5-2023

# StegoCraft: Information Hiding in Minecraft

Nathaniel Joseph Kleffner
*Trinity University*, nathanielkleff@gmail.com

# StegoCraft: Exploring Information Hiding in Minecraft

Nathan Kleffner

**Abstract**

In this thesis, we explore the use of Minecraft as a platform for steganography, which is the practice of hiding a secret message within another, outwardly plain message. Specifically, we investigate how different Minecraft systems, such as textures, in-game user interfaces, and buildings, can be used as a cover for hiding secret messages. To demonstrate these techniques, we created a custom adventure map that showcases various methods for distributing secret information within Minecraft. Our research highlights the potential of Minecraft as a powerful tool for steganography, offering a unique platform for securely exchanging secret messages in an environment where regular cryptographic methods may be prohibited. Through our investigation, we hope to highlight the growing amount of platforms that can be used to distribute information, and how they could be used to increase security and privacy.

# Acknowledgments

Thanks to Dr.Hibbs for his valuable advice and guidance. I would also like to thank Zach, Diego, Kionna, and Ben for their help with the custom map.

**StegoCraft: Exploring Information Hiding in Minecraft**

Nathan Kleffner

A departmental senior thesis submitted to the
Department of Computer Science at Trinity University
in partial fulfillment of the requirements for graduation
with departmental honors.

April 14, 2023

| | |
|---|---|
| Thesis Advisor | Department Chair |

Associate Vice President
for
Academic Affairs

# StegoCraft: Exploring Information Hiding in Minecraft

Nathan Kleffner

# Contents

# List of Figures

# Chapter 1

# Background and Motivation

## 1.1 Why hide information

Information hiding techniques have always been tied to the medium of their time. Long before the digital age, revolutionary soldiers used invisible ink to hide their movements from the British, and Sir Francis Bacon used special characters to encode messages all the way back in 1605 [2]. In the digital era, people wishing to obfuscate information have an abundance of mediums to take advantage of. From text files, digital images, and even DNA strands or video games [2], there has never been more ways to gain security through obscurity. There also may be more reasons to obscure information, whether nefarious or noble. With many governments and law enforcement agencies being more than capable of monitoring their citizens digital affairs, there may be cause for some individuals to want to hide their private information. Additionally, it is not uncommon for governments to ban forms of communication such as social media. From Turkey banning twitter during their election, to Pakistan banning youtube for blasphemous content [14].

The competition between governing agencies and those wishing to undermine their over-


Figure 1.1: Hong Kong Protest in Animal Crossing. By using the in-game screenshot tool, pro-democracy activists were able to protest through their Animal Crossing island when in person protests were prohibited. This picture was adapted from [4].

sight has led to the use of some technology in unexpected ways. More specifically, video games have become an interesting medium to hide information. We have already seen the use of these for a variety of purposes [7]. For example, in China, where protests related to Hong Kong are often dissolved or banned by the Chinese government, activists used *Animal Crossing: New Horizons* to organize and demonstrate their support for the independence of Hong Kong [4].

Animal Crossing is a game that allows you to customize your own island through resource collection. Once you are happy with the state of your island, Nintendo has added a feature which allows you to make a QR code to share your island. Activists have used this feature to share their designs on social media or other sites without being censored since, at a first glance, it just appears to be a picture of Animal Crossing.

## 1.2 What is Steganography

Cryptography is a technique that involves the transformation of plain-text messages into cipher-text to protect the information while transmitting it over public networks. The cipher-text can only be deciphered by its intended recipient, as it requires a unique key to decrypt the message [1]. Public key cryptography is one of the most popular forms of cryptography that uses a pair of keys, namely public and private keys. The sender uses the recipient's public key to encrypt the message, and upon receiving the message, the recipient decrypts it using their private key.

While cryptography focuses on obscuring the meaning of a secret message, steganography takes a different approach by concealing the very existence of the message, which in some cases may make it harder to detect. Steganography is a technique of hiding a secret message within a seemingly plain document such as an image, audio or text file. This enables the sender to communicate without arousing suspicion as to the existence of a hidden message. For instance, when Alice and Bob exchange secret information, it appears to an outsider that they are merely sharing a picture or a video. Modern steganography can be broadly classified into two categories: spatial and frequency domain. One popular method in the spatial domain is Least Significant Bit (LSB) steganography, where the least significant bit in the RGB values of an image is modified to encode the message, without visibly altering the original image [9]. On the other hand, in the frequency domain, techniques such as discrete cosine transform (DCT) can be utilized to encode secret information in the high frequency coefficients of an image [5].

Steganography provides several advantages over cryptography. One significant benefit is that steganography can evade detection by a third party, especially in situations where the sender expects to be monitored. By hiding secret information within apparently normal,

innocent media, steganography allows for secret communication without arousing suspicion. This can be particularly useful in circumstances where encryption may be prohibited. Additionally, the choice of medium used for steganography can also provide further security from potential adversaries, as we will explore in later sections and chapters.

## 1.3 Why hide information in video games

There are many reasons why video games are an excellent medium to distribute secret information. Primarily, their worldwide popularity allows people seeking to obfuscate information to hide in plain sight. A computer having a game installation would not sound any alarms to an adversary. Similarly, having a gaming console in the home should not raise any suspicion.

Additionally, given the ubiquity of custom game content (frequently called "mods" or modifications), a user having a version of a game slightly different from the original source would not sound any alarms either. Whether it's a content focused mod, small optimizations or texture changes, having a modded installation is incredibly common. This can reduce possibility of finding secret information by checking a file's contents for changes.

Even if a file was changed to hide information, it may be difficult for an investigator to pin down the source of the secret information simply because of how common large file sizes are in video games. Multiplayer games can easily take up tens of gigabytes on the disk, and some games, such as Call of Duty: Warzone, can take up to 250 gigabytes of space on their PC ports.

Further, even if changes were detected and located in a game's source files, it could be hard to understand what information they are hiding. A simple message could be incomprehensible within a game's source files, but easily spotted when rendered in a game's

terrain, overlay or dialogue [10].

## 1.4 Background and Related work

There has been an expanding amount of work in hiding information in video games, as well as the field of steganography as a whole. More specifically, there have been instances of people exploiting in-game overlay and modeling systems in games such as *World of Warcraft*, *Starcraft 2*, *Dota 2*, *Battlefield 3*, and *Garry's Mod* [7]. Many of these methods take advantage of tools and techniques used by the community surrounding the game to make custom content. These methods are broken down into one's which add-on to the base game, and one's which modify original game files. In their work with a variety of video games, Ebrahimi and Chen found that it was usually more difficult to detect add-on game content than directly changed source files [7]. In this thesis, all of the methods take advantage of already existing ways to add user-generated content. Much like the techniques done by Hale [10], which used the Hammer Editor to hide information in brush data in the Source Engine. A more technical approach can be seen in *StegoRogue* in which Shashidhar and Gibbs changed the dungeon and loot generation in-game in order to encode certain messages [8].

## 1.5 Choosing Minecraft

Minecraft is a sandbox game developed and maintained by Mojang Studios. The game was originally developed by Markus "Notch" Persson, who fully released the game in November of 2011. Since then, Minecraft has grown to become the best selling game of all time, with 238 million copies sold across every major video game platform. In 2014, Mojang was bought by Microsoft for $2.5 billion. Along with its high popularity, Minecraft has been praised

for its educational potential, and has been used to teach kids chemistry, computer-aided design, and computer science. Minecraft's main gamemode, "survival", involves exploring a blocky, practically infinite procedurally generated world made up of 63 distinct biomes including caves, forest, mountains, and extra dimensional spaces. In their exploration, the player may collect raw materials, build structures and custom terrain, and fight boss enemies. An alternate gamemode, dubbed "creative" mode, gives the player an infinite amount of blocks and the ability to fly, allowing the player to build without the limitations of resource collection. Beyond the game's core game modes, over the decade that the game has been available, the player base has created many custom gamemodes. These include PvP minigames, games dedicated to players fighting to the death; adventure maps, single player or co-op experiences where players play through a custom, linear world; and mods/modpacks, allowing players to add endless amounts of new items, blocks, mobs, and mechanics to their Minecraft world. Throughout Minecraft's lifespan, user generated content has been one of the main focuses of the playerbase. There are servers, which host tens of thousands of active players, solely dedicated to minigames.

When considering the features that make video games a strong choice for hiding information, it is easy to see why Minecraft in particular would be an ideal candidate. Since its release, Minecraft has accumulated 141 million players worldwide. With this broad of a player base, having it installed on a computer or console should not raise any suspicion. Minecraft also has an extremely active modding community. Mods like Optifine, a small optimization mod allowing players with outdated setups to continue playing, have thousands of downloads. Mod creation tools like ForgeMC or Fabric allow anyone to create their own modded version of the game.

This thesis will evaluate Minecraft's value as a cover. We will explore different ways in which two people, Alice and Bob, could use Minecraft's systems to keep information hidden

from some adversary. The methods explored in this will range in difficulty and security, as well as the level of technical ability needed to implement the methods. Additionally, this thesis will examine different ways to then distribute these steganographic technique; from servers to world saves.

# Chapter 2

# Stegocraft possibility space

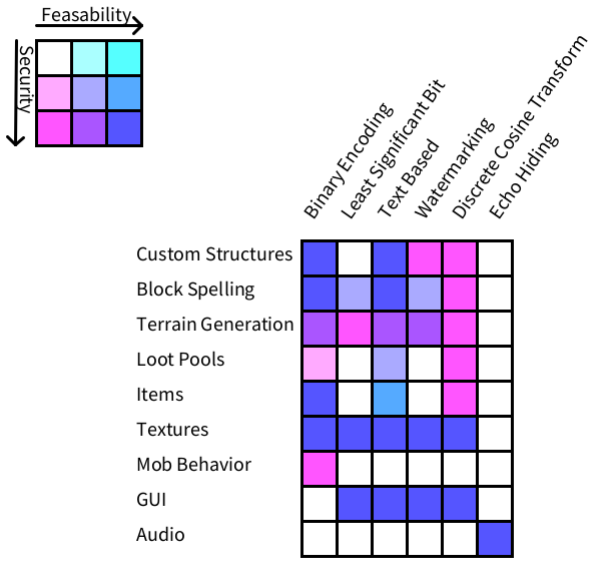


Figure 2.1: Possibility space. This is a visualization different ways of combining steganographic methods and Minecraft's in-game systems. This visualization makes an assumption that the user is not necessarily an expert in programming or steganographic techniques.

## 2.1 Possibility space

Figure 2.1 lays out a speculative possibility space for potential covers in Minecraft, as well as the steganographic technique that could be used alongside it. This graph is making an assumption that the person hiding the information does not have an extensive background in programming or cybersecurity.

The most notable features are the patterns for Textures and GUIs. Texture and GUI are the most secure and practical sources for steganographic techniques, mostly because of the amount of work that has been done in hiding messages within digital images [13] [14] [10]. As discussed in chapter 3, Minecraft's texture pack and UIs provide plenty of opportunity for using any of these techniques. Another common pattern is DCT being less feasible. This is because DCT can be hard to implement, considering the difficulty of the mathematics behind it. However, if implemented correctly, DCT in Minecraft could provide a very secure medium for communication. Other in-game systems, such as building, mob behavior, or loot pools offer a variety of difficulty and security. As shown in chapter 3, many of these systems can be implemented with very little programming experience. Even with this speculative possibility space, there is still question of implementation. In the following sections, we will discuss some of the tools that could be used to implement steganographic methods.

## 2.2 To Mod or not to Mod

Modifications, or mods, are a way for players to create their own changes and additions to the game. These mods can range from small quality of life changes, like updating the UI or increasing performance, to adding new items or enemies to the game. Usually, mods are made using unofficial tools and libraries. The two main tools used for modding in Minecraft

are FabricMC and Forge. Both of these are Java libraries that interface with Minecraft's source code and allow the users to modify almost every one of the games systems. There are plenty of online tutorials for getting started with either of these frameworks; all you need is some base level of Java knowledge, and you create your own mods.

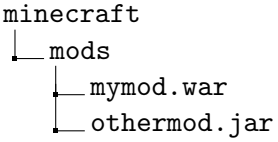


Figure 2.2: Structure of .minecraft directory. In order to add a mod to the modded Minecraft launcher, a jar file can be added to the `mod` subfolder.

The ability to openly mod Minecraft opens many doors for hiding information. In fact, changing some of the games system's, such as terrain generation, requires modding. Modding essentially raises the ceiling for the security of information hiding, but it can also introduce some roadblocks.

Chiefly, creating and installing mods requires at least a small amount of outside knowledge, as it goes beyond what the original makers of the game intended. Some games make this easier than others. Valve's Left 4 Dead 2, for example, has a section in the main menu dedicated to mod management. The Steam Workshop, managed by Valve, allows for players to easily share and update their mods to a centralized source. Managing mods in Minecraft is not particularly challenging, but keeping track of different versions of modded clients and servers can be a technical issue. This is especially true because mods are not necessarily endorsed by Mojang. Additionally, the use of mods may raise some concerns: if the goal is to send a secret message, why not simply send the message? Ultimately, you will need to communicate the mod executable to someone or provide a way to download it, so why not just send the secret message itself? As pointed out in section 1.2, sending a Minecraft mod's jar file is much less suspicious than sending an encrypted message.


Figure 2.3: Steam Workshop page, where users can upload custom mods, maps, and textures through first party, Valve sponsored software. This can be found in the Steam desktop app made by Valve Software.

## 2.3 Commands and Command Blocks

Vanilla Minecraft exposes some internal interfaces to the player to access through chat called commands. These commands range from changing the weather and day cycle, to placing animals, blocks, or particles. While the player can execute these commands by simply typing '/' in the chat, in July of 2012, Mojang added a new way to execute these commands: Command Blocks (Figure 2.4). Command Blocks also have a wider variety of commands and syntax, raising the bar for more complex information hiding systems without having to install any mods.

Similar to chat logs, Minecraft will store executed commands in a log. In Figure 3.2, you can see that switching the gamemode was logged. This can be avoided, however, by


Figure 2.4: Command Block interface. Any command accessible to the admin of a server can use the command block to automatically execute the command. The user can set when the command will be executed, whether is executes all of the time, or needs some sort of redstone pulse to be activated. The Command Block UI will also tell the user if the syntax of what they have entered is incorrect and how to fix it.

```
/gamerule commandBlockOutput false
/gamerule logAdminCommands false
```

Figure 2.5: Commands for disabling logging of commands. There commands will make executed commands, either through chat or through Command Blocks no longer visibile in the world save's `log` file

changing the world save's gamerules, as seen in Figure 2.5.

While these commands open many avenues for information hiding in Minecraft, they have a large roadblock before them: administration permissions. User's without admin permissions or "cheats" (as they are called my Mojang) are not allowed to execute most commands. In fact, on some servers, the attempt to execute a forbidden command may be logged. Ideally, the use of commands would be used on a single player world that you would then share with an accomplice, or would be used on a server that you host or own yourself.

Another issue with using commands over mods is that as complexity of your method increases, the practically of doing it with commands decreases. For example, commands do

not have any way to keep track of variables or lists. The work around for this, which the custom map community uses for creating minigames are scoreboards and NBT tags.

```
/scoreboard <players> <set|add|subtract> <entity> <objective> <value>
                        score_objective = <value>
          /data modify (block <pos>|entity <target>) <path>
```

Figure 2.6: Commands for scoreboards and NBT modification. The scoreboard command can be used to keep track of a variety of events, and can be shown to specific players based on their assigned team.

The scoreboard is an in-game way to assign players different scores depending on certain objectives. Usually these objectives are in-game tasks such as blocks mined, biomes visited, or deaths. Using commands, however, you could set the objectives and a players score to any 32-bit signed integer. Built in to Minecraft are also "teams", of which there are 16. Using the `/scoreboard <...> setDisplay:<team>` command, you can control which players can see the scoreboard.

NBT tags are used to give more specific information to entities and items summoned through commands. Clever use of these systems opens up alice and bob to a wide variety of information hiding methods, at the cost of having to learn the complex commands and being limited by setting, adding, subtracting, and simple lists.


Figure 2.7: Scoreboard example. The deaths counter is an arbitrary label. The name of the scoreboard can be any string. The number displayed can be anything that can be represented by a signed 32 bit integer. Additionally, the scoreboard can be targeted at specific players, so only certain people can see it.

# Chapter 3

# Proofs of Concept

## 3.1 Introduction

The following chapter is dedicated to exploring methods for hiding information. Each of these methods is meant to convey a Minecraft's system's use as a cover, rather than an in-depth discussion of the steganographic technique. While each of these implementations may be simple, they can also be easily iterated upon to meet any required level of security and complexity. The methods will vary in the amount of programming knowledge needed, and a good amount of them will require the use of commands (discussed in 2.3). This means that some methods will work better in different settings. For example, it is impossible to use a method which requires commands on a server without admin permissions. It is also important to discuss ways in which these methods could be tampered with and detected, which will be discussed in the analysis section of each method.

## 3.2 Why using chat is a bad idea

### 3.2.1 Method

The most obvious way of communication might be the one built into the game; chat messaging. If Alice and Bob wanted to communicate secret information, all they would have to do join the same Minecraft server and start messaging in chat. They could even use the `whisper` command so that other player's can't see what they're saying.


Figure 3.1: In-game chat message. Every Minecraft server, and even single-player world has a chat which get's stored in the world save's `log` folder.

### 3.2.2 Analysis

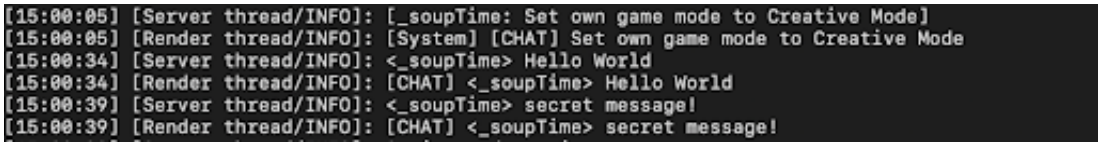


Figure 3.2: Log file from world save. Every chat message is logged with a time-stamp and user. The log also keep's track of commands, unless the gamerule "logAdminCommands" is set to false. If the user does not have access to console commands, this eliminates the possibility of communicating secretly through chat itself.

Whether this message is encrypted or not, this is by far the easiest method of information to detect. All any potential investigator would have to do is check the servers chat logs, which are kept on every Minecraft world's save file. So, the vulnerability of any method using chat would completely rely on the ability to hide a message in plaintext. The only benefit this method provides is the same benefit every method has: it is being done in Minecraft.

As discussed in the first chapter, the idea that it is being done in Minecraft, rather than a more common form of communication, may provide enough cover to hide in plain-sight. As we will see in later sections, there may be more effective ways to secretly communicate.

## 3.3 Texture and Data Packs

### 3.3.1 Method

Resource Packs, called Texture Packs before Minecraft v.1.6.1, are a way for players to customize their visual and auditory experience. This includes textures, models, in game overlays, in-game text, sounds, and music. Resource packs are usually stored as sub-folders or .zip files in the games `resourcepacks` folder. Making a custom resource pack involves unzipping the right version's jar file, which can be found in the `.minecraft` directory.

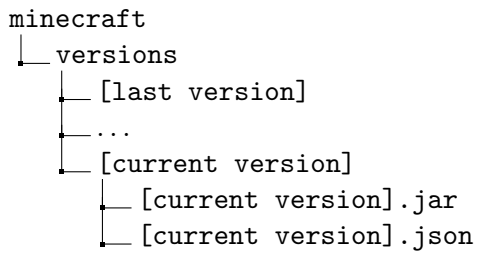


Figure 3.3: Structure of .minecraft folder. This shows how to access the .jar files for different versions of the game. Unzipping the jar file gives access to the default texture pack, as well as other resources including audio files and configurations.

After obtaining the textur's folder, any texture can be edited with any image editor and dropped it back into the texture pack. Sounds and Music require knowledge of how to modify JSON. Since the resource packs are stored as .zip files, there will be no loss of hidden information. Having the ability to modify the game's texture and sound files, and, even better, in an exceedingly straightforward way, opens the door to many possibil-

ities in the world of image and audio steganography. For example: Least Significant Bit (LSB) Steganography. If Alice and Bob wanted to distribute information through LSB on a Minecraft texture pack, all they would have to do is communicate the texture file which was changed. Alice can create the texture pack, as explained above, use LSB on whichever texture file they have agreed on with Bob can decode it. There are even online tools for encoding/decoding .png files [source:`https://stylesuxx.github.io/steganography/`]

There are two main methods to then distribute the texture pack: directly send the texture pack or make it a servers default. The second option is easy enough. Simply setting the `resource-pack=` property in the `server.properties` file will prompt any user who joins the world to download the texture pack, and will automatically re-render the world once the instillation is finished. While the second option may seem risky, we must remember that this secret message is still hidden behind a Minecraft texture pack: something that would normally warrant no suspicion or further investigation.

### 3.3.2 Analysis

Even without Alice and Bob having knowledge of more sophisticated methods, an adversary may have a difficult time finding their secret message. One of the main reasons for this is the difference between textures as they are stored in the game's source, and how those textures are rendered in-game. As demonstrated in Figure's 3.4 and 3.5, certain aspects of the source file are not rendered on the in-game model, this may make it difficult to tell which textures could have been tampered with, for nefarious purposes, simply by looking at them in-game. This, of course, would be completely dependent on the steganographic method Alice and Bob employed. If they used LSB steganography, for example, then an adversary with access to the texture pack should not be able to tell the difference while in-game.

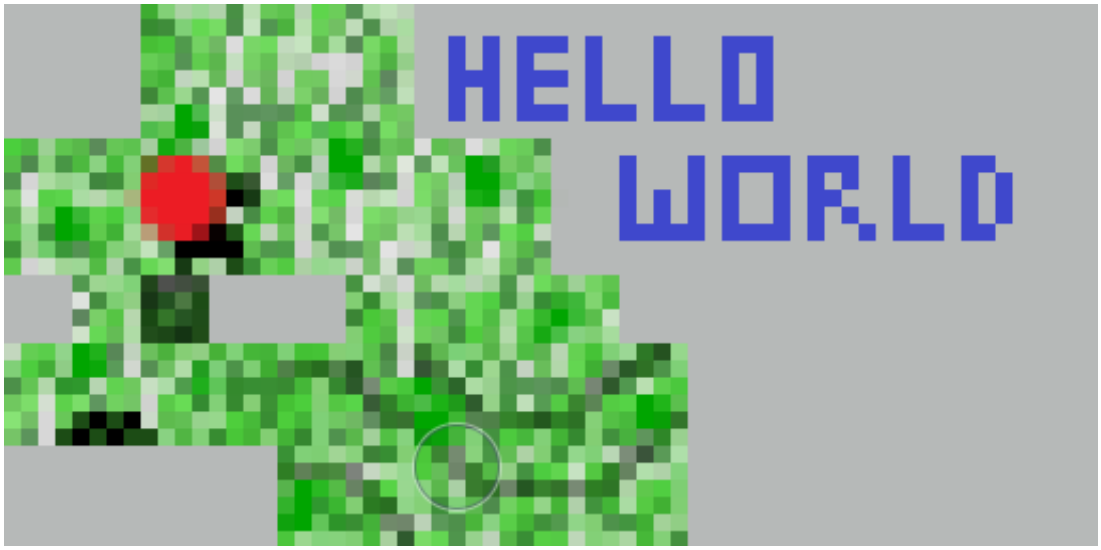


Figure 3.4: Edited creeper.png. The default texture can be obtained by unzipping the minecraft.jar file. The texture files for each 3D model will have extra, blank space because of the method Mojang uses for rendering the models. This could allow for embedding.

This may lead us to believe that it would be easier to always look through each of the games source texture files. However, the default Minecraft texture pack has just under 2,000 .png files, along with hundreds of JSON files that underly the game's animations and sounds. This would take a substantial amount of time and resources to look through, especially if Alice and Bob employ the use of a more advanced steganographic method [9] [13] [5].

Since texture packs are stored locally on the user's machine, it would be hardly possible for an adversary to tamper with such that Bob would not be able to view the message.

While the simplicity of this method lends itself towards accessibility to people of all technical skill levels, it may also be it's biggest downside, as image-steganography is one of the most studied and well known methods of steganography. As we will discuss later in section 3.7, combining an edited texture pack with other methods may bring us many more

Figure 3.5: In game rendering of Figure 3.4. Notice how there is no in-game rendering of the message “Hello World”. This is because the blank space is not rendered in-game.

layers of obfuscation.

## 3.4 Villager Trading and Other GUI’s

### 3.4.1 Method

Minecraft has a wide variety of in game overlays and UIs. Most of them are associated with a crafting mechanism or trading. Some of the UIs can also be edited by editing the default texture pack, which we disuccsed in section 3.3. Other UIs can be manipulated using in-game commands, as seen in Figures 2.7 and 2.7. The most complicated of these UIs is related to Villager trading.

Villager trading was added in Minecraft 1.3.0 (Java Edition). Interacting with a villager will bring up the trading menu, in which the player can drag and drop their items to obtain


Figure 3.6: Hiding a message in a bossbar. This uses the `bossbar` command, which allows the player to set a name for the bossbar, how many hit points the bar has, as well as which players can see it.

Figure 3.7: Villagers with different careers. Fishermen (left) and Librarian (right)

goods. Command blocks, discussed above in section 2.4, can allow the player to customize their own villager trades. In fact, there are online tools [6] which allow the user to enter in the desired trades and receive a command that can be executed through a command block (Figure 3.8). To communicate a secret message, Alice could execute this command and simply have Bob look at it's trade to see the message. There are, as is the case with every method discussed, nearly endless variations. The variation seen in Figure 3.10 simply does an alpha-numeric substitution to encode the message "Hello World".

There are, however, some complexities to consider when using this method. The first being the villagers profession. As seen in Figure 3.7, a villager's profession will change their appearance. More importantly, it will change the items a villager will trade with. For example, a Fishermen villager can trade with fish and string, but not with books or potions. This means that if Alice wishes to not be detected, they would have to stick with the type's of items a villager can trade, and tell Bob to only trade with a Fishermen.

```
/summon villager ~ ~1 ~{
   "VillagerData":{
      "profession":"farmer",
      "level":2,
      "type":"plains"
   },
   "PersistenceRequired":1,
   "Offers":{
      "Recipes":[
         {
            "buy":{
               "id":"emerald",
               "Count":8
            },
            "sell":{
               "id":"coal",
               "Count":5
            },
            "rewardExp":0b,
            "maxUses":1
         },
         {
            "buy":{
               "id":"emerald",
               "Count":12
            },
            "buyB":{
               "id":"coal",
               "Count":12
            },
            "sell":{
               "id":"diamond",
               "Count":15
            },
            "rewardExp":0b,
            "maxUses":1
         }
      ]
   }
}
```

Figure 3.8: Command to summon a villager with trades that encodes "Hello". This is made using the DigiMinecraft website, which allows the user to select the trades they want with a simple UI, and outputs the desired command to be used in a command block [6].

A villager also has a "Career Level". Each time a player trades with a villager, their career levels up, of which there are five (Novice, Apprentice, Journeymen, Expert, Master). Each level up allows the villager to display two more trades to the player. If there are more than two trades in it's pool, than a random new trade may be chosen. If Alice did not want their message to be revealed in a random order, then they could use the NBT tags discussed in section 2.4 to append to the villager's trades (Figure 3.9).

These complexities in the Villager trading system may also provide more complex ways

to hide information. For example, a villager has a maximum number of times they can do a trade within a day (`maxUses` field in Figure 3.8). This has a maximum of 9999999. This could be used to only reveal some letters after a trade has been "maxed out".

```
/data modify entity @e[type=minecraft:villager,limit=1] Offers.Recipes
                                append <trade>
```

Figure 3.9: Command for appending a villager trade. This could be used to append more trades after the initial creation of the villager. This could also be modified to remove trades, which could be useful for getting rid of secret information.


Figure 3.10: Villager Trade encoding "Hello World". This Villager was spawned in using the command in Figure 3.8. It encodes the words by doing a simple alpha-numeric substitution.

### 3.4.2 Analysis

It initially seems that there could be two main faults of using villager trading for hiding information: trading limitations and the villager entity. Luckily (or unluckily, for the adversary), there is virtually no way to tell if an entity was spawned using the summon command or was spawned in naturally. The only way would be to see the use of a summon command, which, as seen in section 2.4, could be easily worked around.

The trades, however, may pose problems to anyone trying to obfuscate information. As discussed in the previous subsection, the complexity of the villager trading system can bring advantages and disadvantages. Mainly, limitations to what the villagers can trade and in what order, at least in vanilla Minecraft. Additionally more layers of complexity necessitates more prior communication between Alice and Bob when the secret message needs to be longer.

However, these limitations may once again also have some hidden advantages. The first being that, if implemented correctly, it would be very hard for an adversary to tell the difference between Vanilla trades and generated ones. This is because of the semi-randomness that is used by Minecraft to produce the initial trade offers and subsequent trades. This "economy" fluctuates prices based on factors such as demand and personal discounts and fines depending on how much a certain player has interacted with the villager. This means that if someone were to come along and trade with the villager Alice and Bob were using as a cover, the trades would change for that person, but not Alice, Bob, or anyone who has not traded with it. This makes it very difficult to for someone to come by and tamper with Alice's secret message.

## 3.5 Loot and Items

### 3.5.1 Method

Items, or "Loot", provide many mediums for distributing information, both complex and more simple. Loot is randomly generated, and can be found in a variety of situations such as in dungeons chests, dropped from mined blocks, dropped from slain mobs, and dropped from fishing. Loot tables, JSON files stored in the source code, determine the type of loot that can be found in these situations. These items are then stored in either the players inventory, or various storage items the player has access to such as chests or barrels. Both the generation and storage of items can be used to communicate information with a wide range of complexity.

The first method is exceedingly easy, and only requires a small bit of coding knowledge. Usually, in order to change what items can drop from an event, Alice would have to change the blocks loot table. However, that could prove to be difficult, as there are limited options into how that loot is distributed and placed. A much easier way could be to just spawn an entity after a player does a certain event. Essentially, this simple mod initializes an array of numbers, which would be a secret message. Each time the player breaks an arbitrarily chosen block, an item, in this case, a stick, will drop in a stack the size of the next number in the secret message array (Figure 3.11). In order to communicate their message, Alice would tell Bob how many oak logs to break, and Bob would count the number of sticks dropped each time to find the secret message. Another method could borrow some ideas from StegoRogue [8] by storing certain items in a chest to communicate a message. To replicate Stegorogue, this could be done by modifying the loot pools of the game. This method would be preferred if Alice wanted the loot to appear randomly generated in dungeons throughout the world. However, this would require Bob to search through the world until eventually finding the

```
public class ModLoot {

    static int[] secret = new int[]{1,2,3,4};
    static int i = 0;
    private static int getNext(){
        int ret = secret[i];
        i++;
        return ret;
    }
    public static void registerLoot()
    {
        PlayerBlockBreakEvents.AFTER.register((world, player, pos, state, entity) ->
        {
                        Block block = state.getBlock();
                        if(block.equals(Blocks.OAK_LOG)){
                                ItemStack stack = new ItemStack(Items.STICK,getNext());
                                ItemEntity itemEntity = new ItemEntity(player.world, pos.getX(), pos.getY(), pos.getZ(), stack);
                                player.world.spawnEntity(itemEntity);
                        }

        });
    }
}
```

Figure 3.11: Code implementing a message through sigple item drops. This uses the Forge mod making API, and only needs this class file and a main file to run. Theoretically, this could be made with very little coding knowledge.


Figure 3.12: Encoding the message “Hello World” items in a chest. Similar to Figure 3.10, this uses a simple alpha-numeric.

type of dungeon that is affected by Alice’s loot pool change. This would be significantly more difficult to locate in Minecraft because of the size of the world. Although this level of complexity could potentially make the message more secure, it may be too difficult to implement and actually communicate with. A much easier version to implement would be placing items in chests manually. For example, Alice could place items in a chest or barrels using some sort of encoding, and all Bob would have to do is check in the chest and view the message. This method has an endless amount of variations such as using certain items to represent a word, or using the number of items to encode a number, such as in Figure 3.12, or even having the first letter of an item represent a character (Figure 3.13).


Figure 3.13: Encoding the message “Attack at Dawn” using items in a chest. This uses the first letter in the name of each item to encode a letter, i.e., “Apple” –¿ ‘A‘, “Torch” –¿ ‘T’, etc.

### 3.5.2 Analysis

Much like other methods described in previous sections, these methods benefit from their ease of use. The second variant, involving putting items in chests, involves very little technical ability and very little prior communication between Alice and Bob. It also has a massive number of messages it could communicate. An encoding a message based on the number of items, as seen in Figure 3.12, has $64^{54}$ possible messages that could be encoded in a single chest. Additionally, since chests are the most common ways players store their items in survival Minecraft, there would be very little reason for an adversary to be suspicious. Even if they were suspicious, it may be difficult for an adversary to intercept or discern possible communication from regular game play. Even in the first example, involving dropping a different number of items each time a player breaks a block, it is easy

to hide in plain sight, as there are many mods which change the loot pools of items for the sake of convenience.

A potential downside of this method, however, is how easy it is to interfered with. All someone would have to do is open the chest and move the items around, or add their own items. This means this method could be 'riskier' on more populated servers. s

## 3.6 Generated Structures and Player Buildings

### 3.6.1 Methods

Building and placing blocks is one of the core features of Minecraft, and is a feature usually not found in other shooter or MMO games. In Minecraft, in both survival and creative mode, there are little limitations on how blocks can be placed. There are only seven blocks out of the seven hundred and two blocks (as of Minecraft 1.19) that have any sort of physics applied to them. This gives very little physical limitations to what can be built, and, by extension, to what can be communicated using buildings. Naturally generated structures, however, are more rigid. Minecraft generates it's terrain through the use of *chunks*, which are $16 \times 16 \times 25$ size blocks. Structures are generated after the terrain of a certain chunk has been generated, and use a variety of randomness and predefined rules. In order to change any of the generations of these structures, say to do StegoRogue style stegonagraphy using dungeons, Alice could mod the game. This would involve defining the placement of each of the of the blocks in the custom structure, and how it would be generated with other structures.

There are ways, however, to place or create custom, generated structures in vanilla Minecraft. Features such place command and the jigsaw block allow custom built buildings to be used for generation. The place command (seen in Figure 3.14) is better suited for our

```
/place template <template> [<pos>] [<rotation>]
```

Figure 3.14: Template for the `place` command. The `<template>` can be a wide variety of built-in Minecraft structures. This includes every type of villager house, dungeons, igloos, etc.

Figure 3.15: Villager buildings placed to spell out “Hi”. This uses the place command, as seen in Figure 3.14, to orient default Villager buildings to spell out a word. Notice that it would only be legible if the user had a birds-eye view of the Village, or if the user had a large enough map to see the entire message.

purposes, since it allows for the manual placement of structures, while the jigsaw block is more meant for stitching existing structures together procedurally.

One straightforward method to encode information would be for Alice to use the ”place” command to construct specific structures that spell out certain words, as shown in Figure 3.15. Bob could easily decode this message by observing the structures from a vantage point. To further aid in decoding, Bob could utilize the ”map” item available in vanilla Minecraft, which provides a bird’s-eye view of the terrain.

An even simpler version of this would not require the use of commands or any previous


Figure 3.16: Red Wool blocks spelling out "Attack at Dawn". This spelling spreads the message across multiple chunks, which would make it more difficult to spot within the world save files. This image is drawn in an superflat world , which has no terrain generation. In a normal world, however, this would be hidden by terrain.

coding knowledge: simply spelling out words with blocks and showing an accomplice (as seen in Figure 3.16). For an extra layer of security, Alice and Bob could erase the blocks after revealing the secret. This is by far the easiest method of communication, being only one step above typing the message in chat in terms of difficulty.

### 3.6.2 Analysis

Despite the simplicity of the above methods, it would be extremely difficult for an adversary to discover Alice and Bob's communication. Suppose that an adversary is an admin on the server, and suspected Alice and Bob of some sort of suspicious behavior. In order to find if communication even occurred, they would had to have gotten lucky enough for the world to save at the moment the message was written out in blocks. If they happened to have that,

and they had a reason to believe that they knew which world save it was, they would then need to either look through the save files of the world to find it or load up the previous save and look through the generated world. Either way, this would be an excruciatingly tedious process.

Suppose that the adversary first tried to fly through the world to try and find the message Bob and Alice wrote. The amount of chunks that they would have to look through is massive, somewhere in the neighborhood of fourteen trillion at maximum world size. This is not accounting for all of the terrain generation that could and most likely would obscure the secret message. It is easy to see that simply flying through the rendered world will not suffice. Next, the adversary may try their chance at looking though the save files. Minecraft will store the world's chunks as tags in a regional Anvil File. This Anvil file is in NBT format, which keeps track of different block states in the given chunk. In order to find any potential hidden messages, the adversary would have to create a program which can visualize and detect text given this NBT file format. It may even have to do detection across NBT files, since a message may have been split up into multiple chunks. More importantly, it would have to do this for every file stored in that world save. In the most charitable case for the adversary, they know a general region in which the message may have been written. This could potentially be obtained if Alice and Bob stayed in the same area as the message was written before they left the game. This is because Minecraft will store the players coordinate position as they leave, so that they can spawn back in the same position. In this best case scenario, the adversary also has some sort of bot to look for text written out in blocks. However, even with state of the art AI powered recognition software, the adversary would only be able to successfully detect Minecraft in-game renderings at a rate of about 5.98% (or at a 17.33% rate when retrained specifically for Minecraft). [14]. All of this work is only for the simplest method deployed by Alice and Bob: simply going into the

server, writing out some letters, and then removing it when they are done. This does not account for any advanced steganographic technique.

The downsides of this method may be in it's simplicity and flexibility. More specifically, it may not be hard for an adversary to both read and change the message. This, however, requires the adversary to first find the message. As we discussed above, this could be somewhat avoided by erasing the message after it has been viewed. Another pitfall this method's simplicity brings is it's "broadcastability". If an adversary were watching Alice and Bob communicate (for example, they stealthily followed Alice and Bob to the location), then it would be very easy for them to see and parse. In the next section, we will discuss how combining building with other methods may make it even more robust and secure.

## 3.7 Combination of Methods

While each of these methods can be effective on their own, combining them offers an even stronger layer of obscurity. For example, Alice and Bob could communicate using carefully placed blocks along with a custom texture pack. As shown in Figure 3.17 and 3.18, Alice could create a pattern using a wide array of blocks. They would then create a custom texture pack which would change the colors of some of those blocks, such that putting on a texture pack would reveal a message; almost like a 3D glasses effect. This method could also take advantage of the fact that some blocks in the default texture pack have the same texture, such as the *Infested* blocks.

If Alice did not want to manually change the texture pack, they could employ the use of the `/fillbiome` command, as seen in Figure 3.19. Changing the biome changes the shading of some foliage blocks, such as grass and leaves. One advantage of this is that it could not be spotted on a map. This is because the map item does not include any shading; it colors


Figure 3.17: Encoding "Hello World" in a wall of stone bricks by changing the texture pack. This takes advantage of the fact that there are *Infested* Stone Bricks, which have a different block ID, but have the same texture in default Minecraft. This allows use to make a simple wall (left), which encodes a message when using the proper texture pack (right).

all grass the same color. This would provide an extra layer of security.

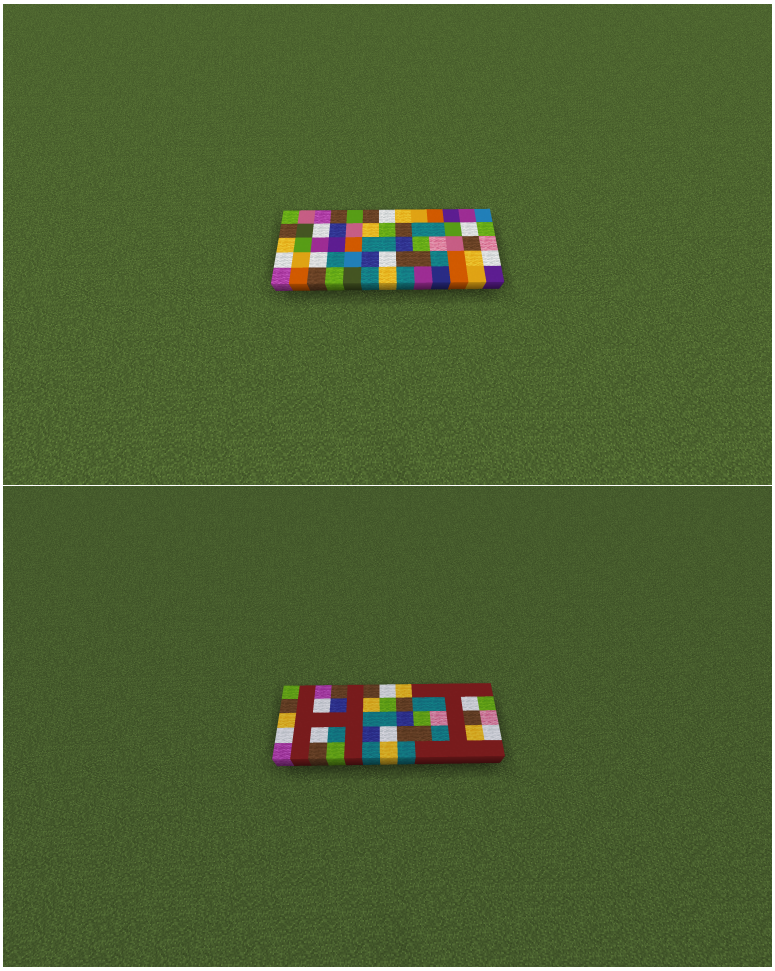

Figure 3.18: Encoding "Hi" in random blocks by changing the texture of certain blocks. Unlike Figure 3.17, this hides the message in a random array of blocks. This is to show that the *Infested* blocks don't neccessarily need to be used.

```
/fillbiome [<pos1>] [<pos2>] <minecraft:biome>
```

Figure 3.19: Template for the `fillbiome` command. This allows the user to change the biome of a specified area, which will change the shading done on the fiolage, i.e., grass and leaves.

Figure 3.20: Encoding "Hi" using the `fillbiome` command. This changes the shading of the grass. Much like the village in Figure 3.15, this could only be seen with a birds-eye view. However, unlike Figure 3.15, it can't be seen using a map because maps only render one color for grass.

# Chapter 4

# Adventure Maps and Servers

## 4.1 Sharing and Distribution

In the previous section, we explored different ways to hide information from potential adversary's both in game and through the game's source files. Once these methods have been implemented, there are still concerns about how the method might be shown to an accomplice. The main platforms for distributions are servers and world saves (along with any texture pack mod that may come along with it).

Issues mainly arise when trying to hide information on a server where there is no access to admin commands as this severely limits the types of methods that can be used. There may also be worry about being "caught in the act", since some server plugins that have ways of keeping track of suspicious behavior. While there are many methods server hosts have developed to detect cheating and harassment, there are little systems in place to automatically detect the distribution of confidential information. Even if server hosts did have these systems in place, it is very difficult to discern information hiding through structures and building from regular game play [14].

Sharing world saves, texture packs, and mods all of the benefit of being obscured by volume of Minecraft content that already exists. As has been discussed in previous chapters, there is nothing inherently suspicious about sending a friend a Minecraft world save or custom texture pack. There are also many forums online for display custom maps and texture packs, of which there is very little regulation or oversight.

## 4.2 Information Hiding within Custom Maps


Figure 4.1: Entrance to the custom Adventure Map. This uses a command block to give the player a welcome message. It reads: “Welcome to Faepond! While you’re here, make sure to check out the abandoned witch house, Coyote’s Gift Shop, and The Capitol Library!”. This message was emitted using a Command Block.

Custom maps used to showcase builds or take the user through some sort of story, often called ’Adventure Maps’, are shared on websites and have tens of thousands of downloads. Usually, this sharing process takes a little as creating an account and uploading a ZIP file

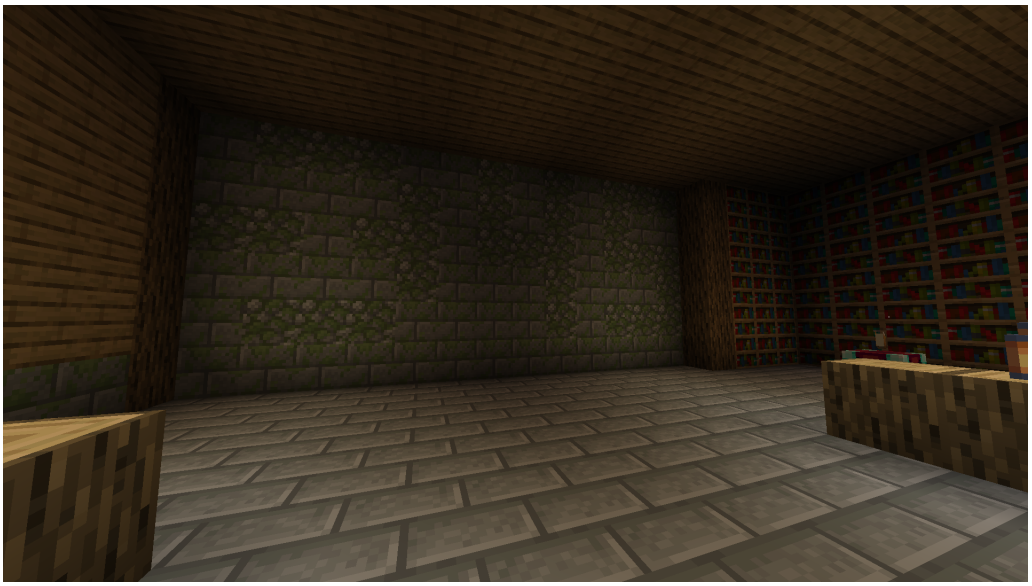

Figure 4.2: This is the first coordinate for the secret message of the adventure map. This uses the difference in textures to encode a number. This number can only be seen if using an altered texture pack which changes the appearence of infested stone bricks

of your world.

In order to put some of these proofs of concepts to use, I created an adventure map, and hid information in it.

I specifically chose to do this in Vanilla Minecraft to prove that there is very little coding knowledge required to hide information in a Minecraft world. As part of this experiment, the most technical I got was making command blocks to provide some sort of context for what the world is supposed to be, as seen in Figure 4.1. I also used commands to help build some of the structures in the world, simply to speed up the process. The most complicated command was `fill` command, which essentially makes placing walls a lot faster. While many custom maps have a narrative or progression, some also allow players to simply explore the creation. For my custom map, I chose to make a swamp city the player can

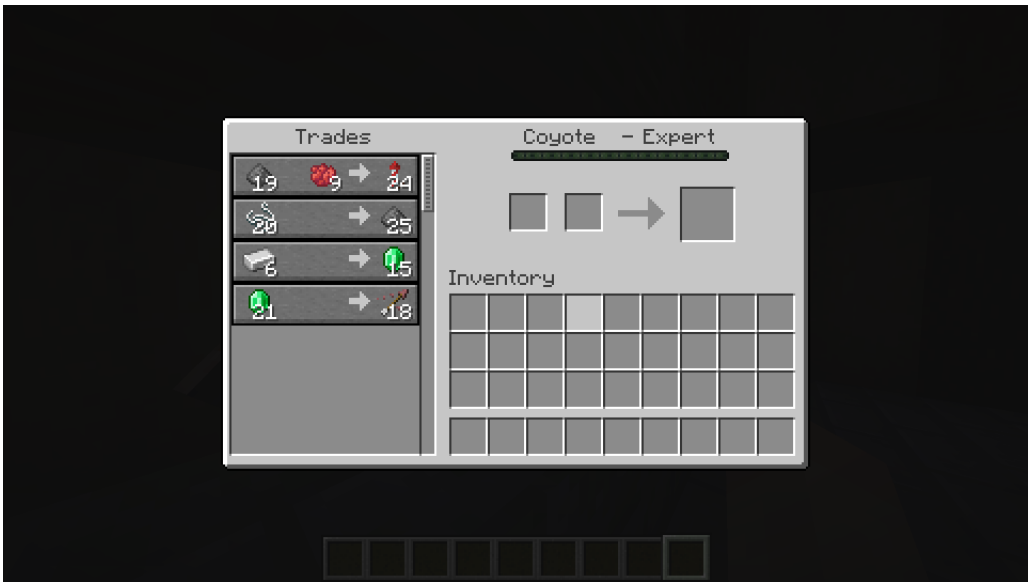


Figure 4.3: Encoding of the message “64”. This uses the villager trades method, using an alpha-numeric substitution. This number is the $y$-coordinate of the secret message.

explore. There are some landmarks in the map, each of which uses a method from Chapter 3 to encode a part of a coordinate. Once the use has each part of the coordinate, they can find a book which has a link to this thesis in PDF format.


Figure 4.4: Encoding of the message “three hundred and eleven”. This uses a simple alphanumeric encoding. This is the $z$–coordinate of the secret message included in the adventure map.

# Chapter 5

# Conclusion and Future Works

## 5.1 Conclusion

Minecraft has proven to be a unique platform for the distribution of secret information through steganography. The techniques discussed in this paper, such as using texture packs, buildings, and villager trading, provide creative and effective ways to conceal information within the game environment. By leveraging the game's expansive world and endless content updates, users can hide in plain sight in ways that would be exceedingly difficult to catch.

The underlying message behind all of these methods is that if anyone wanted to use Minecraft as a platform for secret messaging, there would not be much in the way of stopping them. This is mostly due to Minecraft's widespread popularity, which would give it a barrier from suspicion from a typical adversary.

## 5.2 Future Works

This paper provides starting points and ideas for the deep and nearly infinite possibilities for hiding information in Minecraft. As discussed in the possibility spaces section, there is a system of using the scoreboard and other commands that could be used to make games within Minecraft, which is a method of mod making I did not touch on much in the proof of concepts section. When considering more practical examples, I think the methods that have the most potential are terrain and loot generation. This is mostly because there is already a substantial amount of work on this with StegoRogue, and you could simply implement this style of Steganography in Minecraft. Finally, it is worth noting that, at least at the time of writing this, Minecraft is continuously updated. This would potentially add more and more ways to hide information onto the possibility space. In Minecraft 1.20, chiseled bookshelves were introduced, which allows players to place the book item into a bookshelf. This could be use to do some sort of ternary encoding. Even without looking forward to future update, there are still more systems that could be explored. Some of the most promising are Banners and Jigsaw blocks, which are a level of complexity above what was explored in this paper, but have interesting properties which could make them a good platform for information hiding. More work could also be done to make tools that can automate a lot of the methods discussed. For example, there exists a website which allows you to enter in your desired trades, and it spits out a list of commands to spawn a villager with the trades you want. Similar things could be done to automate the process of spelling out letters with blocks, for example, or even filling in a biome. One could imagine a tool where you specify the width and height of a 'text box' in terms of coordinates in your Minecraft world, and it spits out a series of commands which will spell out the words.

# Bibliography


[1] G. Abboud, J. Marean, and R. V. Yampolskiy, "Steganography and visual cryptography in computer forensics," in *2010 Fifth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*, 2010, pp. 25–32.

[2] R. F. Alan Siper and C. Lombardo, "The rise of steganography," in *Proceedings of Student/Faculty Research Day.* Pace University, 2005.

[3] R. Anderson and F. Petitcolas, "On the limits of steganography," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 474–481, 12 1998.

[4] M. Bernhard, "On lockdown, hong kong activists are protesting in animal crossing," 2020. [Online]. Available: https://www.wired.co.uk/article/animal-crossing-hong-kong-protests-coronavirus

[5] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, "Digital image steganography: Survey and analysis of current methods," *Signal processing*, vol. 90, no. 3, pp. 727–752, 2010.

[6] DigiMinecraft, "Villager trade generator (java edition)," https://www.digminecraft.com/generators/villager_trade.php, 2021.

[7] M. Ebrahimi and L. Chen, "Emerging cyberworld attack vectors: Modification, customization, secretive communications, and digital forensics in pc video games," 02 2014, pp. 939–944.

[8] C. Gibbs and N. Shashidhar, "Stegorogue: Steganography in two-dimensional video game maps," *Advances in Computer Science : an International Journal*, vol. 4, pp. 141–146, 2015.

[9] R. Gono, "Steganography in computer graphics," *Faculty of Informatics. Masaryk University*, 2017.

[10] C. Hale, "A new villain: Investigating steganography in source engine based video games table of contents," 2012.

[11] K. Joshi, "A new approach of text steganography using ascii values," *International Journal of Engineering and Technical Research*, vol. 7, pp. 490–493, 06 2018.

[12] Q. Nguyen, "Incremental improvements on the placement and routing of minecraft redstone circuits," 2018.

[13] N. Subramanian, O. Elharrouss, S. Al-Maadeed, and A. Bouridane, "Image steganography: A review of the recent advances," *IEEE Access*, 2021.

[14] A. Wajid, N. Kamal, M. Sharjeel, R. M. Sheikh, H. B. Wasim, M. H. Ali, W. Hussain, S. T. Ali, and L. Anjum, "A first look at private communications in video games using visual features," *Proc. Priv. Enhancing Technol.*, vol. 2021, no. 3, pp. 433–452, 2021. [Online]. Available: https://doi.org/10.2478/popets-2021-0055