Timing Analysis of Keystrokes and Timing Attacks on SSH

Dawn Xiaodong Song, David Wagner, Xuqing Tian

Published on Usenix Sec'01

Secure Shell

```
$ ssh xxx@192.168.48.215
xxx@192.168.48.215's password:
```

Secure Shell

```
$ ssh xxx@192.168.48.215
xxx@192.168.48.215's password:
```

- MitM
- Fake SSH client
- Key logger
- Eyes behind you
- …?

Introduction

- How does SSH work?

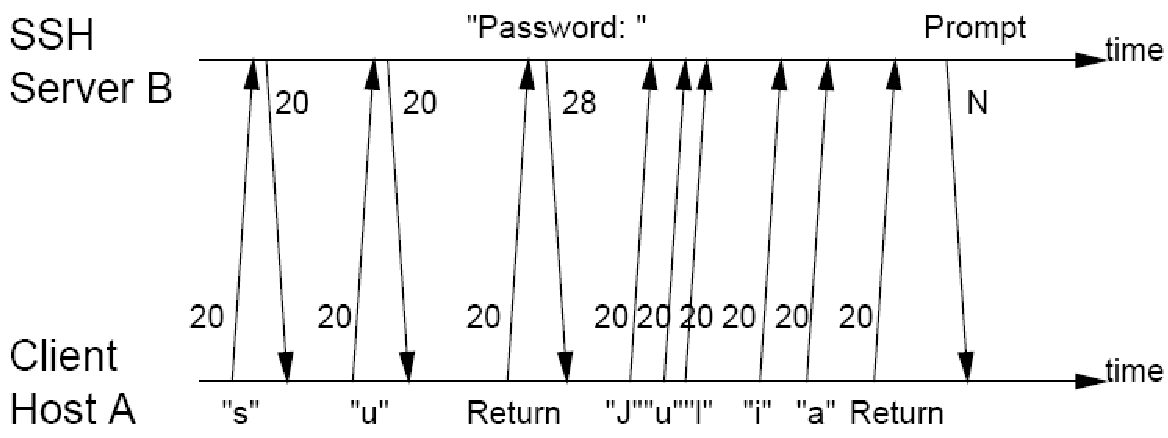
Introduction

- How does SSH work?
  - A TCP-based protocol
  - Low-latency real-time interaction by sending packets immediately
    keystrokes
    screen content updates
    mouse moves

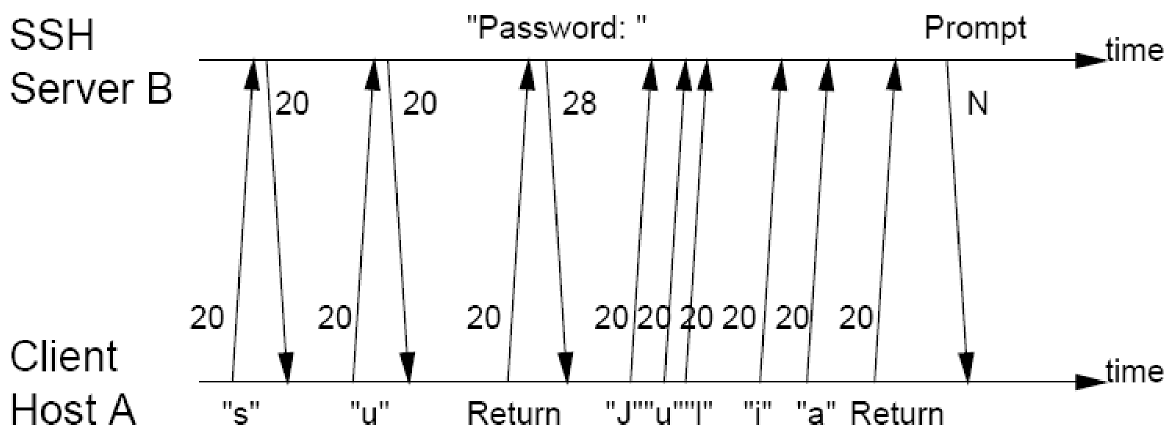
Weakness of SSH

- Packets are padded only to an eight-byte boundary
  - Attacker can estimate the approximate length of the original data
- Every keystrokes is sent immediately in a separate packet
  - Attacker can learn the exact length of user's passwords
  - And the precise inter-keystroke timing, which can be used to crack the password

Eavesdropping SSH

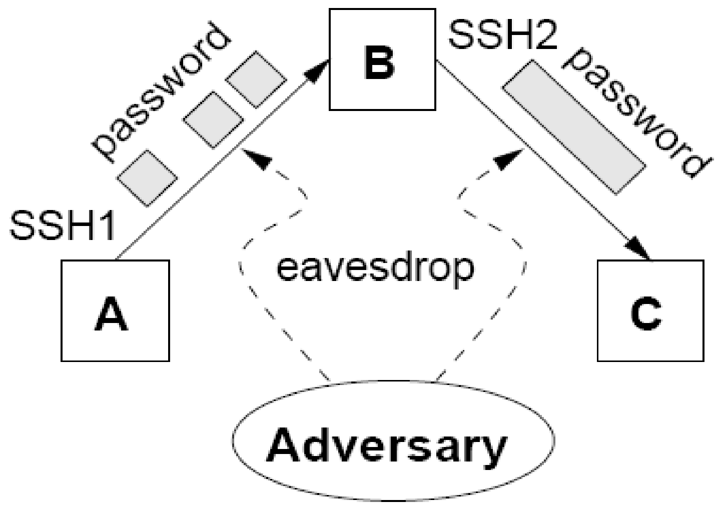


Eavesdropping SSH



- The pattern forms a signature
- The following information is leaked
  - The exact length of the password
  - Precise inter-keystroke timing of the password
  - …without breaking the crypto!

Eavesdropping SSH… Nested



Eavesdropping SSH… keystroke intervals

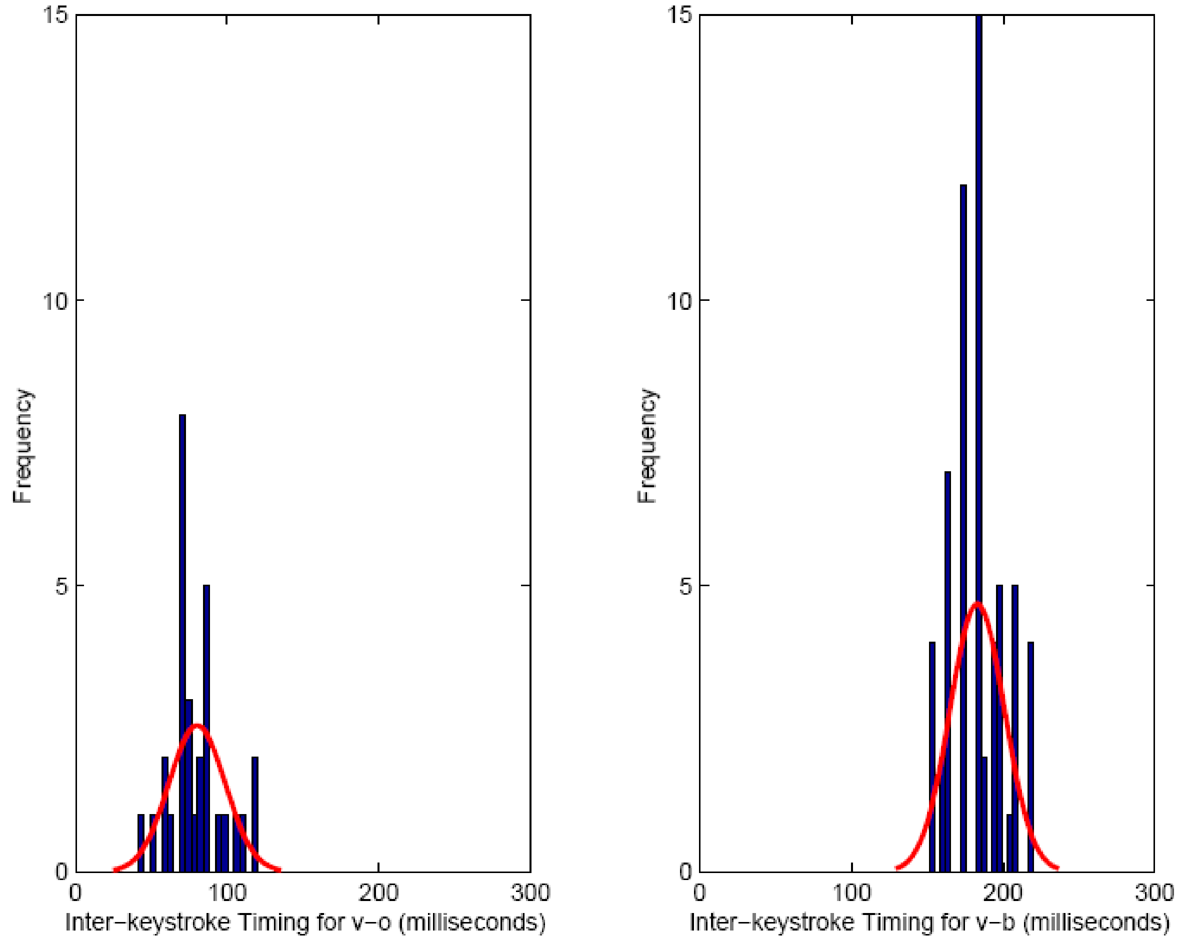
- Used to de-anonymize users by previous researchers

- Further, different password combinations require different time (intervals) when typing
  - With a carefully-designed statistical analysis, revealing information from intervals is possible

Inter-keystroke Timing Analysis

- Data Collection
  - Not possible to gather real passwords due to security and priority reasons
  - Approach 1: pick a random password and ask a user to type
    - Not necessary as only key pairs are needed
    - People tends to type passwords in group of 3-4 characters, which distorts the statistics
  - Approach 2: pick key pairs for user to type
    - We really only need **key pairs**!

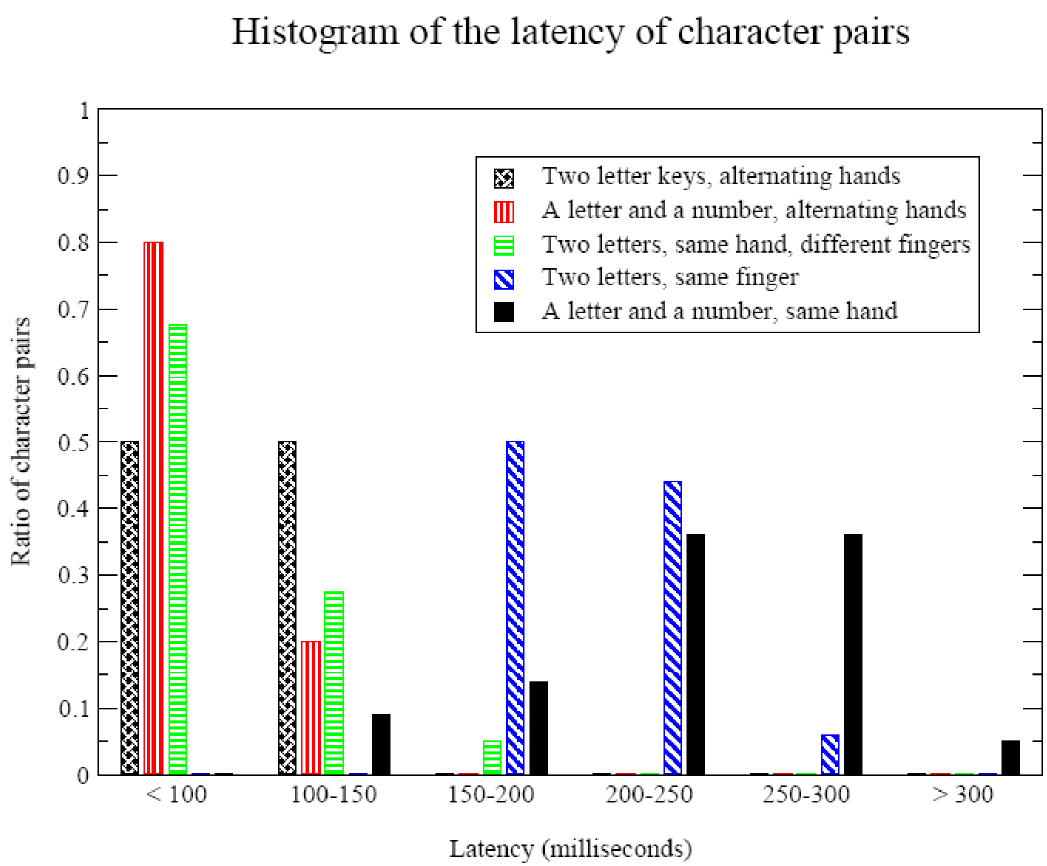
Analysis of Inter-keystroke Timing



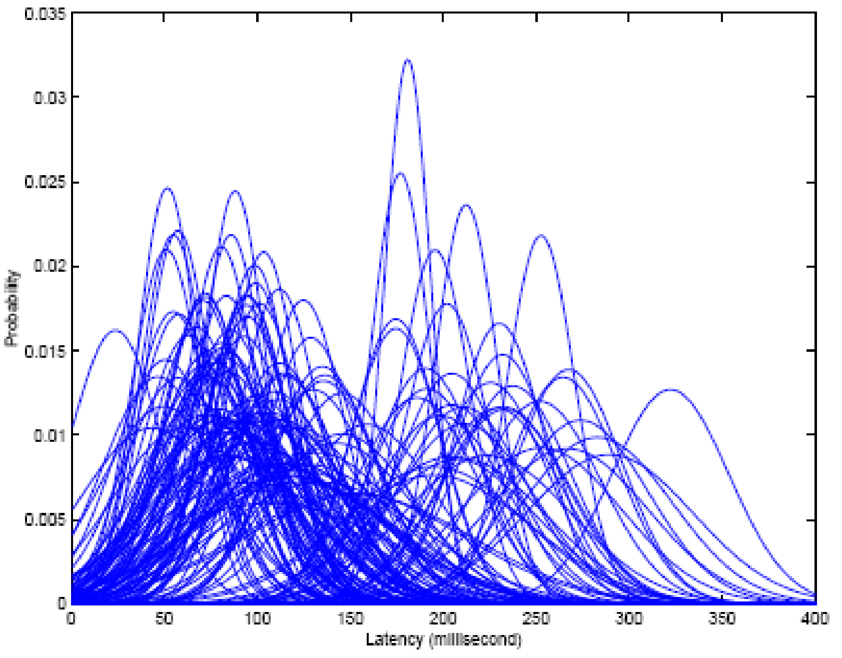
Grouping Key-pairs

- `A-L`
- `A-9`
- `Z-Q`
- `J-H`
- `K-8`

Analysis of Inter-Keystroke Timing

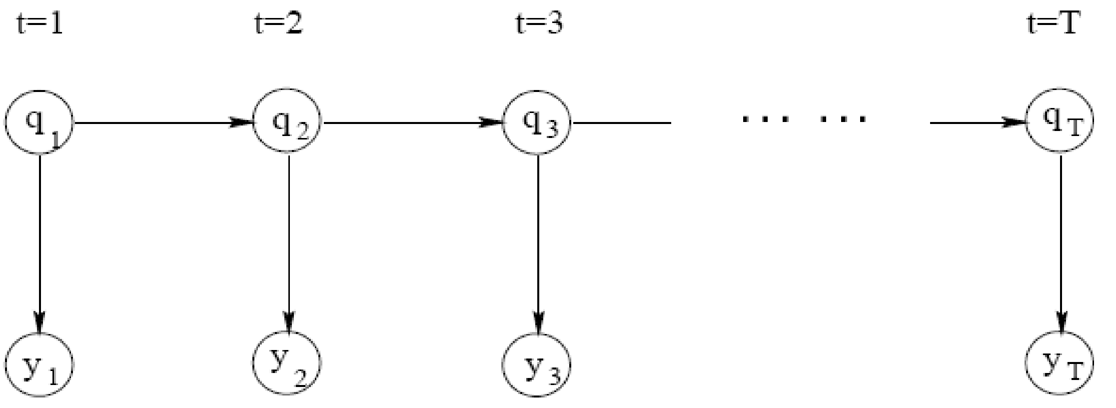


Analysis of Inter-Keystroke Timing



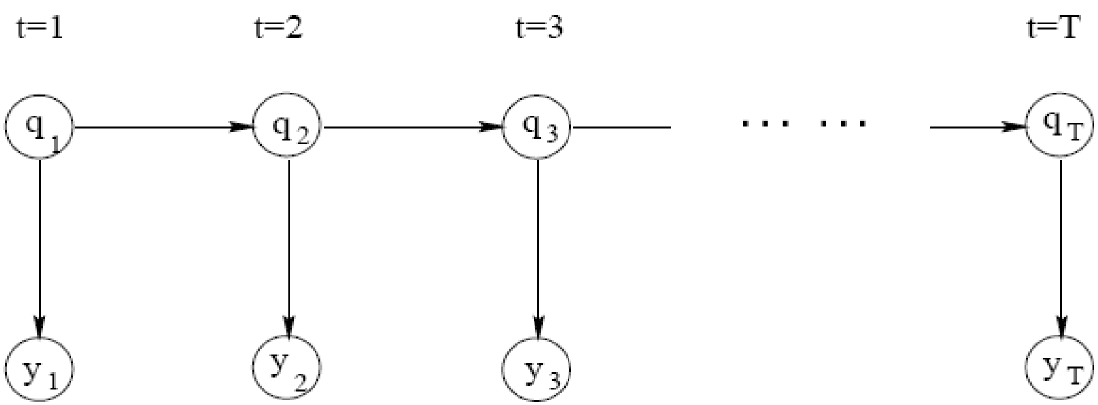
- The latency between the two key strokes of a given key pair forms a Gaussian-like distribution
- Estimated information gain available from latency information is about 1.2 bits per characteristic pair
  - significant compared to the 0.6-1.3 bits per character entropy of written English

Inferring Character Sequences



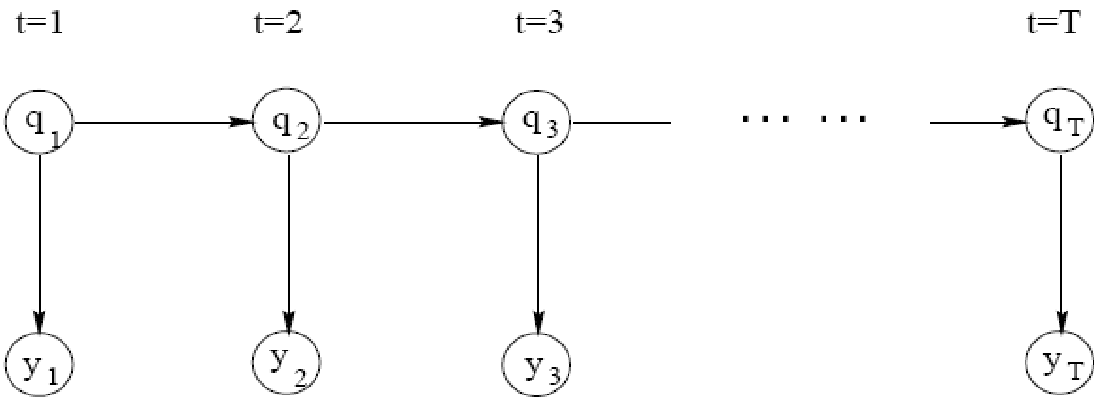
- Markov Model
  - The output (y) is only determined by the current state
  - State transitions with a probability
  - The current state is observable

Inferring Character Sequences



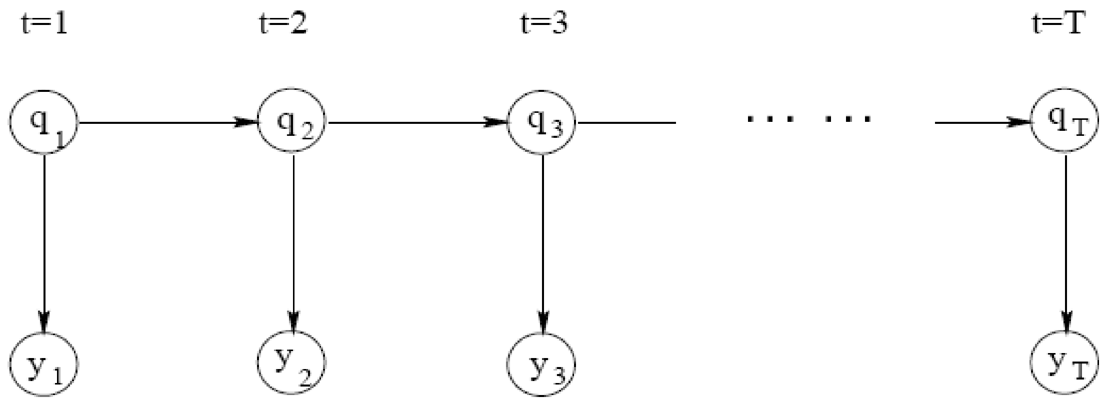
- Hidden Markov Model
  - States are *not* observable!
  - … but (some) outputs are observable, with probability distribution, we can infer the information about prior paths

Inferring Character Sequences



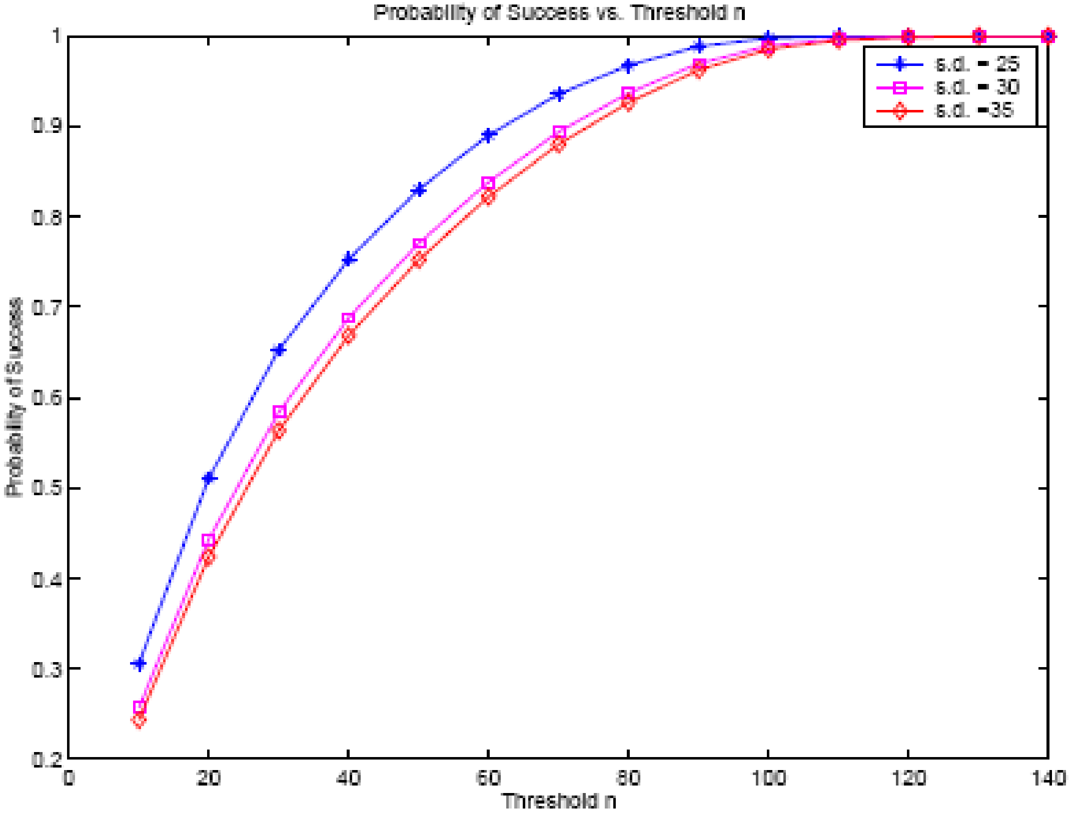
- Hidden Markov Model
  - Hidden state: each key-pair
  - Output observation: interval between keystrokes

Inferring Character Sequences



- N-Viterbi Algorithm
  - Given output *y*, the sequence of latencies, infers the top N possible character sequence
  - Calculate the possibility that a sequence will yield the output *y*

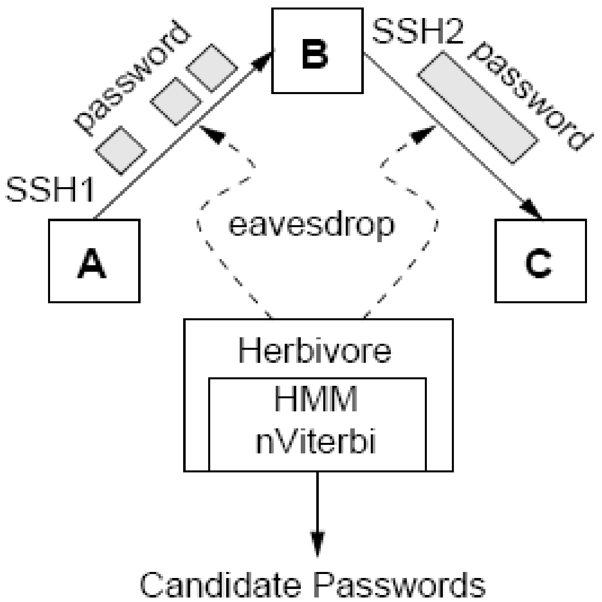
Inferring Character Sequences



Probability that the real character pair appears within the n most-likely key-pairs

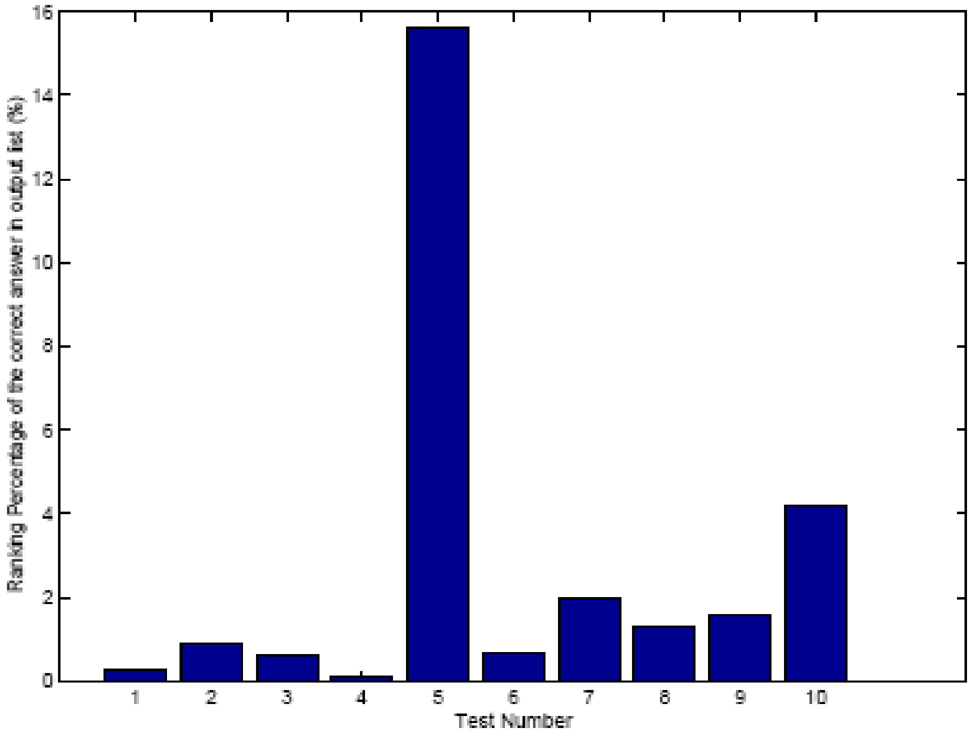
The middle curve: success rate is 90% when n=70

A POC System: Herbivore



- Targeted for nested-SSH
- Herbivore
  - Wait for packets correspond to passwords
  - Measures the inter-arrival times
  - Using n-Viterbi algorithm to generate list of candidate passwords

A POC System: Herbivore



- Percentage of the password space tried by Herbivore
  - On average, only needs to test 1/50 times as many passwords as brute-force search
- Problem
  - Herbivore is trained by the frequencies of the user at first, which is not feasible in reality

Password Inference for Multiple Users

Training Set	Test Set	Test Cases				
		Password 1	Password 2	Password 3	Password 4	Password 5
User 1	User 1	15.6%	0.7%	2.0%	1.3%	1.6%
User 1	User 2	62.3%	15.2%	7.0%	14.8%	0.3%
User 1	User 3	6.4%	N/A	1.8%	3.1%	4.2%
User 1	User 4	1.9%	31.4%	1.1%	0.1%	28.8%
User 2	User 1	4.9%	1.3%	1.6%	12.3%	3.1%
User 2	User 2	30.8%	15.0%	2.8%	3.7%	2.9%
User 2	User 3	4.7%	N/A	5.3%	6.7%	38.4%
User 2	User 4	0.7%	16.8%	3.9%	0.6%	5.4%

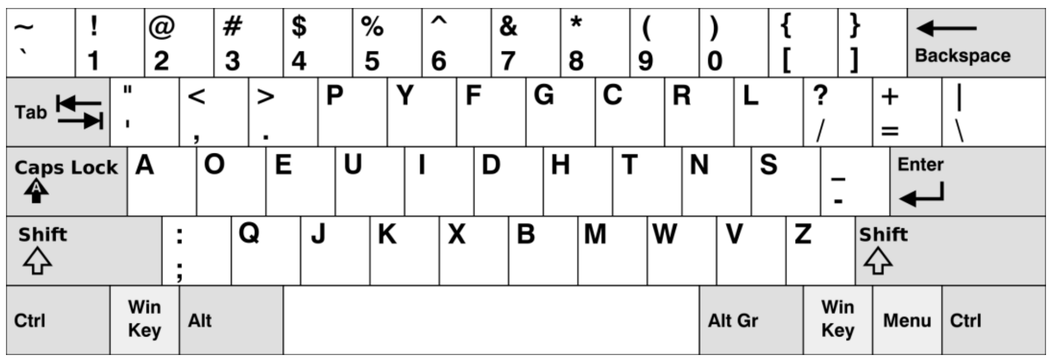
- Observations
  - Inferring is more effectively if trained by the same user
  - Distances between the typing statistics of two users can vary significantly
  - Training data from one user can be applied to infer password of another user

Countermeasures

- Send dummy packets when users are typing password
  - Signature attack will fail
  - Inter-keystroke timing information is still available to the user
- For every keystroke, delay random time before sending out the packet
  - Randomize the timing information of the keystrokes
  - Won’t work if the attacker can monitor the user login many times and compute the average of the latencies
- Send packets at constant rate
  - Breaks the responsiveness

Countermeasures

- Use a different keyboard layout



- Enable certificate-only login

- Type slowly

Contributions

- Show that **minor weaknesses** can have serious security impacts

- Showcase the possibility to infer key sequences from information leaked by keystroke intervals

In Reality…

- Sample sizes are really small…
- The attack is impossible to carry on due to network latency variations [1]
- No such attack has been found in the wild
- SSH is *not* defending against such attacks

[1] http://www.cs.virginia.edu/~evans/cs588-fall2001/projects/reports/team4.pdf