Steganographic Key Leakage Through Payload Metadata

Tomáš Pevný

pevnak@gmail.com

Agent Technology Center, Czech Technical University in Prague

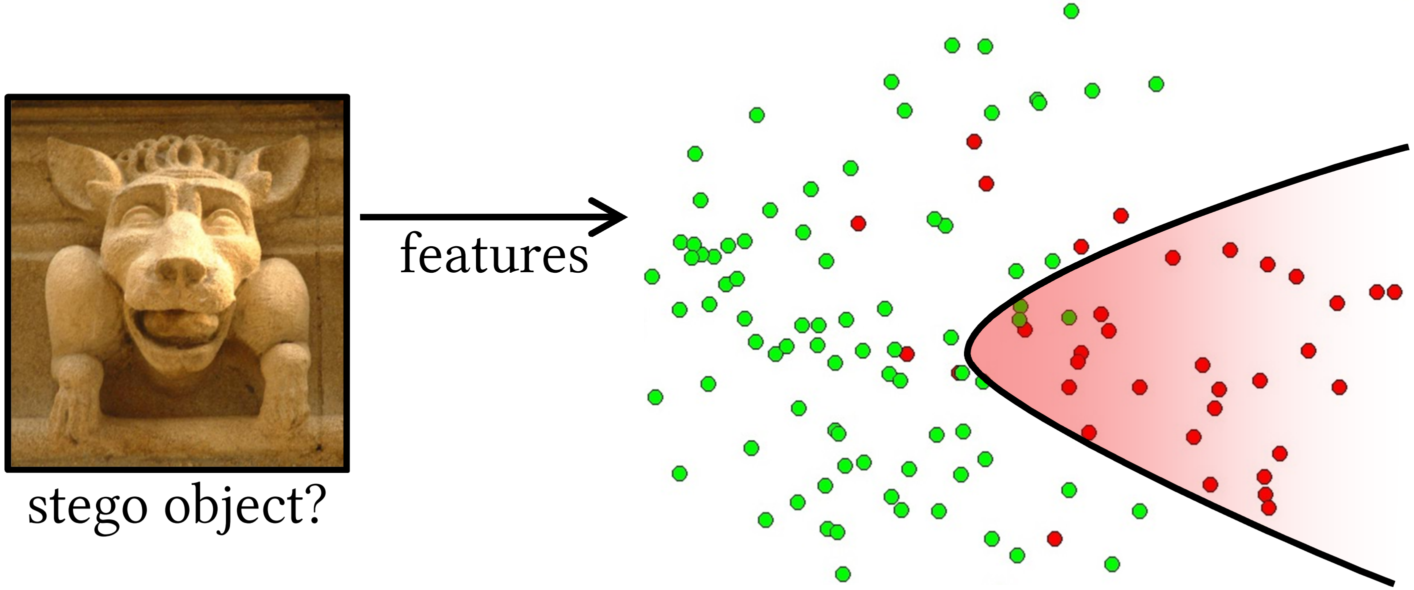
Andrew Ker

adk@cs.ox.ac.uk

Department of Computer Science, Oxford University

2nd ACM Information Hiding Multimedia & Security Workshop
Salzburg, 12 June 2014

Statistical attacks

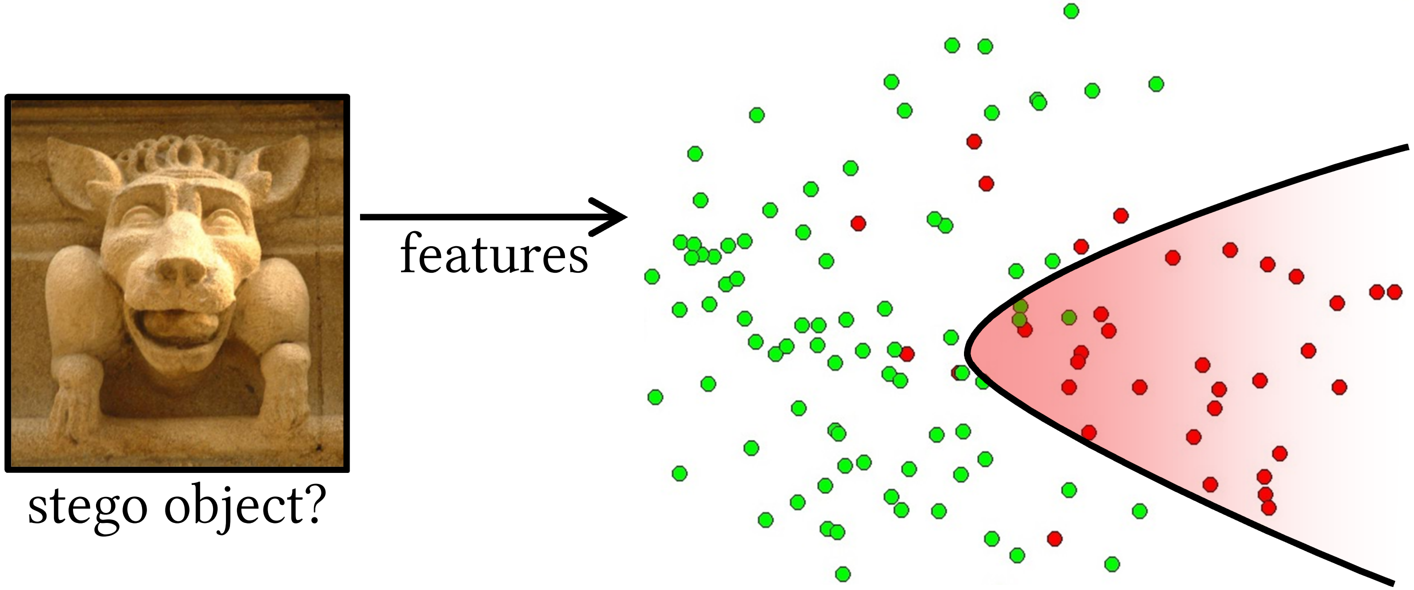


stego object?

features

Sophisticated, powerful, but…

Statistical attacks



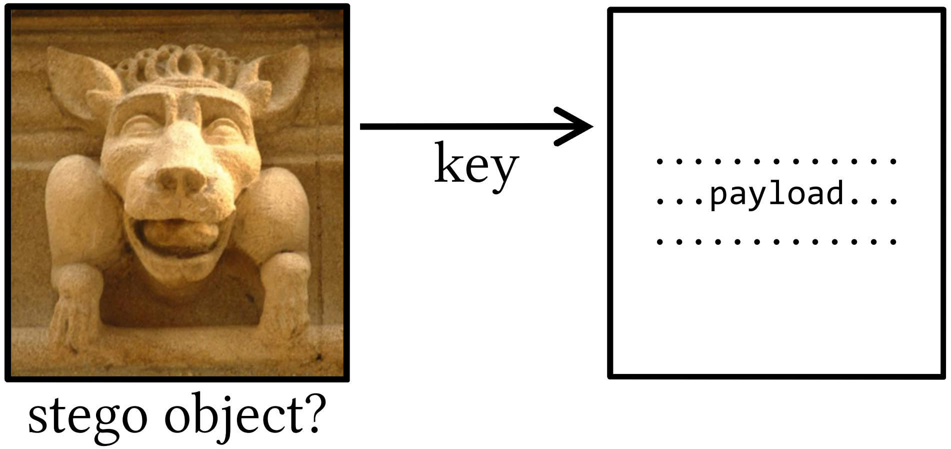
features

stego object?

Sophisticated, powerful, but...

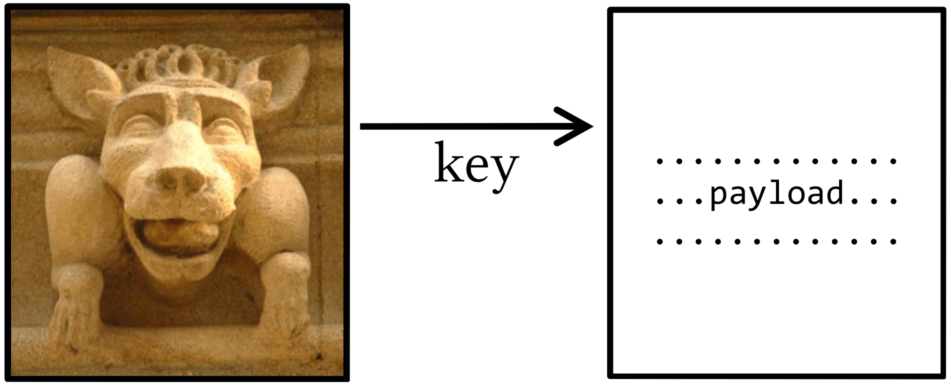
- Can never give certainty.
- Can never know exactly how accurate it is.

Exhaustion attack



Try every key until you recognise a payload.

Exhaustion attack



key

....
...payload...
....

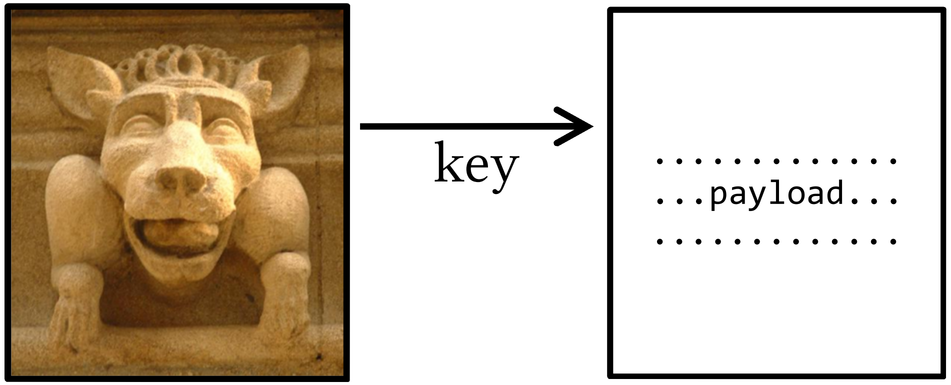
stego object?

Try every key until you recognise a payload.

Not feasible if the keyspace is 64 bits, but

- feasible if 32-bit keyspace, or maps into 32-bit space, or
- feasible if keys derived from passwords.

Exhaustion attack



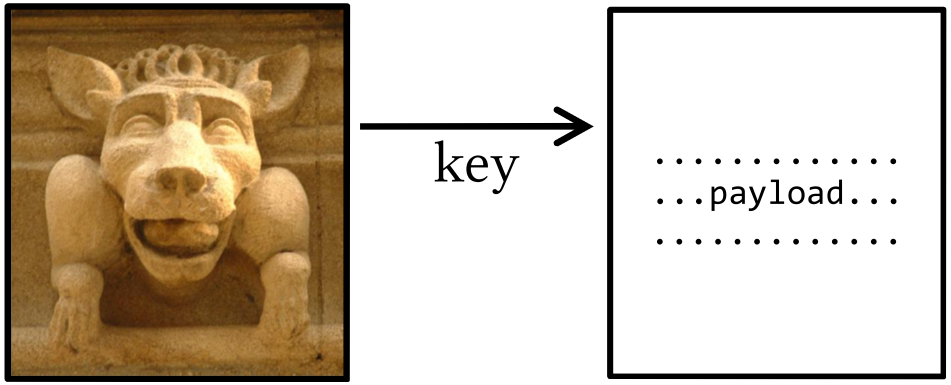
stego object?

Try every key until you **recognise a payload**.

Making payload unrecognisable is difficult:

- use unstructured plaintext?
- encrypt with second password?

Exhaustion attack



key

```
.............
...payload...
.............
```

stego object?

Assumptions

- Keyspace exhaustible.
- Plaintext unrecognisable.

Seek statistical evidence that one key is more likely,
or a short list of keys for a second attack on the plaintext.

Related work

Assumptions

- Keyspace exhaustible.
- Plaintext unrecognisable.

Provos [2001]

For each key, check consistency of OutGuess ‘header block’.

Fridrich et al. [2004], Böhme et al. [2012]

For each key, compare statistics of used vs. unused locations.

Ker [2007], Quach [2011+]

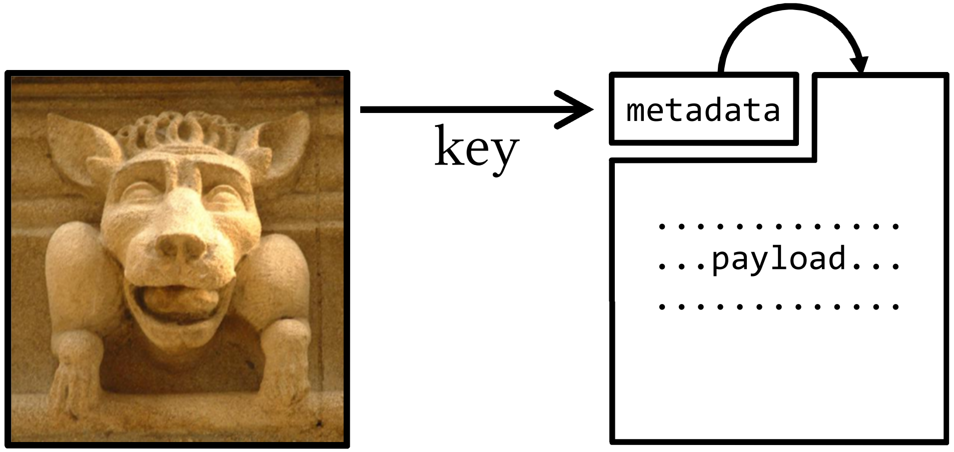
Look for correlated residuals between different stego images.

Model

- Keyspace exhaustible.
- Plaintext unrecognisable.
- Multiple stego objects embedded with same key.

Model

- Keyspace exhaustible.
- Plaintext unrecognisable.
- Multiple stego objects embedded with same key.
- Payload decoded via metadata:



key

metadata

....
...payload...
....

stego object?

Payload metadata

Most implementations use metadata:

- Payload size (to know when to stop decoding).
- Hamming code parameters.
- Syndrome Trellis Code parameters.
- …

Intersection attack

```
For each stego image,
    for each key,
        decode metadata & discard impossible keys.
```

Intersection attack

```
For each stego image,
  for each key,
    decode metadata & discard impossible keys.
```

Example

- OutGuess
- Uniformly random message length
- Keyspace: 2 million passwords
- Metadata = message length

- Discard length > capacity
- Experiment repeated 1000 times

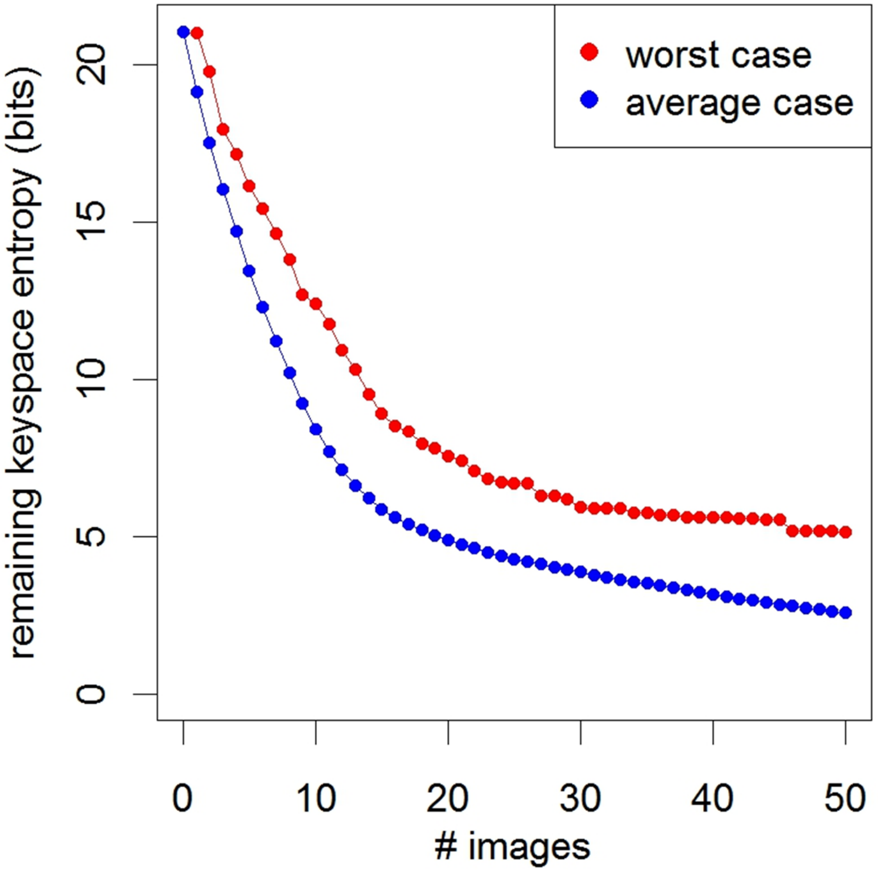
Intersection attack

```
For each stego image,
  for each key,
    decode metadata & discard impossible keys.
```

Example

- OutGuess
- Uniformly random message length
- Keyspace: 2 million passwords
- Metadata = message length

- Discard length > capacity
- Experiment repeated 1000 times



Intersection attack

```
For each stego image,
  for each key,
    decode metadata & discard impossible keys.
```

Countermeasure

Use proper ‘padding’ to make all metadata possible.

e.g. `length = metadata (mod capacity)`

Intersection attack

```
For each stego image,
  for each key,
    decode metadata & discard impossible keys.
```

Countermeasure

Use proper ‘padding’ to make all metadata possible.

e.g. $\texttt{length} = \texttt{metadata} \pmod{\underbrace{\texttt{capacity}}_{\textit{Can this be determined by the receiver?}}}$

Intersection attack

```
For each stego image,
  for each key,
    decode metadata & discard impossible keys.
```

Countermeasure

Use proper ‘padding’ to make all metadata possible.

e.g. $\texttt{length} = \texttt{metadata} \pmod{\underbrace{\texttt{capacity}}_{\textit{Can this be determined by the receiver?}}}$

e.g. $\texttt{code parameter} = \texttt{metadata} \pmod{\texttt{maximum}}$

Quantitative steganalysis

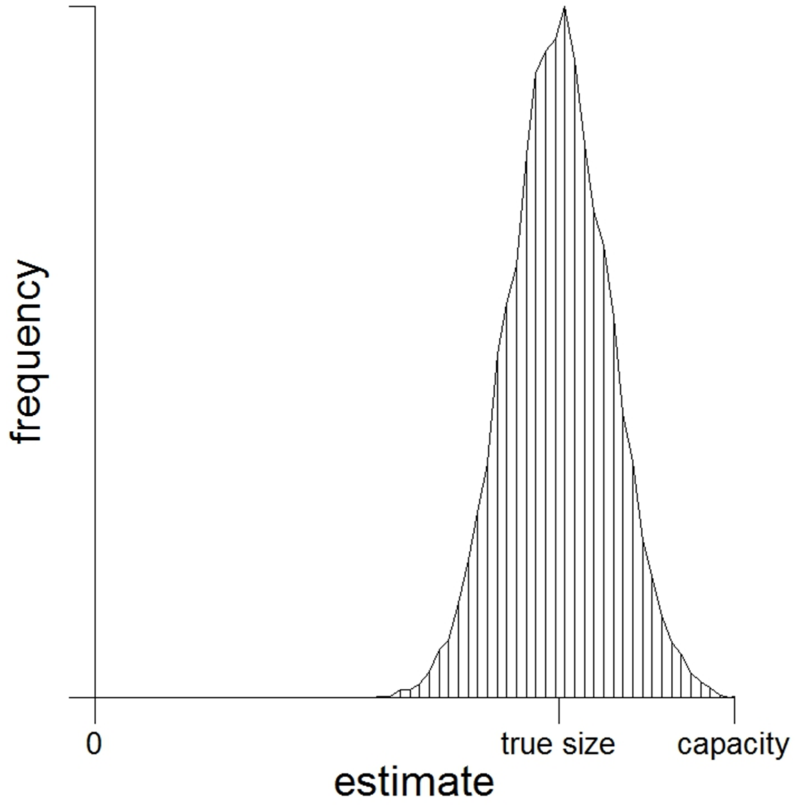
Attacking the embedding, can often estimate the length of payload in a stego image:

- old-fashioned ‘structural steganalysis’,
- support vector regression based on features, etc.

Quantitative steganalysis

Attacking the embedding, can often estimate the length of payload in a stego image:

- old-fashioned ‘structural steganalysis’,
- support vector regression based on features, etc.



Bayesian key inference

For each key,

decode metadata & compute posterior:

length decoded from metadata

key

$$p(k|y) = \frac{\mathrm{P}\big(y|x(k)\big)p(k)}{\sum_{k'} \mathrm{P}\big(y|x(k')\big)p(k')} \propto \mathrm{P}\big(y|x(k)\big)p(k)$$

observed stego object

Bayesian key inference

For each key,

decode metadata & compute posterior:

$$p(k|y) = \frac{\overbrace{\mathrm{P}\big(y|x(k)\big)}^{\substack{\textit{behaviour of estimator}\\ \textit{(determined experimentally)}}}\overset{\textit{prior (uniform)}}{p(k)}}{\sum_{k'} \mathrm{P}\big(y|x(k')\big)p(k')} \propto \mathrm{P}\big(y|x(k)\big)p(k)$$

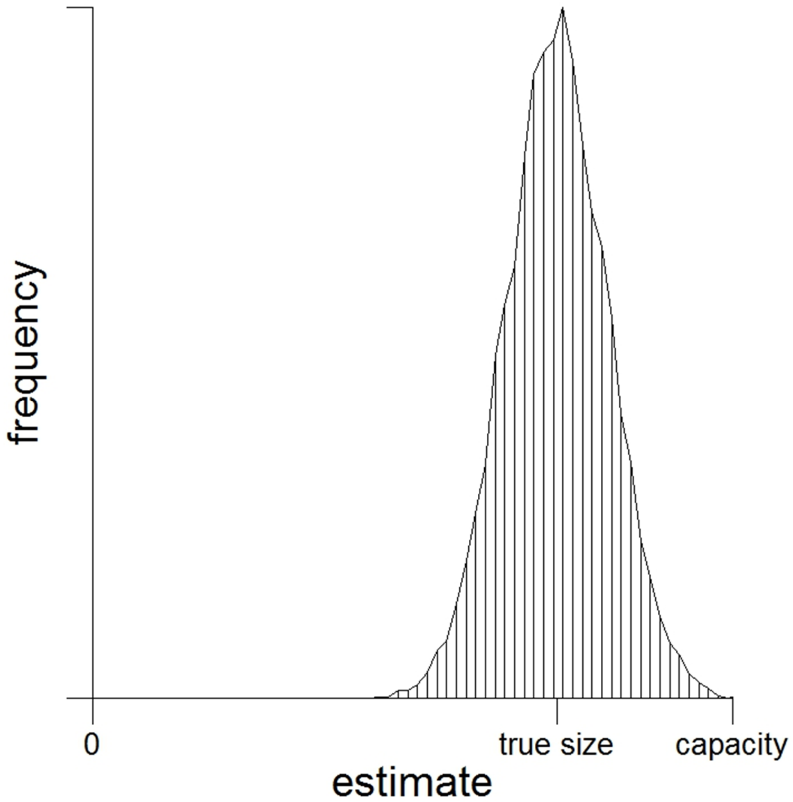
Bayesian key inference

For each key,

decode metadata & compute posterior:

prior (uniform)

$$p(k|y) = \frac{\overbrace{\mathrm{P}\big(y|x(k)\big)}^{\textit{behaviour of estimator (determined experimentally)}}p(k}{\sum_{k'}\mathrm{P}\big(y|x(k')\big)p(}$$



Bayesian key inference

For each key,

decode metadata & compute score

$$\log p(k|y_1,\ldots,y_n) = \log p(k) + \sum_{i=1}^{n} \log \mathrm{P}\big(y_i|x(k)\big)$$

Bayesian key inference

For each key,

decode metadata & compute score

$$\log p(k|y_1, \ldots, y_n) = \log p(k) + \sum_{i=1}^{n} \log \mathrm{P}\big(y_i|x(k)\big)$$

Example

- OutGuess
- Uniformly random message length
- Keyspace: 2 million passwords
- Metadata = message length

- PF-548 features → length estimate
- Experiment repeated 1000 times

Bayesian key inference

For each key,

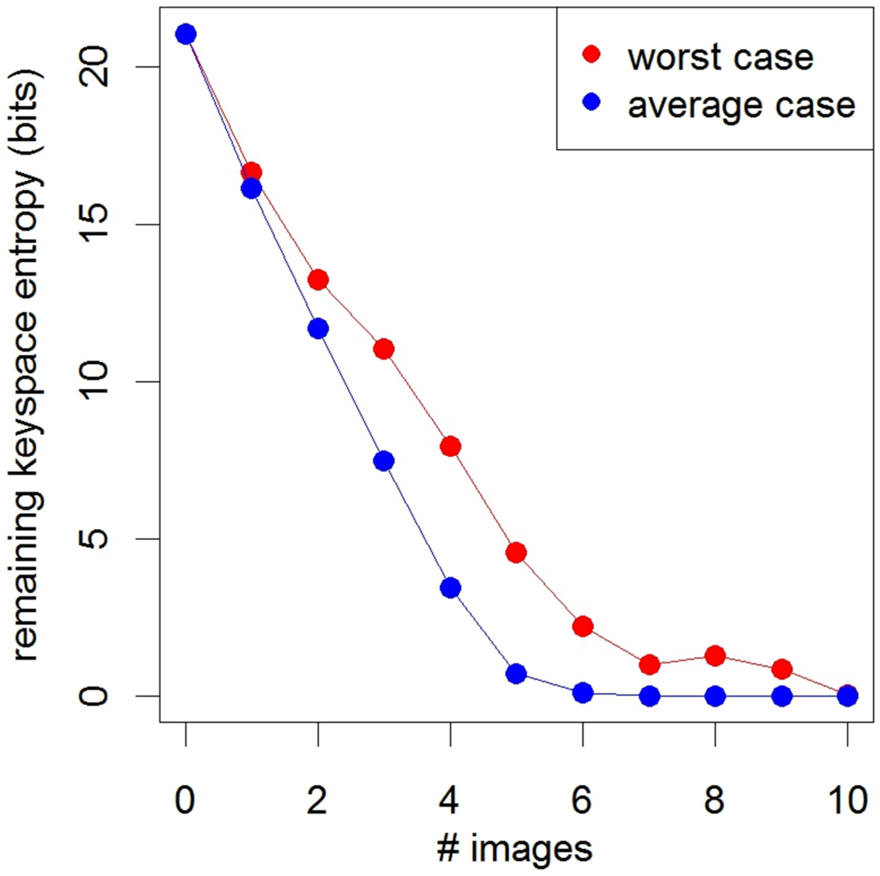
decode metadata & compute score

$$\log p(k|y_1, \dots, y_n) = \log p(k) + \sum_{i=1}^{n} \log \mathrm{P}\big(y_i|x(k)\big)$$

Example

- OutGuess
- Uniformly random message length
- Keyspace: 2 million passwords
- Metadata = message length

- PF-548 features → length estimate
- Experiment repeated 1000 times



Bayesian key inference

For each key,

decode metadata & compute score

$$\log p(k|y_1, \ldots, y_n) = \log p(k) + \sum_{i=1}^{n} \log \mathrm{P}\big(y_i|x(k)\big)$$

Countermeasure?

Key inference has ‘exponential power’:

extracted metadata is **independent** across images (if the key is incorrect).

Try to make it **dependent**, as for correct keys?

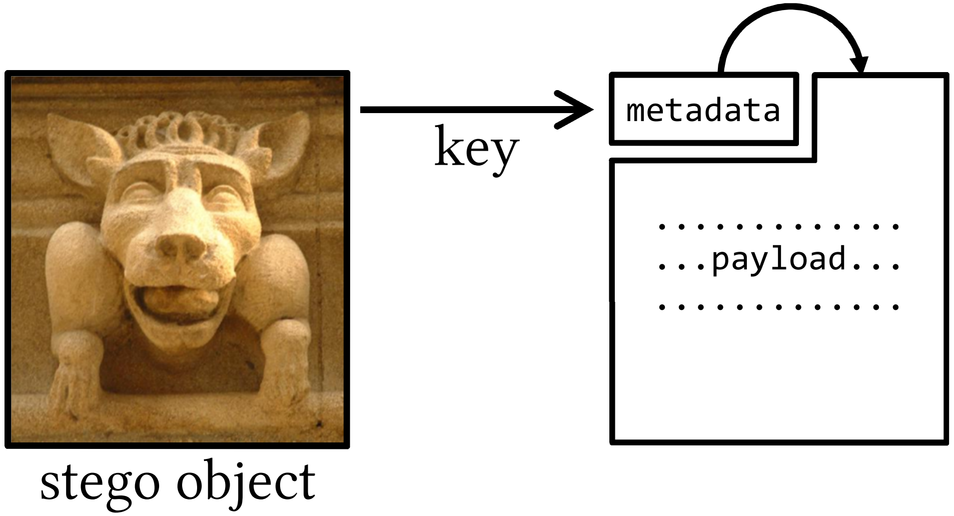
Bayesian key inference

For each key,

decode metadata & compute score

$$\log p(k|y_1,\ldots,y_n) = \log p(k) + \sum_{i=1}^{n} \log \mathrm{P}\big(y_i|x(k)\big)$$

Countermeasure?



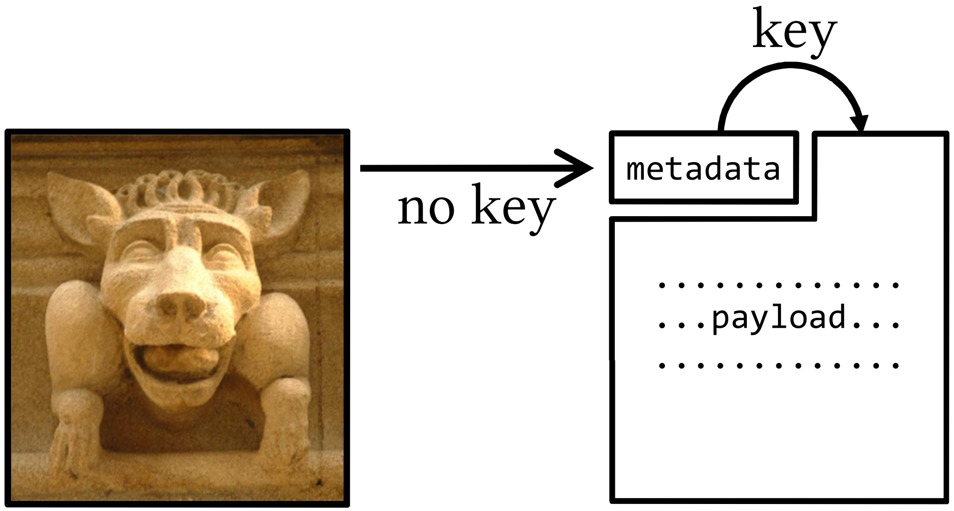
Bayesian key inference

For each key,

decode metadata & compute score

$$\log p(k|y_1, \ldots, y_n) = \log p(k) + \sum_{i=1}^{n} \log \mathrm{P}\big(y_i|x(k)\big)$$

Countermeasure?



key

no key

metadata

....
...payload...
....

stego object

Bayesian key inference

For each key,

decode metadata & compute score

$$\log p(k|y_1, \ldots, y_n) = \log p(k) + \sum_{i=1}^{n} \log \mathrm{P}\big(y_i|x(k)\big)$$

Countermeasure?

e.g. `length = (metadata + key)` (mod `capacity)`

and the metadata is stored at a fixed location

Bayesian key inference

For each key,

decode metadata & compute score

$$\log p(k|y_1, \ldots, y_n) = \log p(k) + \sum_{i=1}^{n} \log \mathrm{P}\big(y_i|x(k)\big)$$

Countermeasure?

- Simulated 16-bit payload size
- Uniformly random message length
- `length = (metadata + key)`
`(`mod `capacity)`

- PF-548 features → length estimate
- Repeated 1000 times

Bayesian key inference

For each key,

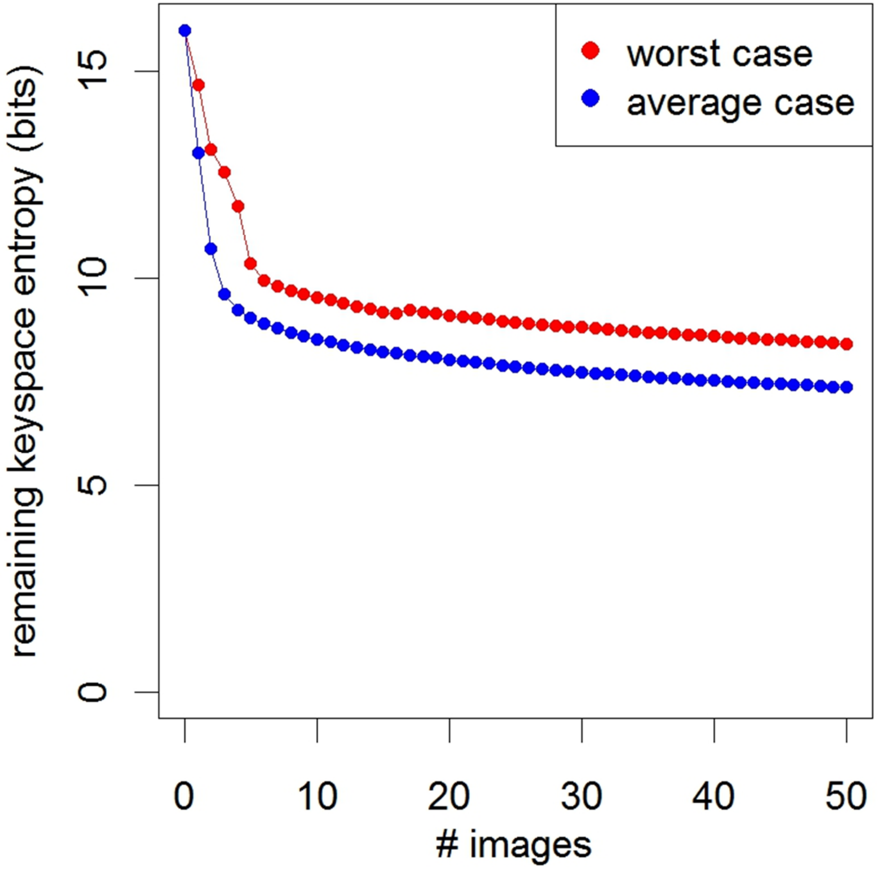
decode metadata & compute score

$$\log p(k|y_1,\ldots,y_n) = \log p(k) + \sum_{i=1}^{n} \log \mathrm{P}\big(y_i|x(k)\big)$$

Countermeasure?

- Simulated 16-bit payload size
- Uniformly random message length
- `length = (metadata + key)`
  (mod `capacity)`

- PF-548 features → length estimate
- Repeated 1000 times



Bayesian key inference

For each key,

decode metadata & compute score

$$\log p(k|y_1, \ldots, y_n) = \log p(k) + \sum_{i=1}^{n} \log \mathrm{P}\big(y_i | x(k)\big)$$

Countermeasure?

e.g. `length = (metadata + key)` (mod `capacity)`

and the metadata is stored at a fixed location

However, this introduces new statistical attacks.

Bayesian key inference II

If the metadata does not determine payload length, it probably gives information about it:

- Optimal Hamming code size determined by relative payload.
- STC width closely related to inverse payload.

Bayesian key inference II

If the metadata does not determine payload length, it probably gives information about it:

- Optimal Hamming code size determined by relative payload.
- STC width closely related to inverse payload.

$$p(k|y) \propto \sum_x \mathrm{P}(y|\overset{\textit{length}}{x})\,\mathrm{P}\big(x|\underset{\textit{coding parameter(s)}}{m(k)}\big)\,p(k)$$

Bayesian key inference II

If the metadata does not determine payload length, it probably gives information about it:

- Optimal Hamming code size determined by relative payload.
- STC width closely related to inverse payload.

$$p(k|y) \propto \sum_x \mathrm{P}(y|x)\overbrace{\mathrm{P}\big(x|\underset{\text{coding parameter(s)}}{m(k)}\big)}^{\text{probably uniform between certain limits}}p(k)$$

Bayesian key inference II

For each key,

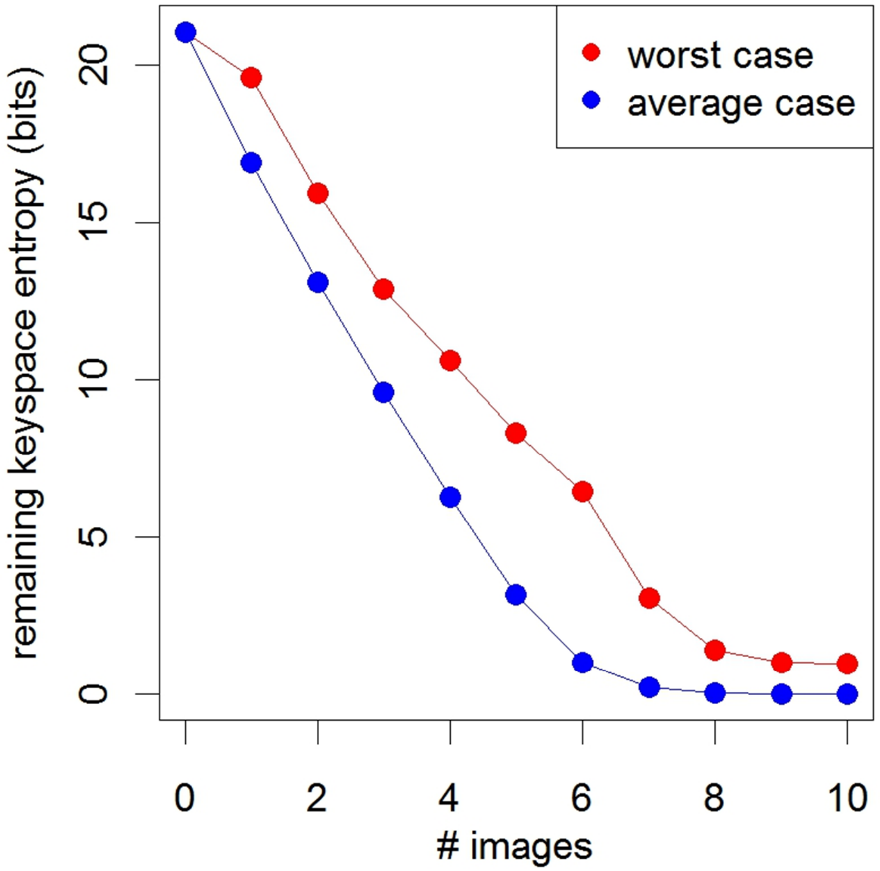
decode metadata & compute score

$$\log p(k|y_1, \ldots, y_n) = \log p(k) + \sum_{i=1}^{n} \log \sum_x \mathrm{P}(y_i|x)\mathrm{P}\big(x|m(k)\big)$$

Example

- OutGuess
- Keyspace: 2 million passwords
- Hamming $[2^p, 2^p - p - 1]$ code
- Metadata = p

- PF-548 features → length estimate
- Repeated 1000 times



Conclusions

Presented ways to improve exhaustion attacks through statistical steganalysis evidence.

We are attacking implementation weaknesses, not steganographic weaknesses.

Conclusions

Presented ways to improve exhaustion attacks through statistical steganalysis evidence.

We are attacking implementation weaknesses, not steganographic weaknesses.

Implementations can avoid all these attacks if:

- their keyspace is not exhaustible, or
- keys are never reused, or
- no metadata is stored…

… but such mistakes are plausible and common.

Conclusions

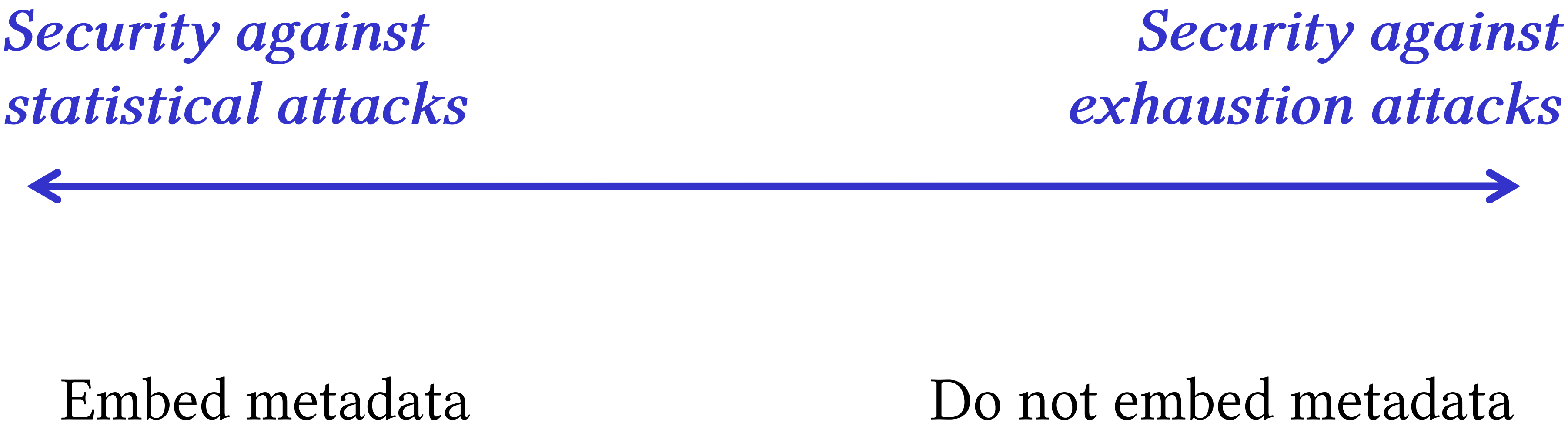
If keys must be re-used, we have to make hard choices:

Embed metadata

Do not embed metadata

Conclusions

If keys must be re-used, we have to make hard choices:



Security against statistical attacks

Security against exhaustion attacks

Embed metadata

Do not embed metadata

Conclusions

If keys must be re-used, we have to make hard choices:

Security against statistical attacks

Security against exhaustion attacks

Embed metadata

Do not embed metadata

Store metadata cryptographically

Do not store metadata cryptographically