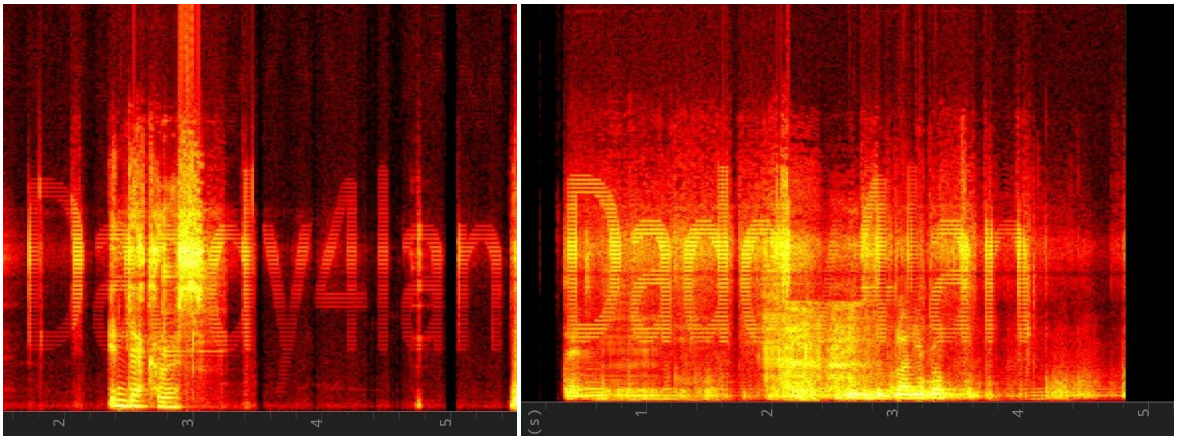
50.042 CTF Challenge 2023 (Solution)
Note: Do not distribute

Team RaisinGang

Jiang Hongbei	(1005870)
Joel Sng Kiat Loong	(1005968)
John-David Tan Ming Sheng	(1005971)
Kwok Keith	(1006344)
Jon Koo Jia Jun	(1006388)

Stage 1: Steganography

Upon loading the video from the URL provided, challengers may hear buzzing noises around every 5 minutes indicating that there are hidden messages in the audio. By using a spectrogram tool which can be found online, they can view the spectrograph of the audio at that time.

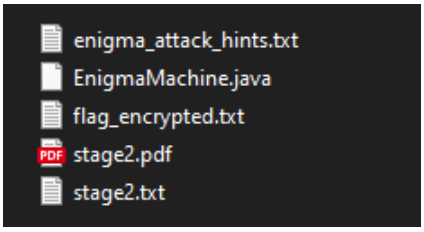


From these two samples, it is obvious that the hidden message is "Daddy4lan".

The challenger can now decrypt the stage2 file using AES-CBC mode with the given IV. A sample python code is given here for the decryption.

```
1 from Crypto.Cipher import AES
2 from Crypto.Util.Padding import pad, unpad
3
4 key = b"Daddy4lan\0\0\0\0\0\0\0"
5 cipher = AES.new(key, AES.MODE_CBC, iv=b"\x12\xad\xf8\x8f\xb3\x10\x02\x92.\x98\t@\x02&\xd2T")
6
7 with open("stage2","rb") as fin:
8     with open("stage2_decrypted.zip","wb") as fout:
9         data = fin.read();
10        out = unpad(cipher.decrypt(data), AES.block_size)
11        fout.write(out);
```

The zip file can now be opened and its contents are as follows:



Stage 2: The Enigma Machine

The method for attacking is a known plaintext attack. First the challenger implements a custom Enigma machine with a variable size charset up to size 255 (each character is a byte). A sample implementation in Rust is shown below. To compile an optimized binary, use this command: `rustc -C opt-level=3 main.rs`. To run it, simply double-click on the main.exe file. (we have also provided this binary if you want a precompiled version)

file: enigma/mod.rs

```
  1 pub mod rotor;
  2 use enigma::rotor::Rotor;
  3 pub mod plugboard;
  4 use enigma::plugboard::Plugboard;
  5
  6 use std::collections::HashMap;
  7
  8 // Actual Machine
  9 pub struct Enigma {
 10     pub slow_rotor: Rotor,
 11     pub mid_rotor: Rotor,
 12     pub fast_rotor: Rotor,
 13     pub plugboard: Plugboard,
 14     pub reflector: HashMap<u8,u8>,
 15 }
 16
 17 impl Enigma {
 18     pub fn new(reflector_rotor: Rotor, slow_rotor: Rotor, mid_rotor: Rotor,
 19                fast_rotor: Rotor, plugboard: Plugboard) -> Enigma {
 20 +--  6 lines hidden: Enigma {---
 26         }
 27     }
 28
 29
 30     pub fn step(&mut self) {
 31 +-- 17 lines hidden: Double Stepping Mid rotor (ANOMALY FOUND IN THE ORIGINAL ENIGMA)---
 48     }
 49
 50
 51     // Encryption method
 52     pub fn encrypt(&mut self, mut input: u8) -> char {
 53 +-- 18 lines hidden: Step the rotor/wheel---
 71     }
 72
 73
 74     // Starting position of the rotor (Part of Key)
 75     pub fn rotor_settings(&mut self, pos1: u8, pos2: u8, pos3: u8) {
 76         self.slow_rotor.pos = pos1;
 77         self.mid_rotor.pos = pos2;
 78         self.fast_rotor.pos = pos3;
 79     }
 80
 81     // Change the settings of the plugboard
 82     pub fn set_plugboard(&mut self, new_plug_board: Plugboard) {
 83         self.plugboard = new_plug_board;
 84     }
 85 }
```

The challenger is also required to implement the rotor and plugboard structs (Java skeleton code is given). They may feed the constructors a 26 character array to simulate the actual Enigma. There are many online simulators they can cross check the functionality against. Otherwise, they may also contact us for hints and test cases.

After implementing the Enigma machine, they should observe that the cribs (plaintext-ciphertext pair) are not the same length. Since Enigma cannot encrypt a character to itself (due to the prescence of the reflector), it is trivial to find which portion of the ciphertext corresponds to the plaintext. This python script demonstrates that:

file: find_ciphertext.py

```
  1 +---  2 lines hidden ---
  3 def possible(a,b):
  4     for i in range(min(len(a),len(b))):
  5         if a[i] == b[i]:
  6             return False
  7     return True
  8
  9 for i in range(len(cipher)):
 10     cipher2 = cipher[i:]
 11     if possible(plain,cipher2):
 12         print(cipher2[:len(plain)])
 13         break;
 14
```

The script is very simple. It simply checks for matches between the plaintext and ciphertext by shifting the plaintext character by character (since the plaintext is shorter) until there are no matches, then that could be the ciphertext substring that corresponds to the plaintext.

Now, they have to attack using the known plaintext-ciphertext pair provided in order to find the rotor settings.

file: main.rs

```
 1 mod enigma;
 2 use enigma::Enigma;
 3 use enigma::rotor::Rotor;
 4 use enigma::plugboard::Plugboard;
 5 use std::{io, thread};
 6 use std::sync::{Arc, Mutex};
 7
 8 +-- 16 lines hidden ---
24 fn main() {
25 +-- 17 lines hidden ---
42     // ***************************
43     // ***** ATTACK
44     // ***************************
45     // Make 8 threads AND ATTACK
46
47
48     let maximum_fitness = Arc::new(Mutex::new(0));
49     let chosen_rotor_config = Arc::new(Mutex::new((0 as u8,0 as u8,0 as u8,0 as u64)));
50     let mut handles: Vec<thread::JoinHandle<()>> = Vec::new();
51
52     let num_threads = 8u8;
53 +--  5 lines hidden ---
59     // Create threads
60     for idx in 0..num_threads {
61         let maximum_fitness = Arc::clone(&maximum_fitness);
62         let chosen_rotor_config = Arc::clone(&chosen_rotor_config);
63
64         let handle = thread::spawn(move || {
65             let plugboard = Plugboard::new(&[]);
66             let ufw_b = Rotor::new(ORIG, UFW_B_MAP, 0);
67             let r1 = Rotor::new(ORIG, R1_MAP, 12);
68             let r2 = Rotor::new(ORIG, R2_MAP, 14);
69             let r3 = Rotor::new(ORIG, R3_MAP, 47);
70             let mut enigma = Enigma::new(ufw_b,r2,r1,r3, plugboard);
71             let partition_size: u8 = 94/num_threads + 1;
72             let start: u8 = idx * partition_size;
73             let end: u8 = std::cmp::min(start + partition_size - 1, 93);
74             println!("Thread {idx} attacking: {start} 0 0 to {end} 93 93");
75             for i in start..=end {
76                 for j in 0..94 {
77                     for k in 0..94 {
78                         enigma.rotor_settings(i,j,k);
79                         let mut decrypted = String::new();
80                         for c in KNOWN_CIPHERTEXT.chars() {
81                             decrypted.push(enigma.encrypt(c as u8));
82                         }
83                         let f = fitness(&decrypted, &KNOWN_PLAINTEXT.to_string());
84                         let mut max = maximum_fitness.lock().unwrap();
85                             if f > *max {
86                                 println!("Rotors: {i} {j} {k}     \t Fitness: {f}");
87                                 *max = f;
88                                 let mut chosen_rotor = chosen_rotor_config.lock().unwrap();
89                                 *chosen_rotor = (i, j, k, f);
90                             }
91                         }
92                     }
93                 }
94                 //println!("Thread {idx} completed");
95             });
96             handles.push(handle);
97     }
```

This attack takes about 40s using 8 threads (compile optimized) produces the correct rotor settings 51, 76, 8.

Next, the possible plugbaord cominations.... This is done by trying all mappings possible by using the partly decrypted ciphertext with the known plaintext and doing a fitness test for each character mapping.

Lastly, a brute force attempt on the remaining plugboard wires are done. This takes a negligible amount of time to run on a single thread.

Finally, the plugboard settings are found: `(D, +) (9, A) (f, ") (4, ]) (N, ‘) (B, U) (I, %) (3, 8) (Z, }) (2, *)`

Now, all the challenger has to do is decrypt the flag using these rotor and plugboard settings to obtain:

`file: flag_plaintext.txt`

fcs23{''Betreff: Streng geheim - Operation Goldener Horizont // // An: Agent Sturmvogel // Von: General Klaus Adler // // Datum: 7. August 1941 // // Sehr geehrter Agent Sturmvogel, // // Wir prasentieren Ihnen den detaillierten Plan fur unsere streng geheime strategische Operation mit dem Codenamen "Operation Goldener Horizont", die sich auf die Invasion des kleinen, unabhangigen Inselstaats Singapur konzentriert. Obwohl Singapur uber keine bedeutenden naturlichen Ressourcen verfugt, verfugt es uber fortschrittliche Technologie, ein Talentpool und erhebliche auslandische Investitionen. // // Geografische Koordinaten: // Singapur ist eine Inselnation genau bei 1.30679 N Breitengrad und 103.84309 E Langengrad gelegen. Die Geografie der Insel umfasst atemberaubende Kustenebenen, bluhende stadtische Zentren und moderne Industriegebiete. // // Ziel: // Die Operation Goldener Horizont zielt darauf ab, die technologischen Fortschritte Singapur zu sichern und von der talentierten Arbeitskraft zu profitieren, um letztendlich eine strategische Prasenz in der Region zu etablieren. // // Wesentliche Uberlegungen: // // Nachrichtendienstliche Erkenntnisse: // Datum: Beginn am 15. August 1941. // Einzelheiten: Setzen Sie unsere Elite-Aufklarungseinheiten unter der Leitung von Major Heinrich Schmidt ein, um Nachrichtendienstliche Informationen aus der Luft und vor Ort zu sammeln. Erhalten Sie prazise Informationen uber Singapurs technologische Fahigkeiten, kritische Infrastrukturen und auslandische Partnerschaften. // // Psychologische Kriegsfuhrung: // Datum: Parallel zu Phase 1. // Einzelheiten: Starten Sie unter der Leitung von Oberst Helga Bauer psychologische Operationen, um Zweifel bei der Singapurianisch Fuhrung zu saen, was ihre Fahigkeit angeht, unserem Vormarsch standzuhalten. Verbreiten Sie gezielte Botschaften uber verdeckte Kanale und nutzen Sie bestehende Spaltungen. // // Begrenzte Gefechte: // Datum: Beginn am 25. August 1941. // Einzelheiten: Betonen Sie begrenzte Gefechte mit den Singapurianischen Streitkraften, mit Leutnant Otto Muller, um unsere Ziele zu erreichen, ohne den technologischen Besitz oder die Zivilbevolkerung zu stark zu schadigen. // // Diplomatische Annaherung: // Datum: Beginn am 30. August 1941. // Einzelheiten: Beginnen Sie diplomatische Verhandlungen mit Singapurische Fuhrern, unter der Aufsicht von Hauptmann Ingrid Fischer. Stellen Sie einen uberzeugenden Fall fur gegenseitige Zusammenarbeit und die Vorteile einer Mitgliedschaft in unserer grossen Allianz dar. Bieten Sie an, auslandische Investitionen zu schutzen und den technologischen Fortschritt unter unserem Schutz fortzusetzen. // // Operativer Plan: // Phase 1: Infiltration und Aufklarung // Datum: 15. August bis 24. August 1941. // Einzelheiten: Setzen Sie Aufklarungseinheiten ein, um Informationen uber Singapurs Technologiezentren, Forschungseinrichtungen und auslandische Partnerschaften zu sammeln. Identifizieren Sie Schlusselziele fur gezielte Schlage. // // Phase 2: Verdeckte Mobilisierung // Datum: 25. August bis 29. August 1941. // Einzelheiten: Stellen Sie Truppen in verborgenen Lagern bei 1.34077 N Breitengrad und 103.96334 E Langengrad zusammen. Auf dem Gelande der Ingenieuruniversitat SUTD gelegen, mussen wir unter der Leitung von Major Friedrich Schafer hochste Vorsicht walten lassen und Tarn- und Ablenkungstaktiken einsetzen, um Truppenbewegungen zu verschleiern. // // Die Universitat beherbergt viele talentierte junge Studenten in den Studiengangen Ingenieurwesen und Cybersicherheit, die bereits militarisch ausgebildet sind. Nutzen Sie die Anwesenheit dieser vielversprechenden Nachwuchskrafte, um sie diskret in die geplanten strategischen Operationen einzubinden. // // Die jungen Studenten sollen mit grosster Sensibilitat und Sorgfalt in die Truppen einbezogen werden, da ihr Fachwissen und ihre Ausbildung von unschatzbarem Wert fur den Erfolg der Operation sind. Major Friedrich Schafer wird personlich die Tarnung und Ablenkungstaktiken uberwachen, um die unauffallige Verlegung der Truppen auf dem Campus zu gewahrleisten. // // Wahrend der Phase 2 sind die Studenten verpflichtet, ihre ublichen akademischen Aktivitaten aufrechtzuerhalten, um nicht den Verdacht der feindlichen Beobachter zu erregen. Die verdeckte Mobilisierung wird streng geheim gehalten, und die jungen Studenten sind darauf geschult, absolute Verschwiegenheit zu wahren. // // Die besondere Aufmerksamkeit, die den talentierten jungen Studenten geschenkt wird, ist von grosster Bedeutung, da ihre Integration in die Operation Goldener Horizont einen wertvollen Vorteil in Bezug auf Technologie und Cyberstrategien darstellt. Ihre Fahigkeiten werden uns helfen, die geplanten Ziele mit Prazision und Effizienz zu erreichen. // // Phase 3: Gezielte Schlage // Datum: 30. August bis 8. September 1941. // Einzelheiten: Fuhren Sie gezielte Schlage gegen Singapurische Technologiezentren durch, um die kritische Infrastruktur minimal zu storen. Konzentrieren Sie sich darauf, ihre Befehls- und Kontrollfahigkeiten zu beeintrachtigen, unter der Aufsicht von Hauptmann Karl Weber, der die taktischen Operationen leitet. // // Phase 4: Diplomatische Verhandlungen // Datum: 10. September bis 20. September 1941. // Einzelheiten: Engagieren Sie die Singapurische Fuhrung in diplomatischen Diskussionen und betonen Sie die Vorteile einer Zusammenarbeit innerhalb unserer grossen Allianz. Unter der Leitung von Oberstleutnant Greta Vogel fuhren Sie diplomatische Bemuhungen durch. // // Phase 5: Wiederaufbau und Stabilisierung // Datum: 25. September 1941 und danach. // Einzelheiten: Nach erfolgreicher Zusammenarbeit oder Kapitulation, richten Sie vorlaufige Verwaltungen ein, um die Stabilitat in Singapur zu gewahrleisten. Unter der Anleitung von General Hermann Schneider fordern Sie das weiterhin schnelle Wachstum des Technologie-Sektors und ziehen auslandische Investitionen an. // // Wir mussen die hochste Vertraulichkeit wahrend der gesamten Operation Goldener Horizont betonen. Unbefugte Lecks oder Enthullungen konnten den Erfolg unserer Mission und unsere Position auf der globalen Buhne schwerwiegend gefahrden. // // Fahren Sie mit unerschutterlicher Prazision und Vorsicht fort und halten Sie jederzeit unsere strategischen Ziele im Auge. Priorisieren Sie immer die Sicherheit und das Wohlergehen unserer Soldaten und der Bevolkerung Singapur. // // Heil Hitler, // // General Klaus Adler''}