Predictable Rain? Steganalysis of Public-Key Steganography using Wet Paper Codes

Matthias Carnein
Dept. of Information Systems
University of Münster
Leonardo-Campus 3
48149 Münster, Germany
matthias.carnein@wwu.de

Pascal Schöttle
Dept. of Information Systems
University of Münster
Leonardo-Campus 3
48149 Münster, Germany
pascal.schoettle@wwu.de

Rainer Böhme
Dept. of Information Systems
University of Münster
Leonardo-Campus 3
48149 Münster, Germany
rainer.boehme@wwu.de

ABSTRACT

Symmetric steganographic communication requires a secret stego-key pre-shared between the communicating parties. Public-key steganography (PKS) overcomes this inconvenience. In this case, the steganographic security is based solely on the underlying asymmetric encryption function. This implies that the embedding positions are either public or hidden by clever coding, for instance using Wet Paper Codes (WPC), but with public code parameters. We show that using WPC with efficient encoding algorithms may leak information which can facilitate an attack. The public parameters allow an attacker to predict among the possible embedding positions the ones most likely used for embedding. This approach is independent of the embedding operation. We demonstrate it for the case of least significant bit (LSB) replacement and present two new variants of Weighted Stego-Image (WS) steganalysis specifically tailored to detect PKS using efficient WPC. Experiments show that our WS variants can detect PKS with higher accuracy than known methods, especially for low embedding rates. The attack is applicable even if a hybrid stegosystem is constructed and public-key cryptography is only used to encapsulate a secret stego-key.

Categories and Subject Descriptors

I.4.9 [**Computing Methodologies**]: Image Processing and Computer Vision – *Applications*; C.2.0 [**General**]: Security and Protection

General Terms

Security, Algorithms, Theory

1. INTRODUCTION

Steganography enables hidden communication between two parties without allowing a third party to notice the hidden communication. To do so, a sender typically embeds hidden messages in a digital medium by slightly modifying parts of the medium. The security of steganography solely depends on the detectability of an embedded message, regardless of whether an attacker is able to read the hidden message. Many steganographic methods require a secret stego-key, shared between the communicating parties. Public-key steganography (PKS) is an approach to allow hidden communication without a shared secret between the sender and the recipient [24]. This is possible by using asymmetric cryptography to encrypt a message before embedding. One way to implement this requires that the positions of the elements used for embedding are publicly known [2], by which the security of this communication only depends on the strength of the cryptographic system [11]. It is often taken for granted that indistinguishability from a random bit sequence is sufficient to achieve steganographic security. Although this is indeed a necessary condition, an attacker can still analyse the local neigbourhood of likely embedding positions to obtain information about the plausibility of the position's value and therefore indications on whether any embedding has taken place.

Even though public embedding positions allow communication without a shared stego-key, they give an attacker a starting point to mount an attack by contrasting the set of elements that might carry a hidden message and those that do not [7, 13]. To prevent this, it is possible to extend PKS by using Wet Paper Codes (WPC) [11, 16]. WPC let steganographic schemes use embedding positions that are not shared with the recipient and are therefore also unknown to an attacker [13]. To apply WPC, both parties have to use the same parameters for embedding and extraction. Since in PKS the communicating parties cannot share a secret stego-key, these code parameters must be public [11].

In this paper, we analyse how public code parameters can be utilized to attack PKS using WPC by identifying more likely embedding positions. We follow an early approach presented by Böhme [4] and calculate the change probability for every element of an object from public parameters. We explore why these probability patterns exist and explain theoretically how they arise. Furthermore we propose an approach that uses this information in order to extend existing steganalysis methods for attacking PKS using WPC.

Our approach is exemplified for hidden messages that are embedded with LSB replacement. We use the information about likely embedding positions by extending Weighted Stego-Image (WS) steganalysis, the state-of-the-art approach to estimate the hidden payload length of messages embedded

IH&MMSec'14, June 11–13, 2014, Salzburg, Austria.

ACM 978-1-4503-2647-6/14/06 ...$15.00.
http://dx.doi.org/10.1145/2600918.2600942.

with random uniform LSB replacement. We propose two extensions of this method to make it applicable for attacking PKS using WPC. First, we use a method presented by Schöttle et. al. [23] and sort the stego object according to its most likely embedding positions. This allows the application of WS steganalysis for initial sequential embedding, originally proposed by Ker [17]. Second, we build on Weighted WS steganalysis and assign a distinct weight to each element of the stego object based on its embedding probability.

Our results show that the calculation of a change probability for every element allows us to identify likely embedding positions if time-efficient encoding algorithms are used. This works exceptionally well in the somewhat artificial situation when a sender does not impose restrictions on possible embedding positions, e.g., based on their predictability, but considers all elements. However, even when restrictions are applied, the approach remains possible. Both proposed extensions of WS steganalysis outperform the current detectors for uniform random LSB replacement when applied to attack PKS using WPC, especially for low embedding rates. We also find that weighting the stego elements gives better results than sorting them.

This paper is organized as follows: Section 2 presents the notation and introduces the relevant concepts of WPC, PKS, and WS steganalysis. Section 3 describes how probability patterns can be generated and proposes two extensions of WS steganalysis tailored to attack PKS using WPC. Section 4 assesses the performance of the proposed estimation of embedding positions and evaluates the detection performance of the proposed extensions. Finally, Section 5 concludes with a summary of the results and discusses their implications.

2. RELATED WORK

2.1 Notation

Matrices and vectors are denoted by boldface symbols. Covers are objects without any embedded messages. Using the notation of [6], a cover is denoted by $\boldsymbol{x}^{(0)}$ using n implicitly to describe its length. After embedding a message $\boldsymbol{m}$ of length q, the resulting stego object is denoted by $\boldsymbol{x}^{(\boldsymbol{m})}$. A stego object with embedding rate $p = q/n$ is denoted by $\boldsymbol{x}^{(p)}$. Symbol $\overline{x}$ denotes an integer value x with the LSB flipped: $\overline{x} = x + (-1)^x$. This notation is also applicable to vectors and matrices, by changing the LSBs of every element. In addition, $\hat{\boldsymbol{x}}^{(0)}$ is the estimated cover, calculated from the stego object $\boldsymbol{x}^{(p)}$, e.g., by estimating the original value of each element by using a weighted average of the adjacent elements. Our approach determines probability patterns by calculating embedding probabilities p_i for every element in a cover $\boldsymbol{x}^{(0)}$. Probability patterns are observable probability characteristics of certain elements when using fixed parameters, in particular irregular distributed embedding probabilities. These patterns can be used to weight the elements of a cover, denoted as ω_i. A stego object sorted from its most probable to least probable embedding position is written as $\boldsymbol{y}^{(p)}$. Even though the described methods are in principle applicable to all digital media, we focus on greyscale images for simplicity.

2.2 Wet Paper Codes

Wet Paper Codes, as first proposed by Fridrich et. al. [15], enable steganographic schemes where a sender is able to select possible embedding position without sharing this selection with the recipient. Their name is a metaphor for a piece of paper that was exposed to rain and got partially wet. A sender is only able to hide a message in dry elements of the paper. During transmission, the wet spots dry out, so that the recipient cannot identify the embedding positions. WPC allow the recipient to read the message without the knowledge of possible embedding positions. A sender can therefore use private selection rules to embed messages. This is supposed to increase the security of steganographic schemes. A detailed description of WPC can be found in [11].

WPC can be seen as a generalization of the selection channel as introduced by Anderson [1]. They embed the message as the syndrome of a linear code [13]. Assuming the sender and recipient share a secret stego-key, they can use this key to generate a shared binary matrix $\boldsymbol{D}$ with q rows and n columns. Furthermore, a public function transforms a cover $\boldsymbol{x}^{(0)}$ or stego image $\boldsymbol{x}^{(\boldsymbol{m})}$ into a binary vector $\boldsymbol{b}^{(0)}$ or $\boldsymbol{b}^{(\boldsymbol{m})}$, respectively. The sender embeds the message into the cover such that the resulting binary vector satisfies:

$$\boldsymbol{D}\boldsymbol{b}^{(\boldsymbol{m})} = \boldsymbol{m}. \tag{1}$$

This matrix product is called the syndrome of $\boldsymbol{b}^{(\boldsymbol{m})}$. To read the message, a recipient calculates the syndrome of $\boldsymbol{b}^{(\boldsymbol{m})}$ using the shared matrix as well as the binary vector of the stego image. To determine which embedding changes are necessary in order to embed a message, a sender can rewrite this equation to (2) by using a change vector $\boldsymbol{v}$, which indicates whether the corresponding element in the cover needs to be changed in order to satisfy Equation (1).

$$\boldsymbol{D}\boldsymbol{v} = \boldsymbol{m} - \boldsymbol{D}\boldsymbol{b}^{(0)} \tag{2}$$

This system of linear equations consists of q equations and n variables. The sender selects k possible embedding positions, the 'dry' pixels, resulting in unknown variables. Note that the recipient does not have to know the value of k. The remaining $n - k$ known variables correspond to the 'wet' pixels that cannot be used to embed a message. By removing all columns from $\boldsymbol{D}$ and all elements from $\boldsymbol{v}$ that correspond to these 'wet' pixels, this equation can be rewritten as

$$\boldsymbol{H}\boldsymbol{v} = \boldsymbol{m} - \boldsymbol{D}\boldsymbol{b}^{(0)}, \tag{3}$$

where $\boldsymbol{H}$ consists of the columns of $\boldsymbol{D}$ corresponding to the changeable pixels in the cover and $\boldsymbol{v}$ is reduced to hold the values of all 'dry' pixels [13].

This system can be solved for $\boldsymbol{v}$, for example, by using Gaussian elimination. Assuming the maximum message length is sent, the complexity of Gaussian elimination is $\mathcal{O}(k^3)$. One approach to reduce the complexity is the structured Gaussian elimination, where the cover is divided into several subsets. The message is then successively embedded into these subsets using Gaussian elimination. A drawback of this approach is that the probability of successfully embedding an entire message is reduced due to the increased number of systems of linear equations, each with a non-zero probability of being singular [13].

An elegant and efficient approach to embed a message using WPC is the matrix LT process [14, 16]. The matrix LT process transforms an under-determined system of linear equations into upper triangle form by permuting its rows and columns. It uses an iterative process, selecting a column with Hamming-weight one in each iteration. This means a column j with only a single value $H_{i,j} = 1$. It then permutes

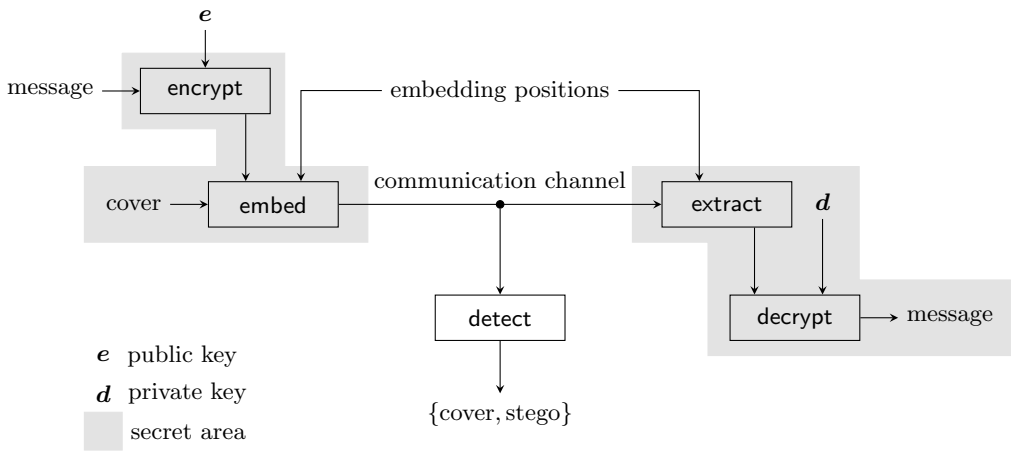


Figure 1: Block diagram of public-key steganography

the columns and rows of the system so that $H_{1,1} = 1$ and $H_{i,1} = 0$ for $i > 1$. In the next iteration the first row and column are ignored, and the rows and columns are permuted such that $H_{2,2} = 1$ and $H_{i,2} = 0$ for $i > 2$ [8]. Given that there will always be a column with Hamming-weight one, this process will transform the system into upper triangle form, which can be efficiently solved using back-substitution [16]. Algorithm 1 gives the pseudo code for the matrix LT process. To increase the probability of a column with Hamming-weight one in every iteration, the columns of the matrix $\boldsymbol{H}$ can be generated so that their Hamming-weights follow the Robust Soliton distribution (RSD)[1] [20]. This is supposed to make the system of linear equations sparsely populated while remaining solvable. The matrix $\boldsymbol{H}$, however, is not directly generated, but derived from the columns of $\boldsymbol{D}$ that correspond to the 'dry' pixels. Both communicating parties therefore generate the shared Matrix $\boldsymbol{D}$ following the RSD. The columns of matrix $\boldsymbol{H}$ then inherit the distribution of the corresponding columns in matrix $\boldsymbol{D}$ [11].

WPC are a first approach that utilizes encoding in connection with steganography. This approach has been further developed, for example, leading to the widely used syndrome trellis codes [10].

2.3 Public-Key Steganography

Steganography uses cryptographic keys to ensure that only authorized parties are able to identify and read hidden messages [6]. Many steganographic methods depend on a shared stego-key between the sender and the recipient. An approach to overcome this restriction is public-key steganography [2].

PKS uses asymmetric encryption in order to avoid a shared secret between the sender and the recipient [11]. To do so, it uses pairs of public and private keys. While the sender holds a private key $\boldsymbol{d}$, there exists a corresponding public key $\boldsymbol{e}$. The public key is used to encrypt a message $\boldsymbol{m}$, i.e., $\boldsymbol{e}(\boldsymbol{m}) = \boldsymbol{m}'$ and the private key is able to decrypt such an encrypted message, i.e., $\boldsymbol{d}(\boldsymbol{m}') = \boldsymbol{m}$ [21].

In PKS, this principle is used to construct a steganographic scheme [11]. Before embedding a message, a sender encrypts the message using the public key of the recipient and embeds the message in publicly available embedding positions [2]. On receipt of a medium the recipient reads the message along the public embedding path and decrypts it using the corresponding private key. This enables the sender to determine whether a hidden message has been embedded into the medium [2]. Figure 1 shows a block diagram of PKS. As long as the encryption function produces uniform i.i.d. ciphertexts, an attacker is not able to distinguish between a random bit-sequence and an encrypted message [21].

Even though an attacker is not able to identify an encrypted message due to the apparently random ciphertexts, the public embedding positions can be used as a starting point to mount an attack on the communication [11]. The attacker can search for traces that indicate the embedding of a bit sequence. These are best exploited when the attacker has knowledge about embedding probabilities of the medium.

Anderson and Petitcolas [2] mention a possible approach for using PKS as key distribution scheme in order to initiate conventional steganographic communication:

> 'the value encrypted under a public key could be a control block consisting of a session key [...], and the session key would drive a conventional steganographic scheme.' [2, p. 480]

This method has its roots in cryptography where it is known as a key encapsulation scheme [9]. The idea is to use asymmetric cryptography only to exchange a symmetric key, which then enables symmetric encryption. In general, a similar approach can be applied in steganography, allowing the exchange of a secret stego-key. This approach reduces the message length sent under public parameters considerably. It must be noted, however, that encapsulating a conventional stego-key of 64–128 bits requires a much larger payload because the key must be encrypted with the public key of the recipient. In a scenario where both parties rely on steganographic communication, we assume a reasonably secure public-key cryptosystem. With the recent revelations, a careful steganographer would be wary of elliptic curve cryptography, which promises shorter minimum message sizes, and consider a system based on RSA with a key size of at least 2048 bits. Its ciphertext is in the order of 2048 bits, independent of the plaintext size. Assuming that the cover image is a greyscale image of size 128×128 pixels, this payload will be $p = 0.13$. Even if the size of the image is increased

[1] See Appendix A for an explanation of the RSD.

to 512×512 pixels, the payload of approximately $p = 0.078$ lies in a range where modern detectors may catch it.

An approach to avoid attacks on PKS is to use, quoting the authoritative textbook,

> 'selection channels that are completely random implemented using wet paper codes.' [11, p. 184]

This would allow a sender to embed the message in secret embedding positions so that an attacker cannot use this information. Since there is no shared secret between the sender and the recipient in PKS, both cannot generate the matrix $\boldsymbol{D}$ based on a shared key. To allow the use of WPC, this matrix (or its seed) can be made public [11, p. 184]. This allows every party to extract the encrypted message, but without revealing the exact embedding positions. Only the owner of the private key is able to decrypt and read the message [11]. Even though the author does not specifically propose the matrix LT process in order to embed messages in PKS, it is by far the most efficient approach when embedding with WPC, and the only improvement to Gaussian elimination that retains a sufficient probability of solving the system of linear equations. This extension of PKS is supposed to enable steganographic communication, now quoting original research work,

> 'without revealing any information about the placement of the embedding changes.' [16, p. 216]

Our contribution challenges this view.

2.4 Weighted Stego-Image Steganalysis

Weighted Stego-Image steganalysis, as presented by Fridrich and Goljan [12], is a quantitative steganalytic method targeting LSB replacement. It uses a so called Weighted Stego-Image $\boldsymbol{x}^{(p,\lambda)}$, which is a weighted average between the stego image and the stego image with every element's LSB flipped. Following [18] and [6], we define $\boldsymbol{x}^{(p,\lambda)}$ as:

$$\boldsymbol{x}^{(p,\lambda)} = \lambda \overline{\boldsymbol{x}}^{(p)} + (1-\lambda)\, \boldsymbol{x}^{(p)}, \tag{4}$$

where λ describes the weighting and $\overline{\boldsymbol{x}}^{(p)}$ denotes the stego image with every element's LSB flipped.

Theorem 1 in [12] states that the Euclidean distance between $\boldsymbol{x}^{(p,\lambda)}$ and $\boldsymbol{x}^{(0)}$ is minimized for $\lambda = q/(2n)$. As the cover is unknown to an attacker, she has to estimate it from the stego image. This can be achieved by using a linear filter, i.e., a weighted average of the local neighborhood. An example for such a filter is presented by Ker and Böhme using a filter of the form (5). The performance of these filters has mainly been evaluated experimentally and needs further research [6, 18].

$$\begin{matrix} -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \end{matrix} \tag{5}$$

We can use twice the Euclidean distance as an estimator for the embedding rate, using Equation (6). By differentiating the Euclidean distance for λ, we can estimate p using Equation (7) [12].

$$\hat{p} = 2 \arg\min_{\lambda} \sum_{i=1}^{n} \left(x_i^{(p,\lambda)} - \hat{x}_i^{(0)} \right)^2 \tag{6}$$

$$= \frac{2}{n} \sum_{i=1}^{n} \left(x_i^{(p)} - \hat{x}_i^{(0)} \right) \left(x_i^{(p)} - \overline{x}_i^{(p)} \right) \tag{7}$$

Several approaches to improve this method have been made. Most notably is Weighted WS steganalysis which adds element weights to the stego image [12]. This second weighting takes differences in local predictability into account. Elements which can be estimated with high confidence contribute more to the estimation than elements where errors are expected:

$$\hat{p} = 2 \sum_{i=1}^{n} w_i \left(x_i^{(p)} - \hat{x}_i^{(0)} \right) \left(x_i^{(p)} - \overline{x}_i^{(p)} \right), \tag{8}$$

using $\sum_{i=1}^{n} w_i = 1$ [12].

Both basic methods assume that changes in the cover are spread uniformly [18]. Ker [17] proposes a variant tailored to initial sequential embedding. He decomposes Equation (6) into two parts, reflecting that embedding changes only occur in the first elements. The first elements are therefore weighted using $\lambda = 1/2$ while the remaining elements are weighted with $\lambda = 0$. Note that in this scenario any change to the weighting, e.g., based on local predictability, will degrade the estimation. This is the case because it is certain that the first elements contain the hidden message. The resulting estimator (Eq. (9)) is minimized for the point where the embedding ends, i.e., $l = q$ [17].

$$E(l) = \sum_{i=1}^{l} \left(\frac{1}{2} \left(x_i^{(p)} + \overline{x}_i^{(p)} \right) - \hat{x}_i^{(0)} \right)^2 + \sum_{i=l+1}^{n} \left(x_i^{(p)} - \hat{x}_i^{(0)} \right)^2 \tag{9}$$

This approach outperforms the previously introduced detectors when applied to initial sequential embedding [17]. However, it is not possible to obtain a closed form by differentiating Equation (9). To overcome this, Ker proposes the recursive function in Equation (10).

$$\begin{aligned} e_0 &= 0 \\ e_l &= e_{l-1} + \left(\frac{1}{2} \left(x_{l-1}^{(p)} + \overline{x}_{l-1}^{(p)} \right) - \hat{x}_{l-1}^{(0)} \right)^2 \\ &\quad - \left(x_{l-1}^{(p)} - \hat{x}_{l-1}^{(0)} \right)^2 \end{aligned} \tag{10}$$

This is possible because Equation (10) satisfies $E(l) = e_l + \sum_{i=1}^{n} (x_i^{(p)} - \hat{x}_i^{(0)})^2$. Since the last term is constant it can be ignored to find the minimum of Equation (10).

The approach presented in [23] utilizes this method to make WS steganalysis applicable for naive adaptive embedding, where the sender embeds a message only in the q elements she considers to be most secure [22]. By reordering the stego image according to the criterion used for locating these elements, this kind of embedding reduces to initial sequential.

There exist further specializations of WS steganalysis, for example with bias correction [6, 18] or tailored to JPEG covers [5]. To the best of our knowledge none of these extensions take information about the encoding process into

account to create an improved detector. This is the starting point for our approach which considers information that can be derived from the encoding process of WPC.

3. CONTRIBUTION

In this section we discuss how the public parameters in PKS can be utilized to mount an attack on PKS using WPC. We first present a general approach on how to estimate likely embedding positions by estimating probability patterns, following the early ideas presented in [4]. We further explore why these patterns exist and present a theoretical explanation on how they arise. Then we explain how to use this information in order to attack PKS using WPC. To do so, we extend WS steganalysis and make it applicable for WPC with public parameters. In a first approach we build on a method presented in [23]. In addition, we present a new attack that enables us to utilize the embedding patterns to attack WPC in PKS.

3.1 Estimating embedding positions

To use WPC without a shared key, the sender and recipient are required to use the same parameters to generate the matrix $\boldsymbol{D}$ [11]. The public matrix, however, determines a large part of the system of linear equations that needs to be solved in order to make the necessary embedding changes [13]. This knowledge gives an attacker information about the embedding process and can be used as a starting point to mount an attack. A first approach to utilize this information is motivated in [4] by using the public parameters to determine probable embedding positions. To use the metaphor of a piece of paper exposed to rain [13], an attacker does not predict where the rain hit the paper but, which of the 'dry' positions were most likely used for embedding.

To estimate the embedding positions, an attacker may embed multiple messages using the same public parameters. By observing the necessary embedding changes in the medium, it is possible to calculate an embedding probability for every element in the cover. This approach is enabled due to the public matrix $\boldsymbol{D}$ as well as the information that can be derived from its dimensions. While the number of columns in $\boldsymbol{D}$ indicates the length n of the cover, the number of rows corresponds to the message length q. The message and the cover are secret and unknown to an attacker. Embedding probabilities therefore have to be calculated using random messages and random covers. This means an attacker can generate random messages of length q and embed them into random covers of length n, using the public matrix.

3.1.1 Without the Use of a Selection Rule

First we discuss the generating process of these embedding probabilities if the sender does not use a selection rule and thus, all elements can be used for embedding. To determine the embedding changes, the system of linear equations (3) must be solved. As discussed in Section 2.2, an elegant and efficient approach to solve the system is the matrix LT process [11, 16]. Preliminary results reported in [4] indicate that for a fixed matrix $\boldsymbol{D}$, probability patterns can be observed after embedding multiple messages. This means that certain elements of the cover are more likely used for embedding than others.

Without the use of a (random or adaptive) selection rule, the embedding probabilities can be determined with almost perfect accuracy. The reason for this can be seen from looking at the pseudo code for the matrix LT process in Algorithm 1. The algorithm is perfectly deterministic and thus, with the same input matrix $\boldsymbol{D}$ the same output, i.e., the same embedding positions will be produced.

To understand why these probability patterns exist, one has to understand the selection of pivot elements during the solving process. The solution of the system of linear equations indicates the necessary embedding changes in the cover. Only elements that correspond to columns that are used during the solving process are therefore subject to change and might carry a hidden message bit. When using the matrix LT process to solve the system of linear equations, the pivot columns are selected based on their column sum. This selection is biased towards sparsely populated columns (lines 3–5 in Algorithm 1).

Algorithm 1: Matrix LT process to solve $\boldsymbol{Hv} = \boldsymbol{z}$ [11]

```
1  j = 1 and t = 0;
2  while j ≤ q & (∃j' ≥ j, Σ_{i>j} H[i, j'] = 1) do
3  |   swap rows j and i_j;
4  |   swap z[i] and z[i_j];
5  |   swap columns v[j] and v[j'];
6  |   t = t + 1;
7  |   τ[t];
8  |   j = j + 1;
9  if j ≤ q then
10 |   failure;
11 v[j] = 0 for q < j ≤ k;
   // back substitution
12 while t > 0 do
13 |   v ← τ[t](v);
14 |   t = t − 1;
```

The consequence of this can be best understood when using a simple example and looking at the relationship between the embedding probabilities and the matrix $\boldsymbol{D}$. Figure 2 shows the embedding process of 1000 random messages of length $q = 2$ in random covers of length $n = 5$. Figure 2(a) shows an example of matrix $\boldsymbol{D}$ where filled rectangles denote ones and empty rectangles zeros. The histogram in Figure 2(b) shows the observed change probability for every element. It becomes obvious that only the elements with index two and four are changed in order to embed a message. This indicates that the matrix LT process is biased towards sparsely populated columns. In this example, the fourth column is always selected as the first pivot column because it is the first and only column with column sum one. During the matrix LT process, the first and fourth column are therefore swapped. The process continues by neglecting the first column and row, searching another column with a Hamming-weight of one. Since this applies to all the remaining columns, the first one is selected in order to solve the system. This is the second column in the example. After this, the message is embedded and no further changes are required. This example shows that for a fixed matrix $\boldsymbol{D}$, the selection of columns during the matrix LT process happens in a deterministic order. As no selection rule is used, the elements corresponding to these columns are always used to carry a message bit. These elements can therefore be exactly identified by an attacker.

3.1.2 With the Use of a Selection Rule

Using a selection rule reduces this effect because only selected elements may be changed. The attacker still has to solve the system of linear equations in (3). But now, she has to use random selection rules and thus, the input to Algorithm 1 changes with every simulated embedding. This leads to different embedding positions in every iteration, as in each iteration different elements may not be used. The selection during the solving process seems still biased towards sparsely populated columns, though. This enables an attacker to express a change probability for every element.

Instead of the random selection rules suggested in [11], a sender could also use an adaptive selection rule and thus select embedding positions that are supposedly harder to detect. We conjecture that if this adaptive selection rule is publicly known, an attacker could use this knowledge to even better estimate the most likely embedding positions.

3.1.3 Gaussian Elimination

The Gaussian elimination is not biased towards sparsely populated columns. The process instead focusses on the first columns of the matrix. Without using a selection rule, most of the first columns of the matrix are used to solve the system. Only the corresponding elements may therefore contain the hidden message. When using a selection rule, this pattern again diminishes. Assuming that this selection happens randomly, the embedding probability for all used elements is approximately the same. Since the embedding probabilities for all elements are reduced equally, this does not allow the generation of embedding patterns.

A third approach to embed a message using WPC is the structured Gaussian elimination as presented in [15]. For this approach the cover is split into subsets. In each of the subsets a part of the message is embedded using Gaussian elimination. As expected, each subset shows a similar pattern as the Gaussian elimination. Since most elements hold the same probability when using Gaussian elimination, they hold similar probabilities when repeating the process for several subsets. Structured Gaussian elimination uses several elements per subset as a buffer to increase the probability that a message can be successfully embedded. These buffer elements hold slightly lower embedding probabilities. Since the remaining elements have similar embedding probabilities, this knowledge can hardly be utilized for an attack.

These results show that PKS using WPC can be attacked when the matrix LT process is used. This is the case because certain columns of the public matrix are selected more likely to solve the system of linear equations. When using Gaussian elimination, columns are uniformly selected, leading to equal change probabilities. The following section presents an approach on how to use these patterns by extending an already existing attack on LSB replacement to make it applicable for PKS using WPC.

3.2 Extending Weighted Stego-Image Steganalysis

As shown in the previous section, the proposed estimation of embedding positions only depends on the publicly available matrix. It is therefore independent from the used embedding method. This enables an attacker to use the estimation in order to extend already existing steganographic attacks on embedding methods and make them applicable for PKS using WPC. This section proposes an extension of WS steganalysis to attack PKS using WPC when LSB replacement is used to embed a message. Although LSB replacement is clearly insecure and should not be used in practice, it is a good candidate to demonstrate our attack because it is best understood and reliable steganalysis benchmarks are readily available.

In the following we propose two extensions of WS steganalysis: The first one builds on the approach of WS steganalysis for naive adaptive embedding as proposed in [23]. The second approach uses the element weights in Weighted WS steganalysis. We call these two WS variants Sorted WS and Probability WS, respectively.

3.2.1 Sorting the Stego Image

The first possibility we discuss to extend WS steganalysis for PKS using WPC is to sort the stego image according to its most probable embedding positions. To determine the embedding probability for each element, we use the method presented in Section 3.1. We then sort the elements of the stego image $\boldsymbol{x}^{(p)}$ according to theses probabilities in descending order to estimate the ordered stego image $\hat{\boldsymbol{y}}^{(p)}$. Assuming the attacker is able to identify each embedding position correctly, the first $p \cdot n$ elements of $\boldsymbol{y}^{(p)} = (y_1^{(1)}, \ldots, y_{p \cdot n}^{(1)}, y_{p \cdot n+1}^{(0)}, \ldots, y_n^{(0)})$ contain the hidden message [23]. This will, again, reduce the embedding to initial sequential embedding, allowing the use of the specialized WS steganalysis approach as proposed in [17].

For a correctly ordered image, all elements containing a hidden message are placed at the beginning. Only the first elements of the ordered image are therefore weighted using $\lambda = 1/2$. Thus, we get the estimator corresponding to Equation (9) as:

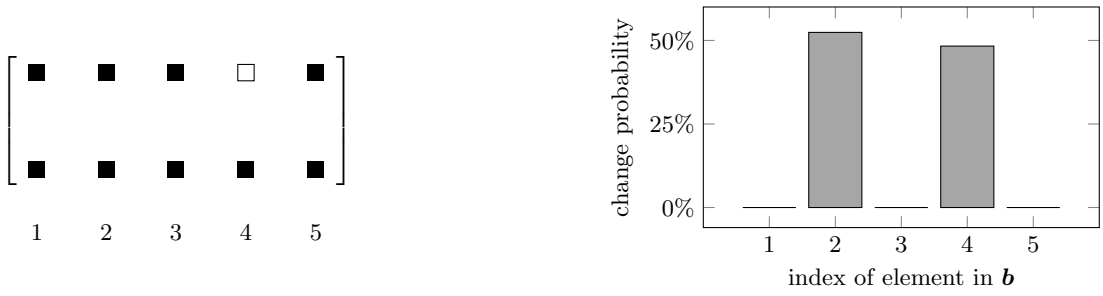
$$E(l) = \sum_{i=1}^{l} \left(\frac{1}{2} \left(y_i^{(p)} + \overline{y}_i^{(p)} \right) - \hat{y}_i^{(0)} \right)^2 + \sum_{i=l+1}^{n} \left(y_i^{(p)} - \hat{y}_i^{(0)} \right)^2 . \quad (11)$$

As described in Section 2.4, this determines the point where embedding ends and therefore the payload length [17]. Again, we use the recursive function (Eq. (10)) of Ker to reduce the complexity. As we work with a sorted stego image here, we will refer to this method as *Sorted WS* in the following.

As discussed in Section 3.1, it is only possible to determine exact embedding positions when no selection rule is used. Considering that WPC allow secret selection rules, it is likely that a sender will use this possibility. When using such a constraint on possible embedding positions, it is only possible to calculate embedding probabilities. When sorting the stego image according to these probabilities, it is therefore likely that not all embedding positions are correctly ordered. This leaves 'gaps' in the first $p \cdot n$ elements, i.e., elements that do not contain a hidden message. Since WS steganalysis for sequential embedding expects a continuous payload at the beginning of an image, it handles these 'gaps' poorly and tends to underestimate the payload length [23].

3.2.2 Weighting the Stego Image

Our second approach to make WS steganalysis applicable for PKS using WPC is to extend Weighted WS steganalysis. Weighted WS steganalysis introduces element weights to the stego image in order to respect different levels of uncertainty



(a) Visualization of matrix $\boldsymbol{D}$

(b) Change probability

Figure 2: Visualization of connection between matrix LT process and observed probability patterns

when estimating the original elements in the cover [6]. The weighting w_i describes the predictability of every element. This weighting can be used to utilize the information about likely embedding positions.

To estimate the embedding rate we use the same approach as Weighted WS steganalysis. As outlined in Section 2.4, this enables us to estimate half the embedding rate. This estimation holds true when using a normalized weighting for every element [12]. We can therefore use this weighting to incorporate the embedding probabilities. In a first step we calculate the embedding probability p_i for every element using the approach presented in Section 3.1. This allows to replace the weighting w_i of Weighted WS steganalysis with a normalized weighting ω_i, based on the change probability. This focuses on more likely embedding positions and therefore relevant parts of the stego image. Using this approach, the embedding rate can be estimated by using Equation (12). By differentiating for λ, we can develop a first estimator for the embedding rate using Equation (13) and the normalized weighting $\omega_i = p_i / \left(\sum_{i=1}^{n} p_i\right)$.

$$\hat{p}' = 2 \arg\min_{\lambda} \sum_{i=1}^{n} \omega_i \left(x_i^{(p,\lambda)} - \hat{x}_i^{(0)} \right)^2 \tag{12}$$

$$= 2 \sum_{i=1}^{n} \omega_i \left(x_i^{(p)} - \hat{x}_i^{(0)} \right) \left(x_i^{(p)} - \overline{x}_i^{(p)} \right), \tag{13}$$

However, it has to be considered that the embedding rate is only estimated for elements with a positive weight. Elements with an assigned weighting of zero, i.e., elements that are never changed during the embedding process are not considered. This leads to an over-estimation, since only elements are considered that are likely to contain a hidden message. To estimate the embedding rate for the entire object, one can translate the estimation using:

$$\hat{p} = \frac{\hat{p}' \cdot |\boldsymbol{\omega}'|}{|\boldsymbol{\omega}|}, \tag{14}$$

with $\boldsymbol{\omega}' = \{\omega_i \in \boldsymbol{\omega} | \omega_i > 0\}$ and $\boldsymbol{\omega} = \{\omega_1, \ldots, \omega_n\}$. As we replace the weights of Weighted WS with the change probabilities, we will refer to this method as *Probability WS* in the following.

A drawback of this approach is that we replace the weighting based on the local predictability with our calculated embedding probabilities. We therefore neglect the knowledge about local predictability. To join both approaches we could combine both weightings, assigning a higher weighting to elements that can be predicted with high confidence and that are likely to carry a hidden message. Although this approach works well for messages that are embedded uniformly over all elements, it collapses when the sender embeds the message adaptively into elements that are harder to predict. Adaptive embedding, however, is one of the main reasons to use WPC. We therefore do not further evaluate this approach, but focus on a single (additional) weighting based on the change probabilities.

4. EXPERIMENTAL RESULTS

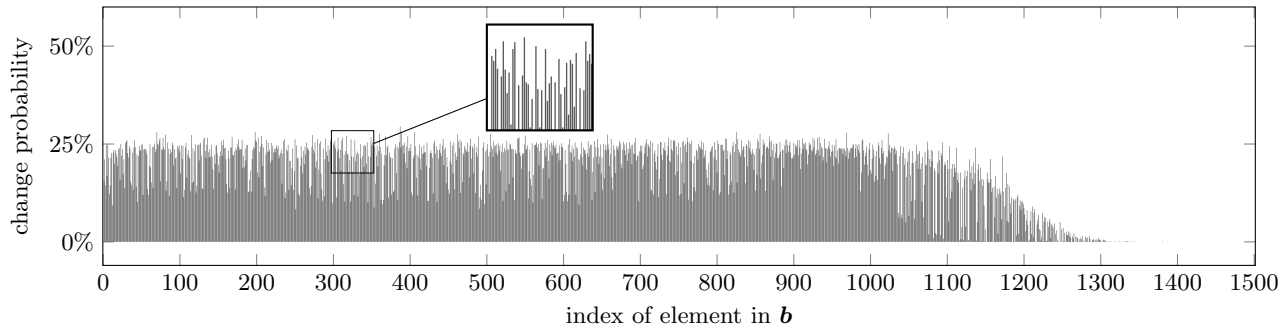
This section evaluates the performance of our proposed methods. We first investigate how accurately we can estimate the embedding positions. We then apply this knowledge to attack PKS using WPC when using LSB replacement as proposed in Section 3.2. We use the full BOSSBase [3] image database with 10 000 grayscale images and embed random messages of different lengths using LSB replacement[2]. To accelerate the calculation of embedding patterns, we crop the images to a size of 128×128 pixels. Steganalysis is generally more difficult in smaller images due to the square root law [19], but the relative performance differentials are unlikely to be affected by smaller test images. Furthermore we use the matrix LT process to solve the system of linear equations. Matrix $\boldsymbol{D}$ is generated from a random key using $c = 0.1$ and $\delta = 0.4$ as parameters for the RSD.

4.1 Estimating Embedding Positions

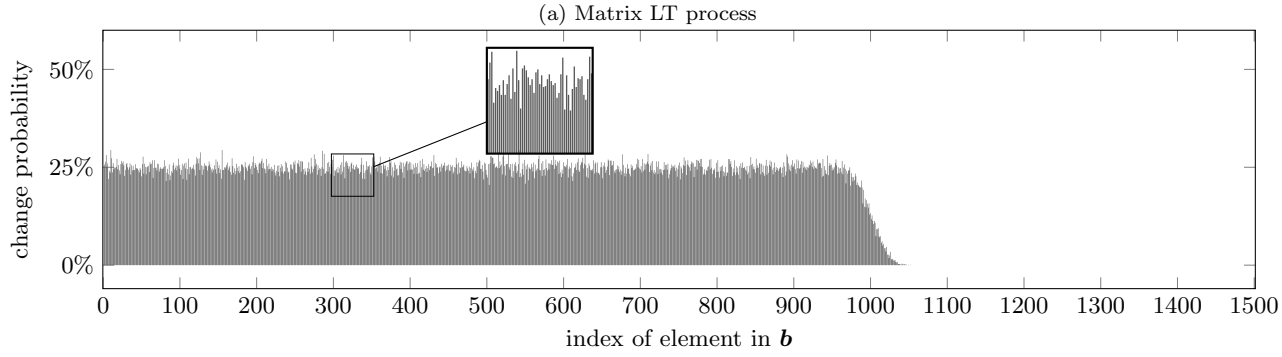
We analyse how accurately embedding positions can be estimated by calculating a change probability for every element. To do so, we embed 1000 random messages of length $q = 500$ in random covers of length $n = 1500$ without using any 'wet' pixels and observe the embedding changes. The results show that the change probability of exactly q elements is approximately 50% while the remaining elements are never used to embed a message. This shows that for each embedding process the very same elements are used to hold a message. On average the elements are changed for every second embedding, since in every second case an element already holds the desired value.

WPC enable using a secret selection rule. It is therefore likely that a sender will use such a selection rule to select possible embedding positions. Repeating the experiment, using a selection rule which randomly selects 50% 'wet' pixels, it becomes obvious that the probability patterns diminish, but remain visible. In our example, a quarter of the elements

[2]Note that we exclude the image borders from embedding and detection attempts to eliminate boundary conditions.



(a) Matrix LT process



(b) Gaussian elimination

Figure 3: Change probabilities for a fixed matrix $\boldsymbol{D}$, using 50% random 'wet' pixels, generated by 1000 embedding iterations

are never used to embed a message. The remaining elements hold a distinct change probability, which indicates how often each element is used during the embedding process. Figure 3(a) shows the observed change probabilities when using the matrix LT process and a random selection rule. The magnification of some of the probability peaks illustrates inconsistent change probabilities. The results show that we cannot determine the exact embedding positions, but are able to calculate a probability for every element to identify more likely embedding positions in the cover. These results show that even though WPC allow a secret selection rule, the attacker can use the public parameters of PKS to estimate where embedding changes are more likely to happen. This gives him a starting point to mount an attack on PKS using WPC.

While the matrix LT process is a very elegant and efficient method for solving the system of linear equations, other approaches exist, e.g., Gaussian elimination. To analyse the probability patterns when using Gaussian elimination, we repeat the experiments using a matrix $\boldsymbol{D}$ with equal probability for ones and zeros and solving the system of linear equation using Gaussian elimination. The results show that the first elements of the cover are always used to hold the hidden message when no selection rule is applied. Similar to the matrix LT process, an attacker is able to exactly identify the embedding positions. However, when using a selection rule, all elements that are used to embed a message are used equally likely. Figure 3(b) shows the observed change probability when using Gaussian elimination and a selection rule that randomly selects 50% 'wet' pixels. When looking at the magnified probability peaks it shows more consistent change probabilities across all elements. The results show that an attacker is not able to identify more likely embedding positions. This supports the suggestions made in Section 3.1 that these patterns arise in an effort to solve the system more efficiently.

4.2 Estimating the Embedding Rate

This section examines the accuracy of the proposed specializations of WS steganalysis. We first assume that an attacker has knowledge about the number of possible embedding positions k by using a fixed amount of $k = 50\%$ 'dry' pixels. Figure 5(a) shows the mean absolute error (MAE), $|p - \hat{p}|$, of the estimated embedding rate for 10000 stego images for different embedding rates p. It has to be considered that the payload estimation for covers depends on the public embedding rate and therefore slightly varies depending on the corresponding probability patterns. To evaluate the performance for covers we therefore calculate the average of all considered embedding rates as the estimation. It is visible that weighting the stego image by embedding probabilities clearly outperforms the current detectors for all evaluated embedding rates. Ordering the stego image according to these probabilities works well for small embedding rates, but quickly loses accuracy when using larger embedding rates. This observation is similar to the results of [23].

In practice, an attacker has no information about the number of possible embedding positions. Since this information is crucial for the calculation of probability patterns, we need to estimate it. We propose to estimate the embedding rate by attacking every stego image multiple times under different assumptions about k. In a first approach we used the maximum estimation of all attacks. However, we found that a better estimation can be made when using the point where both proposed methods estimate a similar embedding rate. Figure 4(a) shows the estimation of the embedding rate of

$p = 0.1$ with 50% 'dry' pixels under different assumptions about k. It shows that the most accurate estimation is the point where both approaches estimate a similar embedding rate, even if the assumption about k is incorrect. We therefore exploit this observation and stick to this tactic in the following. Figure 4(b) shows the influence of this estimation on the performance of the detectors. It can be seen that the reduced knowledge of the attacker has very little influence on the detection performance and leads to similar mean absolute errors. In many cases the estimation of the amount of 'dry' pixels even leads to slightly better results. This can be explained by Figure 4(a), as the estimation of the embedding rate is better where the two methods coincide, than it is for the assumption of the real amount of 'dry' pixels.

Using this knowledge we can estimate the embedding rate by finding the point where the distance between Sorted WS and Probability WS is minimal. We denote these assumptions over k as $\boldsymbol{k}$. In our experiments we attack the stego images using the assumptions $\boldsymbol{k} = \{10\%, 20\%, \ldots, 100\%\}$. To retain solvability, we only use assumptions greater than the actual embedding rate. The used embedding rate is public due to the size of matrix $\boldsymbol{D}$. Figure 5(b) shows the MAE of the same experiment with an unknown, random amount of 'dry' pixels. It can be seen that all methods still perform well for small embedding rates, but lose accuracy for higher embedding rates. For larger embedding rates, Sorted WS benefits from using the minimal distance between both methods as it reduces the estimation error. The good estimation performance for smaller payloads suggests that the attack is also applicable when a hybrid stegosystem is constructed and PKS is only used for a key encapsulation, as described in Section 2.3.

These results show a good estimation of the embedding rate for our specialized methods. However, as already discussed, the payload length is public due to the size of the public matrix. It is therefore more important to identify stego images reliably. To evaluate the performance of such detectors, we use a recipient operating characteristics (ROC) curve. A ROC curve compares the false positive rate to the detection rate for a varying threshold [6]. Figure 6(a) and Figure 6(b) show the empirical ROC curves, again for $k = 50\%$ and for an unknown amount of 'dry' pixels, respectively. In both cases our specialized methods clearly outperform the existing unspecialized methods and allow a more reliable detection of stego images.

When comparing different estimators, it is helpful to use a single value that expresses their steganalytic performance. An example for such a metric is the equal error rate (EER). It expresses the error rate for the point where the probability for a missed detection equals the false positive rate. A smaller value indicates a better detector. Table 1 shows the EERs for our experiment. The result underlines the good performance for our estimators for various embedding rates. For almost all evaluated embedding rates our detectors lead to a better performance. For large embedding rates, this advantage slowly diminishes.

5. DISCUSSION AND CONCLUSION

In this paper we assume that an attacker makes use of all knowledge available to her and thus tries to predict which of the 'dry' pixels have more likely been modified during embedding with WPC. More specifically, we develop an approach to attack PKS using WPC by determining probability patterns that can be calculated from public parameters. We investigate, formalise and quantify an early approach proposed in [4] and observe the change probability for every element when using fixed parameters. We are able to use this approach to calculate embedding probabilities for every element in a stego image based on these public parameters. We investigate why these probability patterns arise and state that the proposed calculation of embedding probabilities is independent of the embedding method. We are therefore able to use the information about more likely embedding positions to extend known attacks on embedding methods to make them applicable for PKS using WPC. This possibility is illustrated by extending WS steganalysis to attack the use of LSB replacement in PKS with WPC. More specifically, we propose two extension of WS Steganalysis. The first approach sorts the elements of the suspect image according to their embedding probabilities. This places likely embedding positions at the beginning, which allows to run WS steganalysis for initial sequential embedding. We further propose to use the knowledge of embedding probabilities as element weights in Weighted WS steganalysis.

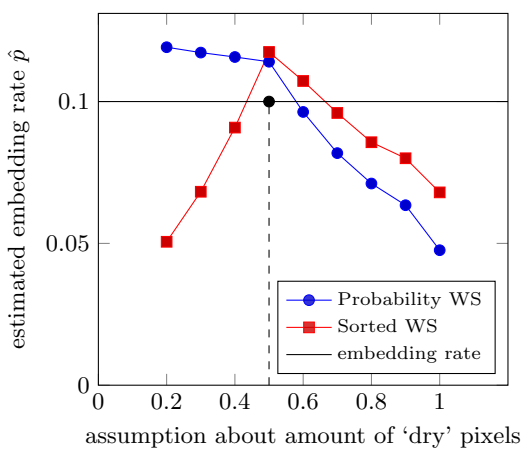
Our results indicate that the probability patterns arise when using the matrix LT process in an attempt to solve a system of linear equations more efficiently than with Gaussian elimination. Solving this system determines the necessary embedding changes when using WPC. We show that an attacker is able to identify the exact embedding positions when no selection rule is used. While these patterns diminish when using a selection rule, it is still possible to calculate an embedding probability for each element. This is in contrast to the assumption that the use of WPC does not reveal any information about the placement of embedding changes. These patterns do not occur when using other algorithms to solve the system, e.g., when using the computationally more complex Gaussian elimination.

We investigate our specialised WS detector on a large dataset. The results show that both proposed methods outperform current unspecialized methods when using a fixed number of possible embedding positions. Both methods show better results in estimating the embedding rate as well as better classification of stego objects and covers. Since the number of possible embedding positions is typically unknown, we further present an approach to estimate the embedding rate without any further knowledge about the amount of embedding positions. Our experiments show that the point where both proposed methods estimate a similar embedding rate is a good estimator for the actual embedding rate. We therefore estimate the embedding rate under different assumptions about the amount of 'dry' pixels. We then exploit the fact that both proposed methods' estimates converges to each other and use the point where the distance between the estimated embedding rates is minimal. Interestingly, this trick gives an even more accurate estimator than if we have the real amount of 'dry' pixels available as side-information.

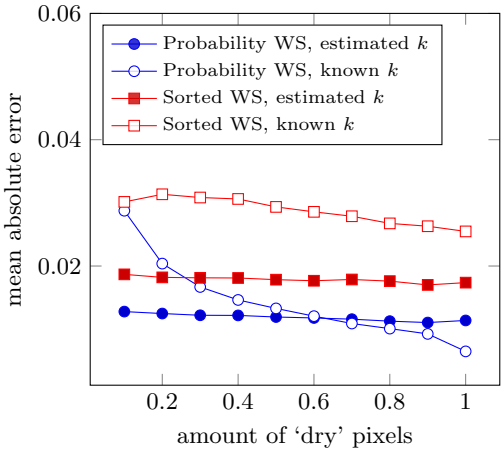
Although the proposed approach is specifically targeted at the use of WPC, the main weakness arises from its public parameters and the efficient solver. It is likely that further steganographic coding schemes are also vulnerable when used in PKS. Future research may therefore investigate the implications of the proposed attack on other steganographic schemes as well as efficient countermeasures. We do not rule out the possibility that a similar approach may be used to attack the widely used syndrome trellis codes [10]. While

p	Probability WS	Sorted WS	Unweighted WS	Weighted WS
0.005	0.37	0.38	0.48	0.47
0.010	0.34	0.36	0.46	0.44
0.025	0.29	0.31	0.41	0.37
0.050	0.24	0.27	0.33	0.27
0.100	0.17	0.21	0.23	0.15
0.150	0.14	0.17	0.16	0.10
0.200	0.10	0.14	0.12	0.07

Table 1: EER of the ROC curves of 10000 covers and 10000 stego images, embedding rate p and estimated amount of 'dry' pixels

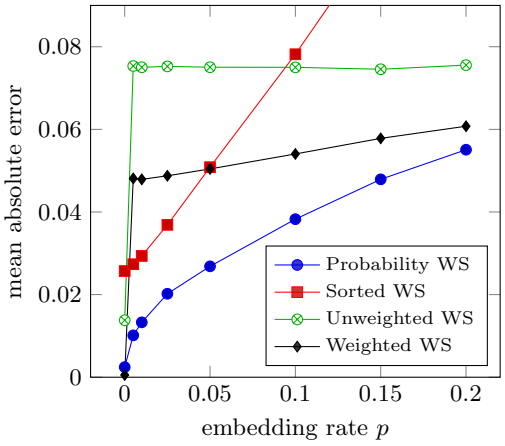


(a) Assumptions about k, with actual $k = 50\%$ and $p = 0.1$

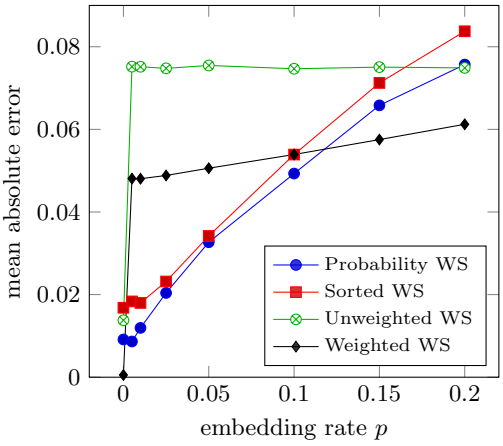


(b) Attacks with and without knowledge of k, $p = 0.01$

Figure 4: Comparison of the performance of Probability WS and Sorted WS

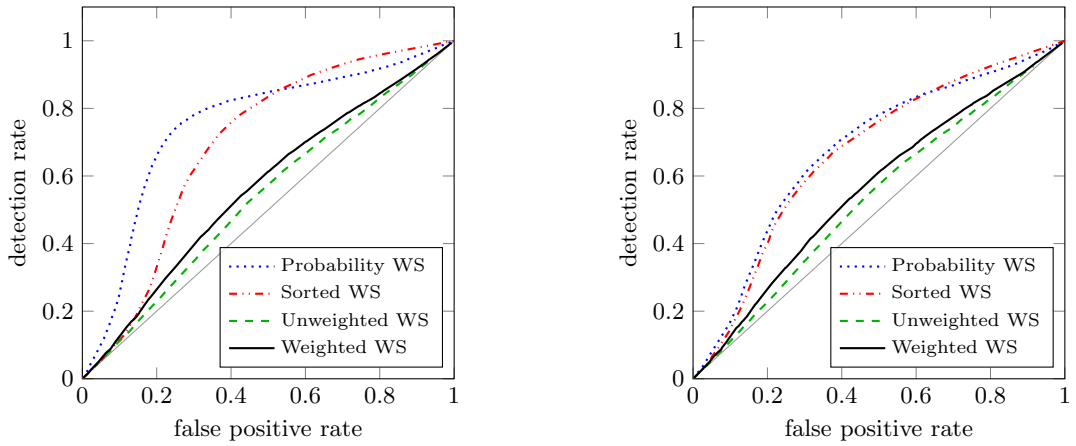


(a) Using 50% 'dry' pixels



(b) Unknown amount of 'dry' pixels

Figure 5: Mean absolute error for the estimated embedding rate $\hat{p}$



(a) Using 50% 'dry' pixels

(b) Unknown amount of 'dry' pixels

Figure 6: Empirical ROC curves for $p = 0.01$

LSB replacement is (hopefully) rarely used in practice, the method is well understood and serves as a starting point. Our work illustrates the possibility to extend known targeted attacks to make them more accurate against PKS using WPC. The generalization to other embedding operations and feature-based detectors is left for future work. Other open research directions are the possibility to extract the embedding probabilities by theoretical means instead of numerical simulation; to combine the estimation of embedding probabilities with knowledge about a selection rule; and finally, to find an approach to solve the PKS problem efficiently without leaking compromising patterns to the steganalyst.

Closing with the metaphor of Wet Paper Codes: although the presented method is not able to predict the rain, it can predict the most likely *hideouts in the storm*. The results show that for certain combinations of building blocks, the prediction of these hideouts can be sufficient information to significantly reduce the security of PKS using WPC.

Acknowledgments

This work has been partially supported by a grant of Deutsche Forschungsgemeinschaft (DFG). The quantitative experiments were carried out on the University of Münster's central high-performance computing cluster Palma.

6. REFERENCES

[1] R. J. Anderson. Stretching the limits of steganography. In R. J. Anderson, editor, *Information Hiding (1st International Workshop)*, volume 1174 of *Lecture Notes in Computer Science*, pages 39–48. Springer-Verlag, Berlin Heidelberg, 1996.

[2] R. J. Anderson and F. A. P. Petitcolas. On the limits of steganography. *IEEE Journal on Selected Areas in Communications*, 16(4):474–481, 1998.

[3] P. Bas, T. Filler, and T. Pevný. Break our steganographic system — the ins and outs of organizing BOSS. In T. Filler, T. Pevný, S. Craver, and A. Ker, editors, *Information Hiding (13th International Workshop)*, volume 6958 of *Lecture Notes in Computer Science*, pages 59–70, Berlin Heidelberg, 2011. Springer-Verlag.

[4] R. Böhme. Wet paper codes for public key steganography? Unpublished rump session talk at the 7th Information Hiding workshop, Presentation slides available at `http://www1.inf.tu-dresden.de/~rb21/publications/Boehme2005_IHW_RumpSession.pdf` (last accessed: April 2014), 2005.

[5] R. Böhme. Weighted stego-image steganalysis for JPEG covers. In K. Solanki, editor, *Information Hiding (10th International Workshop)*, volume 5284 of *Lecture Notes in Computer Science*, pages 178–194, Berlin Heidelberg, 2008. Springer-Verlag.

[6] R. Böhme. *Advanced Statistical Steganalysis*. Springer-Verlag, Berlin Heidelberg, 1st edition, 2010.

[7] R. Böhme and A. Westfeld. Exploiting preserved statistics for steganalysis. In J. Fridrich, editor, *Information Hiding (6th International Workshop)*, volume 3200 of *Lecture Notes in Computer Science*, pages 82–96. Springer-Verlag, Berlin Heidelberg, 2004.

[8] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker. *Digital Watermarking and Steganography*. Morgan Kaufmann, Burlington, MA, 2nd edition, 2007.

[9] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, Berlin Heidelberg, 1998. Springer-Verlag.

[10] T. Filler, J. Judas, and J. Fridrich. Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, 6(3):920–935, 2011.

[11] J. Fridrich. *Steganography in Digital Media. Principles, Algorithms, and Applications*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[12] J. Fridrich and M. Goljan. On estimation of secret message length in LSB steganography in spatial domain. In E. J. Delp and P. W. Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306 of *Proceedings of SPIE*, pages 23–34, San Jose, CA, 2004. SPIE.

[13] J. Fridrich, M. Goljan, P. Losiněk, and D. Soukal. Writing on wet paper. *IEEE Transactions on Signal Processing*, 53(10):3923–3935, 2005.

[14] J. Fridrich, M. Goljan, P. Losiněk, and D. Soukal. Writing on wet paper. In E. J. Delp and P. W. Wong, editors, *Security, Steganography and Watermarking of Multimedia Contents VII*, volume 5681 of *Proceedings of SPIE*, pages 328–340. SPIE, San Jose, CA, 2005.

[15] J. Fridrich, M. Goljan, and D. Soukal. Perturbed quantization steganography with wet paper codes. In *Proceedings of the ACM workshop on Multimedia and Security (MM&Sec)*, pages 4–15, New York, NY, 2004. ACM Press.

[16] J. Fridrich, M. Goljan, and D. Soukal. Efficient wet paper codes. In M. Barni, J. Herrera-Joancomartí, S. Katzenbeisser, and F. Pérez-González, editors, *Information Hiding (7th International Workshop)*, volume 3727 of *Lecture Notes in Computer Science*, pages 204–218. Springer-Verlag, Berlin Heidelberg, 2005.

[17] A. D. Ker. A weighted stego image detector for sequential LSB replacement. In *Third International Symposium on Information Assurance and Security*, pages 453–456. IEEE Computer Society, 2007.

[18] A. D. Ker and R. Böhme. Revisiting weighted stego-image steganalysis. In E. J. Delp, P. W. Wong, J. Dittmann, and N. D. Memon, editors, *Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, volume 6819 of *Proceedings of SPIE*, page 681905, San Jose, CA, 2008. SPIE.

[19] A. D. Ker, T. Pevný, J. Kodovský, and J. Fridrich. The square root law of steganographic capacity. In *Proceedings of the ACM workshop on Multimedia and Security (MM&Sec)*, pages 107–116, New York, NY, USA, 2008. ACM Press.

[20] M. Luby. LT codes. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, FOCS 2002, pages 271–280. IEEE Computer Society, 2002.

[21] B. Schneier. *Applied cryptography: protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., New York, NY, 2nd edition, 1995.

[22] P. Schöttle and R. Böhme. A game-theoretic approach to content-adaptive steganography. In M. Kirchner and D. Ghosal, editors, *Information Hiding (14th International Workshop)*, volume 7692 of *Lecture Notes in Computer Science*, pages 125–141. Springer-Verlag, Berlin Heidelberg, 2012.

[23] P. Schöttle, S. Korff, and R. Böhme. Weighted stego-image steganalysis for naive content-adaptive embedding. In *Proceedings of the IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 193–198. IEEE, 2012.

[24] L. von Ahn and N. J. Hopper. Public-key steganography. In C. Cachin and J. L. Camenisch, editors, *Proceedings of EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 323–341, Berlin Heidelberg, 2004. Springer-Verlag.

APPENDIX

A. ROBUST SOLITON DISTRIBUTION

The Robust Soliton distribution is a discrete probability distribution introduced by Luby [20]. It is an extension of the Ideal Soliton distribution (ISD) $\rho(i)$. It is defined as follows [20]:

$$\rho(i) = \begin{cases} \frac{1}{q} & i = 1 \\ \frac{1}{i(i-1)} & i = 2, \ldots, q \end{cases} \tag{15}$$

The RSD is defined as the normalized sum of the ISD and $\tau(i)$ [11, 20]:

$$\mu(i) = \frac{\rho(i) + \tau(i)}{\eta} \tag{16}$$

$$\tau(i) = \begin{cases} \frac{T}{iq} & i = 1, \ldots \lfloor q/T \rfloor - 1 \\ \frac{T \log(T/\delta)}{q} & i = \lfloor q/T \rfloor \\ 0 & i = \lfloor q/T \rfloor + 1, \ldots, q \end{cases} \tag{17}$$

with $\eta = \sum_{i=1}^{q} (\rho(i) + \tau(i))$ and $T = c \log(q/\delta) \sqrt{q}$ [11, 20].

Generating matrix $\boldsymbol{D}$ according to this distribution is supposed to make the resulting system of linear equations sparse while remaining solvable [11]. Figure 7 shows the distribution of the RSD for parameter $\delta = 0.5$, $c = 0.1$ and $q = 100$. While the high probabilities for low Hamming-weights are supposed to make the matrix sparse, the high probability for Hamming-weight 18 is supposed to ensure that the system remains solvable [11].

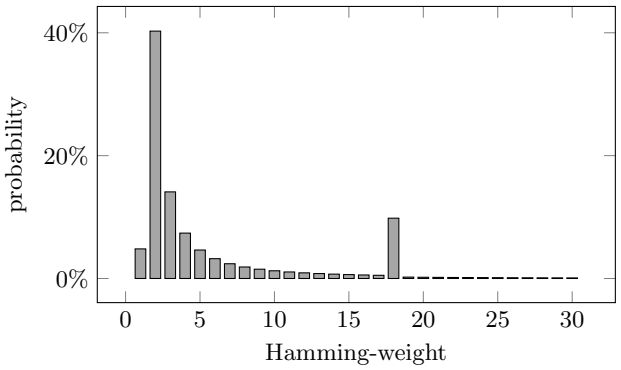


Figure 7: Robust Soliton distribution for $\delta = 0.5$, $c = 0.1$ and $q = 100$