# Acoustic Keystroke Leakage on Smart Televisions

Tejas Kannan, Synthia Wang, Max Sunog, Abe Bueno de Mesquita, Nick Feamster, Hank Hoffmann

*University of Chicago*

*February 2024*

# Smart Televisions (Smart TVs)

Smart TVs are television devices which support Internet browsers and third-party applications


*AppleTV*


*Samsung Smart TV*

Images from:
https://www.apple.com/newsroom/2022/10/apple-introduces-the-powerful-next-generation-apple-tv-4k/
https://news.samsung.com/us/samsung-smart-tvs-launch-itunes-movies-tv-shows-support-airplay-2-spring-2019/

Unlike older TV systems, modern Smart TVs have areas where users input sensitive information to…

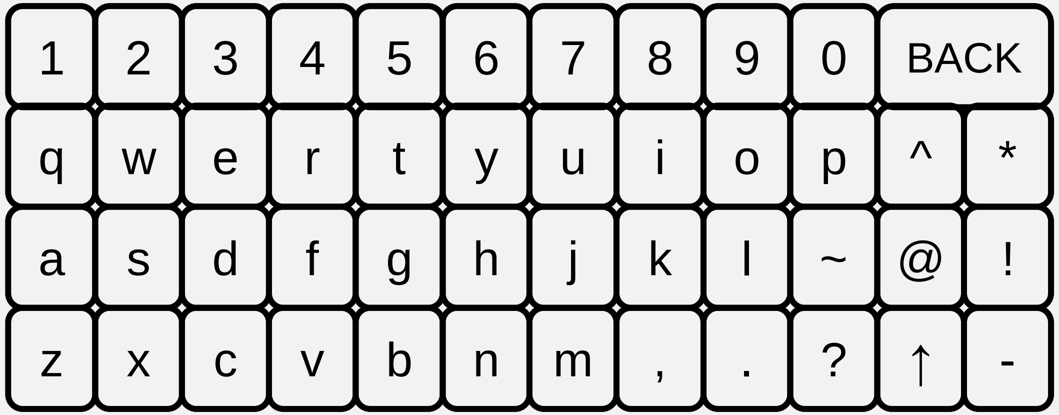
…connect to WiFi networks

…login to accounts

…make purchases

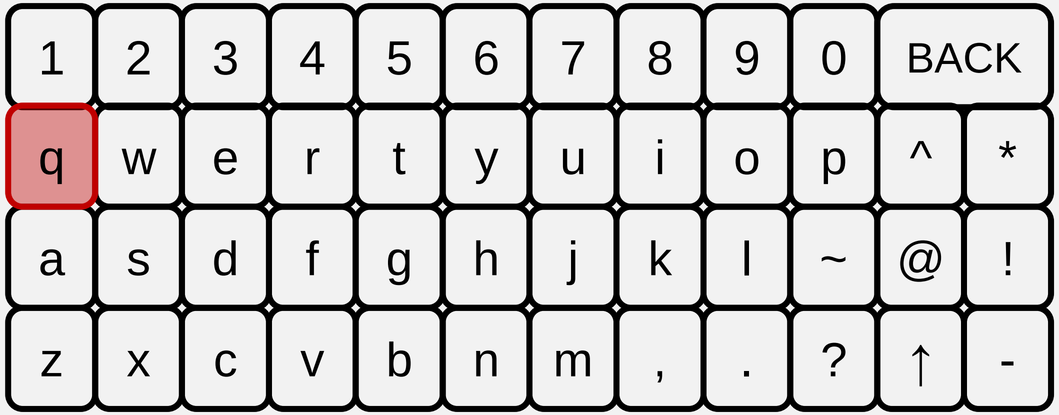
# Typing on Smart TVs

Smart TVs have virtual keyboards which allow users to enter information through a wireless remote

# Typing on Smart TVs

Smart TVs have virtual keyboards which allow users to enter information through a wireless remote

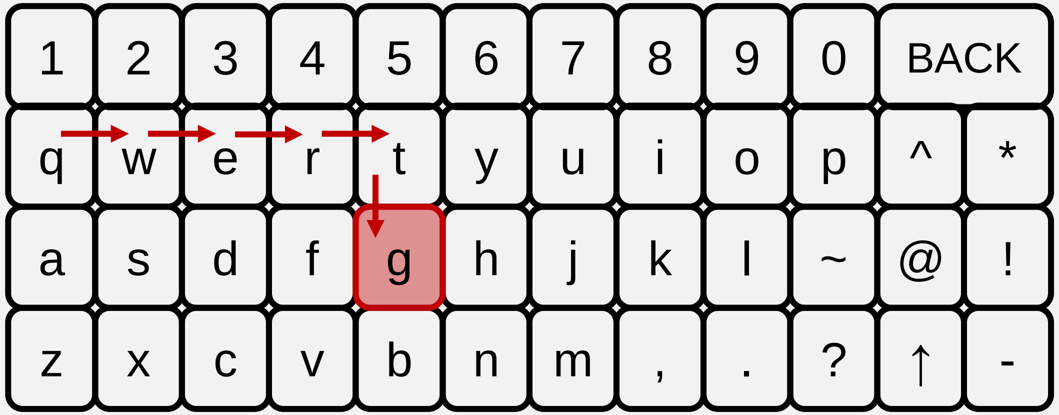




Keyboard interaction involves sequentially moving a cursor

# Typing on Smart TVs

Smart TVs have virtual keyboards which allow users to enter information through a wireless remote

Moving *right* four times and *down* once puts the cursor on ‘g’

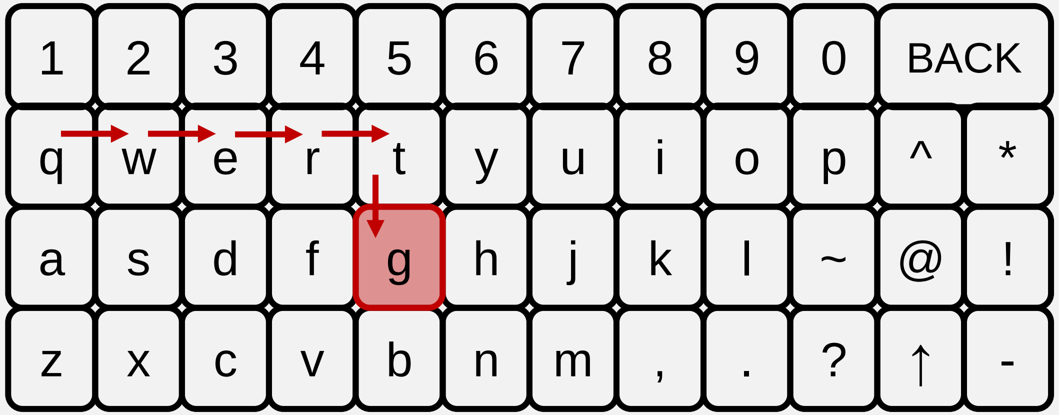


# Typing on Smart TVs

Smart TVs have virtual keyboards which allow users to enter information through a wireless remote

Moving *right* four times and *down* once puts the cursor on ‘g’

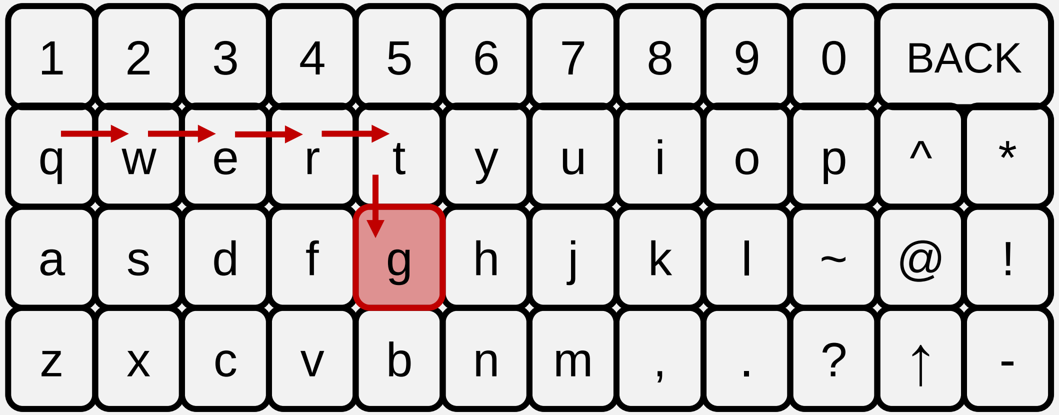
Users need not take a shortest path between keys



# Typing on Smart TVs

Smart TVs have virtual keyboards which allow users to enter information through a wireless remote

Moving *right* four times and *down* once puts the cursor on ‘g’




Users need not take a shortest path between keys

Smart TVs make sounds as the user interacts with the keyboard

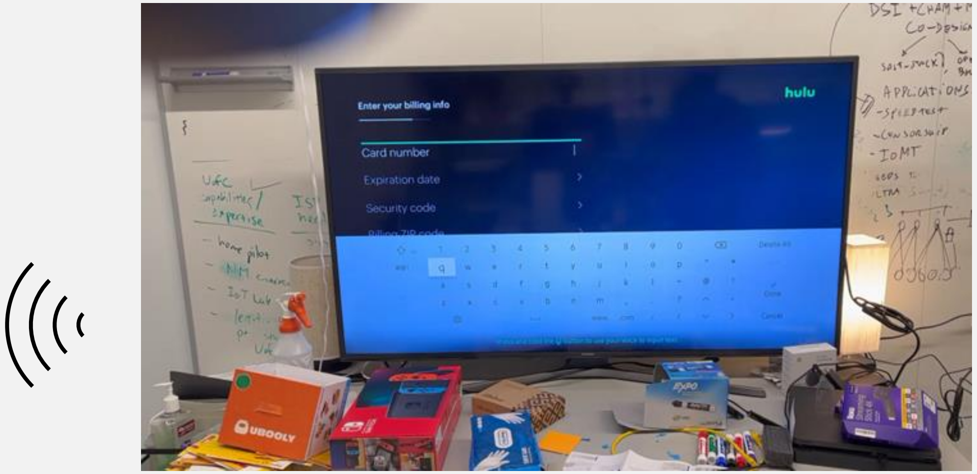
Does the audio from Smart TVs leak information about what a user types on the virtual keyboard?

# Threat Model

We consider an attack who can listen to a Smart TV, either by hijacking a device in the room or placing a malicious recorder

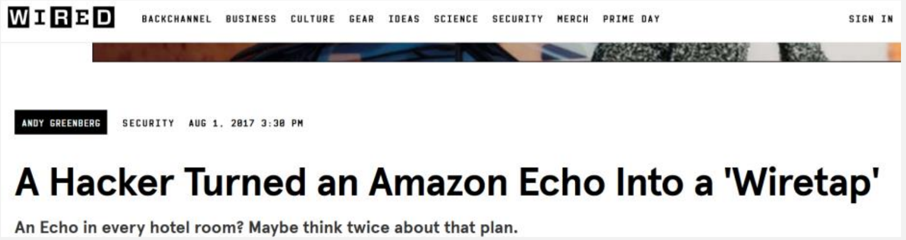
*Attacker listens to TV*

*User types into Smart TV*



# Threat Model

We consider an attack who can listen to a Smart TV, either by hijacking a device in the room or placing a malicious recorder


WIRED BACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY MERCH PRIME DAY SIGN IN

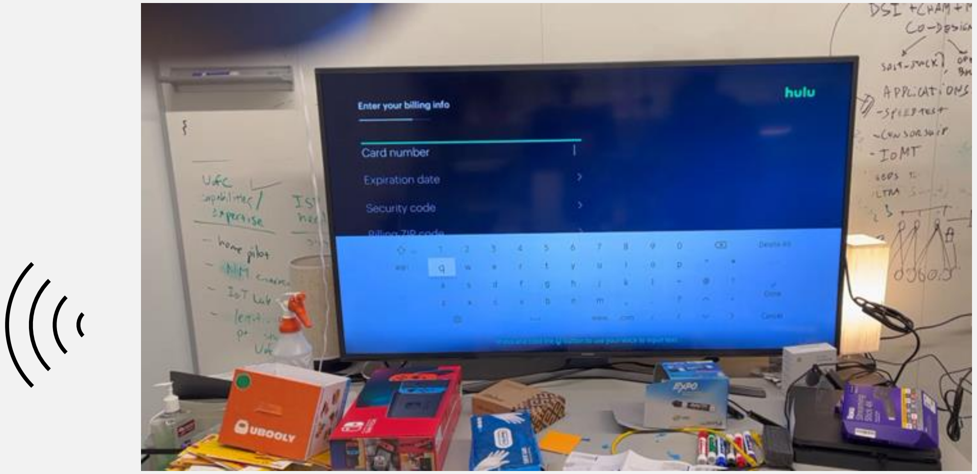
ANDY GREENBERG SECURITY AUG 1, 2017 3:30 PM

A Hacker Turned an Amazon Echo Into a 'Wiretap'

An Echo in every hotel room? Maybe think twice about that plan.

*User types into Smart TV*

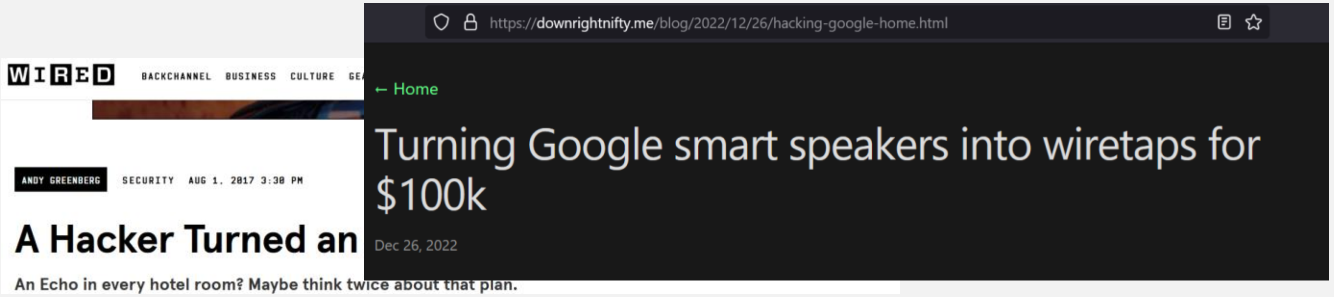
*Attacker listens to TV*




Amazon Echo: https://www.wired.com/story/amazon-echo-wiretap-hack/
Google Home: https://downrightnifty.me/blog/2022/12/26/hacking-google-home.html

# Threat Model

We consider an attack who can listen to a Smart TV, either by hijacking a device in the room or placing a malicious recorder


WIRED BACKCHANNEL BUSINESS CULTURE

ANDY GREENBERG SECURITY AUG 1, 2017 3:30 PM

A Hacker Turned an

An Echo in every hotel room? Maybe think twice about that plan.

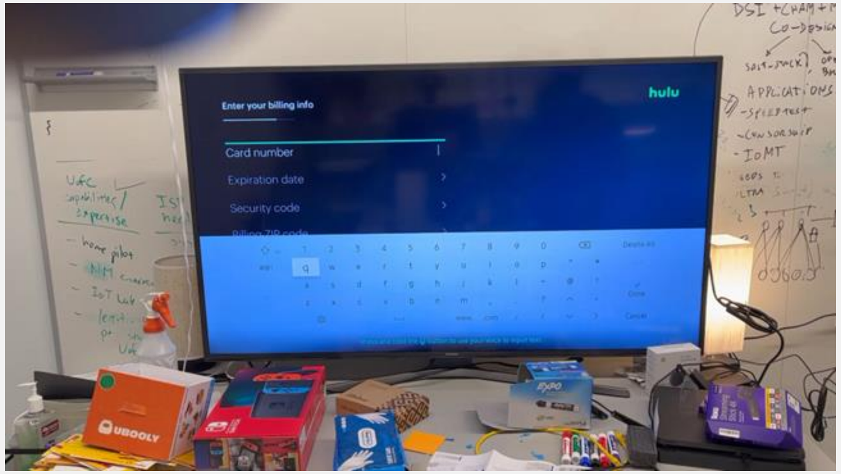
https://downrightnifty.me/blog/2022/12/26/hacking-google-home.html

← Home

Turning Google smart speakers into wiretaps for $100k

Dec 26, 2022

*User types into Smart TV*

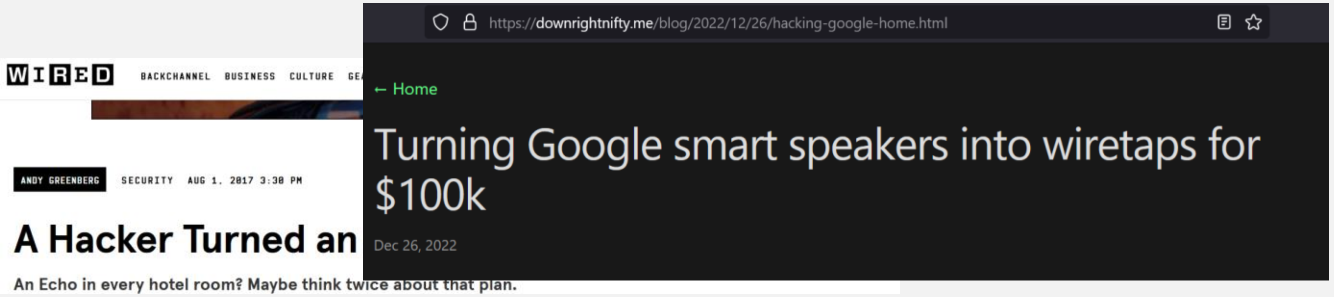



*Attacker listens to TV*

Amazon Echo: https://www.wired.com/story/amazon-echo-wiretap-hack/
Google Home: https://downrightnifty.me/blog/2022/12/26/hacking-google-home.html

# Threat Model

We consider an attack who can listen to a Smart TV, either by hijacking a device in the room or placing a malicious recorder


WIRED BACKCHANNEL BUSINESS CULTURE

ANDY GREENBERG SECURITY AUG 1, 2017 3:30 PM

A Hacker Turned an

An Echo in every hotel room? Maybe think twice about that plan.

https://downrightnifty.me/blog/2022/12/26/hacking-google-home.html

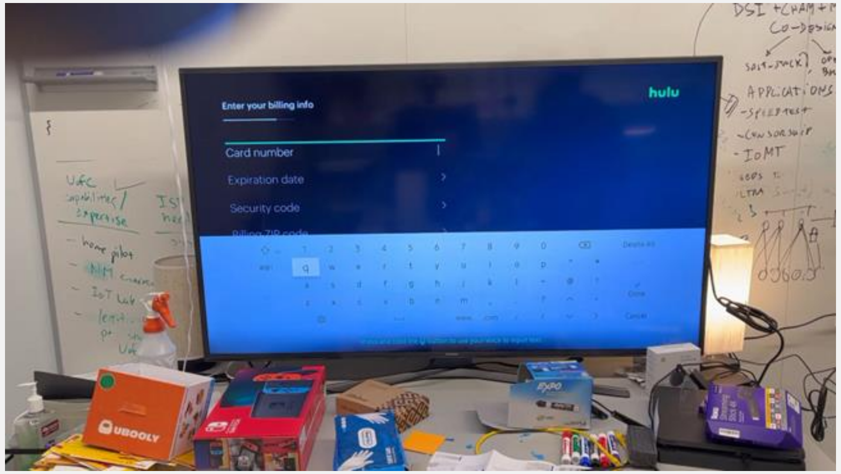
← Home

Turning Google smart speakers into wiretaps for $100k

Dec 26, 2022

*User types into Smart TV*

*Attacker listens to TV*




Use audio information to discover what the user types

Amazon Echo: https://www.wired.com/story/amazon-echo-wiretap-hack/
Google Home: https://downrightnifty.me/blog/2022/12/26/hacking-google-home.html

# Audio and Keyboard Properties

Popular Smart TV brands (AppleTV, Samsung) display the following properties in their default system keyboards:

| | Property | | Implications |
|---|---|---|---|
| 1 | Make different sounds for scrolling, selection, and deletion | → | Can distinguish between actions |

# Audio and Keyboard Properties

Popular Smart TV brands (AppleTV, Samsung) display the following properties in their default system keyboards:

| | **Property** | | **Implications** |
|---|---|---|---|
| 1 | Make different sounds for scrolling, selection, and deletion | → | Can distinguish between actions |
| 2 | The key selection sound is unique to the keyboard | → | Can tell *when* a user is typing |

# Audio and Keyboard Properties

Popular Smart TV brands (AppleTV, Samsung) display the following properties in their default system keyboards:

| | **Property** | | **Implications** |
|---|---|---|---|
| 1 | Make different sounds for scrolling, selection, and deletion | → | Can distinguish between actions |
| 2 | The key selection sound is unique to the keyboard | → | Can tell *when* a user is typing |
| 3 | Keyboards are known and always start the cursor at the same key | → | Can track the cursor from a known starting point on a known layout |

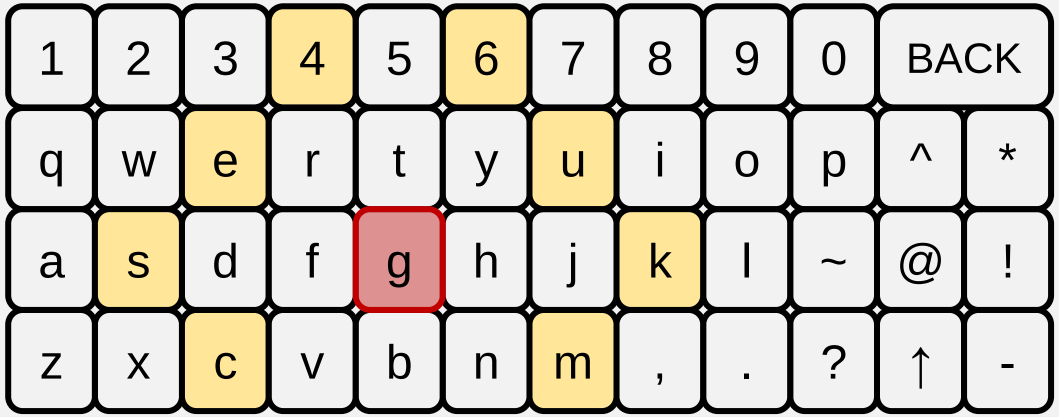
# Challenges of Acoustic Leakage

Using audio alone, the attack must handle a few key challenges:

**1**

Audio does *not* provide the direction of user movements on the keyboard

*Hear 3 movements starting from ‘g’*



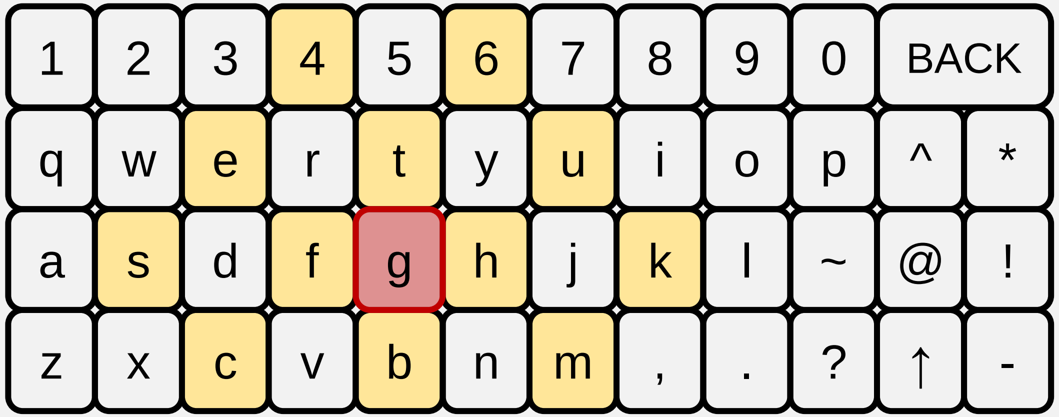
# Challenges of Acoustic Leakage

Using audio alone, the attack must handle a few key challenges:

**1** Audio does *not* provide the direction of user movements on the keyboard

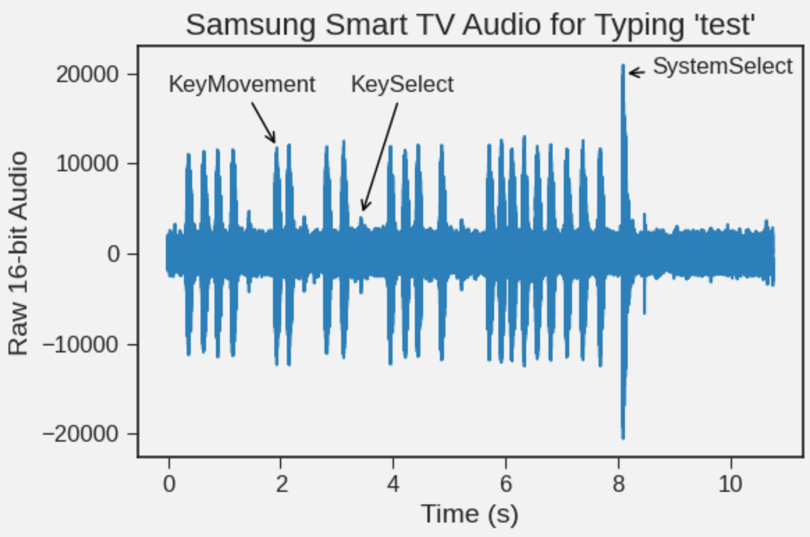
**2** Users can traverse non-shortest (suboptimal) paths between keys

*Hear 3 movements starting from ‘g’*



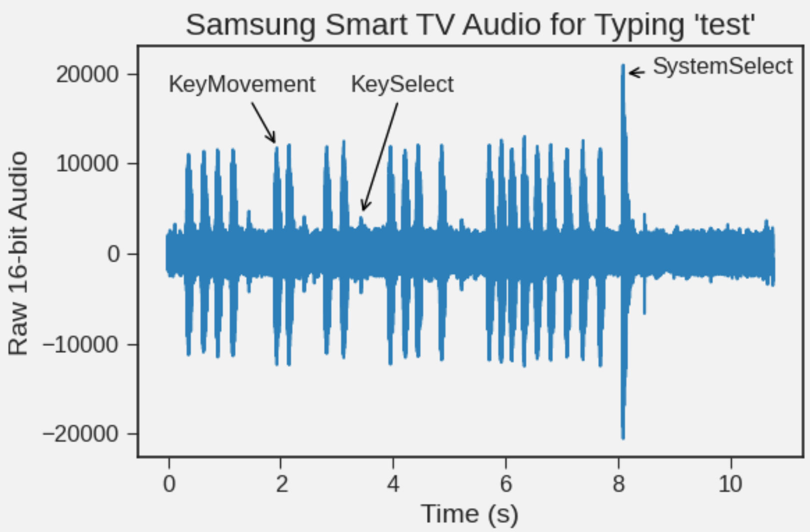
# Attack Overview

Our attack uses two phases, audio extraction and string recovery, to discover user keystrokes from audio signals



# Attack Overview

Our attack uses two phases, audio extraction and string recovery, to discover user keystrokes from audio signals



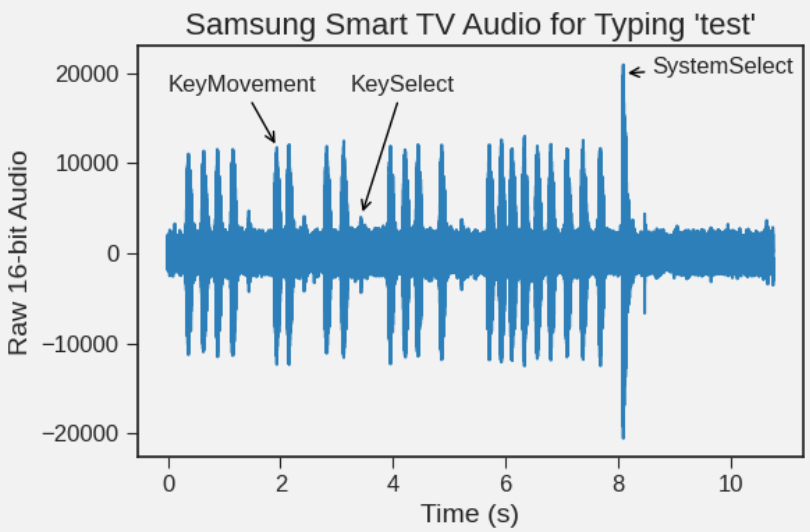

*Move Count Sequence*

[4, 2, 2, 4, 9]

**Audio Extraction**

# Attack Overview

Our attack uses two phases, audio extraction and string recovery, to discover user keystrokes from audio signals




**Audio Extraction**

*Move Count Sequence*

[4, 2, 2, 4, 9]

**String Recovery**

*List of Possible Strings*

test

ft6f

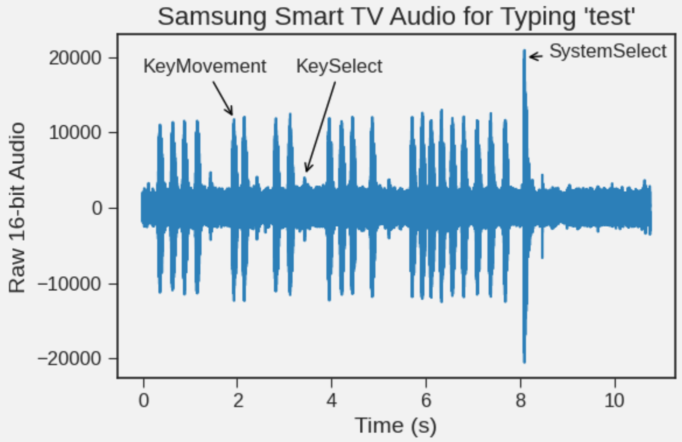
tukt

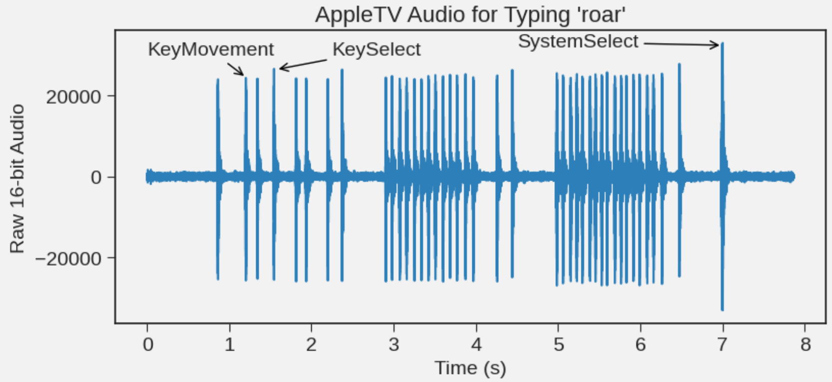
…

*Determine likeliest string using a prior distribution*

# Audio Extraction

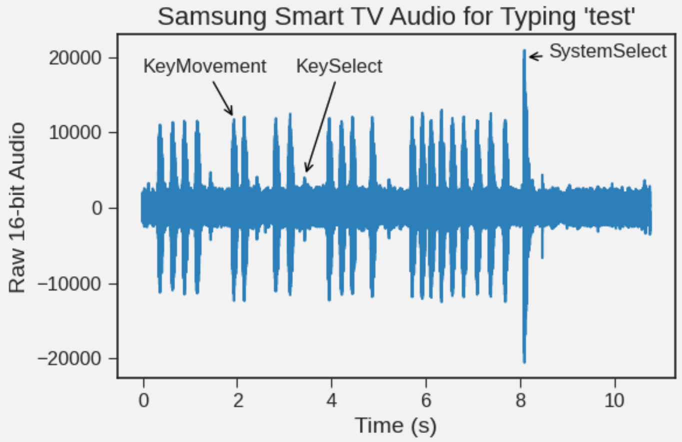
Smart TVs make distinct, consistent sounds for different user actions

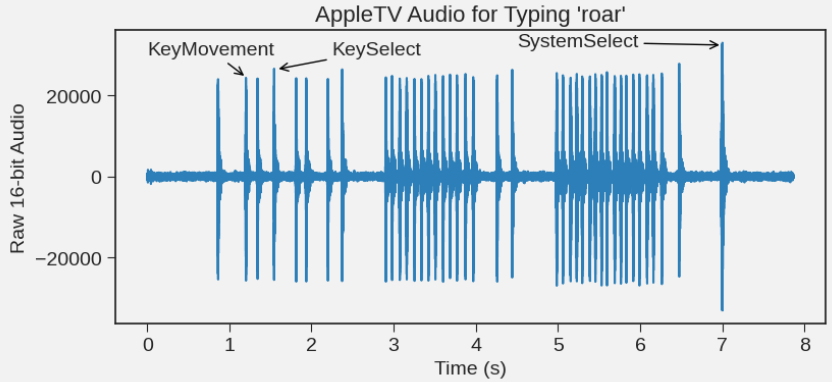


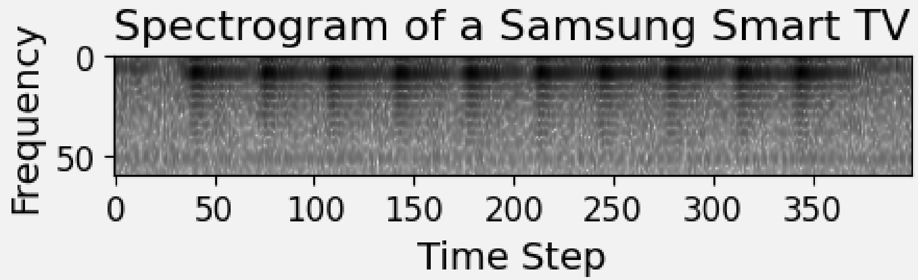
# Audio Extraction

Smart TVs make distinct, consistent sounds for different user actions




Identify specific sounds using nearest-neighbor matching vs pre-recorded versions




*Move Count Sequence*

**Output:** [4, 2, 2, 4, 9]

# String Recovery

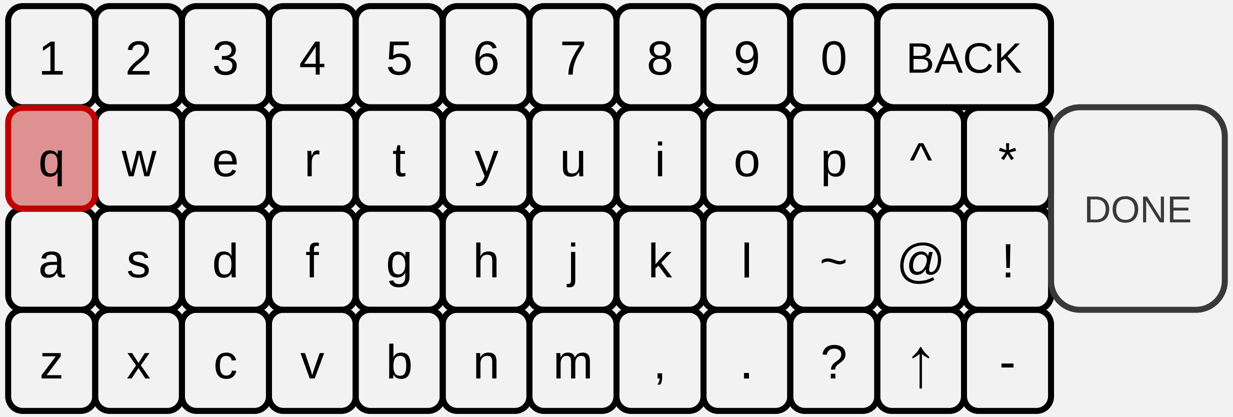
We use the move count sequence to perform a variant of Dijkstra’s algorithm with priorities assigned using a string prior

**Move Count Sequence:** [4, 2, 2, 4, 9]

**Cursor:** q (known start)

**Priority Queue:**

<Empty>



# String Recovery

We use the move count sequence to perform a variant of Dijkstra’s algorithm with priorities assigned using a string prior

**Move Count Sequence:** [4, 2, 2, 4, 9]

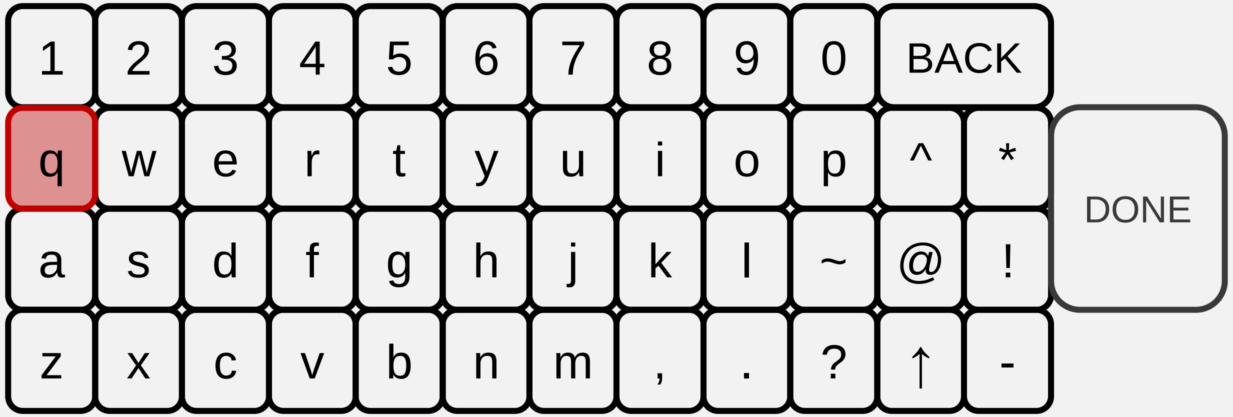
**Cursor:** q (known start)

**Priority Queue:**

<Empty>

In this example:

1. English word prior
2. Only shortest paths



# String Recovery

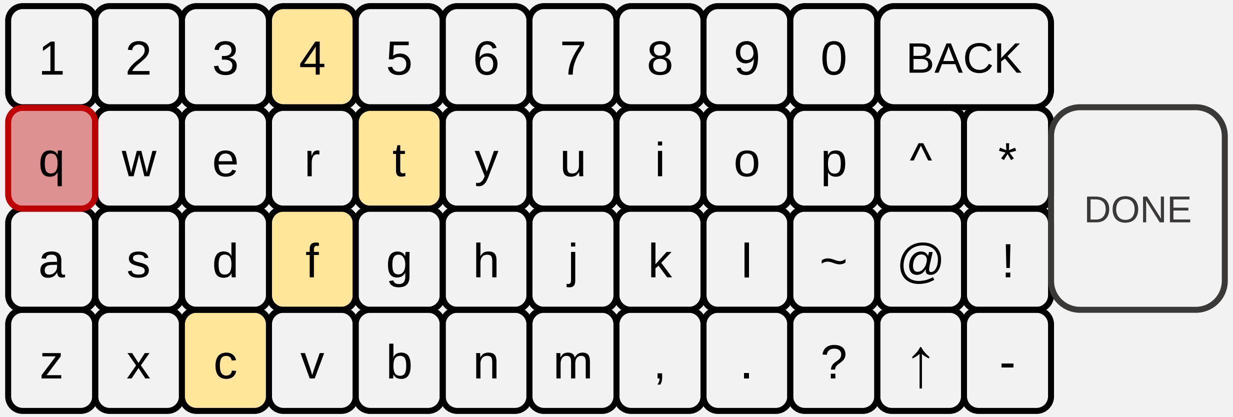
We use the move count sequence to perform a variant of Dijkstra’s algorithm with priorities assigned using a string prior

**Move Count Sequence:** [4, 2, 2, 4, 9]

**Cursor:** q (known start)

**Priority Queue:**

<Empty>



# String Recovery

We use the move count sequence to perform a variant of Dijkstra’s algorithm with priorities assigned using a string prior

**Move Count Sequence:** [4, 2, 2, 4, 9]

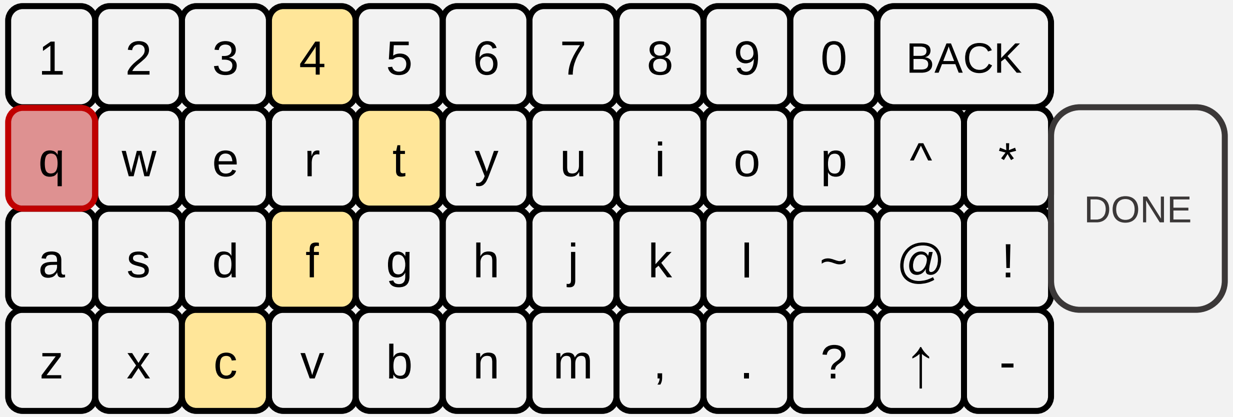
**Cursor:** q (known start)

**Priority Queue:**

(t, 0.0251, ‘’), (c, 0.0094, ‘’), (f, 0.0078), ~~(4, 0.0, ‘’)~~

*Cursor Pos.* *Curr. String*

*Score in Prior*



# String Recovery

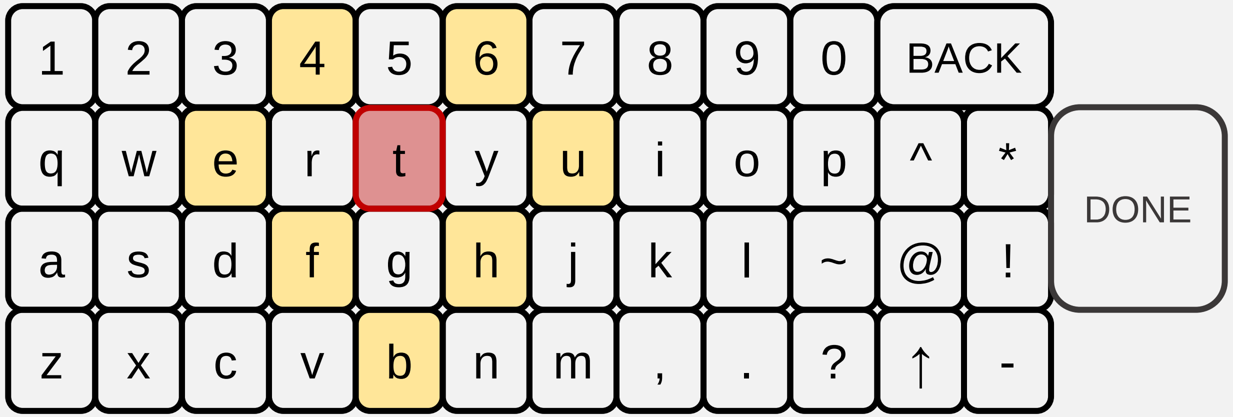
We use the move count sequence to perform a variant of Dijkstra’s algorithm with priorities assigned using a string prior

**Move Count Sequence:** [4, 2, 2, 4, 9]

**Cursor:** t

**Priority Queue:**

(c, 0.0094, ‘’), (f, 0.0078, ‘’), ~~(4, 0.0, ‘’)~~



# String Recovery

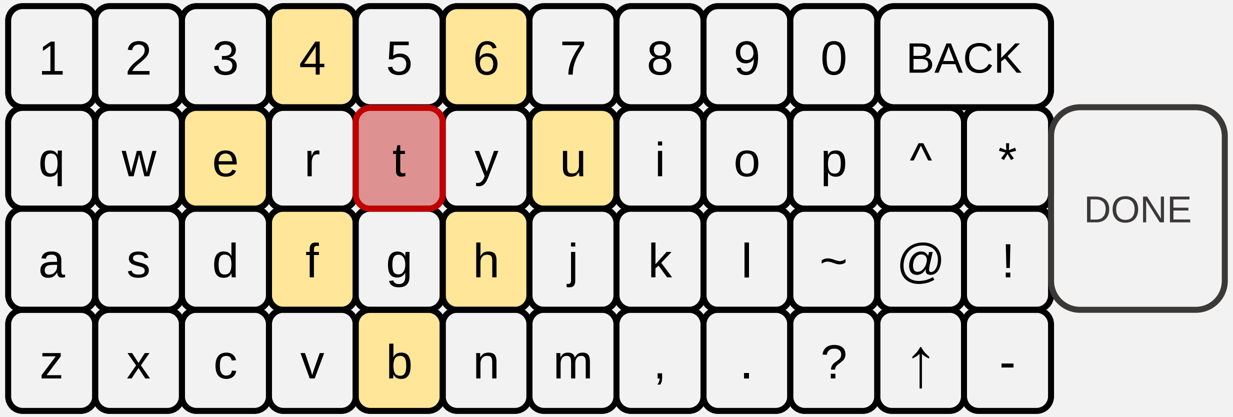
We use the move count sequence to perform a variant of Dijkstra’s algorithm with priorities assigned using a string prior

**Move Count Sequence:** [4, 2, 2, 4, 9]

**Cursor:** t

**Priority Queue:**

(h, 0.0171, ‘t’), (c, 0.0094, ‘’), (f, 0.0078, ‘’), (e, 0.0009, ‘t’), (f, 0.0001, ‘t’), (b, 0.0001, ‘t’), ~~(4, 0.0, ‘’), (4, 0.0, ‘t’), (6, 0.0, ‘t’)~~



# String Recovery

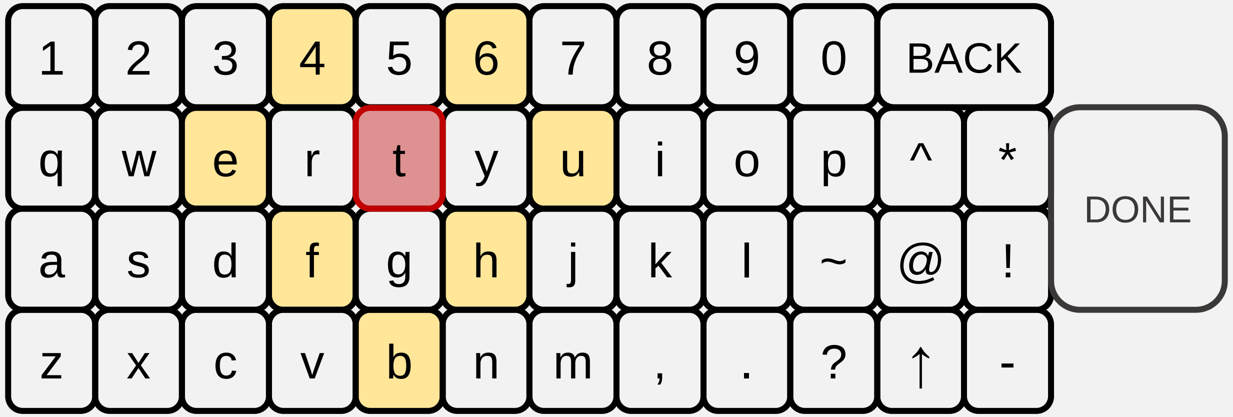
We use the move count sequence to perform a variant of Dijkstra’s algorithm with priorities assigned using a string prior

**Move Count Sequence:** [4, 2, 2, 4, 9]

**Cursor:** t

**Priority Queue:**

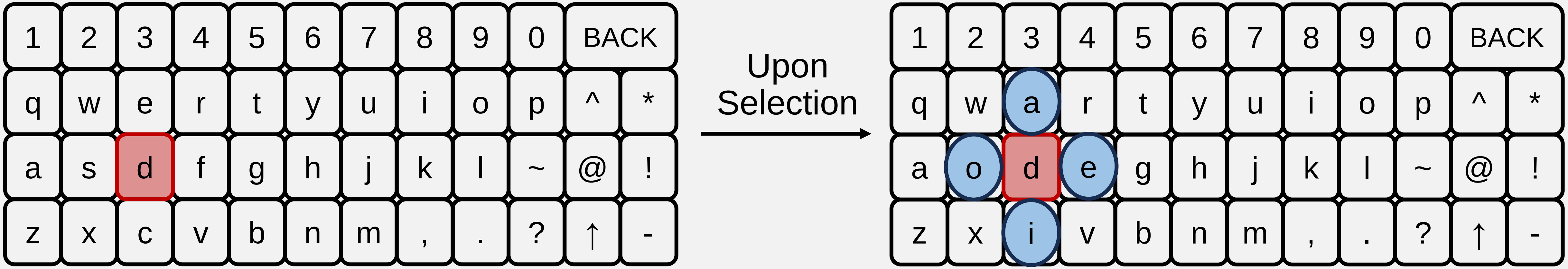
(h, 0.0171, ‘t’), (c, 0.0094, ‘’), (f, 0.0078, ‘’), (e, 0.0009, ‘t’), (f, 0.0001, ‘t’), (b, 0.0001, ‘t’), ~~(4, 0.0, ‘’), (4, 0.0, ‘t’), (6, 0.0, ‘t’)~~




Guess strings upon exhausting all entries in the move count sequence

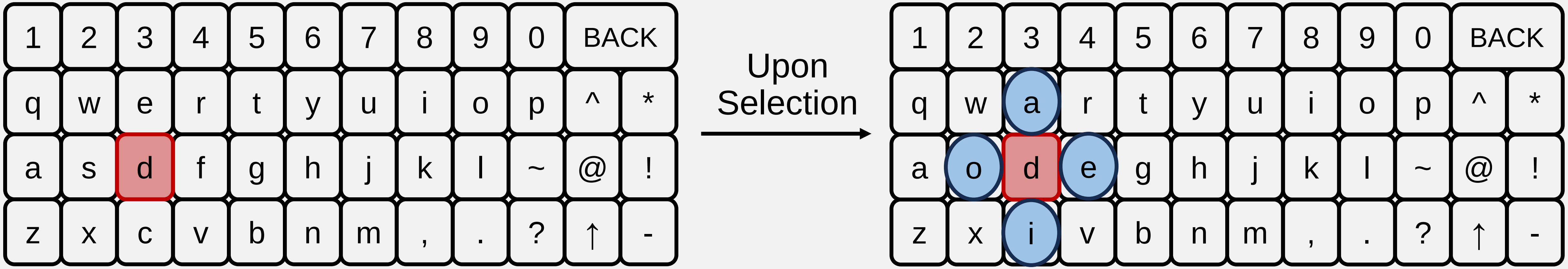
# Dynamic Keyboards

On Samsung Smart TVs, the keyboard makes inline suggestions when users type predictable inputs



# Dynamic Keyboards

On Samsung Smart TVs, the keyboard makes inline suggestions when users type predictable inputs

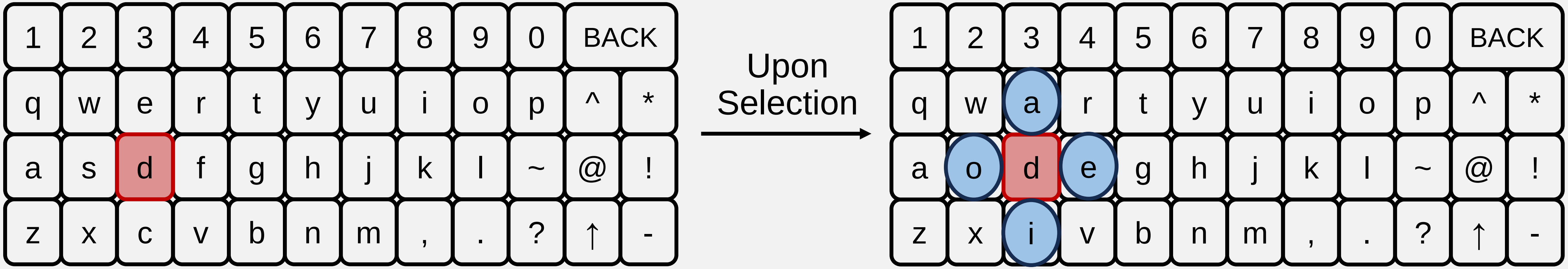



Typing ‘test’ without suggestions: [4, 2, 2, 4, 9]

Typing ‘test’ with suggestions: [4, 1, 0, 1, 1]

# Dynamic Keyboards

On Samsung Smart TVs, the keyboard makes inline suggestions when users type predictable inputs




Typing ‘test’ without suggestions: [4, 2, 2, 4, 9]

Typing ‘test’ with suggestions: [4, 1, 0, 1, 1]

Unpredictable inputs, such as **passwords** and **credit cards**, do *not* use suggestions

# String Priors

The best string prior depends on the type of entered information

| **Information Type** | | **String Prior** |
|---|---|---|
| Passwords | → | Common passwords from leaked lists (e.g., RockYou) |

# String Priors

The best string prior depends on the type of entered information

| **Information Type** | | **String Prior** |
|---|---|---|
| Passwords | → | Common passwords from leaked lists (e.g., RockYou) |
| English words | → | Words from the Wikipedia corpus |

# String Priors

The best string prior depends on the type of entered information

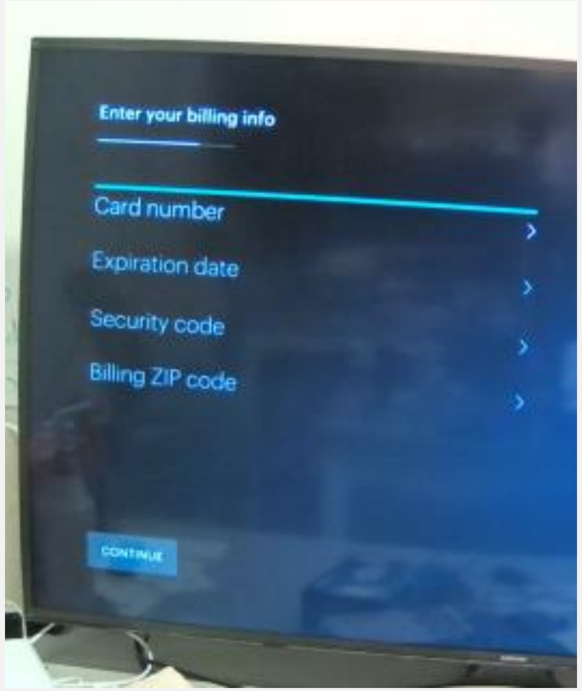
| **Information Type** | | **String Prior** |
|---|---|---|
| Passwords | → | Common passwords from leaked lists (e.g., RockYou) |
| English words | → | Words from the Wikipedia corpus |
| Credit Card Numbers | → | Digits, satisfy Luhn's algorithm |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *CCN* | 3 | 4 | 0 | 6 | 5 | 3 | 6 | 8 | 1 | 9 | 3 | 7 | 2 | 7 | 7 |
| *Multipliers* | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 |
| *Digit Sum* | 3 | 8 | 0 | 3 | 5 | 6 | 6 | 7 | 1 | 9 | 3 | 5 | 2 | 5 | 7 |

*Sum: 70 % 10 = 0* ✓

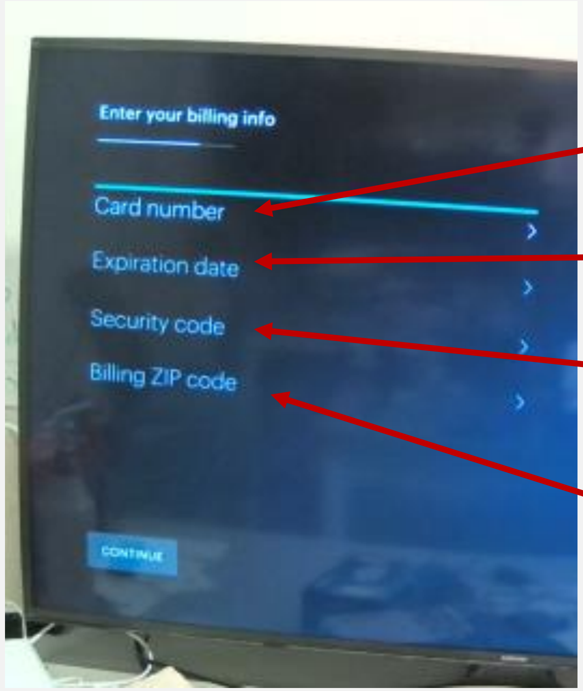
# Inferring the Information Type

The attacker can use string properties and typing dynamics to infer the information type and customize the string prior



# Inferring the Information Type

The attacker can use string properties and typing dynamics to infer the information type and customize the string prior




*15/16 Digits*

*MM/YY*

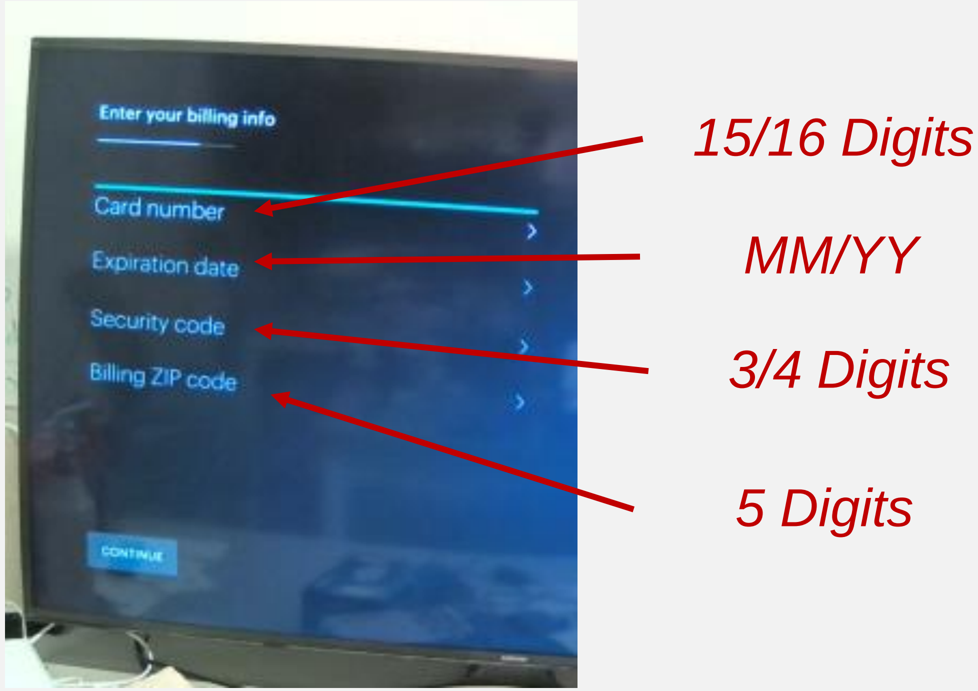
*3/4 Digits*

*5 Digits*

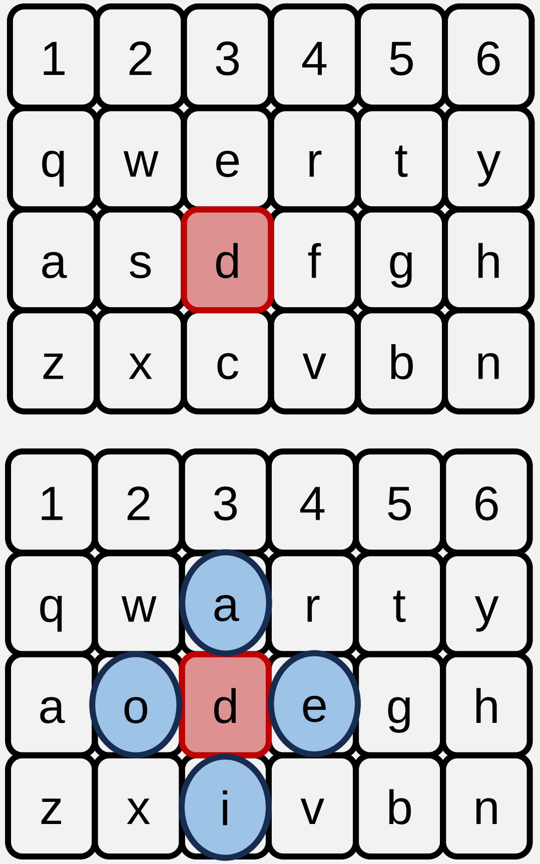
Use lengths of successive inputs to detect payment details

# Inferring the Information Type

The attacker can use string properties and typing dynamics to infer the information type and customize the string prior

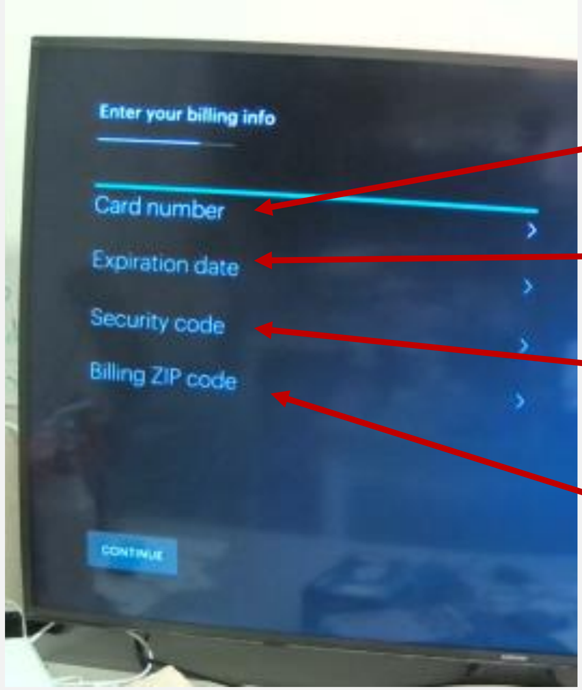



Use lengths of successive inputs to detect payment details



# Inferring the Information Type

The attacker can use string properties and typing dynamics to infer the information type and customize the string prior



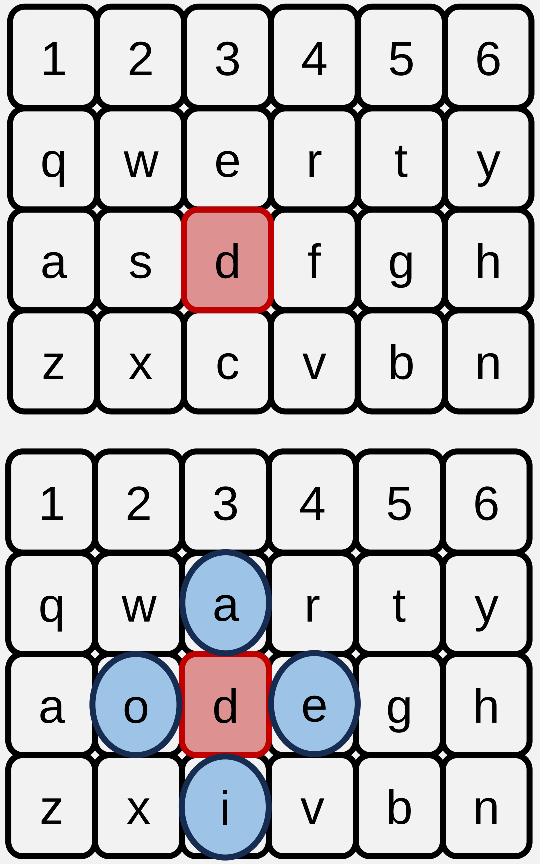

*15/16 Digits*

*MM/YY*

*3/4 Digits*

*5 Digits*

Use lengths of successive inputs to detect payment details




*Move Counts*

[4, 1, 0, 1, 1]

Random Forest

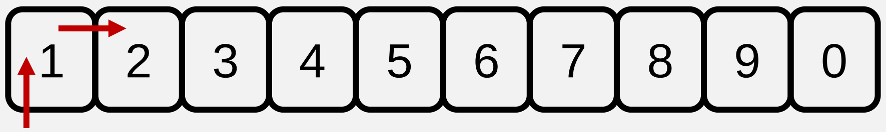
Suggestions OR Password

# Handling Suboptimal Paths

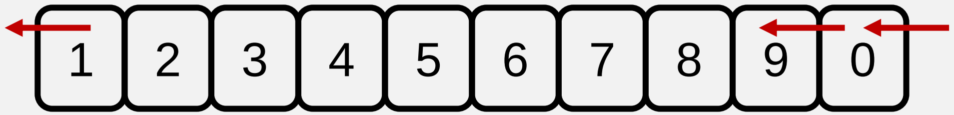
Users tend to pause when correcting suboptimal paths, so the attack detects such pauses to expand the search when necessary

# Handling Suboptimal Paths

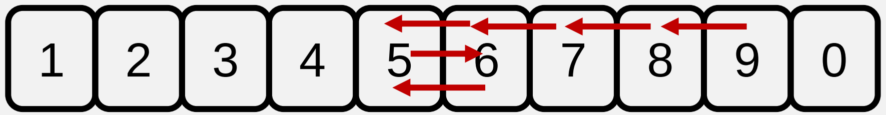
Users tend to pause when correcting suboptimal paths, so the attack detects such pauses to expand the search when necessary



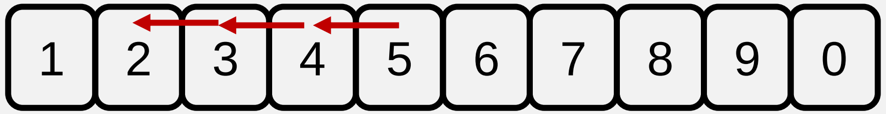

*Gaps:* 0.326s, 0.291s




*Gaps:* 0.187s, 0.168s, 0.434s, 0.169s, 0.311s, 0.249s



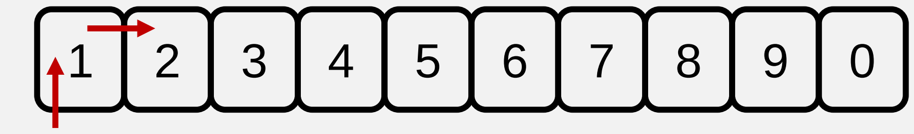

*Gaps:* 0.167s, 0.188s, 0.167s, 0.484s, 0.498s, 0.880s



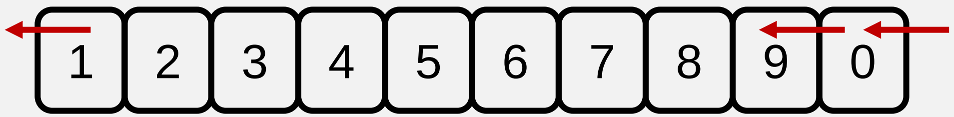

*Gaps:* 0.299s,  0.451s, 0.520s

# Handling Suboptimal Paths

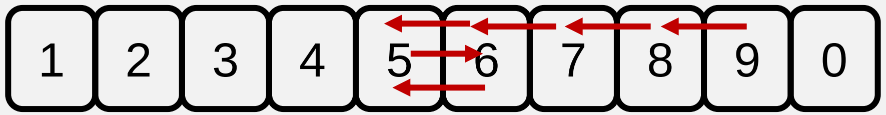
Users tend to pause when correcting suboptimal paths, so the attack detects such pauses to expand the search when necessary



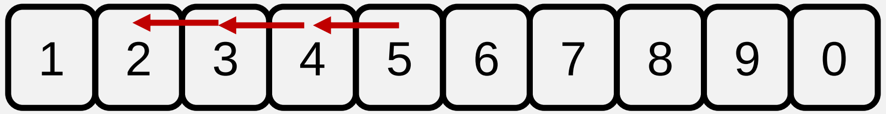

*Gaps:* 0.326s, 0.291s




*Gaps:* 0.187s, 0.168s, 0.434s, 0.169s, 0.311s, 0.249s



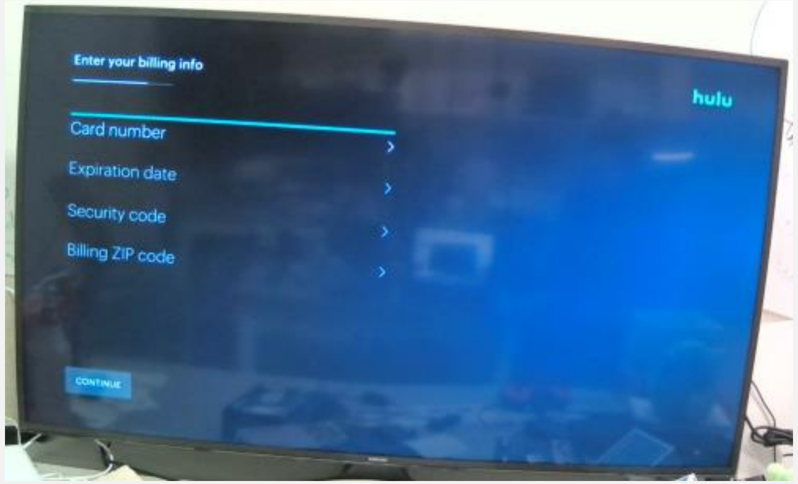

*Gaps:* 0.167s, 0.188s, 0.167s, 0.484s, 0.498s, 0.880s



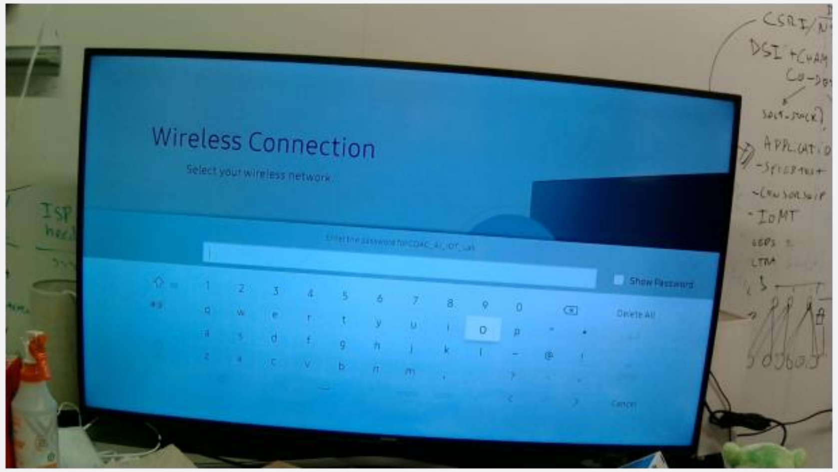

*Gaps:* 0.299s, 0.451s, 0.520s

# Evaluation: User Study

We have 10 users type fake credit card details, common passwords, and web searches into a Samsung TV and an AppleTV




*Payment details into Hulu (Samsung)*



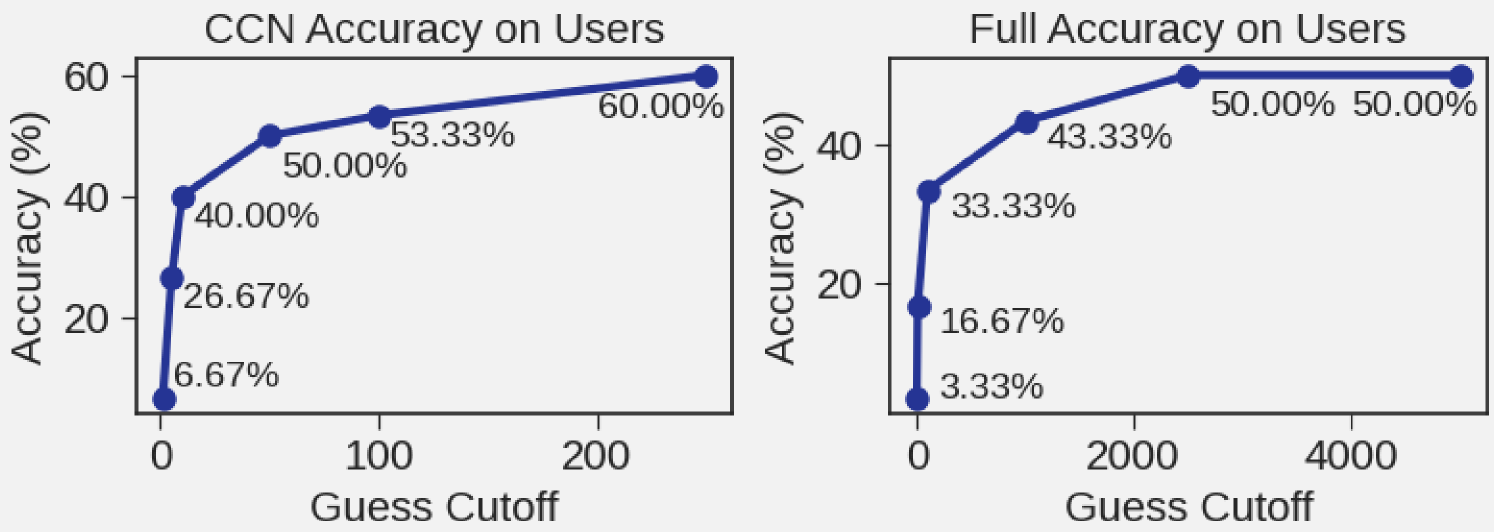

*WiFi Passwords (Samsung)*


*Apple ID Passwords (Apple)*

This study was approved by our university's Institutional Review Board (IRB)

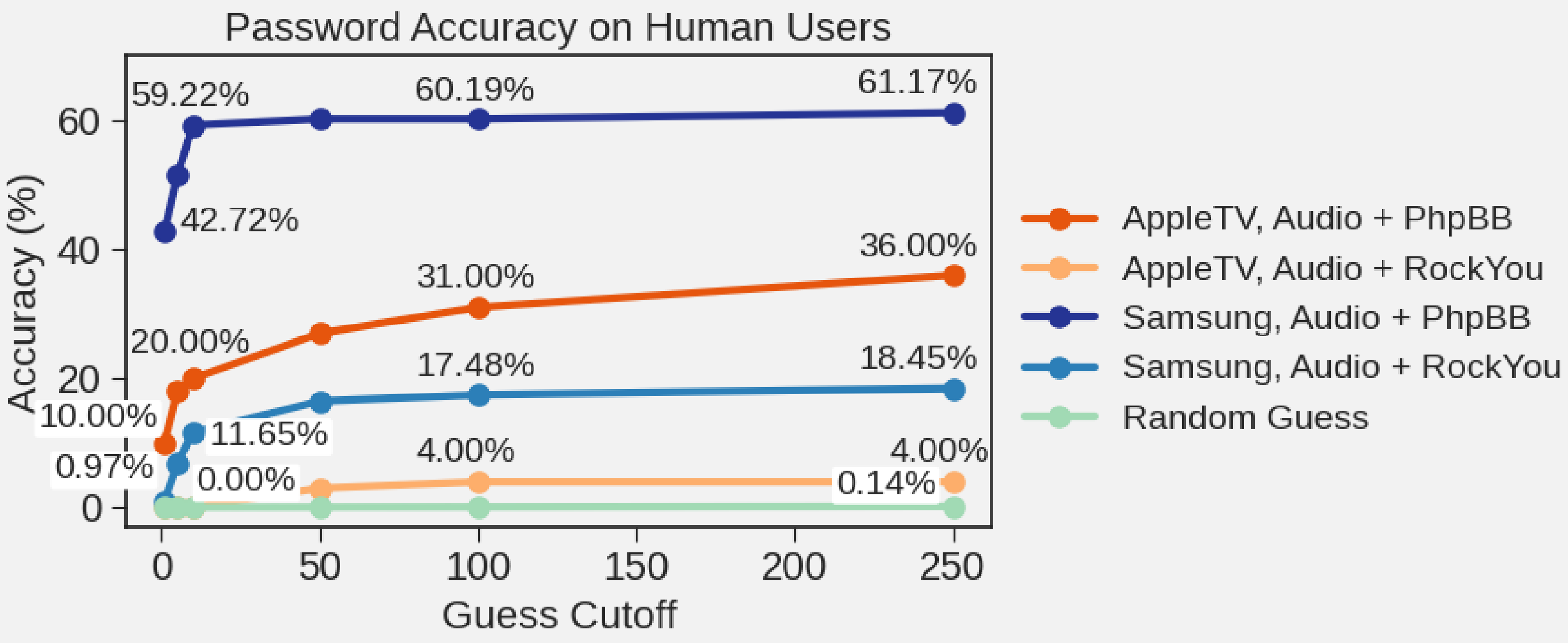
# Evaluation: Credit Cards

The attack successfully discovers 50% of full payment details (credit card number, security code, expiration date, and zip code) within 5,000 guesses



# Evaluation: Passwords

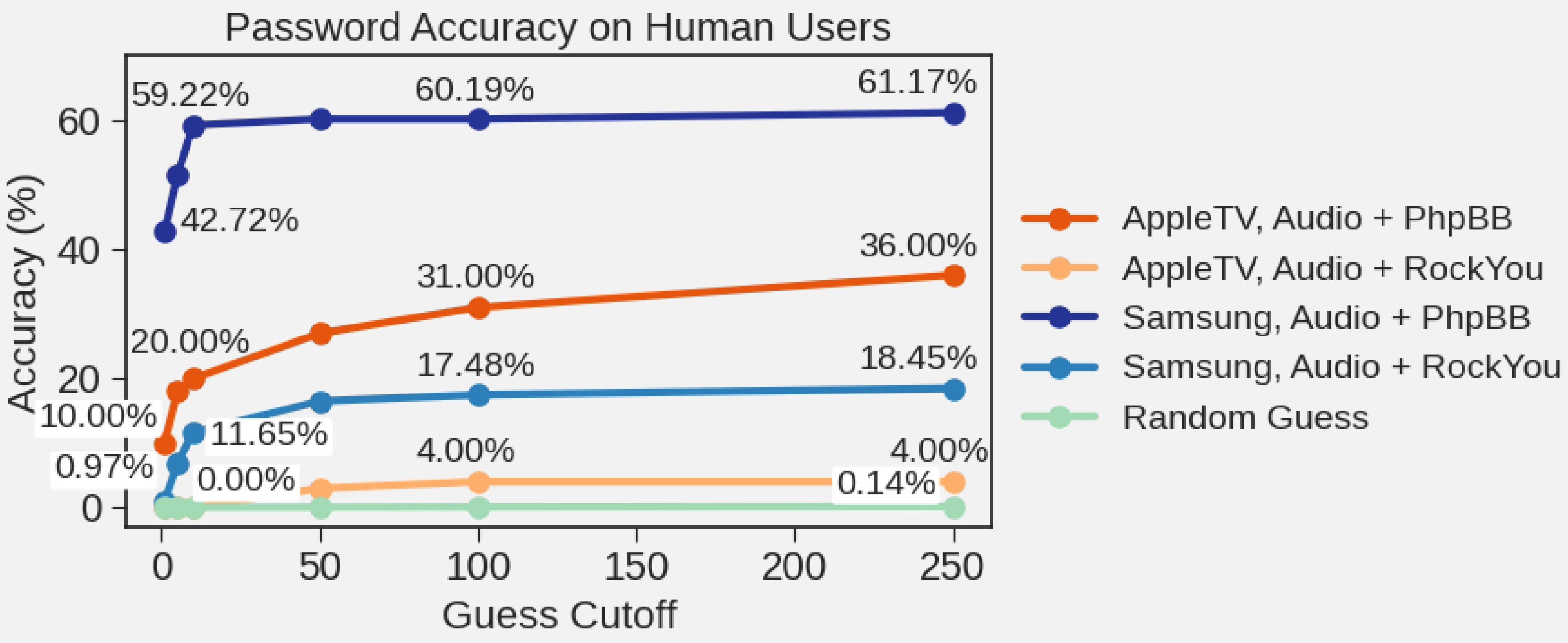
Against passwords drawn from the PhpBB leak, the attack exceeds random guessing by over 100x




Wang, Ding, et al. "Targeted online password guessing: An underestimated threat.“ *Computer and Communications Security (CCS)*. 2016.

# Evaluation: Passwords

Against passwords drawn from the PhpBB leak, the attack exceeds random guessing by over 100x




Attackers can reasonably try 100 passwords against rate-limited services

Wang, Ding, et al. "Targeted online password guessing: An underestimated threat." *Computer and Communications Security (CCS)*. 2016.

# Conclusion

1. Modern Smart TVs make distinct, consistent sounds as users type

2. The acoustic properties leak important user actions during typing

3. We develop a new attack which discovers payment information and common passwords using the audio from Smart TVs

4. As the number of “smart” devices grows, designers must consider the privacy implications of every feature interacting with sensitive data

# Appendix: Responsible Disclosure

We disclosed this vulnerability to both Apple and Samsung. Samsung acknowledged the problem and awarded a bounty

**Samsung Security**
**for Smart TV, Audio and Displays**

Security Post | Security Updates | Bug Bounty Program | Certificates

Home > Hall Of Fame

## Hall Of Fame

### Rewards

The following individuals have qualified for a reward from Samsung Security Bug Bounty for Smart TV, Audio and Displays. On behalf of our millions of Smart TV, Audio and Displays users, we thank you for helping make Samsung products safer.

**Latest 3 reporters**

- Sector 7 from Computest (@sector7_nl)
- Tejas Kannan (https://people.cs.uchicago.edu/~tkannan, Synthia Wang, Max Sunog, Abe Bueno de Mesquita, Nick Feamster, and Hank Hoffmann)
- Mateusz Klement ( https://www.linkedin.com/in/mateusz-klement-7629b6134/, Justyna Graczyk)

# Appendix: Defenses

Users can protect themselves against this issue in a few ways

1. Mute the TV when typing, especially when entering sensitive information

2. Add extraneous movements when typing to add suboptimal paths

3. Use pseudo-random passwords consisting of diverse characters

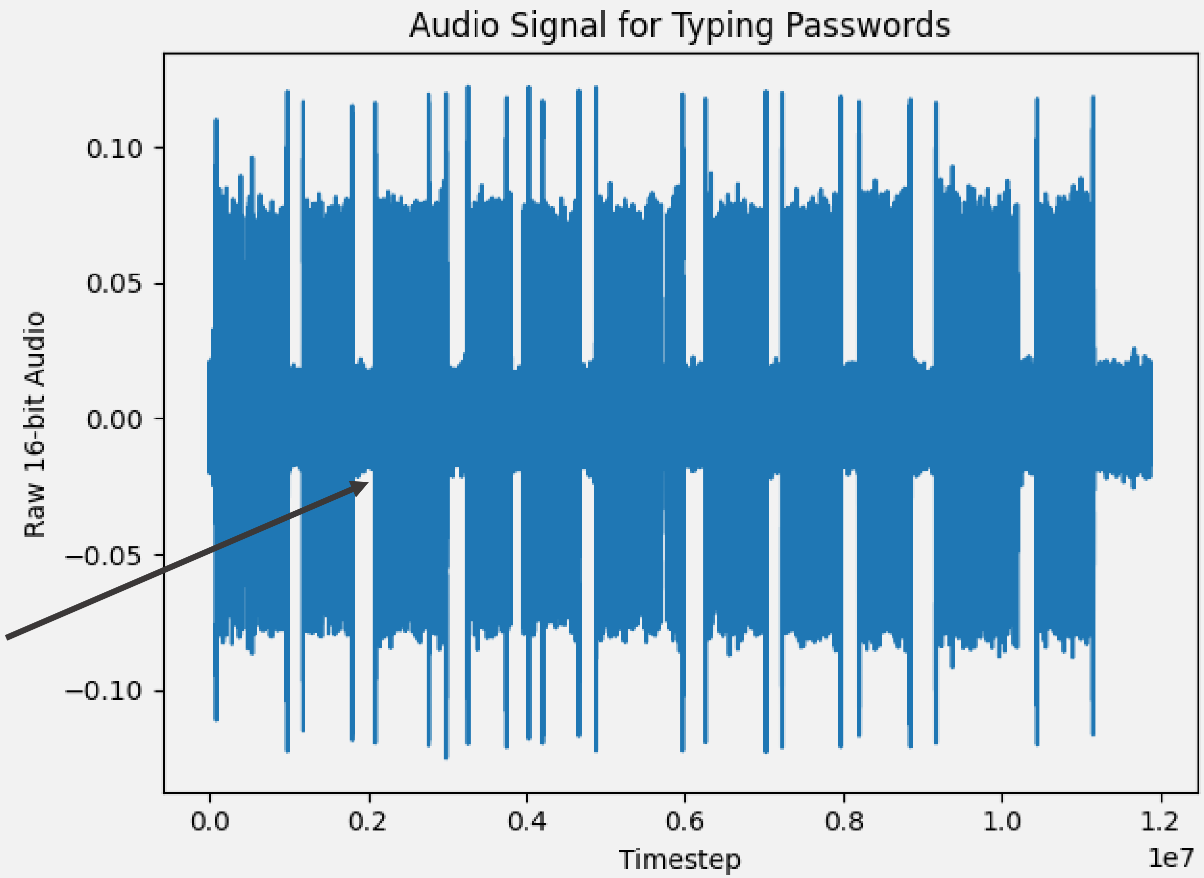
# Appendix: Smart TV User Study

10 total users aged 22 through 29, all either undergraduate or graduate students at the University of Chicago

- All users had used a Smart TV before
- 3 owned a Samsung Smart TV, 1 owned an AppleTV
- Each user enters 10 passwords on each system, 10 web searches, and 3 sets of credit card details
- The strings were provided to the users, so users do *not* type their own sensitive information
- We have users type on a model UN55MU6300 Samsung Smart TV running Tizen OS version T-KTMAKUC-1310.1 and an A1625 AppleTV with tvOS version 16.3.2

# Appendix: Splitting Keyboard Instances

The attack identifies a single use of the keyboard using timing, where long delays signal the end of typing a single string




Time break, separates keyboard uses

# Appendix: User Passwords

We provide all users with passwords to type from the PhpBB list:

- Passwords have at least 8 characters
- Ensure at least 15 passwords have 1+ special characters, numbers, and uppercase letters
- Collect 50 unique passwords, with two users entering each list (of 10) for a total of 100 entries across all users
- On the Samsung TV under the PhpBB prior, 21 / 50 unique passwords were recovered on *both* users

*Example List:* `function84, naarf666, p5ych0#7, chevy_1954, pva81-ph, .sagara., 8b7ce7df, tutphpbb, bubba?51879, williame`

# Appendix: Confidence Intervals

Against user-typed passwords on the Samsung Smart TV, we obtain the following 95% confidence intervals for the mean top-100 accuracy

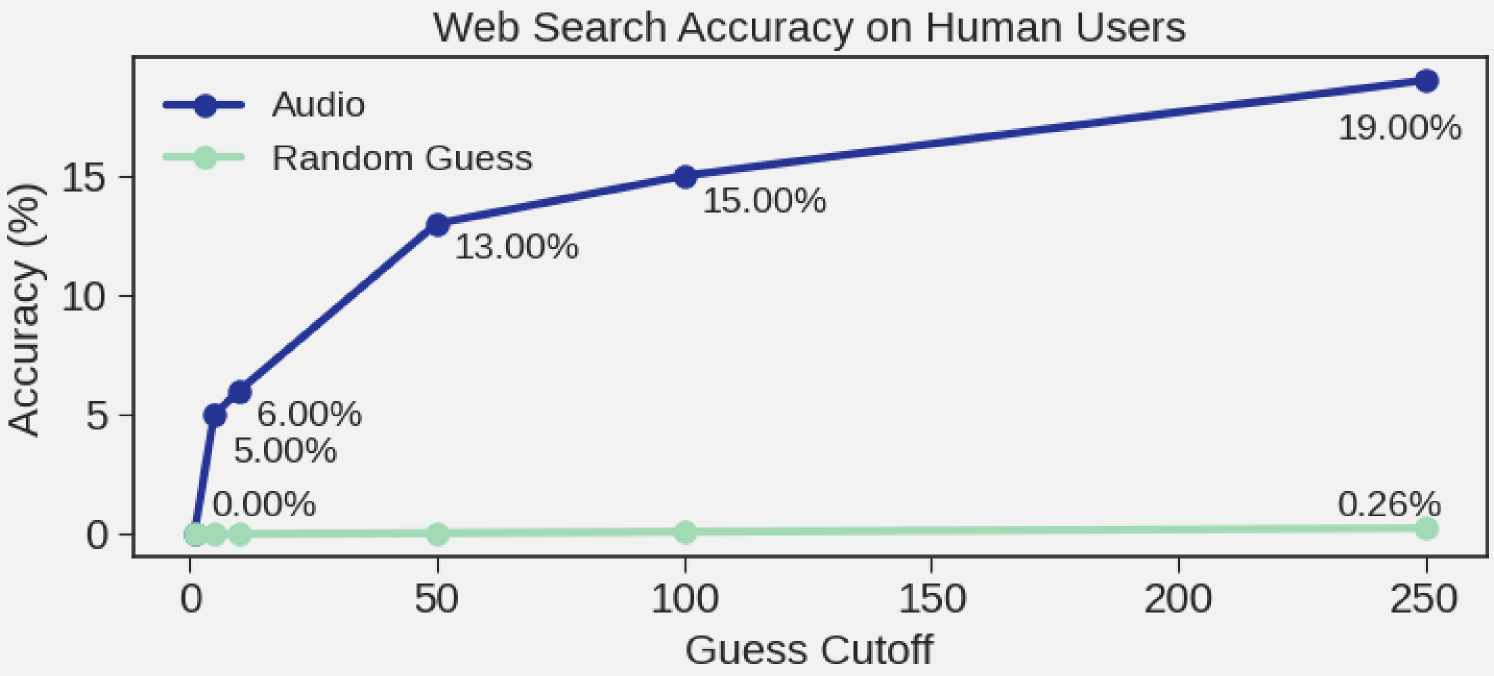
PhpBB Prior: $61.48 \pm 1.97 \times \frac{26.84}{\sqrt{10}} = (44.76\%, 78.21\%)$

RockYou Prior: $17.74 \pm 1.97 \times \frac{14.85}{\sqrt{10}} = (8.49\%, 26.99\%)$

Random Guessing: $\frac{100}{184{,}388} = 0.054\%$ Over 150x increase

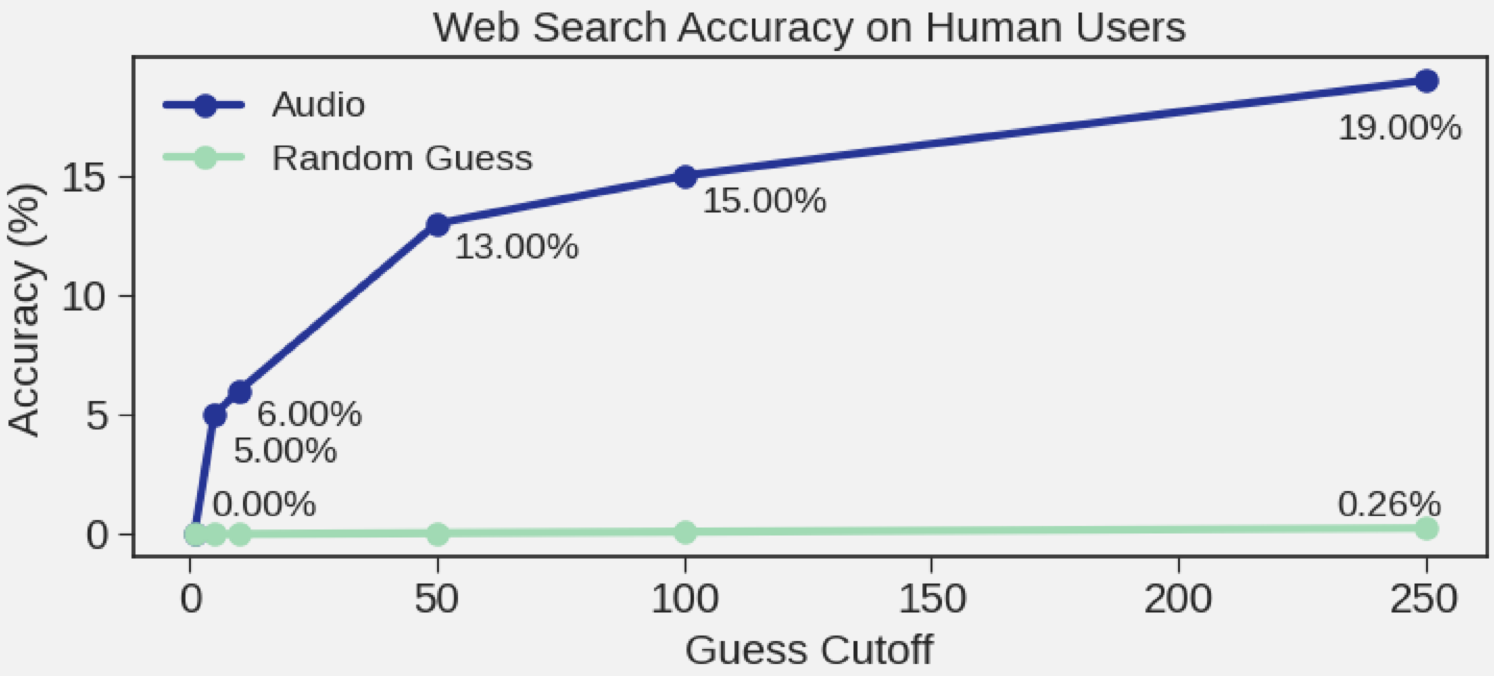
# Appendix: Web Searches

Even on keyboards with dynamic behavior, the attacker can still achieve results far above random guessing



# Appendix: Web Searches

Even on keyboards with dynamic behavior, the attacker can still achieve results far above random guessing




Better performance may be possible if considering relationships between words

# Appendix: Password Characters

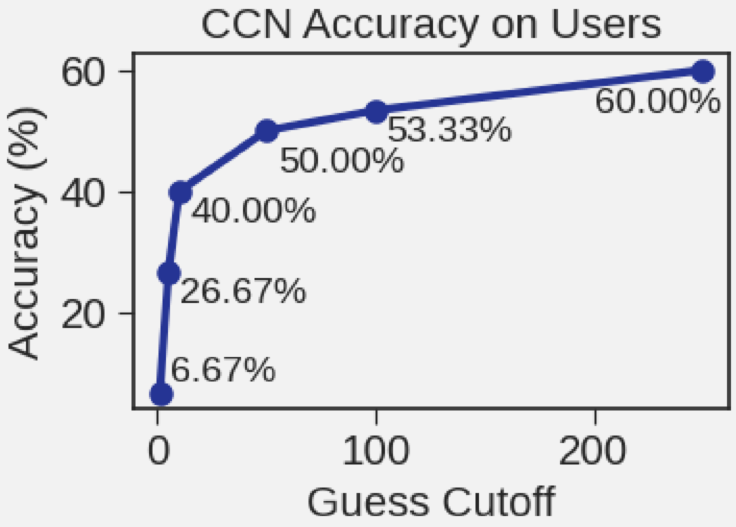
Recovery accuracy under the PhpBB Prior on human users. All chosen passwords have at least 8 characters

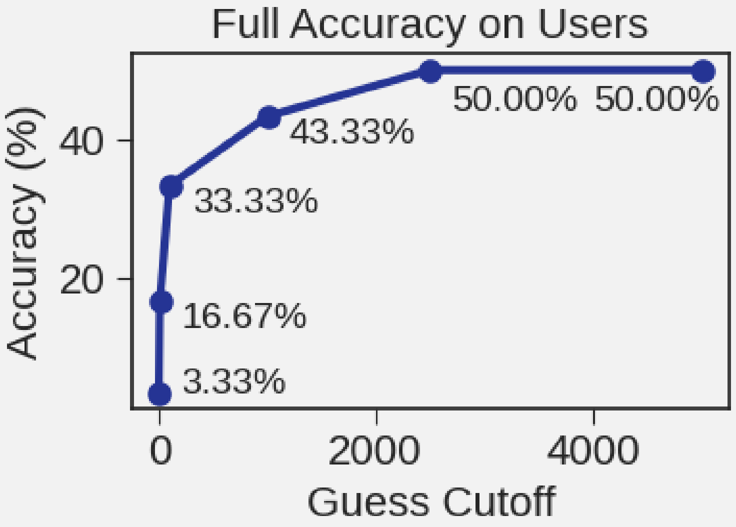
| **AppleTV** | **Guess Cutoff (Top-K)** | | |
|---|---|---|---|
| | K = 1 | K = 10 | K = 100 |
| Number | 13.79% (8 / 58) | 24.14% (14 / 58) | 36.21% (21 / 58) |
| Special | 12.00% (6 / 50) | 22.00% (11 / 50) | 32.00% (16 / 50) |
| Upper | 0.00% (0 / 16) | 6.25% (1 / 16) | 12.50% (2 / 16) |

| **Samsung** | **Guess Cutoff (Top-K)** | | |
|---|---|---|---|
| | K = 1 | K = 10 | K = 100 |
| Number | 48.28% (28 / 58) | 63.79% (37 / 58) | 63.79% (37 / 58) |
| Special | 42.31% (22 / 52) | 59.62% (31 / 52) | 59.62% (31 / 52) |
| Upper | 37.50% (6 / 16) | 43.75% (7 / 16) | 43.75% (7 / 16) |

# Appendix: Credit Cards

The attack successfully discovers 50% of full payment details (credit card number, security code, expiration date, and zip code) within 5,000 guesses

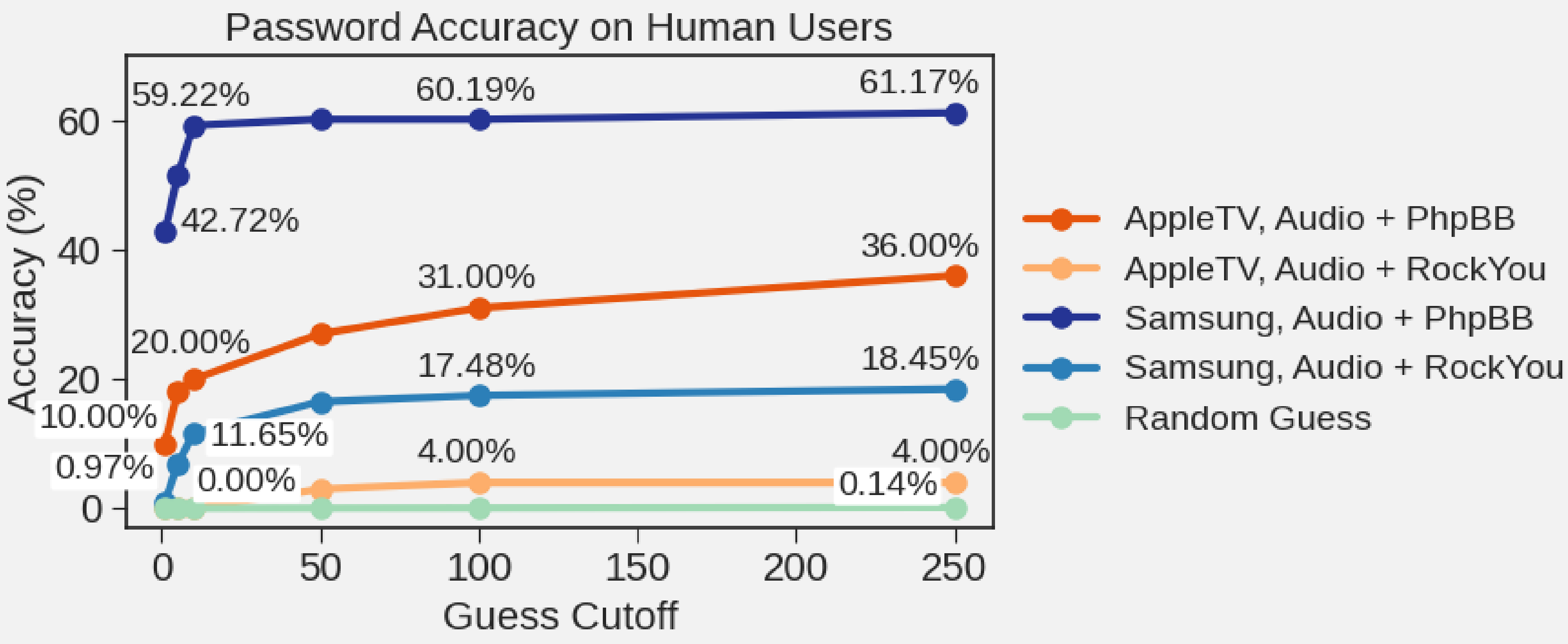



About 15% of strings matching the move count sequence satisfy the credit card number checksum

# Appendix: Passwords

Against passwords drawn from the PhpBB leak, the attack exceeds random guessing by over 100x



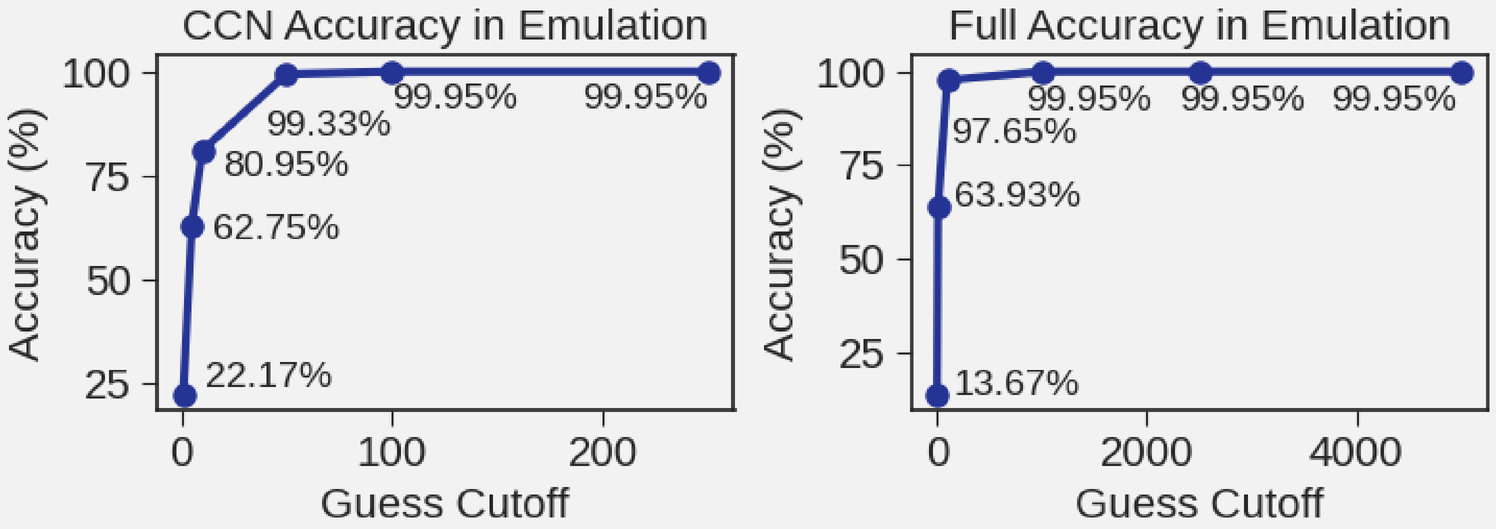

Attackers can reasonably try 100 passwords against rate-limited services

Users traverse optimal paths ~86% (Samsung) vs ~45% (AppleTV) of the time


Wang, Ding, et al. "Targeted online password guessing: An underestimated threat.“ *Computer and Communications Security (CCS)*. 2016.

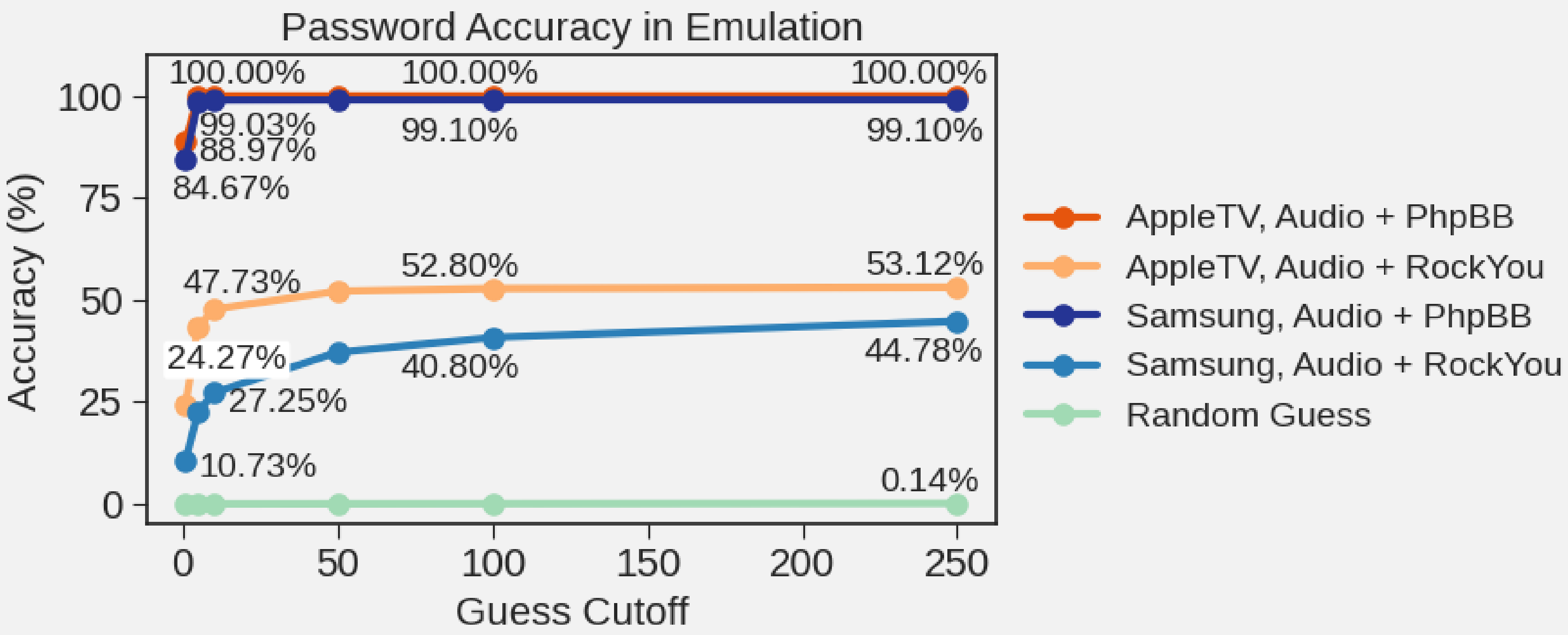
# Appendix: Emulated Credit Cards

Credit card recovery rates in emulation, assumes all optimal paths



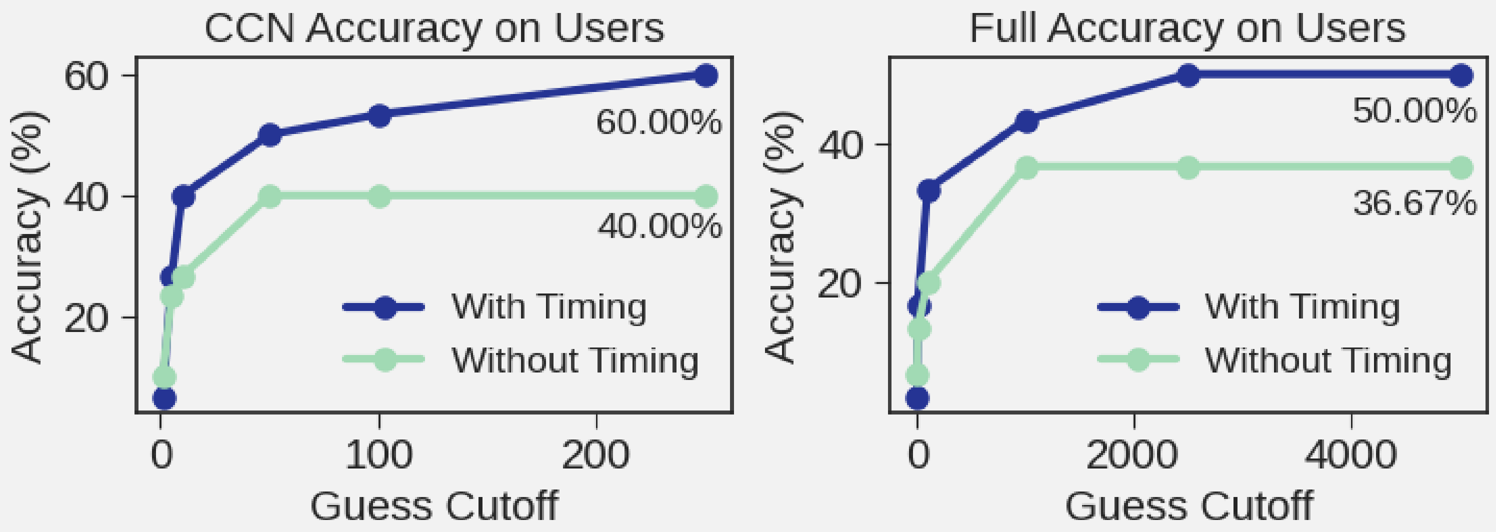
# Appendix: Emulated Passwords

Password recovery rates in emulation, assumes optimal paths

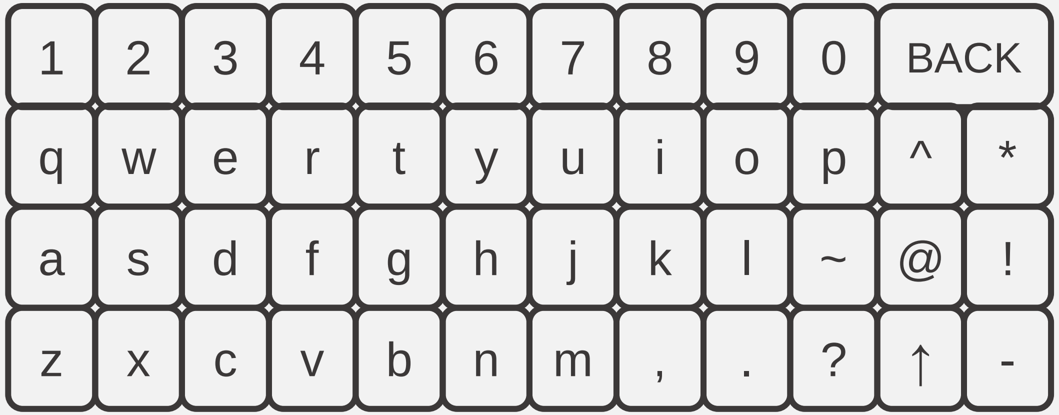


# Appendix: Strategy for Suboptimal Moves

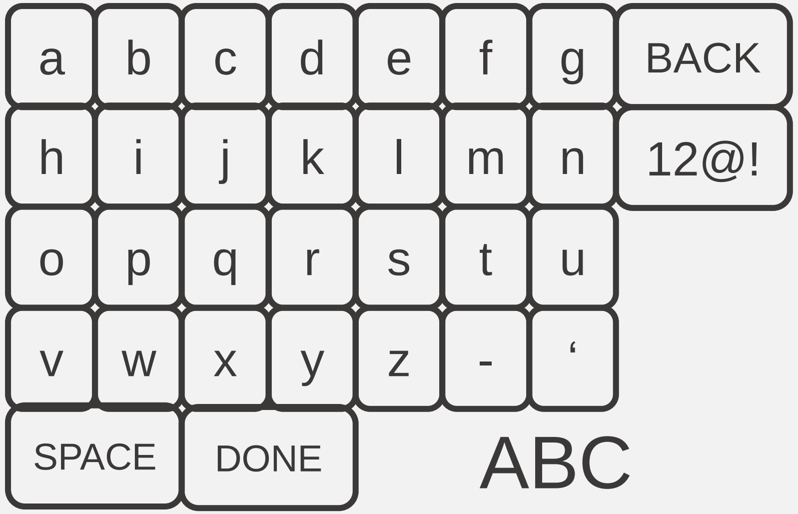
Comparison of suboptimal path detection strategies on user credit card recovery



# Appendix: Keyboard Layouts




Samsung




ABC


AppleTV

# Appendix: Keyboard Layout Results

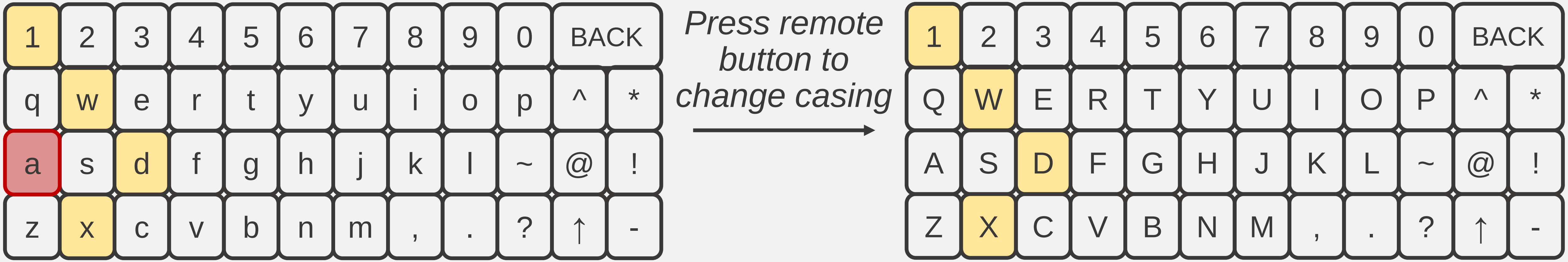
The attack achieves results far above random guessing across different keyboard layouts

| **Layout** | **Guess Cutoff (Top-K)** | | |
|---|---|---|---|
| | K = 1 | K = 5 | K = 10 |
| ABC | 85.40% | 99.55% | 99.93% |
| AppleTV | 88.97% | 99.85% | 100.00% |
| Samsung | 84.67% | 98.48% | 99.03% |
| Random Guess | $5.4 \times 10^{-4}$ % | $2.7 \times 10^{-3}$ % | $5.4 \times 10^{-3}$ % |

Results using the PhpBB prior in emulation against passwords drawn from the PhpBB set

# Appendix: Keyboard Mode Changes

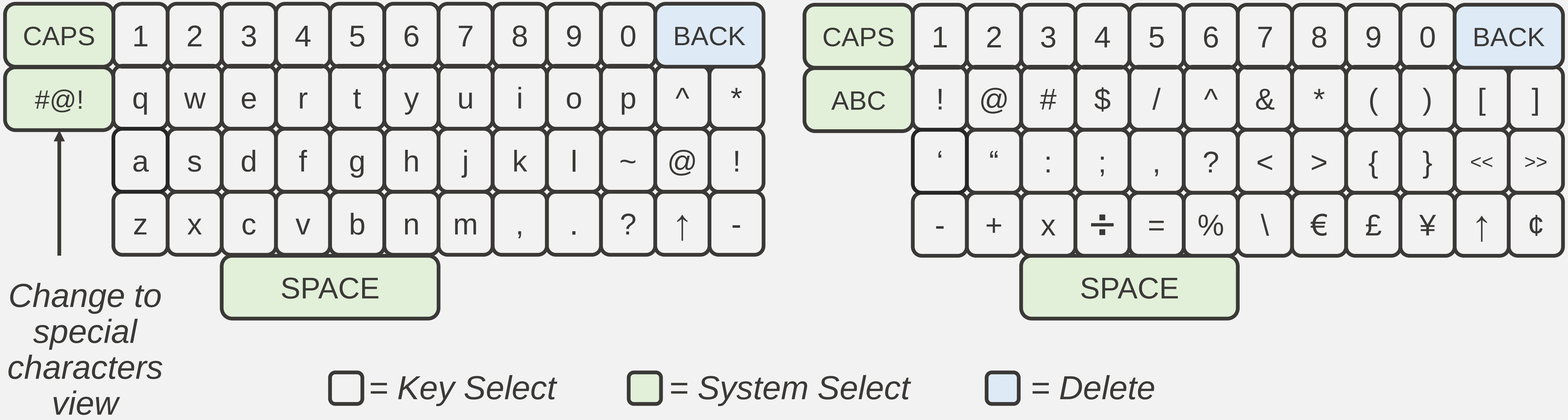
Users can change keyboard modes using inaudible shortcuts on their remote




Upon hearing 2 movements from the key ‘a’, we add keys in both views to the search queue

# Appendix: Keyboard Views

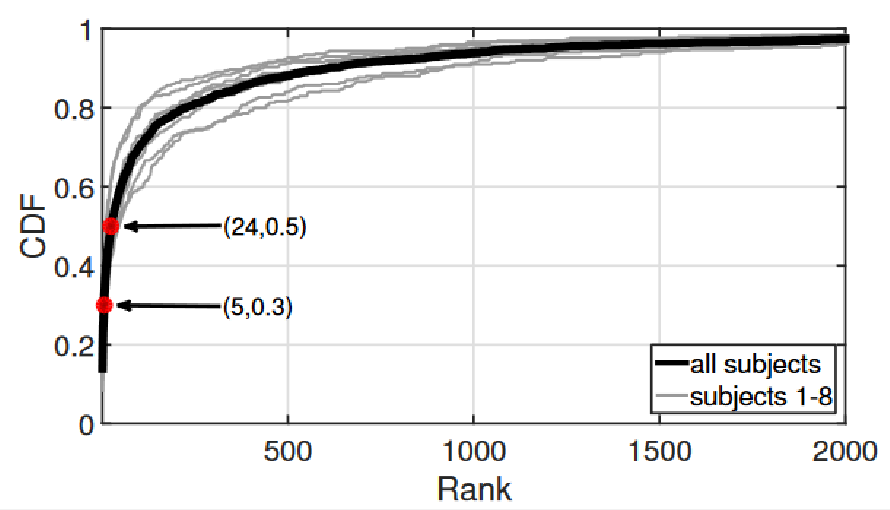
Keyboards have multiple views, each with its own character set.
Users can switch between views through dedicated keys



# Appendix: Accelerometer Side-Channels

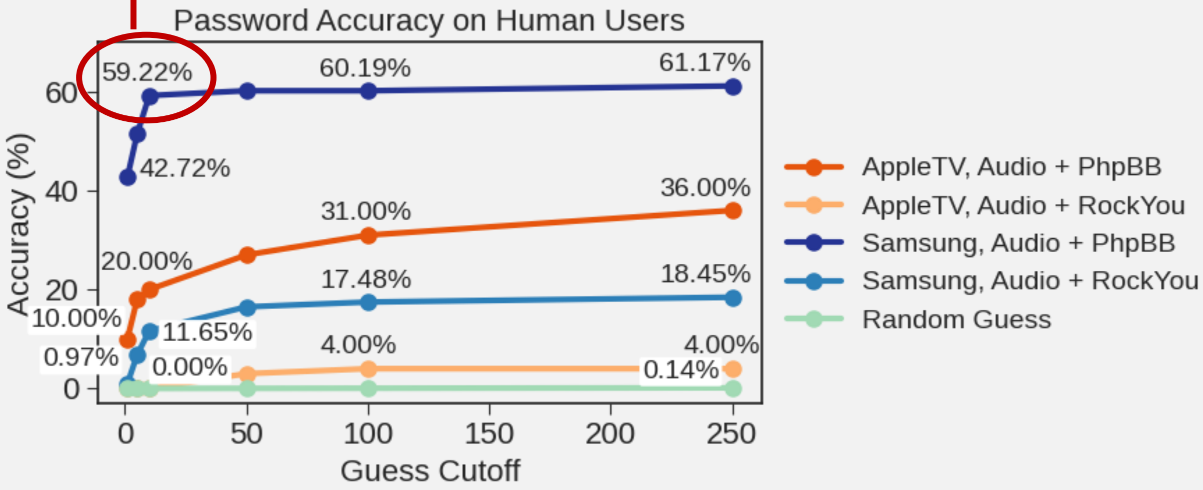
Previous attacks exploit smartwatch accelerometers to discover user keystrokes. These attacks involve similar motion dynamics

Top-K Recovery Rates using Accelerometer Side-Channels




*Users type one of 5,000 English words*

*Top-10 accuracy above 50%*



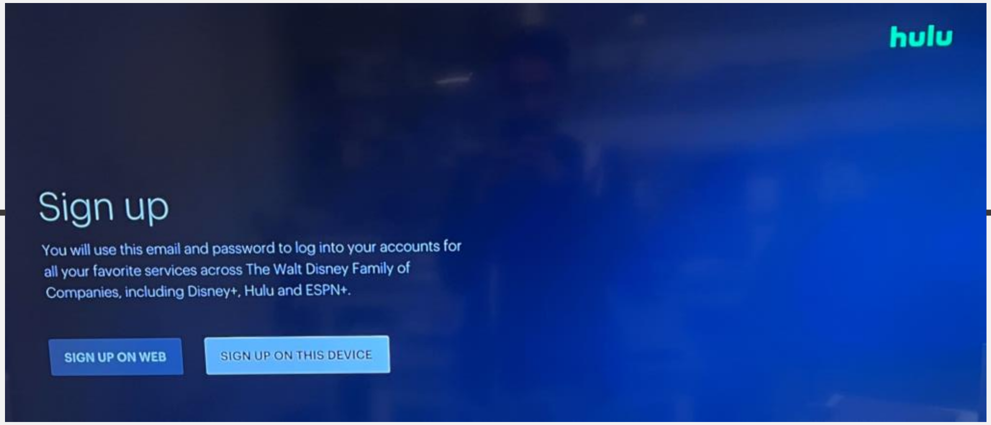

*Users type one of 184,388 Passwords*



Wang, He, Ted Tsung-Te Lai, and Romit Roy Choudhury. "MoLe: Motion leaks through smartwatch sensors." *MobiCom, 2015.*

# Appendix: Keyboard Methods

Applications can allow users to enter details through external devices (e.g., smartphones), avoiding the virtual keyboard




*On Web*

*On Smart TV*