F5—A Steganographic Algorithm

High Capacity Despite Better Steganalysis

Andreas Westfeld

Outline

Benefits of F5

Properties of existing tools

From Jsteg to F5

How secure is F5?

My challenge to you

Benefits of F5

- High steganographic capacity
- High efficiency via matrix encoding
- Prevents visual attacks
- Resistant to statistical attacks (chi square)
- Uses JPEG as carrier (common in e-mails)
- Source code publicly available

Existing Steganographic Tools

- Their properties:
  - High capacity
  - Weak against visual and statistical attacks
  - Remove significant image content
  - Overwrite LSBs
- Way out?
  - Dilute the changes ➨ less capacity
  - Different embedding operation
  - Different carrier medium

From Jsteg to F5

- Jsteg
  - Easily detected by statistical attacks
  - JPEG compression
  - Embedding function of Jsteg
- F4 (improved embedding operation)
- F5 (advanced efficiency)
  - Permutative straddling
  - Matrix encoding

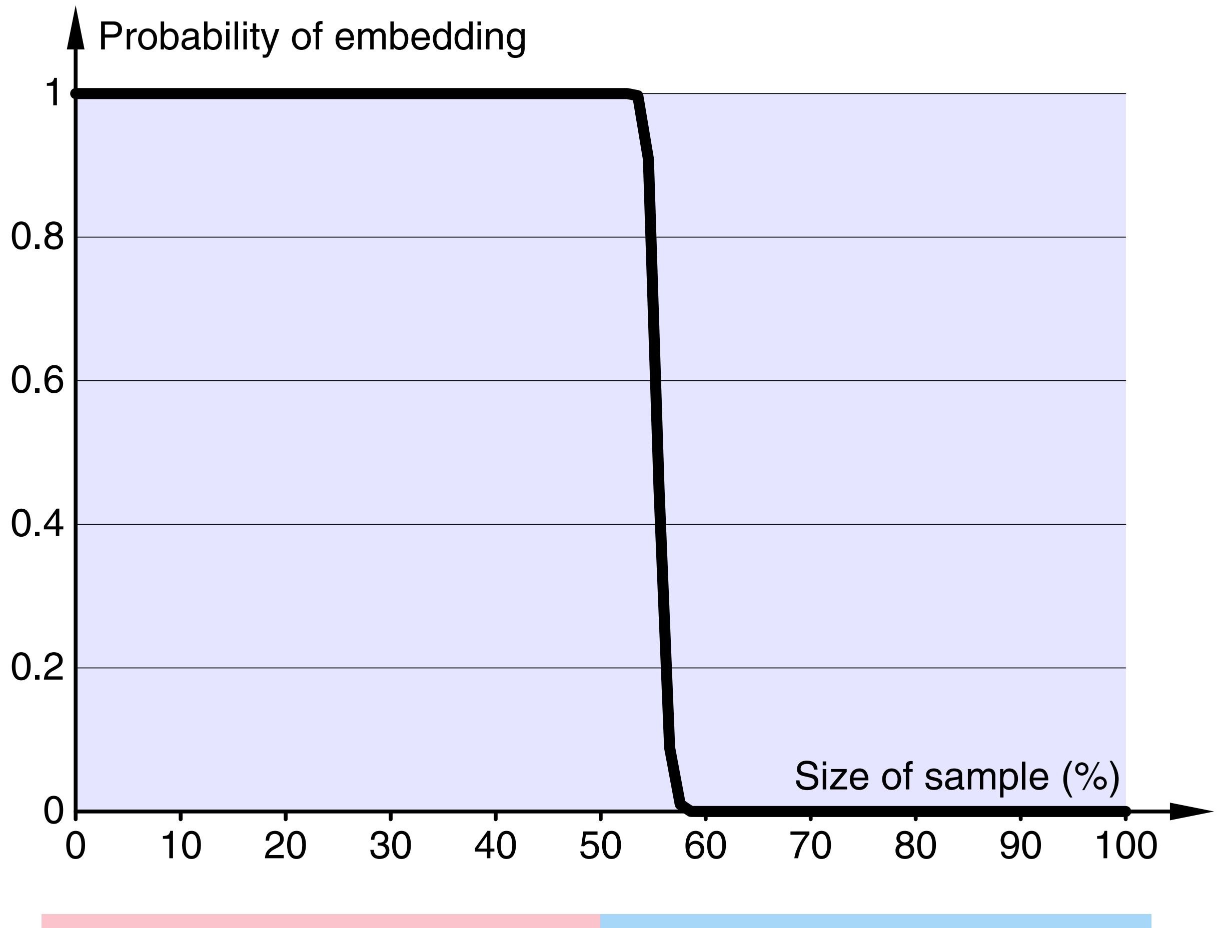
Statistical Attack on Jsteg

Jsteg is safely detected by statistical attack.

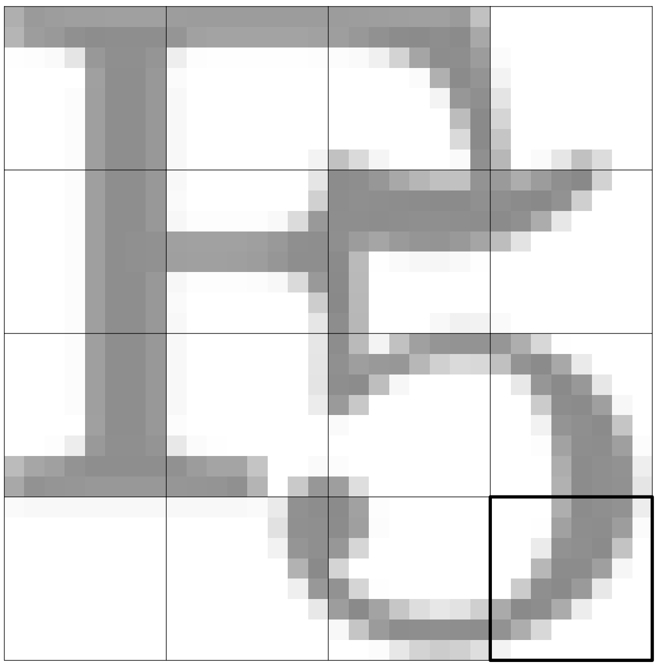
Message

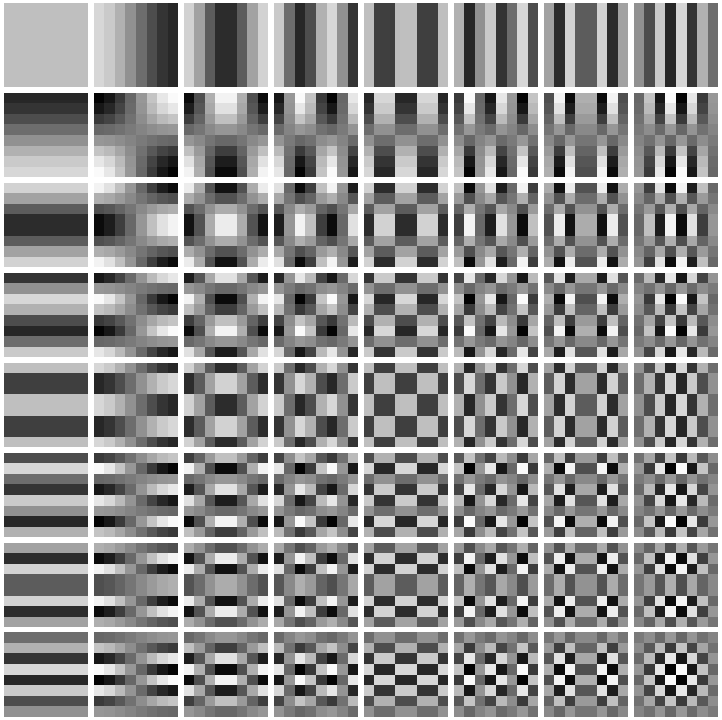
```
M/uogOCS
n21OTx0c
...
```

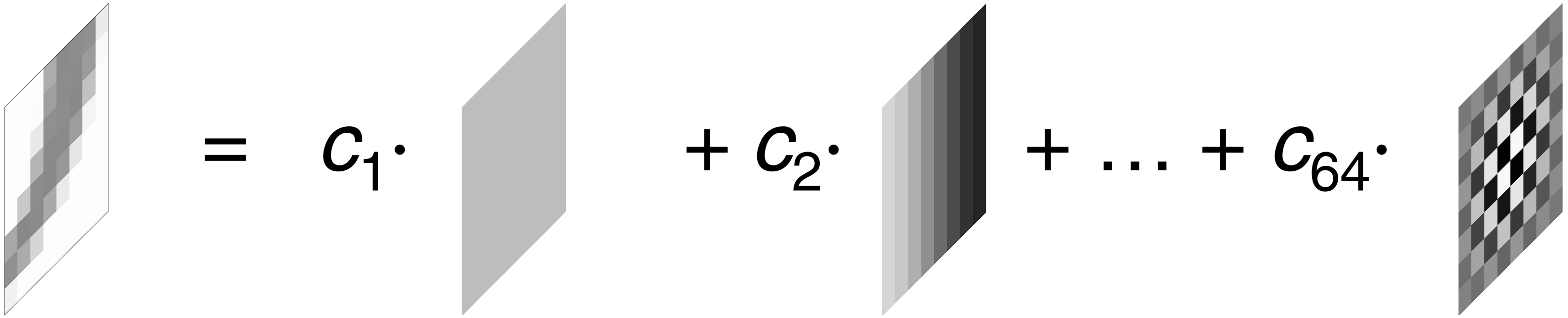
Steganogram



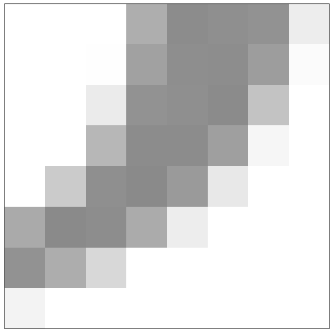
JPEG Coefficients





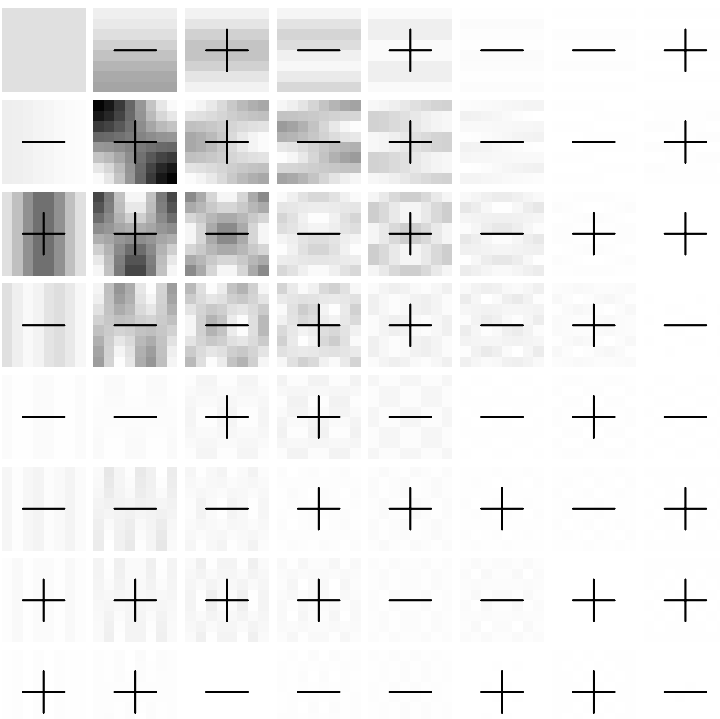


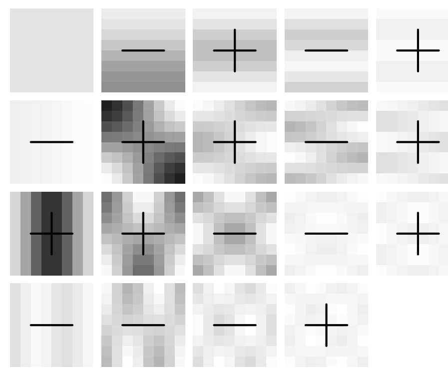
Discrete Cosine Transformation

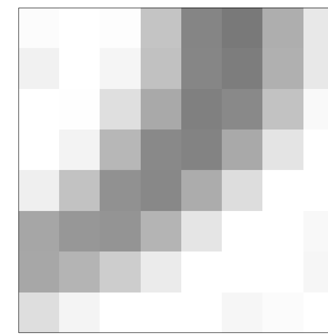


64 brightness values

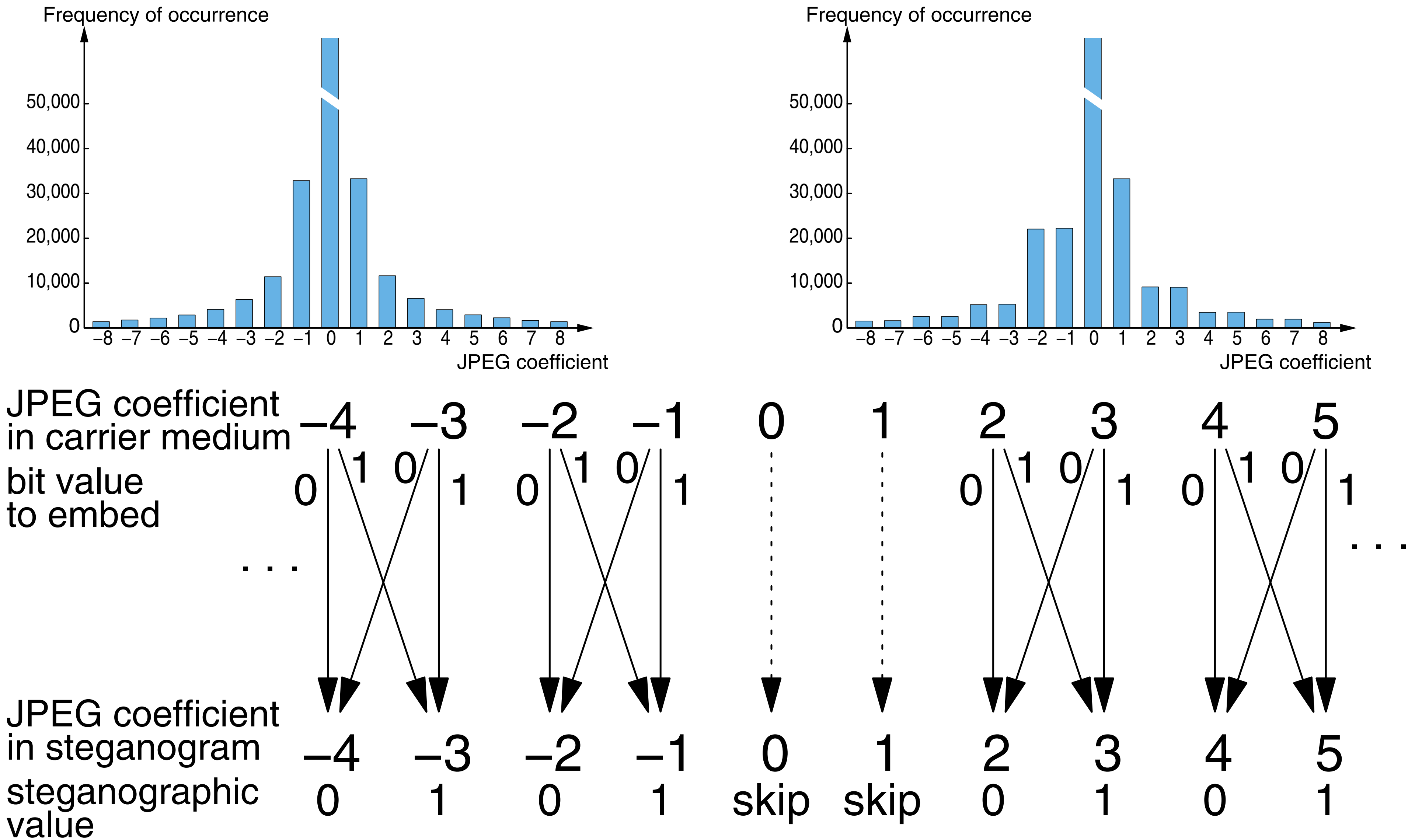
➧ 19 nonzero JPEG coefficients



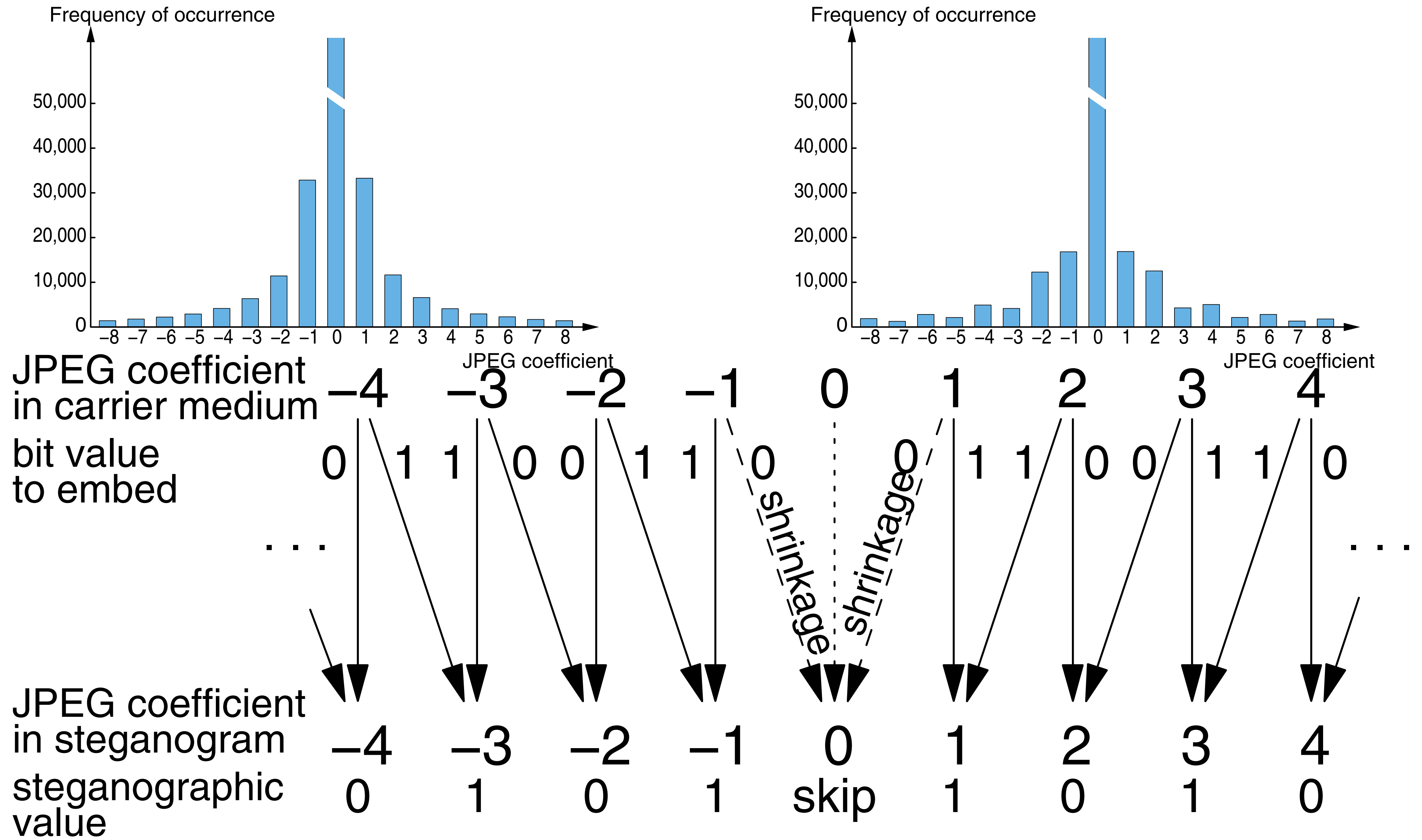




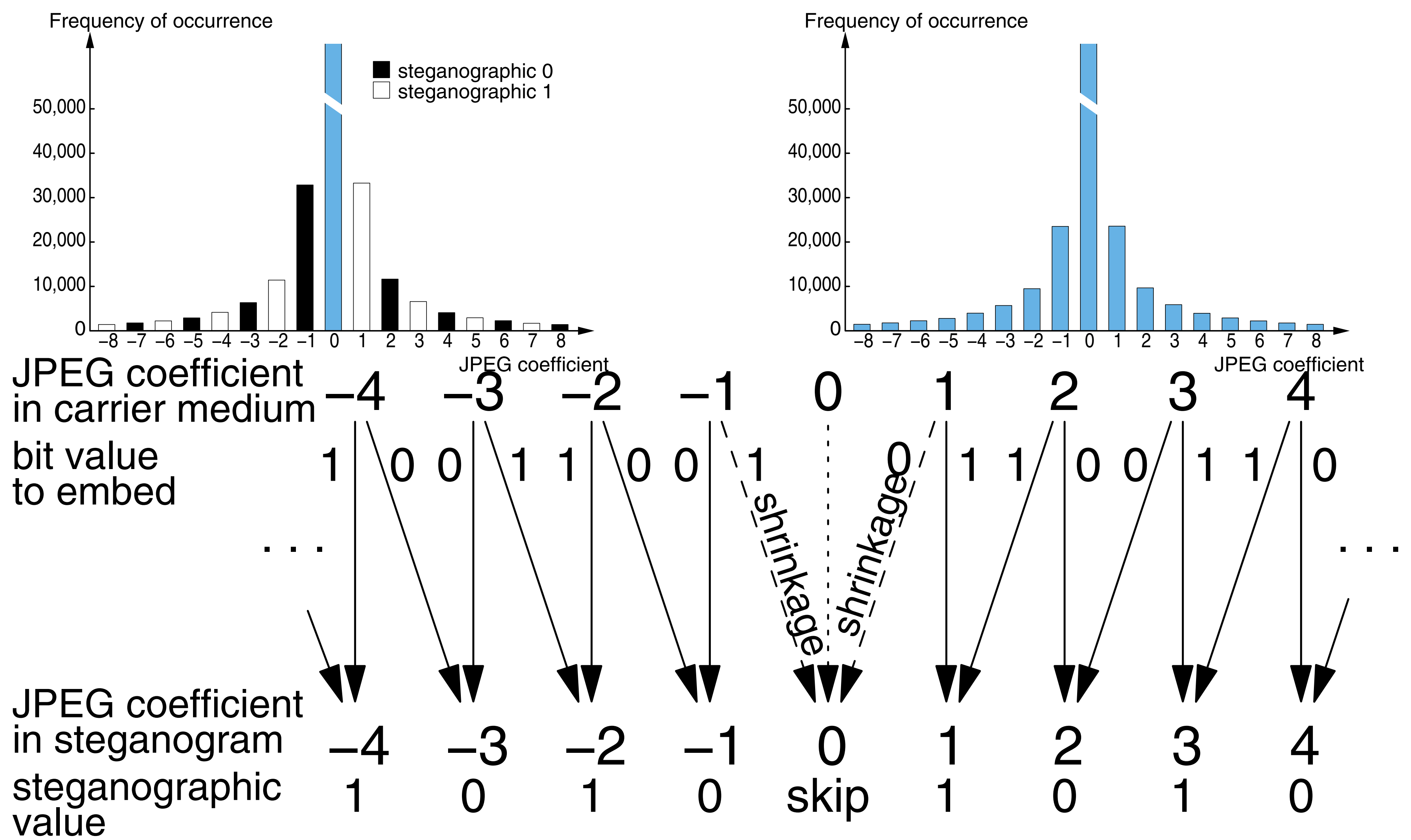
Algorithm Jsteg



Algorithm F3

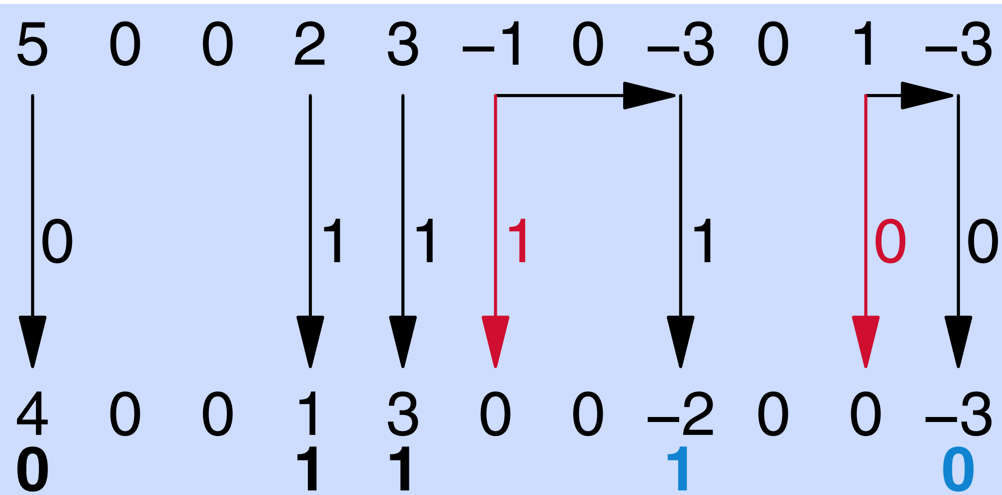


Algorithm F4

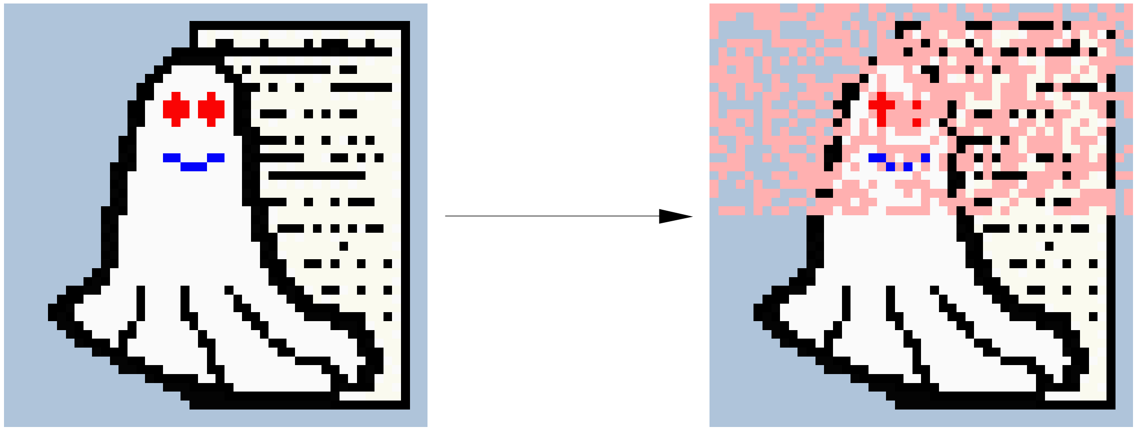


How F4 Embeds “01110”

- Steganographic interpretation
  - Positive coefficients: LSB
  - Negative coefficients: inverted LSB
- Skip 0, adjust coefficients to message bit
  - Decrement positive coefficients
  - Increment negative coefficients
  - Repeat if shrinkage occurs

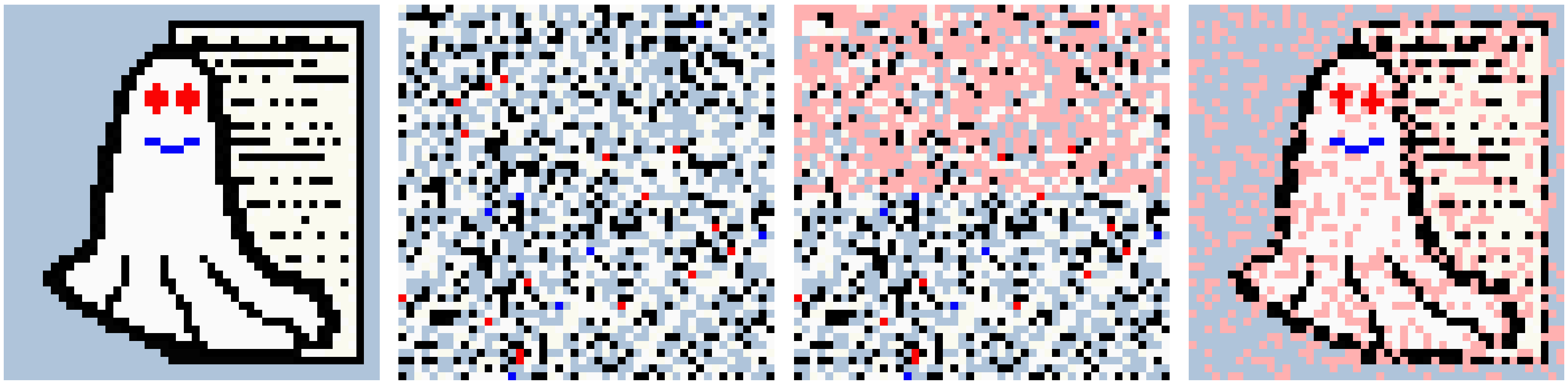


Continuous Embedding



- Changes reside packed at the start of the file (marked with pink)
- Attacks profit from the high change density

Algorithm F5: Straddling



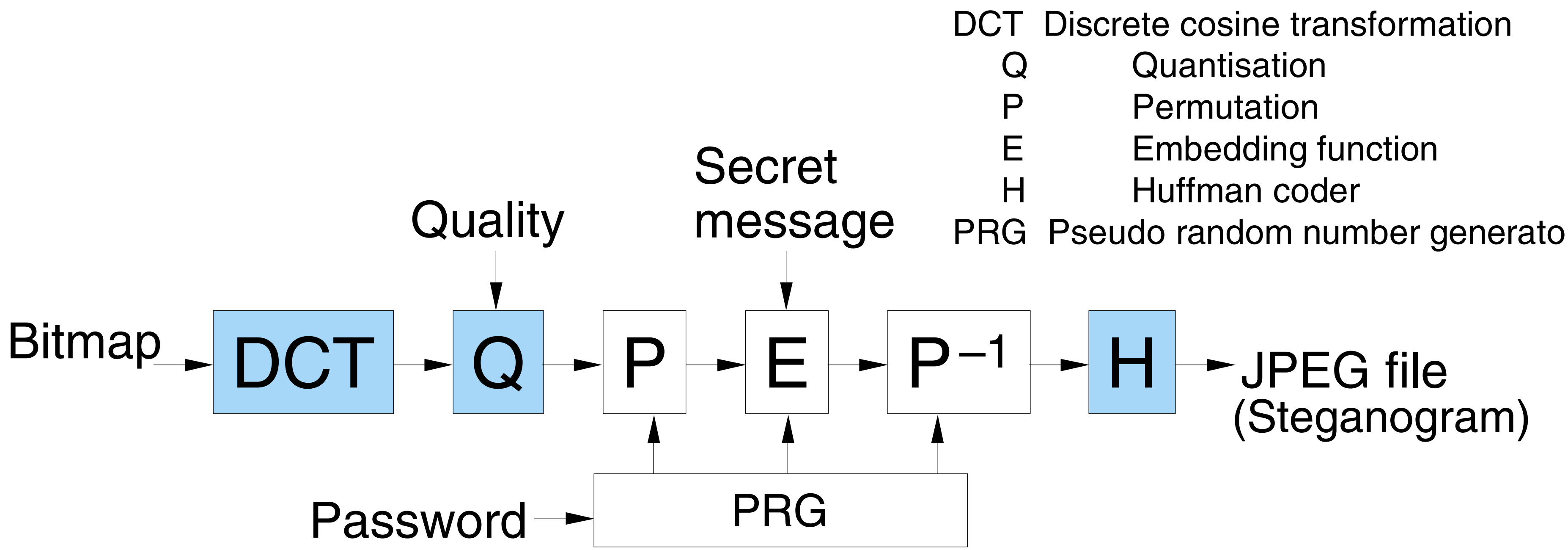
- Permutation equalises the change density
- Scatters the message more uniformly than
  - Key-driven distance schemes
  - Parity block schemes
- Independent of message length

F5 Uses Matrix Encoding

Embed k bits by changing one of $n = 2^k - 1$ places:

k	n	change density	embedding rate	embedding efficiency
1	1	50 %	100 %	2
2	3	25 %	66.7 %	2.7
3	7	12.5 %	42.9 %	3.4
4	15	6.25 %	26.7 %	4.3
k	n			$> k$

Implementation of F5

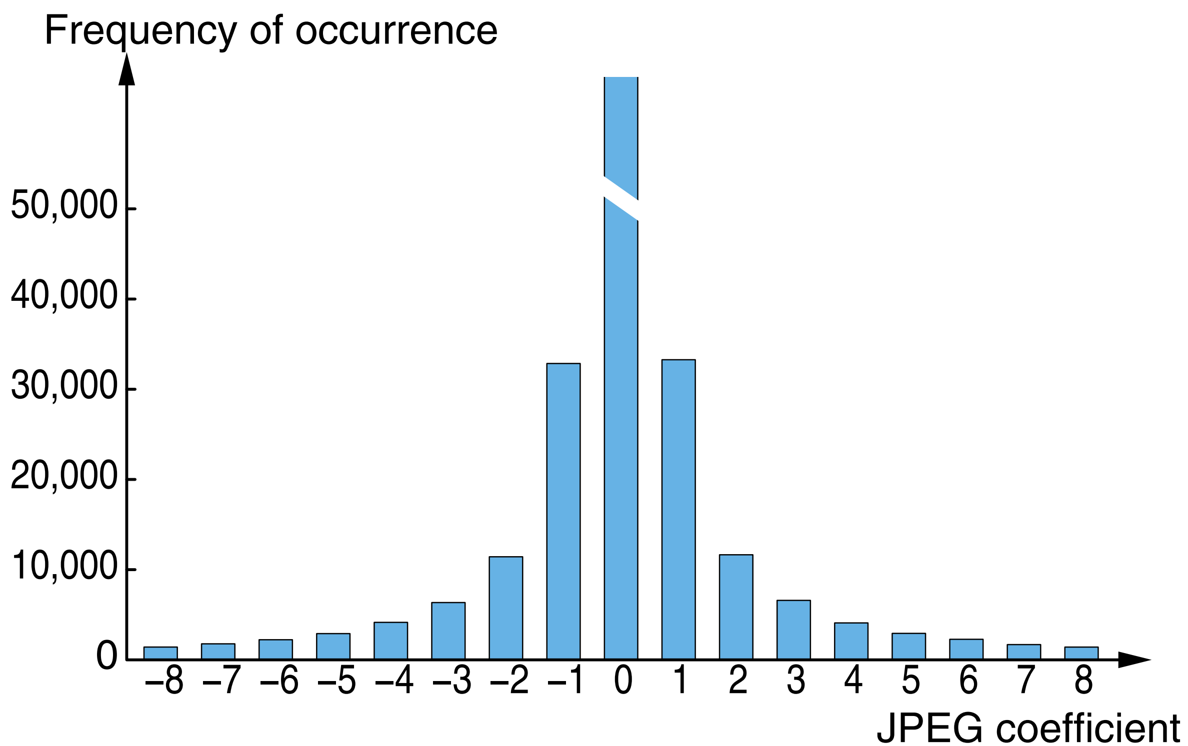


- Password-driven permutation
- Pseudo one time pad for uniformly distributed message
- Matrix encoding with minimal embedding rate
- Core embedding operation like F4

Capacity and Efficiency of F5

- Large carrier medium *expo.bmp (1526 KB)*
- Clean JPEG (80 % Quality) *expo80.jpg (127 KB)*
- Maximum embedded *maxisteg.jpg (113 KB)*
  - Size: 16 KB, i. e. 13 %
  - Efficiency: 1.5 bits per change (incl. shrinkage)
- Short message embedded *ministeg.jpg (127 KB)*
  - Size: 0.2 KB
  - Efficiency: 3.7 bits per change (incl. shrinkage)
  - Without matrix encoding twice as much changes
- Clean JPEG (75 % Quality) *expo75.jpg (113 KB)*

Characteristic Properties



$$P(X=1) > P(X=2) > P(X=3) > P(X=4)$$

$$P(X=1)-P(X=2) > P(X=2)-P(X=3) > P(X=3)-P(X=4)$$

Conclusion

- High steganographic capacity still possible
- High efficiency (embed more bits per change)
- Resistant against statistical attack (chi square)
- Uses a common carrier medium (JPEG)
- Publicly available source code

Further Work

- High steganographic capacity (13 % of a JPEG)
  - stronger attacks possible?
- Finer gradation of embedding efficiency by (d_{max}, n, k) matrix encoding with $d_{max} > 1$
  - too complex (too slow)?

westfeld@inf.tu-dresden.de