

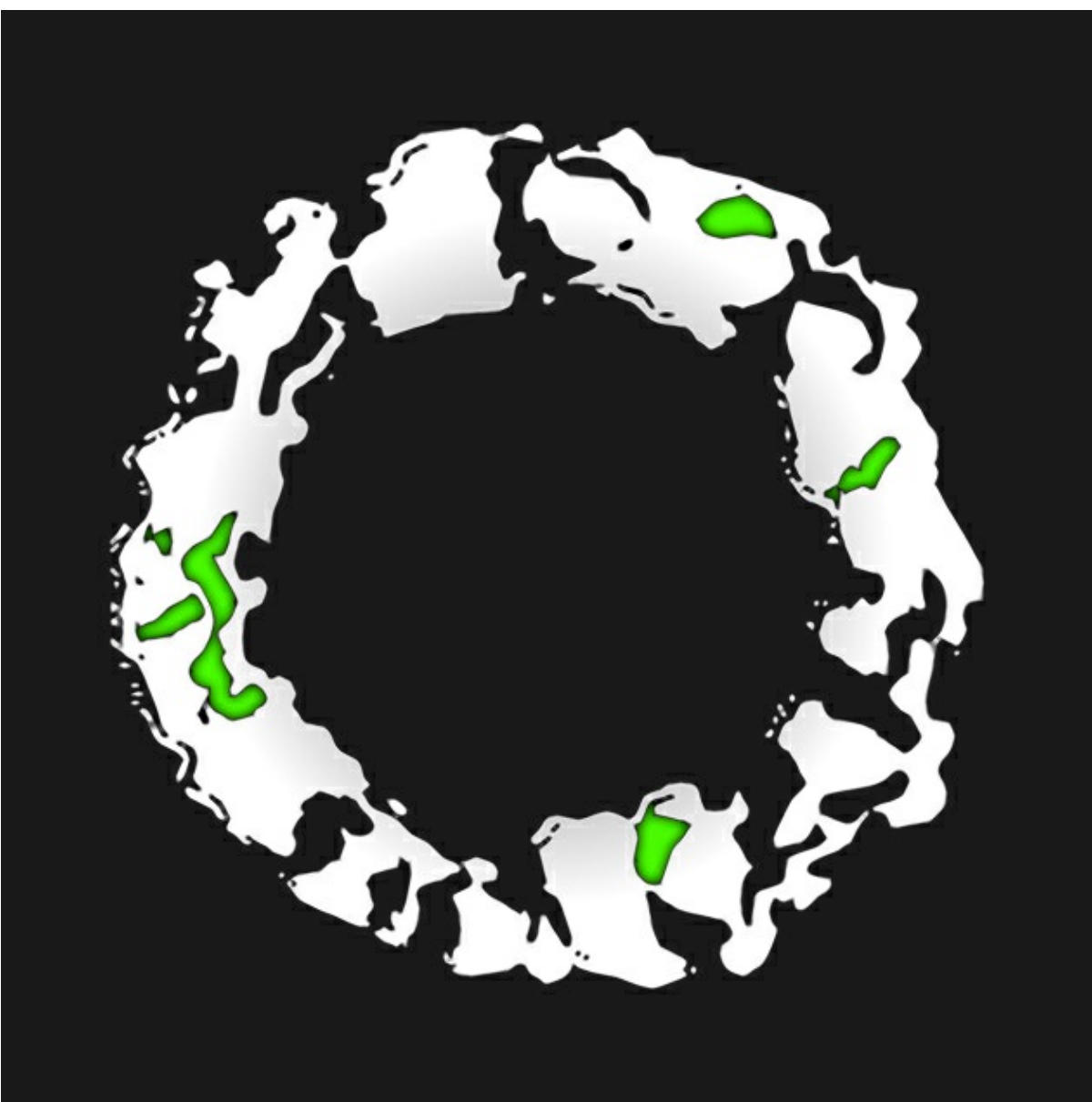
Montecrypto – ARGSS Write-Up

Author: [🔗 Brett Buerhaus](#)

🕒 April 24, 2018    👤 bbuerhaus    🔗 [argss](#), [bitcoin](#), [crypto](#), [gemroseaccent](#), [lemon](#), [montecrypto](#)

Montecrypto: The Bitcoin Enigma

A solve diary & challenge-by-challenge walkthrough by the crew who cracked it



ARG Solving Station: [Luigy](#), [en](#), [JTobcat](#), [ziot](#), [motive](#), [Askin](#), [lucifers\\_cat](#)  
Special thanks to: [Austin](#), [nsrc](#), [pipecork](#)

Meet us and learn more in ARGSS [https://discord.me/arg\\_solving\\_station](https://discord.me/arg_solving_station)

Solvers

- [@JTobcat](#)
- [@bbuerhaus](#) - [ziot](#)
- [@hackerswhoblaze](#) - [en](#)
- [@leemsparks](#) - [motive](#)
- [Askin](#)
- [@Luigy](#)
- [Lucifers\\_cat](#)
- and others on the [ARGSS Discord](#)

Summary

MonteCrypto is a UE4 crypto puzzle game that launched on Steam February 20, 2018. You are tasked with completing 24 Enigma puzzles to obtain words used to open a cryptocurrency wallet containing a 1 Bitcoin prize.

Upon entering the game, you land in a lobby with three exits. As soon as you walk through one of the exits, it begins a 60 minute timer that forces the game to restart when the time runs out.

The objective is to use any means necessary to collect the Enigma solutions and figure out how to unlock the Bitcoin wallet. This extended as far as hacking the game client and finding ways to solve broken puzzles.

This write-up is a journey of the tools we used, the technical approach we took to solve each Enigma, and some of the mistakes we made along the way.

Table of Contents

<a href="#">Summary</a>	<a href="#">2</a>
<a href="#">TOOLS USED FOR SOLVING</a>	<a href="#">4</a>
<a href="#">TIMELINE</a>	<a href="#">5</a>
<a href="#">The Game Map</a>	<a href="#">6</a>
<a href="#">PI SEQUENCE ORDER</a>	<a href="#">9</a>
<a href="#">ENIGMAS</a>	<a href="#">10</a>
<a href="#">Enigma 1</a>	<a href="#">10</a>
<a href="#">Enigma 2</a>	<a href="#">13</a>
<a href="#">Enigma 3</a>	<a href="#">17</a>
<a href="#">Enigma 4</a>	<a href="#">20</a>
<a href="#">Enigma 5</a>	<a href="#">23</a>
<a href="#">Enigma 6</a>	<a href="#">25</a>
<a href="#">Enigma 7</a>	<a href="#">27</a>
<a href="#">Enigma 8</a>	<a href="#">29</a>
<a href="#">Enigma 9</a>	<a href="#">31</a>
<a href="#">Enigma 10</a>	<a href="#">34</a>
<a href="#">Enigma 11</a>	<a href="#">38</a>
<a href="#">Enigma 12</a>	<a href="#">42</a>
<a href="#">Enigma 13</a>	<a href="#">43</a>
<a href="#">Enigma 14</a>	<a href="#">44</a>
<a href="#">Enigma 15</a>	<a href="#">45</a>
<a href="#">Enigma 16</a>	<a href="#">46</a>
<a href="#">Enigma 17</a>	<a href="#">47</a>
<a href="#">Enigma 18</a>	<a href="#">49</a>
<a href="#">Enigma 19</a>	<a href="#">50</a>
<a href="#">Enigma 20</a>	<a href="#">54</a>
<a href="#">Enigma 21</a>	<a href="#">58</a>
<a href="#">Enigma 22</a>	<a href="#">59</a>
<a href="#">Enigma 23</a>	<a href="#">60</a>
<a href="#">Enigma 24</a>	<a href="#">62</a>
<a href="#">Final Solve Steps / MonteCrypto Solution</a>	<a href="#">65</a>
<a href="#">ADDITIONAL FLAVOR PUZZLES</a>	<a href="#">66</a>
<a href="#">Compasses and Skulls</a>	<a href="#">66</a>
<a href="#">Appendix A - Wallet bruting</a>	<a href="#">67</a>
<a href="#">Appendix B - Biham and Kocher's Known Plaintext Attack</a>	<a href="#">70</a>
<a href="#">Appendix C - Disassembling UE4 Blueprints</a>	<a href="#">73</a>
<a href="#">Appendix D - important.jpg</a>	<a href="#">75</a>
<a href="#">Appendix E - Our Thoughts on Hacking the Client</a>	<a href="#">76</a>

TOOLS USED FOR SOLVING

Ninjaripper

<https://gamebanana.com/tools/5638>

A tool that we used for ripping images from memory

Cheatengine

We used this for teleporting and dumping data from game memory early on.

Austin's dll

Austin in the GameDetective community created a dll that did a lot of useful things. While some of it just made traversing the game 9000+ times tolerable, it also made one of the puzzles solvable. Without the ability to dump objects and teleport to them, it is likely that only a very few amount of teams would have solved the Spirit cave due to a bug that put all of the spirits outside of the game hallways in inaccessible areas.

- Anti-tp bypass
- noclip
- Game object dumper
- Dumped all of the disassembled code for Blueprints

Umodel

<http://www.gildor.org/en/projects/umodel>

This tools lets you load Unreal umodel 3d assets and 2d textures

Quickbms

<http://aluiqi.altervista.org/quickbms.htm>

A tool for extracting Unreal pak files. Using the Unreal Editor 4 script we were able to extract all the files from the MonteCrypto pak file.

Forensically

<https://29a.ch/photo-forensics/>

Really useful for quickly spotting weird shit in images without a lot of effort.

Password Generator

Script we wrote to generate possible wallet passwords using our ordering theory, or every possible order. Handles words we're not sure about either.

TIMELINE

An approximate timeline of our Enigma solves:

- (Day 1) 13 - Frog Pond / Jumping
- (Day 1) 17 - QR Codes
- (Day 1) 16 - Achievement
- (Day 1) 22 - Binary Window
- (Day 1) 14 - Server Room
- (Day 1) 6 - Office
- (Day 1) 18 - Blue Room
- (Day 1) 12 - Goblin Statue
- (Day 1) 23 - Painter
- (Day 2) 15 - White Lights
- (Day 3) 3 - Floating Crystal
- (Day 3) 24 - Sun Cave
- (Day 3) 21 - Vault
- (Day 4) 19 - Candle Cave



- (Day 4) 8 - Skull Room
- (Day 4) 5 - Lullaby
- (Day 4) 10 - Library
- (Day 4) 4 - Invisible Staircase
- (Day 4) 7 - Floor Pits-words mined day 1, solved via pi deduction
- (Day 4) 9 - Zelda Forest-words mined day 1, solved via pi deduction
- (Day 4) 1 - Epilepsy-words mined day 1, solved via pi deduction
- (Day 14) 11 - Rain
- (Day 22) 20 - Who Am I?
- (Day 63) 2 - Forest / Outside

## The Game Map

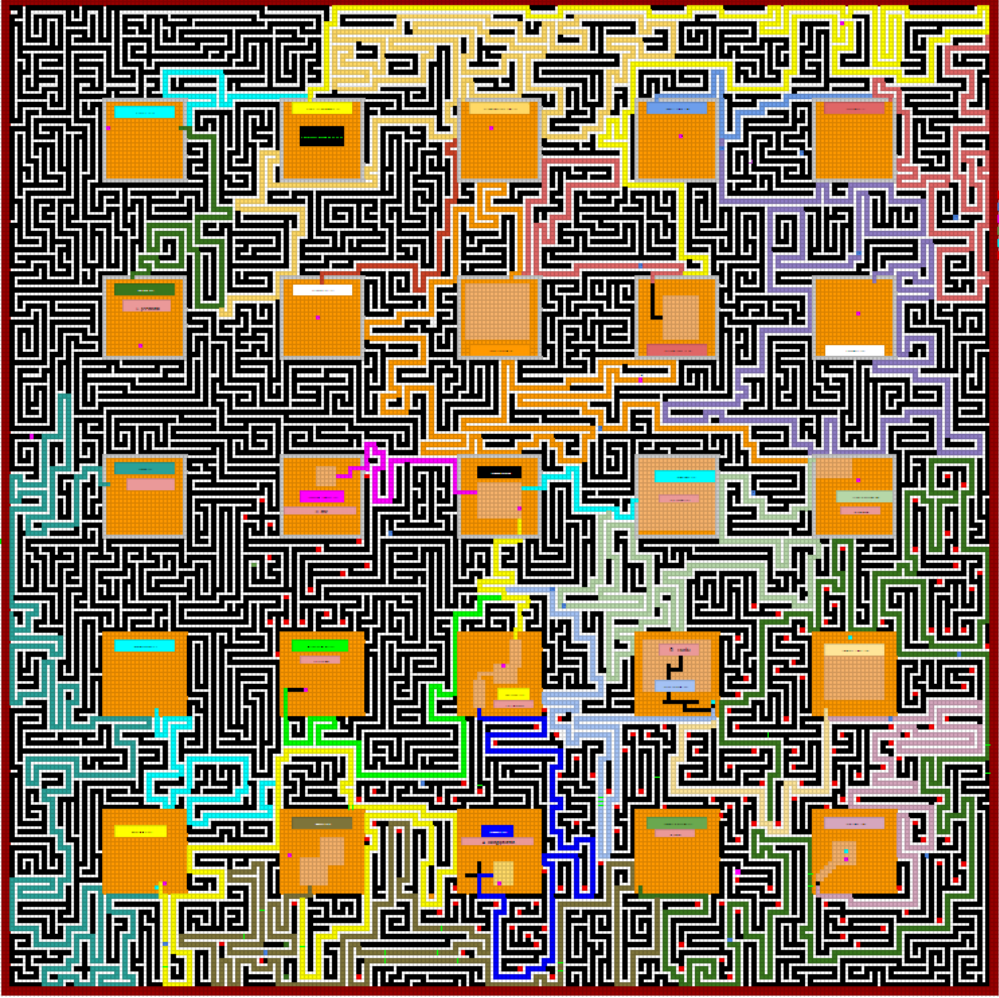
Our first few maps were Photoshop-fu via pulling together screenshots of each others maps as we blindly ran around the maze with one hour-timers preventing us from getting big areas of the maze mapped without needing to retrace our steps. Many times...



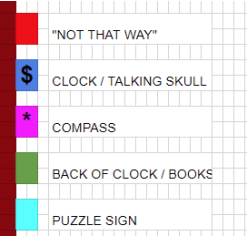
After we realized that the maze extends beyond the confines of the in-game map, this effort quickly diminished and was replaced by mapping it by hand in a Google Spreadsheet, which took 3 people the better part of a week to complete:

Doing this was made easier by a few realizations about how the maze was built as we traversed through it:

- The maze corridors are always odd lengths
- A turn was never made without some length of corridor after the turn
- There are never greater than one distance gaps between parallel corridors
- Other than enigma rooms, there are no large empty spaces



A labor of love - the maze mapped out by hand in a Google Spreadsheet.



The map's legend. We started to list out the locations of all "NOT THAT WAY" mayor appearances as well as the locations of all compasses and skull bookshelves.

## PI SEQUENCE ORDER

One of the first things we discovered in this game when roaming through the halls is that all of the dead-ends contain a creature that shouts "NOT THAT WAY!". This event occurs as you approach the wall with a small slide opening and the creature briefly shouting at you before closing it. No one really knows what this creature is, but some guessed that this might be The Mayor.



Image credits: [GameDetective Wiki](#)

Inside of the room, there is a terminal on the side that is somewhat obscured unless you look closely. The way the game works, it loads in a tiny room that is hidden above the lobby when you approach those dead-ends. By noclipping you can trivially fly into the tiny room and see the terminal.

The terminal says: Pi 121-144.

Taking the numbers of Pi starting at index 121 and going up until 144 gives you the following numbers: 093844609550582231725359

This gives us 24 numbers, the same number of Enigma/solutions in the game.

It was clear very early on that this was likely the order we needed to use to both validate some of our solutions and give a primary sort order to solution words for the wallet password.

## ENIGMAS

The Enigma solves below are not in the order that we solved them but the linear order based on the room numbers in the game.

### Enigma 1

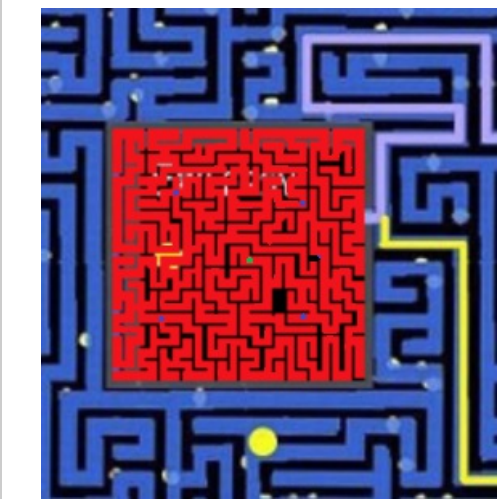
Room Nickname: Epilepsy

The four possible answers from the game files:

- 5-mzuzwana
- 7-random
- 2-bushido
- 63-monaci

Several members of our team were hesitant to try anything in-game in this room, and for good reason. It is a maze within the maze, made out of invisible walls that trigger strobing lights when you touch them. This could cause serious harm to anyone with epilepsy.

Here is a map of the "maze inside of the maze":



Even though we had four possible answers for this room from the game files, we needed to rule out red herrings sooner or later and solve this room in the, what we thought, intended way.

After enduring those strobing lights for what felt like an eternity for the eyes, one of our members found a trigger spot in the back of the room which turned off all lights in the entire maze and gave us, what we called, the "victory sound", but sadly no solution, yet.

After a while we figured out that there is a cube floating below the room, not visible for anyone who is playing the game without cheats, which had the four possible answers from the game files on it:



We got one step closer to verifying our hypothesis, but we were still missing something.

It turned out that we needed to run the game with the unpacked .pak file and delete the "BP\_Black" files to see the correct solution projected to the top of the room - 5 mzuzwana.





Solution: 5 mzuzwana

## Enigma 2

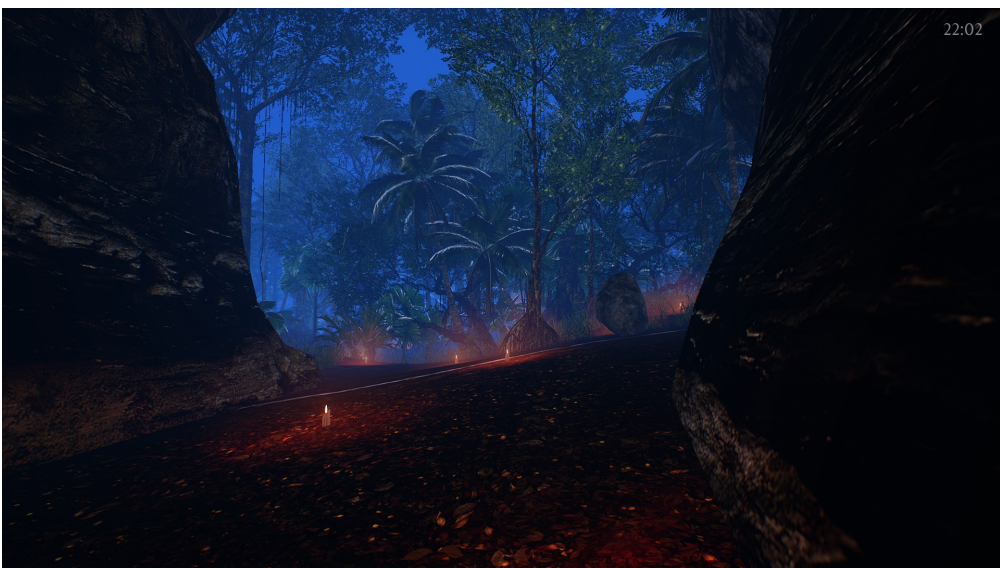
Room Nickname: Forest / Outside

A few hours after we got teleport / noclip cheats working we found the forest but had no idea where to start with it. It was the antithesis of low-hanging-fruit and quickly got put on the backburner until we had more information to work with. Little did we know that it would stay there until the end of the hunt.

The terminal text:

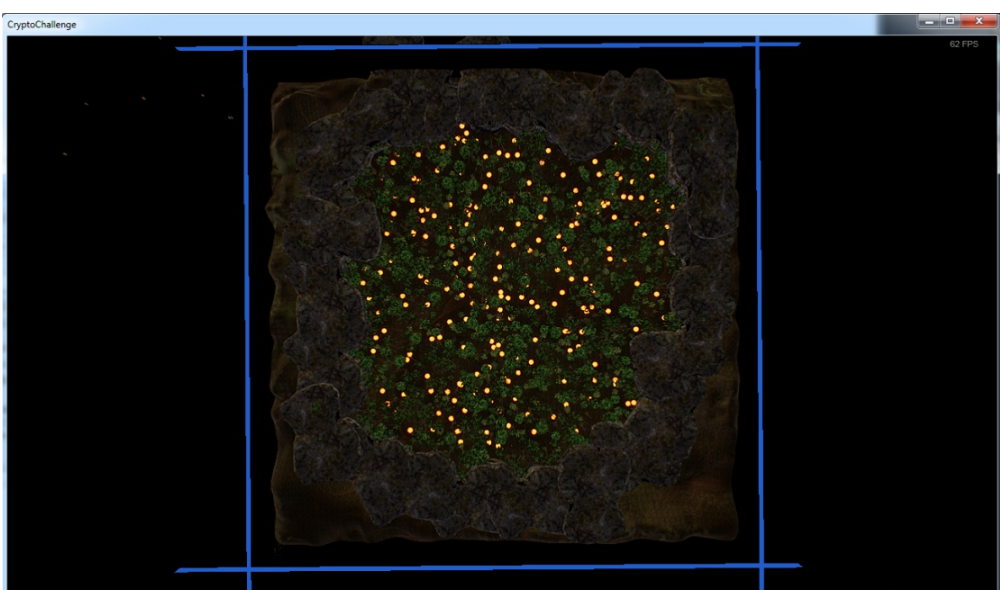
Y is sluggish in the sea;  
Try and try again;  
When and where the least expected;

When you enter the forest from the stairs, this is what you see:



It's a pretty overwhelming sight as you start to trek through the forest and realize there are over a hundred candles scattered across the area.

Using no-clip we could quickly see a top-down view of the forest:



Also being able to remove game files, we could see the area without all of the foliage:

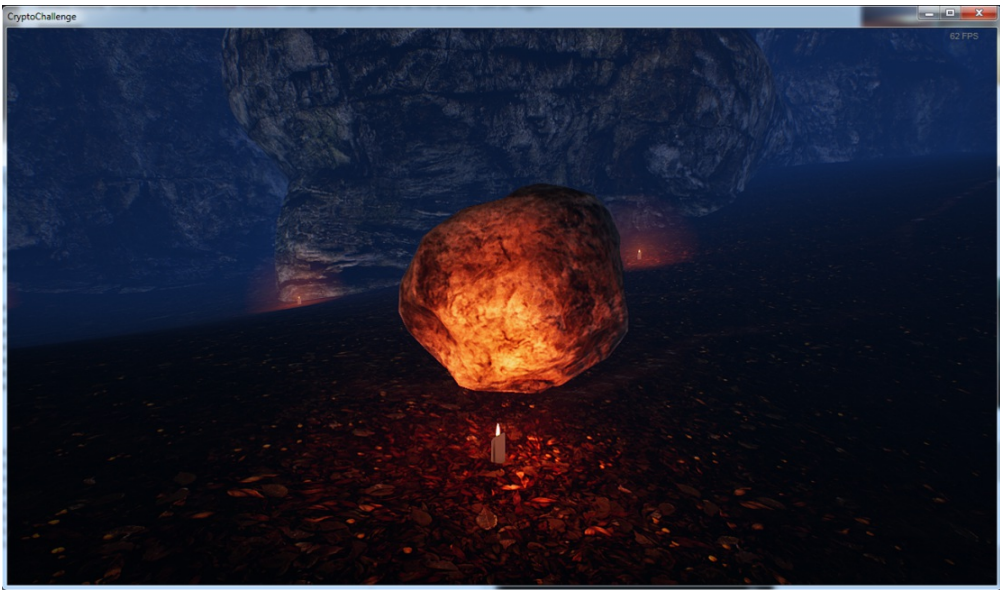


Using similar tricks to datamine info from the game files, we were able to see the Enigma file names and object names contained inside of the files. We discovered that there was a single candle/blueprint named BP\_Actionable\_CandleEn2.

Using Austin's DLL we were also able to dump all the object locations from memory so we could teleport directly to that candle eliminating the efforts of having to click every single candle in the forest.

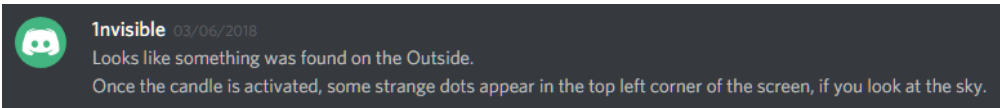
Clicking the candle played the victory noise which enabled a PostProcess effect for around 60 seconds. This gave everything a slight blue outline until the timer expired and would disappear.

It's also worth mentioning that the forest contained four boulders with PhysX enabled on them. As you enter the forest, walking over one of the candles enables the PhysX on it and it starts to roll down a hill in front of you.



We tried many things over the span of a couple weeks, going as far as teleporting or rolling the rocks to different areas or on top of every candle. In the end, these rocks were red herrings that were not used in the solution.

Eventually @1nvisible#9872 in GD noticed that the PostProcess overlay was adding single pixel dots to the top left of the screen.



It required a specific resolution and looking at certain areas such as the sky to visibly see the dots in the correct format. This is what it looked like when you extracted it correctly:

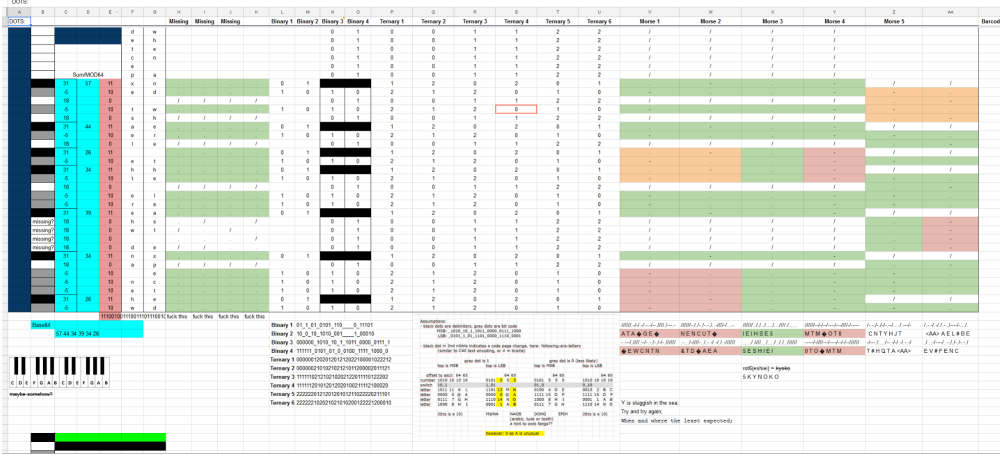


Credits: 1nvisible/GameDetectives

We immediately jumped to the conclusion that this was either morse code or ternary, but neither of those produced useful output. Treating it as morse with an invisible 5 dot starting sequence gave us a 5, which we thought was on the right track. [We started applying every possible decoding scheme](#). On the far side we've got CNTYHJT, which is off from the correct decoding by one character. This is because there is an invisible dot at the end we didn't consider. Treating that invisible dot as a "." gives us CNTYHJN -- which is the intended decoding.

Months later, while shooting the shit with pipecork/nsnc (as we ran [420blaze.in](#)), we were discussing outside. They were confident in their solve and told us rot13(CNTYHJN) = pagluwa, which translates to spit -- and if we translate "spit" back to the original language, we get that word back perfectly. This was the last Enigma we needed; We used it to crack the Bitcoin wallet shortly after.

It's funny how a single pixel mistake kept us from solving something many would consider a trivial crypto puzzle. Here is some insight into what our outside doc looked like:



Solve:



Solution: 5 pagluwa

## Enigma 3

Room Nickname: Floating Crystal

Terminal text:

In the race for success,  
speed is less important than stamina.

When you enter this Enigma room, there is a giant floating crystal that makes noises as you get close to it. When you walk into the crystal, it would make a glass shattering noise. A new crystal will spawn in a random location in the room. If you do not break it quickly enough, the Enigma resets. You have to continue shattering these crystals for approximately 15 minutes until you get a victory sound. After that, a long base64 looking string appears in the room.

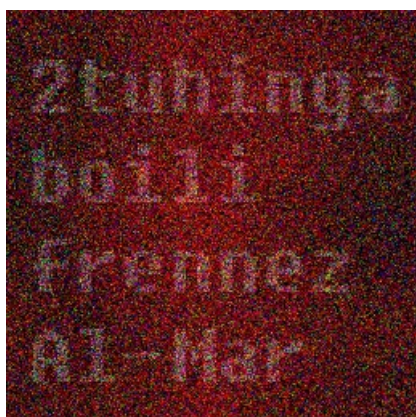


As with the other Enigmas that displayed strings in the game, the string was easily datamined from the Enigma blueprint files. We already extracted this string including two other red herrings. We didn't figure out how to use the string until after we solved the Enigma ingame.





This string turned out to be a password for one of the zips in the game files. When you extract the zip, you are given the following image:



The solution to this Enigma turned out to be 2 tuhiinga. We never figured out what "boili Frennez Al-Mar" meant. The only thing we noticed is that boili could have meant "boile, fish bait made of red herrings." If this is the case, we never figured out what red herring it was referencing other than possibly the final zip that was never opened.

**Solution: 2 tuhiinga**

### Enigma 4

Room Nickname: Invisible Stairs

When you enter this room there is a small platform up some stairs with an opening on it. Eventually you realize you can walk out and there are invisible stairs leading you upwards. There is music playing that gets louder and then you fall to the ground and the music ends. The trick to this room becomes evident quickly. As you walk up the invisible stairs, the music gets louder as you get closer to the edge. So all you have to do is move slowly and listen to the music.

That doesn't sound too bad, until you realize there is a 60 minute time limit and it takes a little bit of time to run to the room. Thankfully, we used Cheatengine (and austin.dll) to disable the timer entirely.

Even after people spent hours trying to run through the room legitimately, they always seemed to fall. Eventually, people tried to just teleport to the top, but it didn't work. It turned out there was a checkpoint system that made sure you hit all of them in the correct order. So people turned to teleporting every x,y,z of the room until the hit the top, but that didn't work either.

With some effort, the community found out they could make the stairs visible. This next screenshot will you show our pain.

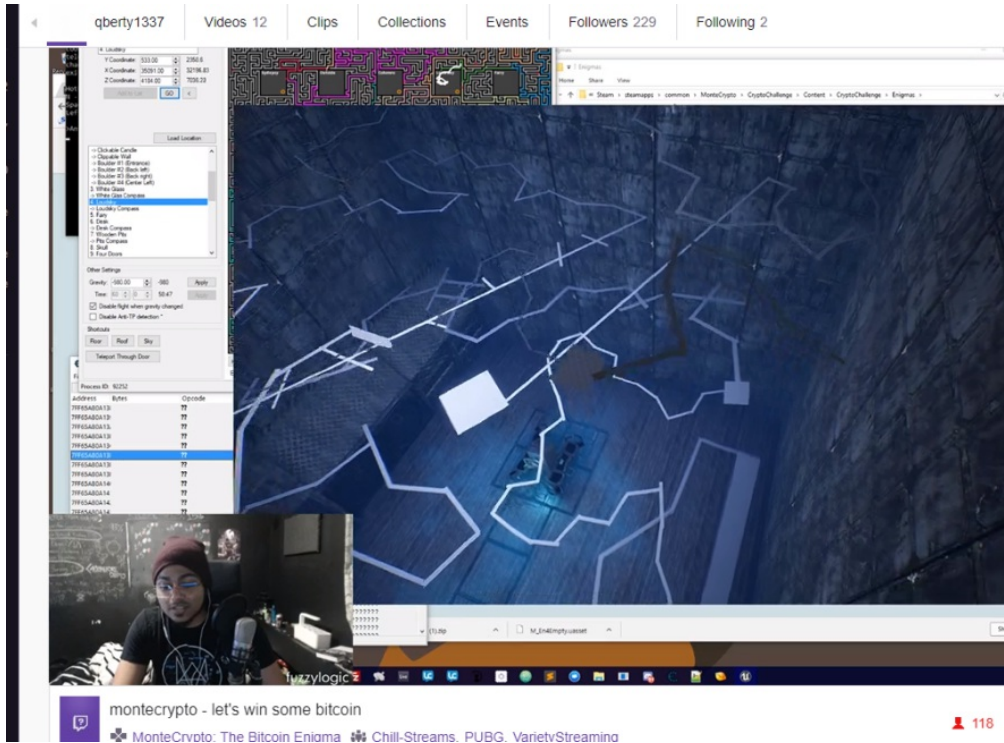


Image credit: <http://twitch.tv/qberty1337>

Notice the false paths? Notice the jumps? Probably the most evil puzzle ever made.

The community eventually powered through it and solved it (without the 60 minute timer active).



Funny enough, this string was extracted probably a month prior to the solve. Like some of the other Enigmas, there were some strings in the uasset files. The BP\_En4Manager.uexp file contained multiple strings such as:

- Gimmeahug
- Doyouwanttoworkforus
- Jobwelldone
- Areyoutired?
- Rickandmortyforever.com

I'm sure other teams had realized earlier on that jobwelldone was a password for one of the three zips in the game asset files. We did not make the connection until after jobwelldone was posted as a solution.

This led to a new image:



Giving us the solution 3-kohokohta for this Enigma.

Interestingly, the image retrieved from solving Enigma 3 fits perfectly inside the gray area of this solve image. Many attempts were made to manipulate the two images together using steganography software, methods like xor, adding byte values between the images etc, but nothing came of this. It's either a coincidence, red herring, or some flavor that we never solved.

**Solution: 3 kohokohta**

### Enigma 5

Room Nickname: Lullaby / Fairy

The Lullaby room contained high piles of books and a cone light in the center of the room.

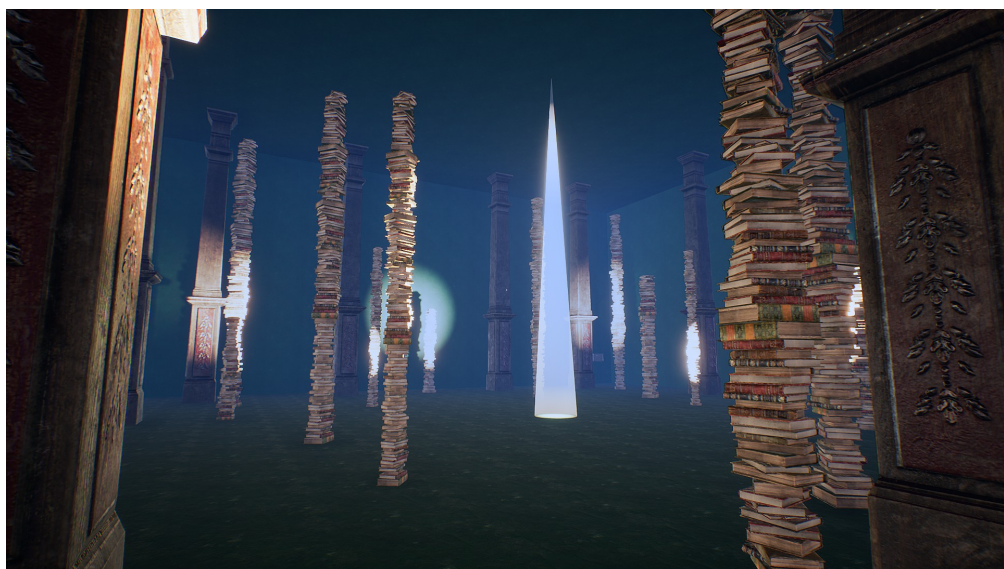


Image credits: [GameDetective Wiki](#)

When you stand in the light, it plays audio for a bit until it finishes. After that you gain a flashlight and a little wisp begins to fly around the room. You have to track the wisp with the flashlight for around 30-45 minutes.

I had a video link on Twitch of someone completing this room but it has since been deleted.

After you successfully track the wisp long enough, it plays the success audio and another audio file plays in the background. This second audio file contains Spectrogram steganography with the solution words.

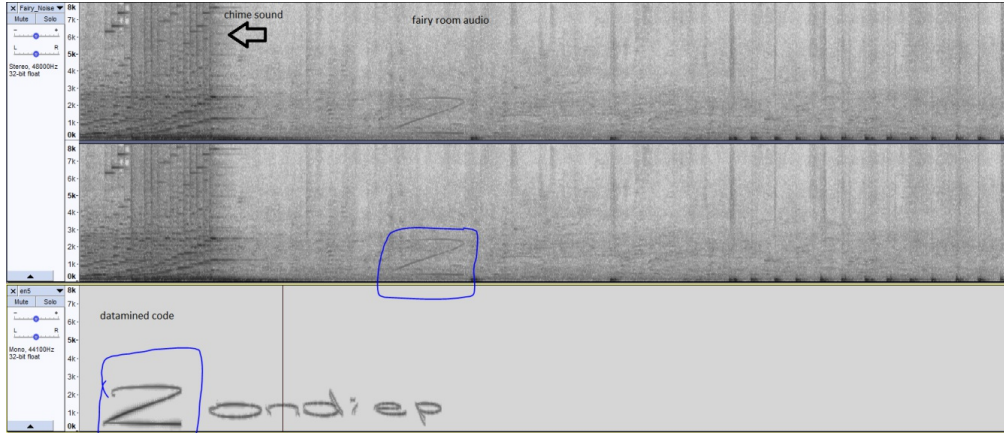


Image credit: Trailbl4z3r from [GameDiscord](#)

Using a tool such as Sonic Visualiser or Audacity, you can find the Spectro fairly trivially. The hard part was trying to trace out the word because it is hard to see.

**Solution: 2 ondiep**

### Enigma 6

Room Nickname: Office





Image credits: [GameDetective Wiki](#)

This Enigma was probably one of the more interesting ones that required effort outside of the game to solve. Our team had eyes on MonteCrypto prior to it launching so we were already looking for clues on their website and social media.

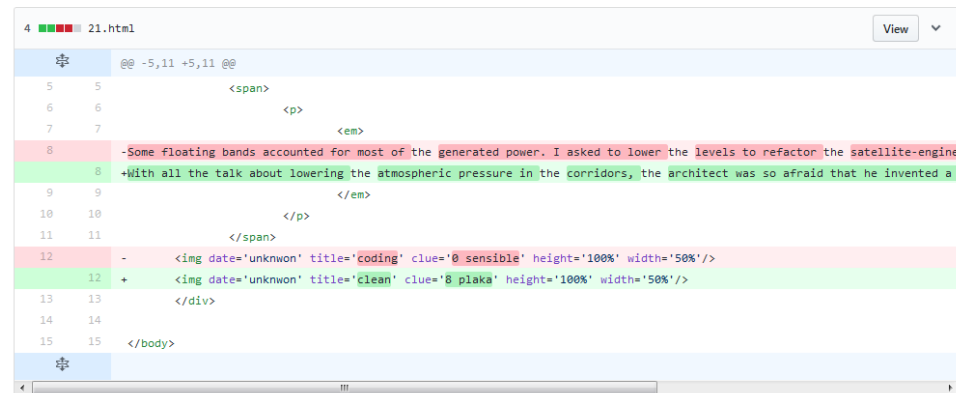
The MonteCrypto website is hosted on Github:

- <http://montecrypto-game.github.io/>

So we started to investigate the repository on Github:

- <https://github.com/montecrypto-game/montecrypto-game.github.io>

Looking at the [commit history](#) we see multiple changes to a file called 21.html. These changes that look like the following:



We didn't know what it meant prior to the game going live, but we already had it documented. By the time we reached the office Enigma in the game, the answer was pretty clear.

The terminal text read:

```
A very very long time ago, Rai had left me an important message...
I must have kept a note of it, but where?
```

In this commit on Github, we can see the answer:

<https://github.com/montecrypto-game/montecrypto-game.github.io/commit/3e64758479f49d273cd7b2e979a32e70378e9d59#diff-2c54dd4f6d3b8154f30f974cd3446ef>

+ <img date="a very very long time ago" title="confidential" clue="5 persamaan" height="100%" width="50%"/>

**Solution: 5 persamaan**

## Enigma 7

Room Nickname: Floor Pit

The Floor pit enigma was a room containing small pits that would show you a word when you fell into them. Without noclip/teleport, this meant you had to restart the game and run back to the room to collect all of the words.

The terminal text for this Enigma:

```
Even Luther Cary, my old friend, couldn't compete (and he held the record!).
Try it yourself.
But don't miss a turn.
```

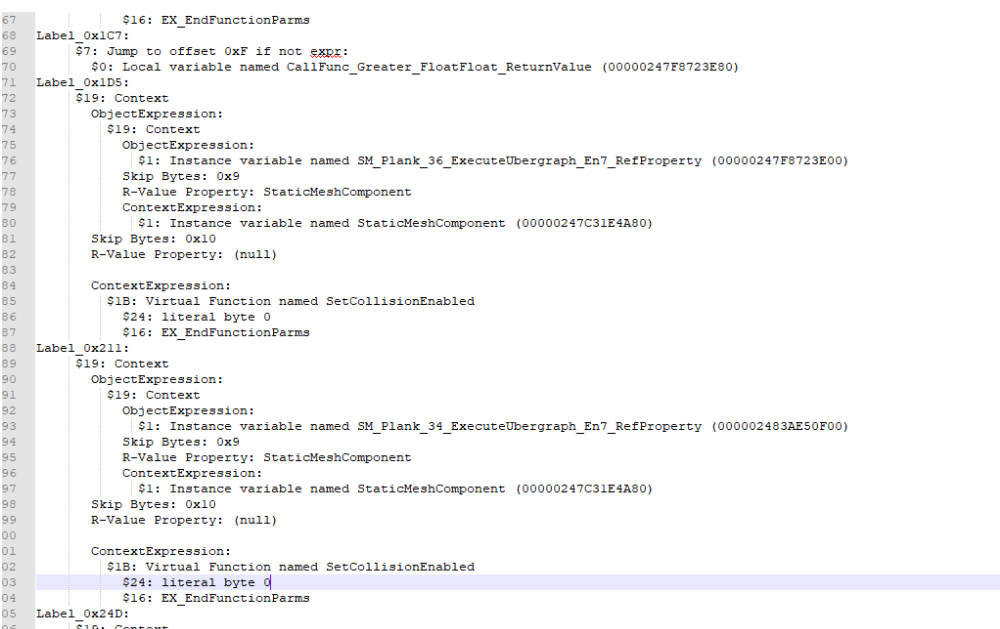
Some of the pits had wooden planks over them which made it impossible to get the words out without noclip. We assumed that the answer was one with the planks on top of it.

Luther Cary was an olympian runner that held records for track and field athletics. We quickly noticed that if you were fast enough, one of the planks had no collision on it. It wasn't until we got access to the blueprint disassembly that we figured out the answer.



Image credits: [GameDetective Wiki](#)

Part of the pits disassembly code:



This confirmed that if you reached the room within a certain amount of time, the sm\_plank game objects above one of the pits would lack collision. After that time limit, the planks would gain collision preventing you from falling into it (without noclip). Validating our theory and knowing we needed an 8, it gave us the solution we needed.

**Solution: 8 perro**

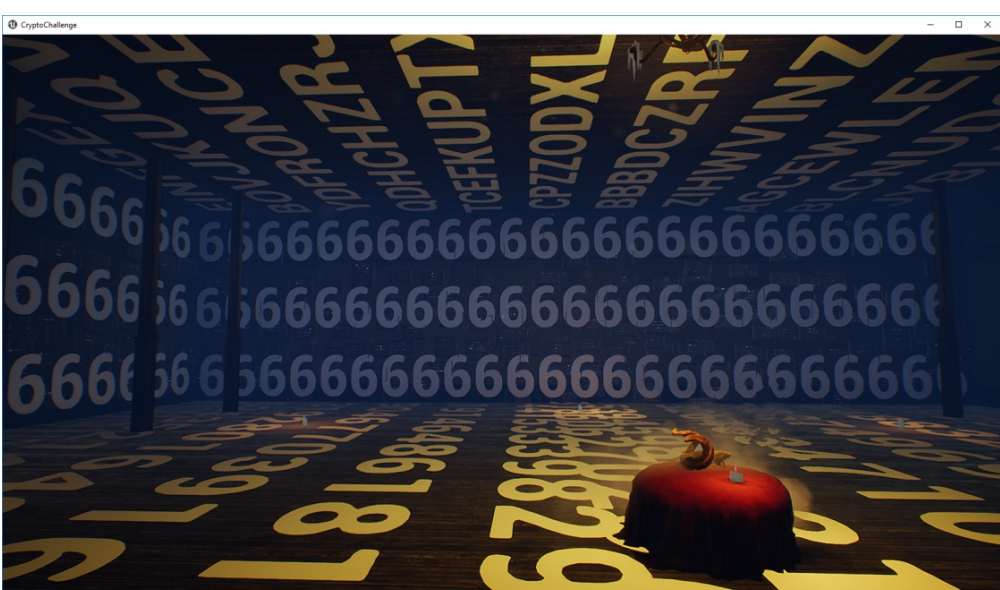
## Enigma 8

Room Nickname: Skull Room

This is one of the more interesting Enigmas in the MonteCrypto game. It did not take long for people to stumble into the Skull room as it's fairly close to the lobby. This led the entire room being filled with message signs everywhere.

```
When their time come,
The dead will answer;
Letters, Number,
Letters & Numbers,
Entered without feedback.
```

At exactly 03:33:00 AM (system clock, so you could change it) for a brief second textures would appear on every surface of the room.



Creepy, right?

Unlike some of the other Enigmas where you could rip strings directly from the uasset/uexp files, we had to pull these strings directly from the game memory.

Using the terminal text, you can guess that we had to input the string back into the game. Using a script, we send only the key inputs from the string on the ceiling (letters only) and floor (numbers only).

Our first attempt to run this script resulted in it opening up UI elements like the map and causing the player to move around the map, since we were pressing valid key inputs. As an extra added step that was likely unnecessary, we ended up changing our keybindings to F1-F12 keys to prevent them from interrupting the input. The "entered without feedback" part of the hint led us to believe this was necessary.



**Solution: 6 okpu**

## Enigma 9

Room Nickname: Zelda Forest

Enigma 9 is a giant steel tower that you enter through a doorway. Inside the first room you have three doors in front of you and one door behind you. If you go through the door behind you, you leave the Enigma and everything resets. When you move through the other three doors, you find yourself in a room similar to the one you just came from. We nicknamed this the Zelda Forest after a similar puzzle in the Nintendo 64 Legend of Zelda: Ocarina of Time.

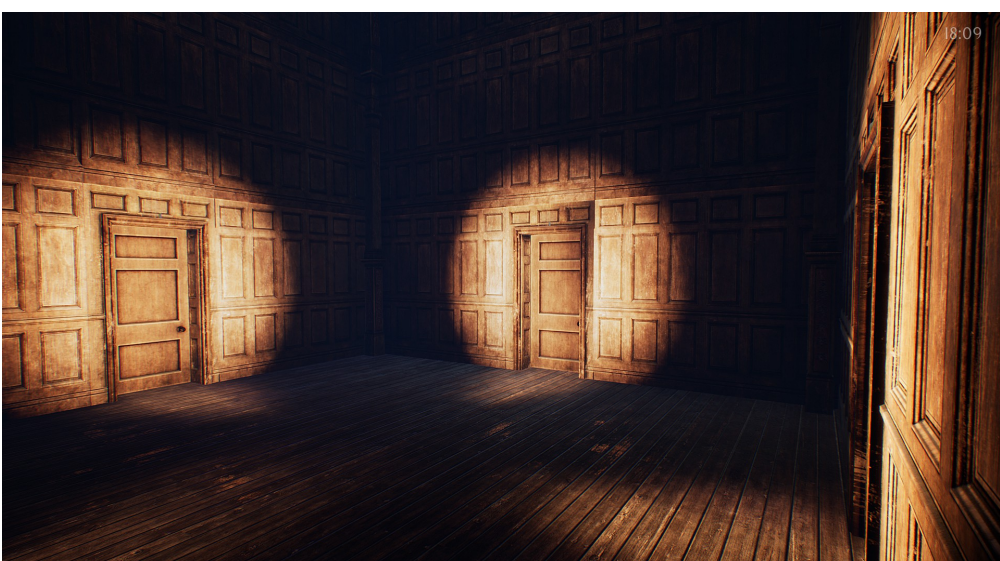


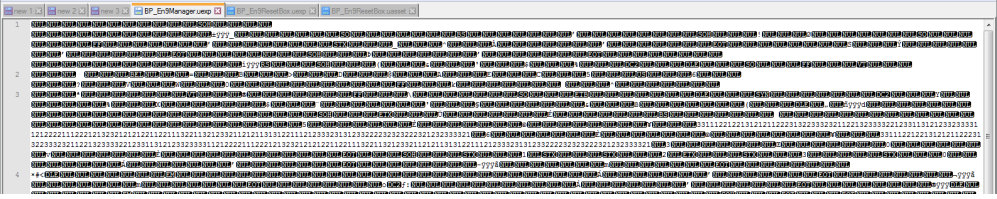
Image credits: [https://wiki.gamedetectives.net/index.php?title=File:4\\_Doors.jpg](https://wiki.gamedetectives.net/index.php?title=File:4_Doors.jpg)



We were able to solve this one a bit earlier than the public discords did. We discovered one of the blueprint files for en9 (BP\_En9Manager.uexp) contained the following string:

33111221221312121122231322333232112213233332212331131212332333311212222111222121323212121221113221132123321121211313122111212333331312332222323222321232333321

Using a text or hex editor you could discover it trivially:



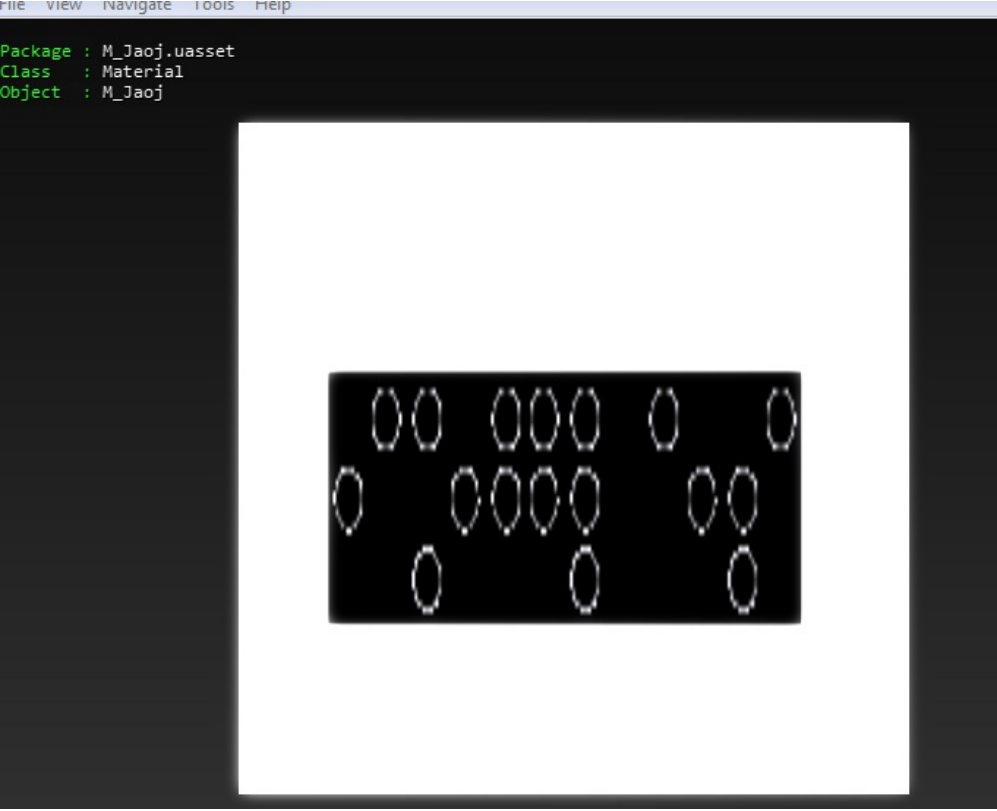
After assigning 1 to the left door, 2 to the middle door, and 3 to the right door, we followed this path start to finish until the last doorway played the victory sound. After that, each room lit up super bright and the ground textures changed.



Through three of the doors, you could see that there were 3 different ground textures. If you went through one of the doors, the puzzle reset. You either had to take pictures from the doorway or use noclip to get a clear view of it.

These dots on the ground were braille that gave you the room solve when translated

This was a solve that had already been extracted within the first day of the game being live, but like some of the other Enigmas it had a few red herrings with it.



Two of the braille images in the game data turned out to be red herrings.

- 1 - boutil - Red Herring
- 5 - jaojet - Red Herring
- 9 - ogles - Solve

We never figured out if there was something in the game that told you which direction to go allowing you to solve it using only in-game mechanics.

**Solution: 9 ogles**

### Enigma 10

Room Nickname: Library

As you enter this Enigma you go into a multi-level library with stairs leading down to a table and terminal at the bottom. The terminal text hints at what you need to do.

This manor isn't a safe place to be alone;  
Paths unlock when people come together;  
Only through shared efforts will you get closer  
To shed the light on the unknown

Throughout the library there are empty plaques on all the walls similar to the one you saw in the lobby at the beginning of the game. The terminal text and these plaque hinted that the community needed to come together to solve this room.

An individual player can only have one active sign at any given time, that means we needed a bunch of players to put signs on all of the plaques until it triggered the next step.

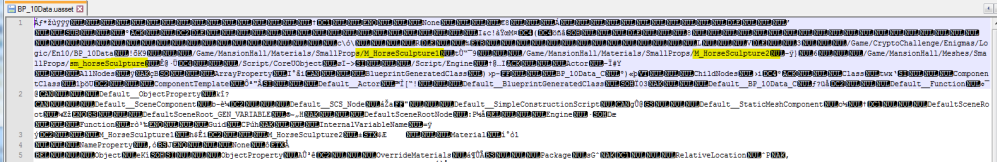


Image credits: [GameDetective Wiki](#)

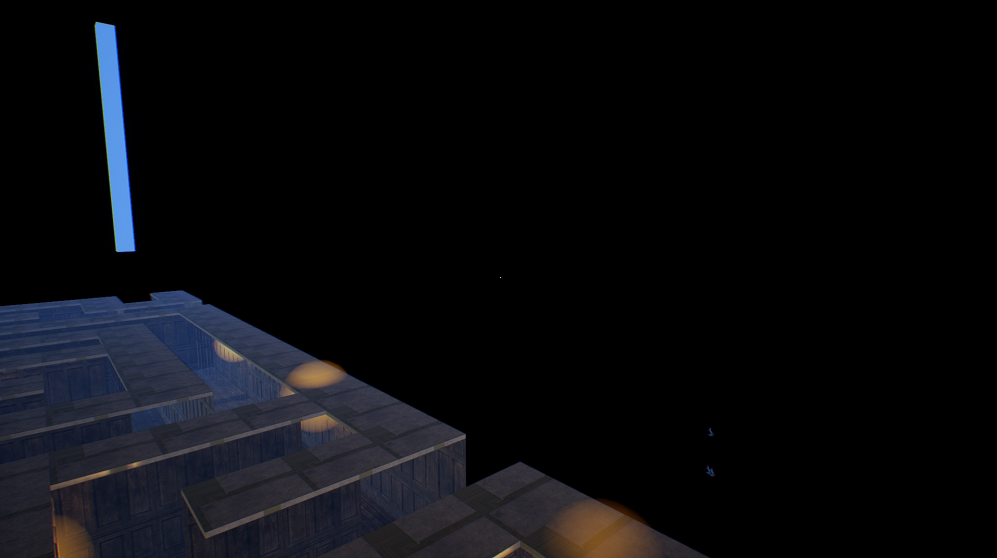
Once the condition was met, it would play a victory sound as soon as all the signs loaded. Given that there were thousands of signs in the game at this point, this could take up to 30-40 minutes for whatever reason.

Here comes the fun part. This Enigma is actually broken and required a bit of datamining and hacking the game client to solve it. After the victory sound, nothing happens.

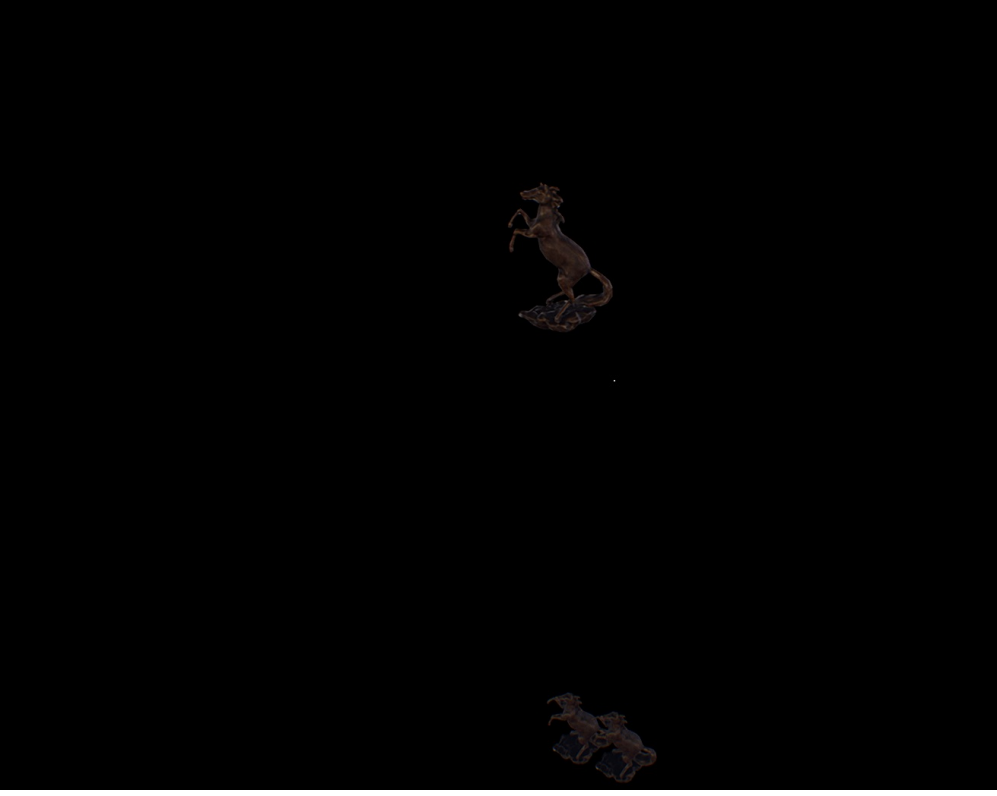
Looking through the game files you'll notice there are references to horse sculptures that are nowhere to be found in the library.



One of the members of our team happened to notice three horse statues were appearing near 0,0,0 xyz.

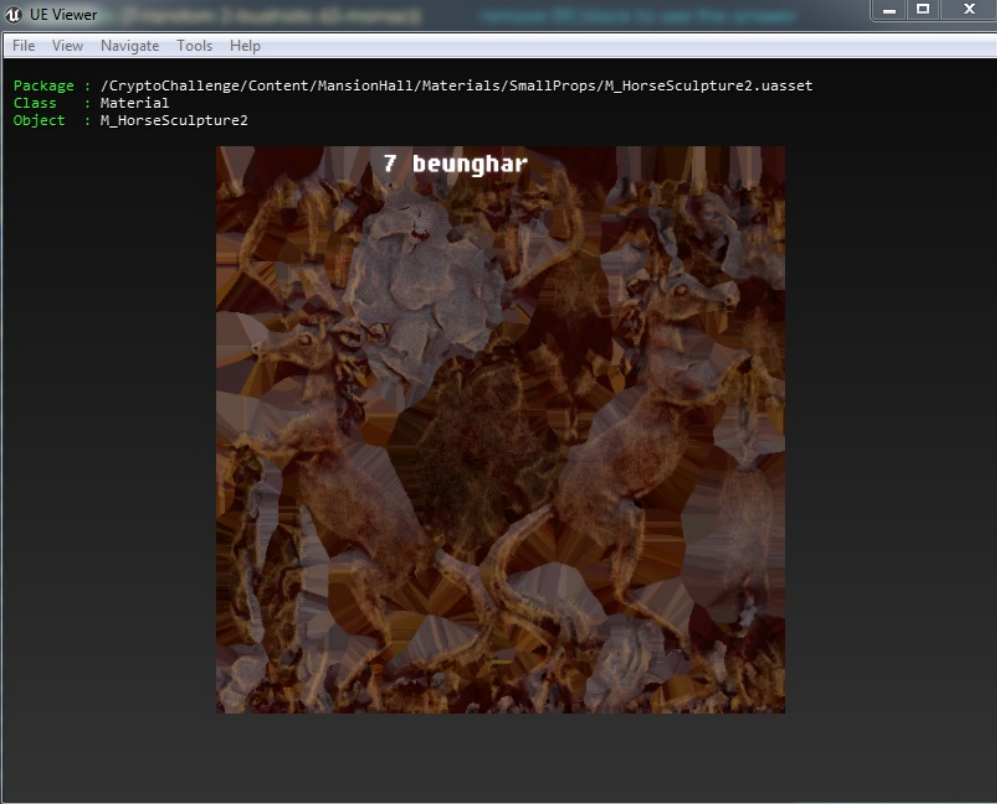


You can see the three horses at the bottom right of this screenshot



A close-up inspection of the horse statues using noclip

This led us to investigating the horse textures in the game assets.



The problem is, there are three different textures (HorseSculpture, HorseSculpture1, HorseSculpture2) with three different solves on them. Like the other Enigmas, any dataminable answer had multiple red herrings with them.

In the game itself, you couldn't actually see the word on any of the models. We were only able to see the word via UE Viewer. One theory is that the horse statues were meant to be on the table in the library and a light in there would expose the word. Either way, we were able to determine the solution based on the pi sequence order. We needed a 7.

**Solution: 7 beunghar**

### Enigma 11

Room Nickname: Raining / Temple

This is probably the only room that was originally intended to utilize cheating to solve as indicated by the enigma text.

Honesty is for the most part less profitable than dishonesty.

Using no-clip and walking through the wall, you would see a series of signs on the backside of the wall that show 50280.0.









On a side note: Much later on we realized that the asset which handles the blinking on the computers in this room is reused in eenigma 11 on the blinking statue, so we ruled it out as a red herring.

**Solution: 9 istisna**

### Enigma 15

Room Nickname: White Lights

This is a really interesting puzzle that was solved by the GameDetective community before we even looked at it. It involved playing a small riddle game using game save files on your computer. I recommend reading the detailed write-up on the wiki:

[https://wiki.gamedetectives.net/index.php?title=MonteCrypto\\_-\\_The\\_Bitcoin\\_Enigma#Robot](https://wiki.gamedetectives.net/index.php?title=MonteCrypto_-_The_Bitcoin_Enigma#Robot)

**Solution: 5 velte**

### Enigma 16

Room Nickname: Achievement

The solve for this enigma is printed on a hidden achievement icon that is obtained through doing a specific action in the game. The game came with Steam achievements but there are a few that are secret. Using a Steam achievement unlocker or just viewing it on some of the Steam achievement/stat sites, you quickly figure it out: That said, even though this Enigma is easily solvable using an out-of game method, we pretty quickly on the first day of the game's release found how to unlock it legitimately.




When you enter this enigma room you're presented with the text:

Just like platform 9 3/4,  
Towards the door go forth;  
Physical illusions,  
everywhere in this maze;  
Don't let yourself be fooled,  
you will be amazed.

This is a reference to the Harry Potter series where at King's Cross station in London Hogwarts Express passengers are able to run through walls to reach the hidden platform. We needed to do just that. Running straight through the wall down the corridor to the right of the console allows you to pass through it and unlock the hidden achievement with the solve to this enigma.



Image credits: [GameDetective Wiki](#)

A_17	Attention to details. Use the zoom.	
A_18	This specific wall. ⚠ This achievement is hidden.	
A_19	Not That Way! Now you know...	

Source: <https://steamdb.info/app/768750/stats/>

**Solution: 8 optree**

### Enigma 17

Room Nickname: QR Code

This is another one of the early Enigmas that you reach from the lobby. As you enter the room, you come across a terminal and a door.

Terminal text

Transcendental yet widely used,  
On March 14th it gets amused.  
Add an E an you can eat it  
Keep it in mind, later you will need it.

Clicking on the door, it asks you for the solution to the riddle which is "Pi."

- March 14th = pi day (3/14)
- Add an e and you can eat it (pie)
- Keep it in mind, later you will need it (the pi digit sequence for the wallet order).

When you get through the door, you see a QR code giving you the solution.



Image credits: [GameDetective Wiki](#)

**Solution: 3 prevoditi**

### Enigma 18

Room Nickname: Blue Room

The enigma text terminal when approaching this room reads:

Find your path amongst the invisible forces.  
Once the red is reached, the goal is almost accomplished.  
One that would need additional clues for this one should reconsider actually winning the prize.

Inside the room ahead is an area that is completely rendered in blue textures and thusly very difficult to distinguish what is a wall vs ground, etc.. when moving around it in 3D space. On the opposite side of the room is a red platform, and between you and said platform is an invisible bridge and underneath it a pit that leads back to the start of the room.

Carefully traversing and falling lots of down into the pit, we were able to reach the red platform.

Upon reaching the platform a long deep sound plays. After some manipulation of this sound in Audacity we realized that it is an extremely slowed down audio of someone speaking the answer to the enigma.

Here's a video of the puzzle:

- <https://streamable.com/l47f5>

Since we likely sped up audio to not exactly its original form, it wasn't completely clear what was being said - There was lots of debate over whether the audio here was saying NODO, NOVO, or NOTO. After some more Audacity-fu we were able to get a version that was leaning 80% towards NODO, but even then since we weren't sure, when we reached the final step of entering our answers into the BTC transaction, we had to consider NOVO or NOTO as possible replacements for this solve.

Turns out nodo was correct.

**Solution: 0 nodo**

### Enigma 19

Room Nickname: Candle Cave / Spirits

This room evaded us for a very long time. And it turns out that it's because it's completely and utterly broken.

Terminal text:

Spirited machines ought to exist  
Those who are correctly equipped  
Can follow them as they communicate  
At last are rewarded those who wait

Entering this room does not trigger anything to happen, nor does moving around it. For a while we were concerned that not getting here legitimately was the problem, but doing so many times did not result in anything new either.

Delving into the game data for this enigma we discovered a strings in \CryptoChallenge\Content\CryptoChallenge\Enigmas\Logic\En19\BP\_En19Manager.uexp that showed the text PLEASEBEQUIETANDTAKEAREST and ASCII art of the text 0 colle.





```
88 BF 02 78 7B 7C 7D 7E 7F 80 .....88BF02787B7C7D7E7F80.....
89 00 00 00 00 00 00 00 .....890000000000000000000000.....
8A 00 00 00 00 00 00 00 .....8A0000000000000000000000.....
8B 00 00 00 00 00 00 00 .....8B0000000000000000000000.....
8C 00 00 00 00 00 00 00 .....8C0000000000000000000000.....
8D 00 00 00 00 00 00 00 .....8D0000000000000000000000.....
8E 00 00 00 00 00 00 00 .....8E0000000000000000000000.....
8F 00 00 00 00 00 00 00 .....8F0000000000000000000000.....
90 00 00 00 00 00 00 00 .....900000000000000000000000.....
91 00 00 00 00 00 00 00 .....910000000000000000000000.....
92 00 00 00 00 00 00 00 .....920000000000000000000000.....
93 00 00 00 00 00 00 00 .....930000000000000000000000.....
94 00 00 00 00 00 00 00 .....940000000000000000000000.....
95 00 00 00 00 00 00 00 .....950000000000000000000000.....
96 00 00 00 00 00 00 00 .....960000000000000000000000.....
97 00 00 00 00 00 00 00 .....970000000000000000000000.....
98 00 00 00 00 00 00 00 .....980000000000000000000000.....
99 00 00 00 00 00 00 00 .....990000000000000000000000.....
9A 00 00 00 00 00 00 00 .....9A0000000000000000000000.....
9B 00 00 00 00 00 00 00 .....9B0000000000000000000000.....
9C 00 00 00 00 00 00 00 .....9C0000000000000000000000.....
9D 00 00 00 00 00 00 00 .....9D0000000000000000000000.....
9E 00 00 00 00 00 00 00 .....9E0000000000000000000000.....
9F 00 00 00 00 00 00 00 .....9F0000000000000000000000.....
A0 00 00 00 00 00 00 00 .....A00000000000000000000000.....
A1 00 00 00 00 00 00 00 .....A10000000000000000000000.....
A2 00 00 00 00 00 00 00 .....A20000000000000000000000.....
A3 00 00 00 00 00 00 00 .....A30000000000000000000000.....
A4 00 00 00 00 00 00 00 .....A40000000000000000000000.....
A5 00 00 00 00 00 00 00 .....A50000000000000000000000.....
A6 00 00 00 00 00 00 00 .....A60000000000000000000000.....
A7 00 00 00 00 00 00 00 .....A70000000000000000000000.....
A8 00 00 00 00 00 00 00 .....A80000000000000000000000.....
A9 00 00 00 00 00 00 00 .....A90000000000000000000000.....
AA 00 00 00 00 00 00 00 .....AA0000000000000000000000.....
AB 00 00 00 00 00 00 00 .....AB0000000000000000000000.....
AC 00 00 00 00 00 00 00 .....AC0000000000000000000000.....
AD 00 00 00 00 00 00 00 .....AD0000000000000000000000.....
AE 00 00 00 00 00 00 00 .....AE0000000000000000000000.....
AF 00 00 00 00 00 00 00 .....AF0000000000000000000000.....
B0 00 00 00 00 00 00 00 .....B00000000000000000000000.....
B1 00 00 00 00 00 00 00 .....B10000000000000000000000.....
B2 00 00 00 00 00 00 00 .....B20000000000000000000000.....
B3 00 00 00 00 00 00 00 .....B30000000000000000000000.....
B4 00 00 00 00 00 00 00 .....B40000000000000000000000.....
B5 00 00 00 00 00 00 00 .....B50000000000000000000000.....
B6 00 00 00 00 00 00 00 .....B60000000000000000000000.....
B7 00 00 00 00 00 00 00 .....B70000000000000000000000.....
B8 00 00 00 00 00 00 00 .....B80000000000000000000000.....
B9 00 00 00 00 00 00 00 .....B90000000000000000000000.....
BA 00 00 00 00 00 00 00 .....BA0000000000000000000000.....
BB 00 00 00 00 00 00 00 .....BB0000000000000000000000.....
BC 00 00 00 00 00 00 00 .....BC0000000000000000000000.....
BD 00 00 00 00 00 00 00 .....BD0000000000000000000000.....
BE 00 00 00 00 00 00 00 .....BE0000000000000000000000.....
BF 00 00 00 00 00 00 00 .....BF0000000000000000000000.....
C0 00 00 00 00 00 00 00 .....C00000000000000000000000.....
C1 00 00 00 00 00 00 00 .....C10000000000000000000000.....
C2 00 00 00 00 00 00 00 .....C20000000000000000000000.....
C3 00 00 00 00 00 00 00 .....C30000000000000000000000.....
C4 00 00 00 00 00 00 00 .....C40000000000000000000000.....
C5 00 00 00 00 00 00 00 .....C50000000000000000000000.....
C6 00 00 00 00 00 00 00 .....C60000000000000000000000.....
C7 00 00 00 00 00 00 00 .....C70000000000000000000000.....
C8 00 00 00 00 00 00 00 .....C80000000000000000000000.....
C9 00 00 00 00 00 00 00 .....C90000000000000000000000.....
CA 00 00 00 00 00 00 00 .....CA0000000000000000000000.....
CB 00 00 00 00 00 00 00 .....CB0000000000000000000000.....
CC 00 00 00 00 00 00 00 .....CC0000000000000000000000.....
CD 00 00 00 00 00 00 00 .....CD0000000000000000000000.....
CE 00 00 00 00 00 00 00 .....CE0000000000000000000000.....
CF 00 00 00 00 00 00 00 .....CF0000000000000000000000.....
D0 00 00 00 00 00 00 00 .....D00000000000000000000000.....
D1 00 00 00 00 00 00 00 .....D10000000000000000000000.....
D2 00 00 00 00 00 00 00 .....D20000000000000000000000.....
D3 00 00 00 00 00 00 00 .....D30000000000000000000000.....
D4 00 00 00 00 00 00 00 .....D40000000000000000000000.....
D5 00 00 00 00 00 00 00 .....D50000000000000000000000.....
D6 00 00 00 00 00 00 00 .....D60000000000000000000000.....
D7 00 00 00 00 00 00 00 .....D70000000000000000000000.....
D8 00 00 00 00 00 00 00 .....D80000000000000000000000.....
D9 00 00 00 00 00 00 00 .....D90000000000000000000000.....
DA 00 00 00 00 00 00 00 .....DA0000000000000000000000.....
DB 00 00 00 00 00 00 00 .....DB0000000000000000000000.....
DC 00 00 00 00 00 00 00 .....DC0000000000000000000000.....
DD 00 00 00 00 00 00 00 .....DD0000000000000000000000.....
DE 00 00 00 00 00 00 00 .....DE0000000000000000000000.....
DF 00 00 00 00 00 00 00 .....DF0000000000000000000000.....
E0 00 00 00 00 00 00 00 .....E00000000000000000000000.....
E1 00 00 00 00 00 00 00 .....E10000000000000000000000.....
E2 00 00 00 00 00 00 00 .....E20000000000000000000000.....
E3 00 00 00 00 00 00 00 .....E30000000000000000000000.....
E4 00 00 00 00 00 00 00 .....E40000000000000000000000.....
E5 00 00 00 00 00 00 00 .....E50000000000000000000000.....
E6 00 00 00 00 00 00 00 .....E60000000000000000000000.....
E7 00 00 00 00 00 00 00 .....E70000000000000000000000.....
E8 00 00 00 00 00 00 00 .....E80000000000000000000000.....
E9 00 00 00 00 00 00 00 .....E90000000000000000000000.....
EA 00 00 00 00 00 00 00 .....EA0000000000000000000000.....
EB 00 00 00 00 00 00 00 .....EB0000000000000000000000.....
EC 00 00 00 00 00 00 00 .....EC0000000000000000000000.....
ED 00 00 00 00 00 00 00 .....ED0000000000000000000000.....
EE 00 00 00 00 00 00 00 .....EE0000000000000000000000.....
EF 00 00 00 00 00 00 00 .....EF0000000000000000000000.....
F0 00 00 00 00 00 00 00 .....F00000000000000000000000.....
F1 00 00 00 00 00 00 00 .....F10000000000000000000000.....
F2 00 00 00 00 00 00 00 .....F20000000000000000000000.....
F3 00 00 00 00 00 00 00 .....F30000000000000000000000.....
F4 00 00 00 00 00 00 00 .....F40000000000000000000000.....
F5 00 00 00 00 00 00 00 .....F50000000000000000000000.....
F6 00 00 00 00 00 00 00 .....F60000000000000000000000.....
F7 00 00 00 00 00 00 00 .....F70000000000000000000000.....
F8 00 00 00 00 00 00 00 .....F80000000000000000000000.....
F9 00 00 00 00 00 00 00 .....F90000000000000000000000.....
FA 00 00 00 00 00 00 00 .....FA0000000000000000000000.....
FB 00 00 00 00 00 00 00 .....FB0000000000000000000000.....
FC 00 00 00 00 00 00 00 .....FC0000000000000000000000.....
FD 00 00 00 00 00 00 00 .....FD0000000000000000000000.....
FE 00 00 00 00 00 00 00 .....FE0000000000000000000000.....
FF 00 00 00 00 00 00 00 .....FF0000000000000000000000.....
```

Though we found this ascii art text in the game data, as with all of the puzzles we did not know if it was a red herring or the actual solution, so we set out to figure out how to actually solve the room in the game.

We had two conclusions going in to the room based on the hint given on the console and from data mining the files:

1. You're going to need some "special equipment" to solve this one
2. There's going to be some waiting involved.

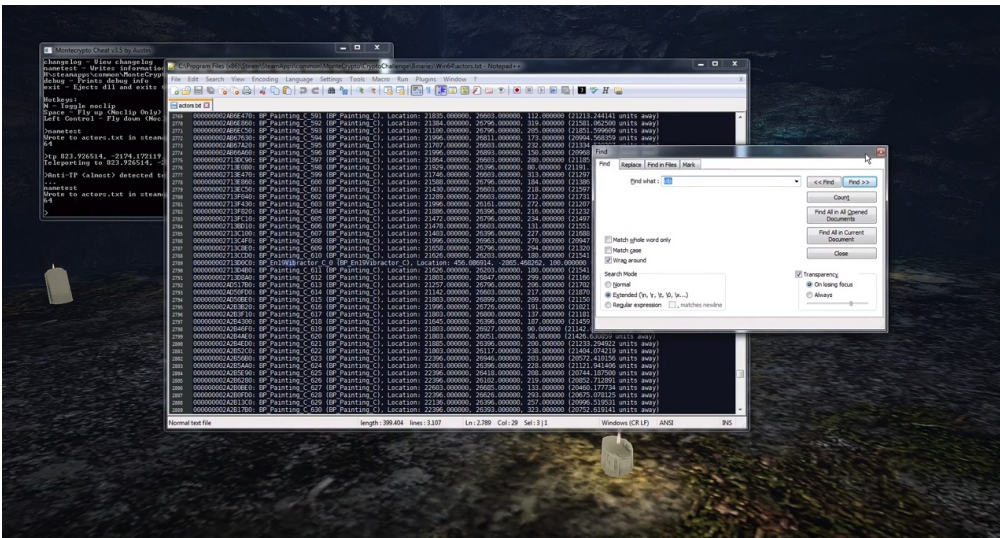
Since the room appears to have a cube-shaped forcefield around it, the working theory for a long time was you needed to use a VR rig. We'd already seem a puzzle that required a controlled, and we didn't think it was much of a stretch that other hardware might be required as well.

By default, the game doesn't boot up properly when either an Oculus or Vive headset is enabled in steam, but after some coercion with command line arguments, we got it to work. The game doesn't really support this, and making our way legitimately back to the cave was difficult since the UI for opening the trivia doors doesn't work properly with the headset on. Once we arrived there though, it didn't seem to trigger anything or make any difference. Finally, after laying down on the floor to "take a rest" with the VR gear on produced no changes, we gave up on that idea.

But, we knew that vibration was involved somehow. The same vibractor references in the game files that we had seen in Enigma 20 were in this blueprint as well. So we dropped the Vive and picked up the controller again.

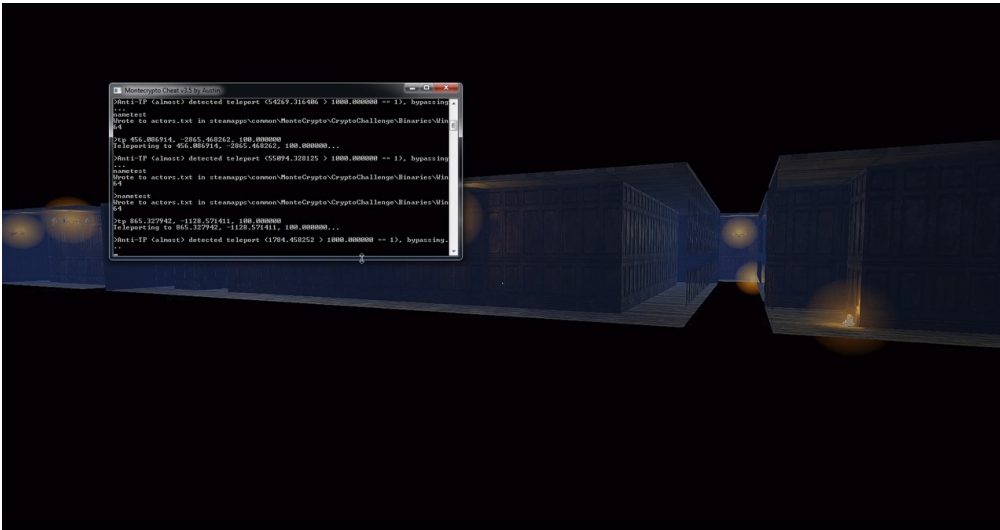
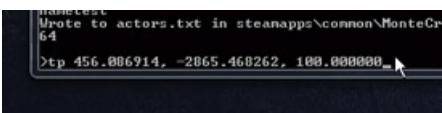
The breakthrough with this puzzle was when we were able to use a tool written by Austin to display every active entity that the game currently has spawned.

Using this tool, we could see that after entering the candle cave, there was one en19\_Vibractor active on the map. The problem? Its location was not only outside of the confines of the cave, but outside the entire maze.



```
000000002713C40: BP_Painting_C_006 (BP_Painting_C) Location: 21996.000000, 26563.000000, 270.000000 (26567.336644
000000002713C60: BP_Painting_C_009 (BP_Painting_C) Location: 21658.000000, 26796.000000, 294.000000 (21520.382813
000000002713C80: BP_Painting_C_010 (BP_Painting_C) Location: 21626.000000, 26203.000000, 198.000000 (21541.692500
000000002713D00: BP_En19Vibractor_C_0 (BP_En19Vibractor_C) Location: 456.866914, -2865.462622, 100.000000 (55094.32
000000002713D40: BP_Painting_C_011 (BP_Painting_C) Location: 21526.000000, 26563.000000, 270.000000 (21541.692500
000000002713D80: BP_Painting_C_012 (BP_Painting_C) Location: 21803.000000, 26567.000000, 299.000000 (21166.716797
000000002713E00: BP_Painting_C_013 (BP_Painting_C) Location: 21257.000000, 26796.000000, 295.000000 (21702.000000
```

Using Austin's tool to teleport to this location (making sure we have Noclip turned on also so that we don't fall to our death and reset) we hear the same glass smashing sound that we heard when solving Enigma 3.



At this point in the blackness of space, you see nothing, but repeating the process above we found that another vibractor had spawned at another unreachable location:

```
000000002713E40: BP_Painting_C_014 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002713E60: BP_Painting_C_015 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002713E80: BP_Painting_C_016 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002713EA0: BP_Painting_C_017 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002713EC0: BP_Painting_C_018 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002713EE0: BP_Painting_C_019 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002713F00: BP_Painting_C_020 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002713F20: BP_Painting_C_021 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002713F40: BP_Painting_C_022 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002713F60: BP_Painting_C_023 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002713F80: BP_Painting_C_024 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002713FA0: BP_Painting_C_025 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002713FC0: BP_Painting_C_026 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002713FE0: BP_Painting_C_027 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714000: BP_Painting_C_028 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714020: BP_Painting_C_029 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714040: BP_Painting_C_030 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714060: BP_Painting_C_031 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714080: BP_Painting_C_032 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027140A0: BP_Painting_C_033 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027140C0: BP_Painting_C_034 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027140E0: BP_Painting_C_035 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714100: BP_Painting_C_036 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714120: BP_Painting_C_037 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714140: BP_Painting_C_038 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714160: BP_Painting_C_039 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714180: BP_Painting_C_040 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027141A0: BP_Painting_C_041 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027141C0: BP_Painting_C_042 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027141E0: BP_Painting_C_043 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714200: BP_Painting_C_044 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714220: BP_Painting_C_045 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714240: BP_Painting_C_046 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714260: BP_Painting_C_047 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714280: BP_Painting_C_048 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027142A0: BP_Painting_C_049 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027142C0: BP_Painting_C_050 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027142E0: BP_Painting_C_051 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714300: BP_Painting_C_052 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714320: BP_Painting_C_053 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714340: BP_Painting_C_054 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714360: BP_Painting_C_055 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714380: BP_Painting_C_056 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027143A0: BP_Painting_C_057 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027143C0: BP_Painting_C_058 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027143E0: BP_Painting_C_059 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714400: BP_Painting_C_060 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714420: BP_Painting_C_061 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714440: BP_Painting_C_062 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714460: BP_Painting_C_063 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714480: BP_Painting_C_064 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027144A0: BP_Painting_C_065 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027144C0: BP_Painting_C_066 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
0000000027144E0: BP_Painting_C_067 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714500: BP_Painting_C_068 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (26567.336644
000000002714520: BP_Painting_C_069 (BP_Painting_C) Location: 27996.000000, 26563.000000, 270.000000 (265
```



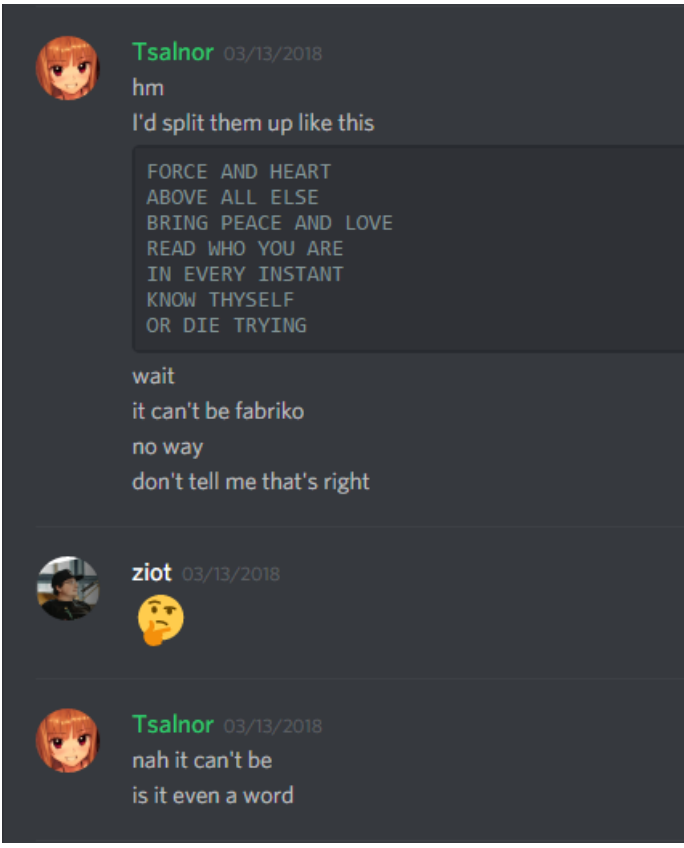
Credits to Tsalnor for the late night discussion that led to this solve with his breakthrough.

Each line with a number was telling you the length of which to split the string obtained from the morse output.

```
1
15 FORCE AND HEART
14 ABOVE ALL ELSE
20 BRING PEACE AND LOVE
16 READ WHO YOU ARE
16 IN EVERY INSTANT
12 KNOW THYSELF
13 OR DIE TRYING
```

Reading the first letter of each line gives you: 1FABRIKO.

To shine some light on Tsalnor, this mad man solved it initially without even using the tweet. We were just discussing how we would split each line up and he did it perfectly. This is how it felt to solve Enigmas in this game:



Solution: 1 fabriko

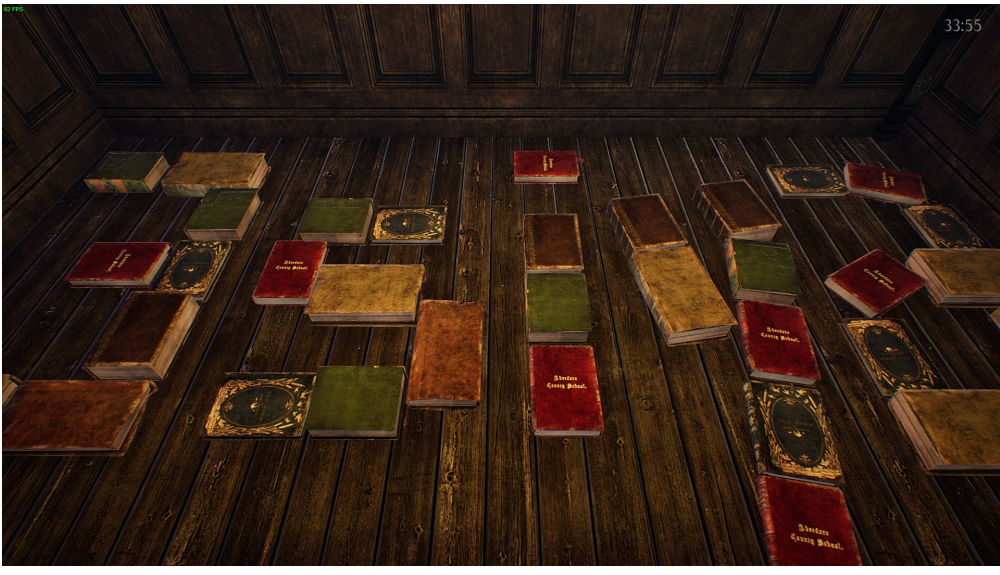
Enigma 21

Room Nickname: Vault

The Enigma for this Room was:

```
Everyday, the same routine...
Accuracy is what matters at the end of the day.
```

This was one of the biggest rooms, and ended up being solved by trying the same method used in the Skull room. You had to be in the room at the exact time of 03:33 am and one of the doors would open. On the ground there would be a set of books. At first the books do not look like anything. After you return at the same time the next few days, more books are added on the ground. Eventually you could read the next word "3siya."



Solution: 3 siya

Enigma 22

Room Nickname: Binary Window

This room was a relatively straightforward room to solve. The enigma states.

```
When back against the wall,
What seems like an obstacle,
Could very well be passable.
```

One of the hallways next to this terminal appears to be a deadend. There were some hidden doorways in the game that allowed you to run through the walls. This one however had collision and you could not run through it. You had to heed the tips of the terminal and move backwards to get through it.

Inside the room if you peer through the window you can see a string of binary (you had to do some guessing for the digits blocked by the window pane). But you should get: 00110100 01110101 01101011 01110101 01101110 01100111 01110001 01110101 01100010 01110101 01111010 01100001 01101110 01100001. Which translates to 4ukungqubuzana.



Solution: 4 ukungqubuzana

Enigma 23

Room Nickname: Painter

The Painter room was modeled after the Van Gogh painting [Bedroom in Arles](#).



Image credits: [GameDetective Wiki](#)

The terminal text read as the following:

```
Several suns are shining,
TEN of which are dying.
As our universe continues to expand,
You are here trying to understand.
Sent to us from an unknown entity,
Terrorising our friends and families
In TEN MINUTES he burnt our symphony;
La-Mar, the one and only,
Lurking in the sky and our city.
```

The first letter in each line spells out "STAY STILL." Given the TEN MINUTES in caps, it was clear the solve was to stand still and not move for 10 minutes. Doing so resulted in the solve being painted across the ceiling and wall.



Image credits: [GameDetective Wiki](#)

Solution: 4 bakar

Enigma 24

Room Nickname: Sun Cave

Enigma 24 is a small cave-like room with a god-ray sun shaft beaming down onto the enigma's terminal.







This was one that we solved some time before the public discord channels figured out what was going on and another example of how it was nerve racking and frustrating to watch them catch up as we struggled on the final two puzzles in the rain room and forest/outside.

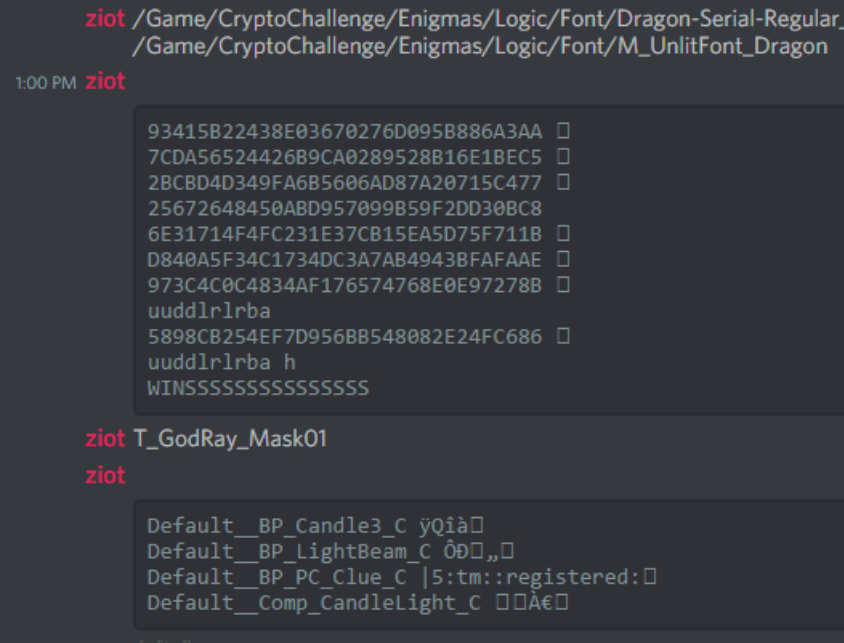
The text on the console reads:

```
I remember it was a code...
Yes that's it: "lake waves"
Was it in German? Or Chinese?
Not sure, I can't seem to remember.
```

From this clue we knew that we needed to translate the "lake waves" text into some language that made more sense to the puzzle that likely wasn't German or Chinese.

We made the connection pretty quickly after someone called out that "waves" in Japanese is "nami" and that instantly connected us with the word "code" in the clue to the Ko nami Code.

1:31 PM **motive @Luigy** "code" + "lake waves" = konami code. waves is nami in Japanese (edited)



We also almost simultaneously noticed the Konami code listed in the bp files

1:02 PM **ziot** uuddlrIrba = konami code  
**ziot** which is in there

Entering into the room and typing out the Konami code using the arrow keys (udlr) and 'b' and 'a' on the keyboard played the success sound and triggered the message:



**Solution: 9 gull**

## Final Solve Steps / MonteCrypto Solution

Having the pi sequence order, number-word solutions, and wallet files. We ran Hashcat with all of the possible permutations until we got a hit. Using that, we were able to use the Bitcoin Core to extract the Bitcoin prize out of the wallet.

Proof:

<https://live.blockcypher.com/btc/tx/b02f1aea315f677d98427ca0bd8c48a3cc5a7225a72436b1dfc2c163f69e02f0/>

en#	pi (021-144) - answer order	english
13	1 0 0-construct	Construct
9	2 9 9-ogles	Coal
4	3 3-kohokohta	Highlight
7	4 8 8-perro	Dog
22	5 4 4-ukungqubuzana	Conflict
23	6 4 4-bakor	Burn
8	7 6 6-okpu	Hot
18	8 0 0-nodo	Node
14	9 9 9-istisna	Exception
1	10 5 5-mzuzwana	Moment
2	11 5 5-pagluwa	Rain
19	12 2 2-reprodukcja	Tie
6	13 5 5-persamaan	Equation
16	14 8 8-optree	Behave
3	15 2 2-kuhinga	Article
5	16 2 2-andiep	Shallow
17	17 3 3-prevoditi	Translate
20	18 1 1-fabrika	Factory
10	19 7 7-beunghar	Rich
12	20 2 2-reprodukcja	Reproduction
11	21 5 5-ola	Wave
21	22 3 3-siya	Saddle
15	23 5 5-velte	Overturn
24	24 9 9-gull	Gull

The final password and solve for MonteCrypto:

construct istisna prevoditi optree bakar ukungqubuzana okpu colle gull velte mzuzwana nodo persamaan perro tuinga reprodukcja siya fabrika beunghar ondiep ola kohokohta pagluwa ogles

See [Appendix A - Wallet brutin](#) for how we bruteforced the final solution from all the possible permutations from the above list.

## ADDITIONAL FLAVOR PUZZLES

Though we did not know this until much later in the solving process after lots of time had been sunk into them, the following puzzles and findings in the maze appear to be lore / flavor only and are note unnecessary for solving the main enigmas.

### Compasses and Skulls

The game included small compasses you could click on that would play audio files. There were also shrines that had skulls on them that would play lore audio when you clicked them. Both of these turned out to be entirely for achievements, game lore, or ciphered troll messages that did not appear to be related to the puzzles for the wallet. We're not going to bother writing up information on them due to this.

### Appendix A - Wallet brutin

#### Setup

john1.8j1-bc1bbc96f (john-jumbo bleeding) and 3 previous versions were used to convert wallet.dat, the zip files, and the dmg file to a brutable hash format for both John The Ripper (JTR) and hashcat. These are bitcoin2john.py, zip2john.exe, and dmg2john.exe.

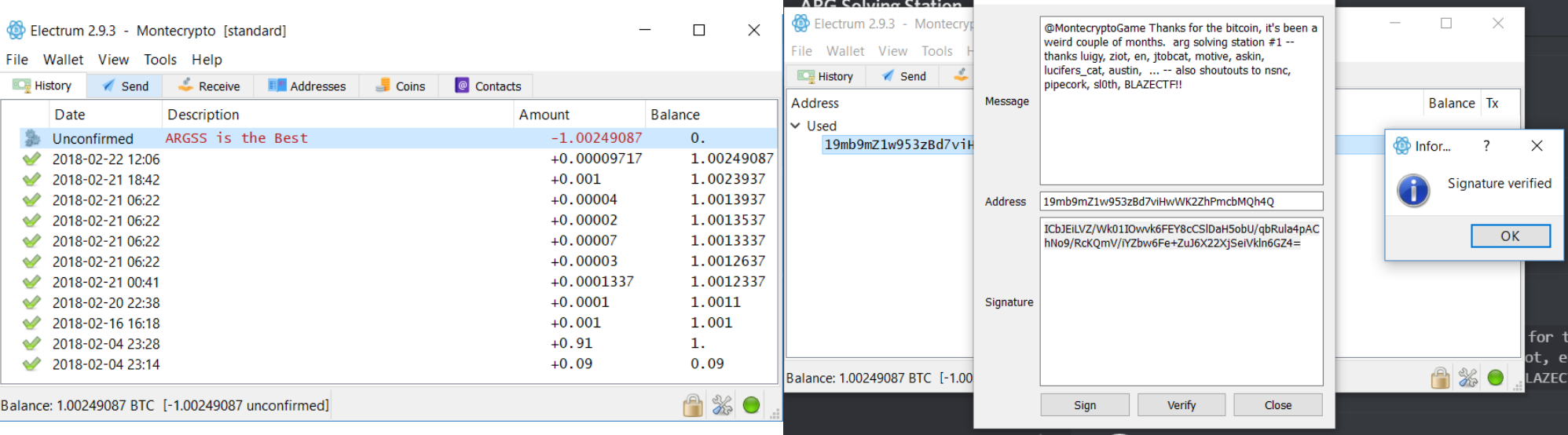
#### Generating wallet passwords

We weren't sure on a few words. We also weren't sure about the final password being the order of the enigmas. Here's the script we wrote to generate those permutations:

<http://archive.is/g7mk9>

Here's what the final solve looked like:

```
PS H:\> C:\Python27\python.exe gen.py
[+] Generating pi-order enigma-suborder passwords with lower/uppercase
[+] Complete: 6 passwords
construct ogles kohokohta perro ukungqubuzana bakar okpu colle istisna mzuzwana pagluwa nodo p
construct ogles kohokohta perro ukungqubuzana bakar okpu colle istisna mzuzwana pagluwa noto p
construct ogles kohokohta perro ukungqubuzana bakar okpu colle istisna mzuzwana pagluwa novo p
CONSTRUCT OGLES KOHOKOHTA PERRO UKUNGQUBUZANA BAKAR OKPU COLLE ISTISNA MZUZMANA PAGLUWA NOOO P
CONSTRUCT OGLES KOHOKOHTA PERRO UKUNGQUBUZANA BAKAR OKPU COLLE ISTISNA MZUZMANA PAGLUWA NOTO P
CONSTRUCT OGLES KOHOKOHTA PERRO UKUNGQUBUZANA BAKAR OKPU COLLE ISTISNA MZUZMANA PAGLUWA NOVO P
[+] Generating words by number sets for each possibly confused word like ['nodo', 'nodo']
[!] We generated 3 sets based on 'unsure' words -- that's 3*311040=933120 possible passwords!
pop, result set: 622086
memo set: 622086
pop, result set: 1244166
memo set: 1244166
pop, result set: 1866246
memo set: 1866246
[+] Final password list size: 1866246
----- wrote to passlist.txt -----
PS H:\>
PS H:\> cd .\hashcat-4.0.1\
PS H:\hashcat-4.0.1> .\hashcat64.exe -m 11300 wallet.txt H:\passlist.txt
hashcat (v4.0.1) starting...
Session.....: hashcat
Status.....: Running
Hash.Type.....: Bitcoin/Litecoin wallet.dat
Hash.Target.....: $bitcoin9965679c108b960fa1e8185d06d7d42df1f970555a0...e5b0a
Time.Started.....: Mon Apr 23 20:33:53 2018 (30 secs)
Time.Estimated.....: Mon Apr 23 21:03:27 2018 (29 mins, 14 secs)
Guess.Base.....: File (H:\passlist.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....: 1065 H/s (9.34ms)
Recovered.....: 0/1 (0.00%) Digests: 0/1 (0.00%) Salts
Progress.....: 26624/1866246 (1.43%)
Rejected.....: 0/26624 (0.00%)
Restore.Point.....: 26624/1866246 (1.43%)
Candidates.#1.....: construct istisna prevoditi perro ukungqubuzana bakar okpu co
truct istisna kohokohta optree ukungqubuzana bakar okpu colle gull velte ola nodo
Mon.Dev.#1.....: Temp: 68c Fan: 33k Util: 99% Core:1316MHz Mem:3004MHz Bus:16
$bitcoin9965679c108b960fa1e8185d06d7d42df1f970555a046ed4c30e701e53843c65d56d162
37373224d8588f468395f35566503ff49e39b184737935593974d9be6d613d6bb9aa186f6c6357
reprodukcja siya fabrika beunghar ondiep ola kohokohta pagluwa ogles
Session.....: hashcat
Status.....: Cracked
Hash.Type.....: Bitcoin/Litecoin wallet.dat
Hash.Target.....: $bitcoin9965679c108b960fa1e8185d06d7d42df1f970555a0...e5b0a
Time.Started.....: Mon Apr 23 20:33:53 2018 (50 secs)
Time.Estimated.....: Mon Apr 23 20:34:43 2018 (0 secs)
Guess.Base.....: File (H:\passlist.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....: 1065 H/s (9.19ms)
Recovered.....: 1/1 (100.00%) Digests: 1/1 (100.00%) Salts
Progress.....: 53248/1866246 (2.85%)
Rejected.....: 0/53248 (0.00%)
Restore.Point.....: 26624/1866246 (1.43%)
Candidates.#1.....: construct istisna prevoditi perro ukungqubuzana bakar okpu co
truct istisna kohokohta optree ukungqubuzana bakar okpu colle gull velte ola nodo
Mon.Dev.#1.....: Temp: 70c Fan: 33k Util: 94% Core:1316MHz Mem:3004MHz Bus:16
Started: Mon Apr 23 20:33:17 2018
Stopped: Mon Apr 23 20:34:45 2018
```



#### Stats

We ran the generator to brute a bunch of ideas that didn't pan out for enigma words. We also ran JTR against the zip files with multiple wordlists + dive rulesets. At one point we ran strings against all extracted assets and passed that as a wordlist (this solves the first two zips).

Discussed below, we ran our B&K attack on iteratively lower numbers of known plaintext on 100 cores for 48 hours. Only the three keys were recovered.

### Appendix B - Biham and Kocher's Known Plaintext Attack

Enigma 3 and 4 produced passwords to two zip files in MonteCrypto. Having only one challenge left (Enigma 2), we had one zip without a password as well. This is "dexterandalmar.zip," it contained a file named "important.jpg". By viewing the file headers, we could identify that it was made using p7zip-full on a unix distro. frVersion was 0x314, but the zip format does a bitwise and of this value with 0xFF to determine the proper version of 0x14. So after [googling around](#) for the number, it's apparently the bitwise or of 0x14 and 0x300 -- 0x300 indicating it's from the unix distro of 7zip. Who would've thought.



dexterandalmar.zip																	
#	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000b:	50	4B	03	04	00	01	00	08	00	F4	A6	44	4C	56	C2	PR.....01DLVA	
0001b:	AA	C1	2A	EE	02	60	68	23	03	00	00	00	00	00	69	6D	FA-1...4+...-18M
0002b:	70	6F	72	74	61	6E	74	3E	6A	70	67	82	5B	6E	6C	91	portant.jpg.inl>
0003b:	DA	6A	F3	AC	C5	BC	95	E3	18	A7	7D	CE	7D	E8	AA	FE	0j0-4+*â..ÿ1i*þ
0004b:	86	4C	4D	B8	36	7B	8D	66	13	3E	0F	BF	C7	0B	B4	63	f1M.61.¿.>..¿¿.¿
0005b:	61	EE	76	73	AE	OC	BF	F3	65	DA	35	06	EE	E9	CC	F5	â1vââ..1ââ1v.âââ
0006b:	FA	27	88	61	1E	71	47	0A	68	84	2E	2E	3E	87	0A	0A	At>1vâ¿ f.ââvâvâ
Template Results - ZIPTemplate.bt																	
Name		Value	Start	Size	Color	Comment											
struct ZIPFILE record		important.jpg	0h	2EE5h	Fg: Bg:												
char fSignature[4]		PK��	0h	4h	Fg: Bg:												
ushort fVersion		788	4h	2h	Fg: Bg:												
ushort fFlags		1	6h	2h	Fg: Bg:												
enum COMPTYPE fCompression		COMP_DEFLATE (8)	8h	2h	Fg: Bg:												
DOSTIME fFileTime		20:55:40	Ah	2h	Fg: Bg:												
DOSDATE fFileDate		02/04/2018	Ch	2h	Fg: Bg:												
uint fCrc		C1AAC258h	Bh	4h	Fg: Bg:												
uint fCompressedSize		192042	12h	4h	Fg: Bg:												
uint fUncompressedSize		207716	16h	4h	Fg: Bg:												
ushort fHeaderLength		13	1Ah	2h	Fg: Bg:												
ushort fExtraFieldLength		0	1Ch	2h	Fg: Bg:												
char fFileName[13]		important.jpg	1Eh	0h	Fg: Bg:												
uchar fCName[22042]			28h	2EE2Ah	Fg: Bg:												

Biham and Kocher's Known Plaintext attack is the first thing that comes to mind when seeing pkzip-encrypted files. The idea is simple -- if you know 13 bytes of the original file before encryption, you can apply this attack to brute force the 12 byte encryption key. How many bytes of the original file do we know? Well, we collected every JPEG we could find associated with montecrypto. Here's what the header bytes of some of them look like:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	FF	E0	00	10	4A	46	49	46	00	01	01	00	00	48			FF..JFIF.....H
0010h:	00	48	00	00	00	00	00	58	45	79	69	66	00	4D	4D		..FF..XEXIF..MM
0020h:	00	2A	00	00	00	00	00	00	01	12	03	00	00	00	01		.....
0030h:	00	01	00	00	87	63	00	04	00	00	00	00	01	00	00	26	.....
0040h:	00	00	00	00	00	03	A0	01	00	03	00	00	00	01	00	01	.....
0050h:	00	00	00	00	00	00	00	00	00	01	00	00	00	00	00	00	.....

Template Results - JPEGTemplate.bt

Name	Value	Start	Size	Color
struct JPGFILE jpgfile		0h	18D3BEh	Fg: Bg:
enum M_ID SOIMarker	M_SOI (FFD8h)	0h	2h	Fg: Bg:
struct APP0 app0		2h	12h	Fg: Bg:
enum M_ID marker	M_APP0 (FFEOh)	2h	2h	Fg: Bg:
WORD szSection	16	4h	2h	Fg: Bg:
char App0Type[5]	JFIF	6h	5h	Fg: Bg:
short versionHigh : 8	1	Bh	2h	Fg: Bg:
short versionLow : 8	1	Bh	2h	Fg: Bg:
ubyte units	0	Dh	1h	Fg: Bg:
WORD Xdensity	72	Eh	2h	Fg: Bg:
WORD Ydensity	72	10h	2h	Fg: Bg:
ubyte xThumbnail	0	12h	1h	Fg: Bg:
ubyte yThumbnail	0	13h	1h	Fg: Bg:

You can immediately see there is a "JFIF" header, followed by an "Exif" header. Looking at all of the JFIF headers from JPEGs related to MonteCrypto, the first 21 bytes had only 3 variations. So we probably know 21 bytes of the original file, right?

Not quite.

The "plaintext" or original file here is compressed before it is encrypted with PKZIP. So those 21 bytes represent a smaller number of bytes before encryption. You might think -- oh, well just try to zip those bytes without a password! That could get you the bytes before encryption. However, 7zip and friends have an optimization to prevent "compressing" files that are too small to be effectively compressed by the DEFLATE algorithm. It won't even attempt to compress the data.

Your options are now:

- Write your own DEFLATE implementation.
- Patch that optimization out and recompile.
- This was a pain and a waste of time.
- Find a DEFLATE library that will actually compress the data.
- Python ended up doing it with some coercion

This is the part where there's more bad news. DEFLATE compressed data is different depending on the bytes that follow. We don't know what bytes could follow our 21 bytes. So, if we just append some null bytes.... Python will give us something that looks correct.

```
>>> import zlib
>>> with open("bytes.bin","r") as fh: dat = fh.read(); do = zlib.compressobj(6, zlib.DEFLATED, -15); d = do.compress(dat+"x00"*30);

>>> print(repr(dat))
```

We actually notice several more possible variations of JPEG header (such as JPEGs that start with EXIF before JFIF). So we are down to a few possibilities:

```
03/13/2018 04:58 21 21bytes_a_compressed
03/13/2018 04:58 PM 21 21bytes_c_compressed
03/13/2018 04:59 PM 21 21bytes_e_compressed
03/13/2018 04:58 PM 35 exif_40_bytes_compressed
03/13/2018 05:01 PM 46 exif_64_bytes_presskitsandgithub_compressed
03/13/2018 04:49 PM 14 thirteen_bytes_b_compressed
03/13/2018 04:48 PM 14 thirteen_bytes_compressed
03/14/2018 10:26 PM 13 twelve_compressed
8 File(s) 185 bytes
```

So, we have our compressed "plaintext" bytes! Now we can use a pre-written tool such as [bkcrack](#) or [yazc](#). These tools implement the B&K attack to reduce the space of possible encryption keys. They then attempt each, and see if they match up to the known plaintext. Plug everything in, bkcrack will quickly produce a key:

```
[Tue Mar 13 18:29:07 DST 2018] Starting /compressed_testcases/21bytes_e_compressed
Generated 4194304 Z values.
[18:29:07] Z reduction using 9 extra bytes of known plaintext
100.0 % (9 / 9)
712488 values remaining.
[18:29:08] Attack on 712488 Z values at index 11
63.8 % (454229 / 712488)
[18:42:34] Keys
e91edb7c 6272c283 70857046
```

That's a valid decryption key! We use bkcrack to dump the decrypted contents, which we then need to monkey-patch into the original 7zip to coerce it to decompress. However, trying it with that key will quickly show you the problem, when DEFLATE fails:

```
$ ./bkcrack -c important.jpg -C ./dexterandalmar.zip -k e055bf07 30532d26 97eaa917 -d compressed_important_2
[22:14:31] Keys
e055bf07 30532d26 97eaa917
Wrote deciphered text.
$ 7z e /mnt/e/fixe3.zip; ls -lhH important.jpg; xxd important.jpg
7-Zip [64] 9.20 Copyright (c) 1999-2010 Igor Pavlov 2010-11-18
p7zip Version 9.20 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,12 CPUs)

Processing archive: /mnt/e/fixe3.zip

Extracting important.jpg Data Error

Sub items Errors: 1

-rw-rw-rw- 1 a a 48 Feb 4 19:55 important.jpg
00000000: ffd8 ffe0 0010 4a46 4946 0026 4f1b 4898 .....JFIF.&O.H.
00000010: 4898 4898 4898 4898 4898 4898 4898 4898 H.H.H.H.H.H.H.H.
00000020: 4898 4898 4898 4898 4898 4898 2409 8ed2 H.H.H.H.H.H.$...
```

That's all the data it could dump. It doesn't look like the correct bytes to follow a JFIF header. That's because the decryption key was only valid for the first X bytes of plaintext we passed bkcrack.

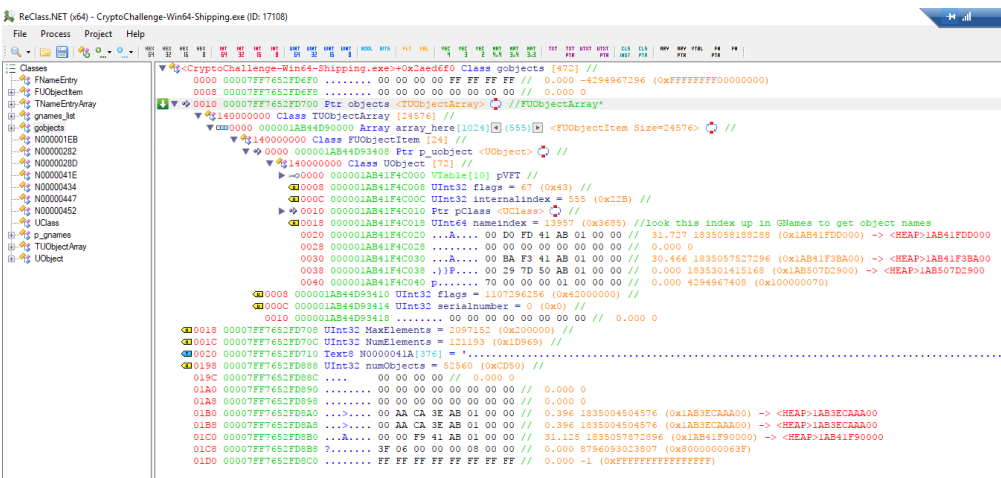
So there can be more than one key that decrypts to a tiny portion of known plaintext. We modified bkcrack to not stop on the first key found, and we modified it to "work" on zip files smaller than 13 bytes (the keyspace is just much, much larger with more false positives). We pointed 100 cores at the problem and came up with these 3 keys:

```
e91edb7c 6272c283 70857046
e055bf07 30532d26 97eaa917
b9712087 33594506 a3b8ed2
```

None of which decrypt the entire DEFLATE-compressed JPEG correctly. Sad end 🙄

## Appendix C - Disassembling UE4 Blueprints

It's [literally done for you](#) (link requires UE4 source code access or it'll say 404 -- request access on UE's site) . You can iterate over a global array of UObject's, check if they're a UStruct, and then pass the disassembler their script pointer. Here's an in-memory screenshot of what that array looks like:



A pointer to the global objects array can be found by searching for this pattern in the game in IDA Pro: 48 8D 05 ?? ?? ?? ?? 48 89 01 33 C9 84 D2 41 8B 40 08 49 89 48 10 0F 45 05 ?? ?? ?? ?? FF C0 49 89 48 10 41 89 40 08

UStruct is the superclass that contains the pointer to blueprint bytecode as a TArray<uint8> -- which is effectively a byte array:

<https://github.com/EpicGames/UnrealEngine/blob/4.18/Engine/Source/Runtime/CoreUObject/Public/UObject/Class.h#L236>

## Appendix D - important.jpg



## Appendix E - Our Thoughts on Hacking the Client

For some time after the game's release our group was torn - should we hack the client to bend it to our will for the sake of solving and getting around the maze more quickly - or is that immoral and breaks the spirit of fair competition.

Just a few days after release though we realized the following:

- The game literally has an achievement that you can only get via falling through the world, something you could never do without noclip or teleporting
- The game has a wall that it explicitly encourages you to teleport or noclip through
- The rain room. The first half of the solve for this room is to teleport up to the listed value on the z axis.

Once we saw all three of these existed we felt like the developers of the game not only expected players to hack the client, but they required it to actually solve the full gambit of puzzles.

From that point forward, the only way to be on the same playing field as our competition, let alone win, was to embrace that hacking the client was an expected part of the race.

## Escalating XSS in PhantomJS Image Rendering to SSRF/Local-File Read

## TWITTER FEED

Tweets by @bbuerhaus

## TAGS

1o57 admin [airbnb](#) anime application security appsec badge\_challenge bounty bounty programs bug bounty burp co9 cross-site request forgery cross-site scripting crypto CSAW csrf css [CTF](#) defcon defcon22 defcon23 detection facebook flickr



RECENT POSTS

- Montecrypto – ARGSS Write-Up
- Escalating XSS in PhantomJS Image Rendering to SSRF/Local-File Read
- Airbnb – Web to App Phone Notification IDOR to view Everyone's Airbnb Messages
- Airbnb – Ruby on Rails String Interpolation led to Remote Code Execution
- Airbnb – Chaining Third-Party Open Redirect into Server-Side Request Forgery (SSRF) via LivePerson Chat

ARCHIVES

- April 2018
- June 2017
- March 2017
- August 2016
- June 2016
- May 2016
- April 2016
- September 2015
- August 2015
- February 2015
- January 2015
- October 2014
- August 2014
- July 2014
- June 2014
- April 2014