ACSIJ Advances in Computer Science: an International Journal, Vol. 4, Issue 3, No.15 , May 2015
ISSN : 2322-5157
www.ACSIJ.org

ACSIJ
WWW.ACSIJ.ORG

StegoRogue: Steganography in Two-Dimensional Video Game Maps

Chance Gibbs[1] and Narasimha Shashidhar[2]

[1] Department of Computer Science, Sam Houston State University
Huntsville, Texas 77341, United States
chance@shsu.edu

[2] Department of Computer Science, Sam Houston State University
Huntsville, Texas 77341, United States
karpoor@shsu.edu

Abstract

Techniques for hiding information in an innocuous carrier medium have been used throughout history. Video games represent an extremely popular, widely distributed, data-heavy medium, which makes them an optimal carrier medium for hidden messages and files. Despite this, the application of steganography to video games is a relatively untouched topic. We present StegoRogue, a content-aware method by which information can be hidden within a two-dimensional video game map.

Keywords: *Steganography, video games, content aware, supraliminal.*

1. Introduction

Steganography is the art and science of concealing a message within a carrier medium. The first recorded instance of this practice dates back to 440 BC, when Demaratus passed a message hidden beneath the wax surface of a writing tablet, in order to warn Greece about an impending attack [1]. Johannes Trithemius was the first to label this practice as *steganography*, in his most famous work, *Steganographia* [2]. The term took on a new meaning in the 1980s, when personal computer users began hiding information in the least-significant bits of image (and later, sound and video) files. This paved the way for modern digital steganography.

Though they are often used together, steganography is different from cryptography. The purpose of cryptography is to make a message unreadable if intercepted by a third party. In steganography, a message is hidden, in an attempt to prevent a third party from knowing that the hidden message is being passed at all. Steganalysis, as a field, encompasses the identification and recovery of messages hidden using steganography.

Steganography has been explored broadly across many types of digital media; however, very little research has been published on its application to video games. Hernandez-Castro et al. [3] published a paper describing a method by which a message could be hidden during a game of Go, at a rate of three bits per player move. Hale et al. [4] proposed four methods by which information could be hidden in games created using Valve's Source Engine and distributed through the Steam platform, and discussed the impact of video game steganography on digital forensic investigations. Vines and Kohno [5] described and implemented Rook, a system by which data is embedded within video game network traffic without altering the amount or length of packets.

In this paper, we demonstrate StegoRogue, a method by which a message may be steganographically hidden within a procedurally-generated two-dimensional video game map.

The rest of this paper is organized as follows. In section II, we explore the viability of video games as a steganographic carrier medium. Section III introduces the reader to Roguelikes, and the basic algorithm behind their map generation. In section IV, we describe StegoRogue, our method for hiding information in two-dimensional maps. Section V details our implementation of a tool for generating maps with hidden messages. Section VI includes pictures of maps generated using the technique and tool described in earlier sections, as well as information about the capabilities of StegoRogue. In section VII, we explore the implications of the use of steganography within video games, and suggest additional avenues of research. Section VIII summarizes the results, discussion, and future work of the paper.

2. Video Games as a Carrier Medium

Despite their relatively recent invention, video games are an extremely popular form of entertainment within the developed world. Global video games revenue in 2014 was

Copyright (c) 2015 Advances in Computer Science: an International Journal. All Rights Reserved.

estimated at $101.62bn [6]. Call of Duty: Modern Warfare 3, the most widely-sold game of all time, sold an estimated 30.42bn copies across all platforms [7]. As of 2014, it is estimated that there are 1.78bn "gamers" worldwide [8]. Video games are distributable through many different online platforms, such as Valve's Steam, Microsoft's Xbox Live Arcade, and Sony's Playstation Network, as well as through retail and physical media exchange. Video games can require many different types of files, and program sizes can differ widely; indie games such as *BootChess* can be feature-complete in fewer than 512 bytes [9], even as AAA games such as *Wolfenstein: The New Order* require a staggering 50 gigabytes of hard drive space [10]. This combination of extreme prevalence, mass distributability, and large data requirement makes video games an optimal steganographic carrier medium.

3. Roguelikes, and Their Dungeon Layouts

Invented in the early 1980s, *Rogue* was an evolution of the then-popular text adventure genre, which included games such as Crowther's *Colossal Cave Adventure*, or Adams' *Adventureland* [6]. Rogue used the newly-created *curses* library to draw primitive maps, comprised of ASCII characters, to a terminal screen. Even more noteworthy was the fact that the game levels were procedurally generated, meaning that each level was generated according to rules and algorithms, rather than by hand. This meant that every playthrough of *Rogue* was unique, which "[made] it possible for even the creators to be surprised by the game" [11]. *Rogue* has had a strong influence on many similar titles, with fans of the genre going so far as to categorize these games using the blanket term *Roguelikes*.

Roguelike map generation requires an understanding of several simple structures and objects. A map is typically a two-dimensional array of squares. Each square is represented by an ASCII character (or lack thereof). Map squares, in their most basic form, can be either open or closed. An open square represents an area of empty floor space, while a closed square is a section of untraversible terrain, such as a rock wall, or a thick patch of forest. There is no standard for how much area is represented by a single space; in many roguelikes, a small creature such as a dog occupies the same amount of space as a gigantic humanoid. A room can be loosely defined as a cluster of open map spaces. Rooms are typically of a varied size, with non-uniform entrances and exits. Within the rooms, items and non-player characters (NPCs) may be placed randomly in map squares, so as to provide challenge and reward the player. Items typically fall broadly into categories such as "food", "usable items", "treasure", and "equipment". NPCs are typically enemies, such as monsters or demons, but may also include friendly characters, such as merchants or combat allies. For the sake of the algorithm below, room generation involves the setting of map squares within the room as open, and placing items and/or NPCs randomly within the newly-opened squares. A tunnel is a series of open map squares which form an uninterrupted path between two points. These points are typically two rooms, though not always. Maps using these features are often referred to as dungeons, as they are rooms carved from stone, and often descend downwards through the use of stairs.

The basic algorithm for generating a dungeon is as follows:

Generate a room at a random location;
Place player character in the room;
Loop until map is complete:
  Create new room;
  Create tunnel to previous room from new room;
End loop;
Place stairs in the final room;

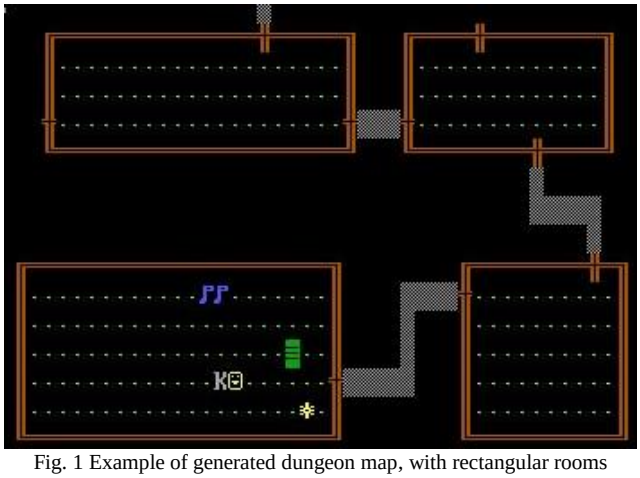

Fig. 1 Example of generated dungeon map, with rectangular rooms connected by gray tunnels.

4. The StegoRogue Generation Technique

Our technique, StegoRogue, creates meaning out of semantic components; this is in contrast to techniques which hide messages in the syntactical components of a message. Meaning is specifically derived from the location and contents of rooms. This makes the technique "content-aware" [12]. This technique could additionally be described as "supraliminal", as it "uses a perceptually significant channel to transmit a secret message" [13], which is "robust against an active warden" [14], or a third party able to intercept and analyze the carrier medium.

Copyright (c) 2015 Advances in Computer Science: an International Journal. All Rights Reserved.

Maps generated using this steganographic technique always begin at the center of the map, and branch outward to the east. The starting room can be viewed as the root of a ternary tree (a tree in which each node has up to three children), having the first room to the east as its only child. From here, each room can have up to three children, one for each remaining direction. No loops are generated within the map structure, as per our restrictions. Thus, a ternary tree perfectly describes the structure of a map generated using this technique. Messages are retrieved from the map by traversing the tree in a specific order; StegoRogue uses reverse postfix traversal, though other standard tree traversals may be used.

For the sake of exposition, we have put in place several map restrictions:

- Map dimensions are 160x86
- rooms are square, and of a uniform size (nine squares, in a 3x3 configuration)
- tunneling into an already-generated room is not allowed, as looping structures introduce ambiguity into the traversal of the map
- rooms are generated at a uniform distance from the preceding room
- map squares may contain one item, at most
- NPCs have been excluded, as their presence is not relevant to the demonstrated technique.

These restrictions are not necessary for use of the technique; however, they allow for the clearest and simplest demonstration of the generation technique.

Each non-empty room represents a character in the hidden message. Each character is represented by a series of items placed randomly within the room. A meaningful item falls within one of four categories: “food”, “usable”, “treasure”, or “equipment”. Additional non-meaningful items may also be generated, in order to further camouflage the message. The key itself is described by a key-value dictionary. In our implementation, alphanumeric characters are the dictionary keys, and the corresponding dictionary values are tuples of four integers, in the range 0 to 9. Each integer represents the number of items of a certain type to be placed within a room. For example, one key-value pair might be “’a’: (1,0,2,1)”, which would mean that the character ‘a’ would be represented on the map by a room containing one food item, zero usable items, two treasure items, and one equipment item. As the rooms are comprised of nine squares, the sum of the integers in the tuple should be no higher than 9. While it is normally possible for multiple items to inhabit a single square, it is not advised, as only one item per square may be represented visually on the map; message retrieval would then require the receiver to verify every map square containing an item, as a covered item remaining undetected by the receiver would corrupt the message.

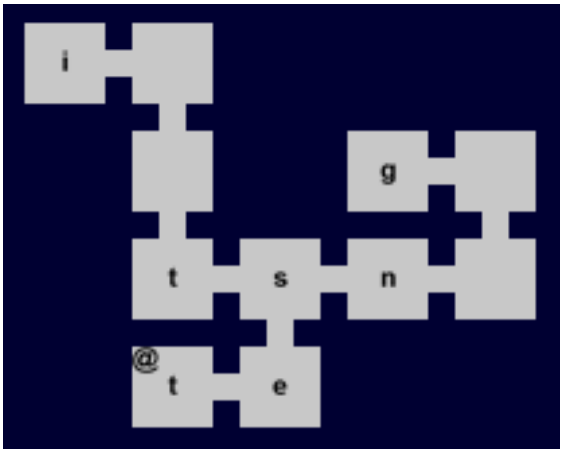

Fig. 2 Rudimentary example of map generated with our restrictions in place. Traversing this “plain-text” version of the map reveals the message “testing”. The root node would be the room containing the ‘@’ character. Note that this is a debug generation, and not a product of the full technique.

5. StegoRogue Implementation

We have created an open-source implementation of the map generation technique proposed in the preceding section. This tool generates a map based on a given message, which is a string of alphanumeric characters. Space characters are inserted into the message at random, so as to reduce the uniformity of rooms, and break up the message.

The generation algorithm is based on a randomized depth-first search algorithm. During generation, rooms branch randomly outward from the “root” room located at the center of the map. Each room represents a character in the message, and items corresponding to characters are generated at the time of room creation. As rooms are generated, they are pushed onto a stack, along with the direction of the previous room. When the stack contains a number of rooms greater than one-half of the length of the message, it is popped a random number of times; this breaks up “branches” of an excessive length.

When a new room is generated, the coordinates of the top room and direction to the previous room are retrieved by peeking at the top of the stack. The areas adjacent to the room at the top of the stack are checked in a counter-clockwise fashion, in order to determine a potential location for the new room. The counter-clockwise check creates a list of open locations, and stops when a non-empty space is encountered; this prevents the characters of the message from being generated out of order. If there are

Copyright (c) 2015 Advances in Computer Science: an International Journal. All Rights Reserved.

no available spaces, the stack is popped and the check repeated, until the point at which there are one or more valid spaces. A copy of the last item in the list of valid spaces is appended to the list, in order to bias the generation toward more fully utilizing the space available for generation. A space is chosen from the list, and a room is created in that location. This repeats until the map is complete (i.e. a room has been generated for each character in the message), or until the stack becomes empty; if the stack becomes empty, the map is considered a failure, and generation begins again with an empty map.

The algorithm can be expressed as follows:

```
Create_Room(location l, char m):
  // make map squares open at l
  …
  item_nums = key[m]; // list
  // these create respective items
  // in room at location l
  Create_Food(l, item_nums[0]);
  Create_Usables(l, item_nums[1]);
  Create_Equipment(l, item_nums[2]);
  Create_Food(l, item_nums[3]);

StegoRogue(String msg):
  insert spaces into msg;
  Create_Room(center of map, msg[0]);
  Create_Room(east of center, msg[1]);
  make tunnel to center of map;
  stack.push(east of center, west);
  while (num_rooms < length(secret)) and
  (length(stack) > 0):
    l, d = stack.top();
    new_d = choose_direction(d);
    if new_d is null:
      stack.pop();
      continue();
    new_l = new_d from l;
    Create_Room(new_l, msg[i]);
    make tunnel to l;
    stack.push(new_l, opposite of new_d);
  if num_rooms < length(secret):
    return false;
  return true;
```

6. Results

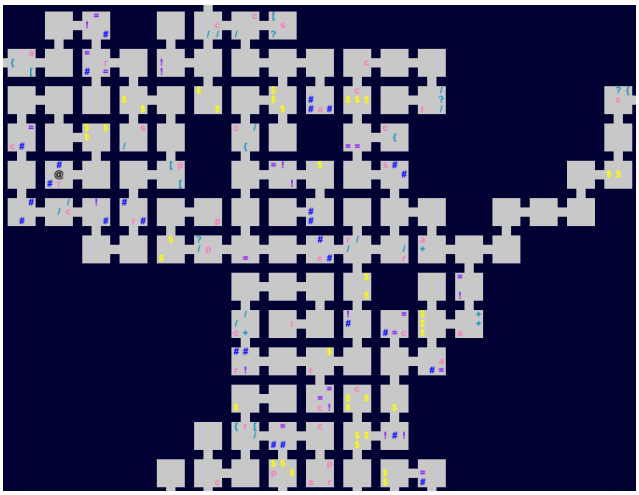

Fig. 3 Cropped example of full technique, demonstrating the use of items corresponding to characters, encoding a 400-character message.

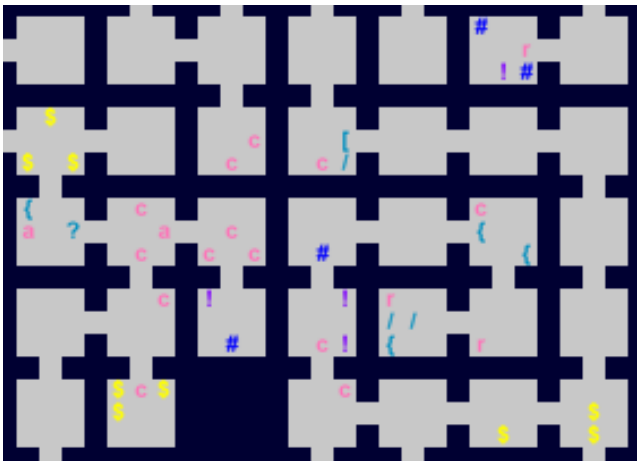

Fig. 4 Closer cropped example. Note the second two rooms in the third row; these rooms each encode a ‘c’. Because of the branching nature of the technique, their proximity is not indicative that these characters are adjacent within the message.

Within the confines of a 160x86 unit map, our implementation is able to successfully generate map configurations hiding a 400-character string. For 100 map generations using 400-character strings, each generation took 2.26 attempts, on average, with a maximum of 4 attempts. Each generation attempt takes roughly 1 second, on average.

7. Discussion and Future Work

The demonstrated method generates maps with irregular, seemingly-random item distribution. Within the context of a normal map, there is no outward indication that a message is concealed within. As the method is content aware, there is no trace of the message string within the

Copyright (c) 2015 Advances in Computer Science: an International Journal. All Rights Reserved.

map in any syntactical capacity. A message character hidden in the map is represented by items placed within a room, but a room has no meaning within the map's data structure; the map holds only square tiles, and a list of objects which exist within the boundaries of the map. The concept of a "room" exists only during level generation, and when the completed map is viewed by the human eye. This further increases the complexity of computationally analyzing a generated map in order to detect a hidden message. When combined with cryptographical techniques, such as RSA encryption, steganalysis becomes exceedingly complex.

This technique can be modified to better fit the level generation algorithms of most games with two-dimensional maps. Additionally, longer messages could be stored by "chaining" multiple dungeon levels together using some sort of thematically-appropriate intermediate path, such as a staircase or teleporter. Fully three-dimensional maps could also be generated using the basic concepts of map generation; so long as the map can be simplified into a tree without loops, the same principles can be applied.

As there has been little academic discussion of video game steganography, these channels offer a means of transferring potentially malicious or illegal data in a largely-unchecked manner. Additionally, the amount of data required by video games serves as a strong barrier against digital forensic investigation. These are the greatest contributors to the strength of this hiding technique; a suspect machine containing hundreds of gigabytes of video game data cannot feasibly be fully investigated without tools and techniques to help automate the process.

Game creation tools such as Unity, Game Maker, and the Unreal Engine have gained considerable popularity among both amateur and professional game developers; similarly, Steam's "Workshop" feature allows user-generated content to be distributed en masse, and loaded into professionally-released games. These tools allow developers to quickly create robust video game maps, which can be distributed through public channels with little concern of raising suspicion. Thus, Digital Forensics experts must be aware of methods by which video games may be used to hide information, and strive to develop robust methods and tools for analyzing and detecting such data.

With so little published work on the topic, there are many possibilities for future work in video game steganography. Further examination of individual content creation tools could provide insight into novel ways of transmitting hidden messages, as well as how to detect such transmissions. Alternative game creation tools, such as Twine (a tool used for creating interactive, nonlinear fiction), could also be explored. Additionally, as Roguelike maps typically include large areas of "empty" space between rooms, a method could be devised by which information is stored within these portions of the map, which are effectively invisible to any onlooker or player. Generation of geographical maps could also be explored, using terrain and city features in order to semantically encode a message.

8. Conclusion

Video games, with their widespread popularity, easy distribution channels, and high-density data requirements, make an optimal medium for transmitting information using steganographic techniques. We have demonstrated a novel, content aware method for hiding information using two-dimensional game maps. The concept of video game steganography is relatively untouched, and presents many new and interesting methods by which steganographic principles can be applied.

References

[1] Petitcolas, F. et al. "Information Hiding - A Survey." *Proceedings of the IEEE* 87.7 (1999): 1062-1078.

[2] Thampi, S. M. "Information hiding techniques: A tutorial review." *ISTE-STTP on Network Security & Cryptography, LBSCE* (2004).

[3] Hernandez-Castro, J. C., et al. "Steganography in games: A general methodology and its application to the game of Go." *Computers & Security* 25.1 (2006): 64-71.

[4] Hale, C., et al. "A New Villain: Investigating Steganography in Source Engine Based Video Games." *Proceedings of the 2012 Hong Kong International Conference on Engineering & Applied Science (HKICEAS), Hong Kong, China, December 14*. Vol. 16. 2012.

[5] Vines, P., Kohno, T. "Rook: Using Video Games as a Low-Bandwidth Censorship Resistant Communication Platform." Univ. Washington, Computer Science & Engineering Technical Report. UW-CSE-2015-03-03, March 2015.

[6] (2014). *Statistics and facts about the Video Game Industry* [Online]. Available: http://www.statista.com/topics/868/video-games/

[7] (2015). *Global sales (in millions of units) per game* [Online]. Available: http://www.vgchartz.com/

[8] (2014). *Number of video gamers worldwide in 2014, by region (in millions)* [Online]. Available: http://www.statista.com/statistics/293304/number-video-gamers/

[9] (2015). *BootChess* [Online]. Available: http://www.pouet.net/prod.php?which=64962

[10] (2015). *Can You Run It: Wolfenstein: The New Order* [Online]. Available: http://www.systemrequirementslab.com/cyri/requirements/wolfenstein-the-new-order/12119/?p=r

[11] Wichman, Glenn R. *A Brief History of "Rogue"* [Online]. Available: http://www.wichman.org/roguehistory.html

Copyright (c) 2015 Advances in Computer Science: an International Journal. All Rights Reserved.

[12] Nissan, Ephraim. *Computer Applications for Handling Legal Evidence, Police Investigation and Case Argumentation.* Netherlands: Springer, 2012.
[13] Mosunov, A et al., "Assured Supraliminal Steganography in Computer Games." *Lecture Notes in Computer Science* v. 8267 (2014): 245-259.
[14] Crawford, H and Aycock, J. "Supraliminal Audio Steganography: Audio Files Tricking Audiophiles." *Lecture Notes in Computer Science* v. 5806 (2009): 1-14.

Chance Gibbs received a B.S. degree in Computing Science from Sam Houston State University in 2015. His research interests include Steganography, Procedural Content Generation, Compiler Theory, and Game Programming and Design.

Dr. Narasimha Shashidhar received the B.E. degree in Electronics and Communication Engineering from The University of Madras in 2001, and the M.S. and Ph.D. degrees in Computer Science and Engineering from The University of Connecticut in 2004 and 2010, respectively. He is currently an Assistant Professor in the Department of Computer Science at Sam Houston State University, Huntsville, TX. His research interests include Cryptography, Information Hiding, Steganography, Electronic Voting and Security, Peer-to-Peer/Sensor Networks and Context-aware pervasive communication. He was a part of the Voting Technology and Research Center (VoTeR) at the University of Connecticut where he advised the State of CT on the security and deployment of electronic voting machines. He has over 25 conference/journal publications and also serves in the editorial advisory/review board and the Technical Program Committee (TPC) of a number of books, journals and conferences.

Copyright (c) 2015 Advances in Computer Science: an International Journal. All Rights Reserved.