A Forensic Investigation into the Murder of Chance

Conducted by b1c

Introduction

This forensics report presents an analysis of the evidence collected in relation to the murder of Chance. The victim, Chance, was found dead[1] on the morning of March 6th, 2020, at 2 am with severe head trauma injuries. The perpetrator's modus operandi — a large stainless steel frying pan — was found on the scene near the victim's body. Prior to our investigation, the circumstances and motives surrounding the murder were unclear, and the perpetrator was undetermined.

Our report provides a comprehensive analysis of the evidence, including the forensic analysis of the crime scene, and other relevant information that reveals the identity of the perpetrator and the circumstances surrounding the murder. The findings of this report detail the murderer's identity, choice of weapon, and motive. We believe this report is crucial to bringing justice to the victim and their loved ones.

The loss of Chance has had a profound impact on the community, and it is our hope that through our thorough investigation and the diligent pursuit of justice, we can honor his memory and bring closure to his loved ones.

Methodology

Tweepy Public Timeline

We have access to the Tweepy account of "sadamana", which allows us to access the public tweeps of Tweepy accounts "chance", "corncob", "long", "karst", and "hubbz". We will refer to these accounts as synonymous with the respective user.

Tweepy homepage: `flag{you_found_twitter}`

Immediate evidence

Karst 2/29/20

```
dSA9IGthcnN0LCBwYXNzID0gRVNTX2Ywcl90aDNfdzFuCmZsYWd7Y3IzZDVfNHIzX3
U1M2Z1MV93MHd9
```

The message that Karst sends appears to be in base64, so we used CyberChef to decode it.

```
u = karst, pass = ESS_f0r_th3_w1n
Flag{cr3d5_4r3_u53fu1_w0w}
```

Flag: `flag{cr3d5_4r3_u53fu1_w0w}`

Through some testing, we find that the plaintext is the username and password for Karst's Tweepy account.

```
sw mywsxq kpdob iye pvkq{k_fobi_lkcsm_mkockb_mszrob}
```

Long 2/21/20 (rot16)

It looks like the last part of the text that long sends is already in flag format. Assuming `pvkq` maps to `flag`, we determine that this flag is encoded with ROT16, which is a simple caesar cipher with a rotation of 16 characters. We run this through CyberChef with recipe `ROT13` with amount 16 to get the following plaintext:

```
im coming after you flag{a_very_basic_caesar_cipher}
```

Flag: `flag{a_very_basic_caesar_cipher}`

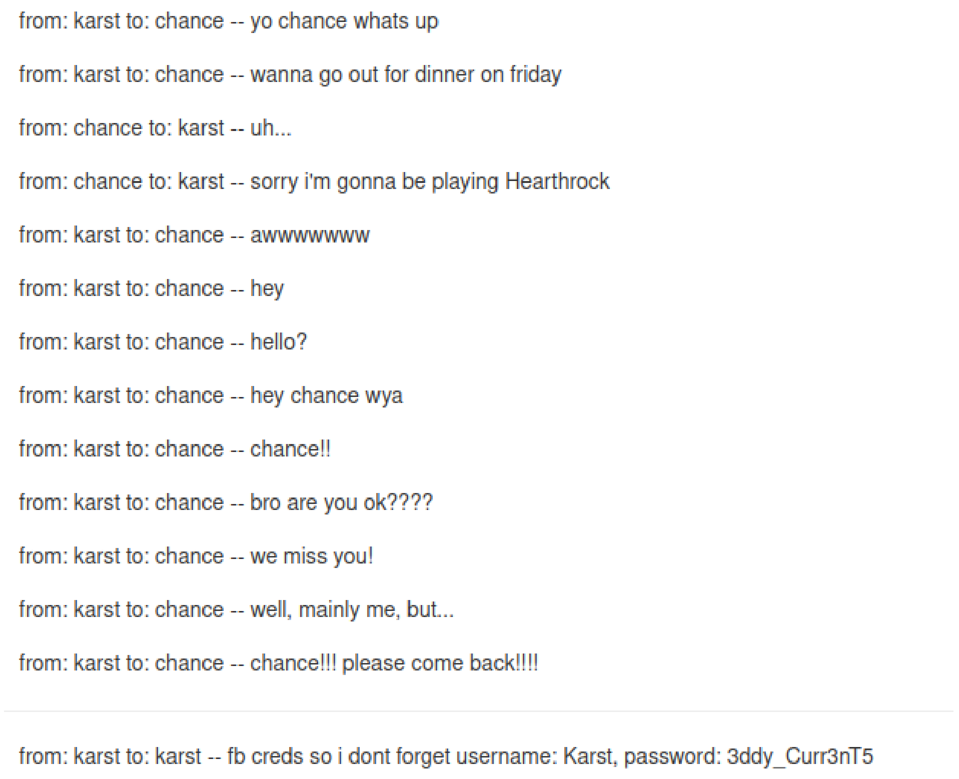
Karst's Address (posted by karst, 2/15/20)

OMG THANK YOU SO MUCH GUYS I'm at 72935 Topography St. btw

From this message, we can see that Karst lives at 72935 Topography Street. Do not share your address with strangers online. Especially not on a public forum, like Tweepy.

Karst's Fesbuc password

Beside the primary timeline, Tweepy also has a Direct Messages feature. We logged into Karst's account using the credentials he posted on the main timeline.



from: karst to: chance -- yo chance whats up

from: karst to: chance -- wanna go out for dinner on friday

from: chance to: karst -- uh...

from: chance to: karst -- sorry i'm gonna be playing Hearthrock

from: karst to: chance -- awwwwwww

from: karst to: chance -- hey

from: karst to: chance -- hello?

from: karst to: chance -- hey chance wya

from: karst to: chance -- chance!!

from: karst to: chance -- bro are you ok????

from: karst to: chance -- we miss you!

from: karst to: chance -- well, mainly me, but...

from: karst to: chance -- chance!!! please come back!!!!

from: karst to: karst -- fb creds so i dont forget username: Karst, password: 3ddy_Curr3nT5

From here, we were able to find Karst's Fesbuc credentials as well.

Further links:

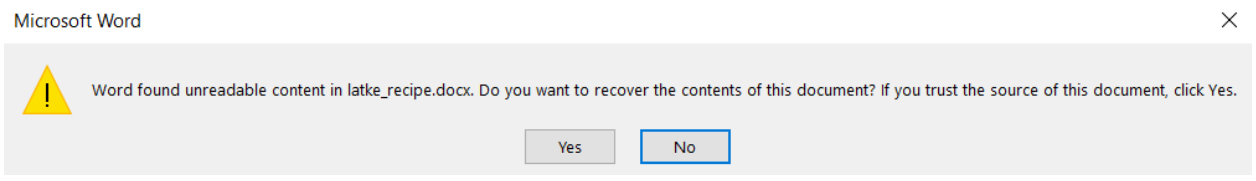
Chance's blog (posted by chance, 2/20/20)
Corncob's website (posted by corncob, 3/1/2020)
Fesbuc website (posted by hubbz, 3/2/2020)

Chance's Blog

Previously we found Chance's blog, which he posted about on Tweepy. It can be accessed at http://jekyll-blog.chals.mcpshsf.com.

Here's a recipe! - 2/20/20

Link: http://jekyll-blog.chals.mcpshsf.com/post/2014-06-08-post2/
We are given a latke recipe .docx file (a Word document). When trying to open it in Microsoft Word, we are greeted with the following warning.



Word found unreadable content in latke_recipe.docx. Do you want to recover the contents of this document? If you trust the source of this document, click Yes.

This suggests that there is something wrong with the document. However, the document just seems to be a recipe for latkes (Appendix A). Fortunately, we know that the .docx file format is essentially a glorified zip archive, so we unzipped the document to find `flag.txt` under the subdirectory `word`, which gave us the flag.

Flag: `flag{00h_l4tk3s!}`

Accidentally locked important information - 2/3/20

Link: http://jekyll-blog.chals.mcpshsf.com/post/2014-06-08-post3/
There were two parts to this challenge. The first involved breaking a physical bike lock.

Even though the bike lock only has 6^4 possible states, we did not brute force the correct combination. Instead, we found the correct combination by trying each rotor individually. A design flaw in these bike locks is that it is possible to create a noticeable gap between the rotors by putting the lock into tension. The key insight is that this gap appears on the side nearest the pin, and choosing a single digit correctly will shift the guessed rotor to the side of the pin and cause the gap to shift away from the pin by one. Therefore, it becomes possible to apply a byte-by-bytes oracle brute-force approach. Thus, we chose the correct digit for each rotor independent of the others starting from the right side, effectively reducing our search space size from 6^4 combinations to 24 and allowing us to quickly open the lock.

Flag: `flag{l33t_l0ckp1ck1ng_br0}`

We are also given `ilovelatkesalot!`, a key for the second part of the challenge, which was to decrypt an AES ciphertext. We were pretty certain that it was an AES ciphertext because the decoded length was 128 bytes, which AES' block size (16 bytes) divides. Further, the key length was 16 bytes. However, we did not know the mode of operation of AES. See Appendix C for more information on AES and modes of operation.

We used this tool to decrypt the ciphertext online. Because there was no IV provided, we initially thought that the mode had to be ECB as it is the only AES mode that does not require an IV. However, we ended up trying CBC where the IV was the key itself, which properly decoded the flag and also revealed a link to a corrupted file.

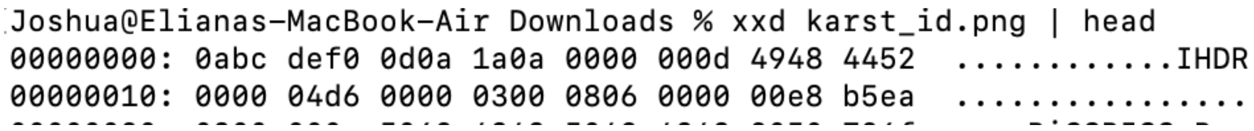
Our plaintext:

```
)òç/k²Id¹¿3...½https://drive.google.com/file/d/1sLaQw-
kMYGhexxUV12y09RJj_blVLzNz/view?usp=sharing flag{f0ll0w_th3_url!}
```

Flag: `flag{f0ll0w_th3_url!}`

Karst's ID

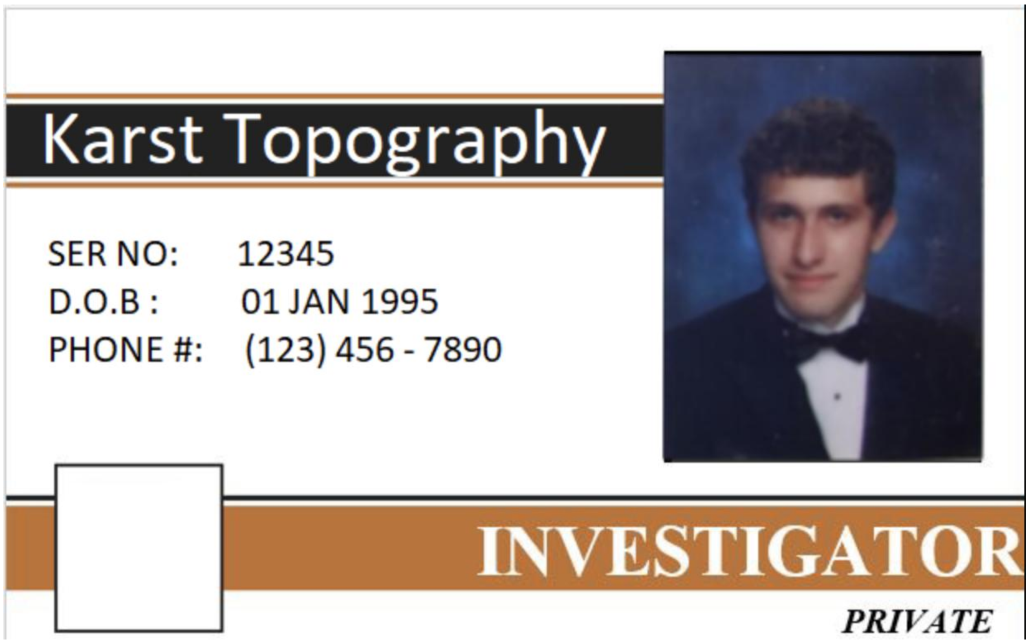
We downloaded the file, but were unable to identify what type of file it was based on the header (the initial starting sequence of bytes at the beginning of the file).



```
Joshua@Elianas-MacBook-Air Downloads % xxd karst_id.png | head
00000000: 0abc def0 0d0a 1a0a 0000 000d 4948 4452  ............IHDR
00000010: 0000 04d6 0000 0300 0806 0000 00e8 b5ea  ................
```

The header looks as if it had been tampered with, as there is a suspicious sequence of abcdef.

However, we spotted an IEND chunk at the end of the image, signifying that the image was a PNG. To repair the image, we fixed the header using the standard header bytes of a PNG file as specified in the PNG specification.


Karst Topography
SER NO: 12345
D.O.B : 01 JAN 1995
PHONE #: (123) 456 - 7890
INVESTIGATOR
PRIVATE

Sus log info

Link: http://jekyll-blog.chals.mcpshsf.com/post/2014-06-08-post4/
We are given an access log, which logs a bunch of GET requests, presumably to a login page.

The log seems to contain the requests from an SQL injection attempt through the user URL parameter. Using a regex extraction and URL-decoding the username with this CyberChef recipe, we arrived upon the raw SQL injection payloads. However, we were not able to recover any useful information from the logs, as every string seemed to be randomly generated. Further, there is evidence of a timing attack being started at the end, but the logged request times do not indicate that. Thus, we did not pursue this challenge further.

Corncob’s website

Site: http://corncobs-sus-website.chals.mcpshsf.com
The hint about robots leads us to check robots.txt, which is a common file for websites to have in order to inform what web crawlers should index. Accessing /robots.txt gives us the flag and also a username and password combination, which we later verified to be for Tweepy.

`https://corncobs-sus-website.chals.mcpshsf.com/robots.txt`

```
User-agent: *
Disallow: flag{domo_arigato}
User: corncob, pass: L4tk3_M4f14_L0rd_123
```

Flag: `flag{domo_arigato}`
Credentials: corncob: L4tk3_M4f14_L0rd_123

Fesbuc website

Site: http://facebook-django.chals.mcpshsf.com

We logged in with Karst's password as recovered from Immediate evidence.

Further links:
Long's Madlibs site (posted by Long, 3/6/20)
LSB Frying Pan (posted by Hubbz, 3/6/20)
Robot images (posted by Karst, 3/6/20)

CornCob's rant photo (posted by CornCob, 3/6/20)
Karst's file sharing website (posted by Karst, 3/6/20)
Crime Scene Photo (posted by Police, 3/6/20)

Long’s Madlib site

Link: http://madlibs.chals.mcpshsf.com
This site allows us to fill in a Madlibs template with arbitrary values. All fields are vulnerable to Jinja2 Server-Side Template Injection. We verified this by submitting a sample payload of {{7+7}}, and we got “14” back.

By using a more advanced payload, we can achieve arbitrary remote code execution:
{{self.__init__.__globals__.__builtins__.__import__('os').popen('ls').read()}}

In the current directory of the server, there are two notable files, flag.txt and letter.txt.

Flag: `flag{th1s_1s_a_t3mp14t3_f0r_fl4sk5_4nd_inj3ct1on5}`

Letter:
Dear Diary, I can't believe how upset I am right now. I asked Chance for some Latkes and he never responded! I was really looking forward to enjoying those crispy, delicious potato pancakes, and I was practically salivating at the thought of smothering them with sour cream and applesauce. Now, I'm sitting here in my kitchen, feeling sorry for myself, and hungry as can be. I can't stop thinking about those latkes, and how amazing they would have tasted if only I had been able to get my hands on some potatoes. It's not fair, really. I mean, who runs out of latkes? I know it's silly to get so worked up over something as simple as food, but I can't help it. I was really looking forward to those latkes, and now I'm just mad that I didn't get to enjoy them. I guess I'll just sulk in my room and try to forget about those delicious, crispy latkes that I missed out on. I might have to hit something if I don't get those latkes.

LSB Frying Pan

Link: http://chal-host.chals.mcpshsf.com/chal-img.png

From the title, we are hinted towards using Least Significant Bit steganography. We can use this tool to extract the flag and relevant information.

Flag: `flag{l0v3ly_sup3r_b3st_3nc0d1ng}`

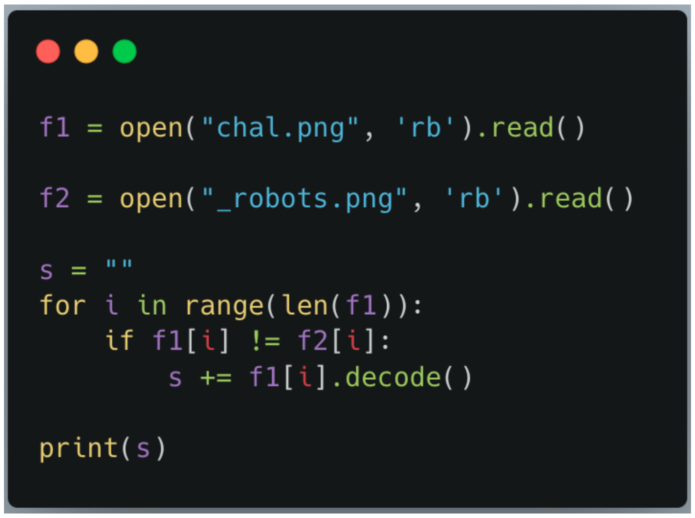
Karst's username: `k4r5t_t0p0gr4phy`

Robot images

Link: http://chal-host.chals.mcpshsf.com/_robots.png, http://chal-host.chals.mcpshsf.com/chal.png

We are given two images, which appear to be the same, except the second half of chal.png is gone. Given that _robots.png and chal.png are similar in the beginning, we chose to find the byte differences between the files.

Using the following script, we were able to extract the flag and relevant information.



```
f1 = open("chal.png", 'rb').read()

f2 = open("_robots.png", 'rb').read()

s = ""
for i in range(len(f1)):
    if f1[i] != f2[i]:
        s += f1[i].decode()

print(s)
```

Flag: `flag{5uch_4_c00l_pr0j3ct!!}`
Karst's password: `y4z00_tr1but4ry`

Corncob's rant photo

Link: http://chal-host.chals.mcpshsf.com/edited_photo.png

We are given a photo of a very messy desk. Using the `strings` Linux command, we were able to find the flag at the end of the file.

Flag: `flag{n0t3p4d++_15_0ur_s4v10r}`

Karst’s file sharing website

Link: http://fileshare-flask.chals.mcpshsf.com

This website requires a username and a password. Using the username found in LSB Frying Pan and password found in Robot images, we are able to log in and access more files.

Further files:
output.wav
login.pcap
vm.tar.gz
edited-photo.png
evidence.zip

output.wav

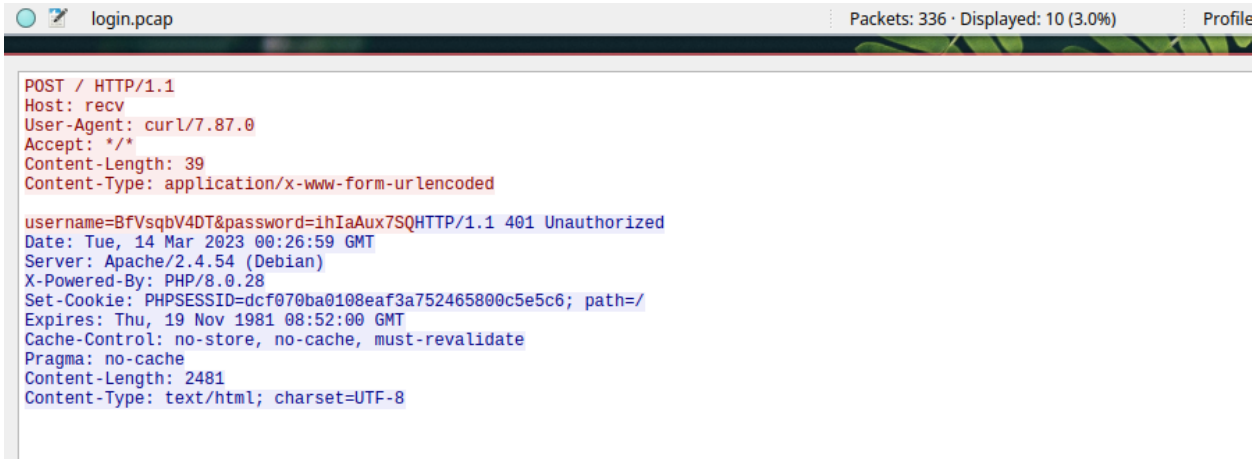
We opened the .wav file in Audacity, and were able to find the flag by choosing to view the audio in the Spectrogram.

Flag: `flag{mus1c_m4j0r5_c4n_h4ck_t00}`

login.pcap

As .pcap files are network captures, we opened the file in WireShark. There were numerous failed login attempts to Tweepy, but we eventually found one that worked.

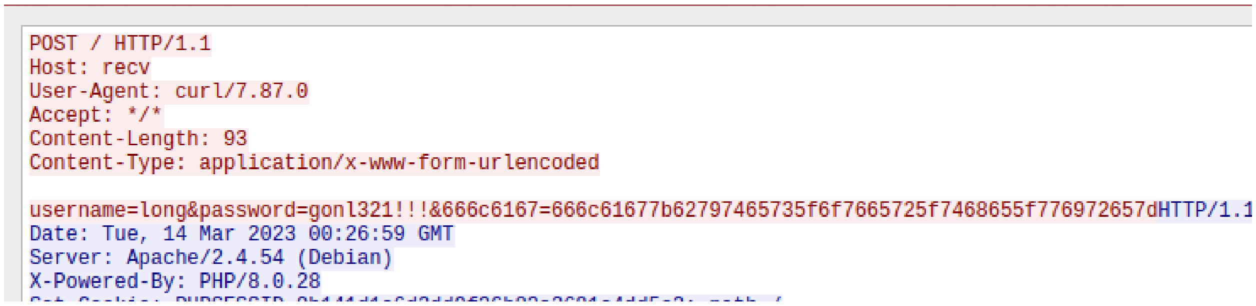
An example of the failed login attempt:



```
POST / HTTP/1.1
Host: recv
User-Agent: curl/7.87.0
Accept: */*
Content-Length: 39
Content-Type: application/x-www-form-urlencoded

username=BfVsqbV4DT&password=ihIaAux7SQHTTP/1.1 401 Unauthorized
Date: Tue, 14 Mar 2023 00:26:59 GMT
Server: Apache/2.4.54 (Debian)
X-Powered-By: PHP/8.0.28
Set-Cookie: PHPSESSID=dcf070ba0108eaf3a752465800c5e5c6; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 2481
Content-Type: text/html; charset=UTF-8
```

The successful login attempt:



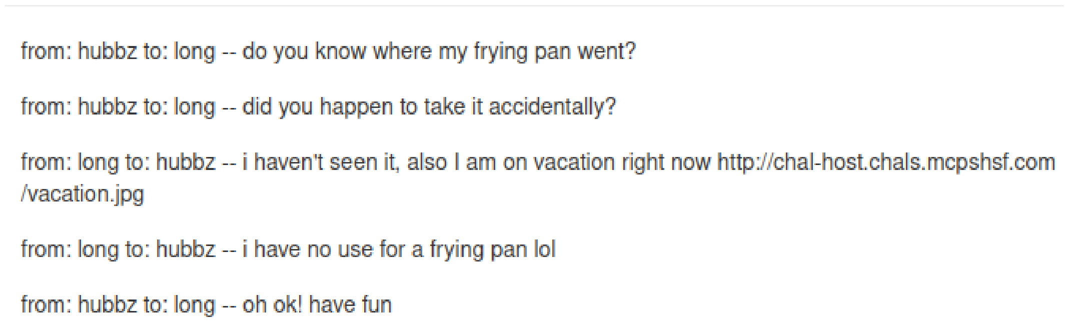
```
POST / HTTP/1.1
Host: recv
User-Agent: curl/7.87.0
Accept: */*
Content-Length: 93
Content-Type: application/x-www-form-urlencoded

username=long&password=gonl321!!!&666c6167=666c61677b62797465735f6f7665725f7468655f776972657dHTTP/1.1
Date: Tue, 14 Mar 2023 00:26:59 GMT
Server: Apache/2.4.54 (Debian)
X-Powered-By: PHP/8.0.28
```

Thus, we recovered Long's username and password as well. At the end of the URL, we also see an extra hex-encoded parameter, which we discovered was the flag.

Flag: `flag{bytes_over_the_wire}`

When we log in as Long and view his DM's, we find one chat with Hubbz:


from: hubbz to: long -- do you know where my frying pan went?

from: hubbz to: long -- did you happen to take it accidentally?

from: long to: hubbz -- i haven't seen it, also I am on vacation right now http://chal-host.chals.mcpshsf.com /vacation.jpg

from: long to: hubbz -- i have no use for a frying pan lol

from: hubbz to: long -- oh ok! have fun

From this screenshot, we recovered http://chal-host.chals.mcpshsf.com/vacation.jpg. However, this yielded no significant results from a steganographic standpoint.

vm.tar.gz

We extracted this file using tar -xvf. In the extracted output, we found a file named git-archive.zip, which contains a .git folder. By using `git log`, we can retrieve all the commits from the git repository. Going through the logs, we find a lot of edits to a file named flag.txt. Since each commit replaces `flag.txt` with an incremental piece of a message, we can use `git checkout $commit_hash` to traverse through all commits and reassemble the flag by hand. Additionally, **we were able to recover Long's confession**.

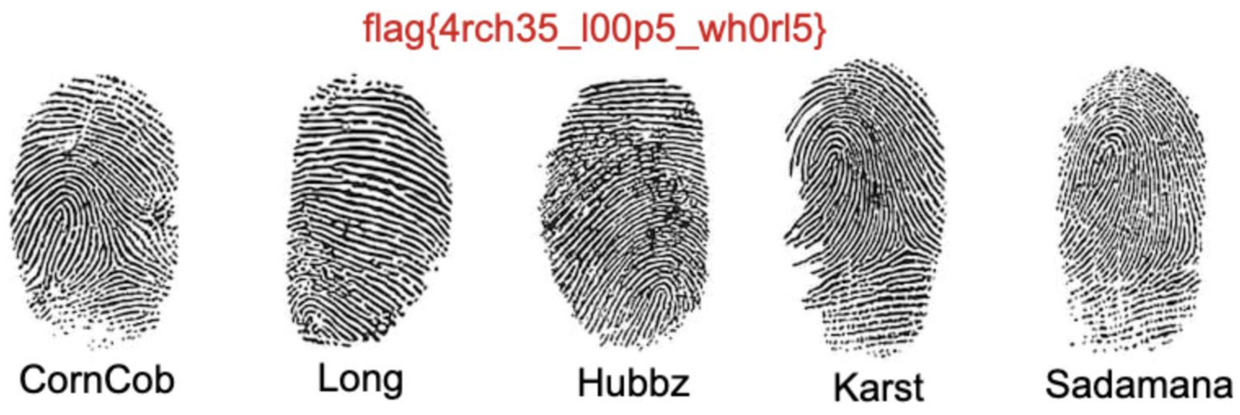
```
In case you didn't know, I did it >:) --Long:flag{g1t_r34ss3mbly}
```

Flag: `flag{g1t_r34ss3mbly}`

edited_photo.png

The first byte of the edited photo was incorrect, and so we fixed it by replacing it with 0x89:
`0x89 0x50 0x4E 0x47 0x0D 0x0A 0x1A 0x0A`
To replace it, we first opened the image in vim, a popular text editor used worldwide by millions of developers (more importantly, used by two out of the three investigators). We then converted vim to the hex editor mode with a combination of other tools (notably `xxd`). From there, we were able to change the first byte, and then save it (using :w). Finally, we quit vim (using :q).

We were then able to open it in an image viewer to see the flag and a set of labeled fingerprints.



Flag: `flag{4rch35_l00p5_wh0rl5}`

evidence.zip

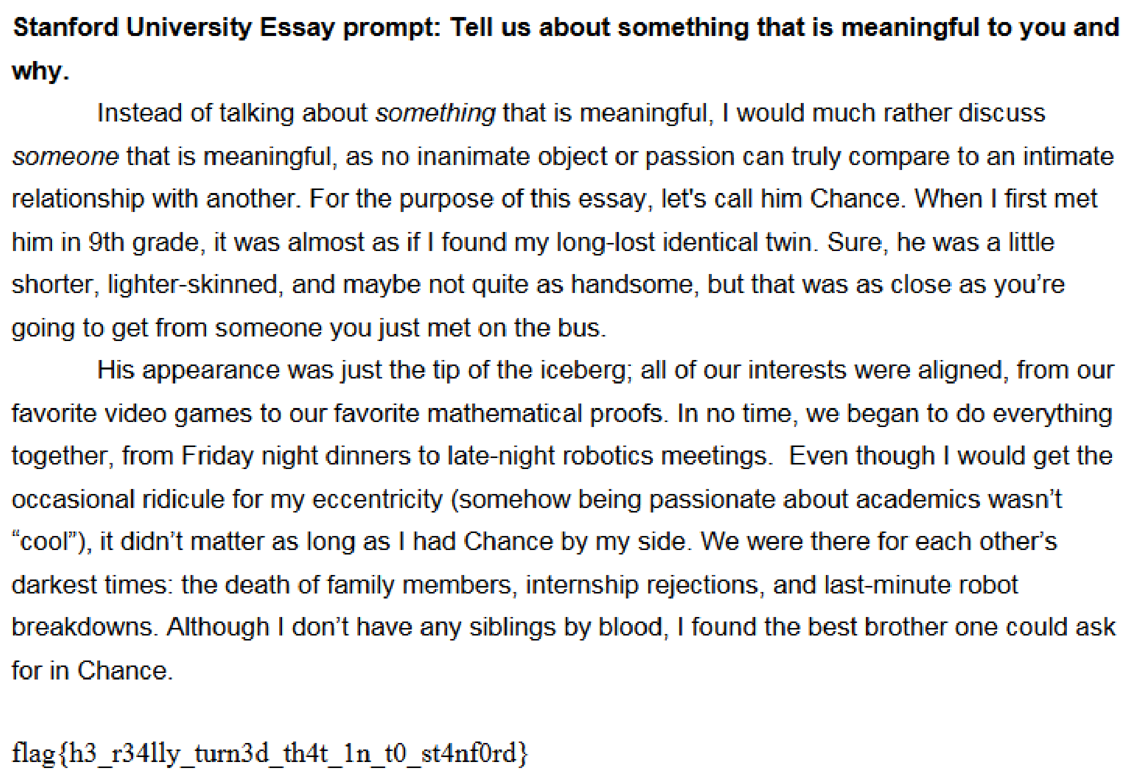
We were given a protected 7-zip file. Using John the Ripper, we brute forced the password to the 7-zip file using the provided wordlist and found that the password was `mischance`.

Inside were 6 files:

evidence.txt

```
Good riddance! Chance deserved to go! Why couldn't he just give me latkes? Who needs to be
so protective of latkes?!?!? If he had just given me latkes in the first time, he would still be
here. But how am I supposed to get my latkes now??? Maybe I'll bug CornCob some more. If
he doesn't agree…
```

ChanceEssay.pdf



Stanford University Essay prompt: Tell us about something that is meaningful to you and why.

Instead of talking about *something* that is meaningful, I would much rather discuss *someone* that is meaningful, as no inanimate object or passion can truly compare to an intimate relationship with another. For the purpose of this essay, let's call him Chance. When I first met him in 9th grade, it was almost as if I found my long-lost identical twin. Sure, he was a little shorter, lighter-skinned, and maybe not quite as handsome, but that was as close as you're going to get from someone you just met on the bus.

His appearance was just the tip of the iceberg; all of our interests were aligned, from our favorite video games to our favorite mathematical proofs. In no time, we began to do everything together, from Friday night dinners to late-night robotics meetings. Even though I would get the occasional ridicule for my eccentricity (somehow being passionate about academics wasn't "cool"), it didn't matter as long as I had Chance by my side. We were there for each other's darkest times: the death of family members, internship rejections, and last-minute robot breakdowns. Although I don't have any siblings by blood, I found the best brother one could ask for in Chance.

flag{h3_r34lly_turn3d_th4t_1n_t0_st4nf0rd}

Flag: `flag{h3_r34lly_turn3d_th4t_1n_t0_st4nf0rd}`

chance_1.png
An image of Chance. No significant results from steganography tools.

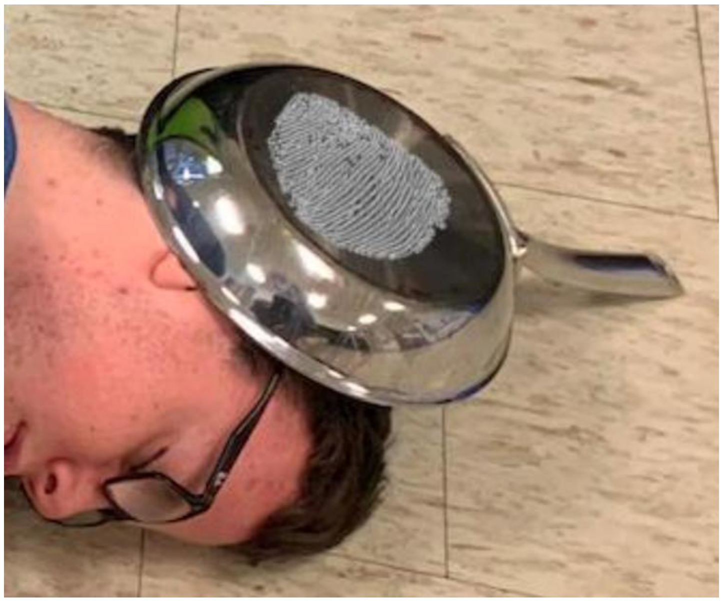
chance_2.png
An image of Chance and two other unidentified individuals holding a box of latkes (likely homemade). No significant results from steganography tools.

chance_3.png
An image of Chance. No significant results from steganography tools.

karst_1.png
An image of Karst. No significant results from steganography tools.

Crime Scene Photo

Link: http://chal-host.chals.mcpshsf.com/chance.jpg



An account called Police posted this photo of the crime scene. The murder weapon was Hubbz' frying pan, as revealed by the conversation in login.pcap. There is a visible fingerprint on the murder weapon, which we identified to match with Long's fingerprint using the labeled fingerprints we found in edited_photo.png. This death corresponds with Long's confession recovered in vm.tar.gz.

Conclusion

We conclude that Long murdered Chance in cold blood in retaliation for his gatekeeping of his homemade latkes, as referred to in Appendix B, item 1. Though Chance's body was discovered by authorities on 3/6/20, we presume the time of death to be on or before 2/26/20 as shown by the timeline and direct message screenshots in Appendix B, item 2. Long stole Hubbz's frying pan (login.pcap) and used it to smack Chance over the head (Crime Scene Photo).

We suspect, though are uncertain of, both Chance and Long's participation in a latke cartel. We would also like to warn CornCob as he may also be in danger.

Appendix A: latke_recipes.docx

Classic Latkes: The Easiest, Simplest Method

MAKES

12 (4-inch) latkes

INGREDIENTS

- 1 1/2 pounds baking potatoes (3 to 4 potatoes)
- 1/2 medium yellow onion, peeled and quartered
- 1 large egg
- 2 tablespoons matzo meal or unseasoned dry breadcrumbs
- 1 teaspoon kosher salt
- 1/8 teaspoon freshly ground black pepper
- 1 cup canola oil or chicken schmaltz, or a combination of both
- Applesauce and sour cream, for serving

EQUIPMENT

- Measuring cups and spoons
- Knife and cutting board
- Food processor with shredding blade
- Cheesecloth or clean, thin kitchen towel

- Wooden spoon
- Mixing bowl
- 10- to 12-inch cast iron skillet
- Fish spatula
- Fork
- Paper towels
- 2 rimmed baking sheets
- Wire cooling rack

INSTRUCTIONS

1. **Heat the oven and fit one baking sheet with paper towels and another with a cooling rack.** Arrange a rack in the middle of the oven and heat to 200°F. Line 1 rimmed baking sheet with a double layer of paper towels. Fit a wire cooling rack into another baking sheet. Set both aside.

2. **Prepare the potatoes.** Scrub the potatoes well, but do not peel. Cut each potato in half crosswise.

3. **Grate potatoes and onion with a food processor.** Grate the potatoes and onion using the shredding disk of a food processor.

4. **Make a cheesecloth tourniquet and squeeze liquid from potato and onion.** Transfer the grated potato and onion onto a large triple layer

of cheesecloth. Gather the corners and tie around the handle of a wooden spoon. Dangle the bundle over a large bowl, then twist and squeeze the potatoes and onion as hard as you can until no more liquid comes out of the potatoes and onion shreds.

5. **Pour off the liquid, but keep the potato starch.** Give the liquid a few minutes to allow the potato starch to settle and then pour off and discard the liquid but leave the potato starch.

6. **Toss the latke ingredients together with your fingers.** Add the potatoes, onion, eggs, matzo meal or breadcrumbs, salt, and pepper to the bowl of starch. Mix with your fingers, making sure that the potato starch breaks up and is evenly distributed with the rest of the ingredients. Set batter aside for 10 minutes.

7. **Heat the oil.** Place the oil or schmaltz (or a combination of the two) in a large skillet so that when melted there is a depth of 1/4 inch (for a 10-inch skillet you'll need 1 cup of melted oil/schmaltz). Heat over medium-high heat until a piece of the latke mixture sizzles immediately.

8. **Form latkes one at a time.** Scoop 1/4 cup of the mixture onto a fish or flat spatula. Flatten with your fingers to a 4-inch patty.

9. **Fry the latkes until golden on both sides.** Slide the latke into the hot oil, using a fork to nudge the latke into the pan. Repeat until the pan is full but the latkes aren't crowded. Cook until deeply golden-brown, 4 to 5 minutes per side, adjusting the heat if necessary.

10. **Drain the latkes.** Transfer the latkes to a paper towel-lined baking sheet to drain for 2 minutes.

11. **Serve with applesauce and sour cream or keep warm in the oven.** Serve immediately with applesauce and sour cream, or transfer the latkes to the wire cooling rack set in the baking sheet and keep warm in the oven for up to 30 minutes while you continue cooking the rest of the latkes.

RECIPE NOTES

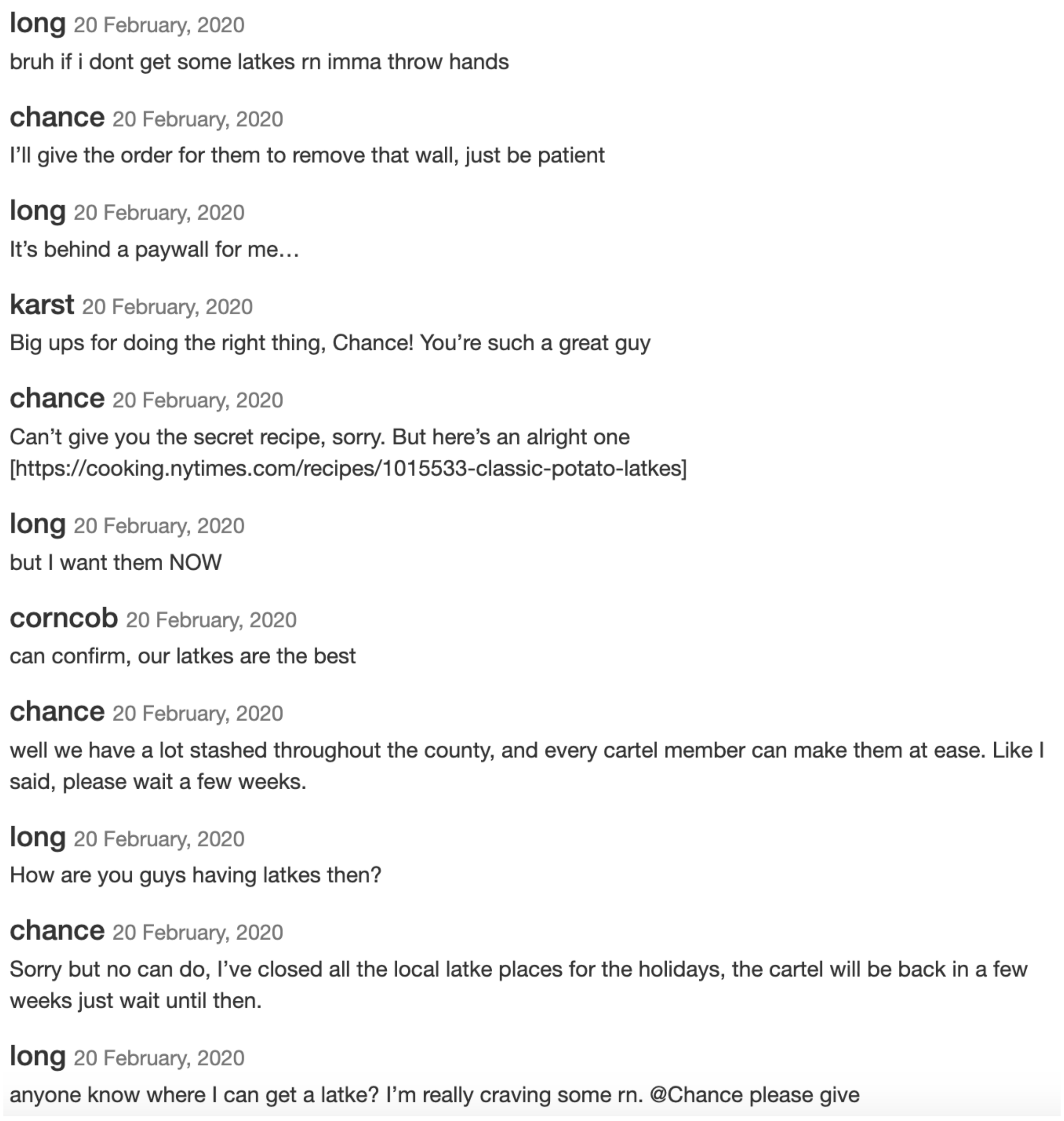
Make ahead: Latkes are best made and served right away. They can be fried and kept warm in a 200°F oven for up to 30 minutes.

Storage: Refrigerate leftovers in an airtight container and recrisp in a 300°F for 5 to 10 minutes. Keep a close eye on the latkes when reheating so they do not burn.

Doubling: The recipe can be doubled, although you will need an extra sheet of cheesecloth to squeeze the extra potato and onion shreds. The oil (and schmaltz, if using) will need to be replaced halfway through frying. Pour the used oil into a heatproof bowl, wipe out the skillet, then heat fresh oil and continue frying.

Appendix B

Item 1: Long threatens violence over not having latkes



long 20 February, 2020
bruh if i dont get some latkes rn imma throw hands

chance 20 February, 2020
I'll give the order for them to remove that wall, just be patient

long 20 February, 2020
It's behind a paywall for me…

karst 20 February, 2020
Big ups for doing the right thing, Chance! You're such a great guy

chance 20 February, 2020
Can't give you the secret recipe, sorry. But here's an alright one
[https://cooking.nytimes.com/recipes/1015533-classic-potato-latkes]

long 20 February, 2020
but I want them NOW

corncob 20 February, 2020
can confirm, our latkes are the best

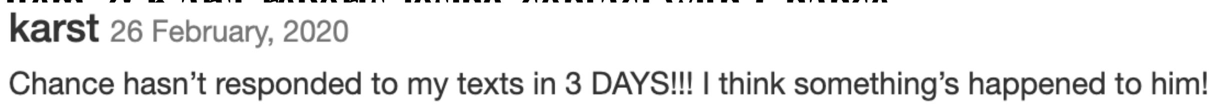
chance 20 February, 2020
well we have a lot stashed throughout the county, and every cartel member can make them at ease. Like I said, please wait a few weeks.

long 20 February, 2020
How are you guys having latkes then?

chance 20 February, 2020
Sorry but no can do, I've closed all the local latke places for the holidays, the cartel will be back in a few weeks just wait until then.

long 20 February, 2020
anyone know where I can get a latke? I'm really craving some rn. @Chance please give

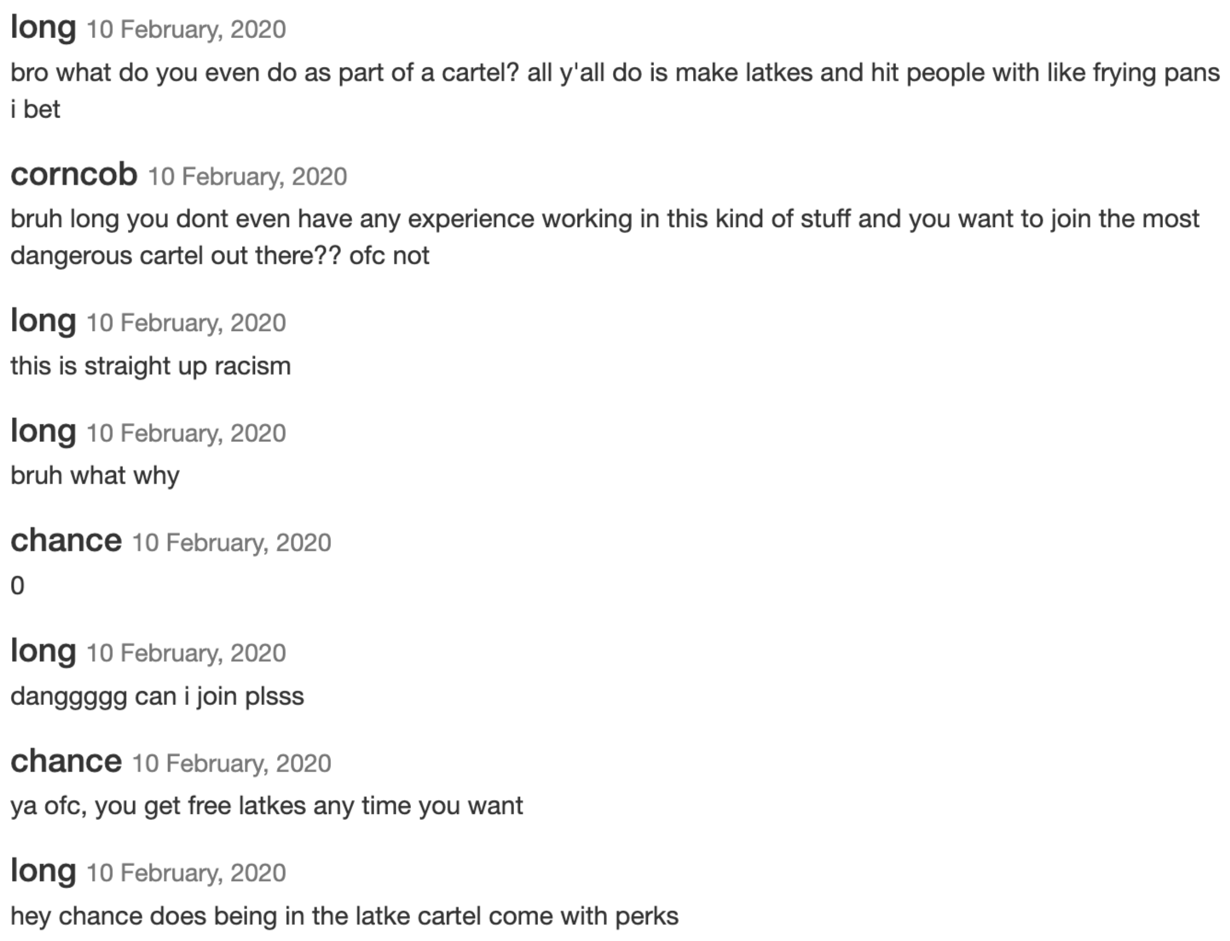
Item 2: Karst reports losing contact with Chance



karst 26 February, 2020

Chance hasn’t responded to my texts in 3 DAYS!!! I think something’s happened to him!

Item 3: Long hints at cartel membership



long 10 February, 2020

bro what do you even do as part of a cartel? all y'all do is make latkes and hit people with like frying pans i bet

corncob 10 February, 2020

bruh long you dont even have any experience working in this kind of stuff and you want to join the most dangerous cartel out there?? ofc not

long 10 February, 2020

this is straight up racism

long 10 February, 2020

bruh what why

chance 10 February, 2020

0

long 10 February, 2020

danggggg can i join plsss

chance 10 February, 2020

ya ofc, you get free latkes any time you want

long 10 February, 2020

hey chance does being in the latke cartel come with perks

Appendix C: Block Ciphers and AES

AES, or the Advanced Encryption Standard, is a block cipher. Block ciphers take a single block of text and encrypt it with a key, usually of the same size, to output an encrypted block. Block ciphers consist of *modes of operation*, which are different methods to encrypt arbitrarily long messages using only block encryption.

The most common mode of operation is known as Electronic Code Book or the ECB mode of operation. In this mode, a message is split up into blocks of equal size, and each block is encrypted separately, as seen in Figure 2.

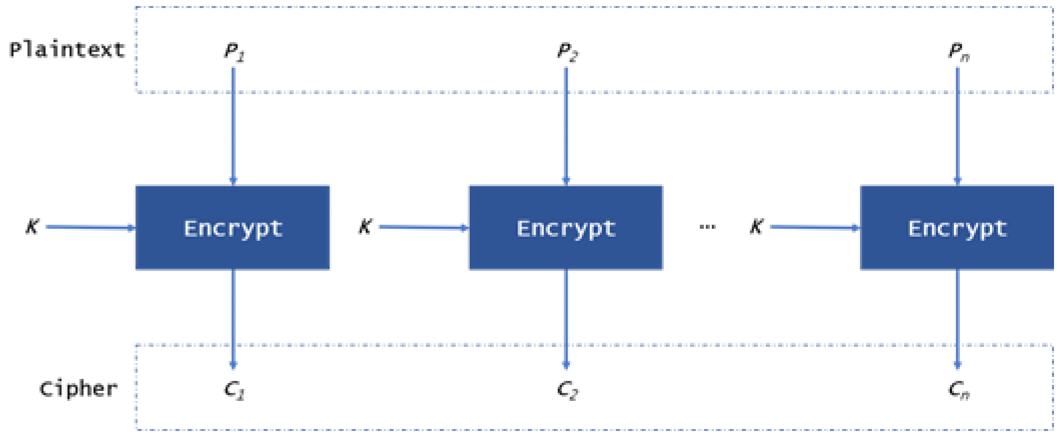


Figure 2: ECB mode of operation

However, the ECB mode is generally considered insecure. The main issue with ECB is a lack of *diffusion*. Though there is diffusion within each block, with large messages, identical blocks will encrypt in the same way. This is seen in the famous penguin image [Figure 3], where the ciphertext still reveals patterns in the plaintext.

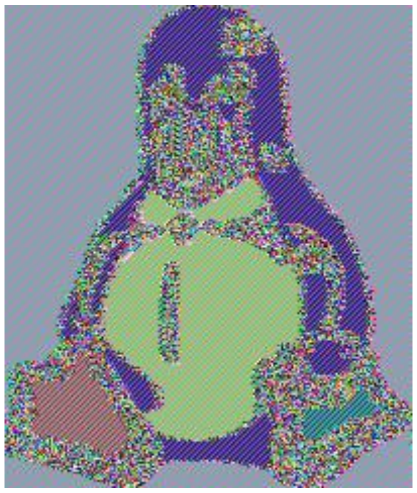
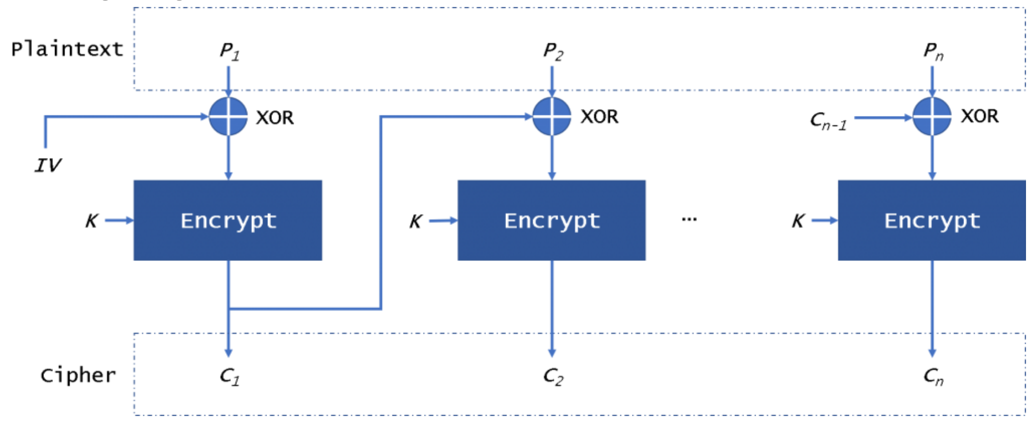


Figure 3: ECB encryption

To mitigate this issue, the Cipher Block Chaining (CBC) was developed in 1981. Instead of encrypting per block, we instead use the previous block to encrypt the next, as seen in the following image.



We now need an IV to start the encryption for the first block. The initialization vector must be public, as it is required for both encryption and decryption. However, it should never be the same as the key, as the key contains private information while the initialization vector must be public. To see the official specification of the block cipher modes of operation, please reference FIPS81, the original paper published for DES.

Footnotes

1.