available at www.sciencedirect.com

SCIENCE DIRECT®

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Steganography in games: A general methodology and its application to the game of Go

Julio C. Hernandez-Castro[a,*], *Ignacio Blasco-Lopez*[b], *Juan M. Estevez-Tapiador*[a], *Arturo Ribagorda-Garnacho*[a]

[a]*Computer Science Department, Carlos III University of Madrid, Avda. Universidad 30, 28911 Leganés, Madrid, Spain*
[b]*División TSO – Área de Telecomunicaciones Tecsidel, 28016 Madrid, Spain*

ARTICLE INFO

Article history:
Received 8 October 2005
Revised 11 November 2005
Accepted 2 December 2005

Keywords:
Steganography in games
Information hiding
Covert channels
Steganalysis
Go

ABSTRACT

Techniques to hide valuable information within seemingly harmless messages have been widely used for centuries. Typically, their use is appropriate when encryption is not available or not adequate (e.g. when available cryptography is too weak), or simply when it is convenient that no external observer can infer that some information is being exchanged. In the digital era, new cover mediums for hiding data in communication are constantly being proposed, from the classical image files (such as bmp, gif, and jpg formats) to audio files (i.e. wav and mp3), text and html documents, emails disguised as spam, TCP/IP packets, executables programs, DNA strands, etc. In this work, we present and analyze a novel methodology that illustrates how games (such as Chess, Backgammon, Go, etc.) can be used to hide digital contents. We also look at some of its possible advantages and limitations when compared with other techniques, discussing some improvements and extensions. Finally, we present the results of a first implementation of an open-source prototype, called StegoGo, for hiding digital contents in Go games.

1. Introduction

The word *steganography* literally means *covered writing*. It was coined in 1499 by Trithemius, a monk who encoded letters as religious words in such a way as to turn covert messages into apparently meaningful prayers. It comprises a broad range of different methods for secret communication that conceal the very existence of hidden data. Among these methods are writing over shaved slave heads, or tables covered with wax, knitting, invisible inks, microdots, phrase and character arrangements, combinations of the dots and dashes on letters *i*, *j*, *t*, and *f* giving Morse codes, null ciphers, covert channels, and spread-spectrum communications.

Steganography is, thus, the art and science of concealing the existence of information within seemingly innocuous carriers (e.g. images, audio files, text, html, etc.) or, as defined in Johnson et al. (2000) "of communicating in such a way that the presence of a message cannot be detected". The objectives of steganography are quite different from those of cryptography. While cryptographic techniques *scramble* messages so that if intercepted, messages cannot be understood, steganography *camouflages* a message to hide its existence and makes it seem almost invisible, thus concealing the fact that a message is being sent altogether. An encrypted message may draw suspicion while an invisible message will not. Steganography provides a means of secret communication which cannot be

* *Corresponding author.*
E-mail addresses: jcesar@inf.uc3m.es (J. C. Hernandez-Castro), ignacio.blasco@mad.tecsidel.es (I. Blasco-Lopez), jestevez@inf.uc3m.es (J. M. Estevez-Tapiador), arturo@inf.uc3m.es (A. Ribagorda-Garnacho)
0167-4048/$ – see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2005.12.001

removed without significantly altering the data in which it is embedded. For the embedded data to be recovered, an attacker should first find a way to detect it, only then being able to mount a classical cryptanalysis attack (it is common practice to cipher messages before hiding). The associated field which tries to detect, recover, or eliminate messages hidden by steganographic techniques is called steganalysis.

Steganography is nowadays in common use, both for copyright protection of digital media as well as for exchanging information without raising suspicions (of law enforcement agencies and/or dictatorial governments, for example). There are many tools for hiding messages in images, audio files, video, and other not so common media such as text, TCP/IP packets (Murdoch and Lewis, 2005; Hintz, 2002; Giffn et al., 2002), executable files (El-Khalil and Keromytis, 2004), DNA strands (Clelland et al., 1999), XML (Inoue et al., 2001), etc. Nowadays, the huge amount of Internet traffic allows easy covering of hidden messages in media where the addition of this extra information is very difficult to detect even when complex, time-consuming steganalytic techniques are applied.

The USA government imposed severe restrictions during some phases of WWII with the aim of trying to render useless as many covert communication channels as possible (Kahn, 1967). The post banned a large class of objects, including crosswords and newspaper clippings; lovers' X's in letters were deleted; watch hands were shifted; orders for flowers could not specify either the kind of flower or the date of delivery; and items such as loose stamps and blank paper were replaced. Thousands of people were involved in reading mail, looking for language which appeared to be forced to accommodate secret messages. They also routinely rephrased and rewrote letters and telegrams. In one case, a censor changed ''father is dead'' to ''father is deceased'', which was followed by the revealing reply ''is father dead or deceased?''

Covert channels in games have not been widely studied, possibly with the only exception of Bridge, which could be used for allowing collusion among participants. The problem with Bridge is especially interesting: algorithms for transmitting information between partners during bidding are legal and might provide a great advantage to the teams that better manage them. Typical schemes usually provide a way for a player to encode information about his hand in the cards he plays. Upon seeing this, his partner could make a more precise contract. An additional twist in Bridge is that, while covert channels are permitted by the rules, if a player is asked what the meaning of a bid is, he should answer truthfully, so the information sent by this channel could not be a secret. Different strategies have been developed to overcome this problem, the most successful to date is presented in Winkler (1983), where the author developed bidding conventions whereby one player could send her partner secret information about her hand, which is totally unrelated to the actual bid and completely indecipherable to the opponents, even though the protocol is public and known to them.

A recent and notable experiment is described in Murdoch and Zielinski (2004), where a team of researchers developed a number of different programs to play and collude in a Connect-4 variant online programming competition at Cambridge University. These programs acted as independent players in the contest, but were able to authenticate themselves and follow different playing strategies whenever playing an opponent which has taken part in the plot or not. The purpose was to maximize the overall winning probabilities of the team by creating programs that will deliberately lose in the presence of other authenticated partners, and play for a sure draw with the rest of contenders. Authors showed how this strategy led to a win in the tournament, even if they did not have especially good algorithms for the Connect-4 variant, or at least not significantly better than those proposed by other contestants.

The critical process of mutual authentication could be carried on by means of different covert channels: authors cite and analyze the timing of moves (by purposely delaying the execution of certain moves) and the choice of equivalent moves, and chose the latter. This approach tries to codify some bits by choosing moves within the set of moves that could be shown not to change the outcome of the game when compared with the best possible move.

The idea is to order the aforementioned set of n equivalent moves by some prearranged criterion, then chose the rth move to send the value r. Authentication is performed by repeating this process a given number of times (10 is shown to be adequate for this variant of the Connect-4 game) and checking if the thus received sequence equals the values generated by a given PRNG with a certain known common seed.

There are many other covert channels that if not directly relevant to computer games might be used in different games after slight modifications. Most of these covert channels have been described in the bibliography for the analysis of multilevel secure computer systems, notably in Light Pink Book (1993).

There are very few additional works that relate data hiding and games, and none in the way proposed in this article. The most remarkable is Ding et al. (2000), where the authors describe techniques extracted from the Tangram and Conway's game for developing an algorithm intended for scrambling pictures.

The rest of this paper is organized as follows. In section 'A framework for hiding data in games', we present a formal framework for hiding data in games, including the theoretical limitation in terms of the channel capacity to hide information. Next, in section 'Approaches to hide data in games' we elaborate on how this methodology can be applied in practical scenarios, focusing on the game of Go to illustrate the main ideas. Section 'Steganalysis' is devoted to discuss steganalysis techniques associated to our proposal, as well as the countermeasures that can be deployed. In section 'An implementation for Go', we describe a working implementation for the game of Go, and further investigate the aforementioned attacks in a real environment. Finally, section 'Conclusions' summarizes our work by presenting the main conclusions.

2. A framework for hiding data in games

Before detailing our methodology, we first provide a brief background on formal games as considered in Game Theory. Interested readers can find a good introduction to the field in Gibbons (1992).

An extensive-form game is defined by the tuple:

$$\langle P, A, Q, p, (\mathcal{I}_i)_{i\in P}, (\preceq_i)_{i\in P}\rangle \quad (1)$$

where,

- P is a set of players.
- A is a set of actions, i.e. *moves* available to players at different stages of the game.
- Q is a set of action sequences, satisfying:
  - $\epsilon \in Q$, where ϵ is the empty sequence,
  - if $(a_k)_{k=1}^{w} \in Q$ and $0 < v < w$, then $(a_k)_{k=1}^{v} \in Q$,
  - if $(a_k)_{k=1}^{v} \in Q \ \ \forall v \geq 1$, then $(a_k)_{k=1}^{\infty} \in Q$.

If q is a sequence of actions and a is an action, then $q\cdot a$ denotes the action composed by q followed by a. A finite sequence of actions $q \in Q$ is said to be *terminal* if there is no a such that $q\cdot a \in Q$. The set of terminal sequences of actions is denoted by Z. Finally, $A(q) = \{a \in A : q\cdot a \in Q\}$ denotes the set of available actions after $q \in Q\backslash Z$.

- p is the player function. It assigns a player $p(q) \in P$ to every non-terminal sequence $q \in Q\backslash Z$. The interpretation is that player $p(q)$ has the turn after the sequence of actions q.
- $\mathcal{I}_i$ is an information partition for player $i \in P$. It is a partition of the set $\{q \in Q\backslash Z : p(q) = i\}$ preserving the property that if the sequences q and q' are in the same information set $I_i \in \mathcal{I}_i$, then $A(q) = A(q')$.
- $\preceq_i$ is a preference relation of player $i \in P$ on Z.

The common interpretation of an extensive-form game is the following. The game can be thought of as a tree, where the edges and the vertices are associated to actions and sequences of actions, respectively. The empty sequence ϵ represents the root of the tree. The game begins at ϵ and ends at a terminal node. After any non-terminal sequence of actions $q \in Q\backslash Z$, the player given by $p(q)$ chooses an available action from the set $A(q)$. Next, q is extended with a, and the current history of the game becomes $q\cdot a$. Terminal vertices are those that cannot be followed by any other action. When a sequence of actions q reaches a terminal vertex, the game finishes.

The sequences $q \in Z$ are the possible outcomes of the game. The preference relation $\preceq_i$ establishes which outcomes are preferred by player i. Thus, if $q, q' \in Z$ and $q \preceq_i q'$, then player i prefers q' to q. The most usual form of representing preference relations is payoffs. A vector $y(q) = (y_i(q))_{i\in P}$ of real numbers is assigned to every terminal sequence of actions $q \in Z$, in such a way that $q \preceq_i q' \Leftrightarrow y_i(q) \leq y_i(q')$. The value $y_i(q)$ can be interpreted as a measure of how much player i gains when the game is developed as described by q.

Information sets represent the information available to players at every stage of the game. When an information set covers several nodes, then the player does not know in which part of the tree she is – or, equivalently, she does not know the last action of her rival. Usually, nodes belonging to the same information set are graphically represented by a dashed line that links them together. When an information set is not a singleton (i.e. it has more than one node), it is necessary to specify the player beliefs. Formally, beliefs are represented by a probability distribution over the nodes belonging to the information sets.

If there exists at least one information set $I_i \in \mathcal{I}_i$ such that $|I_i| > 1$, then the game is called a game of imperfect information. On the contrary, if for all players every information set is a singleton, then the game is called a game of perfect information.

2.1. Strategies

A strategy for player i is a function s_i specifying what action i should carry out at each of her information sets. A strategy s_i assigns an action $a \in A(p)$ to each non-terminal sequence of actions q, preserving that if q and q' are in the same information set of player i, then s_i assigns the same action to both the sequences. The set of all strategies of player i is denoted by S_i.

A strategy profile is a vector $(s_i)_{i\in P}$ of individual strategies, one for each player. Specifying completely a strategy profile determines univocally the outcome of the game. Sometimes a strategy profile is written $(s_j, (s_i)_{i\in P\backslash\{j\}})$ in order to emphasize the specification of strategy s_j for player j.

2.2. Information and channel capacity

The key point of our approach relies on the set of strategies $(S_i)_{i\in P}$ available to each player after her opponent's move. For simplicity, we will assume that $|P| = 2$, with $P = \{i, j\}$, even though extending our results to a general game with n players is straightforward. We also assume the case of a perfect alternating game, in which each player has always the turn after her rival action.

Suppose that

$$q = (a_k)_{k=1}^{v} \in Q\backslash Z \quad (2)$$

is a non-terminal action sequence, satisfying that $p(q) = i$ (i.e. after q, it is player i's turn). We denote by

$$S_i(q) = \left(s_i^1(q), \ldots, s_i^k(q), \ldots, s_i^N(q)\right) \quad (3)$$

the possible strategies available to player i after action sequence q. We also assume that $S_i(q)$ is ordered according to the expected payoff associated to each strategy, in such a way that $s_i^k(q)$ is the k-th best response after q.

In a typical scenario, player i is expected to follow the best response, i.e. $s_i^1(q)$. However, deliberatively selecting a specific move in $S_i(q)$ can be used to send information to the other players. In general, in a case like that illustrated by expression (3), player i can send up to $\lfloor \log_2(N) \rfloor$ bits per move. Assuming a similar behavior by player j and a terminal action sequence

$$z = (a_k)_{k=1}^{w} \in Z \quad (4)$$

players can exchange during the game up to:

$$I(z) = \sum_{i=1}^{w} \left\lfloor \log_2\left(\left|S_{p\left((a_k)_{k=1}^{i}\right)}\left((a_k)_{k=1}^{i}\right)\right|\right)\right\rfloor \text{ bits} \quad (5)$$

Note that the channel capacity is limited by the very nature of the game and, more specifically, by the amount of strategies available to players at each stage of the game. It is clear that for this scheme to properly operate, both players must have access to the structure of ordered strategies given by expression (3). Otherwise, the information sent may be incorrectly decoded.

3. Approaches to hide data in games

The idea of hiding information into game strategies can be applied in two different scenarios. The first and more straightforward one consists in playing a new game from scratch. However, a similar approach can be put in practice by adding extra information to an already played game. In this case, two parties can append comments and variations at different stages of the game, as if they were discussing alternative strategies to those actually played.

Next, we describe briefly these two approaches.

3.1. *Playing a new game*

Both parts of the communication channel should have access to exactly the same software, and share a common secret key (used for encrypting the hidden contents) and some other parameters (i.e. the board size, the software version, etc.) to assure that the software's internal states are reproducible by the two parties.

The main idea is to compute for each position played in the game all the movements which are over a certain threshold value T, and then codify in the selection of the actual played move some bits of the message we want to hide. Say that, at a given position, there are gm (good moves) moves not worst than the given threshold T (analogously, we can fix a certain number n and pick the move within the list of best n moves), then sort them by their value (according to the evaluation function) and select the i-th move to codify the binary representation of the number i. In this way, in each position we will be able to hide around $\log_2(\text{gm})$ bits of information.

The choice of threshold T allows us to adjust some aspects of the scheme. By increasing T, the channel capacity will grow, since we will have more gm to embed hidden bits. Alternatively, decreasing T will result in a higher invisibility of the hidden contents, for chosen strategies do not deviate significantly from optimal.

Other possibilities exist, for example the use of a dynamic T which varies over time or follows a random distribution in order to simulate the game between two players with a good knowledge of openings (i.e. no really bad moves at the beginning of the game) but not so strong in the middle or endgame. The other communicant can easily decode these moves into binary data by arriving at the same position, re-computing the move list and searching for the code of the move actually played.

3.2. *Inserting data into a game*

Among the multiple possible options here, we have focused on which we consider as the most straightforward way to embed hidden data into games. Moreover, it presents the benefits of simplicity, high embedding capability, and security. It simply consists in including variations and comments in already played games.

The general idea is basically to follow the overall aforementioned procedure for hiding some bits in every move, but in this case moves correspond to variants of the main line actually played on the game (see Fig. 1).

```
[Event ''Swiss Open'']
[Site ''Gausdal'']
[Date ''1992.06.05'']
[White ''Taimanov'']
[Black ''Davies'']
[Result ''0-1'']

1.Nf3 Nf6 2.b3 d5 3.Bb2 c6 4.e3 Bg4 5.h3 Bxf3
6.Qxf3 Nbd7 7.g4 e5 8.g5 Ne4 9.h4 Bb4 10.Bh3 Qe7
(10...0-0 11.Bxd7 Qxd7 12.Bxe5) (10...0-0 11.Qf5)
11.Bxd7+ Qxd7 12.Bxe5 0-0 13.a3 Ba5 14.b4 Bc7 15.Bxc7
(15.Bb2 a5 16.d3 Nd6 17.Nd2 Nf5) 15...Qxc7 16.Qf4 Qxf4
17.exf4 Rae8 18.Kf1 f6 19.d3 Nd6 20.Nc3 fxg5 21.fxg5 Nf5
22.Kg2 Nd4 23.Rac1 (23.Rae1 Rxe1 24.Rxe1 Nxc2 25.Rd1 a6)
23...Rf4 24.Kf1 Ref8 25.Nd1 Nf5 26.Rh3 Nxh4 27.c4 d4 28.Rc2
Nf3 29.Re2 Rg4 30.Rh1 Rf5 31.g6 Rxg6 32.Nb2 Kf7 33.c5 Rfg5
34.Rc2 Re6 35.Re2 Rh6 0-1
```

Fig. 1 – Example of a chess game with some variants of the main line pointed out within a frame. We can see that multiple variants are possible, even in the same position.

In this way, at certain positions in the game we would introduce variants of the main line played. Each of these variants could embed bits by various means: choosing at what movement does the variant begin, the length of the variant, and obviously, in every move of the variant, just by using the same algorithm described above for codifying hidden data in moves.

Another possible channel for embedding information is the inclusion of comments. For instance, the following set of 8 comments: {''black losses here'', ''black is loosing'', ''white wins'', ''with a winning advantage to white'', ''white wins easily'', ''white will win'', ''black is lost'', ''very bad for black''} might be used to express the same and, simultaneously, codify three bits. Likewise, any other technique derived from the field of text steganography could be applied in this context.

Generally, this embedding technique will allow to hide much more information than the first one, and will be in most cases harder to detect. However, it also has the important drawback of a much lower robustness: it suffices to delete all comments to erase the hidden data, but at the cost of severely decreasing the attractiveness and usefulness of the game file.

3.3. *Further considerations*

Using this approach, we can develop applications that allow to hide data into apparently innocuous games. As a proof-of-concept, we have created one of such applications devoted to the game of Go, which is based on the open-source GNUGo program (see below section 'An implementation for Go'). Similarly, this could be done for Chess and its dozens of variants, including Bughouse, Fischer Random, Loser's, Atomic and others, not to mention other popular games such as Backgammon and Othello. New games with even higher capacity for embedding hidden information could exist or be created.

In any case, this kind of steganographic application can be used to ease covert online communication through the Internet by connecting to one of the many hosts that provide these services. As an example, at the Internet Chess Club (ICC) more

than 100 tournaments take place everyday, with an average number of connected users of around 2100. They also provide a database of more than 2,000,000 chess games. This offers a huge number of opportunities for making covert communication very difficult to track. Furthermore, ICC is not the only possibility. Similar services are offered at the Free Internet Chess Server (FICS), US Chess Live, Chess AnyTime, Chess.Net, and many other national servers. Likewise, this ease for hidden online communication is also true for Go, for which there are many servers all around the world, like the Internet Go Server (IGS), the Kiseido Go Server (KGS), the No Name Go Server (NNGS), and the Dragon Go Server (DGS).

Creating a new game for hiding data seems the most suitable possibility for online two-party communication, while inserting data into already existing games is much more appropriate for off-line communication and the storing of sensitive information. Both alternatives are possible in any of the mentioned servers, which allow online playing and automatically create a database with all played games.

The main advantages of our approach over other steganalytic techniques are the following:

- Its novelty, compared with classical methods such as image or audio steganography.
- The ability to disguise the communication as casual. Both parties could play dozens of games a day, with only one of them carrying hidden data, which will not probably be noticed. Other classical methods, such as directly sending images as email attachments might more easily raise suspicion.
- Classical steganographic methods, such as the exchange of images or sound files as email attachments, or even posting these files in webpages or news servers, do not really allow for truly online communication. For simulate online communication in all these methods, one needs to send a number of hidden messages, a fact that could easily increase the attention of a warden. Our method, on the other hand, allows easy two-party online communication while playing one or various games online.

4. Steganalysis

The process of detecting steganographic messages is known as steganalysis, and a particular steganalytic technique is called an attack. In the following, we discuss some possible statistical attacks to our proposal.

Statistical attacks try to detect significant deviations, produced by the insertion of hidden data into the stego medium, from a given statistical model. The main game characteristics that might be sensible to these changes are the following:

- *Game length*: in general, the selection of moves other than the best in a certain position should produce games where one player (the one that codifies data) plays steadily worst than the other. This difference should in most cases lead to shorter games. As a general rule of thumb, increasing the number of embedded bits per move should decrease the game length and vice versa.
- *Game level*: for the same reasons, the codifying part would in general be playing at a lower level than the non-codifying part, and routinely making more mistakes (or less accurate moves). The bigger the number of embedded bits per move, the bigger the level difference between the two players.
- *Game result*: consequently, not only the game result will generally be favourable for the non-codifying player, but also the overall winning difference, when compared with games played between players of similar strength.

These values could also give a good hint of the number of zeros in the codified sequence if no encryption or permutations are used. This is easily avoidable by using a block cipher for encrypting the movements following a common secret key.

For the variation wherein we hide data into a game by inserting comments and variations, an obvious attack path is to erase all variations from a given game. This will surely erase all hidden data, but to the prize of loosing valuable comments and variations of games that, otherwise, could be not so interesting to the average player. Literally thousands of games between professionals would not be steadily downloaded and be highly valuable unless they are heavily annotated and full of variations that explain beginners and also advanced players the reasons behind the Pro's moves. One can also try to include more variants for making harder the task of recovering the hidden data, thus desynchronizing the coding and decoding algorithms. Both attacks could be successful, and are hard if not impossible to stop, but at the cost of a lack of functionality. On the other hand, when correctly implemented and applied, this technique could become virtually undetectable.

4.1. *Countermeasures*

Among the possible countermeasures that we can use to avoid these attacks, we have considered the following. In the case of games with handicap, as Go, the *komi*[1] should be increased with the aim of avoiding that the codifying part always loses. This will make the game more fair, shortening the strength differences derived from the codification of information, thus making this codification harder to detect.

Another general solution to the same problem is to force both parties to always codify something, even if the communication is only one-way. The other party could in this case codify the trivial message $Encipher_k(000\cdots00)$. The aim of this is to make its playing strength very close to that of the really codifying part, trying to make the truly codifying party indistinguishable from the other. One important drawback of the proposed system is that its capacity is quite small. To overcome this limitation, we recommend the use of a compression phase (Huffman compression optimized for English text is a reasonable option, and has been implemented in our application) before actually codifying the information. This will typically improve capacity around a 30% at almost no cost.

[1] For those not familiar with Go, the *komi* is a compensation – a number of points – received by white player for the disadvantageous fact that he moves second. The amount of *komi* received is agreed before starting the game.

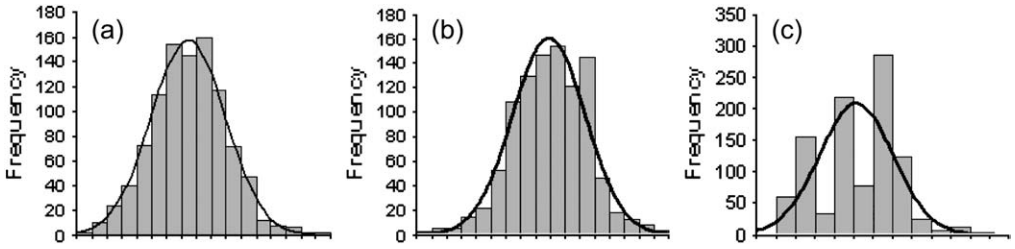

Fig. 2 – Distribution of moves in the sets (a) C_0, (b) C_1, and (c) C_2.

5. An implementation for Go

We have developed an open-source software project called StegoGo which aims to implement our proposed data hiding techniques in the game of Go, and examine the applicability of the proposed attacks and countermeasures.

StegoGo follows our proposed strategy for hiding data in a new game (described in section 'Playing a new game') and works over the GNUGo playing program (The GNUGo project). This first version of the StegoGo program hides three bits of information per move, being capable of compressing (Huffman, 1952) and encrypting (TEA, Wheeler and Needham, 1994) data as suggested in our proposal.

The results obtained with this first implementation were, mostly, as predicted. For testing purposes, we generated three groups of 1000 games named C_0, C_1, and C_2, meaning, respectively, that there were 0 communicants (i.e. a normal game between two instances of GNUGo), 1 communicant, or both parties were communicating.

We then tested the validity of some of the attacks already presented. Our aim is to make the transmission of information hard to detect by the warden, so as to produce Go games hard to distinguish from those with 0 communicants, which are normal Go games of GNUGo against itself. Our general assumption was that the games in C_2 would be much closer to C_0 (i.e. harder to distinguish) than those in C_1. In other words, we believed that if both communicants transmitted information, the not-so-good moves will somehow compensate one another, and the general result would be much similar to when no communication at all took place. But empirically, that was consistently proven to be not the case. As we can see in Fig. 2, the distribution of the number of moves in the case C_1 is actually much closer to C_0 than that of C_2.

This was quite a surprising and unexpected result which was consistent when we performed more experiments to verify its correctness. Intuitively speaking, when the two parties hide data in their moves, we find not a cancellation but an augmentation effect.

However, these differences in the distribution of the number of moves only become apparent (and statistically significant) after a large number of games have been observed and analyzed. For example, for obtaining these figures we needed 1000 games in each of the three groups (C_0, C_1, C_2). However, this could not be quite the case on a real environment where access to the games will not probably be so easy.

The steganalytic technique tested here, based on detecting differences in the distributions of the number of moves per game, could greatly suffer from the lack of enough games. For example, we performed multiple experiments with as many as 50 games in each of the groups, and it was consistently not possible to distinguish the games in C_1 and C_2 from those in C_0 (at a significance level $\alpha = 0.01$). They systematically passed various statistical tests (t-test, sign-test, Wilcoxon signed rank test, and Friedman ANOVA) that decide if two samples are likely or not to belong to the same population. Moreover, in many cases the three populations passed the Friedman test, thus being, to some extend, statistically indistinguishable.

We have repeated these experiments with 100 games in each of the groups, and the results obtained are quite surprising. Although it is generally possible to distinguish C_0 from C_2 (but not always), C_0 and C_1 were still hard to distinguish. This implies that more than 100 games would be needed to mount a successful steganalytic attack against StegoGo being used by only one of the players to secretly communicate data at a rate of three bits per move, based on this scheme. These results are presented in Tables 1 and 2.

Analogously, attacks based on game results could be possible, as stated before. We carried out experiments focusing on the results of the games, that is, who won (Black or White) and the winning difference (in the game of Go, the difference

Table 1 – P-values for different instances of the Friedman ANOVA statistical test comparing the sets C_0, C_1, and C_2 for different sizes

Number of games	vs.	C_1				C_2			
		Test 1	Test 2	Test 3	Test 4	Test 1	Test 2	Test 3	Test 4
$N = 50$	C_0	0.6682	0.4750	0.0321	0.0078	0.4658	0.0172	0.1572	0.6616
$N = 100$	C_0	0.1913	0.1095	0.0066	0.2640	0.0035	0.0208	0.0690	0.0051
$N = 1000$	C_0	<0.0001				<0.0001			

Table 2 – P-values for different instances of the paired sample t-test comparing the sets C_0, C_1, and C_2 for different sizes

Number of games	vs.	C_1				C_2			
		Test 1	Test 2	Test 3	Test 4	Test 1	Test 2	Test 3	Test 4
$N = 50$	C_0	0.7754	0.2064	0.3732	0.0013	0.5601	0.3422	0.0423	0.7159
$N = 100$	C_0	0.1181	0.0712	0.0530	0.4086	0.0042	0.0005	0.1494	0.0041
$N = 1000$	C_0	<0.0001				<0.0001			

between the territory conquered by the winner and the loser). Again, we analyzed the results of 1000 StegoGo games for each type, C_0, C_1, and C_2. This time, the results were as expected, but to some extent opposed to those previously obtained when considering the game length. It seems that from the point of view of the game result, games in C_2 are much similar to those in C_0 than games in C_1. This can be explained by the fact that the party who is sending information is forced to systematically deviate from the optimal strategy. On the contrary, the other player can choose better moves. As the game end approaches, this asymmetry will imply a disadvantageous result for the player who is hiding the information. It is clear that this situation does not happen in the case of C_2, for both players follow identical behaviors. Therefore, in a real scenario even if one party has no data to hide, the best practice against steganalysis based on game result is for both players to hide data (even garbage) in their moves. This result was consistent over a number of tests.

However, we must take into account that these results depend strongly on the number of bits codified in each move. In fact, the default rate of StegoGo (three bits/move) could be considered too high. Reducing this amount to two or one will make all these attacks much more difficult to succeed, requiring an even higher number of games to arrive to statistically significant results, although this will also reduce the capacity of the overall scheme.

One question to consider is to what extent these attacks are relevant for the case where the two communicants want their game to simulate a real game between humans. If they try to simulate a game between two instances of GNUGo, all these analyses are pertinent; they are interesting, anyway, to see how the insertion of hidden information produces changes over what would have been a normal Go game, in other words, to detect the additional load associated with the use of cover channels. In any other case, a model of games between humans should be developed before we can measure any distance or difference against this normal human behavior. The authors believe that this is such a difficult task to accomplish that StegoGo could be, and probably will be for years, safely used to communicate discretely over Internet. We also hope these ideas will soon be applied to many other games, some of which could perhaps show to have a higher hiding capacity than Go.

6. Conclusions

We have presented a new steganographic paradigm for embedding hidden data into games. In doing so, we have tried to be as general as possible, giving a theoretical framework for general games in extensive form.

For this general model, we have introduced some new schemes for statistical steganalysis based on differences in game length, result, and level that could aid in detecting games with hidden data. We have also proposed some general solutions to overcome these novel attacks, which were presented and analyzed over StegoGo, a working implementation on the game of Go. The experiments carried out with this tool have provided us with some useful insights about practical aspects of our proposal, and the severity and real implications of the aforementioned attacks.

Besides its novelty, an important advantage of our proposal is that it is quite difficult to detect, especially taking into account that thousands of games are played everyday in many different Internet servers, both free and commercial. Furthermore, it allows true online communication while playing one or various games, a feature that cannot be provided by classical steganographic methods such as the exchange of images or audio files as email attachments.

Acknowledgments

The authors are grateful to the anonymous reviewers for their insights and fruitful comments during the review process, which greatly contributed to improve the quality of the original manuscript.

REFERENCES

A guide to understanding covert channel analysis of trusted systems. NCSC-TG-030: Light Pink Book; November 1993.

Clelland CT, Risca V, Bancroft C. Hiding messages in DNA microdots. Nature 1999;399:533–4.

Ding W, Yan WQ, Qi DX. A novel digital image hiding technology based on Tangram and Conway's game. In: Proceedings of the 2000 international conference on image processing (ICIP'00), vol. 1; 2000. p. 601–4.

El-Khalil R, Keromytis AD. Hydan: hiding information in program binaries. In: Proceedings of the sixth international conference on information and communications security (ICICS). Lecture notes on computer science. Springer-Verlag; 2004. p. 187–99.

Gibbons R. Game theory for applied economists. Princeton University Press; 1992.

Giffn J, Greenstadt R, Litwack P, Tibbetts R. Covert messaging in TCP. In: Dingledine R, Syverson P, editors. Privacy enhancing technologies. Lecture notes in computer science, vol. 2482. Springer-Verlag; 2002. p. 194–208.

Hintz A. Covert channels in TCP and IP headers. Presentation at DEFCON 10. Available from: <http://guh.nu/projects/cc/>.

Huffman DA. A method for the construction of minimum redundancy codes. Proc Inst Radio Eng September 1952;40(9): 1098–101.

Inoue S, Makino K, Murase I, Takizawa O, Matsumoto T, Nakagawa H. A proposal on information hiding methods using XML. The first workshop on NLP and XML. Tokyo; November 2001.
Johnson NF, Duric Z, Jajodia S. Information hiding: steganography and watermarking – attacks and countermeasures. Norwell, MA, New York, The Hague, London: Kluwer Academic Press; 2000.
Kahn David. The codebreakers. New York, NY: The Macmillan Company; 1967.
Murdoch SJ, Lewis S. Embedding covert channels into TCP/IP. In: Proceedings of the seventh information hiding workshop. Lecture notes in computer science. Springer-Verlag; 2005.
Murdoch SJ, Zielinski P. Covert channels for collusion in online computer games. In: Proceedings of the sixth information hiding workshop. Lecture notes in computer science, vol. 3200. Springer-Verlag; 2004. p. 355.
StegoGo project, <http://sourceforge.net/projects/stegogo/>.
The GNUGo project, <http://www.gnu.org/software/gnugo/>.
Wheeler DJ, Needham RM. TEA, a tiny encryption algorithm. In: Preneel B, editor. Fast software encryption: second international workshop. Lecture notes in computer science, vol. 1008. Springer-Verlag; 1994. p. 363–6.
Winkler P. The advent of cryptology in the game of bridge. Cryptologia October 1983;7(4):327–32.

Julio Cesar Hernandez-Castro is Associate Professor at the Computer Security Group of the Computer Science Department of Carlos III University, Madrid, Spain. He has a B.Sc. in Mathematics, a M.Sc. in Coding Theory and Network Security, and a Ph.D. in Computer Science. His interests are mainly focused in cryptology, network security, steganography and evolutionary computation. He likes playing chess, and is a decent opponent. He also likes Go, but he is not.

Ignacio Blasco-Lopez has a M.Sc. in Computer Science. Currently he works at the Telecommunications Department of Tecsidel, a Spanish telecommunications company while pursuing his Ph.D. in Computer Science at Complutense University. His areas of interest mainly comprise steganography and steganalysis.

Juan M. Estevez-Tapiador is Associate Professor at the Computer Science Department of the Carlos III University of Madrid. He has a M.Sc. in Computer Science from the University of Granada (2000), where he obtained the Best Student Academic Award, and a Ph.D. in Computer Science (2004) from the same university. His research is focused on cryptography and information security, especially in formal methods applied to computer security, design and analysis of cryptographic protocols, and some theoretical aspects of network security. In these fields, he has published around 20 papers in specialized journals and conference proceedings. He is member of the program committee of several conferences related to information security and serves as regular referee for various journals.

Arturo Ribagorda-Garnacho is Full Professor at Carlos III University, where he is also the Head of the Computer Security Group, and currently acts as the Director of the Computer Science Department. He has a M.Sc. in Telecommunications Engineering and a Ph.D. in Computer Science. He is one of the pioneers of computer security in Spain, having more than 25 years of research and development experience in this field. He has authored 4 books and more than 100 articles in several areas of information security.