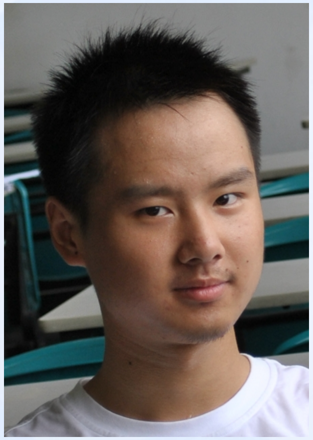
Data Leak Detection As a Service

Xiaokui Shu and Danfeng (Daphne) Yao

Department of Computer Science

Virginia Tech

Blacksburg, Virginia, US

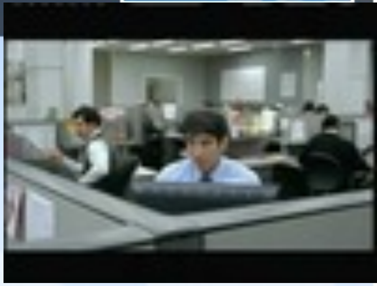


Xiaokui Shu
(3rd year PhD student)

danfeng@cs.vt.edu
http://people.cs.vt.edu/~danfeng/

SECURECOMM 2012, Padua Italy

Data loss incidents – accidental or intentional



Accidental data leak

E.g., email forwarding, web posting of sensitive data inadvertently

E.g., An Eli Lilly's lawyer sent documents to a NY Times reporter by mistake '08

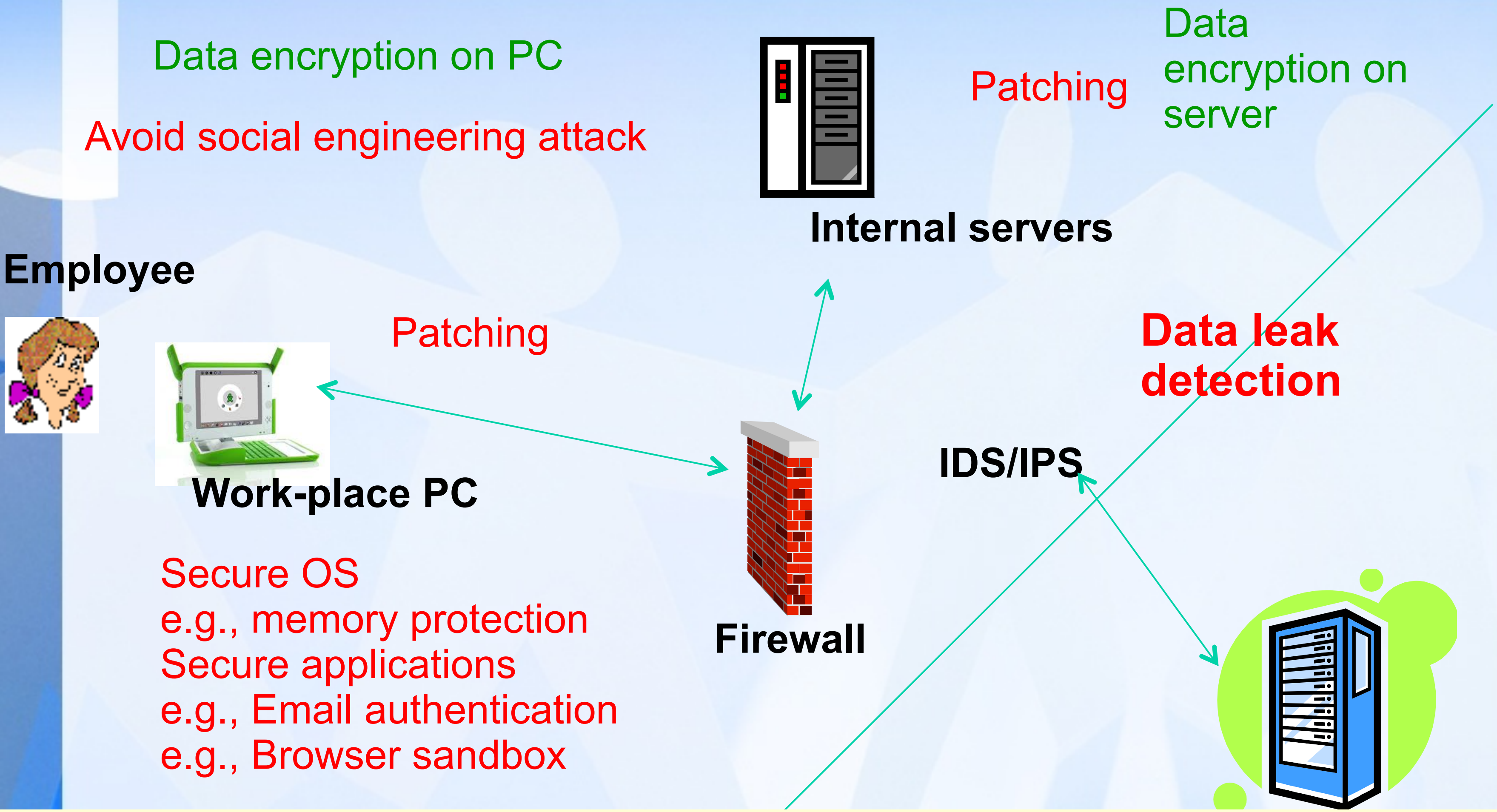
REPLY-ALL by mistake http://www.youtube.com/watch?v=beF0LTvbdfw

Survey results reveal that 59% of ex-employees admit to stealing confidential company information [Symantec]

E.g., employees emailing sensitive content to personal Webmail accounts or

E.g., downloading it onto USB drives

Multiple points where you may stop some data leak



Data encryption on PC

Avoid social engineering attack

Patching

Data encryption on server

Internal servers

Employee

Patching

Data leak detection

IDS/IPS

Work-place PC

Firewall

Secure OS
e.g., memory protection
Secure applications
e.g., Email authentication
e.g., Browser sandbox

How to minimize the exposure of sensitive data during inspection?

Our solution: inspection based on special irreversible digests

Data Loss Prevention in the Cloud

Problem: Data leaked through human errors, malware, insiders

e.g., Hydraq malware, Wikilea

Solution: e DLP

(A HP , r s

t e lu

Challenge: To preserve data privacy

Issues: providers' trustworthiness, cloud's security

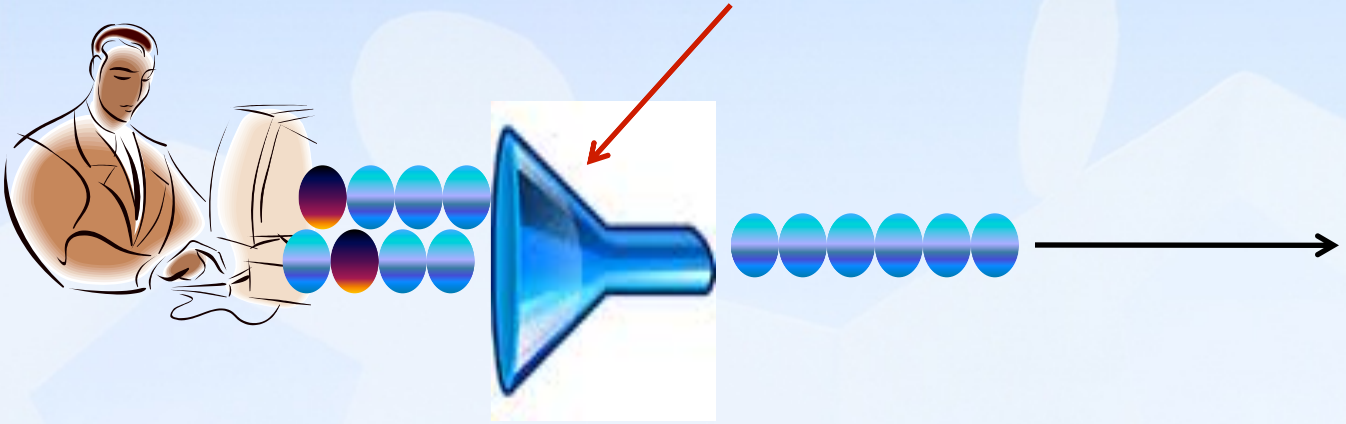
→ data owner does not reveal sensitive data to providers

Our algorithm: Providers inspect traffic for patterns, without knowing what sensitive data is.

Other DLP deployment scenarios and data exposure

- Personal firewall on PC

User-defined traffic filters for data sanitization



Internet

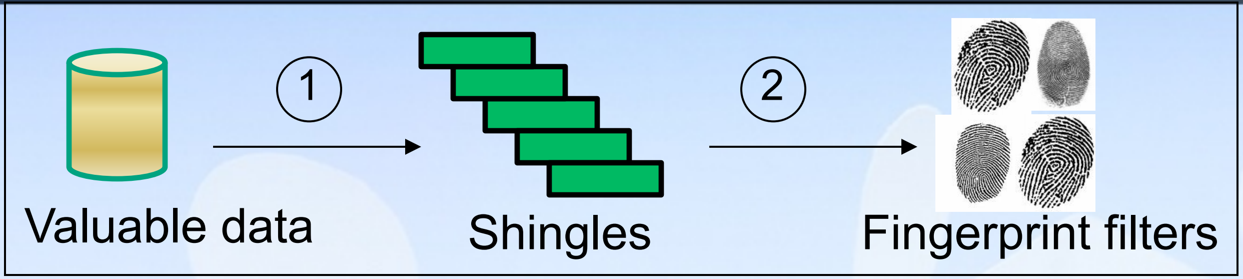
- Local area networks of organizations

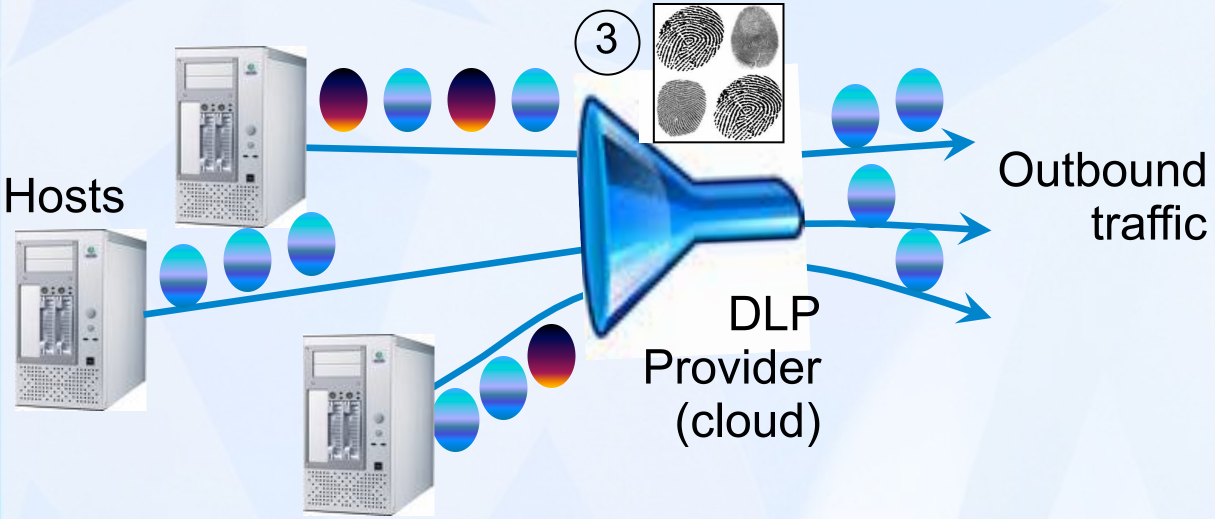
  To deploy DLP filter at gateway routers

  Data may be of any size or type

Need to avoid exposing sensitive data at filters

Overview of Our Architecture





Types of players:

1. Data owner
2. User
3. DLP provider **(honest-but-curious)**

● Sensitive data

Shingles are a sequence of fixed-size contiguous words (q-gram);

Mozilla is aware of a critical vulnerability

Mozilla is
ozilla is a
zilla is aw
illa is awa

Our Security/Privacy Goal:

Data owner delegates DLP provider to detect data leak caused by malicious attackers (i.e., malware infecting hosts or insider),

without revealing sensitive data to provider.

Assume that the traffic is not encrypted;

Host-based detection needed for encrypted traffic.

An example of fingerprints on shingles of two similar messages

Sensitive data to be protected

Critical vulnerability in Firefox 3.5 and Firefox 3.6
10.26.10 - 02:30pm
Update (Oct 27, 2010 @ 20:12):
A fix for this vulnerability has been released for Firefox and Thunderbird users.
Firefox 3.6.12 and 3.5.15 security updates now available
Thunderbird 3.1.6 and 3.0.10 security updates now available
Issue:
Mozilla is aware of a critical vulnerability affecting Firefox 3.5 and Firefox 3.6 users. We have received reports from several security research firms that exploit code leveraging this vulnerability has been detected in the wild.
Impact to users:
Users who visited an infected site could have been affected by the malware through the vulnerability. The trojan was initially reported as live on the Nobel Peace Prize site, and that specific site is now being blocked by Firefox's built-in malware protection. However, the exploit code could still be live on other websites.

10 smallest fingerprints: (**4482868, 5207155, 5538456, 16590970, 18891336, 28959745, 29523072, 30605011, 46912339, 47163843**)
Total fingerprints set size: **756**
SHA-1:
3c1e4ca6505e5d307cfe105104233e1b82b39b33

Captured payload in outbound traffic

<p>Critical vulnerability in Firefox 3.5 and Firefox 3.6</p>
<p>10.26.10 - 02:30pm</p>
<p>Update (Oct 27, 2010 @ 20:12):<br />
A fix for this vulnerability has been released for Firefox and Thunderbird users.</p> <p>Firefox 3.6.12 and 3.5.15 security updates now available<br /> Thunderbird 3.1.6 and 3.0.10 security updates now available</p> <p>Issue:<br />
Mozilla is aware of a critical vulnerability affecting Firefox 3.5 and Firefox 3.6 users. We have received reports from several security research firms that exploit code leveraging this vulnerability has been detected in the wild.</p>
<p>Impact to users:<br />
Users who visited an infected site could have been affected by the malware through the vulnerability. The trojan was initially reported as live on the Nobel Peace Prize site, and that specific site is now being blocked by Firefox's built-in malware protection. However, the exploit code could still be live on other websites.</p>

10 smallest fingerprints: (**4482868, 5538456, 16590970, 18891336, 28959745, 29523072, 30605011, 46912339, 47163843, 60018488**)
Total fingerprints set size: **806**
SHA-1:
e86d8771e82c613706fab67adbee2e2b0e8e762e

Rabin’s Fingerprint

$$A(t) = a_1 t^{m-1} + a_2 t^{m-2} + \cdots + a_m$$

$$f(A) = A(t) \bmod P(t)$$

A=(a_1, a_2, …, a_m) is a binary string

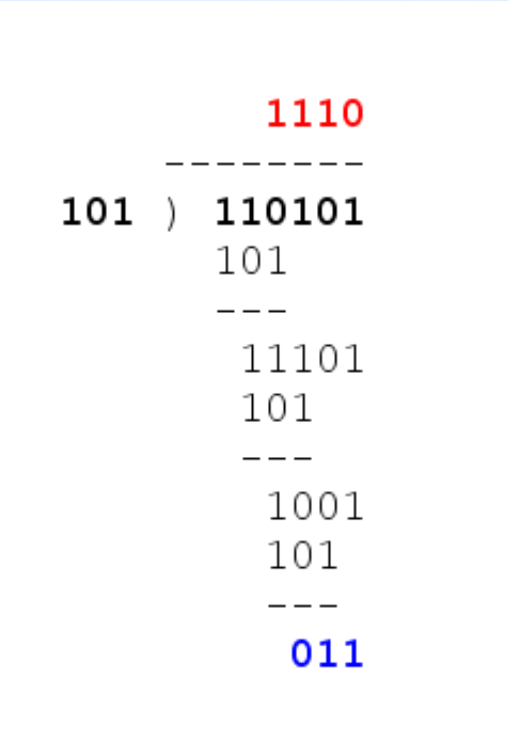
P is a irreducible polynomial.

An example

110101 mod 101 = 11 is equivalent to:

$X^5 + X^4 + X^2 + 1 \bmod X^2 + 1 = X + 1$

Advantages: oneway, fast



```
        1110
    --------
101 ) 110101
      101
      ---
       11101
       101
       ---
        1001
        101
        ---
         011
```

In binary:

- 1 – 0 = 1
- 0 – 1 = -1 = 1
- So it is just XOR operation

A naïve data-loss detection protocol

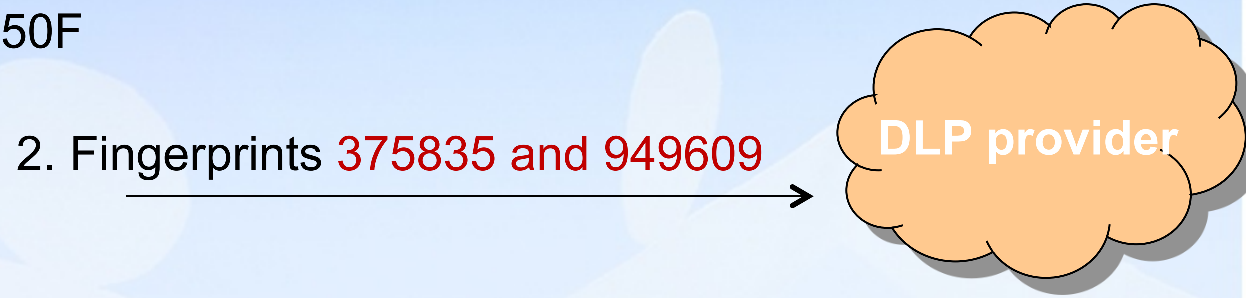
1. *Data pre-processing* -- data owner computes digests; and reveals to DLP provider a subset of the digests
   - e.g., to select a smallest 20 fingerprints to release
2. *Traffic pre-processing* – DLP provider collects outbound network traffic of data owner; and computes digests of packets
3. *Inspection* – DLP provider alerts data owner if traffic digests match data digests

   e.g., based on pre-defined threshold

Sensitivity test $\frac{\text{Number of sensitive-data fingerprints per packet}}{\text{Total fingerprints per packet}}$

The naïve detection leaks info to DLP provider if there is a match ☹

Company A has a secret recipe:
fish with garlic bake 20-min 450F



2. Fingerprints 375835 and 949609

DLP provider

1. Compute digest = f(data)

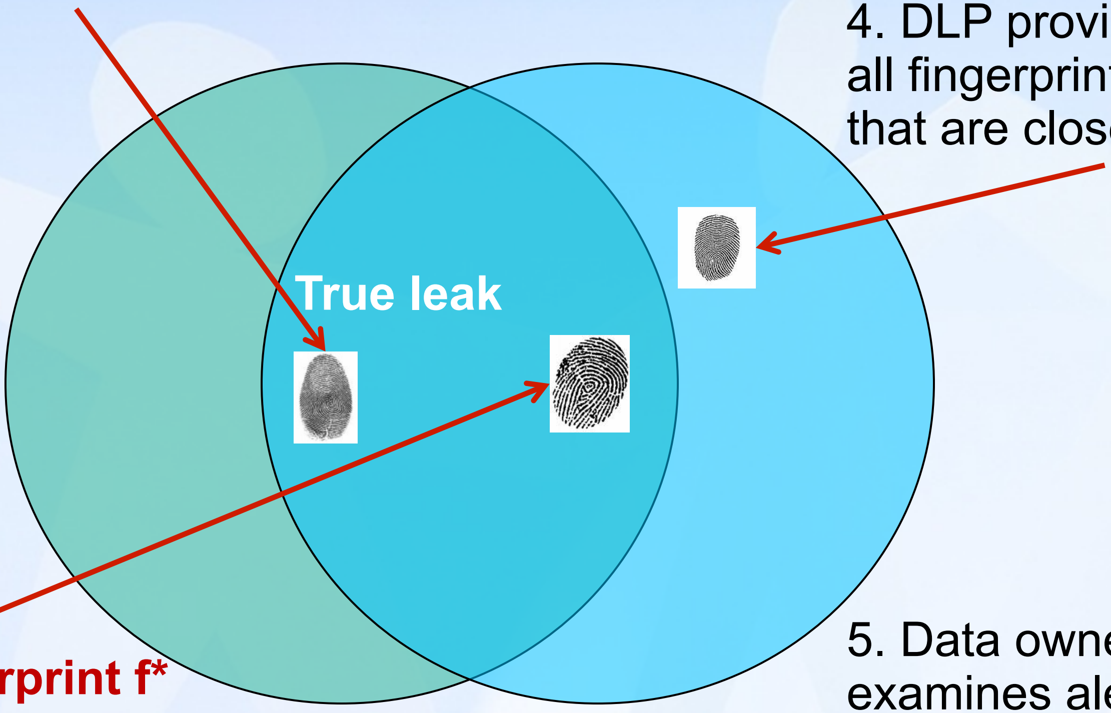
8-gram	fingerprint
Fish wit	**375835**
ish with	907948
sh with	867025
h with g	098600
with ga	114534
with gar	**949609**
…	…

3. Monitor the traffic of A

4. Find a packet whose fingerprints contain 375835 and 949609

DLP has the content of the packet,
Thus learns the secret recipe ☹

Our solution: fuzzy fingerprint – to hide sensitive fingerprint in a crowd



1. Original sensitive fingerprint f

4. DLP provider alerts all fingerprints of traffic that are close to f*

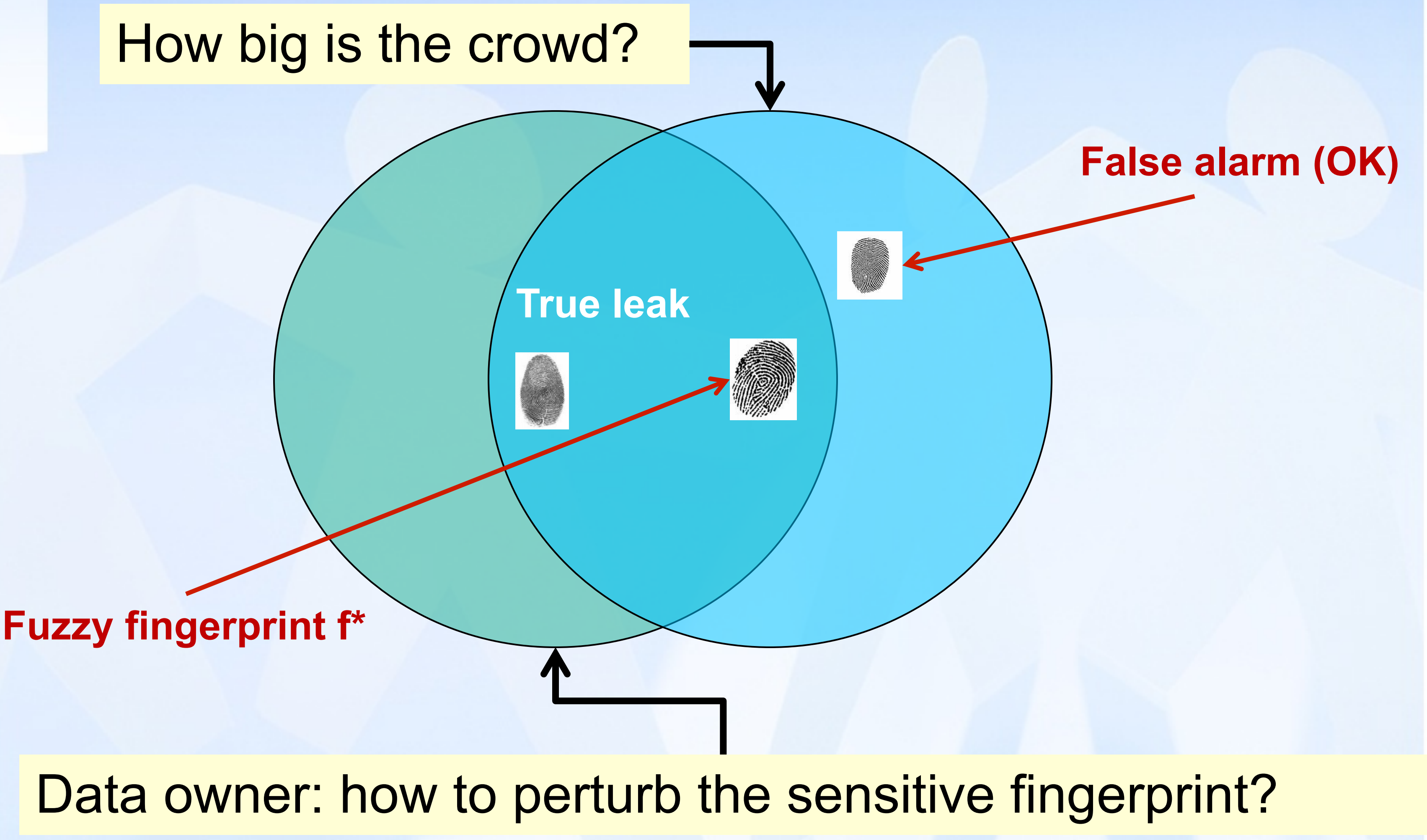
3. **Fuzzy fingerprint f*** given to DLP provider

5. Data owner examines alerts for **true leaks**

2. Perturb f by randomizing least significant bits

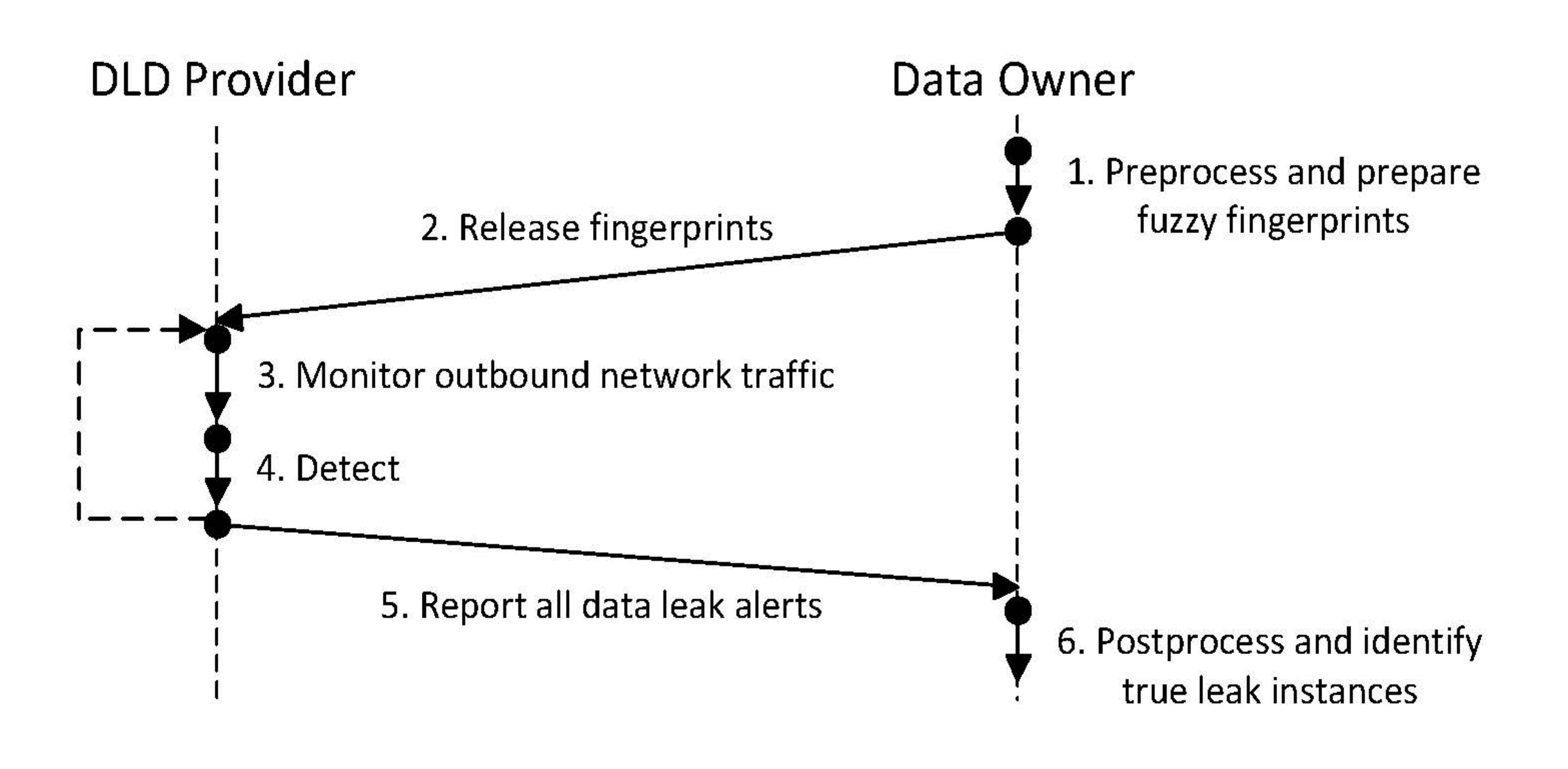
Similar to the k-anonymity in relational DB

Hide fingerprints in a crowd


How big is the crowd?

Data owner: how to perturb the sensitive fingerprint?

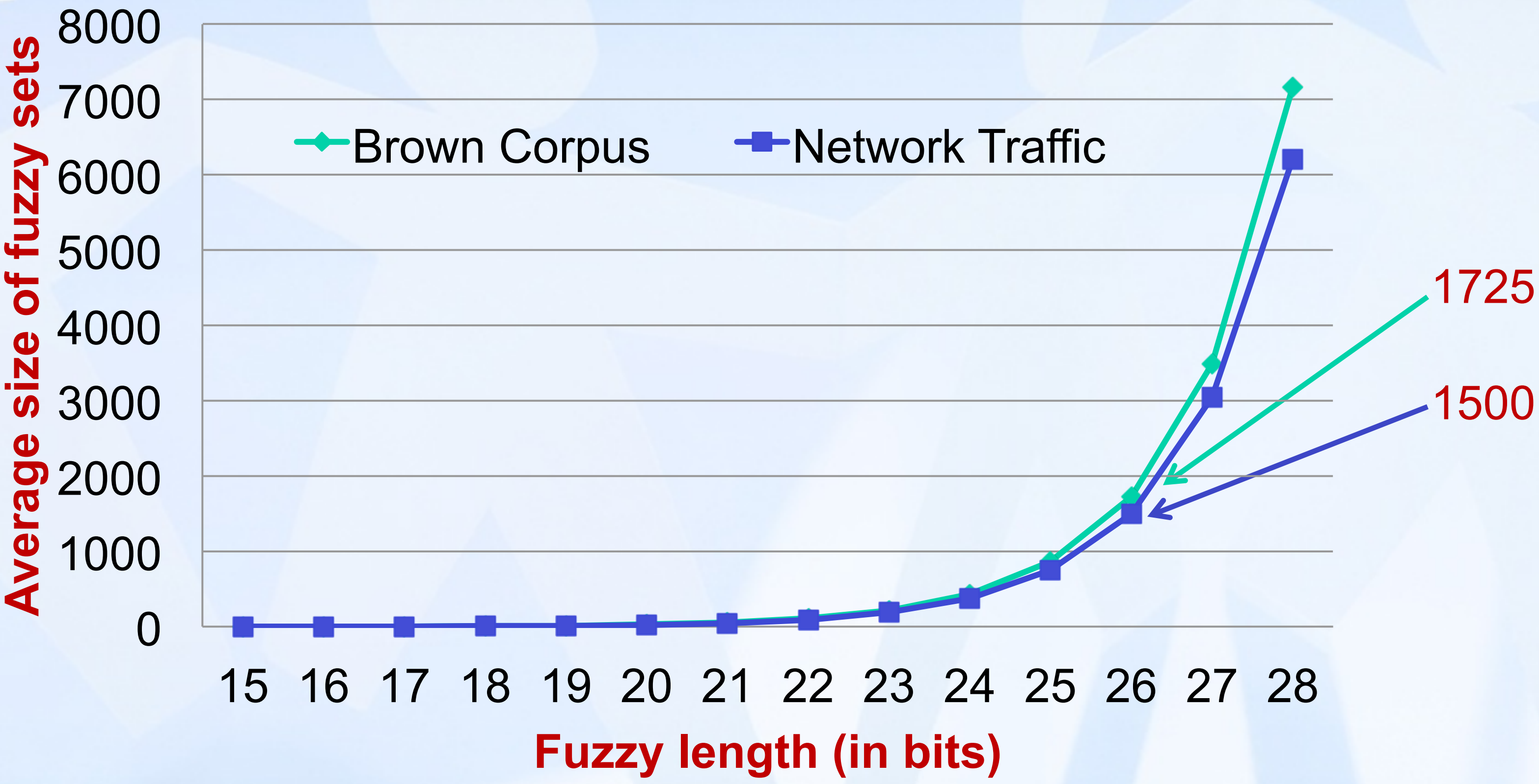
Operations in Fuzzy Fingerprints


DLD Provider
Data Owner
1. Preprocess and prepare fuzzy fingerprints
2. Release fingerprints
3. Monitor outbound network traffic
4. Detect
5. Report all data leak alerts
6. Postprocess and identify true leak instances

DLD provider cannot distinguish true leaks and false alarms

Fuzzy set size

Average sizes of fuzzy sets per fingerprint in Brown Corp and network traffic using 32-bit polynomial modulus



Generalization – bit mask

Sensitive fingerprint f 01000101111011010111100010

Fuzzy fingerprint f* 01000101111011100010111011

Perturb least significant bits

Data owner may randomize arbitrary bit positions

Sensitive fingerprint f 01000101111011010111100010

Bit mask _+++_+++_+__+_+_+++__++_++

Bit may change

No change

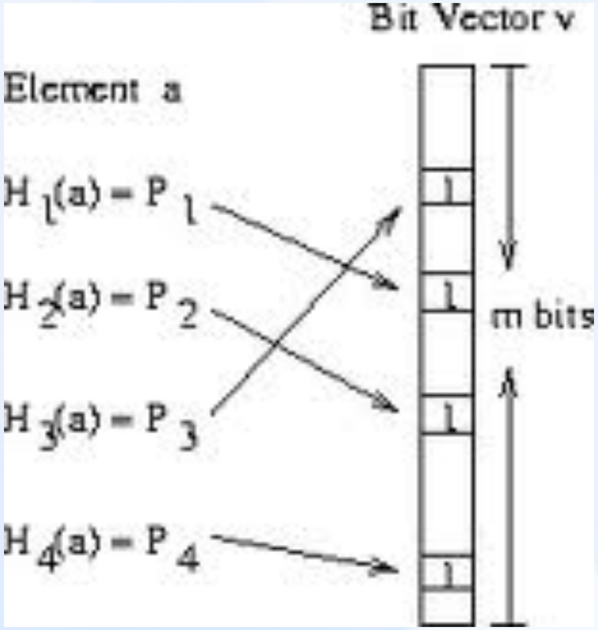
Fuzzy fingerprint f* 11000101010011010110100110

DLP provider applies bit mask to traffic; and
reports fingerprint that matches non-changing bits;

Implementation and experiments

Implemented all components of our framework in Python including packet collection, shingling, Rabin fingerprinting

Fingerprint filter = Bloom filter + Rabin fingerprint



Bloom filter for membership test

Space saving

Pybloom library

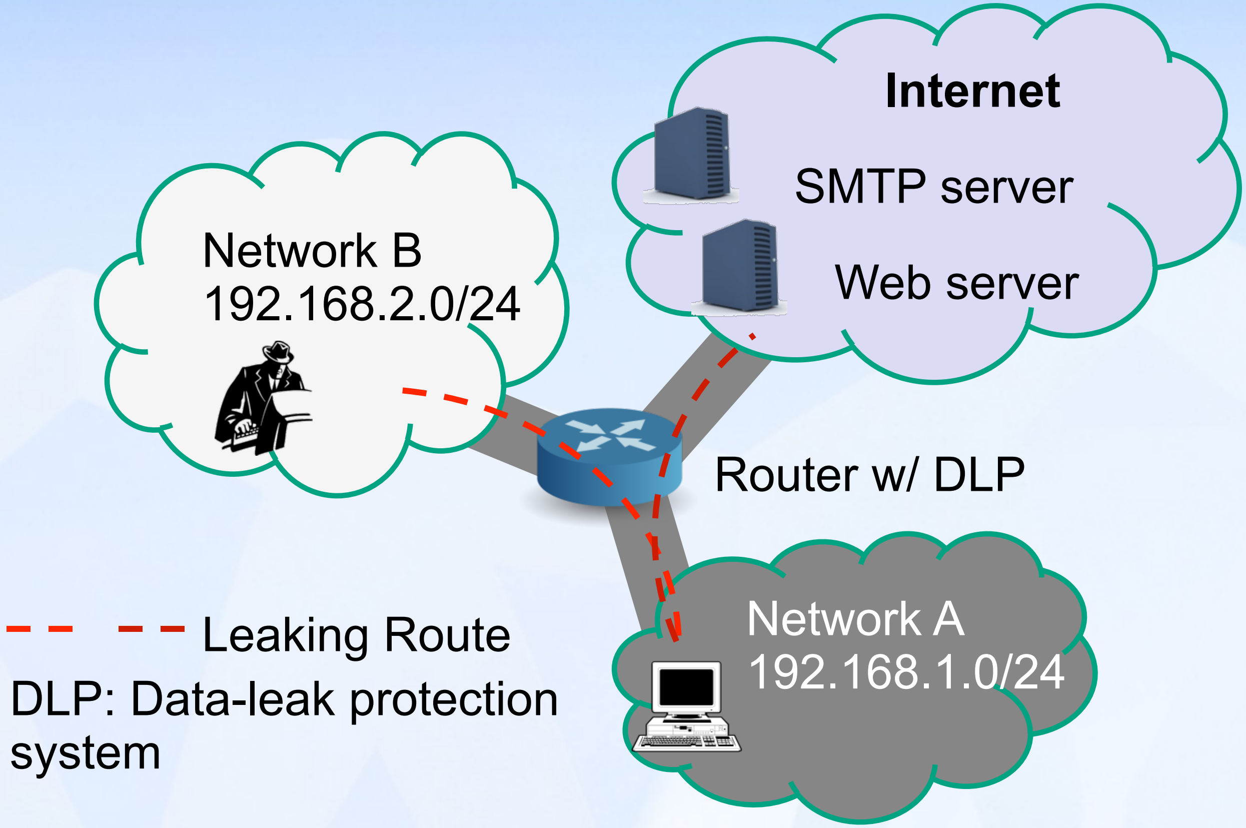
Experimental condition:

8-byte shingle

32-bit polynomial

1024-byte packet payload

Setup of the malware test



Internet

SMTP server

Web server

Network B
192.168.2.0/24

Router w/ DLP

Network A
192.168.1.0/24

Leaking Route

DLP: Data-leak protection system

We detect packets whose sensitivity values are above a threshold

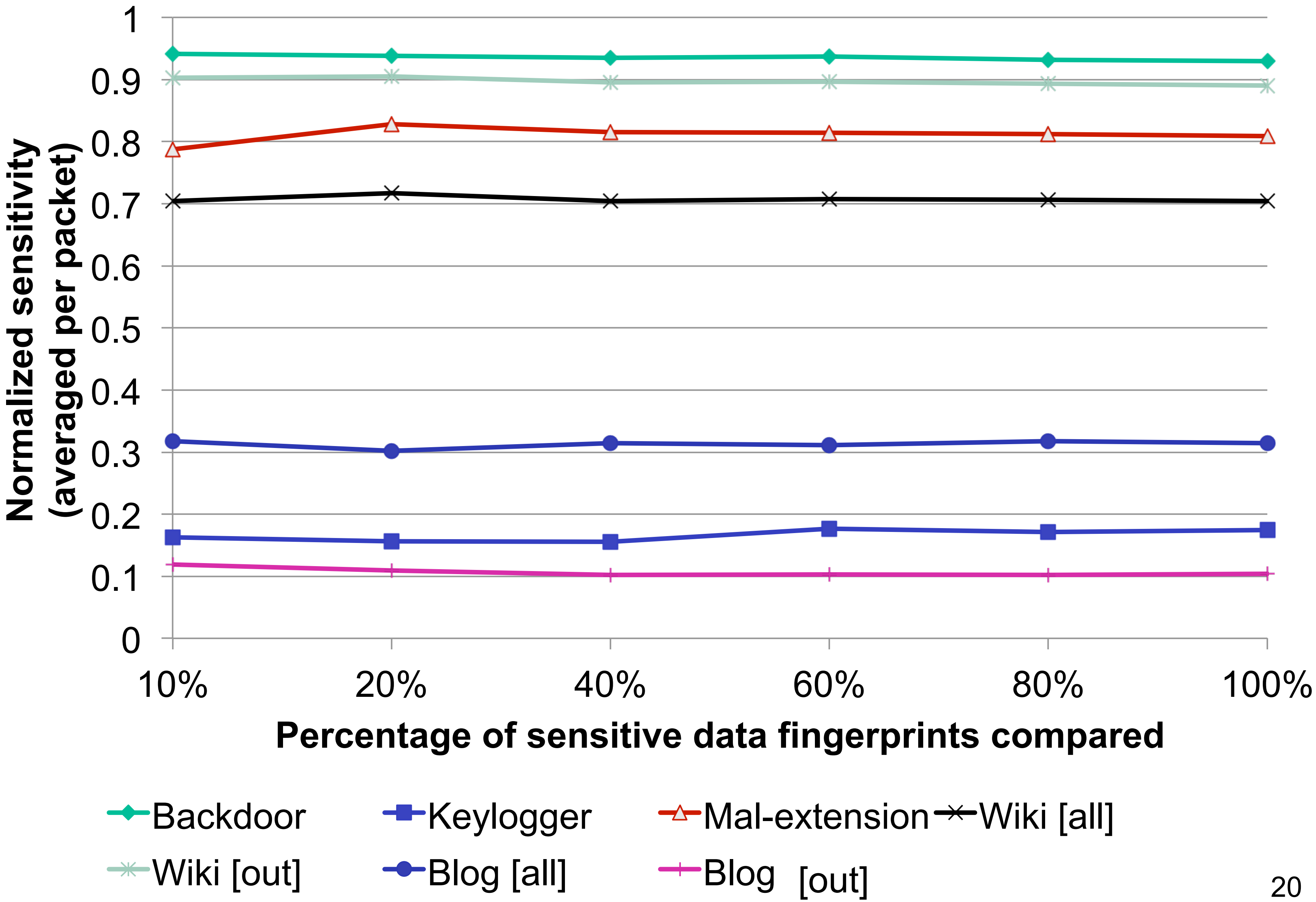
Sensitivity test:

$$\frac{\text{Number of sensitive-data fingerprints per packet}}{\text{Total fingerprints per packet}}$$

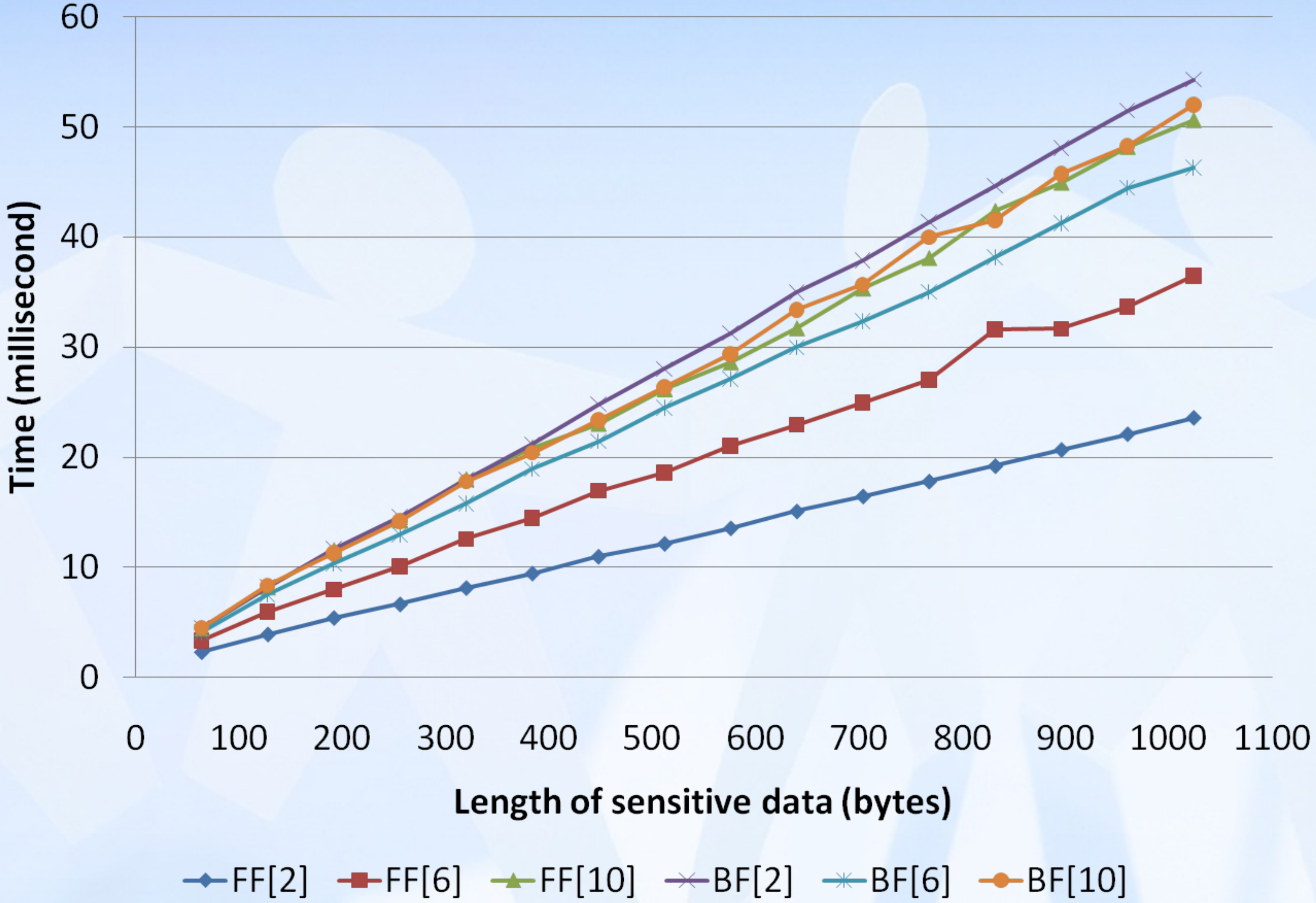
Preliminary experiments on privacy-preserving network traffic filtering

Leaking Methods	Protocol	Traffic	# of sensitive pkt found	Maximum sensitivity	Average sensitivity in sensitive pkts
Backdoor	TCP	Out	19	0.97	0.93
Keylogger	SMTP	Out	3	0.23	0.18
Malicious Browser Extension	SMTP	Out	20	0.97	0.81
Wiki System (MediaWiki)	HTTP	All	41	0.97	0.70
		Out	20	0.97	0.89
Blog System (WorldPress)	HTTP	All	37	0.95	0.31
		Out	22	0.25	0.10

Detection rates vs. size of partial fingerprint sets used



Overhead of detection with Bloom filter (BF) and fingerprint filter (FF)



FF is slightly faster than BF for detection (fingerprinting is faster than hashing)

Summary on data leak detection as a service

- Detection rates do not decrease much with fewer fingerprints ☺
  - Even when 7 fingerprints used
  - Better privacy for data owner, revealing less info to provider
- Noise tolerance if local data features are preserved
  - E.g., Wiki
  - Pervasive noise destroys patterns, e.g., Blog
    - Shorter shingles increase false positives
- Set intersection based tests are fast
- Experimentally validate min-wise independence
  - Allowing the use of partial fingerprints for detection

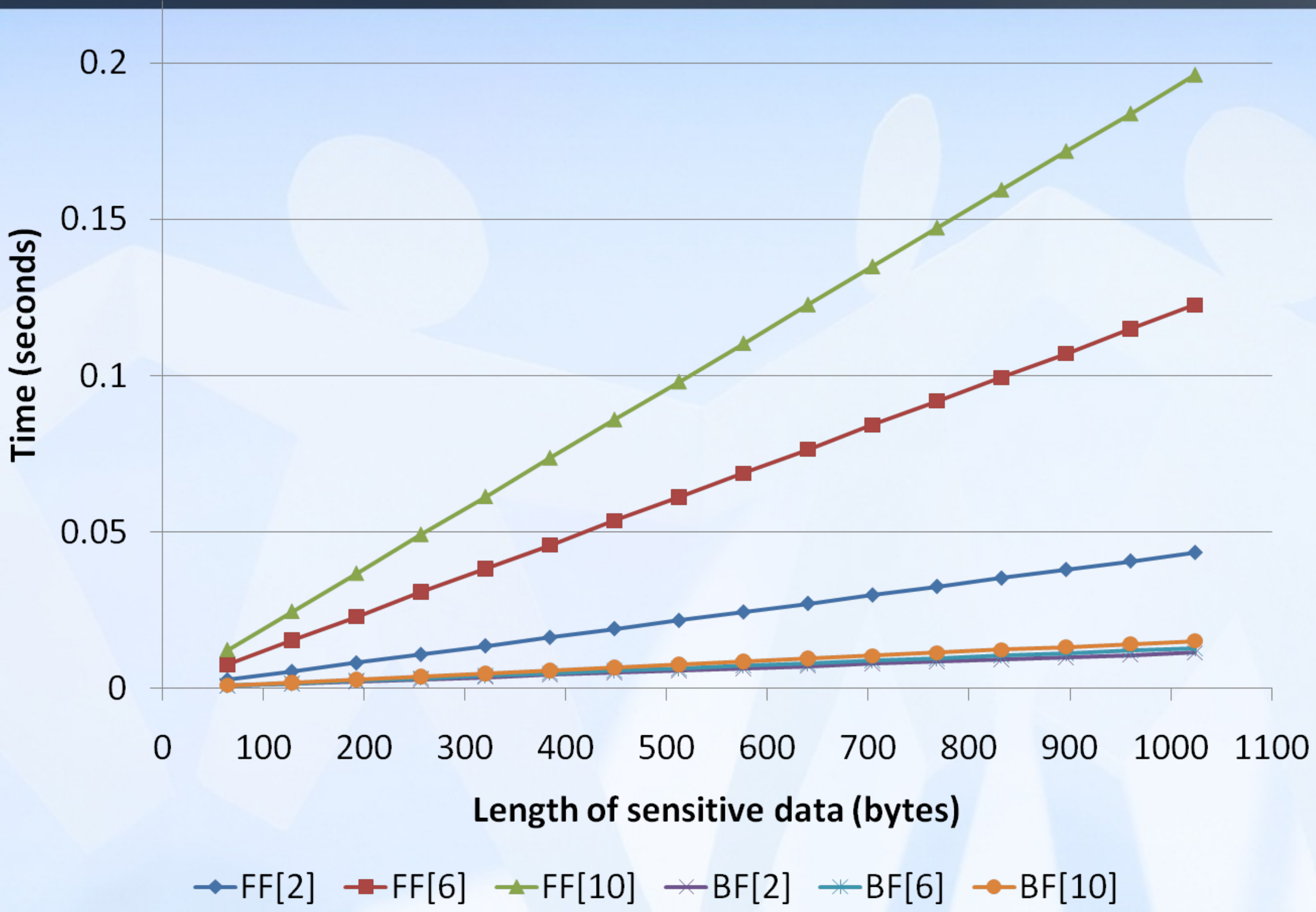
http://malaga.cs.vt.edu/demo/shingle.html for our demo

The first privacy-aware data leak protection solution

Thank you very much!

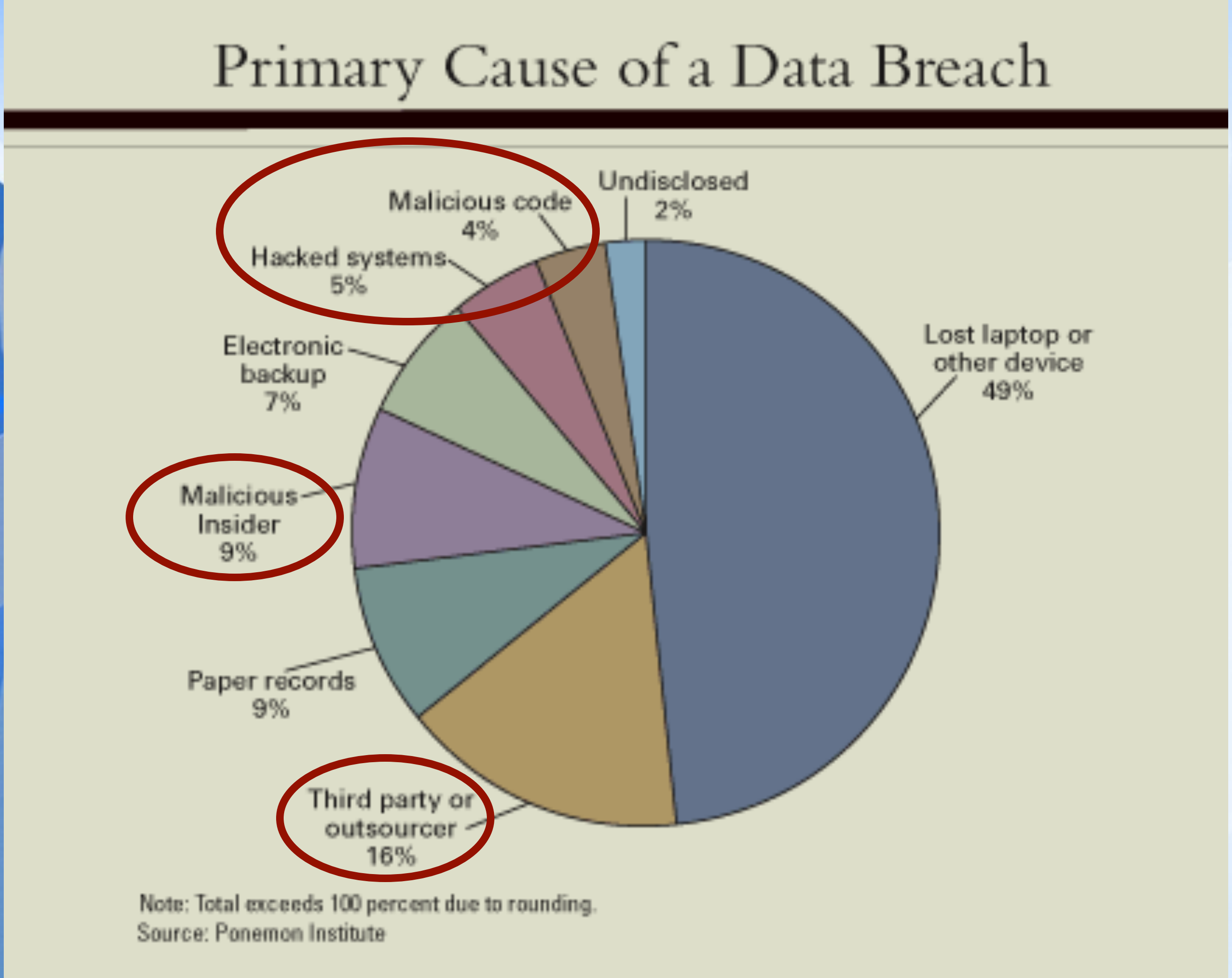
danfeng@cs.vt.edu

Overhead for preparing the Bloom filter (BF) and fingerprint filter (FF)



BF w/ SHA-1 is slightly faster to prepare than FF

Data breach, data leak, data exfiltration, data exportation



Primary Cause of a Data Breach

Note: Total exceeds 100 percent due to rounding.
Source: Ponemon Institute

2007 data from Wall Street Technology