



MAY 11-12

---

BRIEFINGS

# **E-Meet (or Emit?) My Keystrokes**

## **How Benign Screen-sharing Meetings Could Leak Typing Behaviors**

**Chrisando Ryan P. Siahaan**

Security Researcher & Lecturer Specialist in Cybersecurity

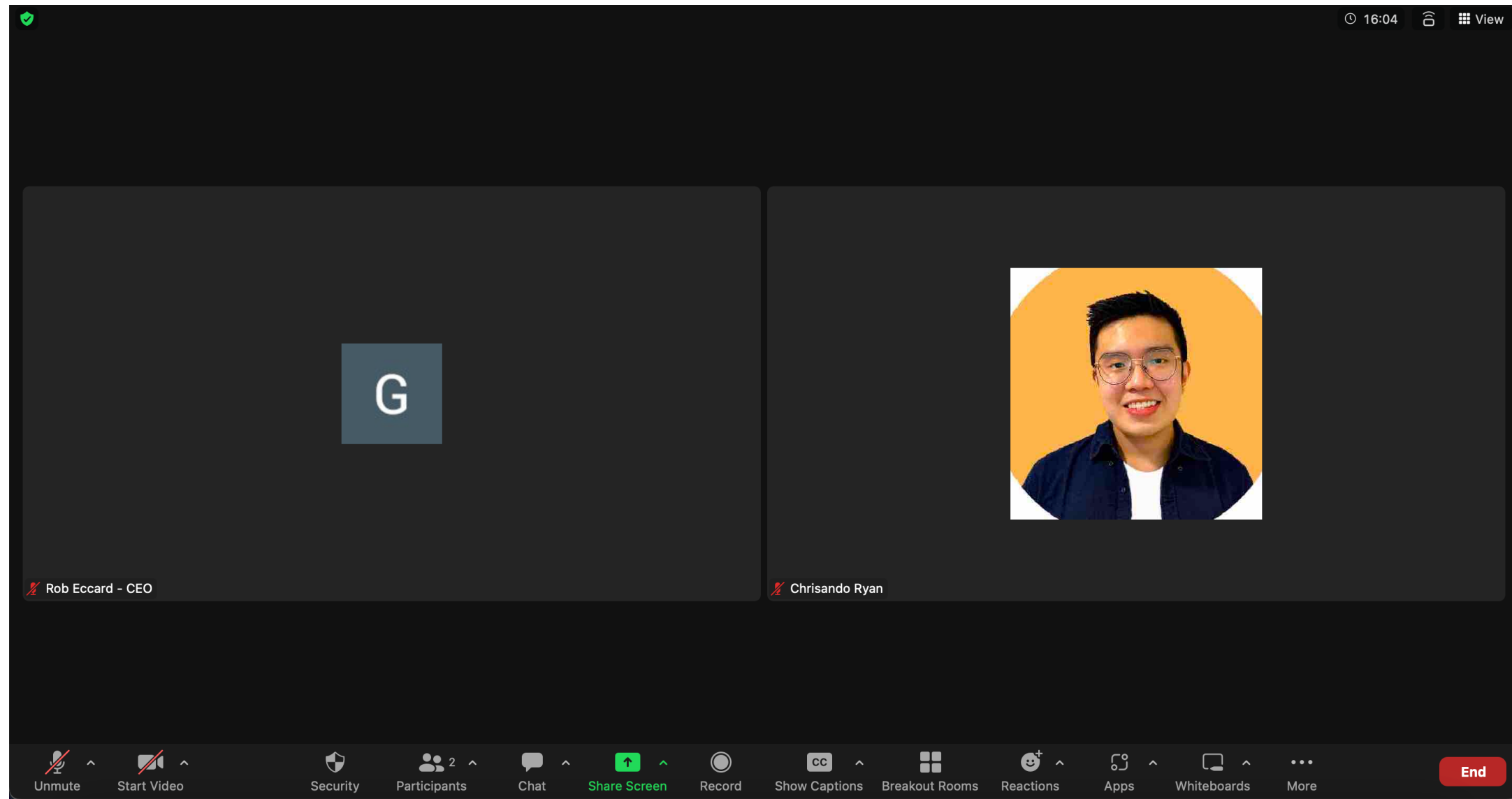
- Call me **Chrisando Ryan**, **@chrisandoryan** or **Siahaan**
- Lecturer Specialist in Cyber Security, BINUS University, Indonesia
- CEO of Questlabs ID, a security-centered software development agency in Indonesia.
- Driven to a T-shaped culture by extensively studying AI, Computer Vision, and Big Data domains as well, and intertwine them with Cyber Security.
  
- Black Hat Asia Arsenal speaker, back in 2020 (covid-era ☹️).
- CTF problem setter & judge at various competitions in Indonesia.
- Enjoy bounty hunting, building ventures, and conducting multi-disciplinary projects

# Agenda

- Backstage story.
- Others who have tried...
- Our approach.
- The danger behind all these.
- Is there any cure?
- Takeaways.

# Backstage Story

One ordinary day, you're in a Zoom meeting with colleagues.





# Backstage Story

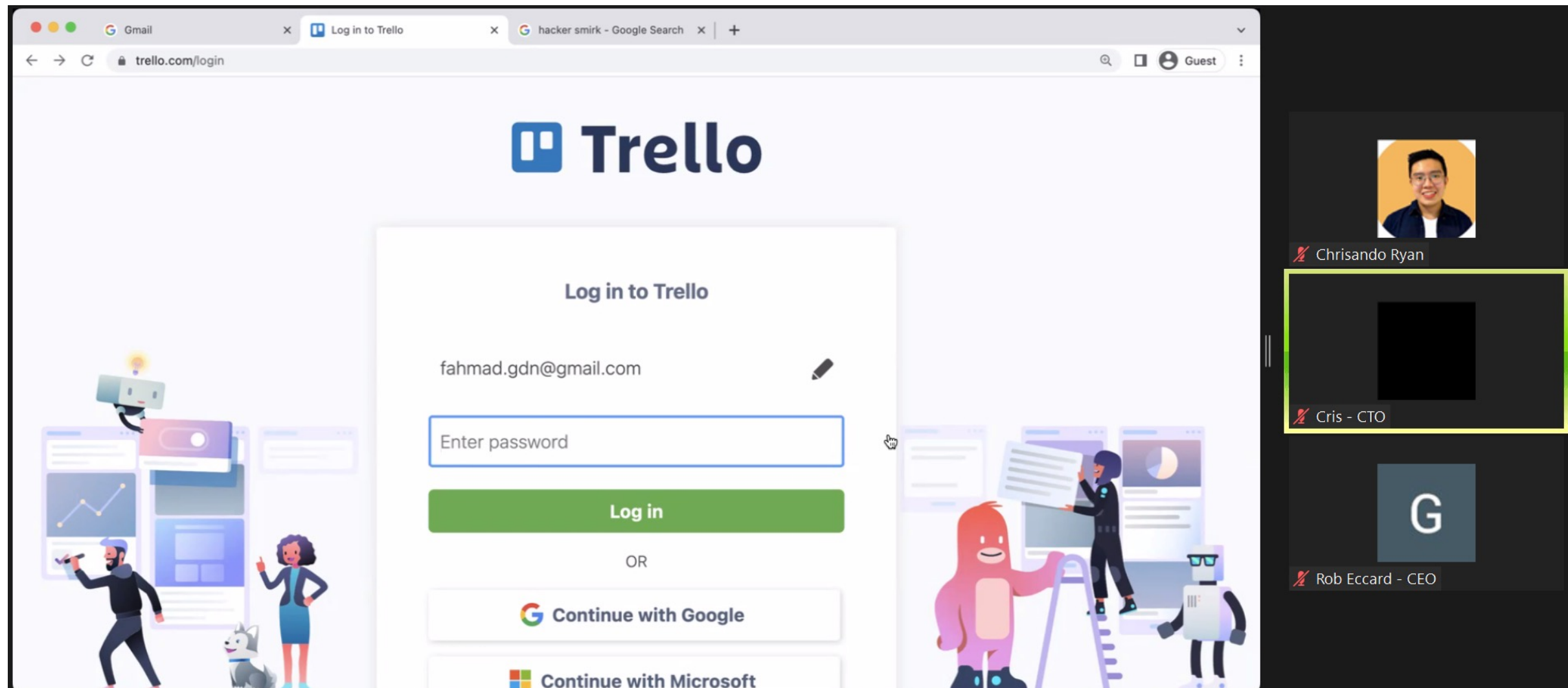
You guys are doing your stuffs, discussing back and forth.

The screenshot shows a Microsoft Word document titled "Harga.docx" with a search bar and various toolbars. The main content is a table with the following data:

Lalap	5	320	1.600
Serundeng	5	100	500
<b>Total Modal: 63.435</b>			
<b>Total Jual: 96.000</b>			
<b>Total Untung: 32.565</b>			
Total kerju			

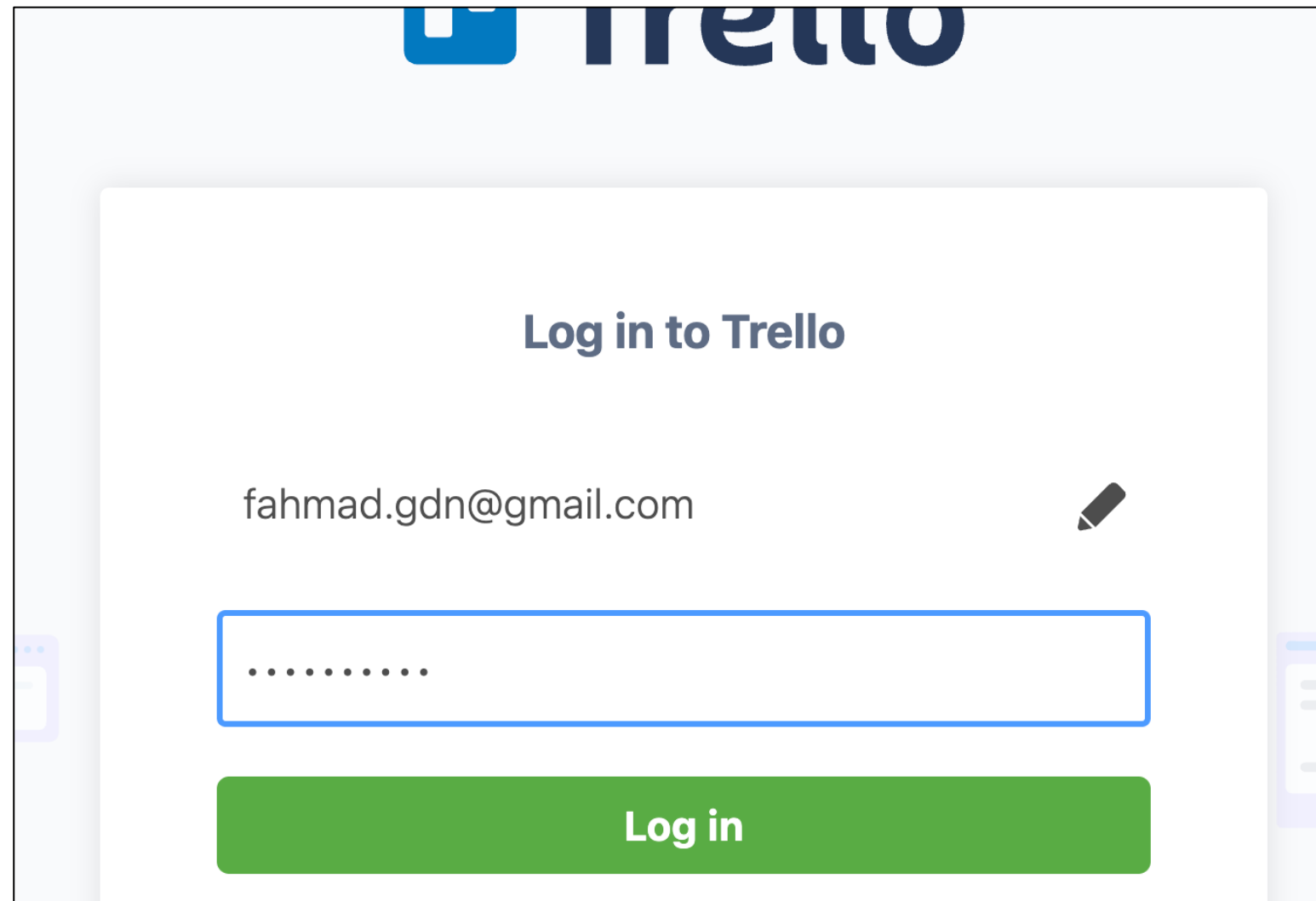
On the right side of the document, there is a video call overlay showing a participant's profile picture. The interface includes tabs for "Comments", "Editing", and "Share". The bottom status bar shows "Accessibility: Good to go" and "100%" zoom.

Suddenly, while a colleague's sharing their screen, they stumbled upon a page which forces them to do a sign-in using password.



# Backstage Story

**Too lazy** to stop the screen-sharing temporarily, they choose to continue typing their password;  
Thinking that these **black bullet-mask symbol** **will protect them...**



All those led to our attempt to test a hypothesis:

*If we are to live in a world where most meetings will be conducted through online video meetings...*

*Then it might be possible to **leak** and **mimic** a user's typing behaviour through a **screen-sharing video** alone.*



So, about **typing behavior**.

- A term coined as **keystroke biometrics**, that is: the process of measuring and analysing an individual's unique typing patterns or rhythms on a computer keyboard.
- Used for?

**User authentication**

**Fraud detection**

**Forensic analysis**

- But, how?

**Time between keystrokes (called *Inter-key Latency*)**

**Duration of each keystroke (called *Hold Latency*)**

**Pressure or force applied to the keys**

## The Recipe (Simplified):

How short (or long) **the delay between** each keypress

+

How short (or long) **you hold-press** each character

+

(\*\*) **How hard/strong you press** the character keys

+

(\*\*) **How many typos/mistakes** you made

=

**Authentication/Classification.** [i.e., Welcome, Bob!]

(\*\*): less-common metrics

- Although not as widely-accepted as fingerprint, but **keystroke biometrics** are gaining attention.

**coursera**

**Verification:** Signature Track is the first program of its kind on a MOOC platform that links your online coursework to your real identity. By matching your photo and creating a biometric profile of your unique typing patterns, Coursera lets you easily participate wherever you may be in the world.

**The New York State DMV and European Banking Authority approve typing biometrics as a compliant method of identity authentication.**



- Some major players in the **keystroke dynamics** industry.



**KeyTrac**

**typingdna**



**BehavioSec**

# The Idea, Originally

~~Create a better **keystroke dynamics** approach as a way to robustly  
**authenticate legitimate users**~~

Create a technique to extract **typing key-delay** out of a screen-  
recorded video,

...and maybe use them for despicable reasons >:)

(we'll come back for this)



# The Idea, Originally

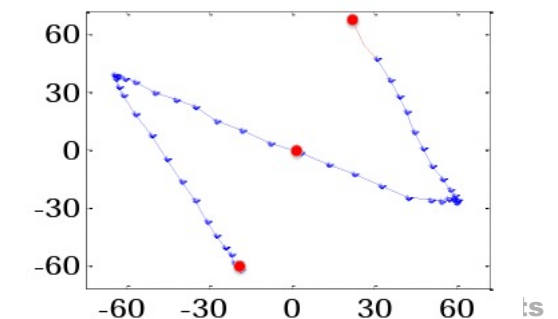
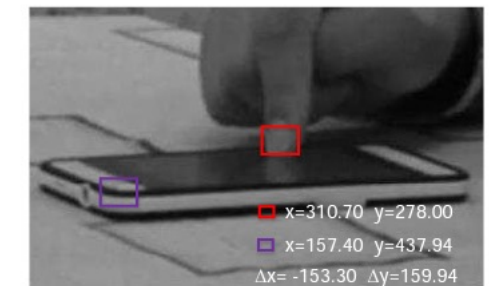
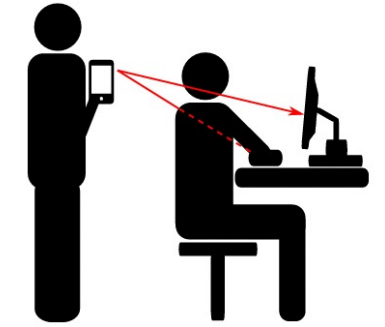
We call this technique, **Camstroke**.

## **Camstroke** Private

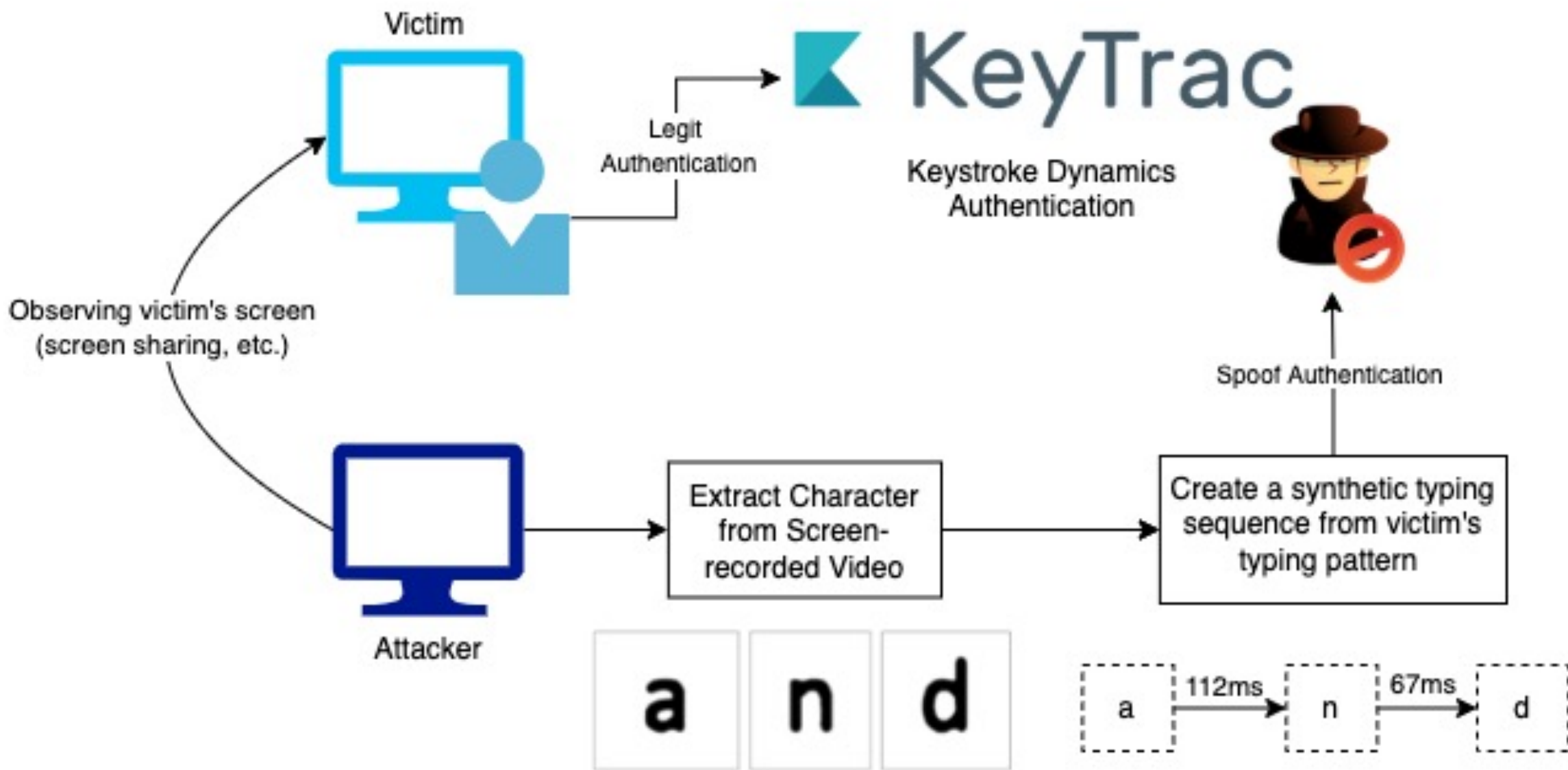
Annotate keystroke from recorded typing video, a utility for video-based  
Keystroke Inference Attack

# Others Who Have Tried...

- Silk-tv: Secret information leakage from keystroke timing videos (Balagani, et al., 2020)
  - Studied the **leakage of user secrets** (*password* and PIN) from typing activities.
  - Use video footage of a computer/ATM machine screen where password masking characters are displayed when users type their password/PIN.
  - Extract **inter-keystroke** timing information from the video and feed them to Random Forest (RF) classifier to predict the typed password/PIN.
- Cracking Android pattern lock in five attempts (Ye, et al., 2017)
  - Proposed a novel video-based attack to **reconstruct Android lock patterns**.
  - **Does not require** the video to capture **any content displayed on the screen**, only **fingertip movements**.
  - Use TLD (*tracking—learning—detection*) algorithm to generate movement trajectory.



# Our Approach, Originally



# Our Approach, Originally

During this study, we encountered (at least) **four** of the most pain-staking, mind-bending, and brain-melting obstacles:

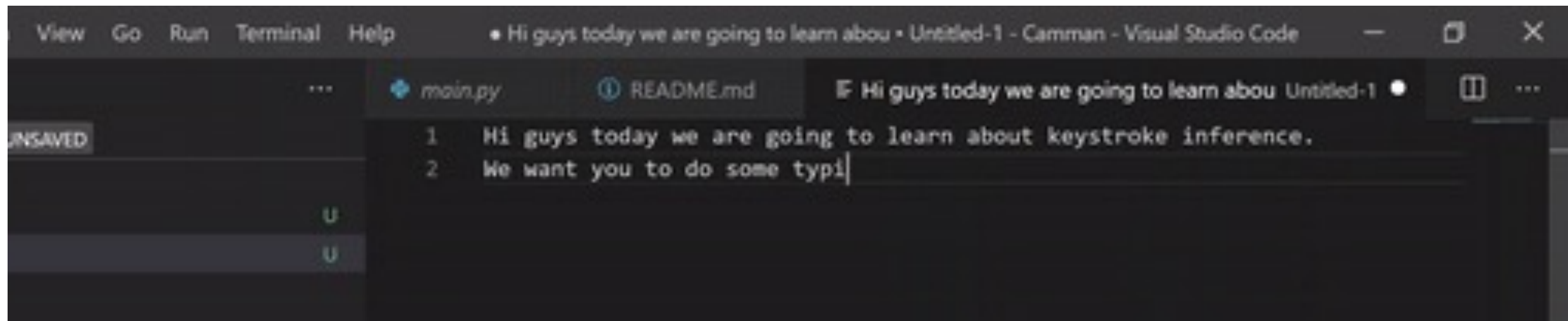
- **Challenge #1:** How to **detect when a user is typing** from a mere screen-recording video data?
- **Challenge #2:** How to **detect what character is typed** by the user **on each millisecond** of the video?
- **Challenge #3:** How to **reconstruct the victim's typing pattern** (extract each keystroke's timing/delay information)?
- **Challenge #4:** How to **predict/expose a victim's password** from **the leaked typing pattern**?



- First things first. From a mere **screen-recording video**, we need to:
  1. **Isolate the segment** of the video where a **typing activity** occurs.
  2. **Extract what character** is being **typed** by the user **on each millisecond** of the video.

**The AHA! 💡 Assumption:**

**the most recent typing activity** must be **occurred to the left of a text cursor object** that appears on the screen.



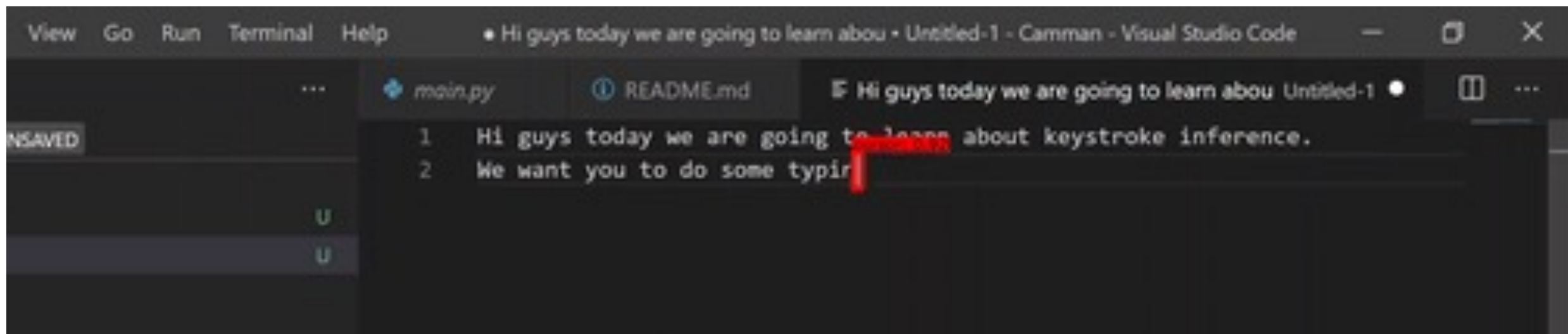
The screenshot shows the Visual Studio Code interface. The top menu bar includes 'View', 'Go', 'Run', 'Terminal', and 'Help'. The title bar reads 'Hi guys today we are going to learn about - Untitled-1 - Camman - Visual Studio Code'. The editor area shows two files: 'main.py' and 'README.md'. The 'main.py' file is open and contains two lines of text: '1 Hi guys today we are going to learn about keystroke inference.' and '2 We want you to do some typi|'. A cursor is positioned at the end of the second line. The 'UNSAVED' indicator is visible in the top left corner of the editor area.

## Challenge #1. Isolate the segment of the video where a typing activity occurs.

- Use **OpenCV** to identify a **moving rectangle-shaped object** (i.e., the **Text Cursor**)
  - Grayscale Conversion & Otsu's Thresholding
  - Canny Edge Detection & Bitwise XOR
- Identifies location of the **Text Cursor**, called **Cursor Bounding Box (CBB)** **(the red box)**

## Takeaway

The occurrence of CBB marks the start of the video segment with typing activities.



The screenshot shows the Visual Studio Code interface. The main editor window displays a text file with the following content:

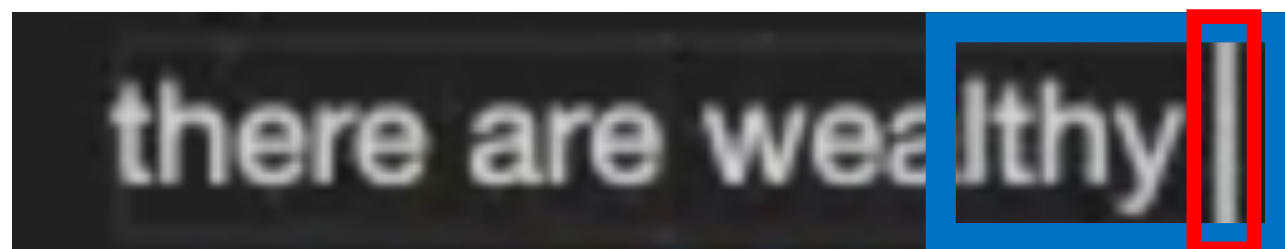
```
1 Hi guys today we are going to learn about keystroke inference.  
2 We want you to do some typir
```

A red vertical bar, representing the Cursor Bounding Box (CBB), is positioned at the end of the second line of text. The interface includes a menu bar (View, Go, Run, Terminal, Help) and a file explorer on the left showing files like 'main.py' and 'README.md'.

## Challenge #2. Extract what character typed by the user on each millisecond of the video.

- We generate another bounding-box, called **Isolation Bounding Box (IBB)** relatively to the **left** of the **Cursor Bounding Box (CBB)** coordinates.
- However, in a single IBB region, there might be more than one character captured 😞.
- Hence, we need to **know, which one is the most recently-typed character?**

Frame #087



- Cursor Bounding Box (CBB)
- Isolation Bounding Box (IBB)

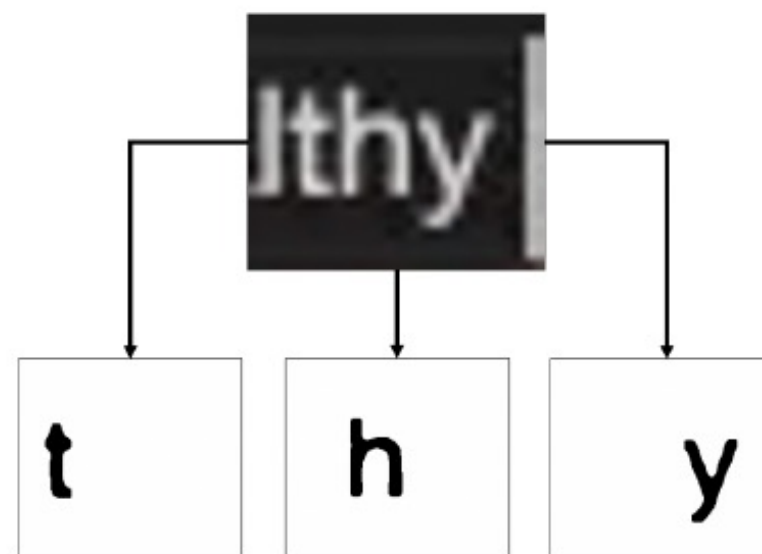


Fig: IBB with multiple characters captured

**Challenge #2. Extract what character typed by the user on each millisecond of the video.**

- Use **Connected-Component Labeling (CCL)** to separate multiple characters from the IBB frame.
- Yields the following components:
  - Background Region
  - Tallest Region (aka Text Cursor)
  - Rightmost Character
  - Previous-typed Character

## Takeaway

The **Rightmost Character**, always located to the left of Tallest Region, indicates the most-recently typed character on the frame.

Frame #087



- Background
- Tallest Region
- Rightmost Character
- Previous-typed Character



## Challenge #3. Reconstruct the victim's typing pattern (extract each keystroke's timing/delay information)

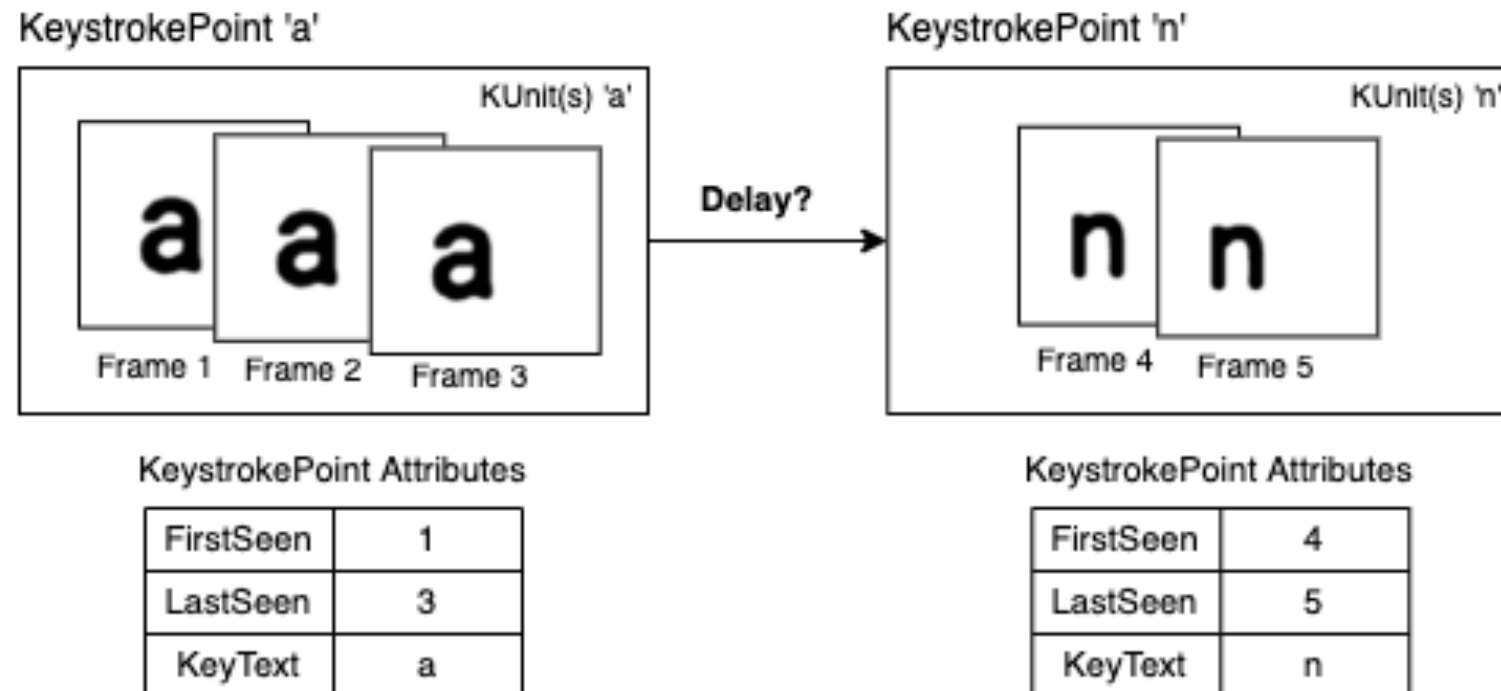
- From every frame in the video, we extract a character (aka the **Rightmost Character** component) and convert them to digital data with **OCR (Optical Character Recognition)**.
- A single character in a single video frame is called a **KUnit**.
- But we observe that the same character might **appear in more than one frame consecutively**.
- **Why?**



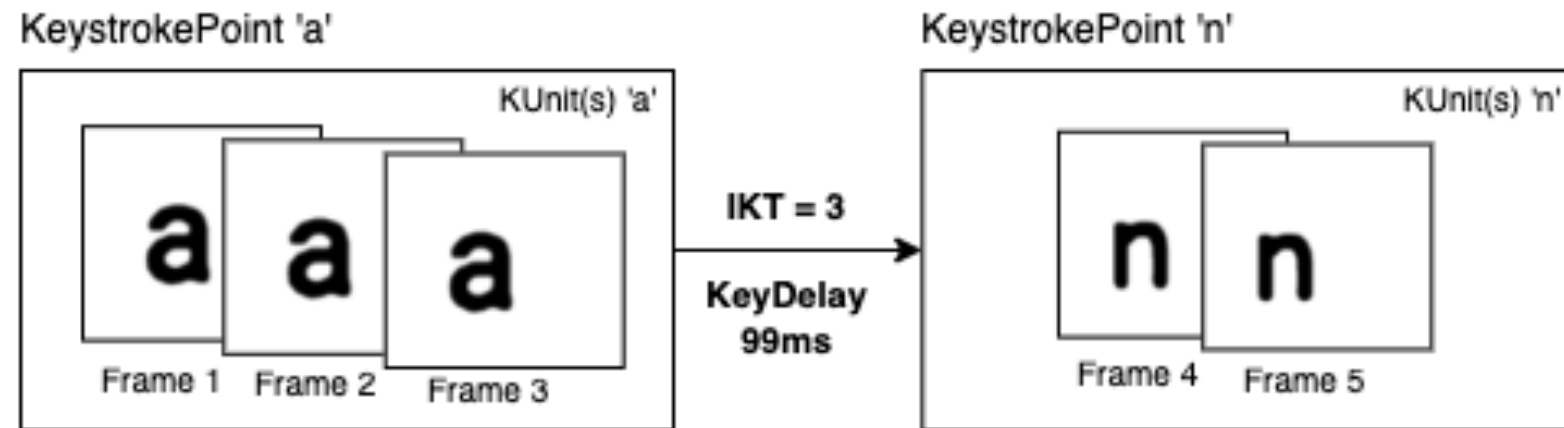
# Typing Pattern Reconstruction

**Challenge #3. Reconstruct the victim's typing pattern (extract each keystroke's timing/delay information)**

- If the same character appears in more video frames adjacently, we can **assume the longer the key-delay** of that character.
- Hence, we group characters from different frames based on similarity of the character's coordinates relative to each other. We named it **KeystrokePoint**.



**Challenge #3. Reconstruct the victim's typing pattern (extract each keystroke's timing/delay information)**



- On the image, character 'a' is displayed in 3 video frames. Hence, the **key-delay** is **99ms**.
- **Why?**
- Because every video frame lasts for 33ms (30FPS). Thus, 3 video frames last for **99ms**.

## Takeaway

The higher the number of KUnits (frame) are inside a KeystrokePoint (group), we assume the longer the key-delay of that character.

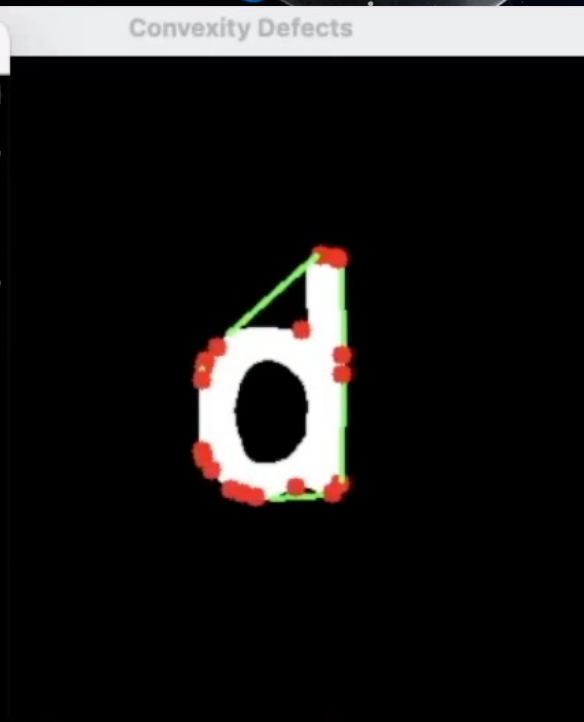


# Attack in Action

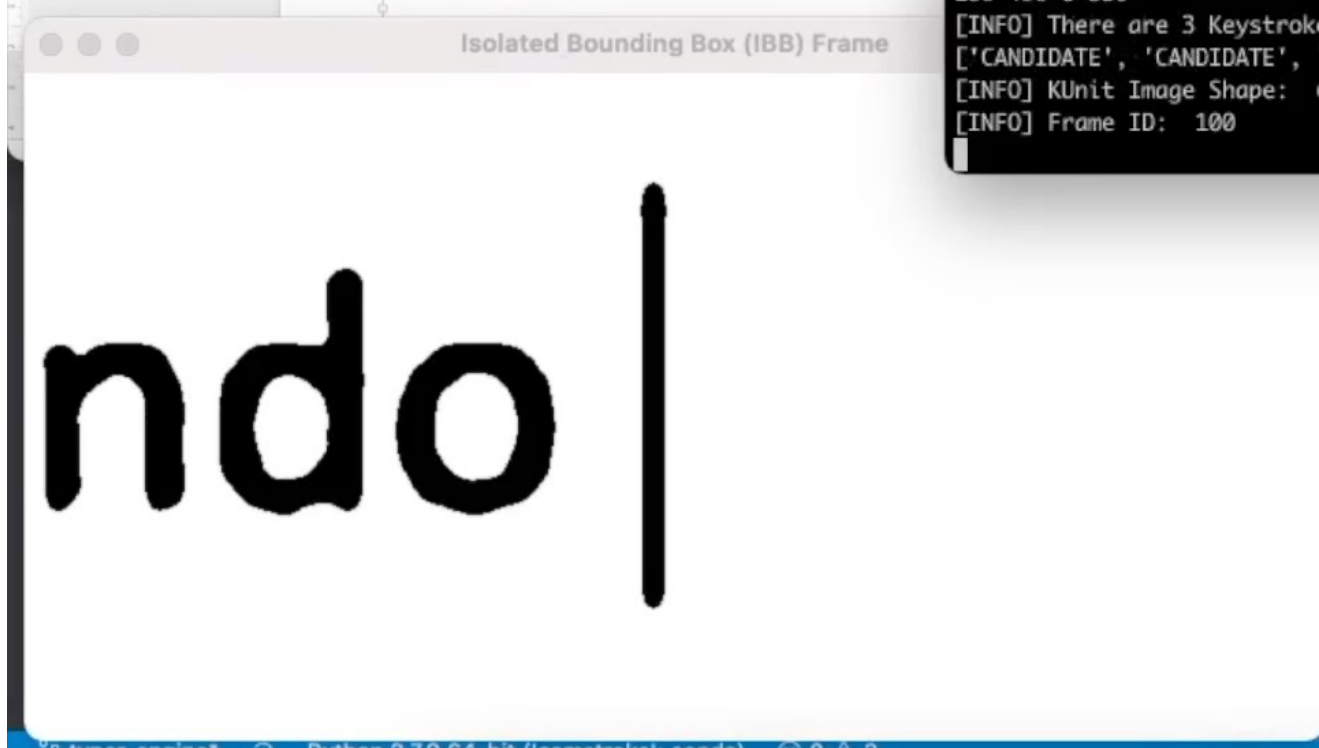
The image displays a handwriting analysis application interface. On the left, a document titled "Project Camstroke Data Collection" contains text: "Subject: S-002", "Gender: Male", "Age: 22", "Zoom: 150%", "Font: Calibri, 12pt", "Video: MacBook Pro, 1440x900, 30FPS". Below this, a "Predetermined Sentence Phase" section shows "Text to type:" with a text box containing "hi my name is <name> and i want to go and change the world". A cursor is positioned at the end of the text "hi my name is ando".

In the center, a terminal window titled "Default (python)" displays the following log output:

```
[INFO] Storing KUnit to Last KeystrokePoint with ID: 376c52e3-0e8e-4aed-b42f-eadccfabf75b
[INFO] Current KUnit coordinates: (537.7887092317341, 704.2, 546.188709231734, 713.2)
[INFO] KUnit is a RIGHTMOST candidate; Updating lastseen data...
[INFO] KUnit Shape: 84 90
[INFO] This KeystrokePoint has coordinates: (537.7887092317341, 704.2, 546.188709231734, 713.2)
[INFO] Detected Keystroke: o
Frame: 100
[INFO] White composition in Background: 0.9337941176470588
XY Ratio (0.07291221127487131, 1.0) (1.065533774809484, 1.0)
26.50334491338435 363.4966550866157 180.55227454110138 169.44772545889865
19 409 0 350
XY Ratio (0.2706543608548607, 1.0) (1.0181869779539596, 1.0)
83.07152911542451 306.9284708845755 176.5770199573726 173.4229800426274
61 451 0 350
XY Ratio (0.5308016317298414, 1.0) (1.1321980158994676, 1.0)
135.23152319919404 254.76847680080598 185.8501427212179 164.14985727878212
100 490 0 350
[INFO] There are 3 Keystroke Candidates
['CANDIDATE', 'CANDIDATE', 'RIGHTMOST']
[INFO] KUnit Image Shape: (350, 390)
[INFO] Frame ID: 100
```



Final Keystroke Image





# Wait. Why the Fuss?

- At this point, **key-delay** between characters can be extracted through a video alone.
- It is **permission-less**. No need to insert/install **keylogger** into the victim's machine.
- This attack can occur when:
  - A victim makes their screen visible to the attacker (i.e., in a Zoom meeting).
  - The attacker records the victim's screen, where the victim's typing activities is visible.
  - The delays between characters is extracted from the obtained video frames.
- Thus, this attack can be conducted remotely, without the victim even realizing.
- If the victim's account is protected with **keystroke dynamics** authentication, we can **mimic** their typing pattern and **replay them** to bypass the authentication.

# Okay. But is it Working?

Before we go into the more despicable part. **Let's see how good the algorithm performs.**

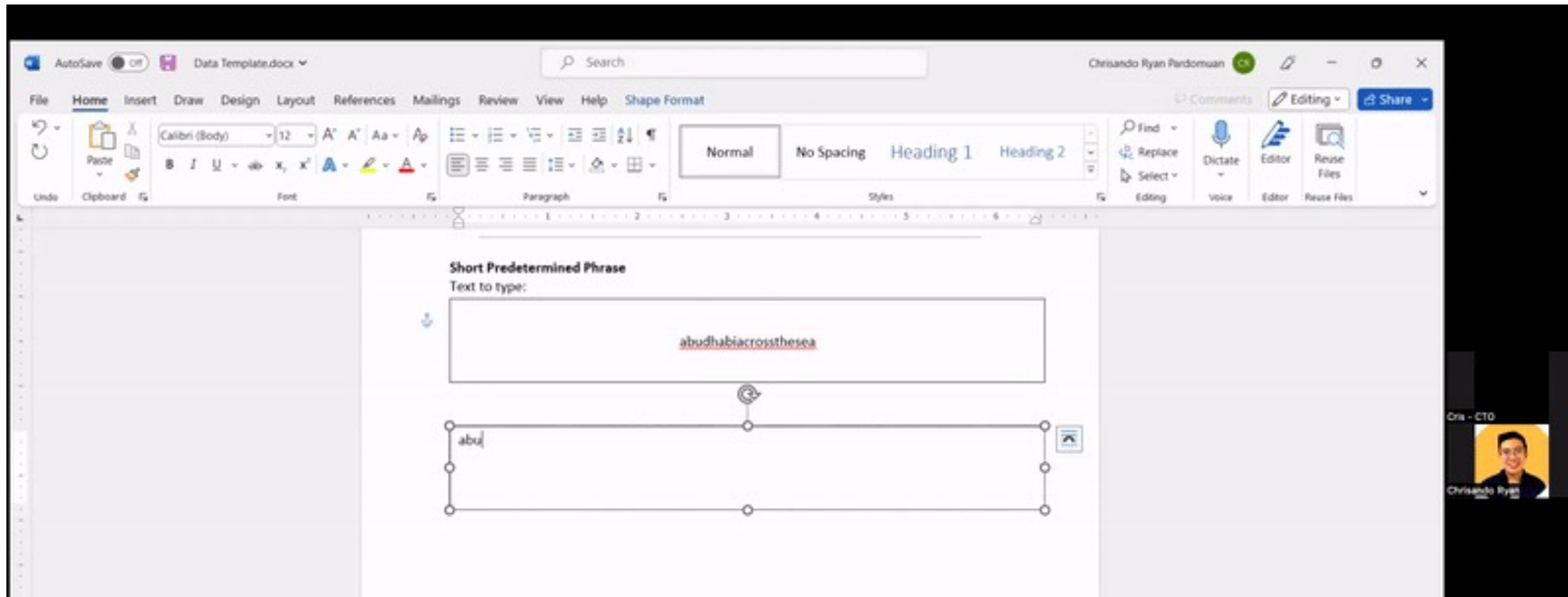
# Benchmarking the Attack

- Aspects evaluated:
    - **How similar** the **reconstructed typing pattern** of the victim? (Statistic Similarity)
    - **How effective** the attack against **KeyTrac** authentication (Evasion Rate & EER)
- 
- We tested the attack on 14 victims (with consent), each on different Zoom meetings.
  - The victims were asked to perform these **typing activities**:

No	Evaluation Group	Typed Text	Total Samples
1	Password Phrase	abudhabiacrossthe sea	210
2	Greeting Sentence	hi my name is [NAME]	
3	Long Sentence	i want to go and change the world	

# Benchmarking the Attack

We asked each victim to perform a typing activity, and record them:



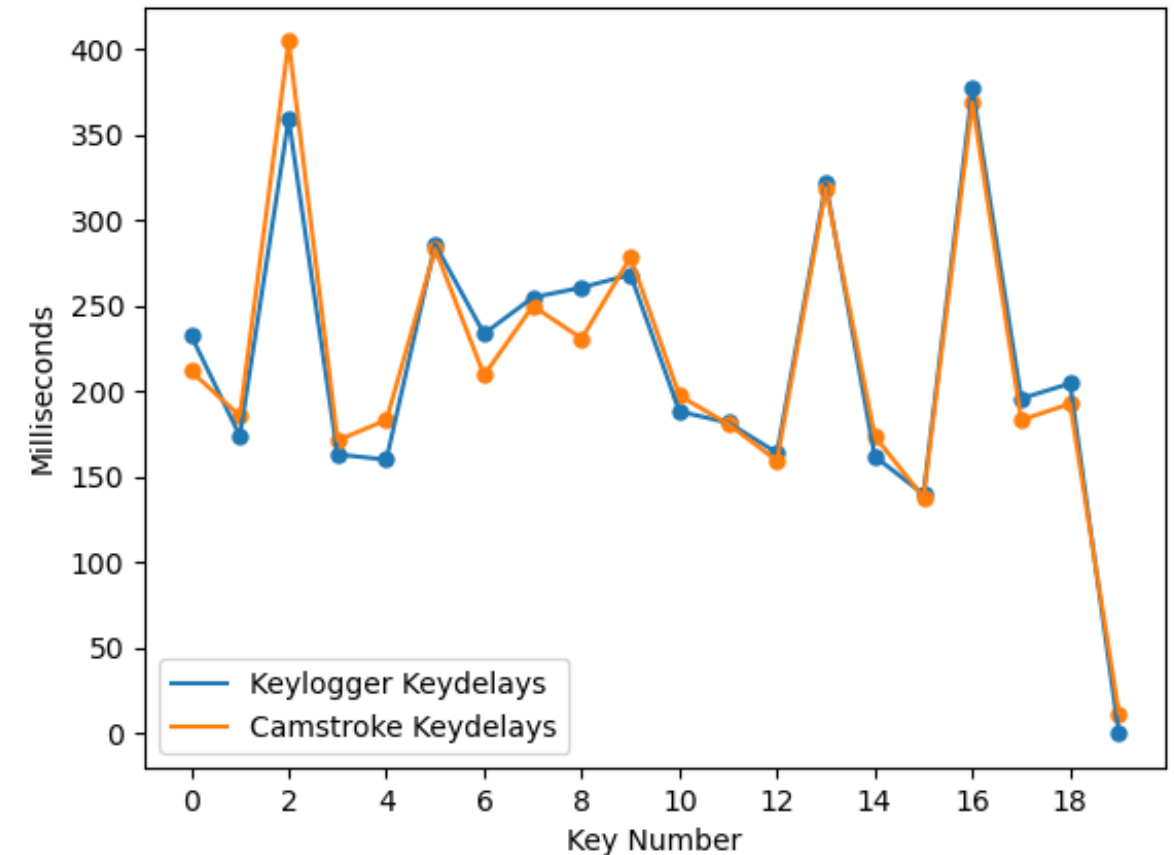
And, to collect actual key-delay data, we use **keylogger** installed on the victim's device.



# Similarity Test: Password Group

Text typed: **abudhabi across these sea**

- Tested using Shapiro-wilk Test (Normality Test), Levene's Test (Variance Test), and Wilcoxon Signed-rank Test (Mean Similarity Test).
- Both key-delays are **distributed normally** and have **equal variance**.
- There is **no significant mean differences** between the **reconstructed key-delays** and the **actual key-delays**; the data can be considered similar.
- Process time is **5.61FPS**, or **5.35x longer** than the actual video duration (30FPS).

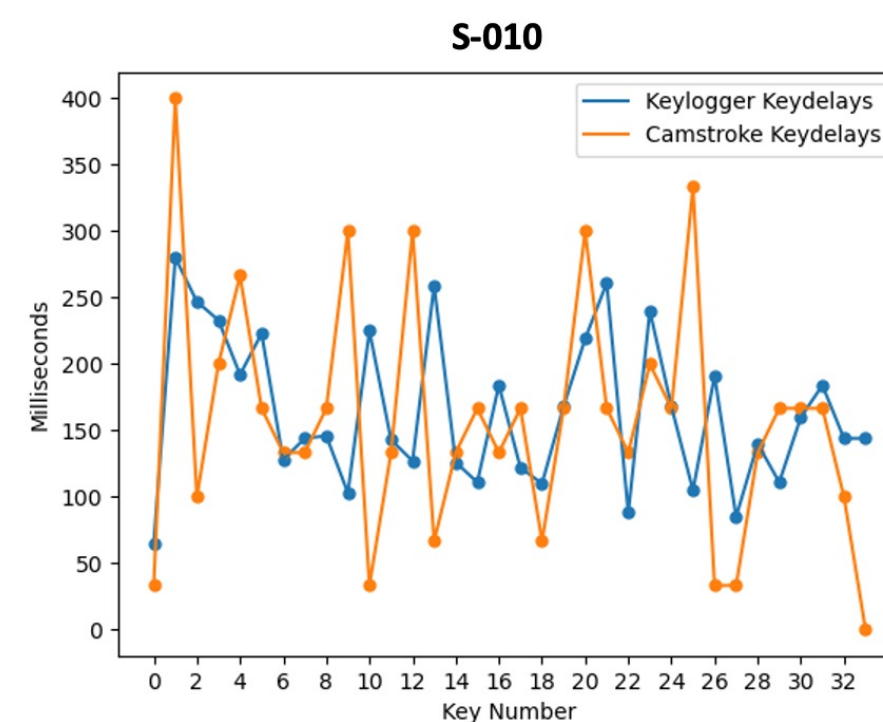
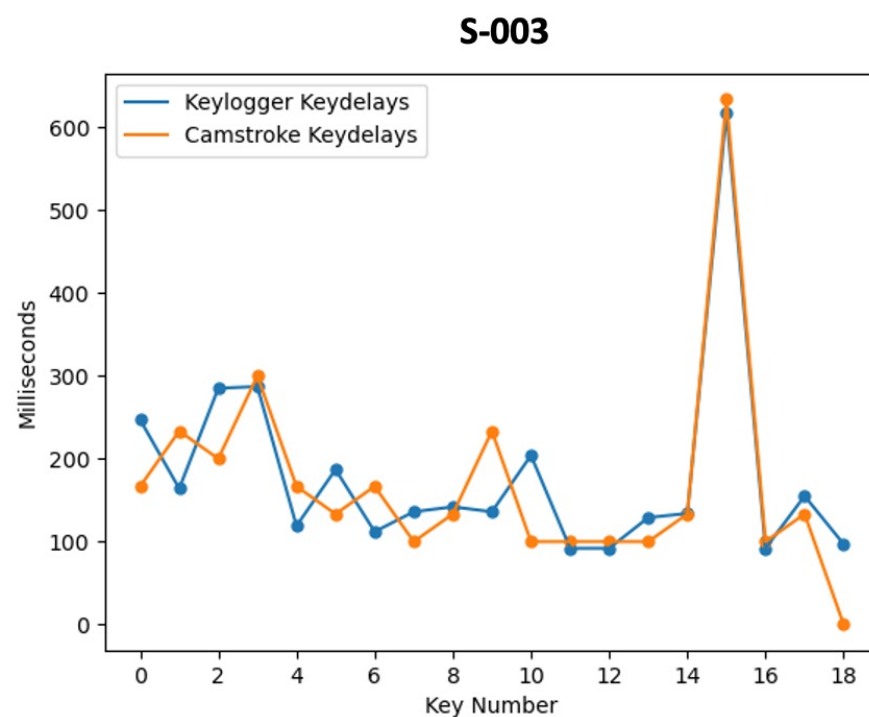
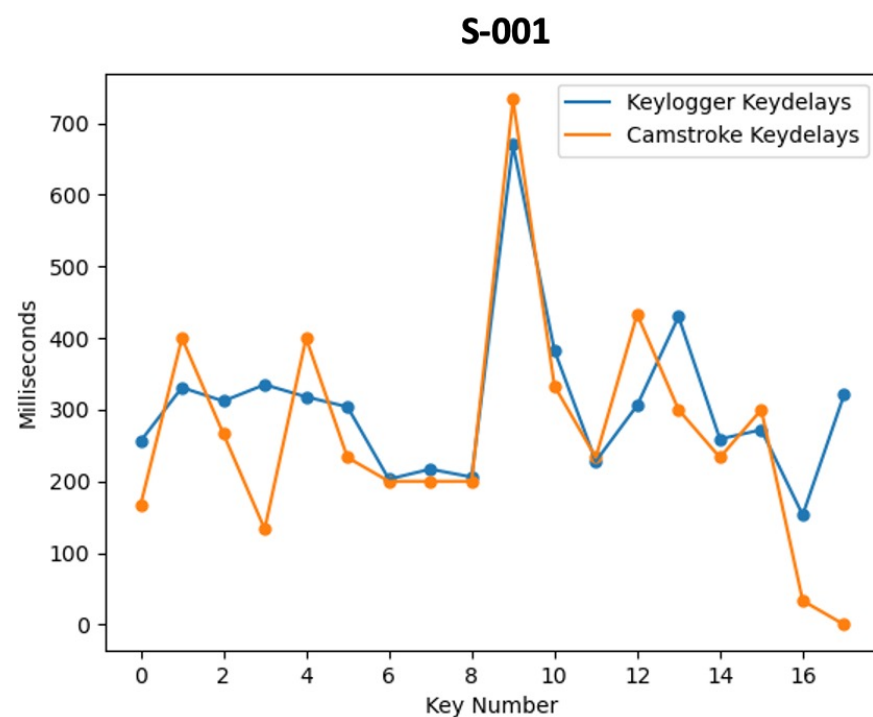


**Comparison of the Reconstructed Key-delay and the Actual Key-delay (averaged).**

# Similarity Test: Greeting Group

Text typed: **hi my name is [NAME]**

- Suffers **lower performance** compared to the **password text group**.
- There is **still no significant mean differences** between the **reconstructed key-delays** and the **actual key-delays**; the data can be considered similar.
- Process time is **6.78FPS**, or **4.43x longer** than the actual video duration (30FPS).

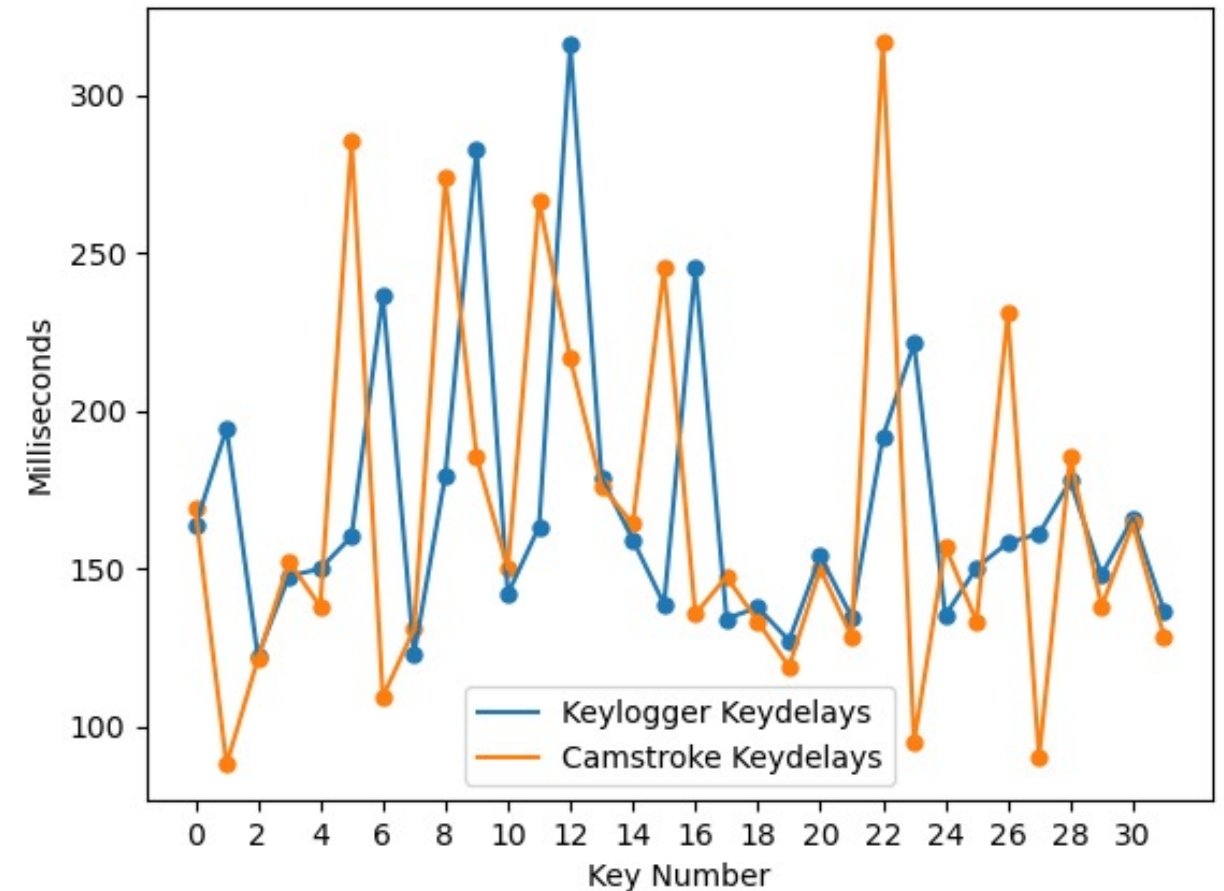


**Comparison of the Reconstructed Key-delay and the Actual Key-delay (NOT averaged).**

# Similarity Test: Longtext Group

Text typed: **i want to go and change the world**

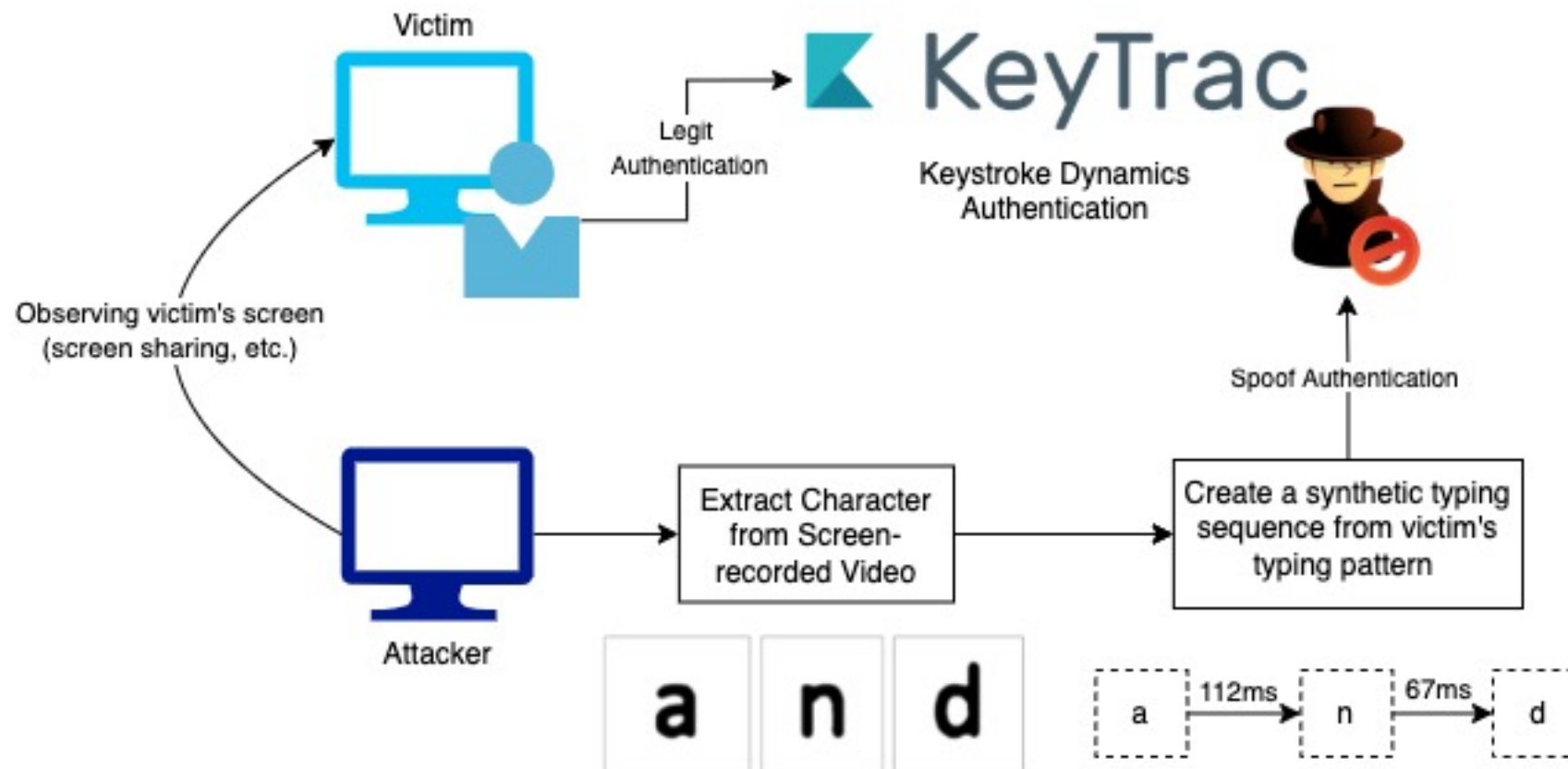
- Also suffers **lower performance** compared to the password text group.
- There is **still no significant mean differences** between the **reconstructed key-delays** and the **actual key-delays**; the data can be considered similar.
- Process time is **5.91FPS**, or **5.08x longer** than the actual video duration (30FPS).



**Comparison of the Reconstructed Key-delay and the Actual Key-delay (averaged).**



- KeyTrac is a AaaS (Authentication-as-a-service) platform that's widely used by global companies around the world.
- KeyTrac supports two modes: **Password-hardening** mode and **Freetext** mode.
- **How do we perform the attack against KeyTrac?**



## Performance Metrics

- Evasion Rate (**ER**): measures the rate of the attack being undetected.
- Equal Error Rate (**EER**): measures the increase/decrease of performance of **KeyTrac** authentication service.



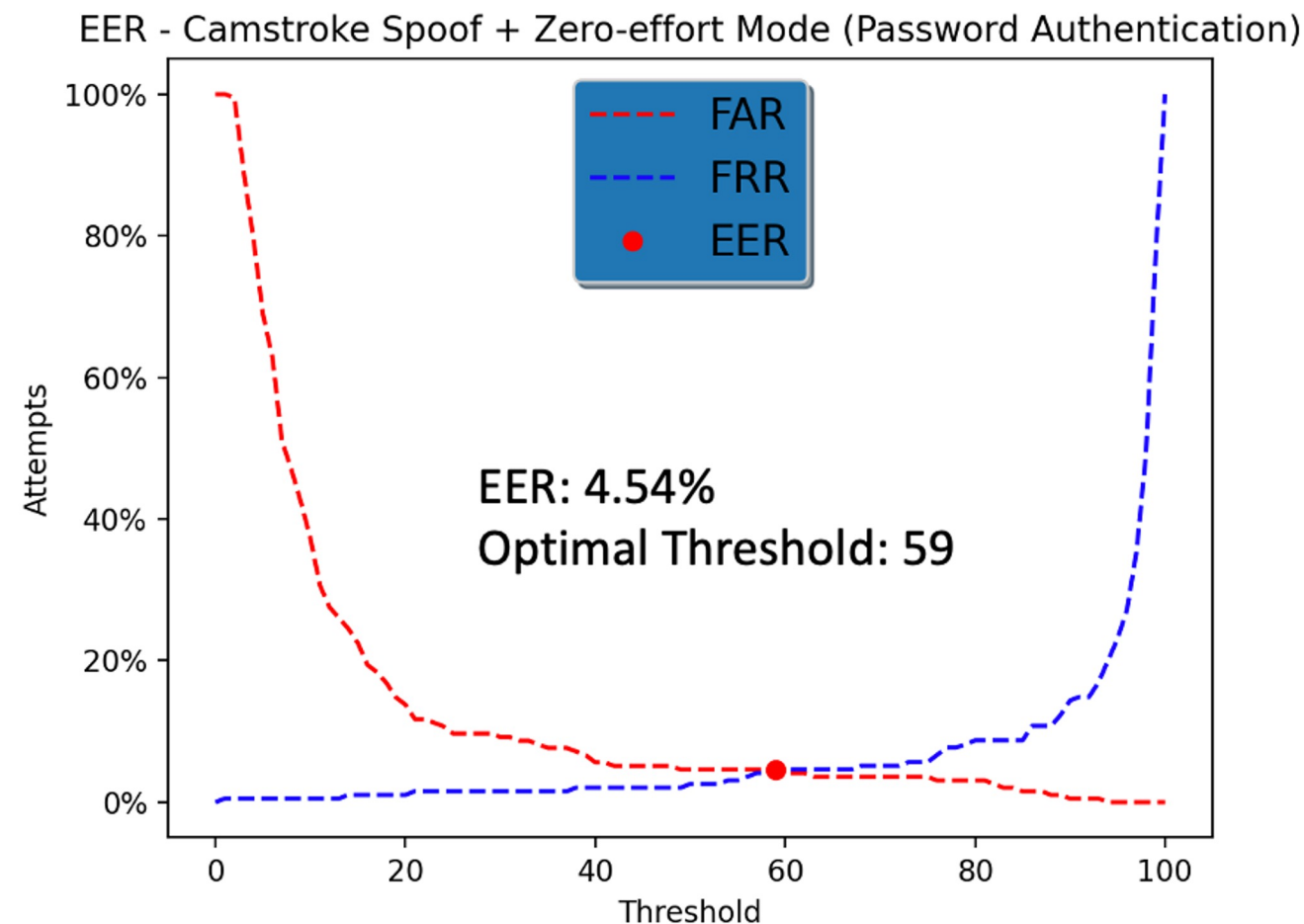
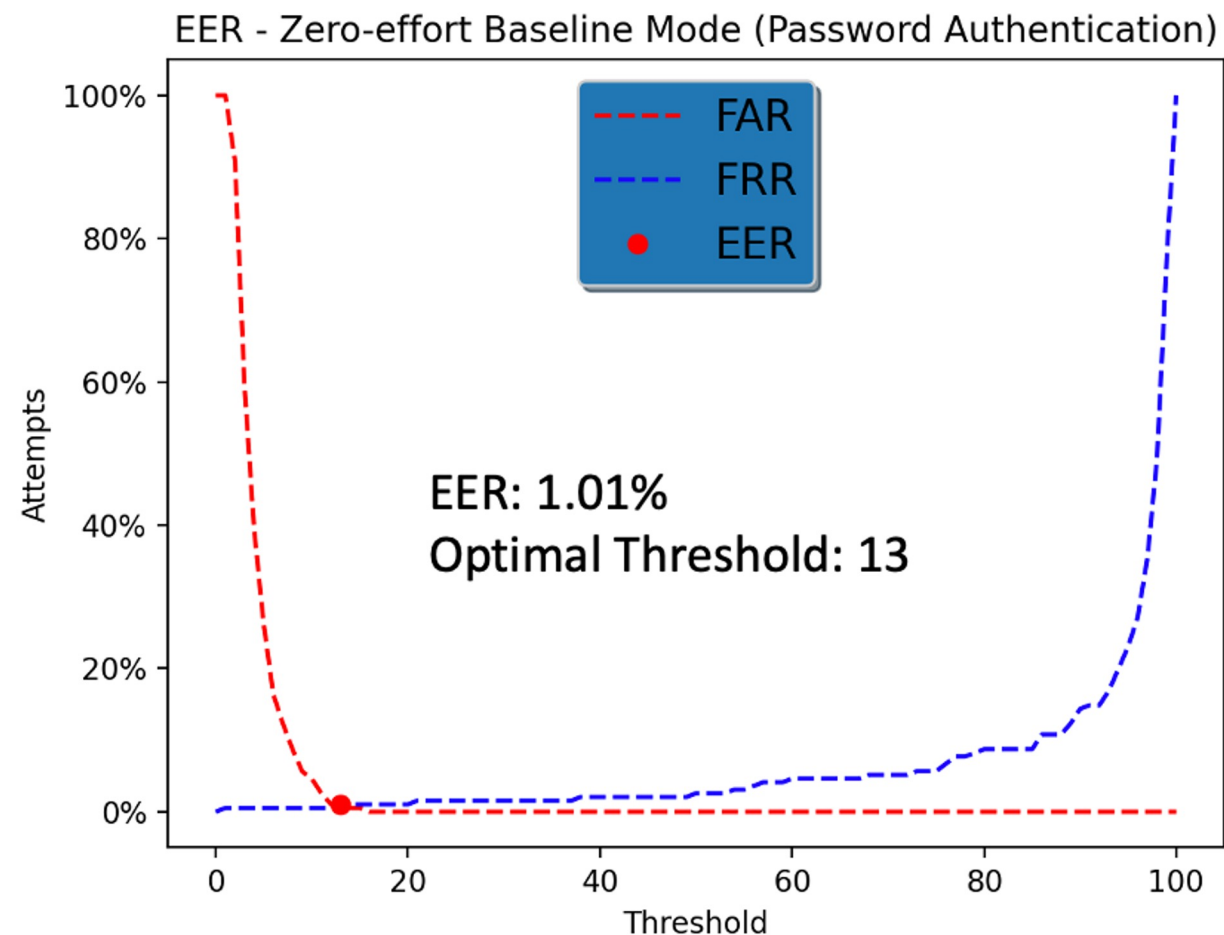
# Attack Effectiveness: FAR-FRR-ERR

KeyTrac performance in Password mode, before (left) and after (right) the attack.

EER increased **349.5% post-exploitation**,

Optimal authentication threshold **increased from 13 to 59**

**High ERR indicates the decreasing accuracy of the biometric system (due to FAR is increasing).**



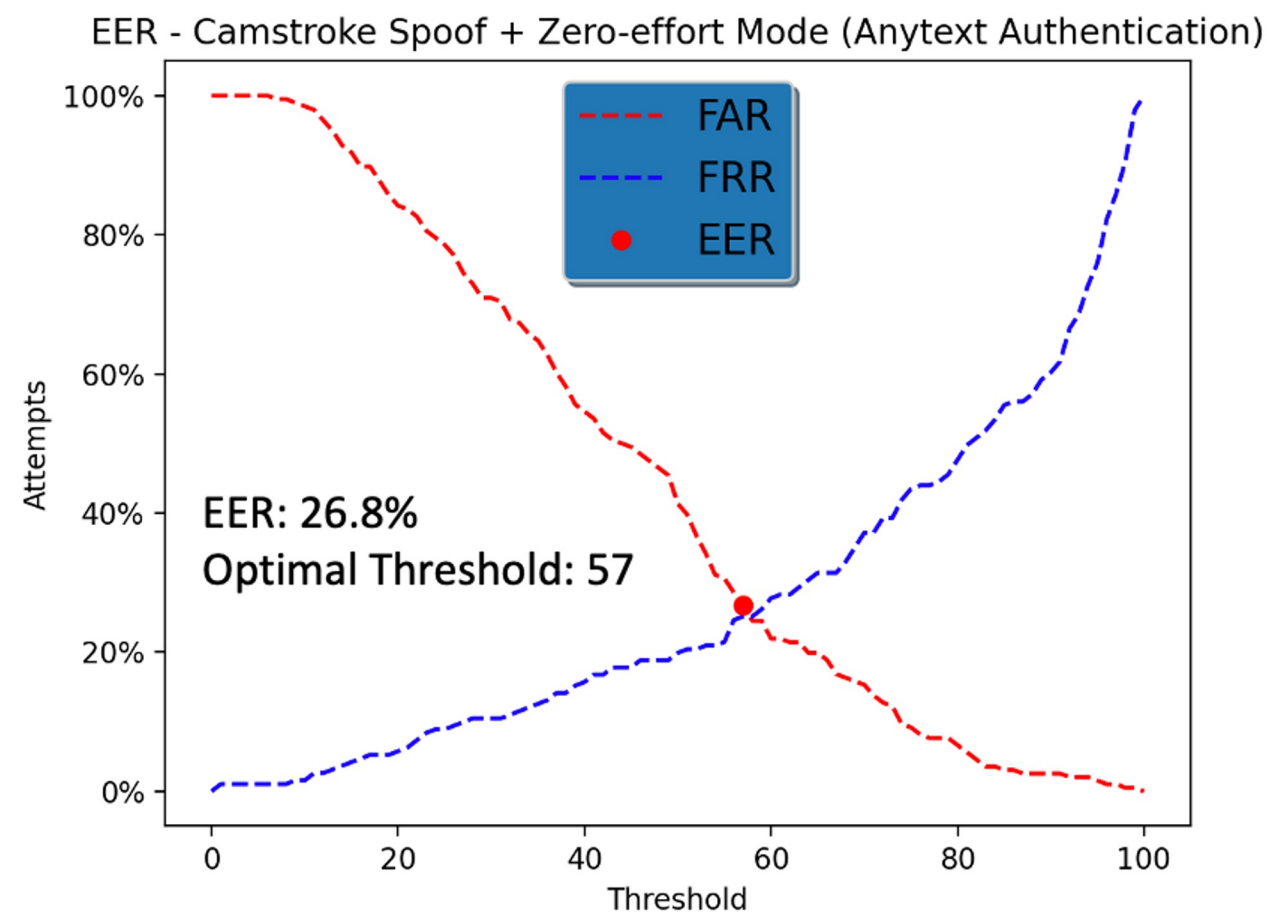
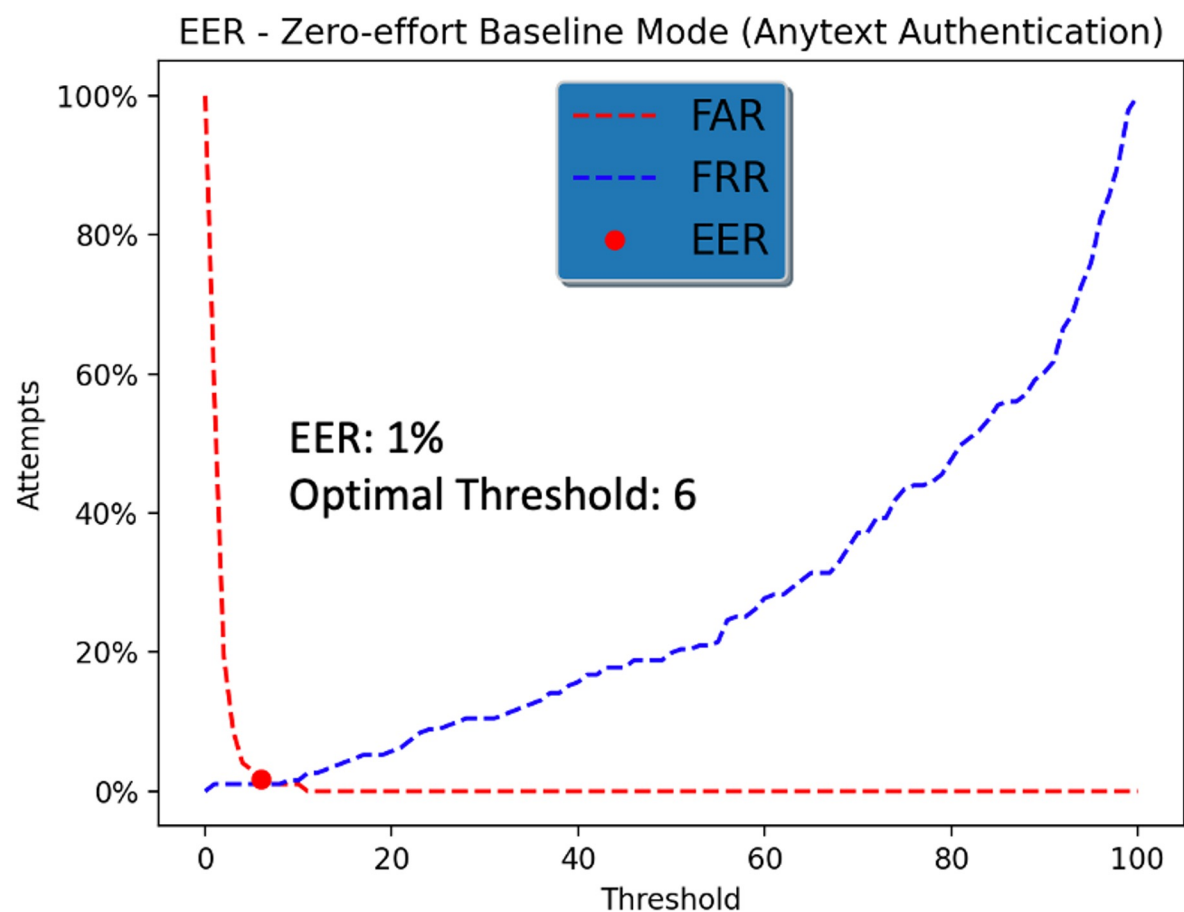
# Attack Effectiveness: FAR-FRR-ERR

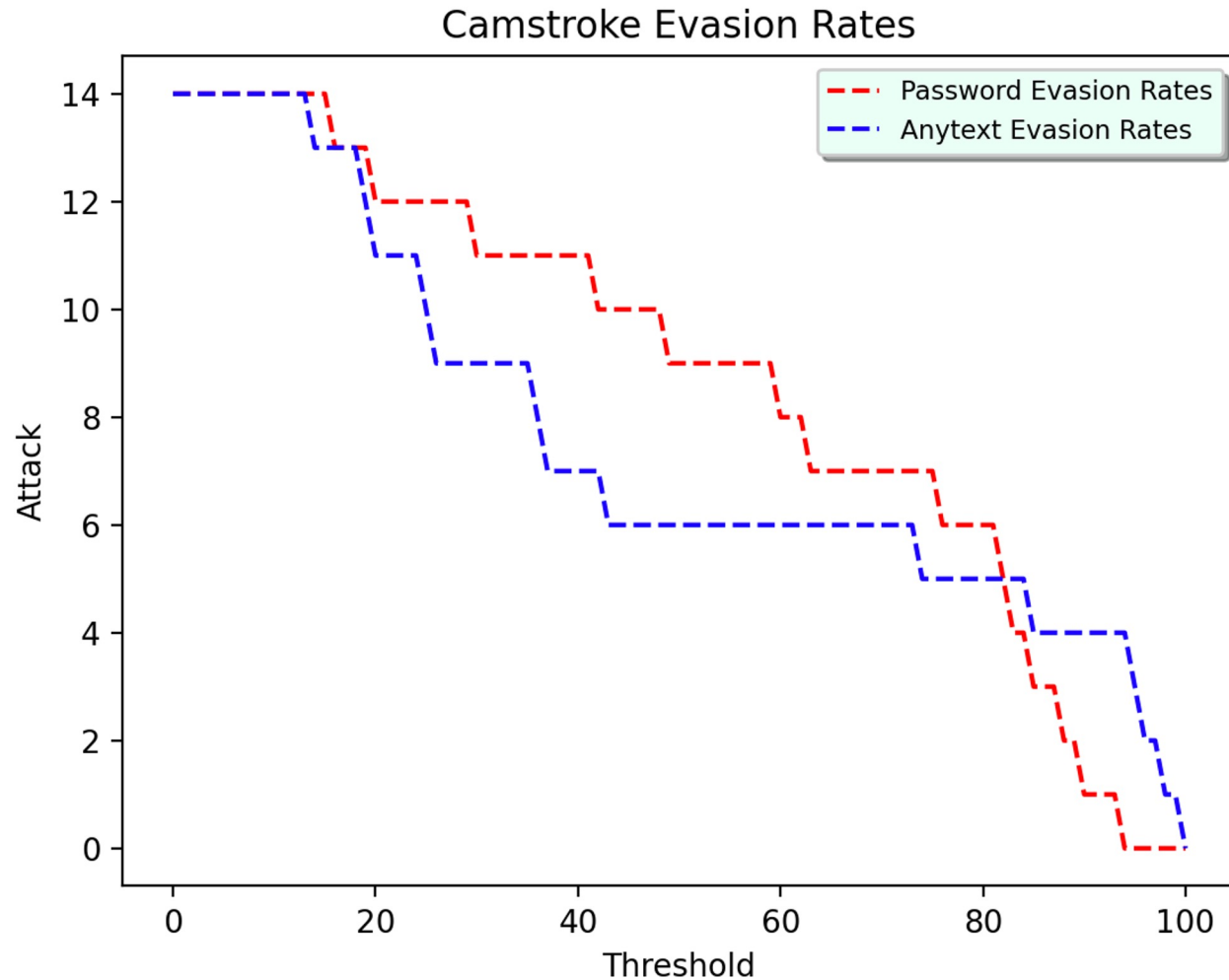
KeyTrac performance in **Freetext (Greeting + Longtext group)** mode, before (**left**) and after (**right**) the attack.

EER increased **2553.5% post-exploitation**,

Optimal authentication threshold **increased from 6 to 57**

**High ERR indicates the decreasing accuracy of the biometric system (due to FAR is increasing)**





**Evasion Rates (ER) on different authentication thresholds**

- On **Password mode** with authentication threshold of 60%, 9 out of 14 attempts successfully spoof KeyTrac into allowing the authentication to pass through.
- That means, **Evasion Rates (ER) is 67%**, or almost **2 of 3 attacks is successful**.
- Unfortunately, on **Freetext mode** with authentication threshold of 60%, only 6 out of 14 attempts successfully spoof KeyTrac into allowing the authentication to pass through.
- That means, **Evasion Rates (ER) is 43%**.

# What Have We Learned?

**The evasion rates (ER) were not able to reach beyond 70%, why?**

- This is mainly affected by the typing speed of the respondent (WPM or word-per-minute).
- The higher the WPM, the lower the number of captured frames.
- Hence, the delay similarity of the reconstructed typing pattern is also decreased.
- High WPM should be compensated with high video frame rates (FPS).

**So, are we done here? Well, not apparently.**



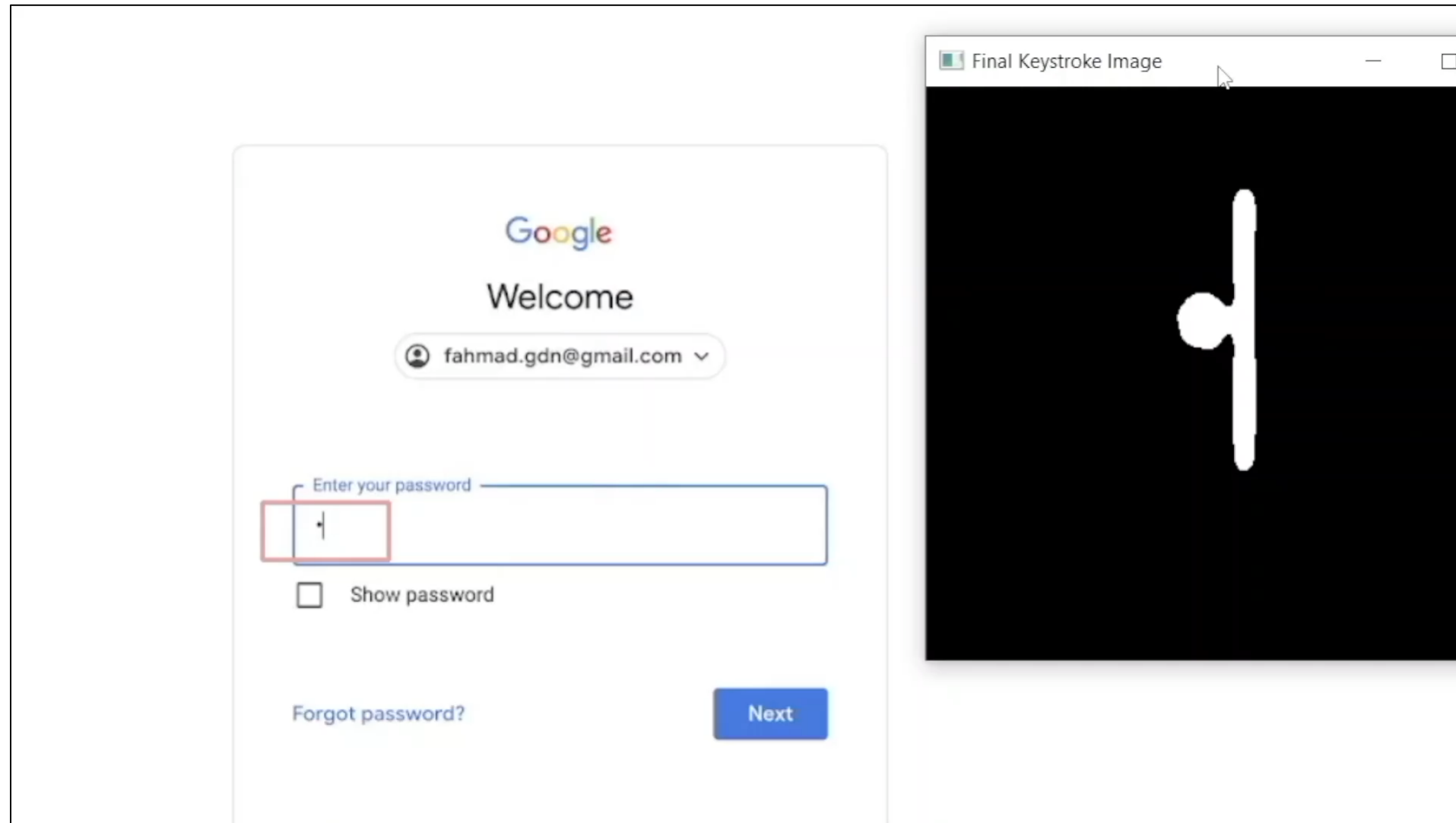
We believe the method is working, but **perfection** is still far away...

Amidst the study, we encountered **another WHAT-IF question scratching our curiosity.**

*If we're able to **track text cursor** and **extract typing pattern** out of screen-recording video,  
then what happens if **inside the video** occur **a user typing their password**?*

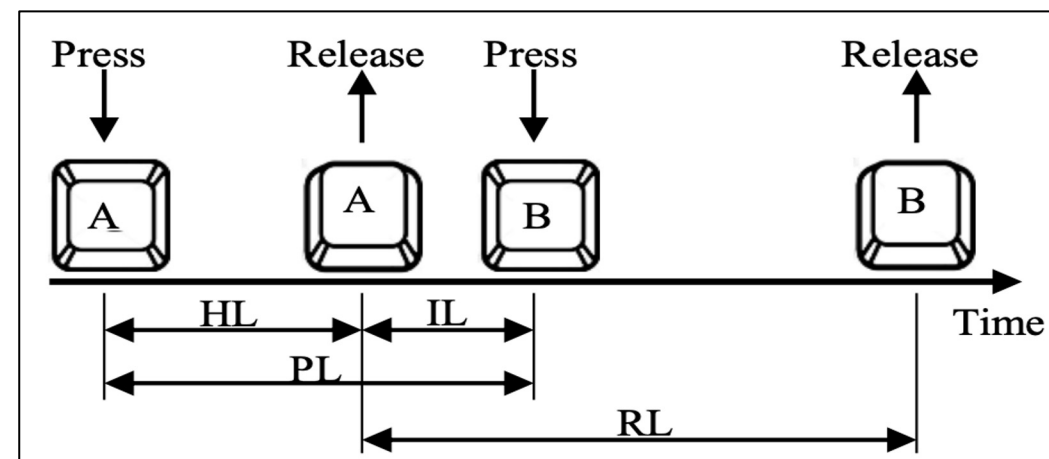
# Another Layer of Curiosity

We're able to track them as well.



# Limitations

- Typing pattern extraction sensitivity drops when there is a lot of movements, e.g., video playing, heavy screen-scrolling, etc.
- The higher the WPM requires more FPS to maintain high accuracy of the extraction. Most online-meeting platforms only support 30/60FPS video recording.
- Many **keystroke biometrics** authentication also uses **hold-delay** metric. As of now, we're only able to extract the **inter-key delay** metric.



# Is There Any Cure?

- As in many other behavior-based attacks, there are no better solution than to applies a secure user behavior to prevent the leakage.
- That means, **being mindful** on who's your audience during screen-sharing meetings.
- And **being eager** enough to stop the screen-sharing whenever we're about to input something sensitive and confidentials.
- However, we also found some projects interesting to inhibit **typing pattern** extraction, such as:
  - **Kloak** by Vinnie Monaco: introduces random delay to keyboard typing at the device level.
  - Keystroke Dynamics Anonymization System (Migdal, D., & Rosenberger, C., 2019)

 **kloak** Public

Keystroke-level online anonymization kernel: obfuscates typing behavior at the device level.

● C ☆ 406 🍷 32



1. Should **keystroke biometrics** adoption growth consistently in the next few years, we expect that more advanced mimicry (side-channel) attack will be demonstrated from unexpected sources of data.
2. By using a screen-recorded video, someone can achieve a statistically staggering similarity in key-delay timings as if they used a keylogger.
3. Relying on videos allows for the elimination of the need for any external hardware or modifications on the victim's computer (i.e., **keylogger**).