The Enigma Machine & Cryptanalysis

When math, machines, and mistakes changed the course of war

CЯ4CK1NG-THΞ-C0D3

Today’s Roadmap

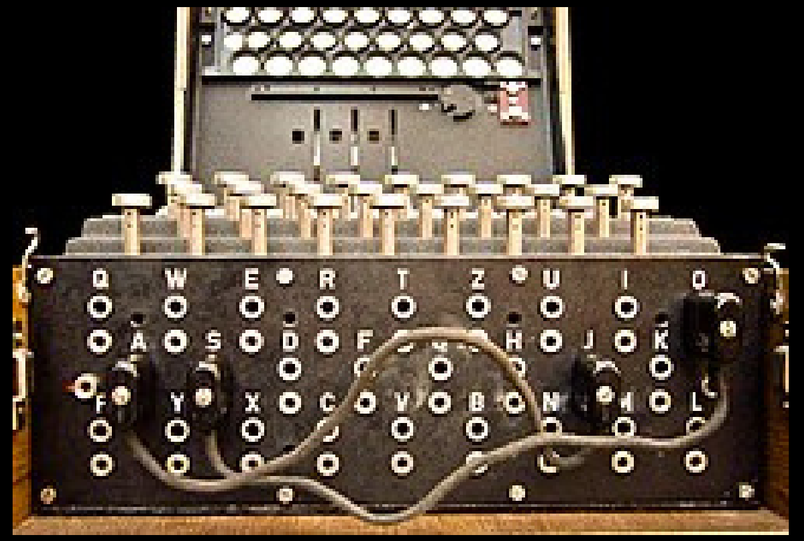
- The story behind Enigma
- Cryptanalysis of Enigma
- Manual vs Automated Cryptanalysis
- Hands-on Cryptanalysis using CrypTool


Video unavailable
Watch on YouTube

The Enigma

“I like solving problems, Commander. And Enigma is the most difficult problem in the world.” - Alan Turing

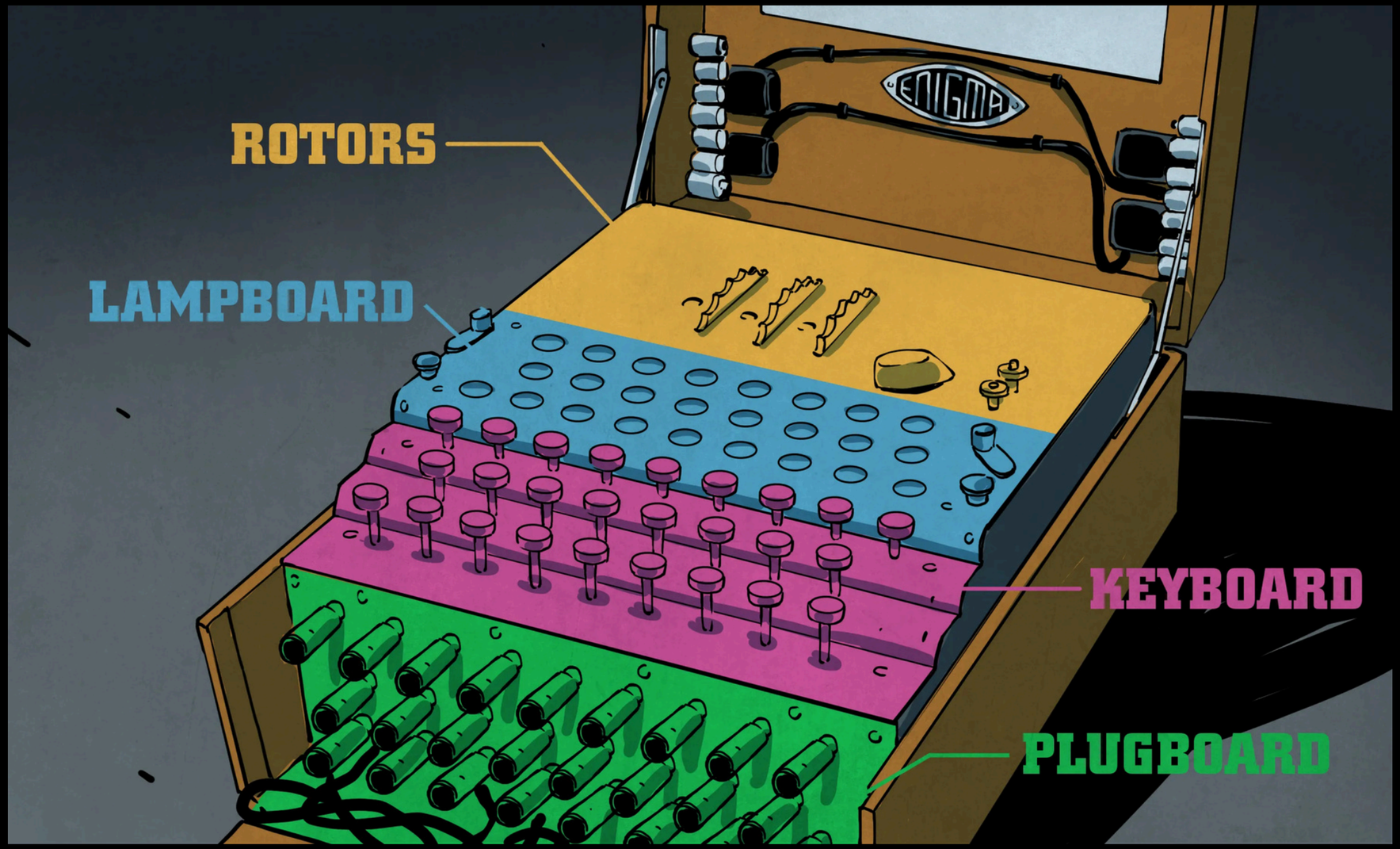
- Used by the Germans in WWII to encrypt military communication.
- Replaced simple substitution ciphers with rotating electrical rotors.
- Each key press changed the substitution pattern → no letter ever encrypted to itself!
- Believed to be unbreakable due to billions of combinations.



How did it work?

- 3 to 5 rotors, each rotating after every keystroke.
- A plugboard (Steckerbrett) swapped letters for added complexity.
- The machine was mechanical + electrical — both analog and digital thinking in one.

The machine



Lets look at its input and output

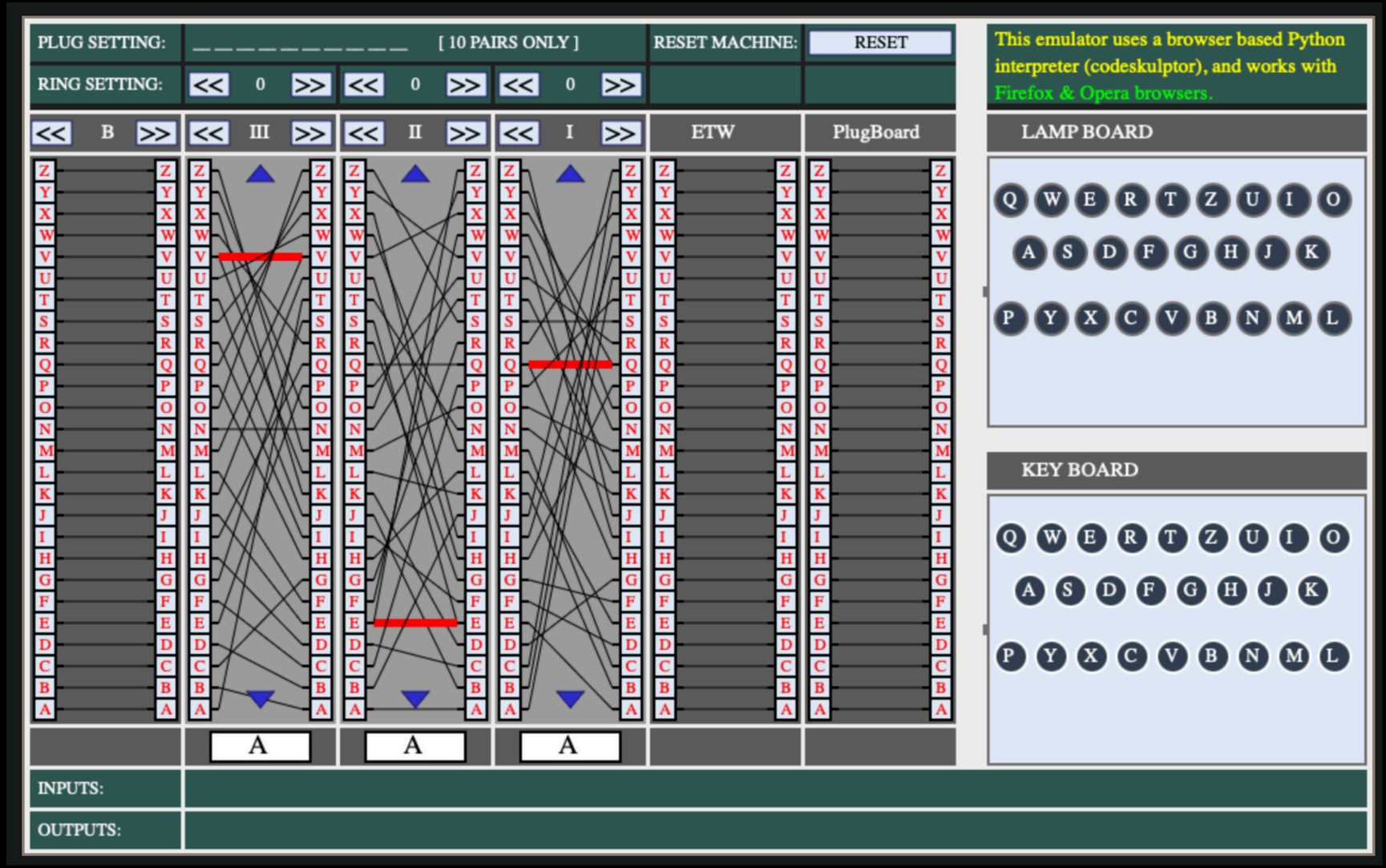
https://matheusportela.com/enigma/

Why did it seem unbreakable?

- Over 150 quintillion possible key combinations (10^{23}).
- Daily key settings changed rotor order, ring settings, and plugboard pairs.
- Messages looked completely random — no visible patterns.
- Operators were trained to believe it was mathematically perfect.

Lets try our hands on it

https://piotte13.github.io/enigma-cipher/



Manual Cryptanalysis

“Brains Before Machines”

- Relied on human logic, patterns, and language frequency.
- Cryptanalysts exploited operator habits and repeated phrases.
- “Heil Hitler” and weather reports gave starting points.
- Used cribs — guessed plaintext fragments to deduce wiring.

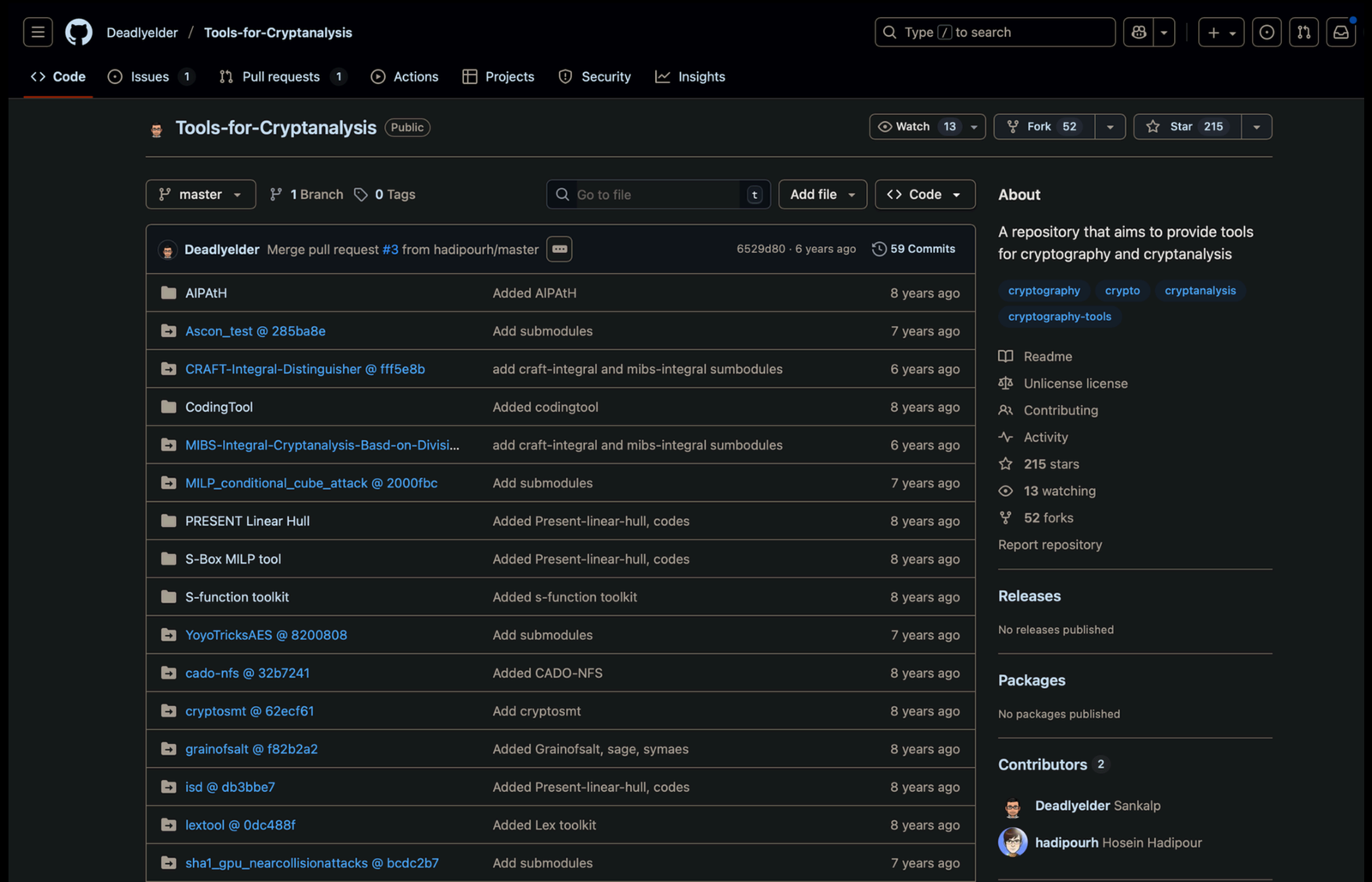
Automated Cryptanalysis

“When Logic Meets Machines”

- Alan Turing and his team at Bletchley Park built the Bombe.
- It systematically tested rotor positions using parallel circuits.
- Reduced days of manual work to hours.
- Introduced mechanical brute force + logic pruning — the basis for today’s attacks.

Automated Cryptanalysis

Some extensive modern software tools:



Layers of Security

- Many believe stacking ciphers = stronger security (e.g., Caesar + Vigenère + RSA).
- But combining without understanding dependencies can create new vulnerabilities.
- Enigma layered rotors, plugboard, and reflector — yet this layering created predictable symmetry.
- Example flaw: the reflector ensured no letter ever encrypted to itself — cryptanalysts used this clue.
- Modern parallel: “Encryption inside encryption” (e.g., HTTPS + VPN) can still fail if both rely on weak keys.

So lets look at the largest factor of error - Humans

Case Study – Human Error

- Cryptographers at Bletchley Park didn’t defeat Enigma through math — they exploited human shortcuts.
- Common mistakes made by German operators:
- Reusing keys and messages (e.g., “AAA,” “HHH,” or date-based keys).
- Predictable openings like “WETTERBERICHT” (“Weather Report”).
- Sending test messages repeatedly.
- These patterns helped Alan Turing’s team align cribs with ciphertext segments.
- Moral: The system was secure; the humans weren’t.

Lets try the crib the team at Bletchley park used

Example of crib analysis

Imagine the Allied forces intercept a German Luftwaffe message sent from a known airbase at 6:00 a.m.. They assume the daily weather report is in the message and will likely contain the German phrase `WETTERBERICHT`.

The intercepted ciphertext:

`PJZ CQP EGLV WXG YJFG`

The guessed plaintext (crib):

`WETTERBERICHT`

The codebreakers would test the crib against the ciphertext, shifting its position one character at a time. In most positions, there would be a "clash"—a letter in the crib would align with the same letter in the ciphertext. Because of Enigma's flaw, this combination of rotor and plugboard settings is impossible for that position.

Crib Position	**P**	**J**	**Z**	**C**	**Q**	**P**	**E**
Possible Crib	W	E	T	T	E	R	B
Clash (T=T)				**X**			
Possible Crib		W	E	T	T	E	R
Clash (E=E)							
Possible Crib			W	E	T	T	E
No Clash							

In this simplified example, by checking against the "no-letter-encrypts-to-itself" rule, the codebreakers quickly eliminate almost all possible locations for the crib, leaving just a few possibilities for the machine's settings.

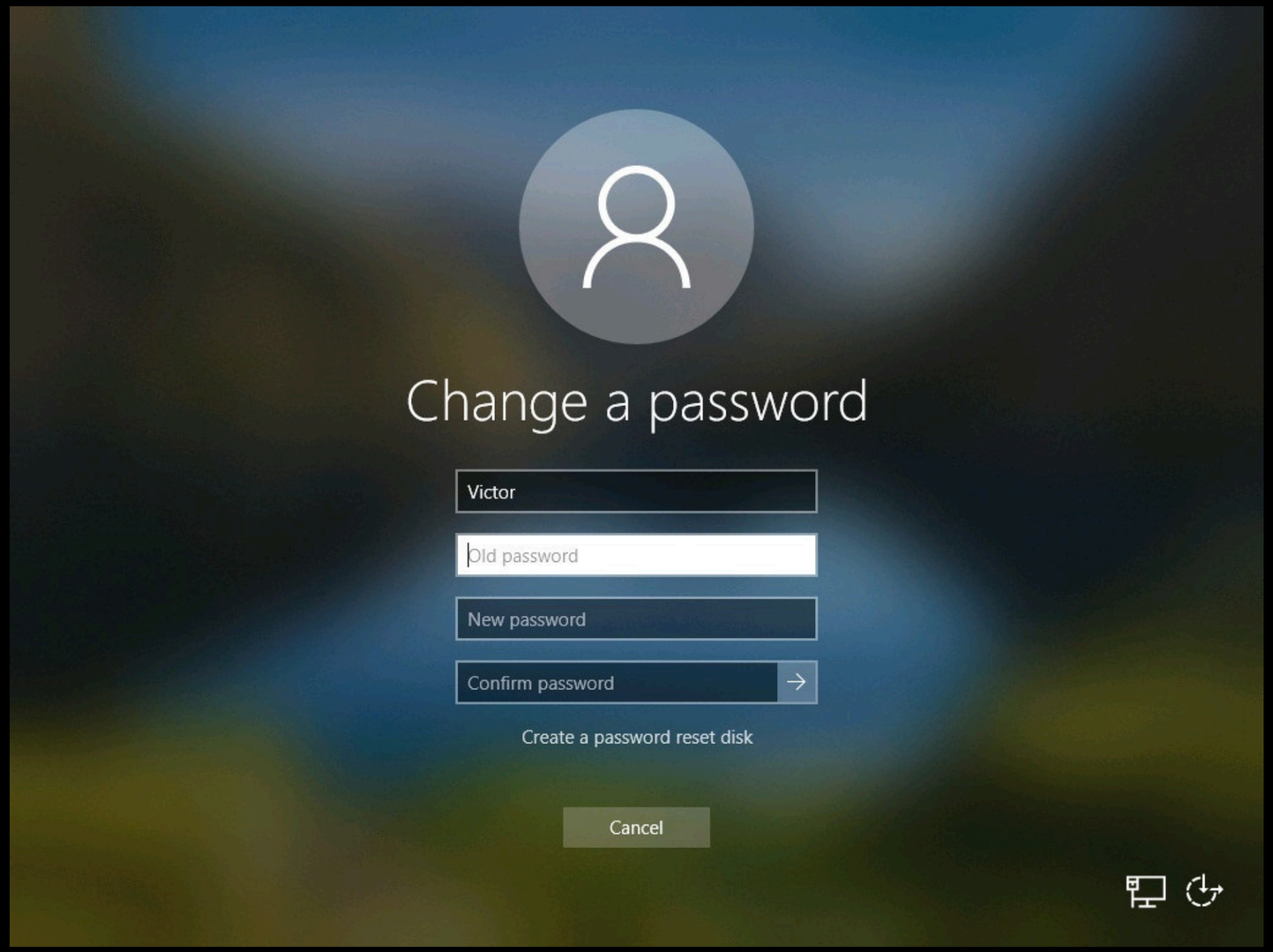
What lessons do we take back from this?

Operational Security (OpSec)

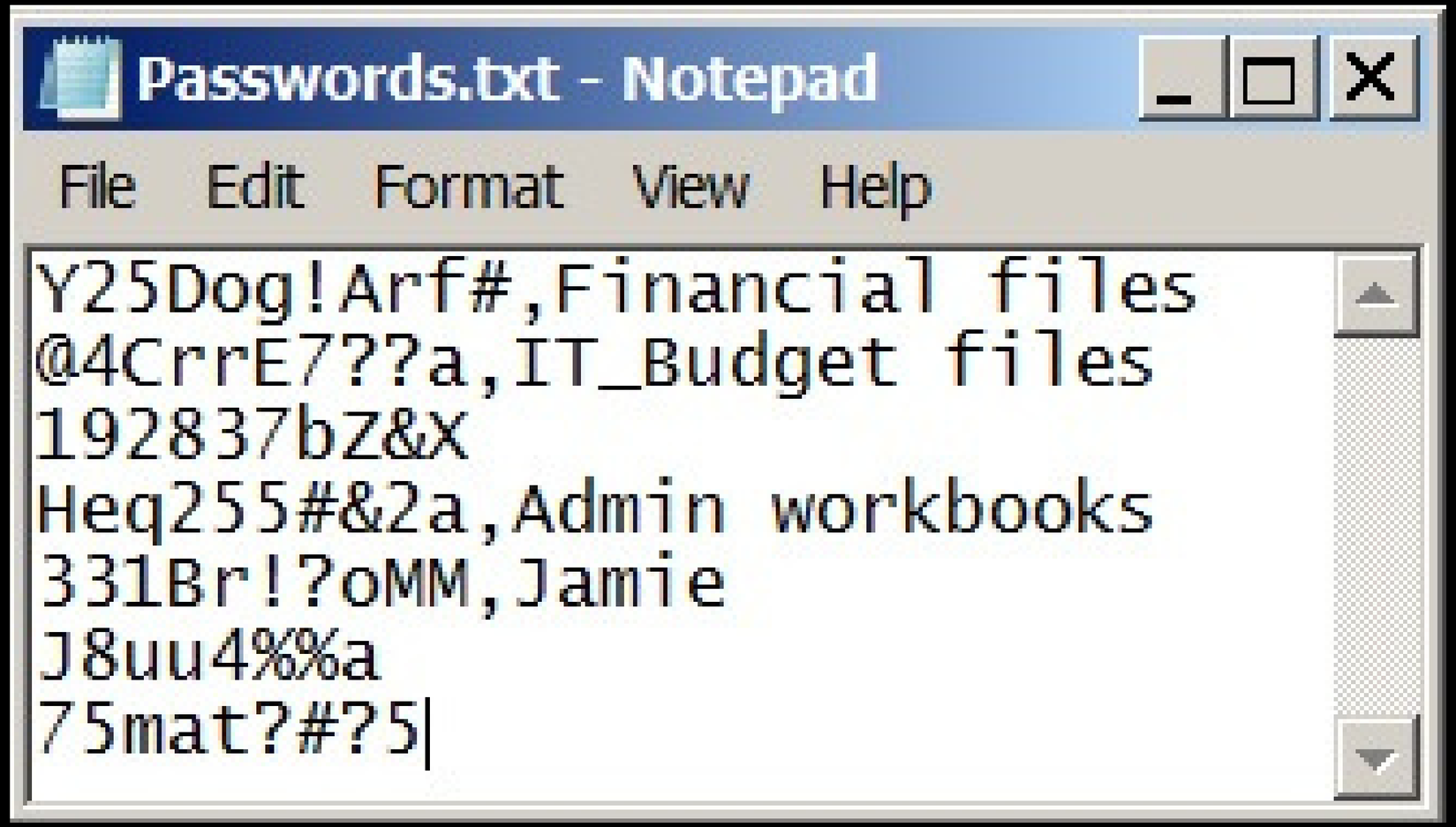
- OpSec = “Operational Security” = ensuring humans don’t compromise mathematical security.
- Key practices:
  - Rotation: Change keys often, use unique keys per session.
  - Separation: Keep encryption keys away from encrypted data.
  - Training: Teach users why “It’s just one password reuse” is dangerous.
- Poor OpSec turned Enigma’s strength into a weakness.
- Today’s crypto can fail even with AES-256 — if private keys are stored in plaintext.

Some good digital habits

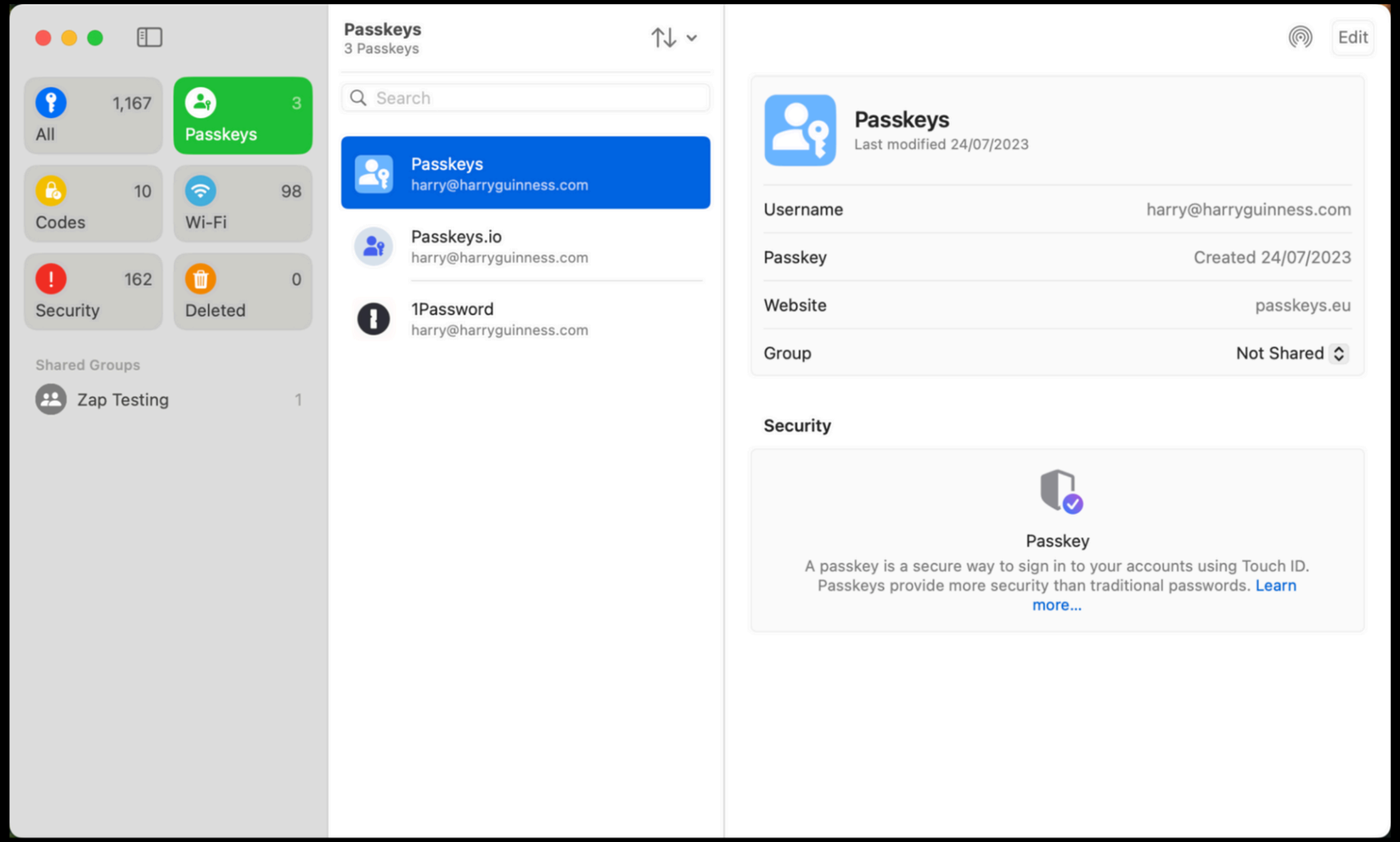
Change your passwords often!!!



do not write your passwords in text files!



Use good password managers



Real-World Parallels

- 2019 – Facebook: Stored hundreds of millions of user passwords in plaintext logs.
- 2021 – AWS S3 Bucket Leaks: Sensitive databases left publicly accessible.
- 2023 – IoT Devices: Hardcoded encryption keys discovered in firmware.
- All are modern echoes of Enigma: strong math, weak usage.
- Attackers don’t need to crack AES if they can steal credentials or exploit human oversight.

Key Takeaways

- Even the most secure algorithm fails when humans are predictable.
- Both manual and automated cryptanalysis rely on pattern recognition.
- The future of security lies in minimizing human exposure to secret material.

Golden rule:

- *“Don’t trust users to be random. Design systems that don’t depend on them being perfect.”*

Wrap-Up Challenge – “Outthink the Machine”

- Imagine designing a cipher that anticipates human error.
- How will it handle reused keys?
- How will it resist predictable inputs?
- Create a mini concept (even pseudocode) for a “human-proof cipher.”
- Bonus points for creativity — “The Enigma 2.0” but smarter.
- *“A chain is only as strong as its weakest human.”*

Concept

- A “human-proof” cipher should not trust the user to:
  - remember or reuse secure keys,
  - avoid using the same password twice,
  - or keep track of nonces properly.
- So this cipher will generate all that automatically, and refuse to encrypt twice with the same key.

Hands on for implementing the same

Where can you try more of this out?

CrypTool-Online
Cryptography for everybody

DE

Portal

CTO Applications | Knowledge | Tools | Source Code | Links

CrypTool-Online

Apps to explore, play around with, and learn about cryptology. For students, teachers, and anyone interested.

Search applications

Historical encryption 11 apps

Atbash
Simple monoalphabetic substitution cipher originally used on the Hebrew alphabet

Caesar / ROT13
Famous shifting cipher used by Julius Caesar With Python code

Monoalphabetic Substitution
Cipher that replaces letters with letters/characters With Python code

Railfence / Redefence
Transposition cipher that uses a rail fence pattern

Show 7 more

Modern encryption 4 apps

RSA (explained step by step)

OpenSSL

Display a menu