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.
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 an 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 \textit{(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
Which of these pointers do we choose?

By choosing one of these pointers we are “hiding” two bits of the watermark. Note that we are not changing LZ-77
“Dear Bob, How are you doing today? …”

W = H_k(T)

T.gz

document T

secret key k

0110100010010

LZS-77

T.gz

watermarked text T’

T.gz

LZ-77

“Dear Bob, How are you doing today? …”

- Authentic
- Integral

T.gz

0110100010010

secret key k

T.gz

LZS-77

text T

watermarked T’

T.gz

LZS-77

text T

watermarked T’
Method

Multiplicity

- **Definition:** 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 $H(k,i,p_0,p_1,...,p_{q-1})$
  - Generate a uniformly distributed random permutation $R$ of the set $\{0,1,...,q-1\}$
  - Reorder the pointers based on $R$, i.e., $p_{R[0]}, p_{R[1]},..., p_{R[q-1]}$
  - 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
Remark: more bits can be embedded relaxing the greediness
Conjecture: The average multiplicity is $O(1)$, as $n \to \infty$.

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

<table>
<thead>
<tr>
<th>gzip</th>
<th>gzipS</th>
<th>file</th>
<th>bits embedded</th>
</tr>
</thead>
<tbody>
<tr>
<td>43,871</td>
<td>44,087</td>
<td>bib</td>
<td>11,317</td>
</tr>
<tr>
<td>365,005</td>
<td>375,746</td>
<td>book1</td>
<td>154,932</td>
</tr>
<tr>
<td>248,846</td>
<td>255,070</td>
<td>book2</td>
<td>34,678</td>
</tr>
<tr>
<td>69,810</td>
<td>71,343</td>
<td>geo</td>
<td>15,105</td>
</tr>
<tr>
<td>164,199</td>
<td>167,401</td>
<td>news</td>
<td>45,319</td>
</tr>
<tr>
<td>10,707</td>
<td>10,828</td>
<td>obj1</td>
<td>1,429</td>
</tr>
<tr>
<td>93,906</td>
<td>95,494</td>
<td>obj2</td>
<td>17,988</td>
</tr>
<tr>
<td>21,612</td>
<td>22,088</td>
<td>paper1</td>
<td>6,335</td>
</tr>
<tr>
<td>35,078</td>
<td>36,011</td>
<td>paper2</td>
<td>12,541</td>
</tr>
<tr>
<td>20,819</td>
<td>21,320</td>
<td>paper3</td>
<td>6,911</td>
</tr>
<tr>
<td>6,073</td>
<td>6,167</td>
<td>paper4</td>
<td>1,487</td>
</tr>
<tr>
<td>5,424</td>
<td>5,509</td>
<td>paper5</td>
<td>1,130</td>
</tr>
<tr>
<td>15,282</td>
<td>15,607</td>
<td>paper6</td>
<td>4,108</td>
</tr>
<tr>
<td>65,549</td>
<td>66,316</td>
<td>pic</td>
<td>7,538</td>
</tr>
<tr>
<td>15,435</td>
<td>15,807</td>
<td>progc</td>
<td>4,063</td>
</tr>
<tr>
<td>20,038</td>
<td>20,483</td>
<td>progl</td>
<td>5,114</td>
</tr>
<tr>
<td>13,382</td>
<td>13,661</td>
<td>progp</td>
<td>3,105</td>
</tr>
<tr>
<td>23,966</td>
<td>24,346</td>
<td>trans</td>
<td>4,642</td>
</tr>
</tbody>
</table>

### 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

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 $a_4$ by running BBS. We use $a_0, a_4$ 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**

![Graph showing the average length of the longest prefix vs. position in the text.](image.png)