Compression-Compatible Fragile and Semi-Fragile Tamper Detection

Lisa M. Marvel
George W. Hartwig, Jr.
Charles Boncelet, Jr.

Presentation by Peter Macko

Motivation

- Direct Applications
  - Establishing credibility of the image
  - Tracking small changes to the image
- Contribution of the Paper
  - Fragile watermark, which detects any modification of the image
  - Semi-fragile watermark, which detects only significant modifications, ignoring lossy compression and noisy channels

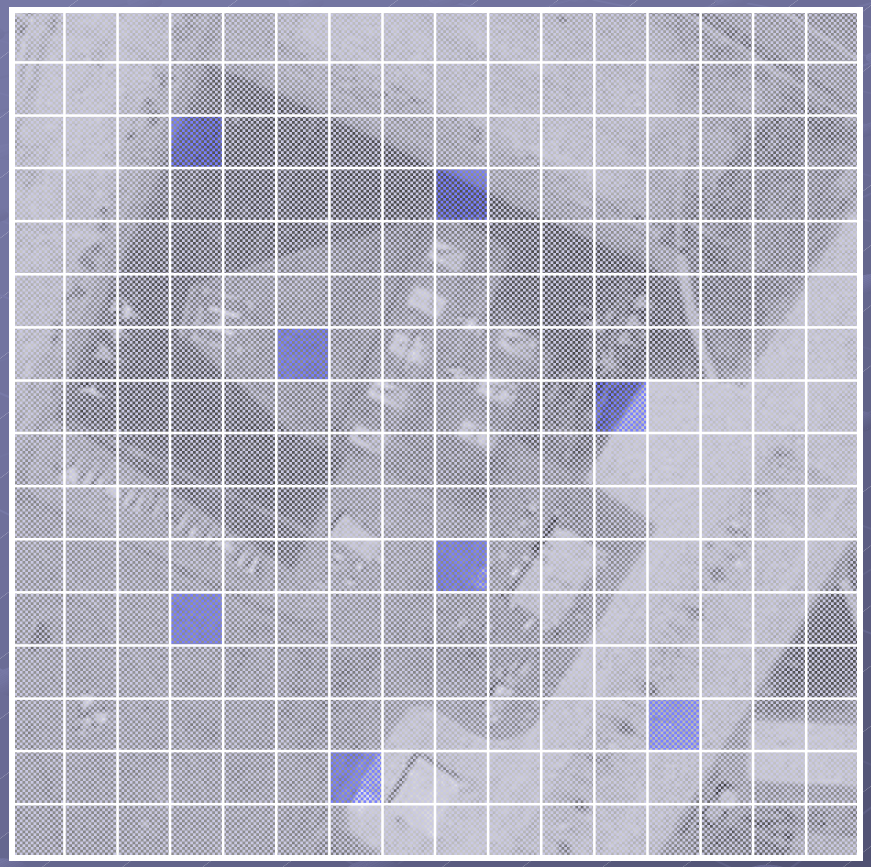
Basic Idea

- Embed a specially crafted watermark to the image using some steganographic method
- Use the watermark (both its presence and its content) to check whether any changes were made to the image after the watermark was embedded

Fragile Tamper Detection

- Any change to the image will either:
  - Destroy the watermark
  - Be detectable using the information embedded in the watermark
- The watermark consists of a cryptographic hash embedded to several random blocks of the image using Stego-JPEG

Embedding the Watermark



Algorithm:

1. Choose *N* random blocks of the image (remember the seed)
2. Compute cryptographic hash of the other blocks
3. Embed the hash into the *N* blocks

Tamper Detection

Algorithm:

1. Using the same seed as in the watermarking stage (key), find the *N* blocks containing the hash
2. Compute the hash of the other blocks
3. Compare the hashes

Practical Issues

- If the hash is strong enough (such as MD5 or MAC), the probability that the hash of the image is not changed by tampering is extremely small (about 2^{-128} for MD5)
- The person tampering with the image does not know the seed, without which the watermark cannot be updated to reflect the changes

Stego-JPEG

- Stego-JPEG is a natural choice of the embedding algorithm for DCT compressed images
- Embeds one bit per one DCT block without practically any loss of visual quality

Stego-JPEG Embedding (1)

To embed bit *b* to block Z:

1. Choose the DCT coefficient closest to the midpoint between two quantization levels
   - Change to this coefficient results in smallest degradation of visual quality

DCT Coefficients	689	-768	248	52	54	-123	108	-43
Quantized	43.1	-64.0	15.5	1.8	0.8	-1.2	0.9	0.43
Rounded	43	-64	16	2	1	-1	1	0

Closest to the midpoint between two quantization levels

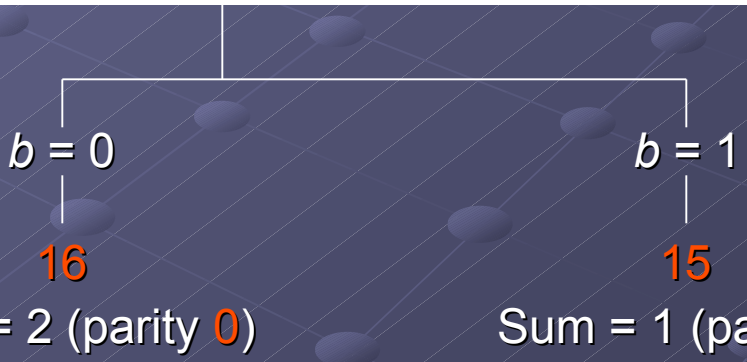
Stego-JPEG Embedding (2)

To embed bit *b* to block Z:

2. Change the parity of the coefficient to make the parity of the sum of the coefficients to match the parity of *b*

DCT Coefficients	689	-768	248	52	54	-123	108	-43
Quantized	43.1	-64.0	15.5	1.8	0.8	-1.2	0.9	0.43
Rounded	43	-64	16	2	1	-1	1	0

Sum of the Coefficients = 2



Stego-JPEG Decoding

1. Compute the sum of the DCT coefficients within the given block
2. The parity of the sum is the embedded bit

Semi-Fragile Tamper Detection

- The watermark is not destroyed by lossy compression or additive noise
- Large changes will destroy the watermark
- Smaller changes can be detected using the embedded thumbnail
- Very small changes will not be detected
- A thumbnail of the image is embedded using an error-resilient stego-technique

Embedding the Watermark

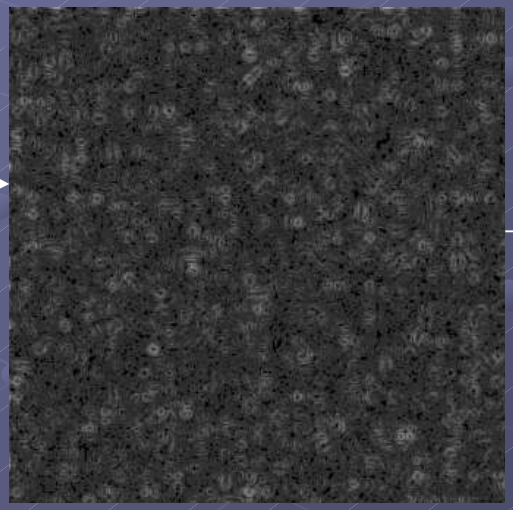
Algorithm:

1. Create the thumbnail of the image
2. Embed the thumbnail using Spread Spectrum Image Stegonagraphy (SSIS)

Thumbnail

SSIS

0xCAFEBABE
Secret Key



SSIS Gaussian
Pseudo-Noise

Add

Cover Image

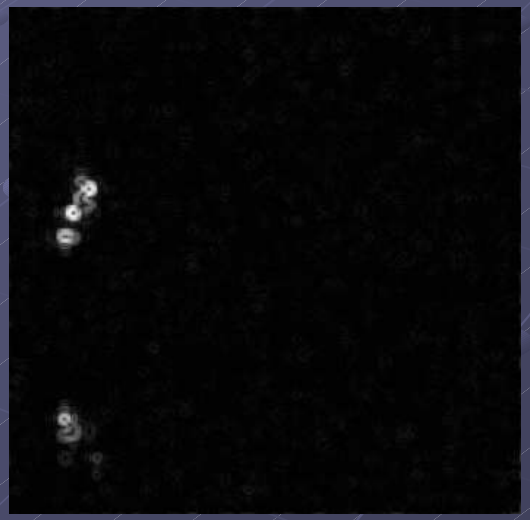
Tamper Detection

Algorithm:

1. Decode the thumbnail
2. Create a new thumbnail of the image
3. Compare the two thumbnails

Thumbnail of a Tampered Image

Decoded Thumbnail (of the original image)



Absolute Difference

Spread Spectrum Image Steganography (SSIS)

- An error-resilient steganographic method
- Can embed long messages with none to small loss of visual quality
- Resistant to small modification of the image, lossy compression, and additive noise
  - Resistant to JPEG compression up to quality level 80

SSIS: Embedding

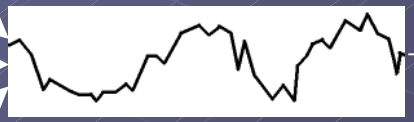
Cover Image

Stego Image

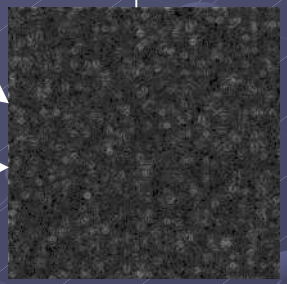
101100100110111…
Encrypted Message

0xCAFEBABE
Secret Key

Error Codes



Modulated Signal



2D Interleaved Signal

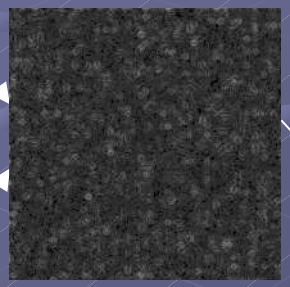
Pseudo-Random Noise

SSIS: Decoding

Stego Image

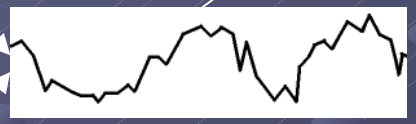
Alpha Trimming

Restored Image



Difference

0xCAFEBABE
Secret Key

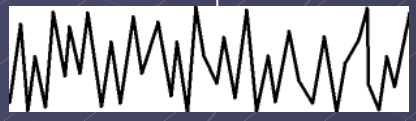


Restored Signal

101100100110111…
Original Message

ECC Decoder

1111010101011011…
Restored Message



Pseudo-Random Noise

Conclusion

- Fragile Watermarks
  - Detects any change to the image
  - Constructed using cryptographic hashes
  - Embedded using a block-based steganographic method
- Semi-Fragile Watermarks
  - Detects only significant manipulations
  - Constructed using thumbnails
  - Embedded using an error-resilient spread-spectrum steganographic method