Holding Intruders Accountable on The Internet

Stuart Staniford-Chen

Todd Heberlein

Department of Computer Science
University of California, Davis

stanifor@cs.ucdavis.edu

http://everest.cs.ucdavis.edu/arpa/

Our thanks to ARPA for funding

5/8/95

Outline

Introduction

- Title/Credits
- Outline

The Problem

- The problem
- Previous Approaches

Our Work - Overview

- Basic idea
- Advantages and the Drawback
- Design criteria

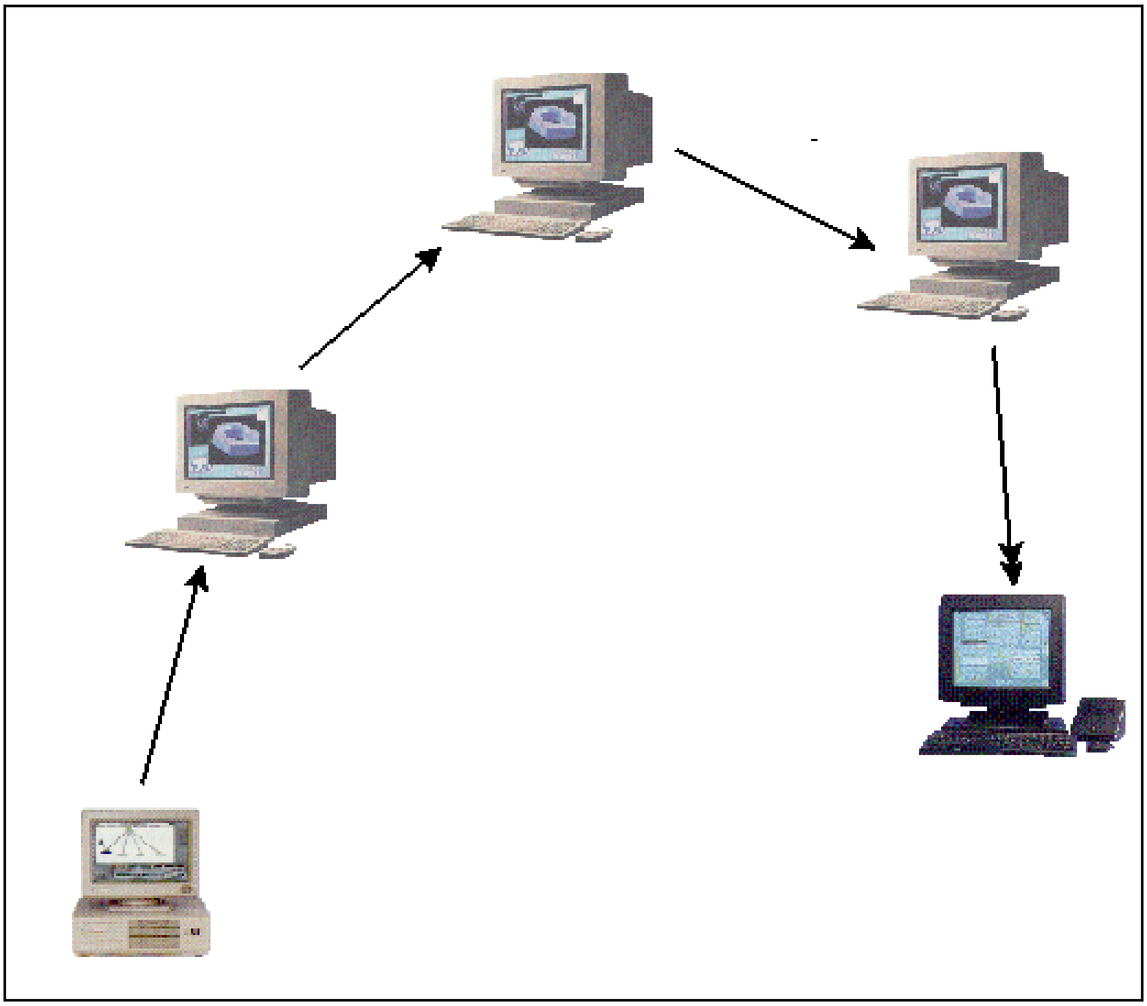
Our Work - Technicalities

- Local thumbprints
- Principal Component Analysis
- Choice of thumbprint function
- Experimental results
- Conclusion and future directions

The Problem

The FBI hires you to trace a hacker across the Internet.

How do you do it?



Previous Approaches

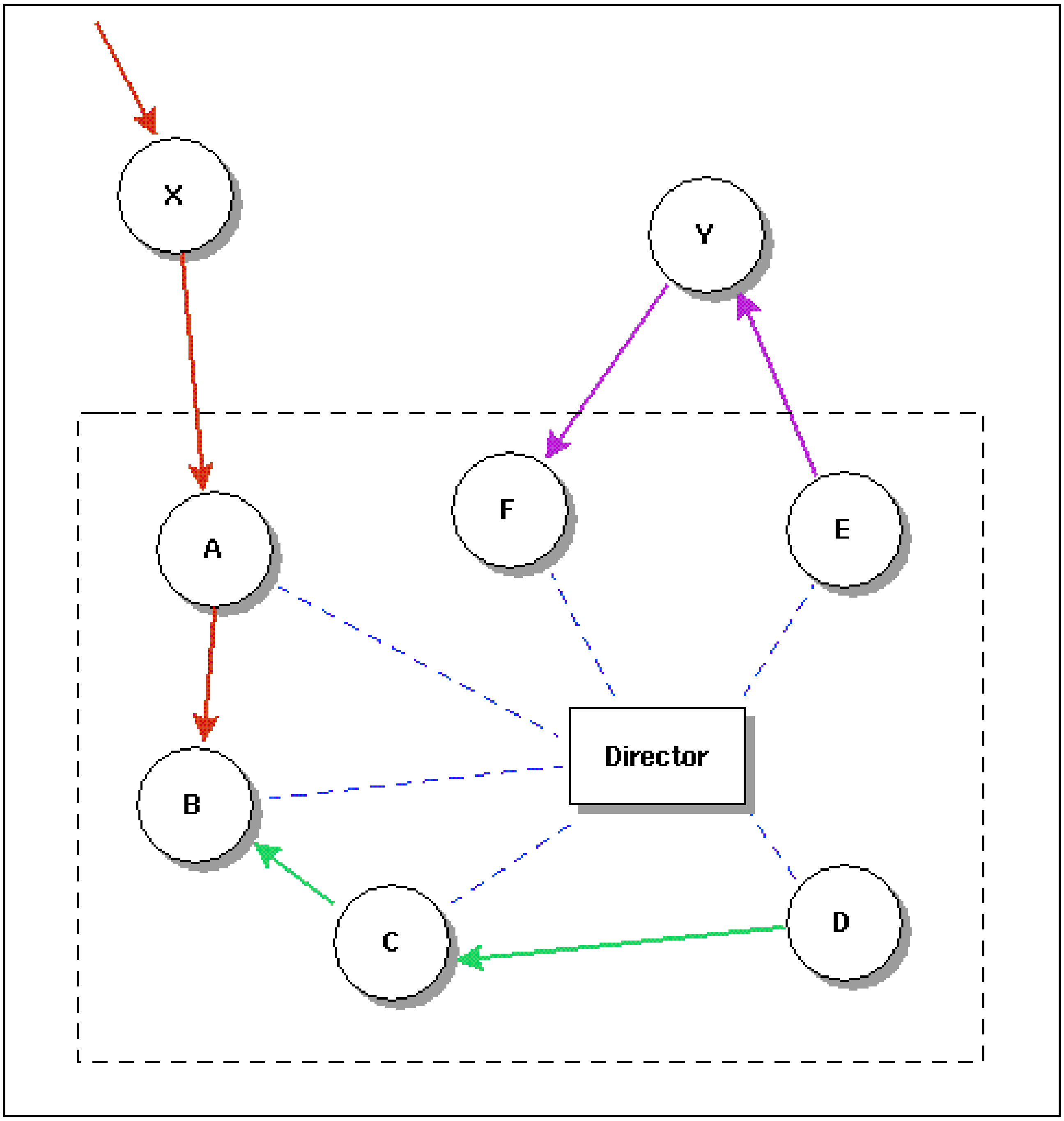
Manual techniques

- Comparing connection times in logs
- Wading through printouts of keystrokes

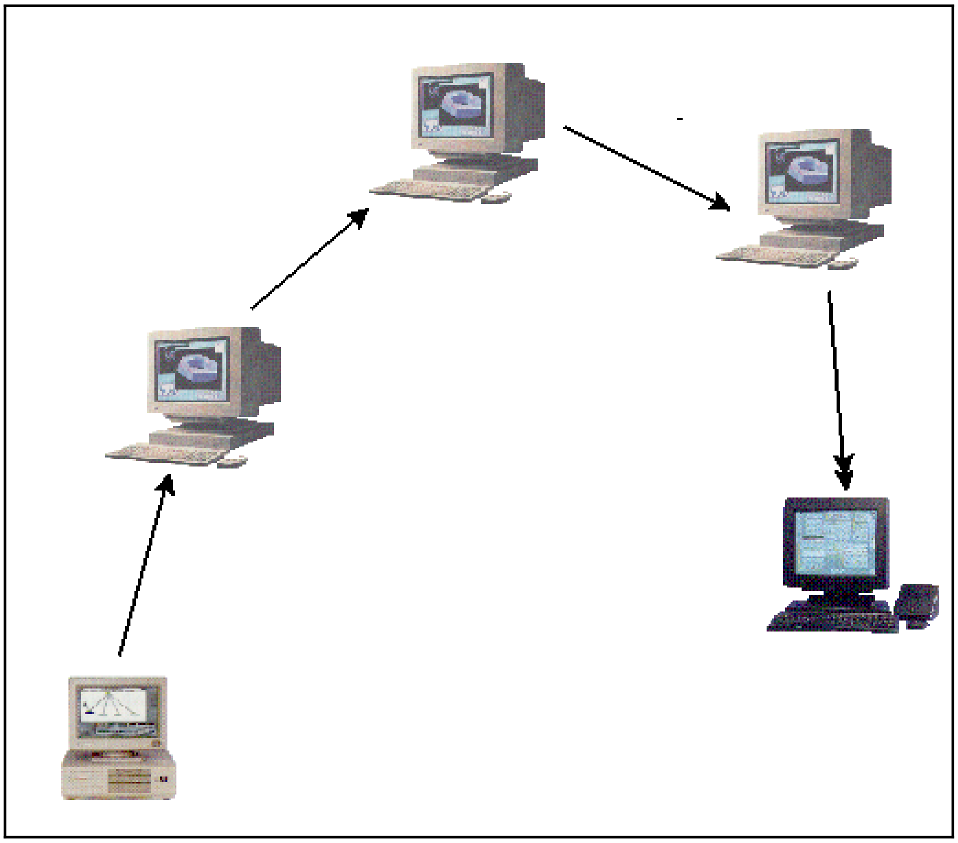
Automated Tracing

- Caller-id (USAF)
- DIDS (UC Davis, Trident)
- CIS (Jung et al)

DIDS



Thumbprints - Idea



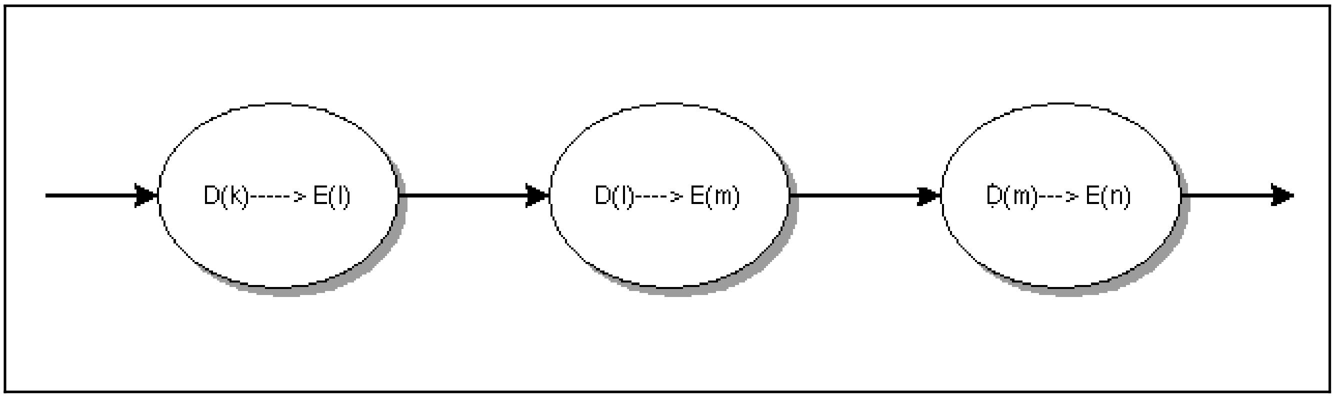
- Content is the same at each place
- So save a "robust hash" or thumbprint of connection data at each place.
- Compare the thumbprints later to see if the traffic was the same.
- Thumbprint each minute separately.

Advantages

- Does not invade privacy
- Space efficient
- Not dependent on every host in the chain

The Drawback

- Useless in presence of encryption.



Thumbprints - Design criteria

- **Easy to compute**
- **Small**
- **Sensitive**
- **Tolerant of errors such as**
  - Clock skew
  - Propagation delays
  - Missed Packets
- **Additive**

1	2	3	4	5	6
38.2	6741.1	6975.7	2587.2	3446.5	2451.2
11.8	13505.5	92.8	2569.7	3388.7	2446.0

Local thumbprints

Local Thumbprints

- Definition:

$$\boxed{T = \frac{1}{n}\sum_{i=1}^{n}\phi(a_i)}$$

Here, $T \in \Re^k$ and $\phi:\{a_i\} \to \Re^k$

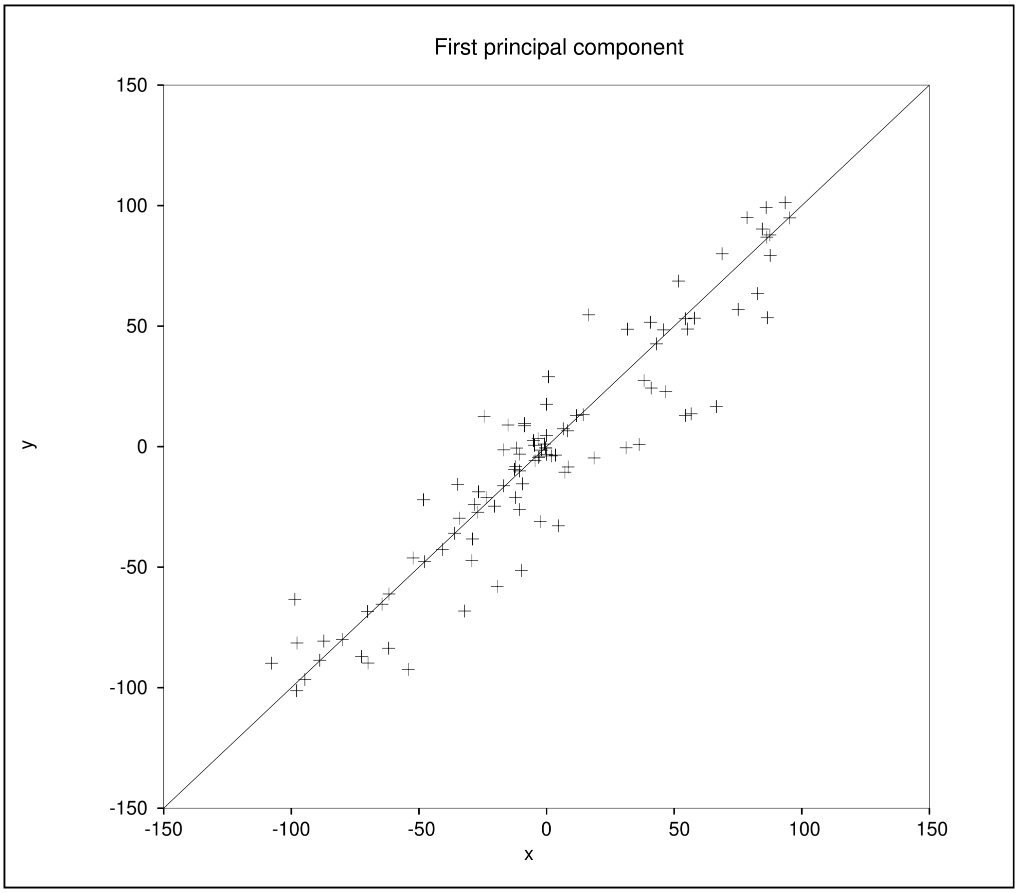
Advantages:

- Robust
- Additive
- Small
- Easy to compute

Issues:

- Sensitivity

Principal Component Analysis



- Minimizes the RMS distance of the data from the line
- Achieved by diagonalizing the sample covariance matrix
- Each eigenvector is a principal component direction
- Associated eigenvalue measures what proportion of total variation is explained by this component.

Which Thumbprint Function?

$$\boxed{T = \frac{1}{n}\sum_{i=1}^{n} \phi(a_i)}$$

$$\boxed{T_j = \sum_{a=1}^{L} \phi_j(a) f_a}$$

- Apply Principal Component Analysis to the frequency vectors (f_1, f_2,...,f_n)
- Gives the linear combinations of the f_i which explains the largest amount of the variation in the f_i
- Then gives succeeding linear combinations which explain the largest amount of the variation after the previous ones have been factored out
- This is what we want as the character frequencies which are most variable will be most useful in distinguishing different connections
- Thus gives the optimal choice of ø
- We get frequency vectors from real network traffic.

Thumbprints - Comparison Algorithm

- Take two sequences of thumbprints, and compute the probability that they would be as close as they are under the assumption that they are sequences of thumbprints of randomly chosen connections off the network.

- This requires Monte Carlo estimates of the distribution or unrelated thumbprints, together with probability calculations.

- Poor comparisons are additively combined in an attempt to improve them.

- We actually work with -log(prob) which is big for similar connections and small for unrelated ones.

Disclaimer: This is a simplification and suppresses some important technicalities and caveats -- see the paper for full details.

Thumbprints - Experimental Setup

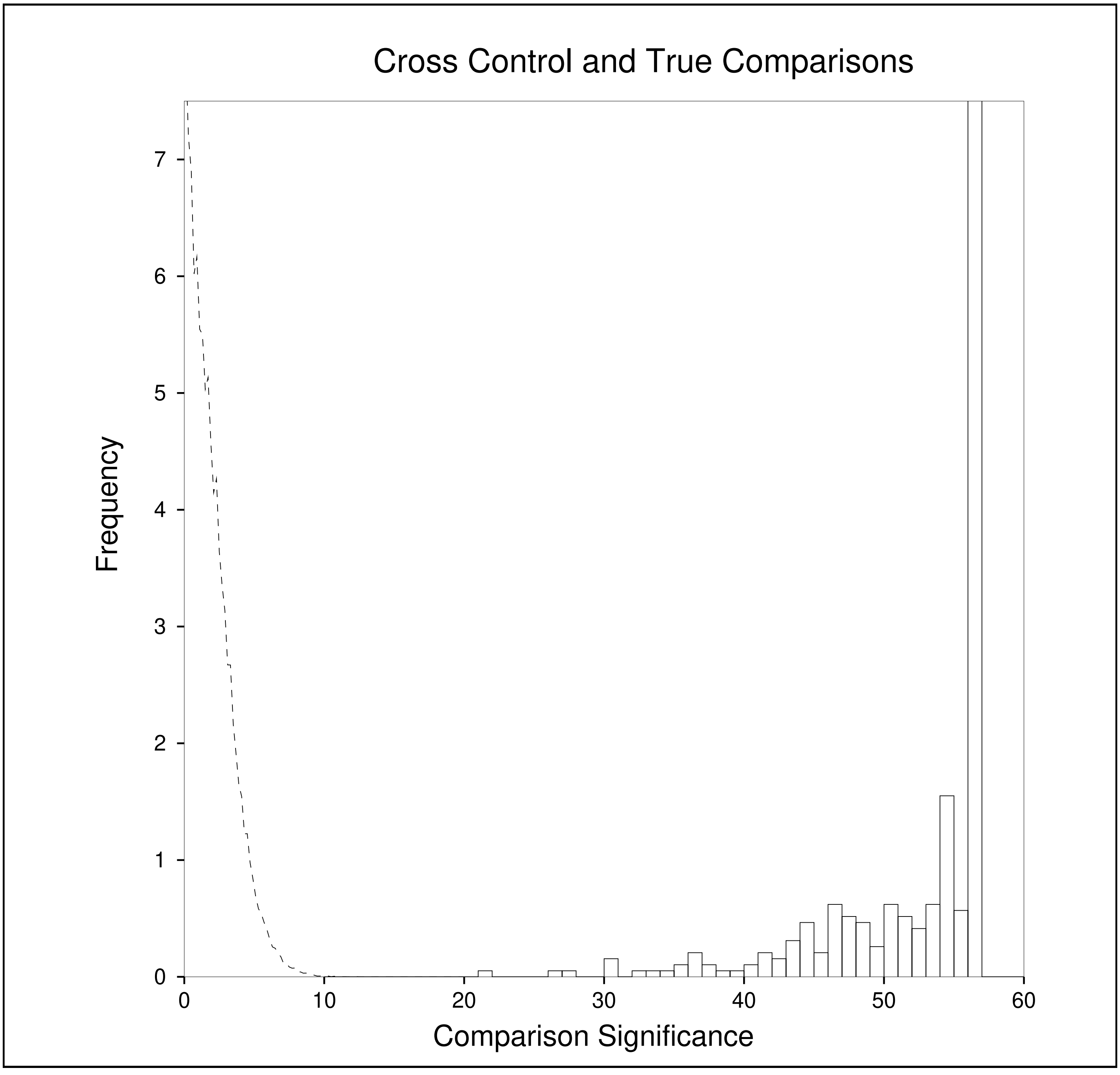
Collect data

- Sniff network traffic on a Sun 4/280 through /dev/nit
- Thumbprint the connections
- Currently only doing rlogin/telnet
- All this generally requires ~ 2% - 10% of the processor

Inject connections

- Use Expect to log through multiple machines so we can control the exact chain
- Create a circular chain, and compare the two ends - calculate the significance of the comparison.
- Also compare these connections with ambient connections off the network.
- This data is based on four minute connection segments.

Experimental Results



Conclusions and future directions

What we can do so far

- Efficiently compute thumbprints which take 24 bytes per minute and allow connections to be compared with tiny probability of error. (On ethernet with moderate data-rates).

Issues for the future

- Refine thumbprinting algorithms
  - Just how few bytes can it be done in?
  - How long should the interval be?
  - Can we avoid time division of the character stream?
  - Can we sample only some of the characters?
- What data structure should thumbprints be stored in?
- What should a tracing protocol look like?
- Build a prototype tracing system