Privacy

SBA Research &
Vienna University of Technology

Edgar R. Weippl

ANONYMOUS COMMUNICATION

Agenda

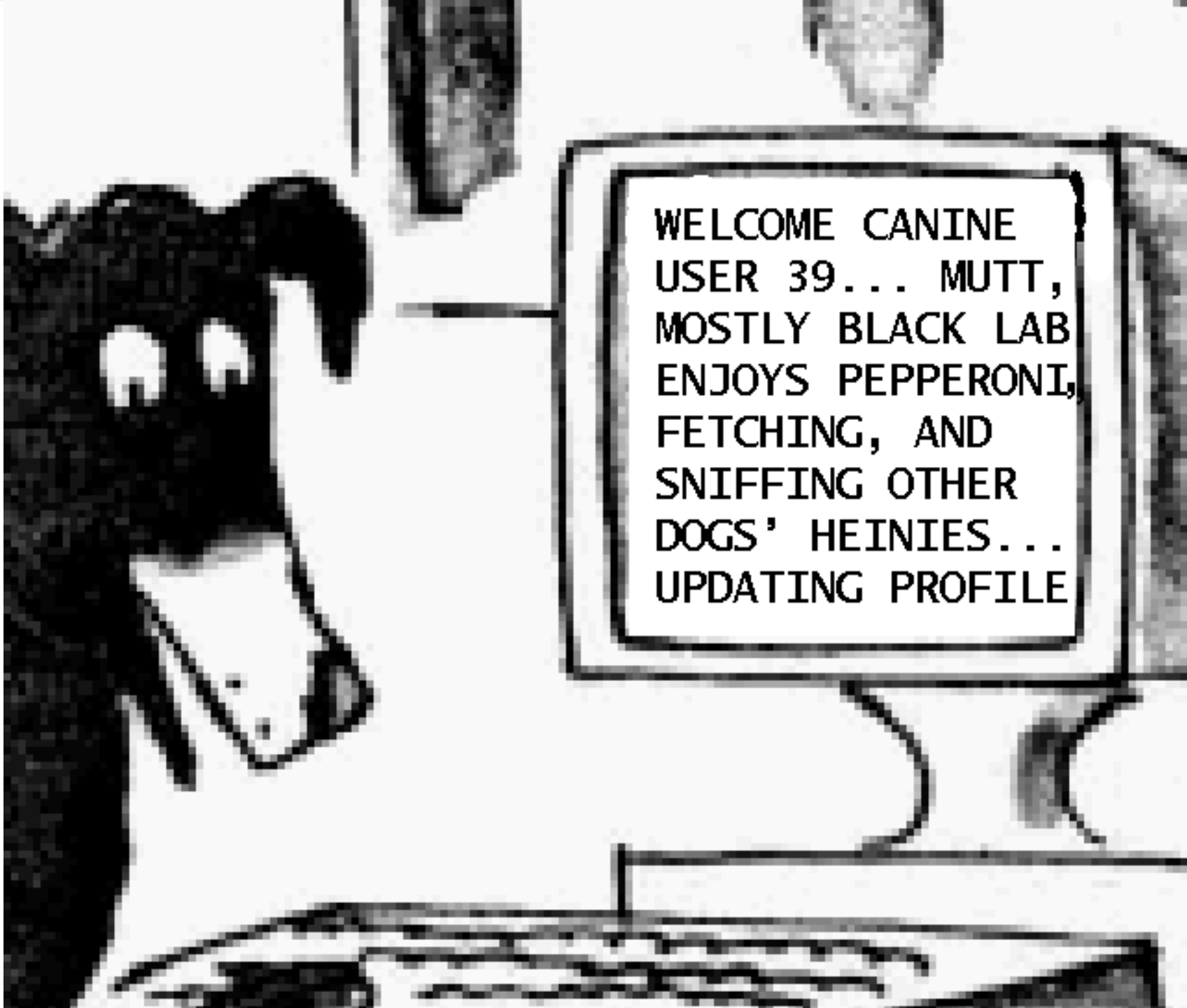
- Overview
- What is anonymity?
- China & Firewall
- Attacks on anonymizers
- Anonymity on the Internet:
  - Tor
  - JAP / JonDonym

Definition of Anonymity

- Greek for „without name“
- Means „non-identifiable, unrecognized“
- Relates to connection of people and events
- Examples?

"On the Internet, nobody knows you're a dog."

"On Facebook, 273 people know I'm a dog.
The rest can only see my limited profile."



Examples

- Alcoholics Anonymous®
- Traffic fines
- Elections
- Letters
- Phone booths
- Subway
- …

Anonymity

- Oponents
  - Law enf
  - Compar
  - Dictator
  - Intellige
  - …
- Hard to fi
  - Eg. Blizzards forums: posting only with real name - withdrawn
  - Eg. Data retention

CNET › News › Crave › Blizzard backs off real-name forum mandate

Blizzard backs off real-name forum mandate

Days after announcing a plan that would have required users to use their real first and last names on its online forums, WoW developer listens to user gripes and reverses its stance.

by Leslie Katz | July 9, 2010 6:25 PM PDT

Follow

Game developer Blizzard has backed down on a plan to require users of its official online

Anonymity

- „Data Retention“
  - EU directive2006/24/EG
  - Phone connections, IP addresses, IMSI/IMEI of cell phone, cell ID, ...
  - In Germany and Check Republic courts have ruled that the laws were against their constitution.
  - April 1, 2012: Austria implemented it using a central point for data clearance

Anonymity Set

- Anonymity always in regard to a group of individuals
- Larger sets are better

“Anonymity is the state of being not identifiable within a set of subjects, the anonymity set."

Different Forms of Anonymity

- Sender Anonymity
- Receiver Anonymity
- Anonymous Cash
- Unlinkability – for certain messages
- Unobservability – of communication
- …

Pseudonyms

- Greek for „named so in error“
- Cover names = Pseudonyms
- Examples?
  - Names of artists
  - Email addresses
  - Phone numbers
  - User name
  - Credit card numbers

Anonymity on the Internet

- Not easy
- Even more difficult to implement it correctly
- Demand exists without doubt
- Challenges:
  - Highly dynamic protocols (HTTP),
  - Interactivity
  - Local storage of cookies &other data (HTML 5)

Anonymous

Anonymity & Disclosure

Anonymity on the Internet

- „Invented“ by David Chaum
  David Chaum: ***“Untraceable electronic mail, return adresses, and digital pseudonyms“,***
  Communications of the ACM, 1981
- Basis for Onion Routing are **Mix Servers** or a Chaum Mix Cascade
- Requires Public Key Kryptographie (1976)
  - Diffie, Hellman: “New Directions in Cryptography”

Onion Routing

- Message is encrypted several times by the client
- Message is sent from server to server
- Each server removed its layer by decypting the message
- Last server sees content and final destination
- Developed by the US Navy

Anonymity and Protcols

- Chaum, 1988: *„**The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability**”*
- DC-Nets are an alternative to mix-networks and onion routing
- Dining Cryptographers are not *“Dining Philosophers”*!

Dining Cryptographers

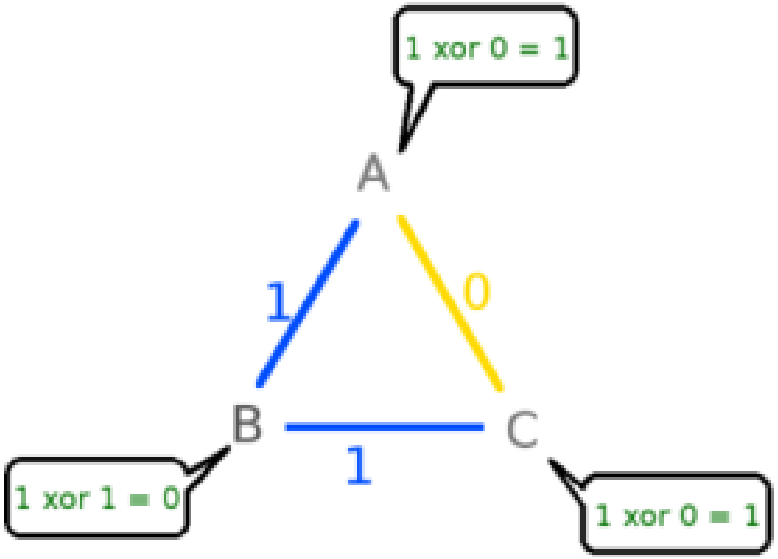
- 3 Cryptographers eat after work in the same restaurant
- They want to pay the check but it has already been paid
- Who paid? Was it one of them or NSA?
- Problem: They want to keep their individual privacy, respect the privacy of the other cryptographers but they want to know whether NSA paid.

Dining Cryptographers Protocol

- Each person throws a coin (1 or 0) together with her neighbors (once with each neighbor, 2 in total)
- She computes XOR of both results
- If she has paid the check, she inverts the result
- Every person discloses her result
- Every person calculated XOR of all results
  - If 0: one of the cryptographers picked up the bill
  - If 1: NSA paid the food

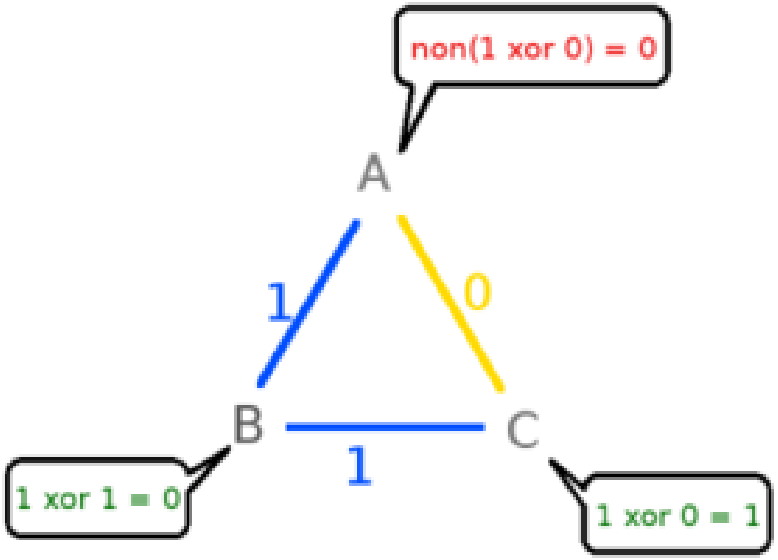
Dining Cryptographers Protocol

Non of them paid:



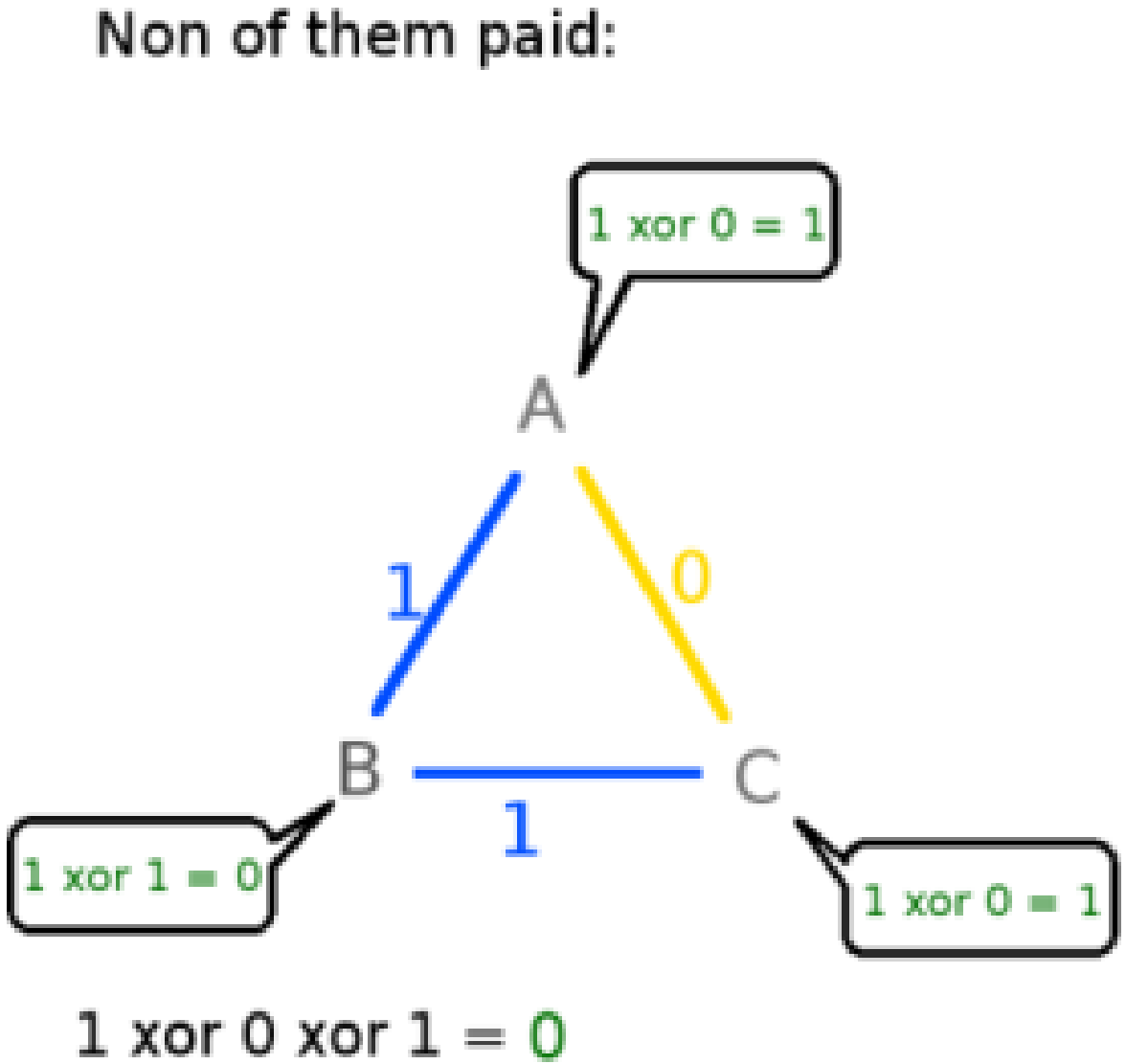
1 xor 0 xor 1 = 0

A paid:

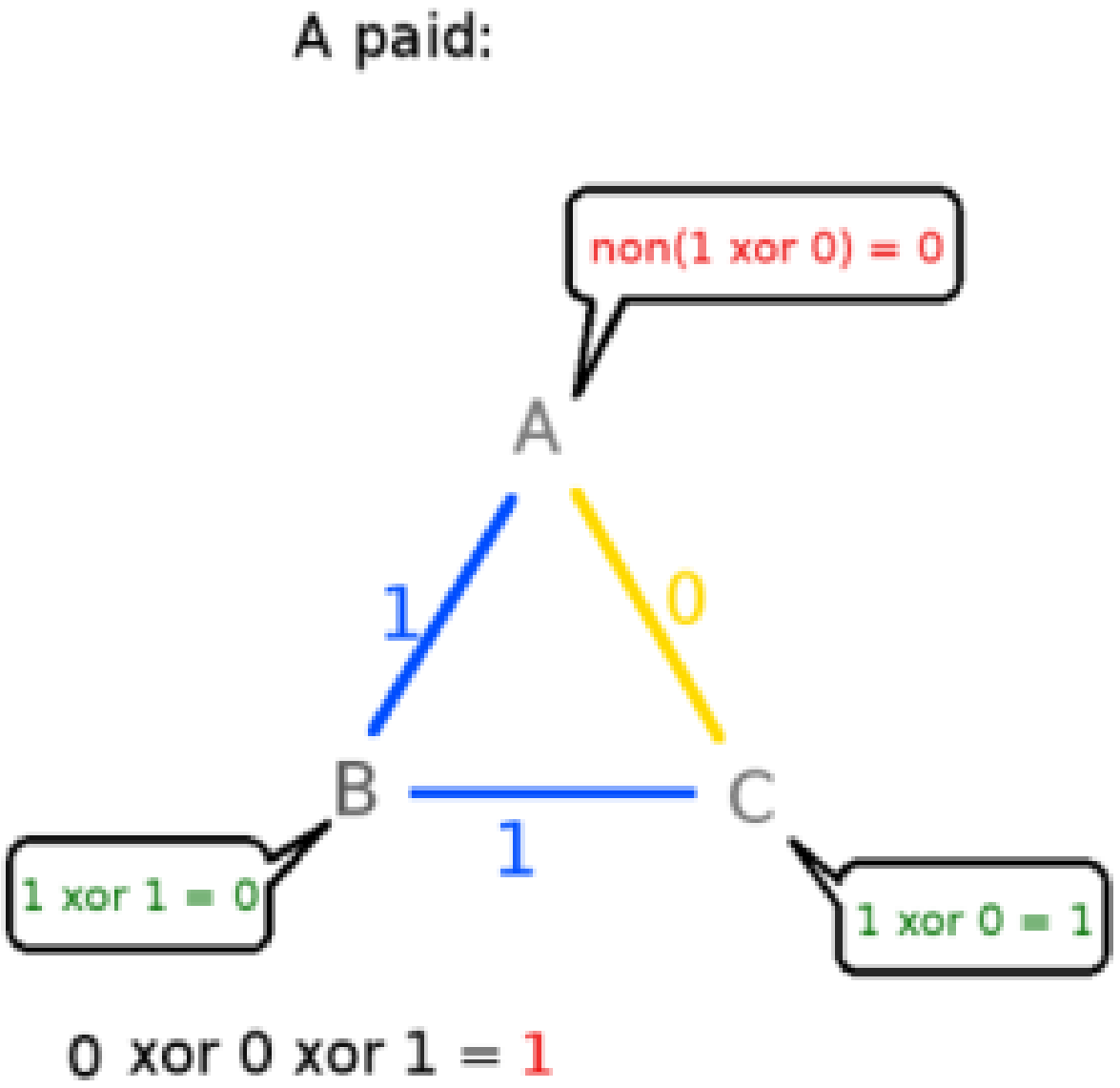


0 xor 0 xor 1 = 1

Dining Cryptographers Protocol



Dining Cryptographers Protocol



Differences to Onion Routing

- Multicast required
- Based on several rounds
- 2 steps of communication required
  - With both neighbors
  - Annonce the results
- Malicious sender can interfere with the protocol

Designproblem: Which Layer

- Application Layer (HTTP, DNS, SSH, FTP, …)
- Transport (TCP, UDP, …)
- Internet (IP, ICMP, …)
- Network (Ethernet, FDDI, …)

Flexibility vs Anonymity

- Flexibility:
  - Many different applications supported
  - This enables many attacks that reveal (partial) identities
- Anonymity:
  - For (one) specific application
  - Possibly not enough users for large anonymity sets

Service Quality

- Distinguish between
  - High Latency Anonymity Systems
  - Low Latency Anonymity Systems

High Latency

- First systems
- Messages (Emails) had no or only little time constraints
- Delay several hours
- Useful for Email and Usenet (one way communication)

High Latency

- First system: **anon.penet.fi**
- 1993, Johan Helsingius
- Pseudonymous remailer
- How did it work?
  - User sends email to the remailer
  - Remailer strips all information that could identify the sender
  - Remailer sends content to receiver
  - Remailer stores all (real) sender addresses so that it can forward answers from the receiver to the sender

anon.penet.fi

- Problem: Database with email addresses
- 1995 – Scientology:
  - Stolen files were sent to alt.religion.scientology („Miss Blood“ & Tom Klemesrud)
  - Scientology calls FBI & Interpol
  - Finnish Police gets acess to user data
  - 1st time: Emailadresse of„-AB-“ is revealed
  - 2nd time: no success, since alpha.c2.org Remailer was used
  - The entire story: „What really happened in INCOMM“

High Latency

- Additional weaknesses:
  - Single server – DoS
  - No encryption
  - Anonymity cannot be guaranteed
- 1996 Server was shut down

High Latency

Newer systems:

- Cypherpunk remailer, Early 1990
  - Uses PGP/GPG, User encrypts email and sends it to the remailer
  - Remailer decrypts and sends email on
  - Multiple layers are supported
- Mixmaster, Mid 1990
  - Last version in 2003
  - Prevents „Traffic Analysis“ by shuffling messages and having packets of identical size.

High Latency

- Mixminion, since 2003
  - Reordering messages („pooling“), sending batches of messages in bursts („batching“) & all packets have the same size
  - Mix connections are encrypted (TLS)
  - Permits replys in a a „Single-Use Reply Block“.
  - Keys are switched in intervals to prevent replay attacks
  - Mixes increase number of messages
  - However: only few users

High Latency

- A few hours of delay are unaccaptable for many applications.
- Therefore: Low Latency Anonymity Systems
  - Support interactive applications
  - Many different applications possible if protocols such as HTTP, IRC, SSH are supported
  - Goal: minimize delay

Low Latency

- Challenges:
  - Time cannot be used to increase anonymity
  - Support for different protocols
  - Support for many different types of users
  - Real time applications (VoIP) not (yet really) possible

Low Latency

- Best known architecture: Onion Routing
  - Tor best known implementation
  - Caveat: Onion Routing != Tor
- Also used: Mix Cascades
  - Fixed sequence of mixes
  - It is essential to used different operators in different jurisdictions
  - E.g. JAP or JohnDonym

China‘s Firewall

- Project „Golden Shield“
- Censorship and surveillance
  - „China‘s Intranet“
- Unwanted content is being blocked, e.g.
  - Falun Gong
  - Dalai Lama
  - Tian‘anmen square protests
  - …

China‘s Firewall

- How does it work – simple overview
  - IP blocks of static IPs (could be implemented with basic tools such as iptables)
  - DNS blocking – Name servers do not respond
  - Service filtering – modified software such as Google‘s search index

China‘s Firewall

- How does it work – simple overview
  - DNS redirection – Name servers return fake IP addresses
  - TCP Reset – „Deep Packet Inspection“, forged RST Packete terminate connection, e.g. URL filtering
  - Specialized software such as TomSkype, GreenDam

China‘s Firewall

- **TomSkype**
  - Chinese version of Skype, Joint-Venture, mid 2007
  - Specialized „feature“: Filter for IM messages; they are routed to a ‚special‘ server
  - Moreover, server was not well protected: hacked in 2008: „**Breaching Trust: An Analysis of Surveillance and Security Practices on China's TOM-Skype Platform”**
  - Both communication parties and content were saved
  - Follow-up paper: **Three Researchers, Five Conjectures: An Empirical Analysis of TOM-Skype Censorship and Surveillance**

China‘s Firewall

- „**GreenDam**“
  - Filter software for Windows, originally planned to be mandatory on Jul 1, 2009. Law was suspended on June 30, at least partially.
  - How does it work: DLL Injection bzw. API hooking im user land
  - Used ‚voluntarily‘ by schools and Internet Cafes
  - URL filter, text filter, image filter: recognizes skin tones
  - „Dam Burst“ (Jon Obereide, Defcon 18): Injected Dam Burst & deleted Green Dam without admin privileges

China‘s Firewall

- „**GreenDam**“ – Fun Facts:
  - Mid June 2009 2 Remote Code Execution Vulnerabilities were found
  - Image filter blocks pictures of pigs (skin tone similar to humans)
  - Blacklist of URLs probably copied from CyberSitter
  - Admin password ist stored as MD5 Hash in a text file with the extension .dll

China‘s Firewall

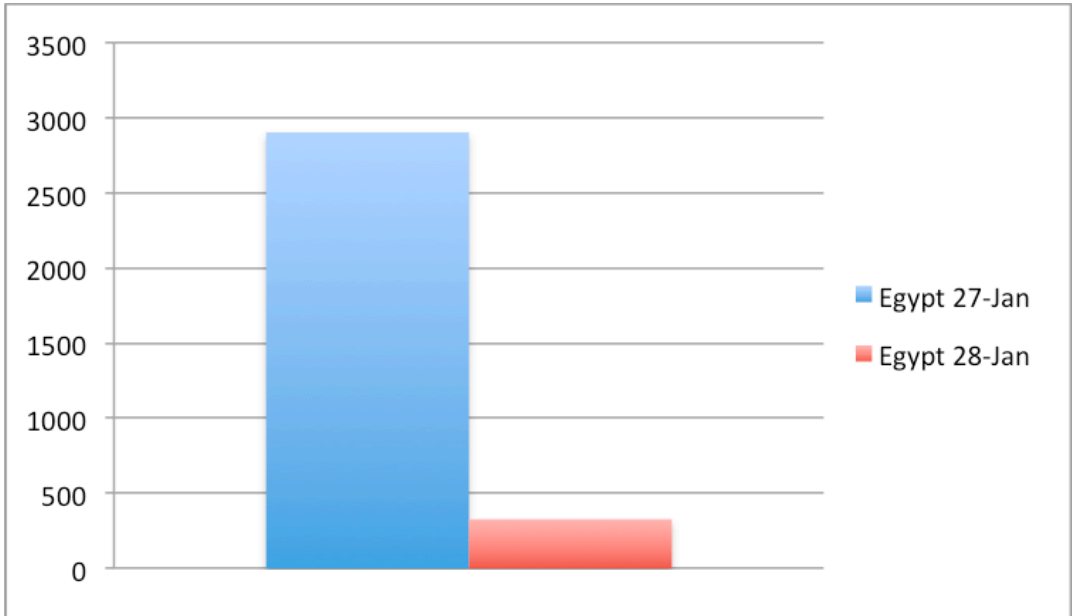
- How to cirumvent it:
  - Encryption (HTTPS, SSL, SSH, VPN, …)
  - Tor Bridges?!
  - Multimedia content
- China has 505 million Internet users (as of Jan2012)
  - = +50 % compared to US
  - approx. 1 billion cell phone subscribers!

China‘s Firewall

- China is only one example
- Many countries filter Internet access
  - Saudi Arabia, Iran, Vietnam, South Korea, Singapore, ...
  - RIM (Blackberry) in United Arab Emirates & India
  - But also Google in Germany or France
- OpenNet Initiative: http://opennet.net/

China‘s Firewall

- Egypt’s protests in Feb 2011: all BGP routings stopped (http://www.bgpmon.net/egypt-offline/).
- In Tunesia: JS-Injection
  - HTTPS Ports blocked, Facebook HTTP by default
  - Access credentials for Gmail, Yahoo, Facebook, …
  - Facebook groups, images deleted,



Attacks on Anonymity

- Technical: DDoS, bruteforcing of login credentials , …
- Crypto: weak algorithms
- Analytical: Intersection attack, partition attack
- Active / passive: modifying packets, eavesdropping
- Social: Social Engineering and more

Heavily depends on threat profile

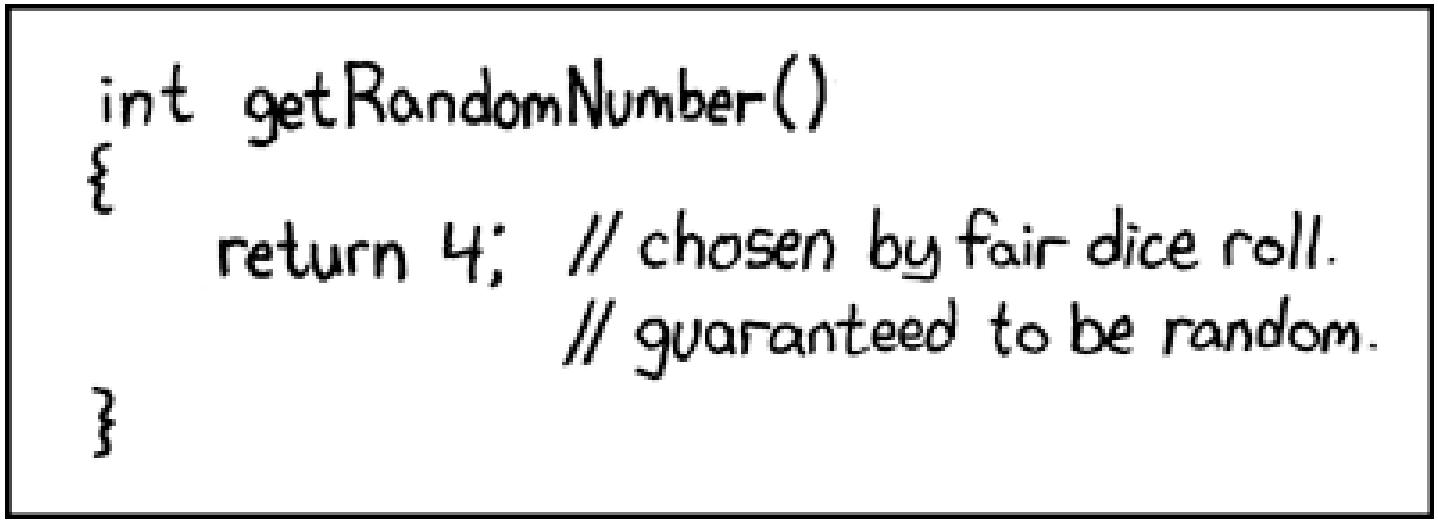
Attacks on Anonymity

- Example Penet Remailer:
  - Summer 1994, talk @ DEFCON cancelled
  - Has Penet been compromised?

http://www.citizendia.org/Penet_remailer

Attacks on Anonymity

- Example Debian Weak Keys:
  - OpenSSL Weakness im May 2008



Attacks on Anonymity

- Traffic Analysis:
  - Dates back to military tradition
  - Who talks to whom is essential, even if content is encrypted
  - Timeing, number of messages can reveal information

- Eg. “Low-Cost Traffic Analysis of Tor”, 2005, IEEE Security & Privacy
  - 3 tor nodes could be identified
  - In 2008 6 Directory Nodes, 3 running Debian!
  - Attack would need network consensus 4 of 6
  - CPU and networks load was analyzed
  - In 2005 Tor was much smaller than it is today

Attacks on Anonymity

- E.g. „Timing Analysis of Keystrokes and Timing Attacks on SSH”, 2001, USENIX Security Symposium
  - Keys that are next to each other are hit faster
  - The researcher claim a speed up of up to 50 times compared to brute force
- More attacks:
  - Long term Intersection Attack: Tor does not protect against a long term global passive attacker
  - Partition Attack: Trick the user into believing that all servers are offline except the ones controlled by the attacker

Tor – The Onion Router

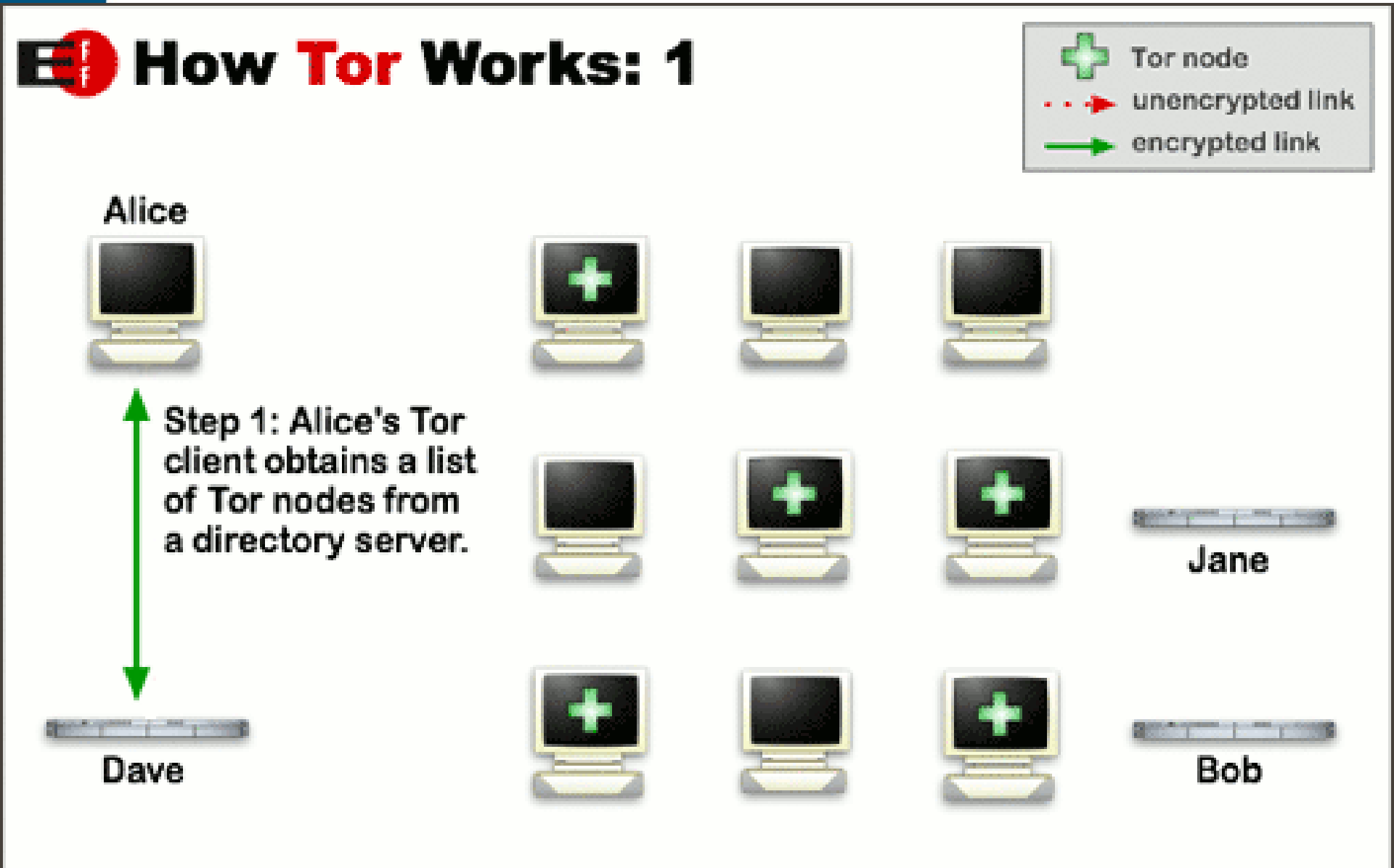
- Public since 2003
- Usenix Security: Dingledine, Mathewson & Syverson ***"Tor: The Second-Generation Onion Router",*** 2004
- Largest anonymity network with >100.000 users
- IP address disguised
- Based on onion routing
- Delay <1 sec (in the best case)

Tor – The Onion Router

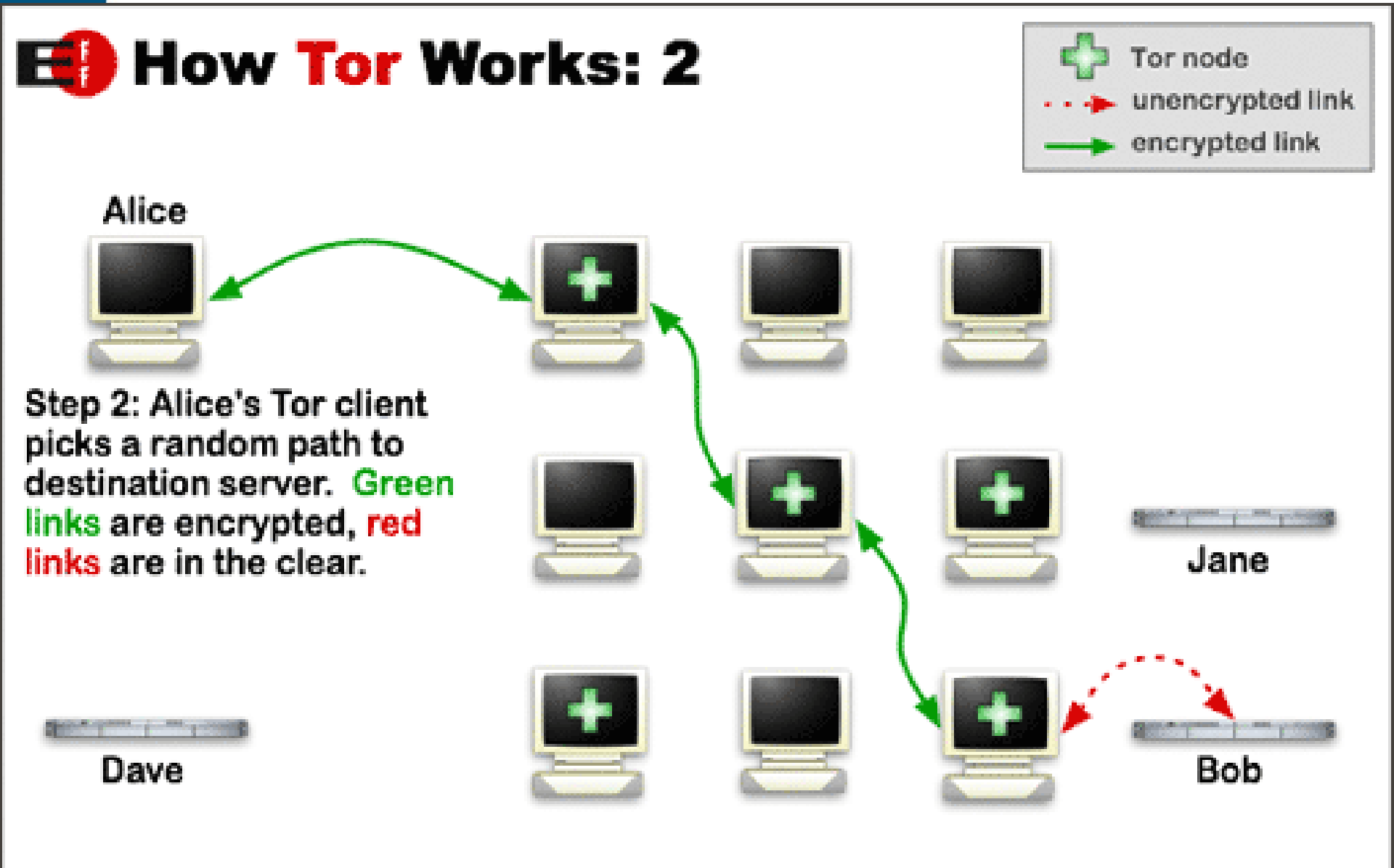
- Target group
  - Journalists, regime critics, …
  - Police, governments, …
  - companies, …
  - everybody!

  - And also criminals

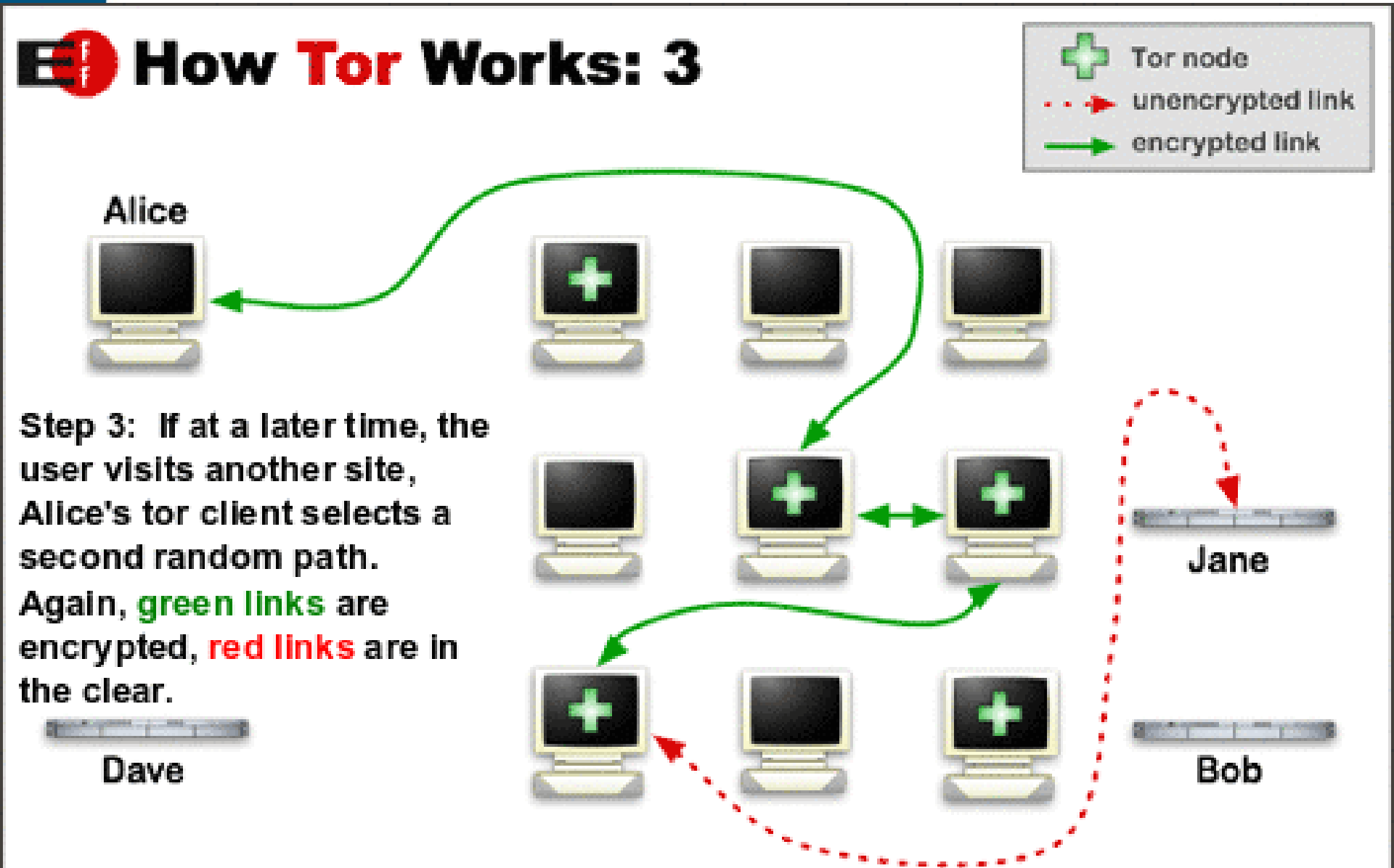
Tor – The Onion Router



Tor – The Onion Router



Tor – The Onion Router



Tor – The Onion Router

- Open source & all popular platforms
  - *nix, Windows, MacOS X
  - Android (Orbot)
  - OpenWRT (Torouter)
  - VM „Tails“ (https://tails.boum.org/)
- SOCKS Proxy Interface – supports many applications
- Can be installed without admin privileges
  - Tor Browser Bundle

Tor – The Onion Router

- Onions:
  - Tor anonymizes TCP Streams, not individual packets
  - Fixed cell size, 512 Byte
  - Cell is encrypted 3 time = 3 Tor Server = 1 path
  - Clients determines path (starting at the end)

- First Tor server sees origin IP address
- Last Tor server (Exit) sees content and target

Tor – The Onion Router

- Tor Directory Server
  - Globallv 8-9
    - THREAT LEVEL | hacks and cracks
    -
- T
    -
    - Dan Egerstad, the Swedish computer security consultant I interviewed in August who obtained log-in and password information for 1,000 e-mail
    - accounts belonging to foreign embassies, corporations and human rights organizations, had his house raided on Monday by Swedish officials, who took him in for questioning.

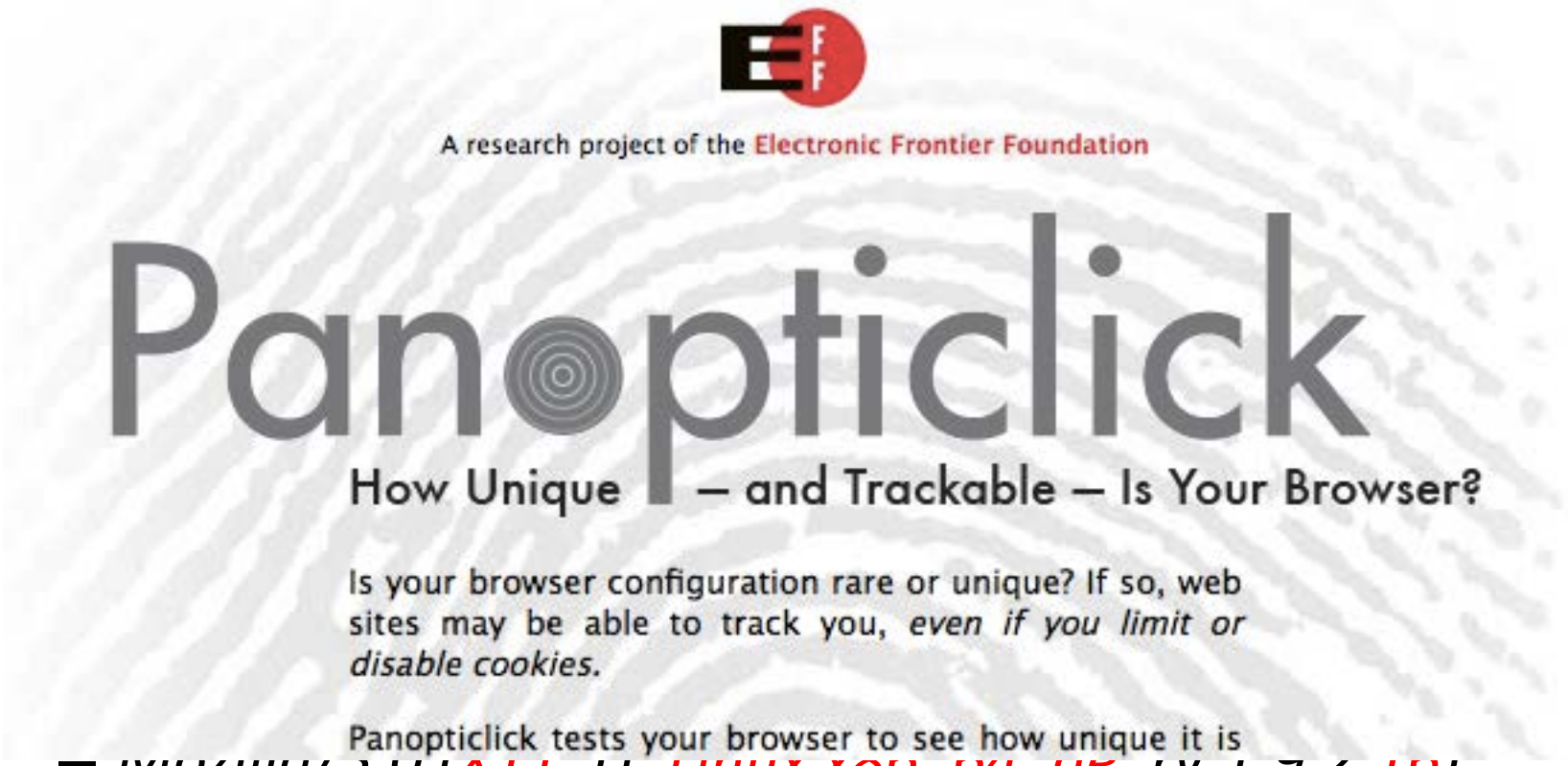
Tor Researcher Who Exposed Embassy E-mail Passwords Gets Raided by Swedish FBI and CIA

BY KIM ZETTER 11.14.07 4:13 PM

Tor – The Onion Router

- Tor Bridges
  - approx. 1000 available
  - Against IP based blocking
  - Bridge is at the beginning of each path
  - No public directory
  - Each client (theoretically) knows only few bridges
  - Tor Cloud: Bridges at EC2 https://cloud.torproject.org/

Tor – The Onion Router

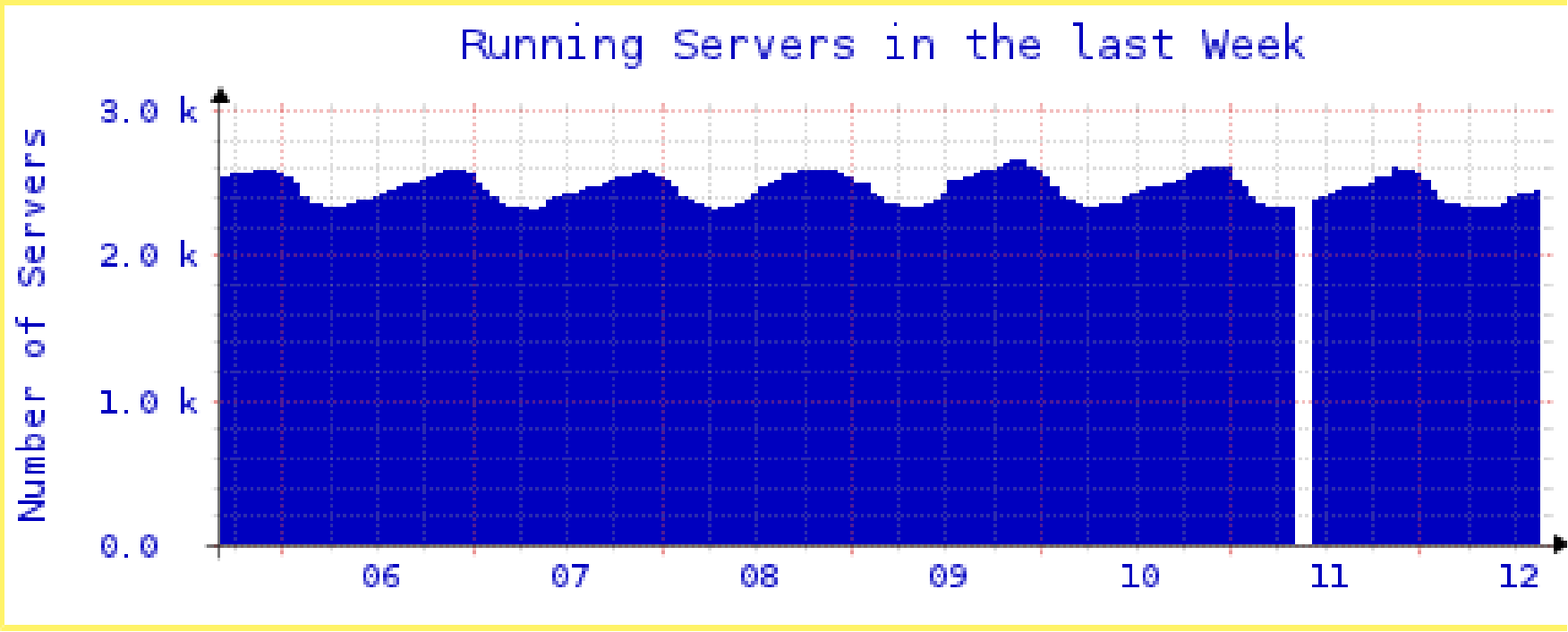
- 

- *Mozilla/5.0 (X11; U; Linux x86_64; de; rv:1.9.2.16) Gecko/20110323 Namoroka/3.6.16*

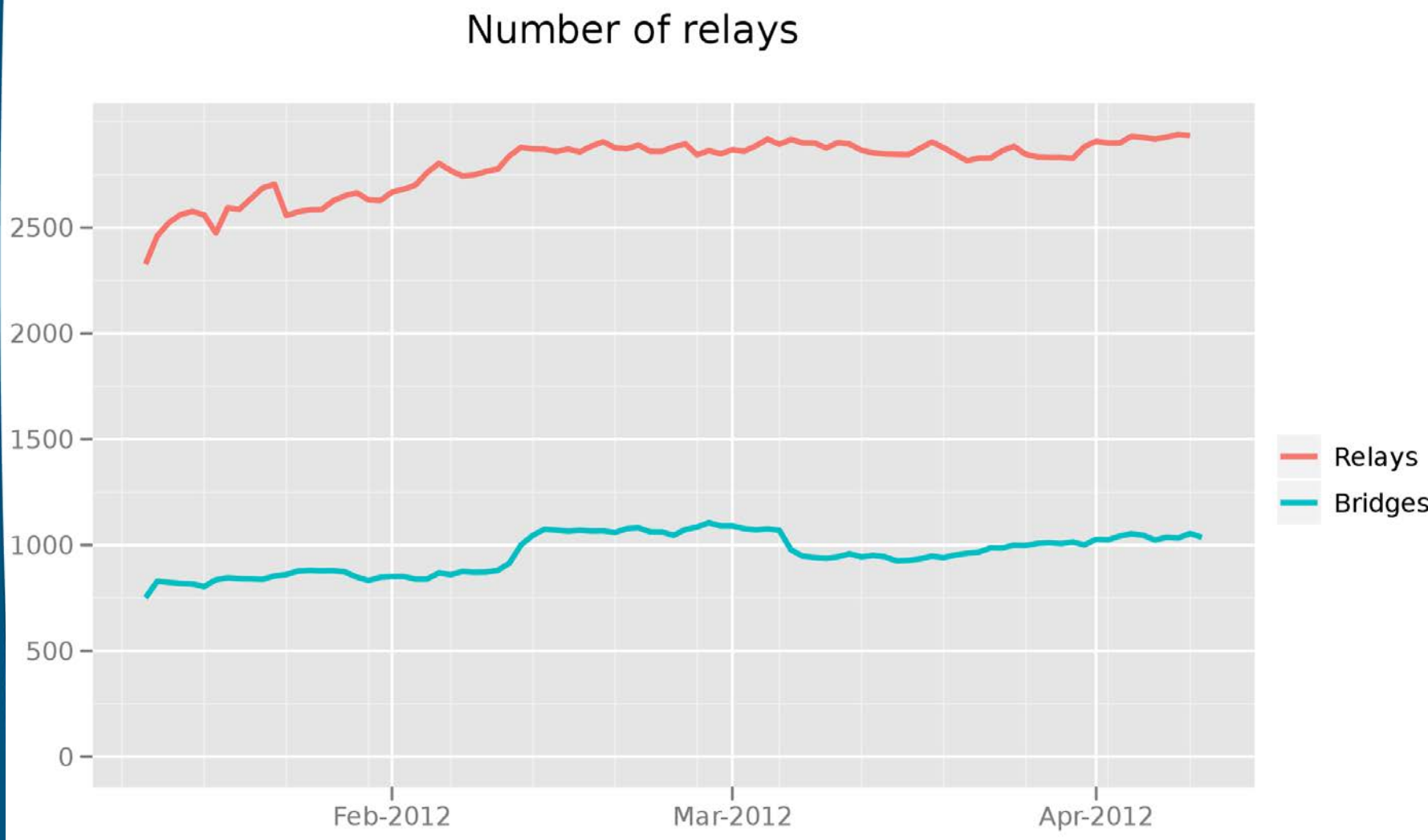
Tor – The Onion Router

- Additional software:
  - Privoxy: Local proxy, ad blocker
  - Vidalia: GUI for Tor
  - HTTPS Everywhere: FF Extension by the Electronic Frontier Foundation (EFF), HTTPS for many pages such as Facebook, Google, …
  - NoScript: FF Extension, blocks JavaScript
  - OnionCoffee: Alternate implementation of Tor protocol, in Java (outdated)
  - …

Tor – The Onion Router

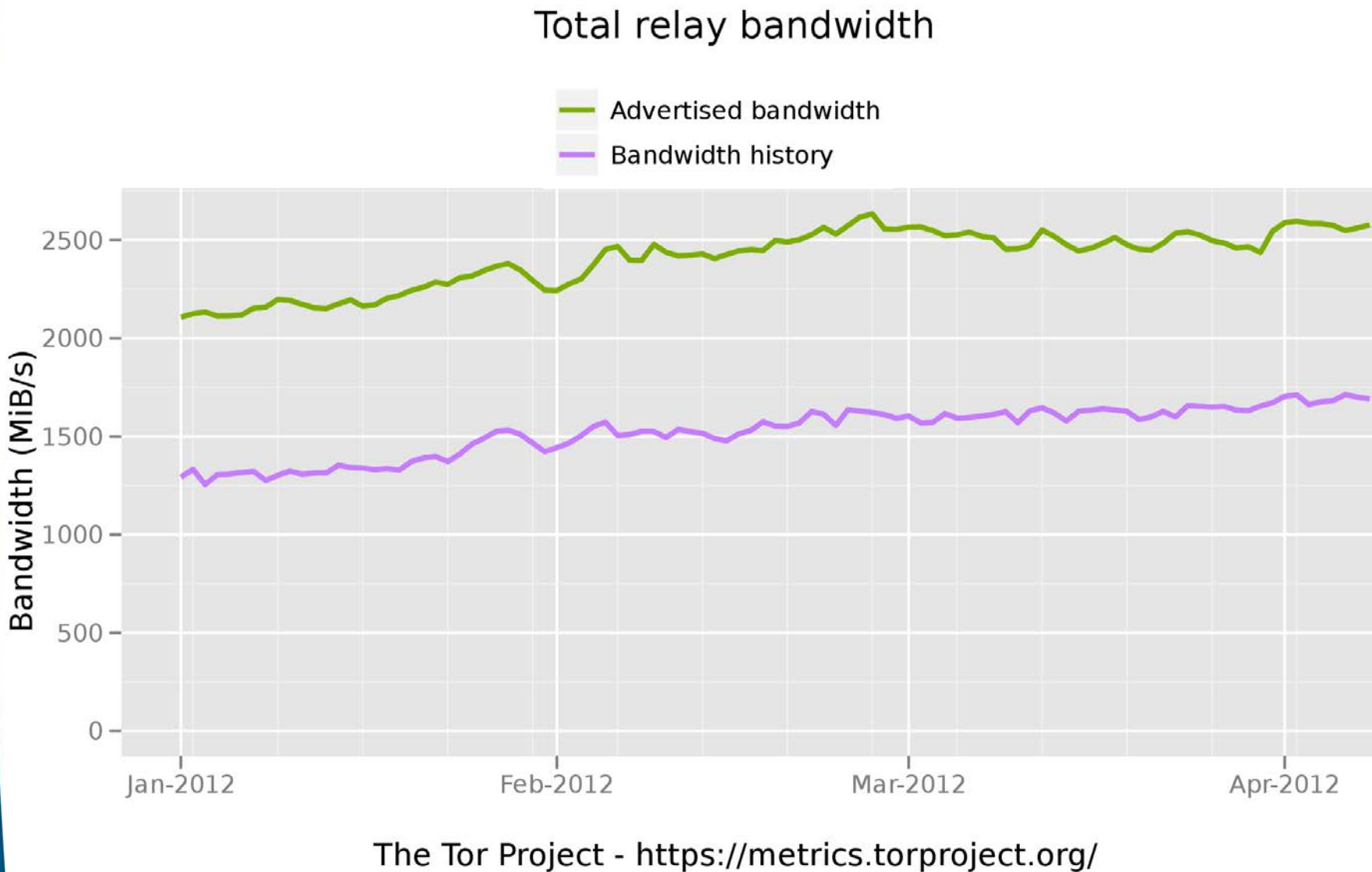


Tor – The Onion Router

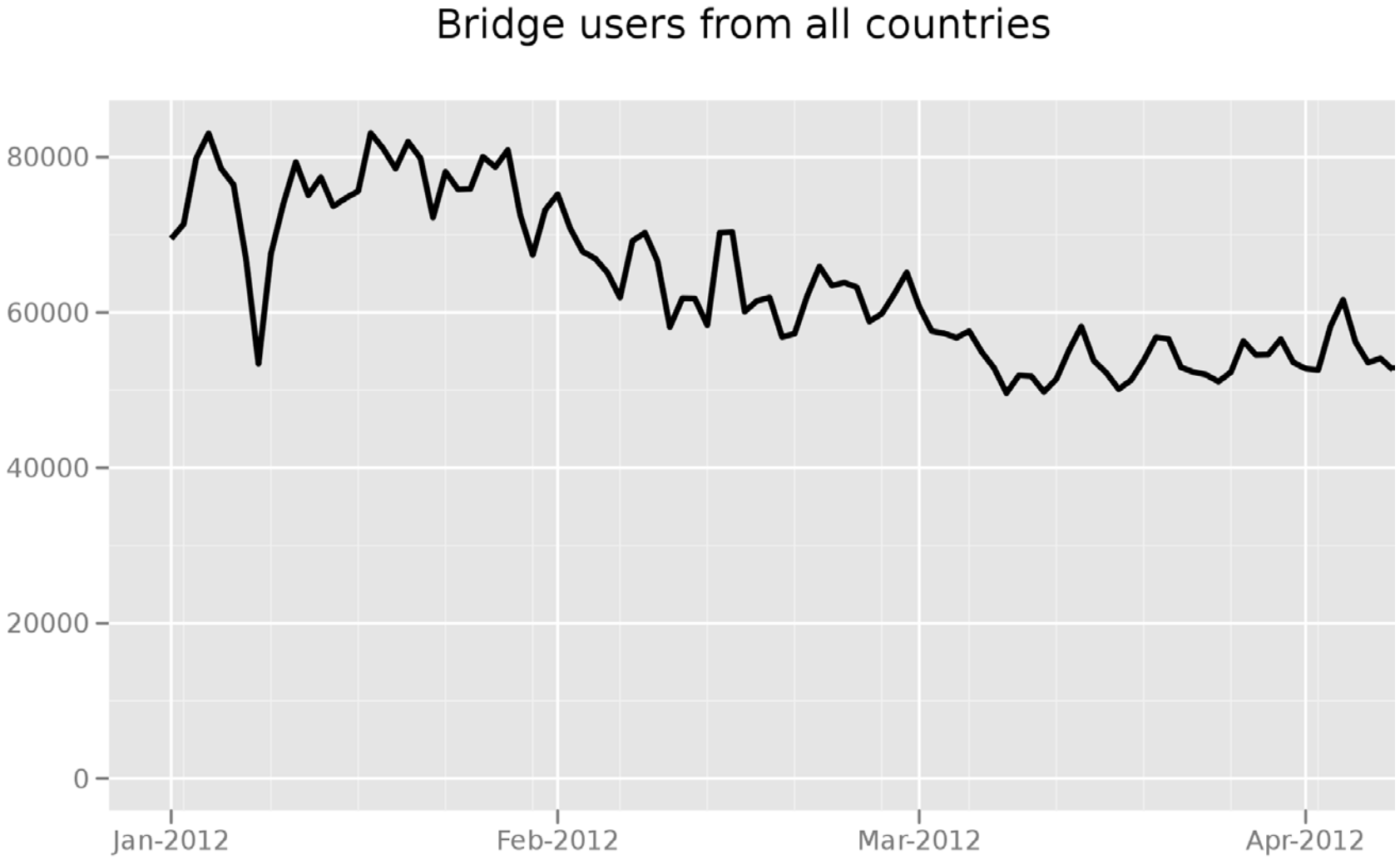


The Tor Project - https://metrics.torproject.org/

Tor – The Onion Router



Tor – The Onion Router



The Tor Project - https://metrics.torproject.org/

Tor – The Onion Router

- Cryptography:
  - TLS between Tor servers
  - Public Key Crypto: each server has 2 key pairs
  - Diffie Hellman key exchange
  - Symmetric encryption 128bit AES
  - Hash: SHA-1

Tor – The Onion Router

- Usage (as of 2008):

Protocol	Connections	Bytes	Destinations
HTTP	12,160,437 (92.45%)	411 GB (57.97%)	173,701 (46.01%)
SSL	534,666 (4.06%)	11 GB (1.55%)	7,247 (1.91%)
BitTorrent	438,395 (3.33%)	285 GB (40.20%)	194,675 (51.58%)
Instant Messaging	10,506 (0.08%)	735 MB (0.10%)	880 (0.23%)
E-Mail	7,611 (0.06%)	291 MB (0.04%)	389 (0.10%)
FTP	1,338 (0.01%)	792 MB (0.11%)	395 (0.10%)
Telnet	1,045 (0.01%)	110 MB (0.02%)	162 (0.04%)
Total	13,154,115	709 GB	377,449

- PETs 2008 ***„Shining Light Into Dark Places: Understanding the Tor Network“***

Tor – The Onion Router

- Bittorrent & Tor:
  - Bad idea
    1. Too much traffic
    2. Bittorrent can lead to de-anonymization
  - Both *Tracker-Only* and *Tracker+Content* viaTor
  - Put together this can be used to de-anonymize HTTPS streams
  - *„One Bad Apple Spoils the Bunch: Exploiting P2P Applications to Trace and Profile Tor Users“*, USENIX LEET 2011

Tor – The Onion Router

- Hidden Services:
  - Servers without revealing IP address
  - Double anonymity (Rendezvous Point = 2 times Tor)
  - Address: 16characters.onion (=LSB 80 bit of Public Key)
  - Eg. http://sx3jvhfgzhw44p3x.onion

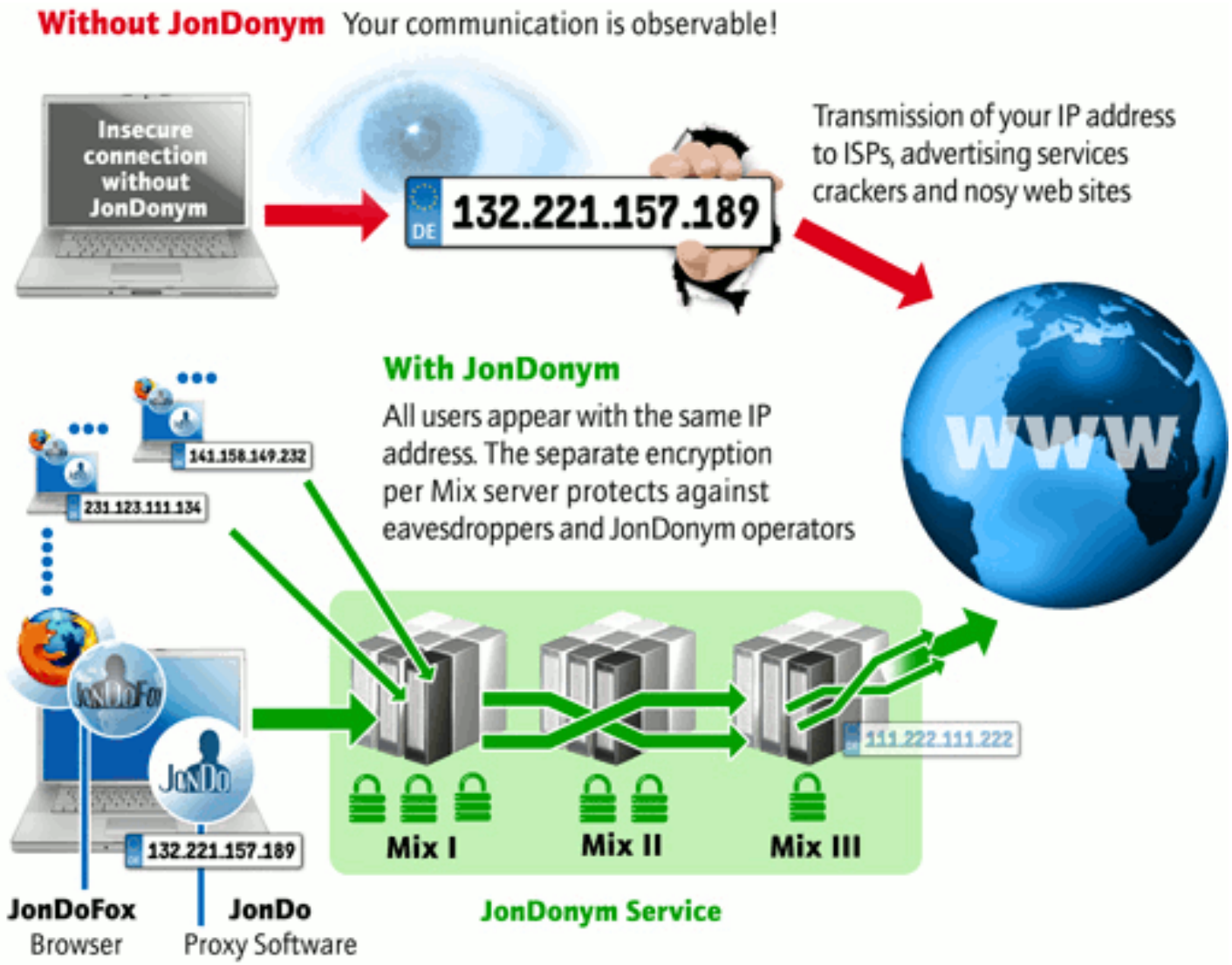
Tor – The Onion Router

- However: TCP only
  - OnionCat: Tunnels IP over Tor (also UDP)
  - CORE: Connectionless Onion Routing, each packet uses 3 different relays, many retransmissions
  - *"Anonymity Loves Company: Usability and the Network Effect"* 2006: Tor Design Decisions (Network consensus, Bootstrapping, fewer options for clients, …)
- Other issues:
  - Too slow
  - Bridges are sometimes blocked

JAP - JonDonym

- „Java Anon Proxy“ – now „JonDonym“
- free
- Premium paid service(> 64 kb)
- Uses fixed Mix cascades with 3 servers (about 10 with premium)

JAP - JonDonym



More Anonymizers

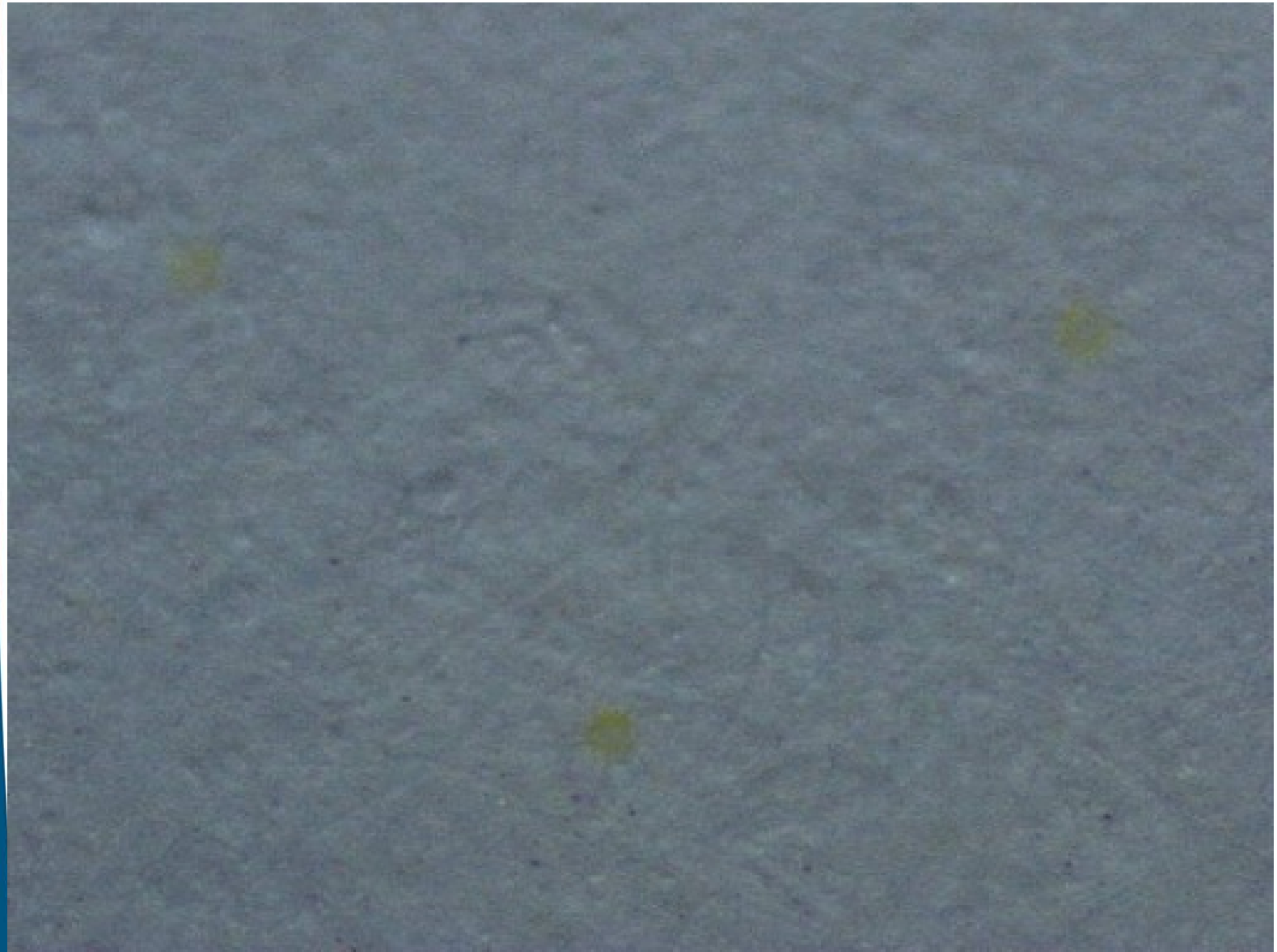
- I2P
- Freenet (uses distributed hash tables)
- Phantom protocol (https://code.google.com/p/phantom/)
- CliqueNet / Herbivore (2003, DC Net)
- Tarzan, Babel, Crowds, Infranet, Morphmix
- …

Steganography

- Example laser printer & copy machines– Machine Identification Code
  - Very small yellow dots
  - Identify manufacturer and model
  - EFF, 2005

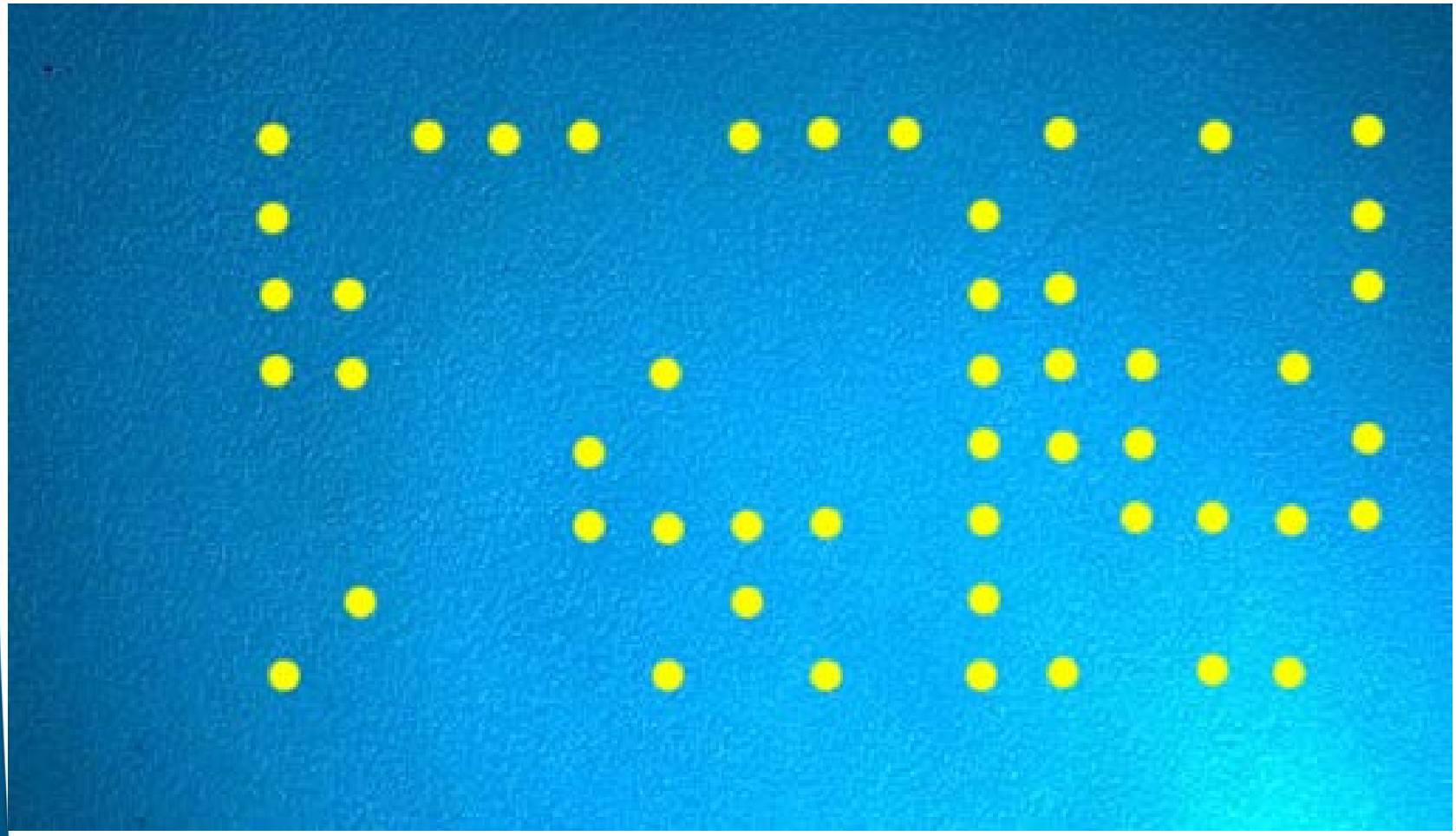
Yellow Dots – 10x

Yellow Dots – 60x

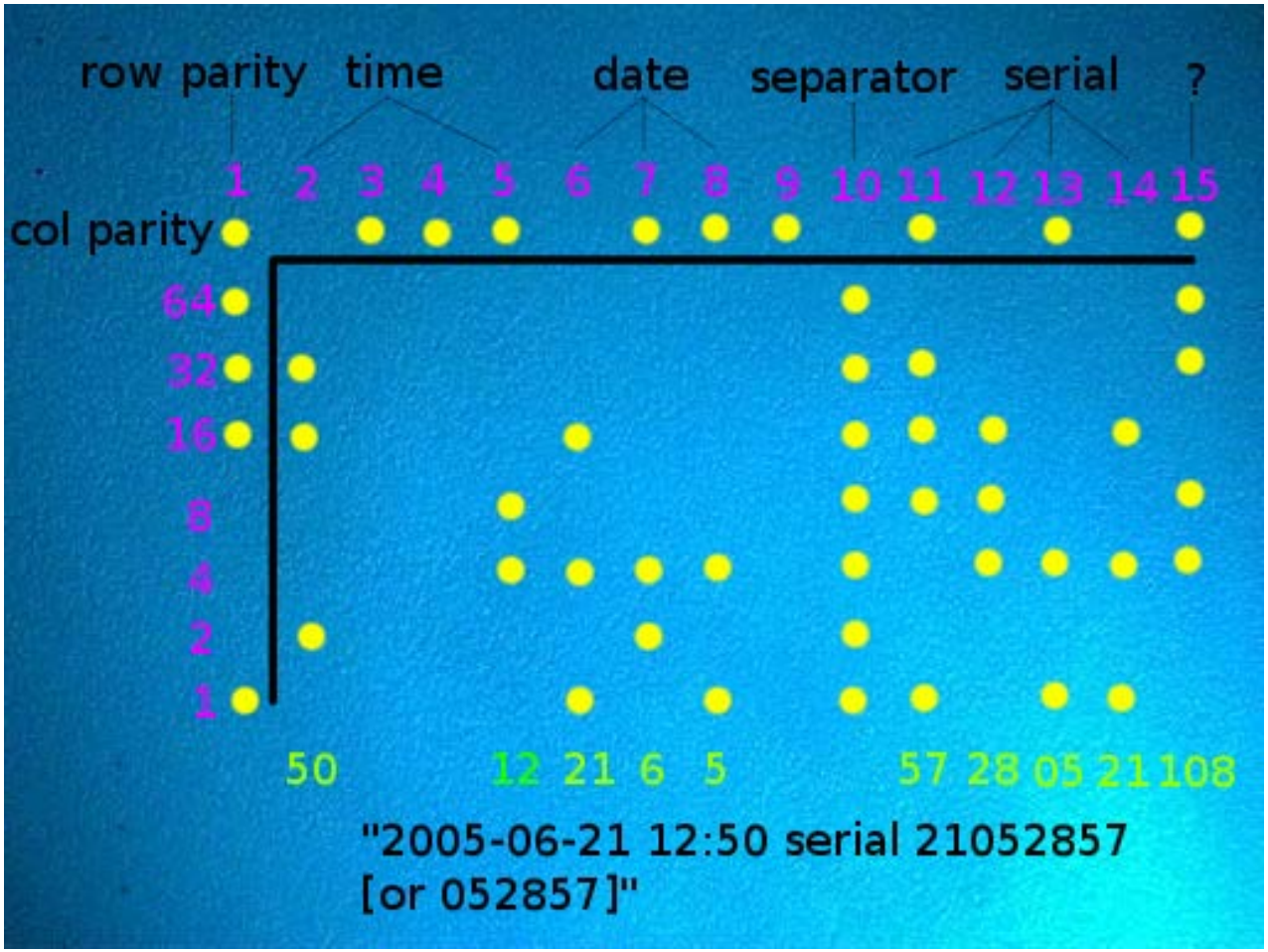


Yellow Dots – 10x blue LEDs

Yellow Dots – 10x blue LEDs



Yellow Dots – 10x blue LEDs



Yellow Dots

- Time of printing, serial number of printer
- Example Xerox DocuColor
  - ca. 10 Bytes information
  - On each print out
  - Also Canon, Epson, HP, …

Other ways of identifying printers

- Usage of “cheap” components with individual characteristics
  - Paper feed
  - Print
  - Similar to type writer forensics

PRIVACY

Well known concepts

- K-anonymity
- L-diversity

Linking data

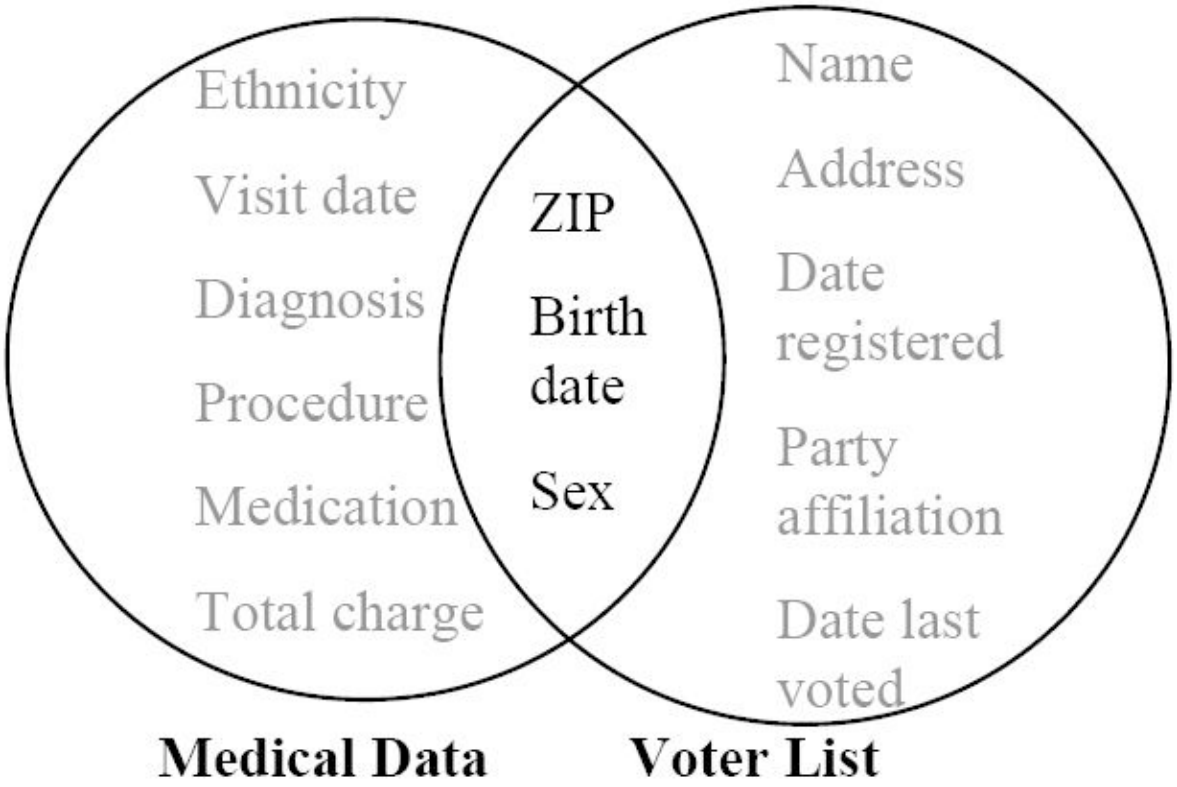


Figure 1 Linking to re-identify data

4-anonymous

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	13053	28	Russian	Heart Disease
2	13068	29	American	Heart Disease
3	13068	21	Japanese	Viral Infection
4	13053	23	American	Viral Infection
5	14853	50	Indian	Cancer
6	14853	55	Russian	Heart Disease
7	14850	47	American	Viral Infection
8	14850	49	American	Viral Infection
9	13053	31	American	Cancer
10	13053	37	Indian	Cancer
11	13068	36	Japanese	Cancer
12	13068	35	American	Cancer

Figure 1. Inpatient Microdata

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	*	Heart Disease
2	130**	< 30	*	Heart Disease
3	130**	< 30	*	Viral Infection
4	130**	< 30	*	Viral Infection
5	1485*	≥ 40	*	Cancer
6	1485*	≥ 40	*	Heart Disease
7	1485*	≥ 40	*	Viral Infection
8	1485*	≥ 40	*	Viral Infection
9	130**	3*	*	Cancer
10	130**	3*	*	Cancer
11	130**	3*	*	Cancer
12	130**	3*	*	Cancer

Figure 2. 4-anonymous Inpatient Microdata

4-anonymous

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	*	Heart Disease
2	130**	< 30	*	Heart Disease
3	130**	< 30	*	Viral Infection
4	130**	< 30	*	Viral Infection
5	1485*	≥ 40	*	Cancer
6	1485*	≥ 40	*	Heart Disease
7	1485*	≥ 40	*	Viral Infection
8	1485*	≥ 40	*	Viral Infection
9	130**	3*	*	Cancer
10	130**	3*	*	Cancer
11	130**	3*	*	Cancer
12	130**	3*	*	Cancer

L-Diversity

Similarity Attack

Bob	
Zip	***Age***
47678	27

A 3-diverse patient table

Zipcode	Age	Salary	Disease
476**	2*	20K	Gastric Ulcer
476**	2*	30K	Gastritis
476**	2*	40K	Stomach Cancer
4790*	≥40	50K	Gastritis
4790*	≥40	100K	Flu
4790*	≥40	70K	Bronchitis
476**	3*	60K	Bronchitis
476**	3*	80K	Pneumonia
476**	3*	90K	Stomach Cancer

T-Closeness

“We propose a novel privacy notion called t-closeness, which requires that the distribution of a sensitive attribute in any equivalence class is close to the distribution of the attribute in the overall ta- ble (i.e., the distance between the two distributions should be no more than a threshold t).”

Ninghui, Tiancheng Li, and Suresh Venkatasubramanian. "t-closeness: Privacy beyond k-anonymity and l-diversity." *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE, 2007.

T-Closeness

	ZIP Code	Age	Salary	Disease
1	47677	29	3K	gastric ulcer
2	47602	22	4K	gastritis
3	47678	27	5K	stomach cancer
4	47905	43	6K	gastritis
5	47909	52	11K	flu
6	47906	47	8K	bronchitis
7	47605	30	7K	bronchitis
8	47673	36	9K	pneumonia
9	47607	32	10K	stomach cancer

Table 3. Original Salary/Disease Table

	ZIP Code	Age	Salary	Disease
1	476**	2*	3K	gastric ulcer
2	476**	2*	4K	gastritis
3	476**	2*	5K	stomach cancer
4	4790*	≥ 40	6K	gastritis
5	4790*	≥ 40	11K	flu
6	4790*	≥ 40	8K	bronchitis
7	476**	3*	7K	bronchitis
8	476**	3*	9K	pneumonia
9	476**	3*	10K	stomach cancer

Table 4. A 3-diverse version of Table 3

T-Closeness

	ZIP Code	Age	Salary	Disease
1	4767*	≤ 40	3K	gastric ulcer
3	4767*	≤ 40	5K	stomach cancer
8	4767*	≤ 40	9K	pneumonia
4	4790*	≥ 40	6K	gastritis
5	4790*	≥ 40	11K	flu
6	4790*	≥ 40	8K	bronchitis
2	4760*	≤ 40	4K	gastritis
7	4760*	≤ 40	7K	bronchitis
9	4760*	≤ 40	10K	stomach cancer

Table 5. Table that has 0.167-closeness w.r.t. Salary and 0.278-closeness w.r.t. Disease

Privacy – won't work

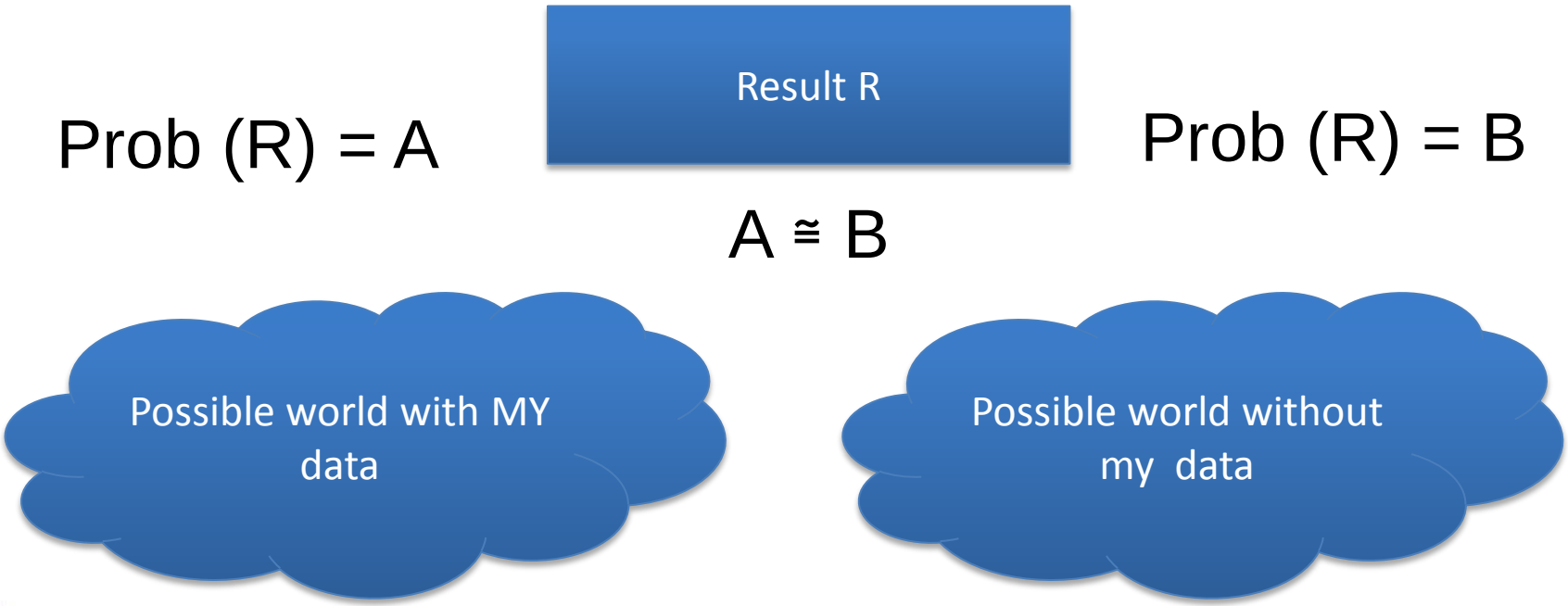
- My data should not have an impact on the results released.
- One should learn nothing about me.

- Ad 1) then the results have no utility.
- Ad 2) even if you do share your data, the trend is true for you.

Privacy that might work

Weaker assumption: Differential Privacy

Given result R can anyone guess which possible world it came from?



Looking for terrorists

- Let's assume we have an analysis tool with 98 % sensitivity and 99% specificity.

- Our software flags a person as a possible terrorists. Heavily armed you enter the apartment. How likely is that you really have a terrorist looking into barrel of your gun?

a. 98 % (sensitivity)?
b. 99 % (specificity)?
c. None of the above?

~ 100 / 5.000.000 … ~ 1 / 50.000 …

Statistics refresher

500 Mio people

100 terrorists

499 999 900 innocent people

2 terrorists are not found

98 terrorists are found

Approx. 5 Mio innocent people are interrogated by police…

They were lucky

This will never happen…

- Use of suspicious terms such as „Gentrifizierung“

- http://www.tagesspiegel.de/berlin/stadtsoziologe-andrej-holm-man-weiss-jetzt-was-gentrifizierung-ist/1826246.html

Java Script Fingerprinting

- Browser Fingerprinting:
  - Accurately identify the browser used by the client
  - Webserver point-of-view
  - Motivated by nmap for TCP/IP fingerprinting
  - Limitations of UserAgent string:
    - Can be set arbitrarily
    - Not a security feature
- Different use cases:
  - Detect User Agent string manipulations
  - Detect session hijacking
  - Browser-specific malware

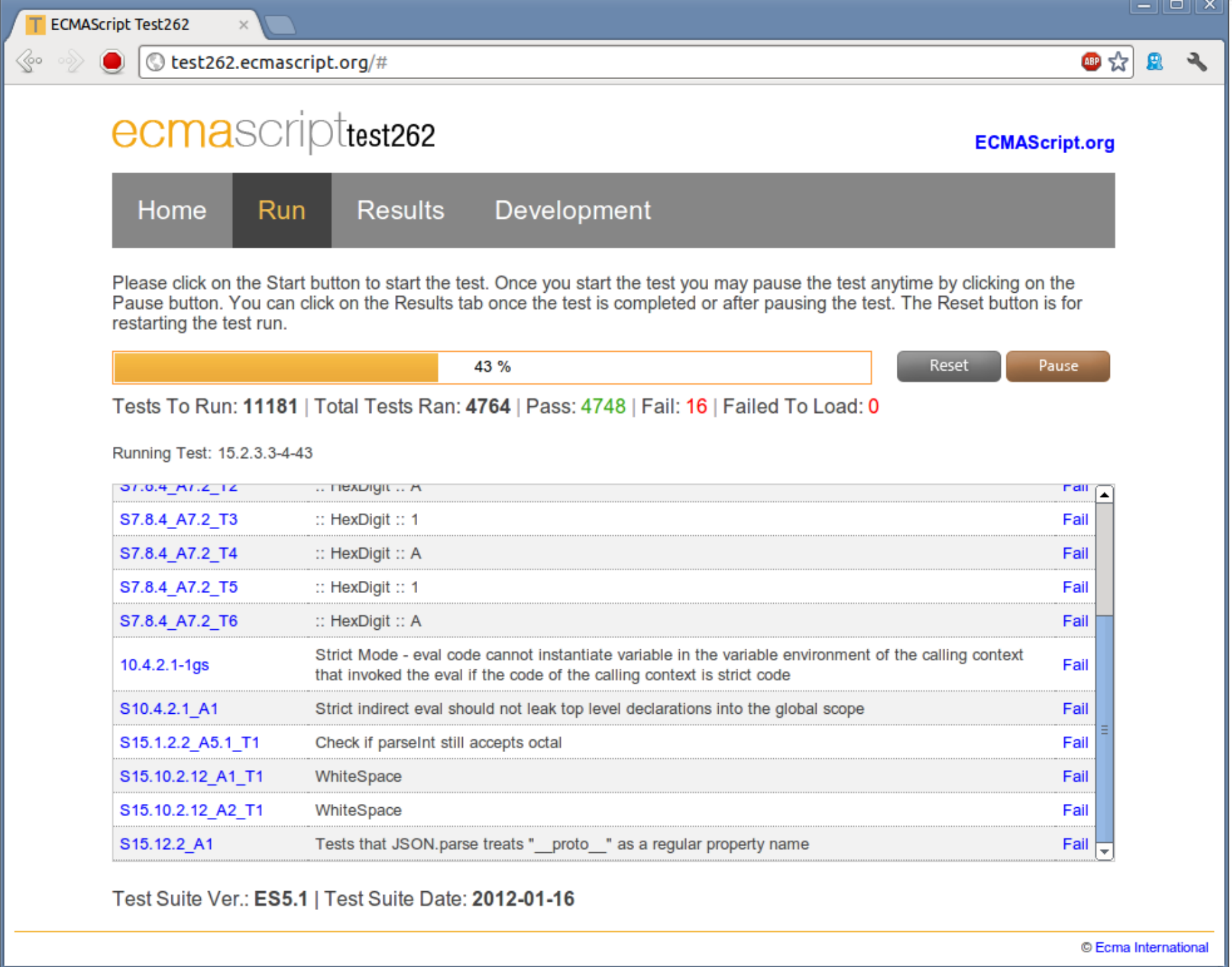
- Browser market currently very competitive:
- Man-years of development time
- Fight for market shares, especially smartphones
- Become more & more powerful (e.g., Cloud computing, HTML5, ...)
- New features:
  - JIT, GPU rendering, remote rendering, Sandboxing
  - Mostly performance or security

Java Script Fingerprinting

- Our approach:
  - Use JavaScript (ECMAScript 5.1) conformance tests
  - test262 - http://test262.ecmascript.org
  - Sputnik - http://sputnik.googlelabs.com
  - More than 11.000 test cases
  - Javascript engines fail at different test cases
- In the future:
  - Enhance session security by locking session to specific browser version
  - Increase user privacy by detecting (attacking) fingerprinting

Java Script Fingerprinting

- Recent paper by Mowery et.al, W2SP 2011
  - Use 39 Javascript benchmarks e.g., Sunspider
  - Generate normalized fingerprint based on time pattern
  - On average 190 seconds runtime
- Our approach:
  - Takes less then 100ms (3 orders of magnitude faster)
  - Few hundred lines of Javascript max.
  - Collected > 150 OS and browser combinations


ECMAScript Test262
test262.ecmascript.org/#
ecmascripttest262
ECMAScript.org
Home
Run
Results
Development
Please click on the Start button to start the test. Once you start the test you may pause the test anytime by clicking on the Pause button. You can click on the Results tab once the test is completed or after pausing the test. The Reset button is for restarting the test run.
43 %
Reset
Pause
Tests To Run: 11181 | Total Tests Ran: 4764 | Pass: 4748 | Fail: 16 | Failed To Load: 0
Running Test: 15.2.3.3-4-43
S7.8.4_A7.2_T3 :: HexDigit :: 1 Fail
S7.8.4_A7.2_T4 :: HexDigit :: A Fail
S7.8.4_A7.2_T5 :: HexDigit :: 1 Fail
S7.8.4_A7.2_T6 :: HexDigit :: A Fail
10.4.2.1-1gs Strict Mode - eval code cannot instantiate variable in the variable environment of the calling context that invoked the eval if the code of the calling context is strict code Fail
S10.4.2.1_A1 Strict indirect eval should not leak top level declarations into the global scope Fail
S15.1.2.2_A5.1_T1 Check if parseInt still accepts octal Fail
S15.10.2.12_A1_T1 WhiteSpace Fail
S15.10.2.12_A2_T1 WhiteSpace Fail
S15.12.2_A1 Tests that JSON.parse treats "__proto__" as a regular property name Fail
Test Suite Ver.: ES5.1 | Test Suite Date: 2012-01-16
© Ecma International

test262: Browser - OS Combinations

Browser	Win7	WinXP	MacOS		Browser	Win7	WinXP	MacOS
Chrome 7	1022	1022	1022		Firefox 3.5.19	3959	3959	3959
Chrome 8	1022	1022	1022		Firefox 3.6.20	3955	3955	3955
Chrome 9	1020	1020	1020		Firefox 3.6.26	3955	3955	3955
Chrome 10	715	715	715		Firefox 4	290	290	290
Chrome 11	489	489	489		Firefox 5	264	264	264
Chrome 12	449	449	—		Firefox 6	214	214	214
Chrome 13	427	427	—		Firefox 7	190	190	190
Chrome 14	430	430	430		Firefox 8	167	167	167
Chrome 15	421	421	—		Firefox 9	167	167	167
Chrome 16	420	420	420		Firefox 10	163	163	163
Chrome 17	210	210	210					
Chrome 18 beta	35	35	35		IE 6 (Sputnik)	—	468	—
					IE 7 (Sputnik)	—	472	—
Opera 10.54	3834	3834	3834		IE 8 (Sputnik)	—	473	—
Opera 10.63	3834	3834	3834		IE 9	394	—	—
Opera 11.01	3828	3828	3828					
Opera 11.11	3828	3828	3828		Safari 4.0.3	—	—	4074
Opera 11.51	3827	3827	3827		Safari 5.0.5	777	1585	1513
Opera 11.52	3827	3827	3827		Safari 5.1	777	853	—
Opera 11.61	4	4	4		Safari 5.1.2	777	777	776

Distinguish Current Browsers

Web Browser	15.4.4.4-5-c-i-1	13.0-13-s
Opera 11.61	✓	✗
Firefox 10.0.1	✓	✗
Internet Explorer 9	✗	✓
Chrome 17	✗	✗

Web Browser	S15.2.3.6_A1	10.6-7-1	S10.4.2.1_A1
Opera 11.61	✗	✗	✗
Firefox 10.0.1	✗	✓	✗
Internet Explorer 9	✗	✗	✓
Chrome 17	✓	✗	✓

Minimal Fingerprints

- Goal: Determine minimal fingerprints
  1. Define the test set (=set of browsers)
  2. Collect failed test cases
  3. Calculate minimal fingerprints
  4. For every client: Run fingerprints

Result: If browser version $\in$ test set: confirm browser version

- "Mind the gap:"
  - Probably not for every test set solvable
  - Can become "big"

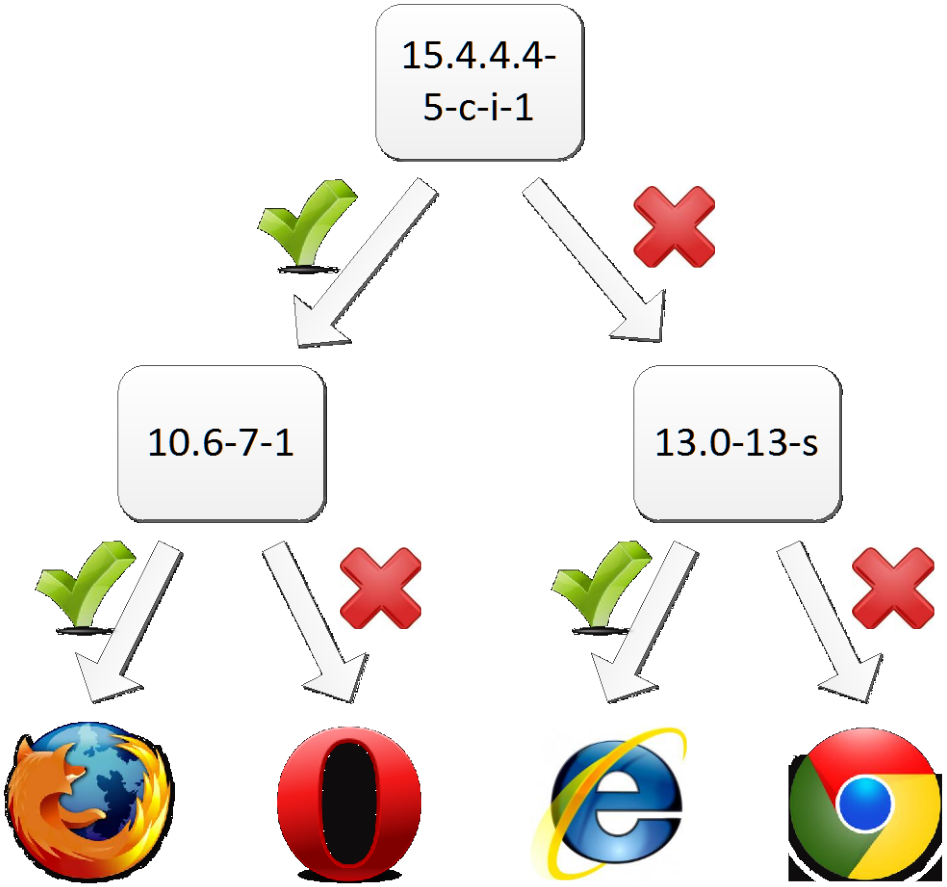
Decision Tree

- Goal: Minimize number of tests run at the client
  1. Define the testset (=set of browsers)
  2. Collect failed test cases
  3. Calculate uniqueness of every failed test case
  4. Build binary decision tree, iteratively

Result: Minimal path through decision tree for unknown browsers

- Benefits:
  - O(logn) instead of O(n)
  - Thus even faster
  - Can be used as first stage for minimal fingerprinting

Decision Tree



Evaluation - Tor Browser Bundle

- Basics Tor:
  - Internet anonymization network
  - Hides a user's real IP address
  - Hundreds of thousands users every day
  - Approx. 3000 servers run by volunteers
- Tor Browser Bundle:
  - Everything prepackaged (Tor, Vidalia, Firefox, ...)
  - Runs without admin rights
  - Among other features: Uniform UserAgent to increase size of the anonymity set

Evaluation - Tor Browser Bundle

- Uniform UserAgent:
  - Tor - Mozilla/5.0 (Windows NT 6.1; rv:5.0) Gecko/20100101 Firefox/5.0
  - Real - Mozilla/5.0 (X11; Linux x86 64; rv:9.0.1) Gecko/20111222 Firefox/9.0.1
- Vulnerable to Javascript Fingerprinting?
  - Yes!
  - Every Firefox > 3.5 can be easily distinguished
  - Can harm user privacy and decrease anonymity set
  - However, not a real attack on Tor

Evaluation - Tor Browser Bundle

Version TBB	Browser	UserAgent	test262	exp. test262	Attackable?
2.2.35-3	Firefox 9.0.1	Firefox 5.0	167	264	✓
2.2.34-3	Firefox 8.0.1	Firefox 5.0	167	264	✓
2.2.33-2	Firefox 7.0.1	Firefox 5.0	190	264	✓
2.2.32-3	Firefox 6.0.2	Firefox 5.0	214	264	✓
2.2.30-2	Firefox 5.0.1	Firefox 5.0	264	264	✗
2.2.25-1	Firefox 4.0.1	Firefox 4.0	290	290	✗
2.2.24-1	Firefox 4.0	Firefox 3.6.3	290	3956	✓

Evaluation

- Tested our fingerprinting with a survey:
  - 189 participants
  - Open for a few weeks in Summer 2011
  - 10 test cases per browser in test set
  - Test set:
    - IE8
    - IE9
    - Chrome 10
    - Firefox 4
- Ground truth:
  - UserAgent String
  - Manual identification by participant

Evaluation

- Performance:
  - All files: 24 Kilobytes
  - Fingerprints: 2.500-3.000 Bytes
  - 90 ms on average on PC
  - 200 ms on average on smartphone
- Results:
  - 175 out of 189 browsers covered by test set
  - 100 % detection rate
  - No false positives!
  - 14 not covered were mostly smartphones
  - 1 UserAgent manipulation discovered

secure sba-research.org

COMET Competence Centers for Excellent Technologies

Preserving demanded privacy constraints:

An algorithm for collusion-resistant anonymization and fingerprinting of sensitive microdata

Peter Kieseberg, Sebastian Schrittwieser, Martin Mulazzani, Isao Echizen , Edgar Weippl

bmvit Bundesministerium für Verkehr, Innovation und Technologie

bmwfj Bundesministerium für Wirtschaft, Familie und Jugend

Christian Doppler Forschungsgesellschaft

FFG

wirtschaftsagentur wien zit Die Technologieagentur der Stadt Wien

Overview

- Regulatory compliance requires **anonymization** of personal data prior to transmission to other parties.
- Transmission always implicates some **loss of control** over the data since further dissemination is possible without knowledge of the data owner.

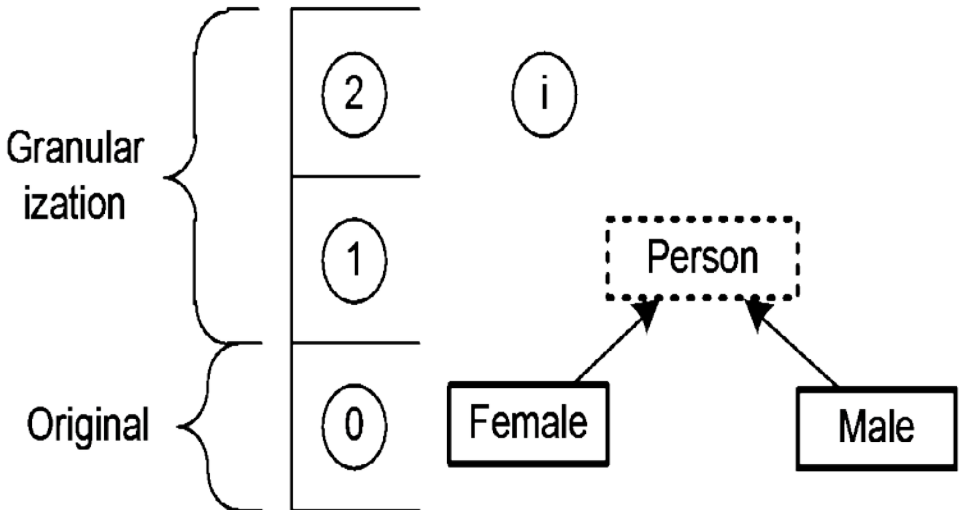
- In one single step
  - anonymization and
  - fingerprinting

  of microdata such as database records is combined.
- Detectability of colluding attackers.

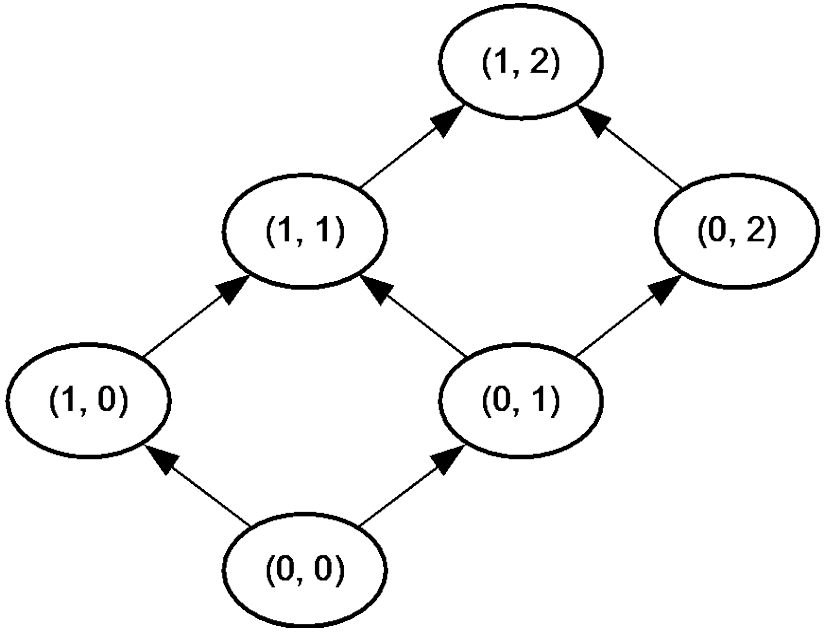
2-anonymity

Birthday	Sex	Disease
1970	F	Chest pain
1970	M	Short breath
1970	F	Obesity
1970	M	Short breath

Generalization Patterns



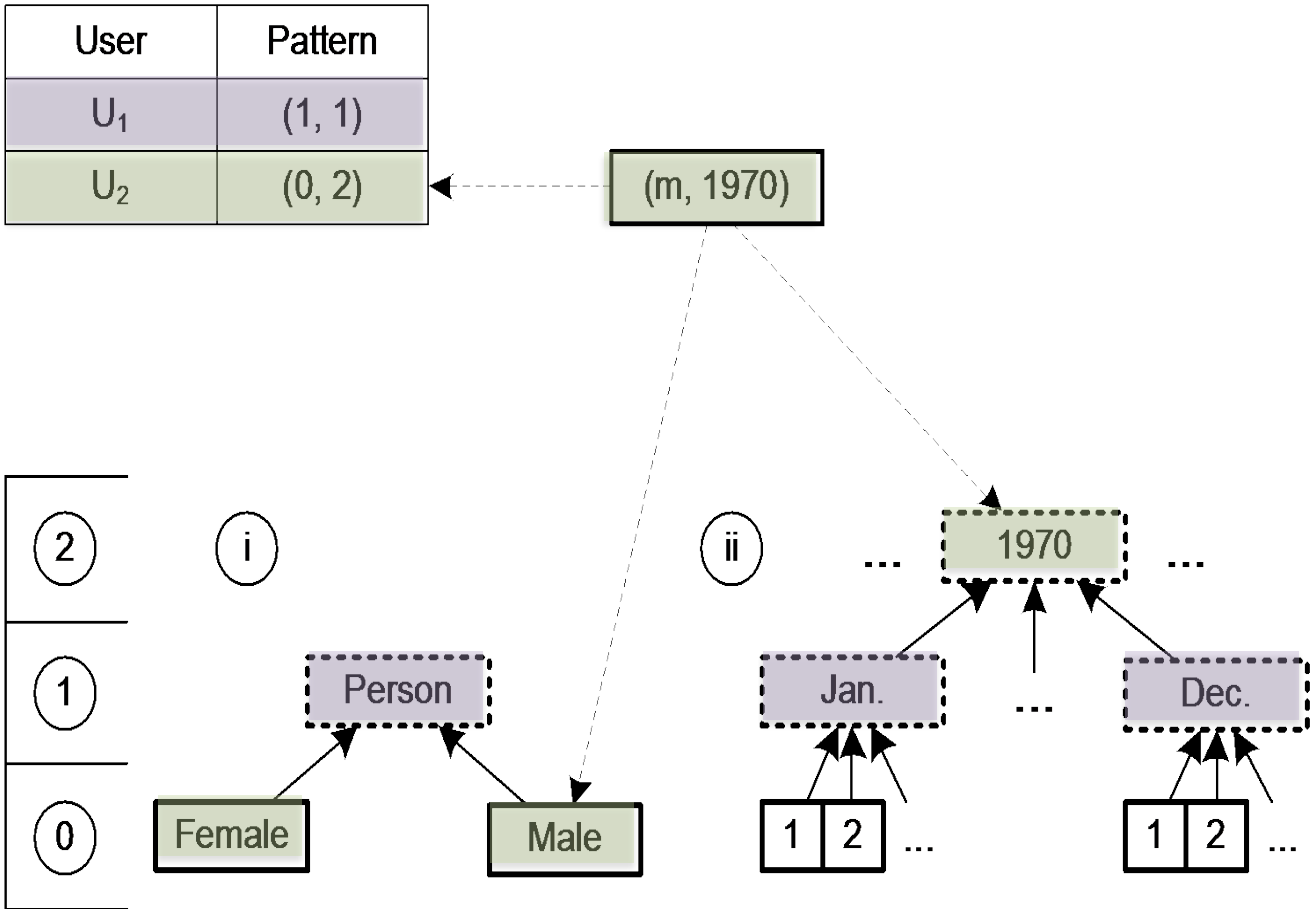
Lattice diagram showing the generalization patterns



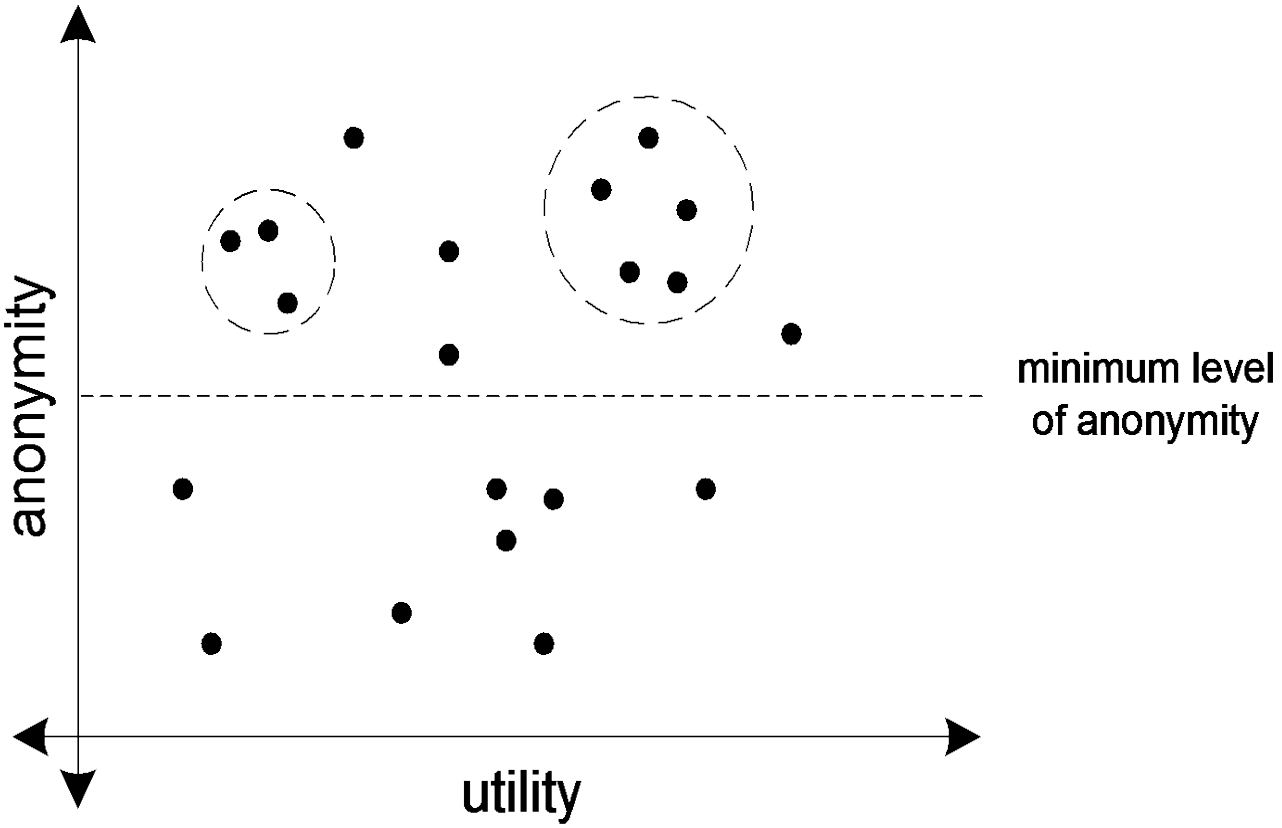
Original data and two anonymized sets (k=2)

Original data (a_0, b_0)				First Set (a_0, b_2)			Second Set (a_1, b_1)		
name	***sex***	***birthday***	***disease***	***sex***	***birthday***	***disease***	***sex***	***birthday***	***disease***
Bob	M	19.03.1970	chest pain	M	1970	chest pain	P	03.1970	chest pain
Dave	M	20.03.1970	short breath	M	1970	short breath	P	03.1970	short breath
Alice	F	18.04.1970	obesity	F	1970	obesity	P	04.1970	obesity
Eve	F	21.04.1970	cancer	F	1970	cancer	P	04.1970	cancer

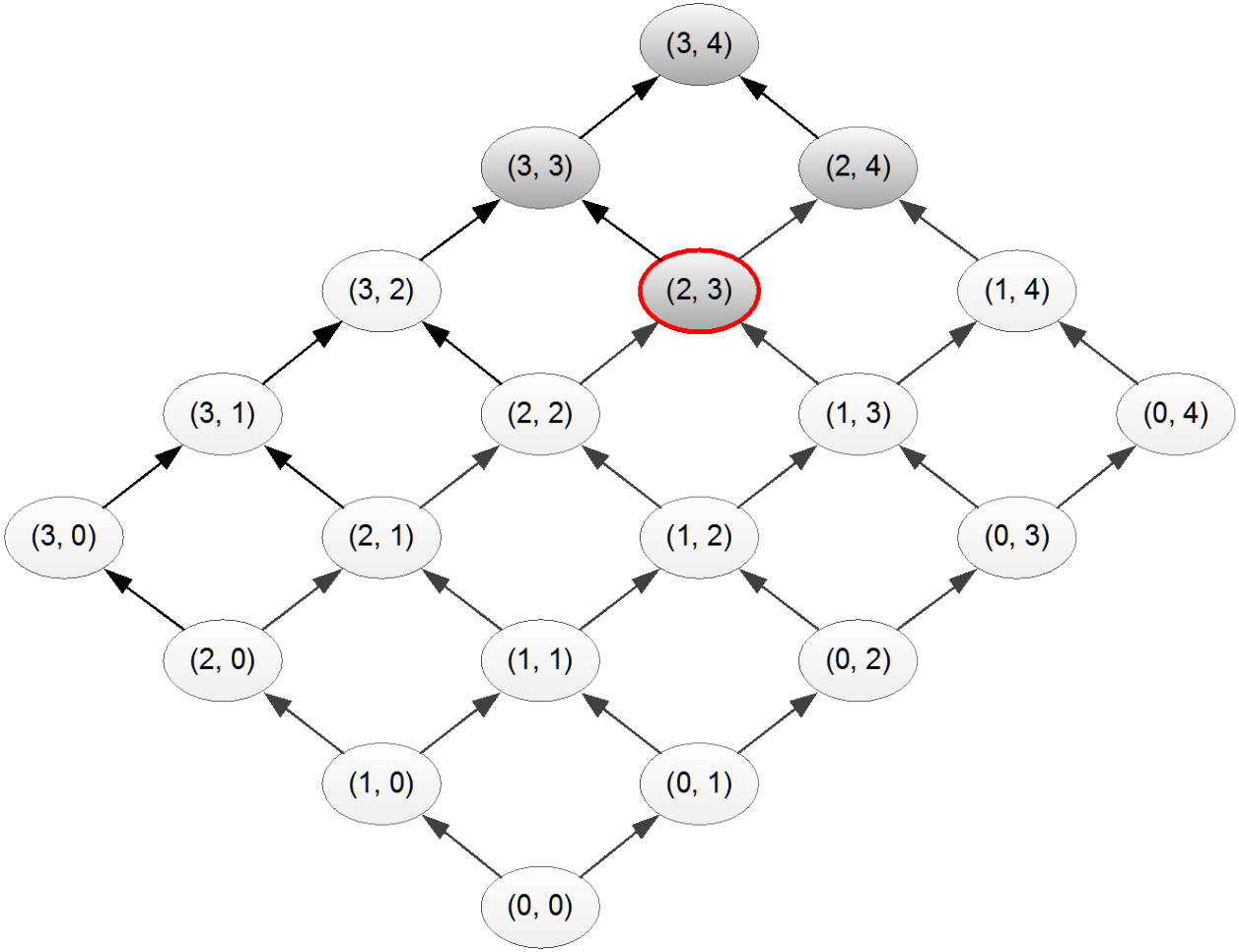
Identifying the source of data leakage based on patterns



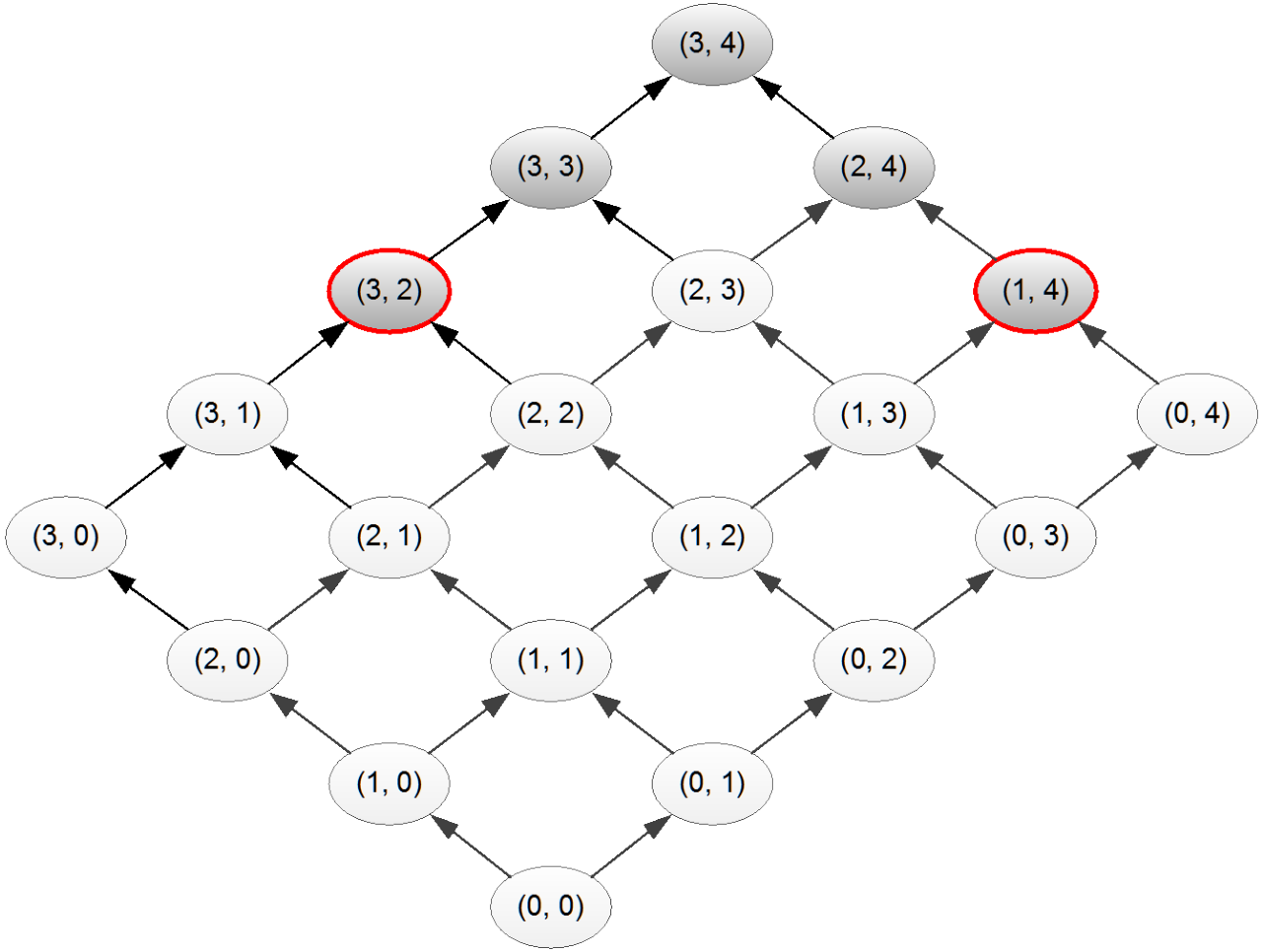
Clustering anonymization strategies



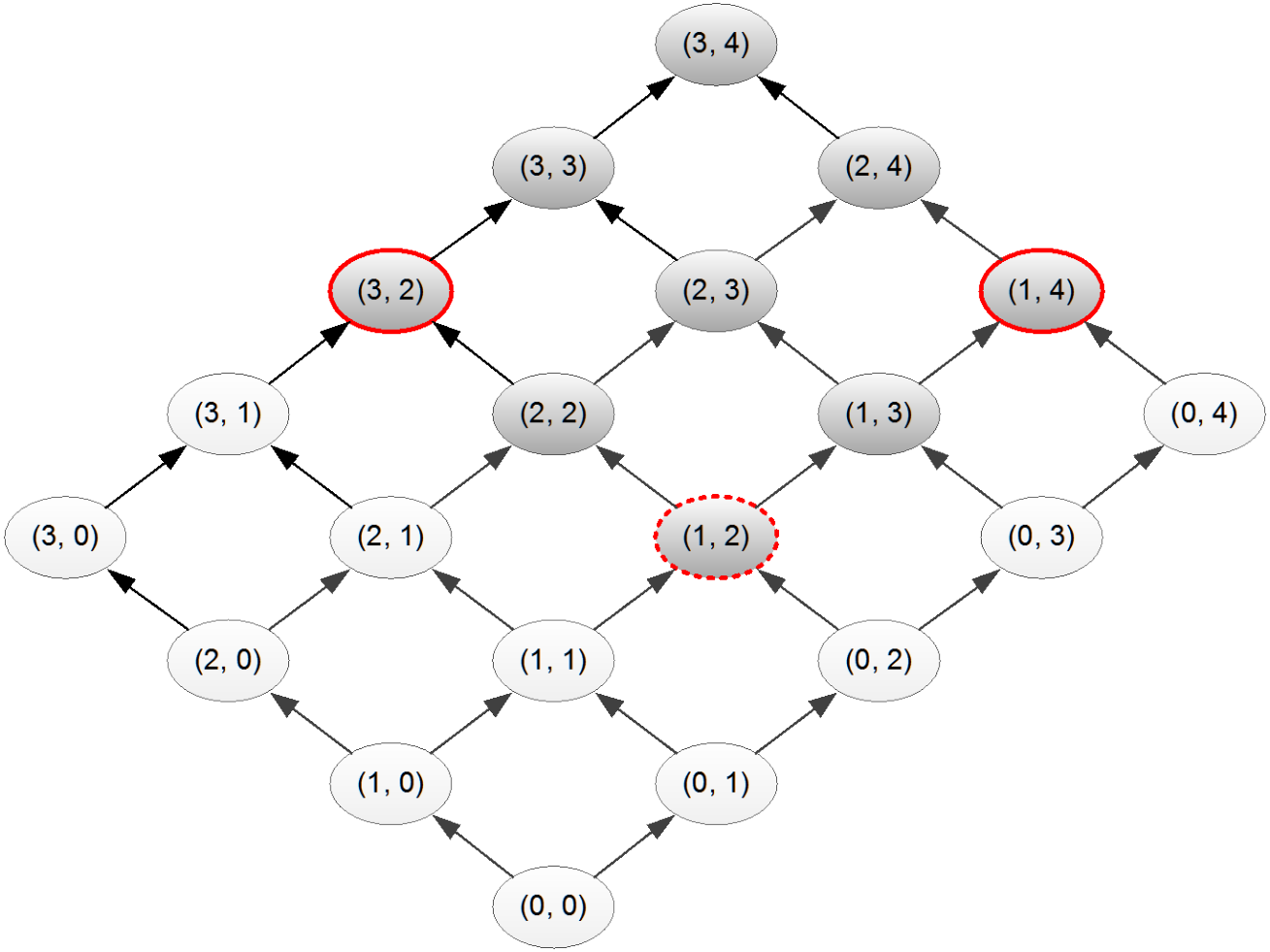
The hull of a generalization pattern



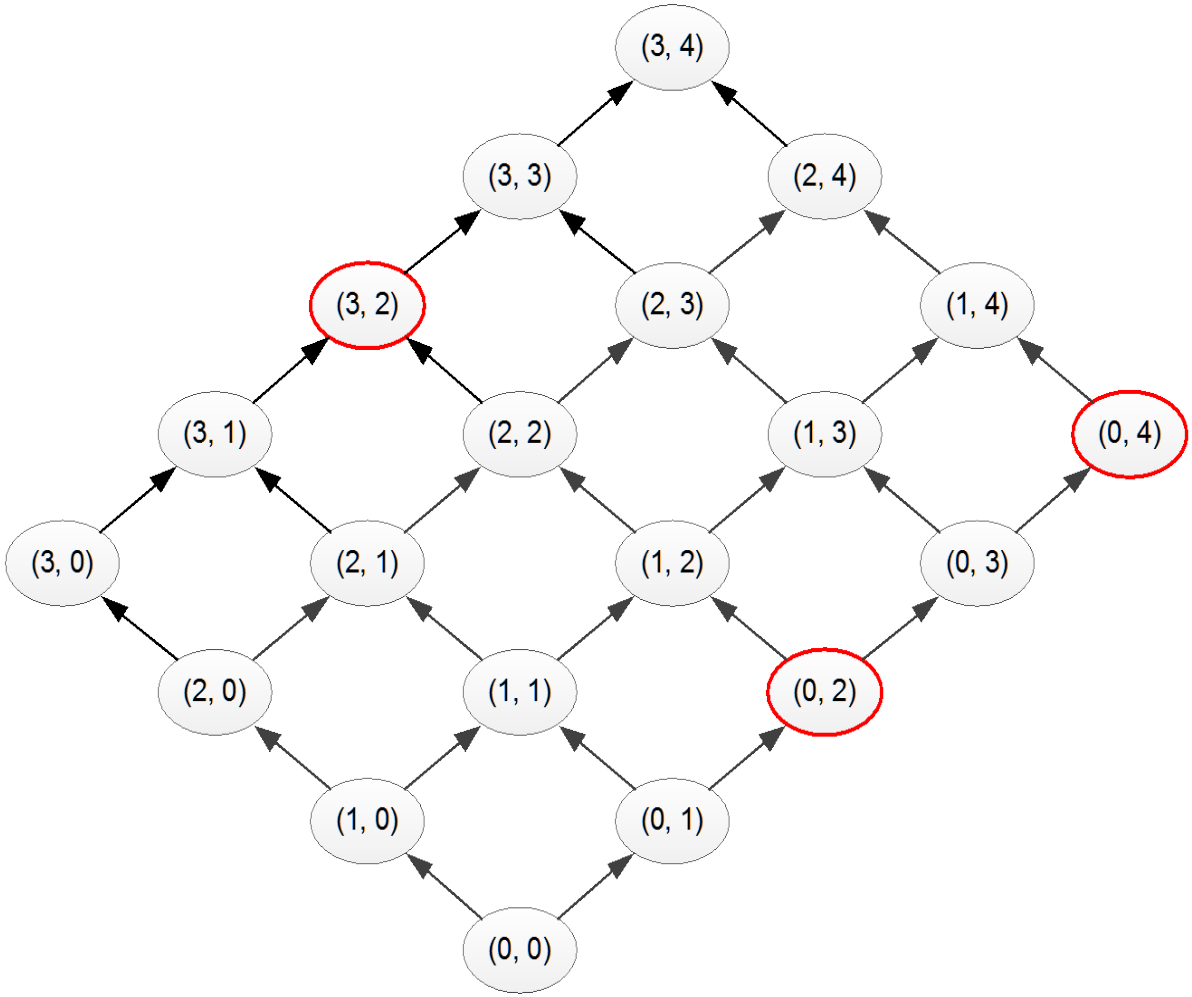
Union of the hulls of two generalization patterns



Hull generated by colluding two generalization patterns



Example of a collusion attack



Number of possible data consumers

call-detail-records (CDRs)

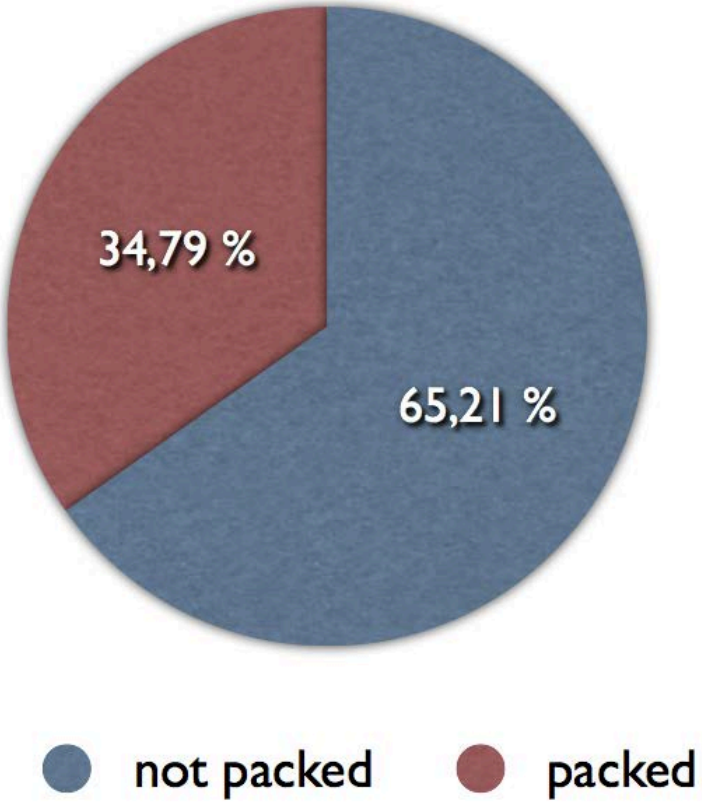
- the typical data set possesses much more quasi identifiers (more than 50 for a data set in a typical ICB-system)
- than possible data consumers (usually less than 10), thus rendering this approach perfectly feasible.

Engineering Science

- Covert Computation
  Hiding Code in Code for Obfuscation Purposes

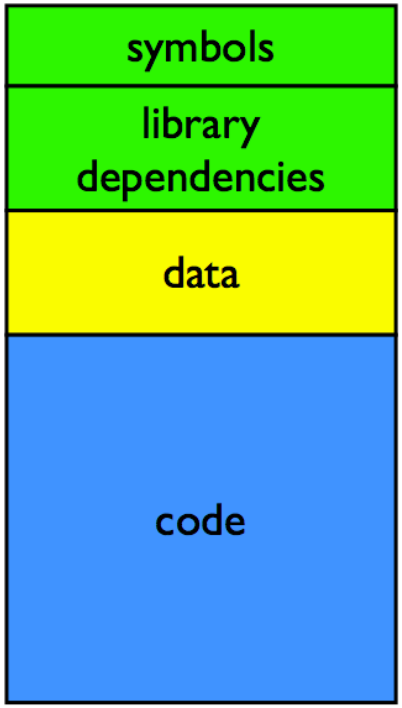
Sebastian Schrittwieser and Stefan Katzenbeisser and Peter Kieseberg and Markus Huber and Manuel Leithner and Martin Mulazzani and Edgar R. Weippl, "**Covert Computation – Hiding Code in Code for Obfuscation Purposes**," in Proceedings of the 8th International Symposium on ACM Symposium on Information , Computer and Communications Security (ASIACCS 2013), 2013.

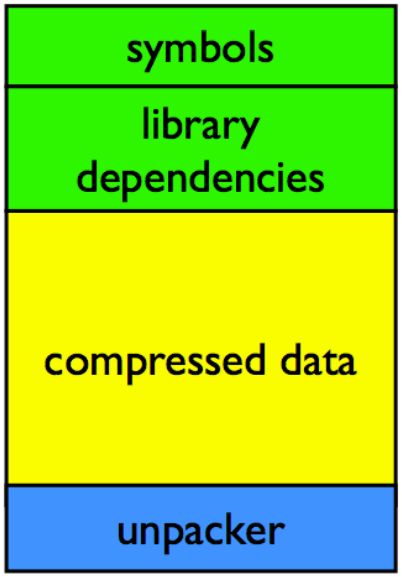
Malware Obfuscation



Source: Rodrigo Rubira Branco (2012)

Detecting Packed Code





Mimimorphism: A New Approach to Binary Code Obfuscation

Zhenyu Wu, Steven Gianvecchio, Mengjun Xie; and Haining Wang
The College of William and Mary
Williamsburg, VA 23187, USA
{adamwu, srgian, mjxie, hnw}@cs.wm.edu

ABSTRACT

Binary obfuscation plays an essential role in evading malware static analysis and detection. The widely used code obfuscation techniques, such as polymorphism and metamorphism, focus on evading syntax based detection. However, statistic test and semantic analysis techniques have been developed to thwart their evasion attempts. More recent binary obfuscation techniques are divided in their purposes of attacking either statistical or semantic approach, but not both. In this paper, we introduce mimimorphism, a novel binary obfuscation technique with the potential of evading both statistical and semantic detections. Mimimorphic malware uses instruction-syntax-aware high-order mimic functions to transform its binary into mimicry executables that exhibit high similarity to benign programs in terms of statistical properties and semantic characteristics. We implement a prototype of the mimimorphic engine on the Intel x86 platform, and evaluate its capability of evading statistical anomaly detection and semantic analysis detection techniques. Our experimental results demonstrate that the mimicry executables are indistinguishable from benign programs in terms of byte frequency distribution and entropy, as well as control flow

1. INTRODUCTION

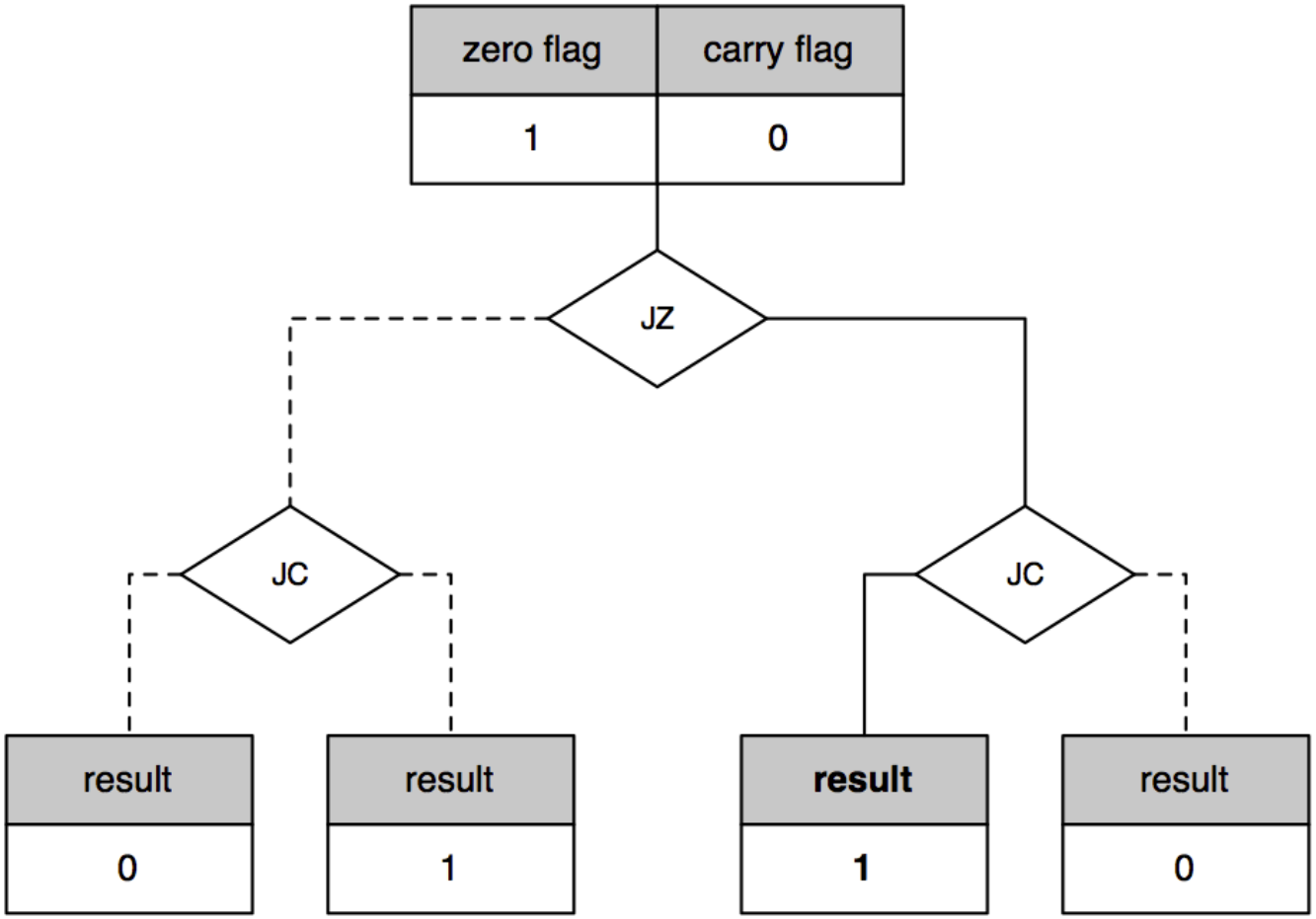
Real-time malicious binary detection is the first line of defense against malicious software. With the prevalence of anti-malware software nowadays, in order to be executed on a host computer, a piece of binary code is subjected to a number of detection scans during transportation and before execution. Consequently, evading real-time binary detection is critical for malware to succeed in propagation.

To date, real-time malware detection largely relies on static binary analysis, due to its significant speed and resource consumption advantages over dynamic executable analysis [1, 2, 3, 4, 5]. Malware mainly evades static analysis detections through binary obfuscations, namely oligomorphism, polymorphism, and metamorphism [6]. Oligomorphism is used to evade byte sequence signature detections on the malware functional code. It utilizes simple operations such as XOR to scramble malware functional code before propagation, and decodes it while executing. Evolved from oligomorphism, polymorphism encodes malicious code by "packing" (i.e., compressing or encrypting), and then camouflages the "unpacker" (the decompressing or decrypting code) by using binary mutation techniques, such as instruction substitution and register remapping. Instead of packing program binaries, metamorphism

Side Effects

```
ADD EAX, EBX
```

XOR



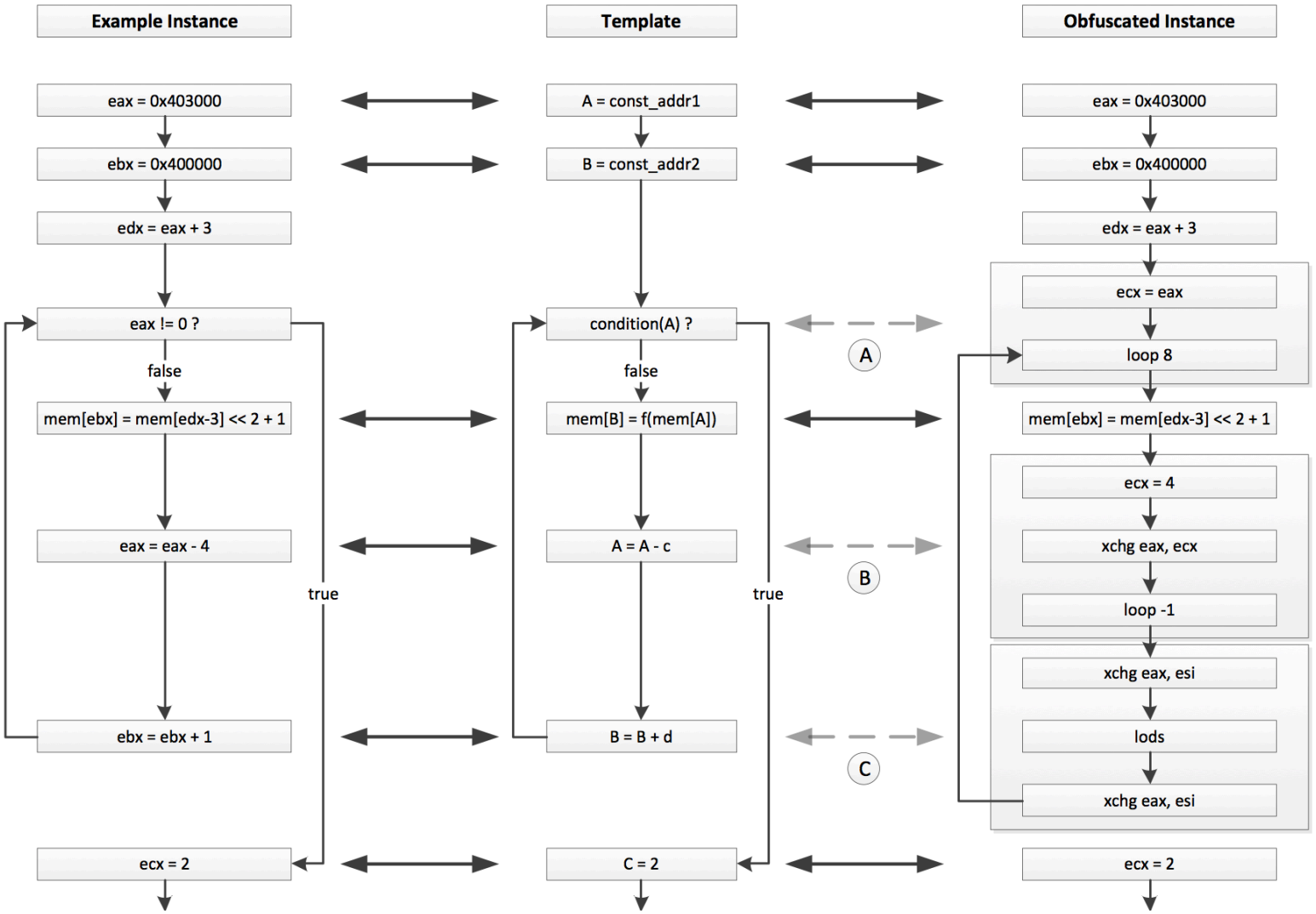
```
MOV ECX, 200
XCHG EAX, ECX
LOOP -1
```

Semantic-aware Malware Detection

- Static analysis with advanced patterns
- Templates for malicious behavior
- Decision based on result instead of actual implementation
- Model of the microprocessor

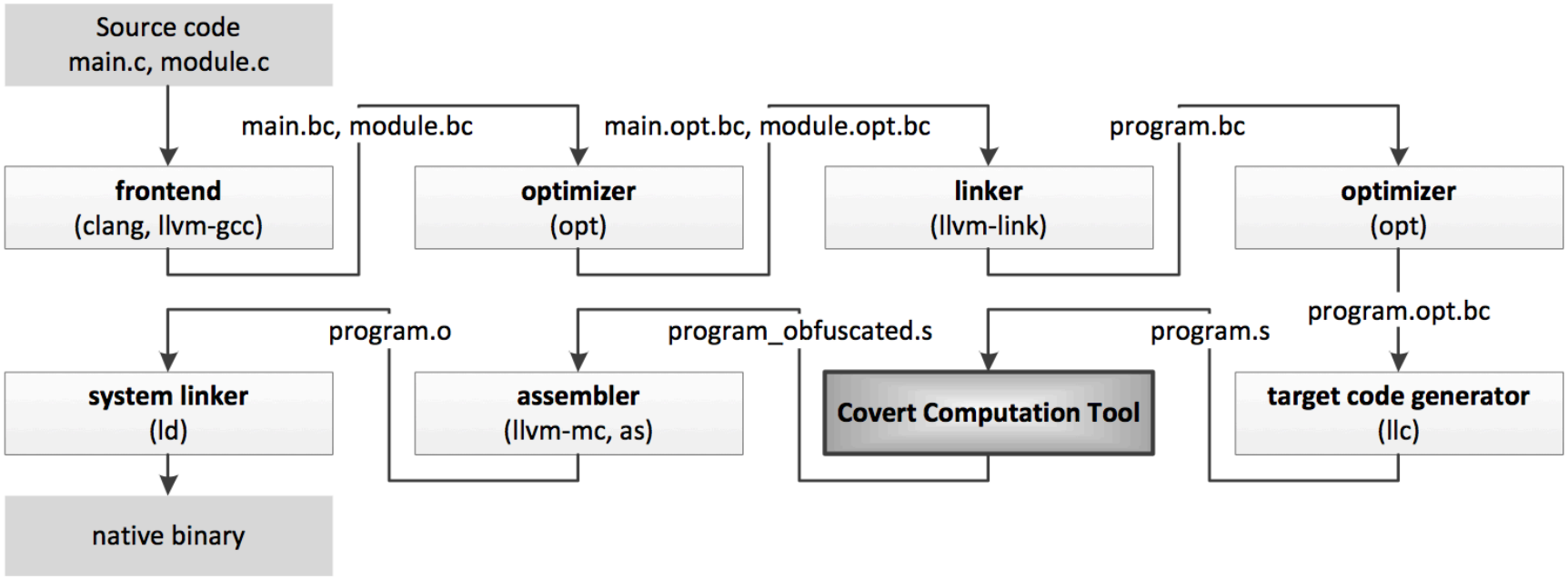
Christodorescu, Mihai, et al. "Semantics-aware malware detection." 2005 IEEE Symposium on Security and Privacy

Mapping Functionality



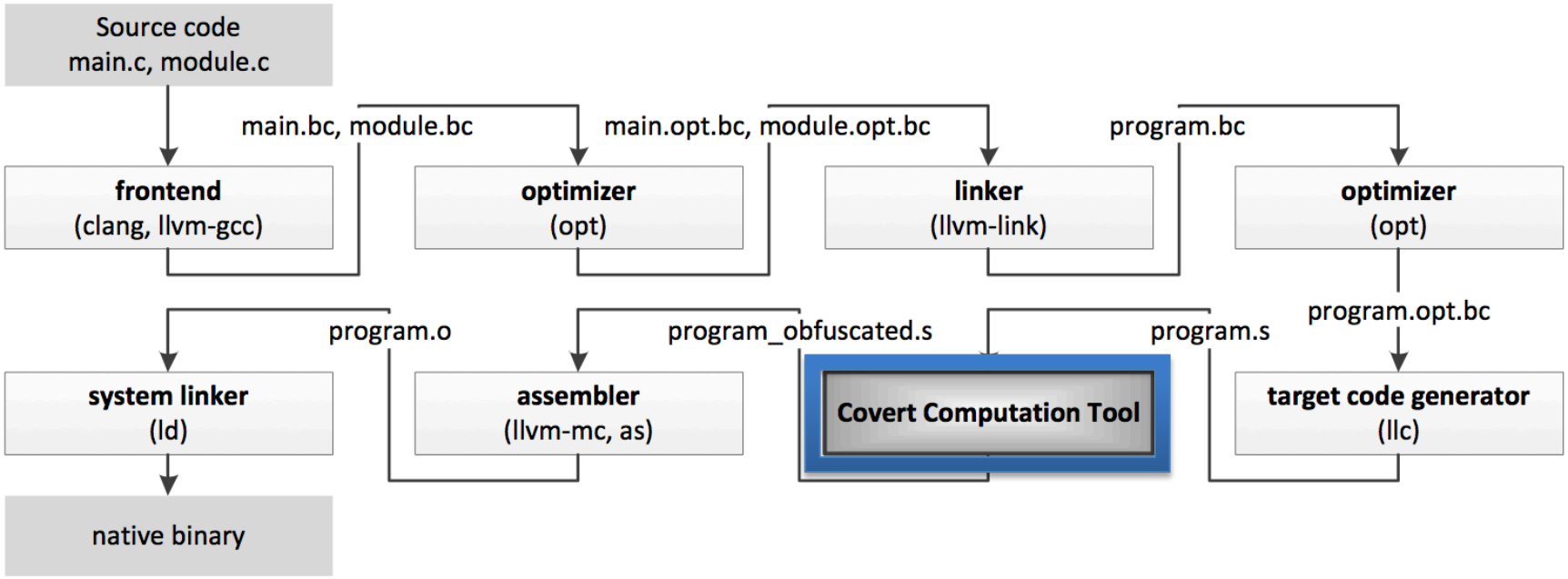
Implementation

Compile-time Obfuscation

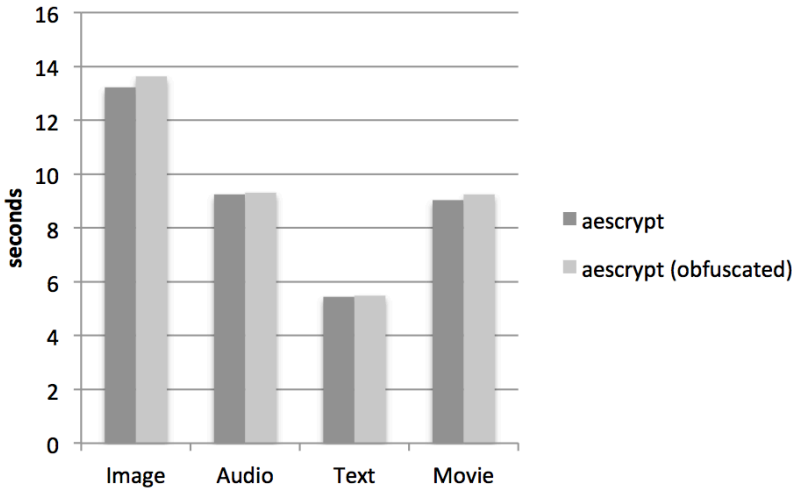


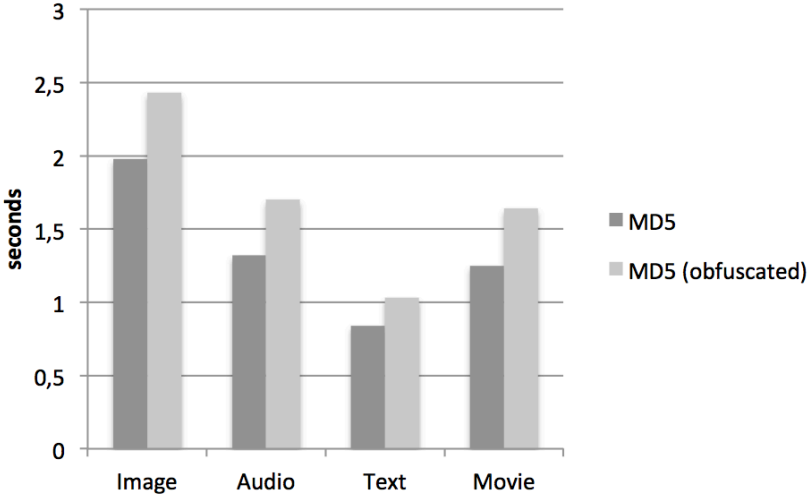
Compile-time Obfuscation

- Implemented by intercepting the compilation process of LLVM
- Compiler wrapper



Performance Evaluation





Summary Cloud Services & Privacy

- **App providers** (and **advertising providers** used by them) have access to user data.
- **Instant messaging** applications that use a phone number for authentication give the **impression** to be as secure as SMS but are much weaker.
  - 6 out of 9 similar applications had the same problems
- Things that students should learn in Security 101.
  - Trust in client application (Dropbox)
  - Missing input validation (WhatsApp)
  - Attacker model (WhatsApp & SSL; Facebook & weak hosting at 3rd party providers)
  - Security assumptions (no friend relationship in SMS)
- Software obfuscation as possible temporary solution to hide sensitive data.

Conclusion Code in Code

- Today's microprocessor architectures are highly complex
- Side effects are difficult to map with machine models - *knowledge gap*
- Hiding code in code
- Moderate effects on binary size, increased complexity (relevance?)

secure
sba-research.org