing today? …"
text T
watermarked T'
T.gz
0110100010010
secret key k
LZS-77
"Dear Bob,
How are you
doing today? …"
- Authentic
- Integral
text T

Method

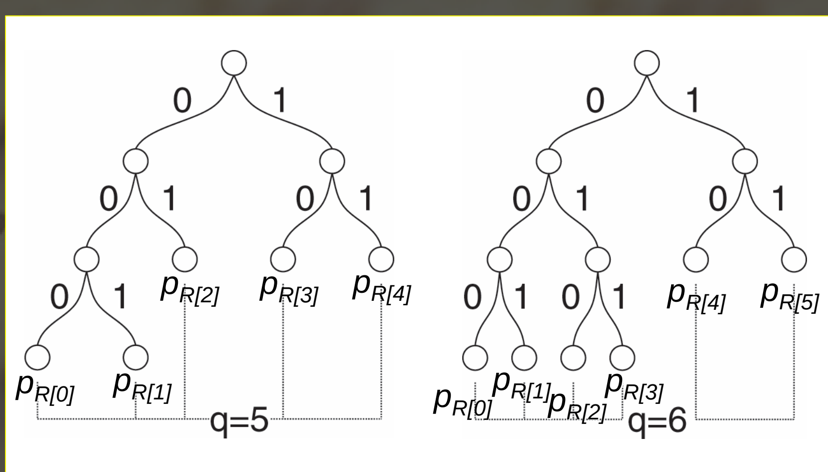
Multiplicity

- <u>Definition</u>: a position *i* in the text *T* has *multiplicity q* if there exists exactly *q* matches of the longest prefix of *T[i,n]*
- Given a position with multiplicity *q*, we denote by $p_0, p_1, \ldots, p_{q-1}$ the *q* choices for the pointer

Encoding

- For each phrase *i* with multiplicity $q>1$
  - Initialize the seed of a random generator with $H(k,i,p_0,p_1,\ldots,p_{q-1})$
  - Generate a uniformly distributed random permutation *R* of the set $\{0,1,\ldots,q-1\}$
  - Reorder the pointers based on *R*, *i.e.*, $p_{R[0]}, p_{R[1]}, \ldots, p_{R[q-1]}$
  - Assign each pointer $p_{R[i]}$ a binary code
  - Choose the pointer which binary code matches with the next bits of *W*

Binary trees for q=5 and q=6



Security

- Finding the watermark is at least as hard as breaking the pseudo-random generator
- Finding the key requires to be able to invert a one-way hash function

Security

- If one uses some crypto-secure RNG, like BBS [Blum, Blum, Shub 86], the pseudo-random sequence *cannot* be reproduced in a reasonable amount of computing time without the knowledge of the seed $H(k,i,p_0,p_1,\ldots,p_{q-1})$

Experiments

Prototype

- We implemented a suffix tree-based LZ-77
- We measured
  - the numbers of bits embedded vs. the length of the text
  - the average multiplicity of pointers
  - the length of the longest prefix

Number of bits embedded

# of bits embedded	*length of the prefix of* `paper2`
128	1,149
256	1,692
1,024	4,778

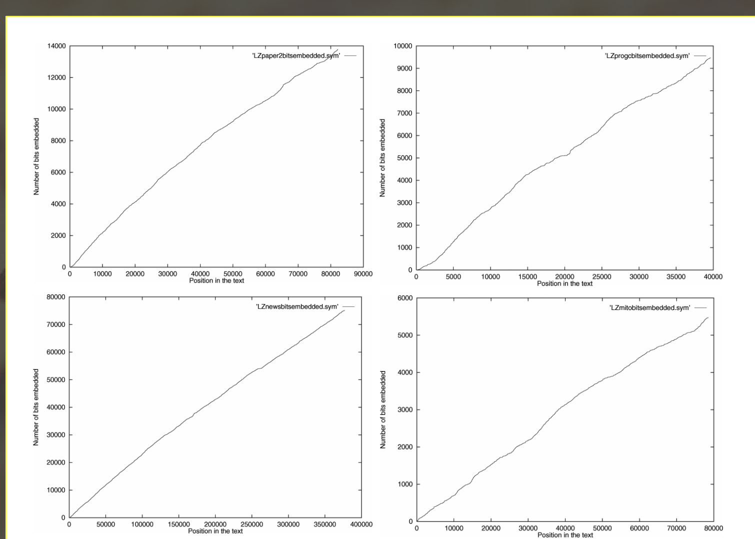
# of bits embedded	*length of the prefix of* `progc`
128	863
256	1,729
1,024	4,401

# of bits embedded	*length of the prefix of* `news`
128	1,115
256	1,825
1,024	5,195

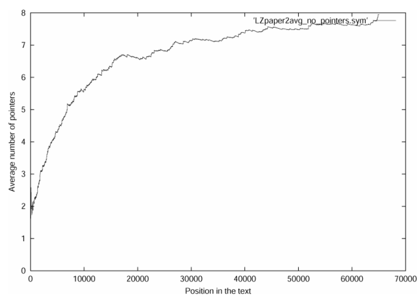
# of bits embedded	*length of the prefix of* `mito`
128	1,488
256	3,078
1,024	14,310

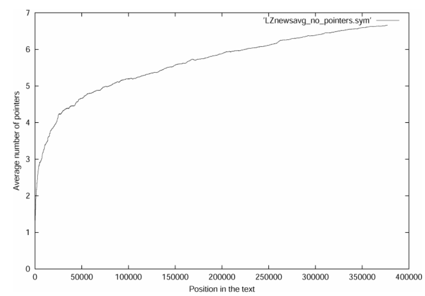
Remark: more bits can be embedded relaxing the greediness

Number of bits embedded



Average multiplicity of pointers





Conjecture: The average multiplicity ? *O(1),* as *n? 8*

gzip

- gzip issues pointers in a sliding window of 32Kbytes (typically)
- The length of phrases is represented by 8 bits (3-258)
- Strings smaller than 3 symbols are encoded as literals

gzip

- gzip always chooses the most “recent” occurrence of the longest prefix

  “…the hash chains are searched starting from the most recent strings, to favor small distances and thus take advantage of the Huffman coding…”

gzip

- We modified gzip-1.2.4 to evaluate the potential degradation of compression performance due to changing the rule of choosing always the most “recent” occurrence

- As a preliminary experiment, we simply chose one pointer at random

Gzip vs. GzipS

gzip	gzipS	*file*	*bits embedded*
43,871	44,597	bib	11,317
365,005	375,746	book1	154,632
248,846	255,070	book2	84,378
69,810	71,343	geo	15,105
164,199	167,401	news	45,319
10,707	10,828	obj1	1,429
93,906	95,494	obj2	17,988
21,612	22,088	paper1	6,335
35,078	36,011	paper2	12,541
20,819	21,320	paper3	6,911
6,073	6,167	paper4	1,487
5,424	5,509	paper5	1,130
15,282	15,607	paper6	4,108
65,540	66,316	pic	7,338
15,455	15,807	progc	4,063
20,038	20,483	progl	5,114
13,382	13,661	progp	3,105
23,966	24,346	trans	4,642

375,746-
365,005=

10,741*
8=

85,928

Conclusions

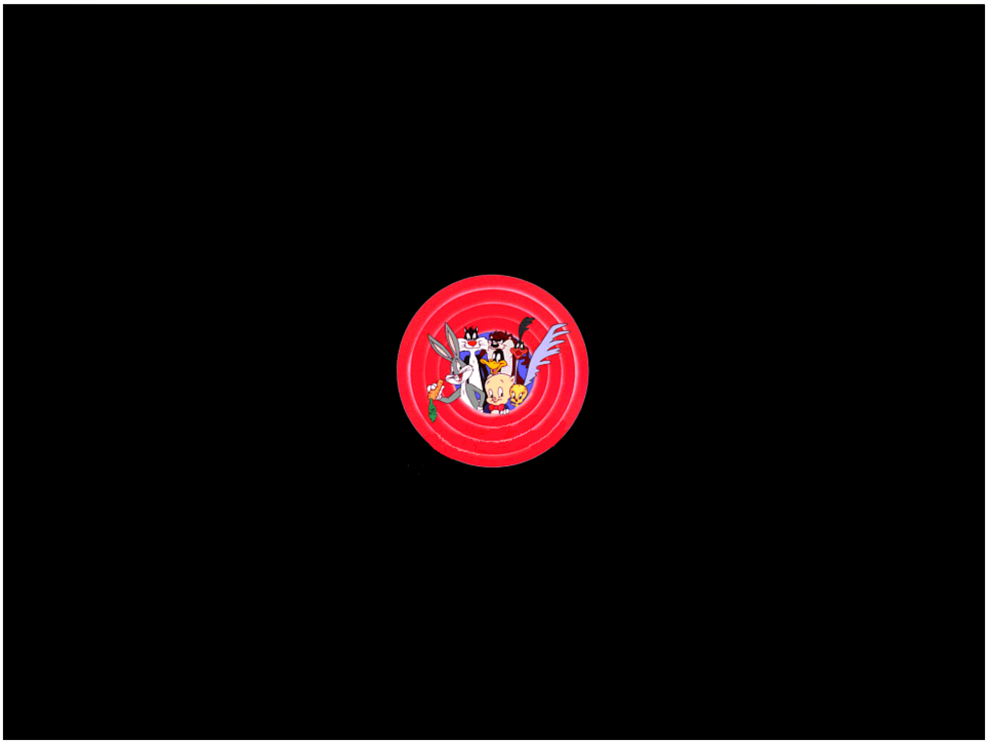
- Authenticity and integrity for LZ-77 files can be obtained efficiently and elegantly
- The degradation of the compression due to the embedding is almost negligible (1%-3% when re-shuffling randomly *all* pointers)

Open problems

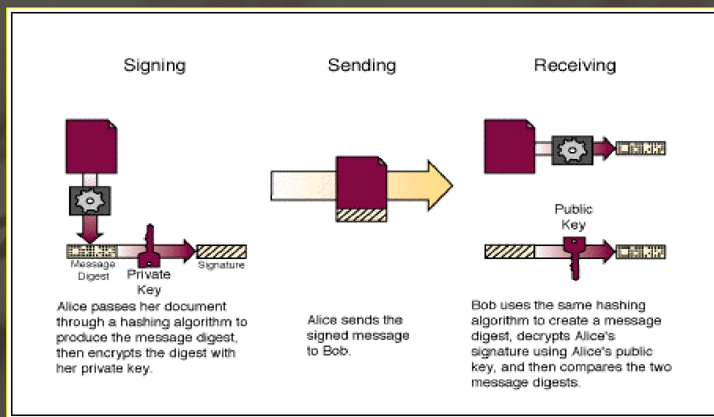
- Can we design a steganography system for LZ-77 compressed texts?
- Can we design a robust watermarking method for LZ-77 compressed texts?
- What about the other types of lossless compression?

"Recompression" attack

- This scheme cannot be used as a stego-system
- Mallory can use a very powerful attack, which removes the secret message
  - Decompress *T'* with standard LZ ? *T*
  - Compress *T* with standard LZ ? *T''*
  - Compare *T'* with *T''*
  - If *T'*? *T''* then send T'' … the message is gone



Typical solution using PKC



Picture from Verisign.com

Advantages over PKC signatures

- No additional data, simplifies file manipulation
- Allow one to embed any information (self-embedding?)
- A casual observer would hardly suspect the presence of the watermark

Security

- Proof: Suppose there exists an algorithm *A* which retrieves the watermark from the text *T'* in poly-time. Choose *T*="ababab", set *i=4*, and run LZS-77. We have $a_0=H(k,5,1,3)$. We get a_1 by running BBS. We use a_0,a_1 to compute the random permutation. If *A* is able to retrieve the watermark it is also capable of predicting a_1, which is known to be computationally hard.

Discovery, Compression, IH

- Pattern discovery: repetitive patterns are unveiled as carriers of information and structure
- Data compression: repetitive patterns are regarded as redundancies and sought to be removed

---

- Information hiding: exploit redundancy to hide secret messages

$|T|$ vs. $|W|$

- If the text is too short, then append some irrelevant data at the end of *T*
- If the text is too long, then use a randomly chosen subset of the phrases with multiplicity *q>1,* for all the others phrases choose pointers randomly

Avg length of the longest prefix