A Shared Steganographic File System with Error Correction

Josep Domingo-Ferrer and Maria Bras-Amorós

Universitat Rovira i Virgili
UNESCO Chair in Data Privacy
Department of Computer Engineering and Mathematics
Av. Països Catalans 26, E-43007 Tarragona, Catalonia
{josep.domingo,maria.bras}@urv.cat

Abstract. Steganographic file systems are file systems where the location and even the existence of files are unknown to the users not having stored them. If the file system can be written to by several users, a user may inadvertently damage the files stored by other users. In this paper, solutions to the collision problem are proposed which rely on error-correcting codes. The storage efficiency and the privacy offered by the proposed protocols are analytically assessed.

Keywords: Information privacy and security, steganographic file systems, error-correcting codes.

1 Introduction

Steganographic file systems [1] were introduced a decade ago as file systems where the location and even the existence of files are unknown to the users not having stored them. This feature is a problem when the file system is a shared one that can be written to by several users: as a result of a (unknown) collision, a user may inadvertently damage the files stored by other users [5].

There are two simplistic approaches to deal with the collision problem:

- *Privacy reduction.* A possible solution is to reduce the risk of collisions between users by reducing the freedom of each user for placing her file in the system. However, the smaller the freedom, the smaller is privacy: the location of the files of a user becomes easier to guess by other users or by external intruders.
- *Efficiency reduction.* If the total size of user files stored in the steganographic file system is less than the size of the file system by several orders of magnitude, collisions are unlikely to occur. However, this entails a very inefficient storage use.

1.1 Contribution and Plan of This Paper

Intermediate solutions between the two simplistic approaches above are explored in this paper. We present a shared steganographic file system protocol whose aim is to offer a tradeoff between privacy and storage efficiency.

V. Torra and Y. Narukawa (Eds.): MDAI 2008, LNAI 5285, pp. 227–238, 2008.

Section 2 describes a collision-resistant file system where user files are stored as segments of straight lines in a two-dimensional space. Section 3 analyzes the effects of discretization on privacy and storage efficiency. Section 4 sketches conclusions and future work.

2 A Collision-Resistant Shared Steganographic File System in the Plane

Consider a steganographic shared file system FS whose storage space consists of a publicly readable bit matrix

$$\mathbf{M} = \{m_{ij} : 0 \leq i, j \leq N-1\}$$

of size $N \times N$, for a large positive integer N. Let the bits m_{ij} be randomly initialized (*e.g.* a bit can be set to 0 or 1 with probability 1/2). We assume that the file system FS is trusted to keep secret the locations where files are recorded.

A first naive protocol allowing a user U to store a file in $\mathbf{M}$ is given below. The protocols in this paper all use pseudorandom number generators seeded by an initial value only known to each user, so that the user can reproduce at any moment the pseudorandom values she has previously generated (*e.g.* to recover from the file system the files she has stored in it). Also, the protocols in this paper assume that the user is able to privately send information to the file system (*e.g.* by encrypting it under the file system's public key).

Protocol 1

1. *Let $F = (f_0, \cdots, f_{l-1})$ be a bit vector representing the file that U wants to store in FS. Let $\{S_i^F : i \geq 0\}$ be a pseudorandom sequence generated by U specifically for file F. For $i = 0$ to $l-1$, U computes $E = (e_0, \cdots, e_{l-1})$, where $e_i := f_i \oplus S_i^F$ and $\oplus$ is addition modulo 2.*
2. *U pseudorandomly selects a slope $a \in \mathbb{R}$, an integer intercept b and an integer offset c, the latter randomly drawn from $\{0, \cdots, N-1\}$. (In this protocol and in the remaining ones of this paper, intercepts and slopes are sampled from a uniform distribution, and slopes are selected by uniformly sampling an angle from $[-\pi/2, \pi/2)$ and then taking as slope the tangent of that angle.)*
3. *U privately sends a, b, c and E to FS.*
4. *For $i = 0$ to $l-1$, FS stores e_i in the component (x_i, y_i) of $\mathbf{M}$, where if $|a| \geq 1$ or $a = 0$*

$$x_i = (i + c) \bmod N$$

$$y_i = [a(i + c) + b] \bmod N$$

and if $|a| \in (0, 1)$

$$x_i = [(1/a)(i + c) + b] \bmod N$$

$$y_i = (i + c) \bmod N$$

with $[\cdot]$ *being the integer rounding operator. Thus, E is stored as a segment of a pseudorandomly chosen straight line. (If slopes* $|a| \in (0, 1)$ *were not handled separately, one would come up with many points sharing the same ordinate* y_i *due to rounding, which could result in a lot of overlap at the crossing of lines with slopes under 1.) In order to prevent E from wrapping around itself regardless of the slope a, one must require* $l \leq N$.

To recover file F from FS, a user U must know (a, b, c, l) and the pseudorandom sequence $\{S_i^F : i \geq 0\}$. Thanks to the use of the pseudorandom sequence $\{S_i^F | i \geq 0\}$, Protocol 1 fulfills the standard requirement of steganographic file systems that stored files should be indistinguishable from the random, unused positions. As an additional precaution, FS is assumed to randomly tweak straight segments formed by unused bit positions from time to time; those fake file insertions thwart intruders from trying to infer the location of files by observing the changes in the bit matrix.

A problem when using Protocol 1 in a shared file system is that the segments corresponding to files of different users might cross each other. This is a concern only for shared file systems: if there was a single user, one could assume she can select the parameters for her files so that no crossing occurs. If a file F' is stored after a file F and the segments of both files cross each other, the bit of F' at the crossing position overrides the bit of F. With probability $1/2$, this causes an error in file F. This has two undesirable effects, which we next describe along with possible solutions:

- Errors damage the integrity of the stored files. Using error-correcting codes (ECC) to encode files before storage appears as a natural way to mitigate this problem.
- Even if errors can be corrected, their very existence may leak to a user the location of the files belonging to other users. Indeed, assume that a user U stores two files F_1 and F_2 in the file system and then keeps retrieving both files very often in order to detect any simultaneous appearance of errors in them due to their being crossed by a new file. Now, if a user U' stores a new file F' that crosses F_1 and F_2 and both crossings cause simultaneous single-bit errors in F_1 and F_2, respectively, U can infer that F' lies on the straight line connecting both erroneous bit positions. A possible way to repair this weakness would be to require that the bits of each file be randomly shuffled by the file system using a secret permutation different for each file; but this would require the help of the file system for file retrieval, which would be a disadvantage with respect to Protocol 1 above. A better option is to divide $\mathbf{M}$ into several tiles and to split a file into fragments and store each fragment in a different tile, which makes it difficult for an intruder U to locate all the fragments of a file by merely watching the crossing errors.

From now on, by a $[n, k]$ ECC we mean an error correcting code with length n and dimension k. Its transmission rate is $R = \frac{k}{n}$ (*e.g.* see [4]). If the minimum distance of the code is d, then its correction capability is $t := \lfloor \frac{d-1}{2} \rfloor$.

Protocol 2 below incorporates the above solutions to deal with crossing errors. The idea is to make tiling compatible with the storage of large files, by using several tiles to store a file: thus, only a fragment of the file is stored in a particular tile. The $N \times N$ bit matrix $\mathbf{M}$ is considered to be divided into h^2 tiles of size $N/h \times N/h$ for some integer divisor h of N. For each tile $T_{r,s}$, the file system maintains a global counter $\nu_{r,s}$ initially set to 0 that counts the number of file fragments stored in the tile.

Protocol 2

1. *Let $F = (f_0, \cdots, f_{l-1})$ be a bit vector representing the file that U wants to store in FS.*
2. *U encodes F using a binary $[n, k]$ ECC with $n \leq N/h$, to obtain an encoded file $E = (e_0, \cdots, e_{m-1})$.*
3. *U generates a pseudorandom sequence $\{S_i^E : i \geq 0\}$ specifically for file E and computes $E' = (e'_0, \cdots, e'_{m-1})$, where $e'_i := e_i \oplus S_i^E$ for $i = 0$ to $m - 1$.*
4. *U computes a pseudorandom enumeration of the tiles whose global counter is less than $t + 1$; each tile appears in the enumeration a number of times equal to the difference between $t + 1$ and its global counter (in general, repetitions of the same tile appear in different positions of the pseudorandom enumeration). Let the enumeration be $T_{r_0,s_0}, T_{r_1,s_1}, \cdots$ Each file fragment in a tile can contain up to N/h bits of E', so as many tiles from the enumeration will be taken in turn as needed to store the m bits of E'. If the number h' of necessary tiles is greater than the number of tiles in the enumeration, then exit the Protocol (there is insufficient storage to hold E'). Note that, initially, all h^2 tiles can be used,* so m can be as large as $Nh(t + 1)$ bits, which implies a maximum l as large as $\lfloor \frac{Nh(t+1)}{n} \rfloor k$; *the maximum size of storable new files will decrease as the number of tiles holding already $t + 1$ file fragments increases.*
5. *For $0 \leq j < h'$ user U pseudorandomly selects a slope $a_{r_j,s_j} \in \mathbb{R}$, an integer intercept b_{r_j,s_j} and an integer offset c_{r_j,s_j}, with the latter two randomly drawn from $\{0, \cdots, N/h - 1\}$.*
6. *U privately sends to FS the indexes of the h' chosen tiles and the slopes, intercepts and offsets chosen for each tile. U publicly sends E' to FS.*
7. *For $0 \leq j < h'$ the file system FS does:*
   (a) *Let $\nu_{r_j,s_j} = \nu_{r_j,s_j} + 1$;*
   (b) *For $i = 0$ to $N/h - 1$, store the bit $e'_{jN/h+i}$ of E in the component (x_i, y_i) of $\mathbf{M}$, where if $|a| \geq 1$ or $a = 0$*

$$x_i = r_j N/h + ((i + c_{r_j,s_j}) \bmod (N/h))$$

$$y_i = s_j N/h + ([a_{r_j,s_j}(i + c_{r_j,s_j}) + b_{r_j,s_j}] \bmod (N/h))$$

and if $|a| \in (0, 1)$

$$x_i = r_j N/h + ([(1/a_{r_j,s_j})(i + c_{r_j,s_j}) + b_{r_j,s_j}] \bmod (N/h))$$

$$y_i = s_j N/h + ((i + c_{r_j,s_j}) \bmod (N/h))$$

If slopes $|a| \in (0, 1)$ were not handled separately in the above protocol, one would come up with many points sharing the same ordinate y_i due to rounding, which could result in a lot of overlap at the crossing of lines with slopes under 1. The following lemma is a quantification of storage efficiency.

Lemma 1. *The maximum storage efficiency achievable by Protocol 2 when using a* $[n, k]$ *ECC is* $\lfloor \frac{Nh(t+1)}{n} \rfloor \frac{k}{N^2}$. *Consequently, if* $\frac{Nh(t+1)}{n} \simeq \lfloor \frac{Nh(t+1)}{n} \rfloor$ *then the maximum storage efficiency is approximately* $(t+1)Rh/N$, *where* R *is the transmission rate of the code.*

Proof: The optimum case is the one mentioned in Step 4 of Protocol 2: a file of size $\lfloor \frac{Nh(t+1)}{n} \rfloor k$ is stored across the tiles. By diving this file size by the total storage available N^2, we get the efficiency above. □

In the above protocol, tiles should stay large enough so that they can still be viewed as plane regions and bits can be viewed as "points" in those regions: *e.g.* if a tile consists of very few bits, rounding when computing straight segments causes a lot of crossings to occur (see Section 3.2 below for an analysis of the impact of multibit crossings on efficiency).

As to privacy, in Protocol 2 tiles only contain fragments of a file, which is thus harder to locate. Indeed, to locate a file an intruder needs to determine which tiles store fragments of the file and, within each of those tiles, where does the line storing the corresponding fragment lie. See Section 3.1 below for a discussion on the difficulty of guessing a specific line within a certain tile.

The price paid for the above advantages of Protocol 2 is that the user needs to keep more information to recover a file than in Protocol 1: h' slopes, intercepts and offsets (instead of a single slope, intercept and offset required by the previous protocols).

Example 1. Consider a shared steganographic file system with a bit matrix $\mathbf{M}$ of size $2^{20} \times 2^{20}$ (1 Terabit). Divide $\mathbf{M}$ into $2^5 \times 2^5$ tiles of $2^{15} \times 2^{15}$ bits each. Consider the primitive BCH code of length $2^{15} - 1$ over $\mathbb{F}_2$ with designed correction capability equal to $t = 10$ (i.e. designed minimum distance equal to 21). Its dimension is 32617. This means that within each tile we can encode up to 32617 bits of $t + 1 = 11$ file fragments as 11 codewords of length $2^{15} - 1$. If these codewords are inserted in the file system and they only cross each other at one bit position, it will be possible to correct at retrieval time any error in any of the 11 file fragments that is due to crossings. A total of 1024×11 file fragments with at most 32617 bits each can be inserted in $\mathbf{M}$. In this case, the maximum storage efficiency is approximately $0.000334146 \simeq 3 \cdot 10^{-4}$.

Alternatively, we can also divide $\mathbf{M}$ into $2^{10} \times 2^{10}$ tiles of $2^{10} \times 2^{10}$ bits each. Consider the primitive BCH code of length $2^{10} - 1$ over $\mathbb{F}_2$ with designed correction capability equal to $t = 10$ (i.e. designed minimum distance equal to 21). Its dimension is 923. This means that within each block we can encode 11 file fragments with at most 923 bits each as 11 codewords of length $2^{10} - 1$. If these codewords are inserted in the file system and they only cross each other at one bit position, it will be possible to correct at retrieval time any error in any

of the 11 stored file fragments that is due to crossings. A total of $2^{20} \times 11$ file fragments with at most 923 bits can be inserted in $\mathbf{M}$. In this case, the maximum storage efficiency is approximately $0.0096921 \simeq 10^{-2}$. □

3 The Effects of Discretization on Privacy and Efficiency

In Section 2, we have used the idealization that the bit matrix $\mathbf{M}$ can be regarded as a plane, where files are stored as straight lines. In fact, $\mathbf{M}$ is a grid, so files are stored as near-straight lines with discretized slopes. This has some practical consequences:

- Unlike in a tile in a continuous plane, in an $N \times N$ grid, the maximum length of a straight line as we define it can be no more than N regardless of its slope, which is consistent with the limitation on the length of the stored files in the above protocols.
- In a grid, the range of possible values for the slopes and the intercepts is finite [3], which has privacy implications: the uncertainty of an intruder about the location of the line storing a particular file is finite.
- In a grid, two discretized "straight" lines may cross each other in more than one bit position. This has implications for efficiency: there may be more than one error caused by the crossing of two files, which further limits the number of storable files in an error-free manner with respect to the continuous idealization used in Section 2.

In the next subsections, we analyze the above mentioned privacy and efficiency implications.

3.1 Discretization and Privacy

The protocols above encrypt the file by adding a pseudorandom sequence to it in order to conceal its redundancy in front of an intruder. However, the intruder could blindly (*i.e.* randomly) try to guess the line segment where a file or file fragment is stored. If she succeeded at that, she could for example tweak all bits along that line to destroy the file or file fragment; or she could attempt its decryption. Therefore, the intruder's uncertainty about the slope and the intercept of the line are measures of privacy.

For the sake of clarity, we make the following simplifications:

- We will initially assume that no tiling is used and that straight lines are stored in the entire $N \times N$ bit matrix $\mathbf{M}$. (To adapt the discussion for the case of tiling, the length N/h of the tile side must be used instead of N.)
- We will assume that the length of the file is the maximum value N. This is the worst case for privacy, because for files with maximum length, the intruder does not need to worry about the file length l and the offset c.

If $\mathbf{M}$ is an $N \times N$ grid, the intercept b is an integer value between 0 and $N-1$, where all values in the range have the same probability from the intruder's

Table 1. Discretized slopes and their probabilities in an $N \times N$ bit matrix

Index i	Slope $\hat{a}_i$	Probability $p(\hat{a}_i)$
0	$-\infty$	$(\arctan(-2(N-1)) + \pi/2)/\pi$
1	$-(N-1)$	$(\arctan(-2(N-1)/3) - \arctan(-2(N-1)))/\pi$
2	$-(N-1)/2$	$(\arctan(-2(N-1)/5) - \arctan(-2(N-1)/3))/\pi$
3	$-(N-1)/3$	$(\arctan(-2(N-1)/7) - \arctan(-2(N-1)/5))/\pi$
...	...	...
$N-1$	-1	$(\arctan(-(N-3/2)/(N-1)) - \arctan(-(N-1)/(N-3/2)))/\pi$
N	$-(N-2)/(N-1)$	$(\arctan(-(N-5/2)/(N-1)) - \arctan(-(N-3/2)/(N-1)))/\pi$
$N+1$	$-(N-3)/(N-1)$	$(\arctan(-(N-7/2)/(N-1)) - \arctan(-(N-5/2)/(N-1)))/\pi$
...	...	...
$2N-2$	0	$(\arctan(1/(2(N-1))) - \arctan(-1/(2(N-1))))/\pi$
$2N-1$	$1/(N-1)$	$(\arctan(3/(2(N-1))) - \arctan(1/(2(N-1))))/\pi$
$2N$	$2/(N-1)$	$(\arctan(5/(2(N-1))) - \arctan(3/(2(N-1))))/\pi$
...	...	...
$3N-3$	1	$(\arctan((N-1)/(N-3/2)) - \arctan((N-3/2)/(N-1)))/\pi$
$3N-2$	$(N-1)/(N-2)$	$(\arctan((N-1)/(N-5/2)) - \arctan((N-1)/(N-3/2)))/\pi$
...	...	...
$4N-3$	$N-1$	$(\arctan(2(N-1)) - \arctan(2(N-1)/3))/\pi$
$4N-4$	$+\infty$	$(\pi/2 - \arctan(2(N-1)))/\pi$

viewpoint. Thus, Shannon's entropy can be used to measure the intruder's uncertainty about the intercept as

$$H(b) = \log_2 N \tag{1}$$

The analysis for the discretized slope $\hat{a}$ is a bit more complex. We will give a lower bound for the entropy $H(\hat{a})$. A subset of the possible slopes in an $N \times N$ matrix is listed in the second column of Table 1: those $4N-3$ slopes are obtained when a straight segment starting at one corner of the $N \times N$ grid successively touches the bit positions in the opposite edges of the grid (similar to the hand of a clock touching the marks of the seconds). If these were the only possible slopes, from the intruder's point of view, the probability $p(\hat{a})$ of each discretized slope $\hat{a}$ is proportional to the size of the fraction of the angular range $[-\pi/2, \pi/2)$ such that the continuous slopes with angles in that fraction round to $\hat{a}$. For example,

$$p(+\infty) = (\pi/2 - \arctan((N-1)/(1/2)))/\pi$$

$$p(N-1) = (\arctan((N-1)/(1/2)) - \arctan((N-1)/(3/2)))/\pi$$

and so on (see third column of Table 1). In this way, the intruder's uncertainty on the slope can be lower-bounded as

$$H(\hat{a}) \geq -\sum_{i=0}^{4N-4} p(\hat{a}_i) \log_2 p(\hat{a}_i) \tag{2}$$

The following conclusions on privacy can be drawn:

- The overall privacy can be measured as the joint entropy $H(\hat{a}, b) = H(\hat{a}) + H(b)$, where additivity holds because $\hat{a}$ and b are independently selected.

- As it could be expected, both $H(b)$ and the lower bound for $H(\hat{a})$ increase with N. Since using tiles instead of the entire matrix involves replacing N with N/h, the privacy of the slope and the intercept within a tile is reduced.
- In Protocol 2, a file is stored across several tiles, each with its own slope and intercept. Assuming the worst case in which the intruder has managed to determine the h' tiles holding the file, the intruder's uncertainty on the tile intercepts and slopes can be measured by the following joint entropies

$$H(b_{r_0,s_0},\cdots,b_{r_{h'-1},s_{h'-1}}) = \sum_{i=0}^{h'-1} H(b_{r_i,s_i}) \tag{3}$$

$$H(\hat{a}_{r_0,s_0},\cdots,\hat{a}_{r_{h'-1},s_{h'-1}}) = \sum_{i=0}^{h'-1} H(\hat{a}_{r_i,s_i}) \tag{4}$$

  where we have used that the slopes and intercepts are independently chosen for each tile.

3.2 Discretization and Efficiency

Two randomly chosen straight lines over the plane cross each other with probability 1 and the crossing consists of a single point. However, two discretized straight lines over an $N \times N$ grid may cross each other at more than one point. We analyze in this section the implications of this fact. For simplicity, we assume that the file system consists of a single tile (no tiling); the adaptation to several tiles is straightforward.

The first thing to note is that two discretized straight lines may have several crossings because of the modular operations. See the left-hand side of Figure 1 for an illustration. The discretized versions of $y = x$ (black dots) and $y = 5x$ (white dots) are depicted on an $N \times N$ grid, where $N = 2^4$; the number of crossings is four.

Secondly, the overlap at each crossing can consist of several bits, depending on the slopes of both lines. See the right-hand side of Figure 1. There, the discretized versions of $y = 10x/3$ and $y = 16x/5$ are depicted; there is a single crossing which involves four bits.

Analytically counting the expected number of crossings and the number of overlaps per crossing is by no means straightforward. We refer the reader to [2] for a preliminary discussion of this problem. For the sake of pragmatism, we have chosen here a simulation approach.

For $N \in \{2^i : i = 9,\cdots,21\}$ and $t \in \{1,\cdots,9\}$ we have conducted the following experiment:

1. Repeat 5000 times
   (a) Throw $t+1$ random digital straight lines of length N like the ones described in Protocol 1 into an $N \times N$ bit matrix;
   (b) Count the number $x_{N,t}$ of overlaps between the first line thrown and the subsequent t lines;

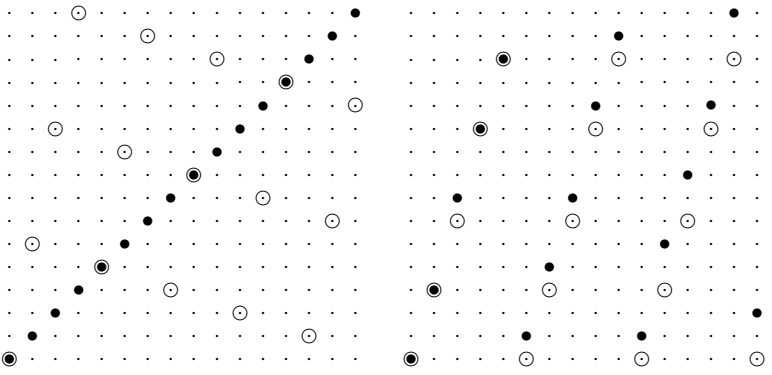

Fig. 1. Crossings of two discretized straight lines. Left, two lines with several crossings. Right, two lines with one crossing involving several bits.

2. Compute the histogram of the relative frequencies of the random variable $X_{N,t}$ modeling the number of overlaps;
3. Compute the expected value $E(X_{N,t})$ of the number of overlaps;
4. Compute the variance $\sigma^2_{X_{N,t}}$ of the number of overlaps;

For each choice of N and t, we are interested in finding a value t' such that $P(X_{N,t} \leq t') \approx 1$. We make the simplifying assumption that successive overlaps in a line occur independently. In this case, when throwing two discretized straight lines, the probability that a bit in the first line thrown is "trodden" by the second line can be estimated as $p = E(X_{N,1})/N$. When throwing $t+1$ lines, the probability that a bit in the first line is trodden by any of the subsequent t lines can be estimated as $p' = 1 - (1-p)^t$. Now, the number of bits in the first line that are trodden by any of the subsequent t lines can be modeled as a binomial random variable with N trials and success probability p'. Therefore, under the above independence assumption, $E(X_{N,t})$ can be approximated as

$$Np' \tag{5}$$

and $\sigma^2_{X_{N,t}}$ can be approximated as

$$Np'(1-p') \tag{6}$$

In Figure 2, for $N = 2^9$ and $N = 2^{12}$ and several values of t we depict $E(X_{N,t})$ and $E(X_{N,t}) + 3\sigma^2_{X_{N,t}}$, as well as their approximations resulting from Expressions (5) and (6). Figure 3 is analogous for 2^{15} and 2^{18}, respectively. Two observations are in order here:

- The experimental results obtained show that over 99% of the area in the histograms of $X_{N,t}$ for all N and t tried lies left of $E(X_{N,t}) + 3\sigma^2_{X_{N,t}}$.
- The independence-based approximations resulting from Expressions (5) and (6) overestimate the corresponding empirical magnitudes for all t tried, except $t = 1$.

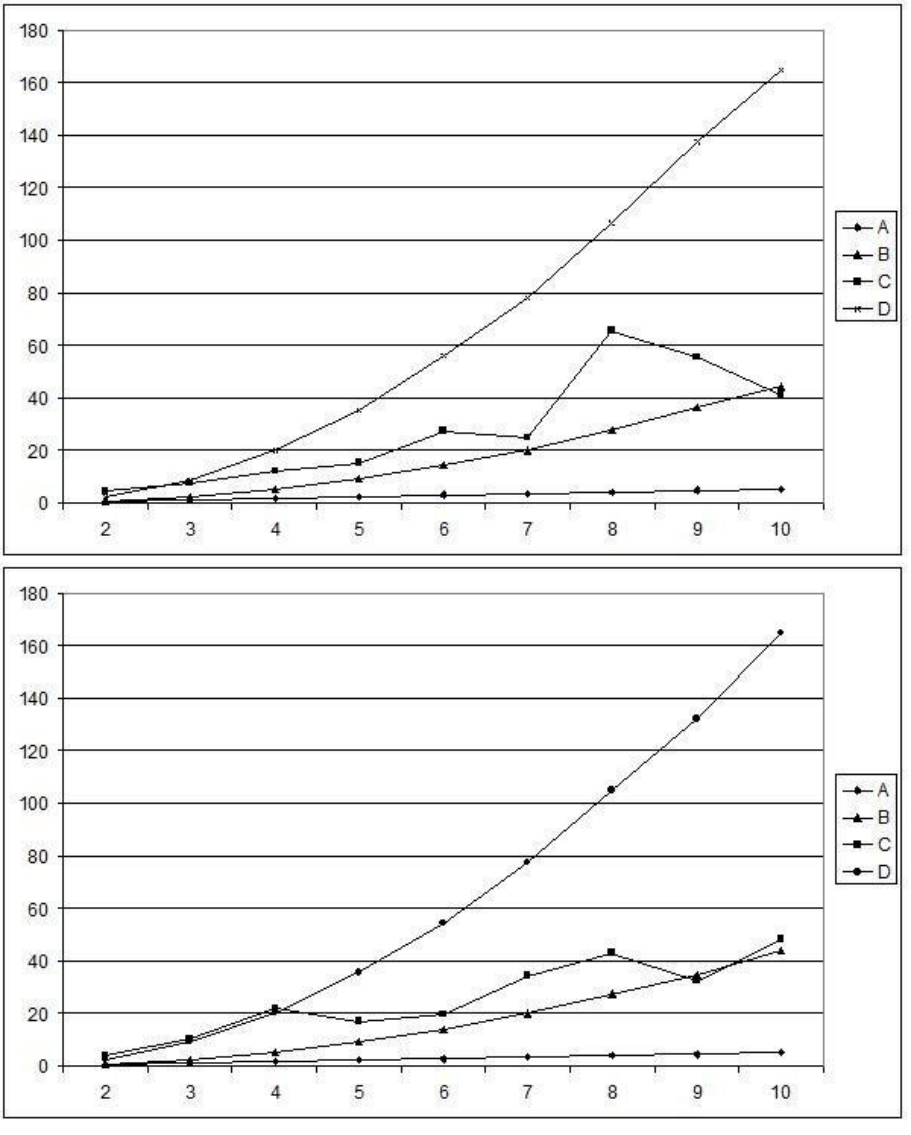

Fig. 2. Top: For $N = 2^9$ and several values of t (abscissae), A) Empirical $E(X_{N,t})$; B) Approximation Np'; C) Empirical $E(X_{N,t})+3\sigma^2_{X_{n,t}}$; D) Approximation $Np'+3Np'(1-p')$. Bottom: same for $N = 2^{12}$ and several values of t.

Therefore, to adapt protocols in Section 2 and Lemma 1 on efficiency to the real situation of discrete lines with multibit overlaps we must use a binary ECC with error-correction capability t' which, with probability almost one, is greater than the number of errors caused by overlaps in any stored file. It follows from the empirical discussion above that a suitable choice when $t > 1$ is

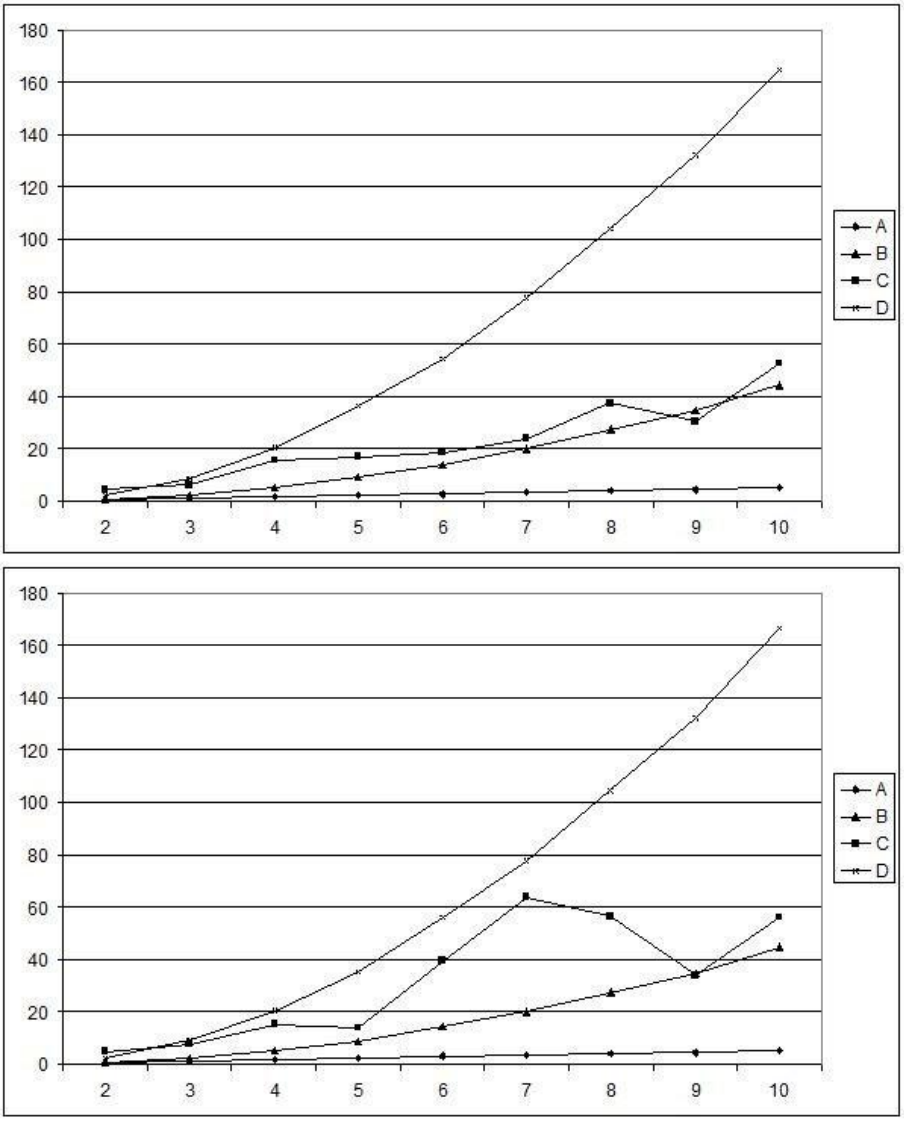

Fig. 3. Top: For $N = 2^{15}$ and several values of t (abscissae), A) Empirical $E(X_{N,t})$; B) Approximation Np'; C) Empirical $E(X_{N,t})+3\sigma^2_{X_{n,t}}$; D) Approximation $Np'+3Np'(1-p')$. Bottom: same for $N = 2^{18}$ and several values of t.

$$t' := Np' + 3Np'(1 - p') \tag{7}$$

To use Expression (7), only $E(X_{N,1})$ needs to be computed empirically. For $t = 1$, a suitable choice is

$$t' := E(X_{N,1}) + 3\sigma^2_{X_{N,1}}$$

Note that we are placing ourselves in the worst case in which every overlap causes one error (the actual probability of an overlap causing an error is 1/2). As one would expect, usually $t' > t$, which decreases storage efficiency with respect to the continuous idealization of Section 2 because a t'-correcting code will normally have a lower transmission rate than a t-correcting code.

4 Conclusion

This contribution has presented protocols for storing files in a shared steganographic file system. Their novelty is that they deal with the errors caused by successive file insertions. The privacy and the storage efficiency offered by the proposed approach have been quantified.

Future work will include finding alternative ways to store files which, without degrading privacy, are more storage-efficient than straight lines and/or can guarantee 100% correction probability.

Acknowledgments and Disclaimer

Thanks go to Markku Saarinen for providing the original motivation for this work and suggesting the use of ECC in shared steganographic file systems over the plane. We are also indebted to Francesc Sebé, Josep M. Mateo-Sanz and an anonymous reviewer for their comments. We also wish to thank Glòria Pujol for her help with the simulation work. This work was partly supported by the Spanish Government through projects TSI2007-65406-C03-01 "E-AEGIS" and CONSOLIDER INGENIO 2010 CSD2007-00004 "ARES", and by the Government of Catalonia under grant 2005 SGR 00446. The authors are with the UNESCO Chair in Data Privacy, but they are solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization.

References

1. Anderson, R., Needham, R., Shamir, A.: The steganographic file system. In: Aucsmith, D. (ed.) Information Hiding, 2nd International Workshop, Portland, Oregon, USA (1998)
2. Bras-Amorós, M., Domingo-Ferrer, J.: On overlappings of digital straight lines. In: Jornadas de Matemática Discreta y Algorítmica, Lleida, Catalonia (2008)
3. Koplowitz, J., Lindenbaum, M., Bruckstein, A.: The number of digital straight lines on an $N \times N$ grid. IEEE Transactions on Information Theory 36(1), 192–197 (1990)
4. Roth, R.M.: Introduction to Coding Theory. Cambridge University Press, Cambridge (2006)
5. Zhou, X.: Steganographic File System. Ph.D thesis, School of Computing, National University of Singapore (2005)