Leaving Timing Channel Fingerprints in Hidden Service Log Files

By

Bilal Shebaro, Fernando Perez-Gonzalez and Jedidiah R. Crandall

Presented At

The Digital Forensic Research Conference

DFRWS 2010 USA Portland, OR (Aug 2nd - 4th)

Leaving Timing Channel Fingerprints in Hidden Service Log Files

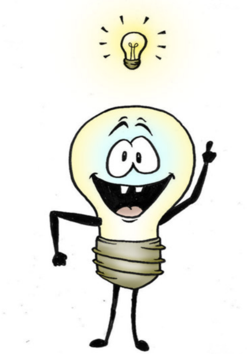
Bilal Shebaro (CS-UNM)

Fernando Perez-Gonzalez (ECE-UNM & Theory & Communication-Uni. Of Vigo)

Jedidiah R. Crandall (CS-UNM)

The University of New Mexico

Basic Idea



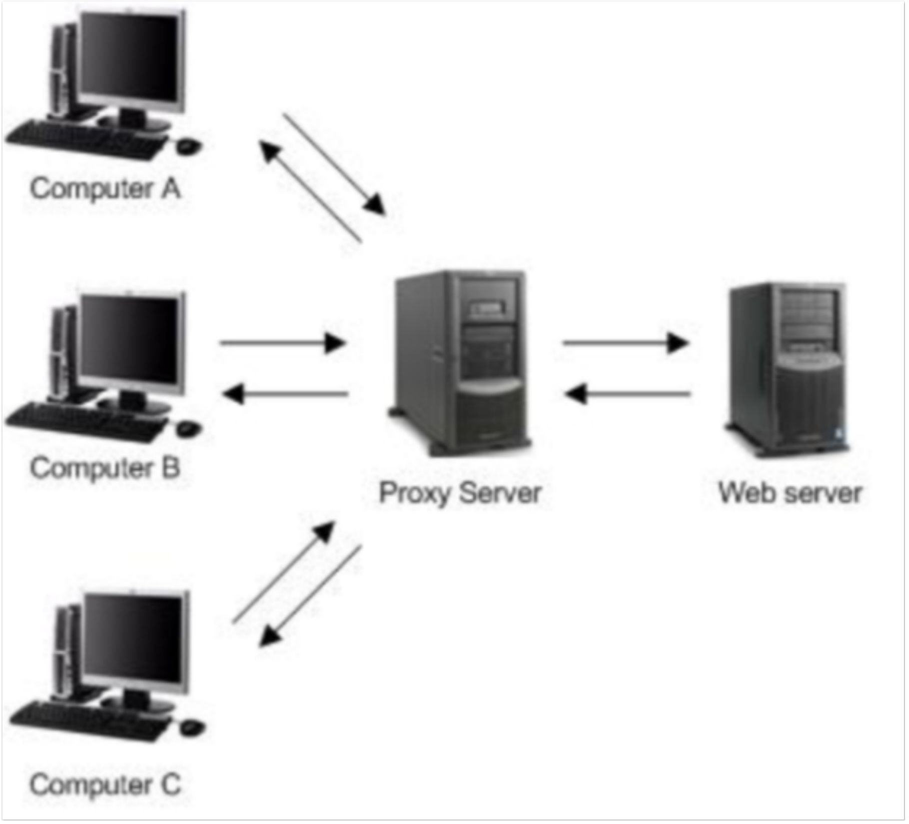
- Someone serving illegal content using Tor hidden service
- This physical machine is confiscated
- Our job is to prove that this machine is in fact the machine that had been hosting the illegal content
- We'll be fingerprinting its log file: leaving an identifiable fingerprint in this log file as a timing channel that can be recovered

Outline

- Basic Idea
- Requirements
- Tor Network
- Tor Hidden Services
- Webserver Traffic Analysis
- Tor Network Relay
- Our Fingerprinting Algorithm
- Results
- Discussion and Future Work

Requirements

- Tor hidden web server (apache)
- Tor client
- ECC: Reed Solomon Error Correction Code

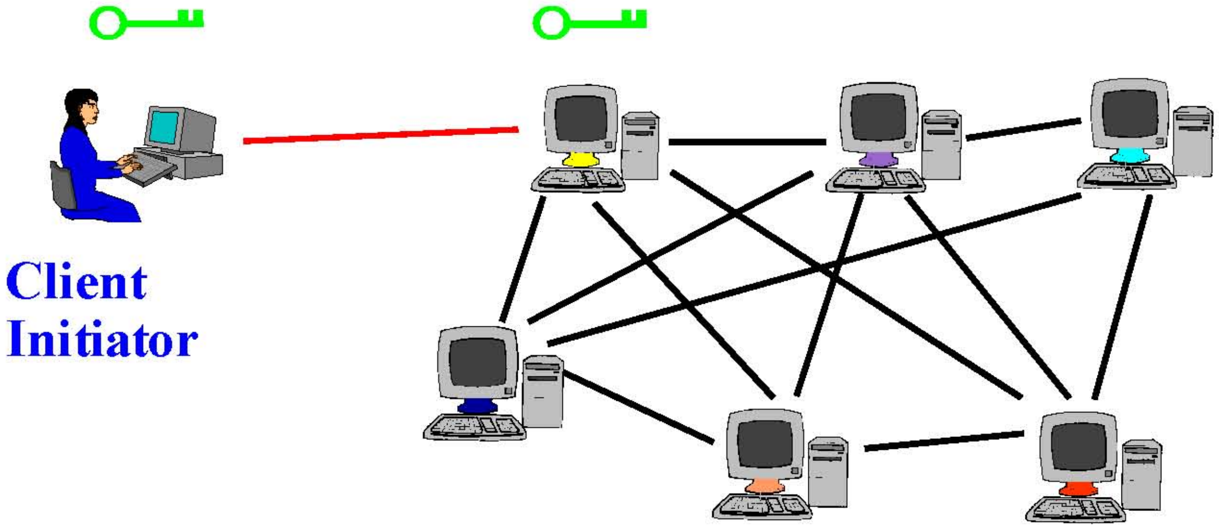


Tor

- Second-generation onion routing network
  - Developed by Roger Dingledine, Nick Mathewson and Paul Syverson
  - Specifically designed for low-latency anonymous Internet communications
- Running since October 2003
- 100 nodes on four continents, thousands of users
- "Easy-to-use" client proxy
  - Freely available, can use it for anonymous browsing

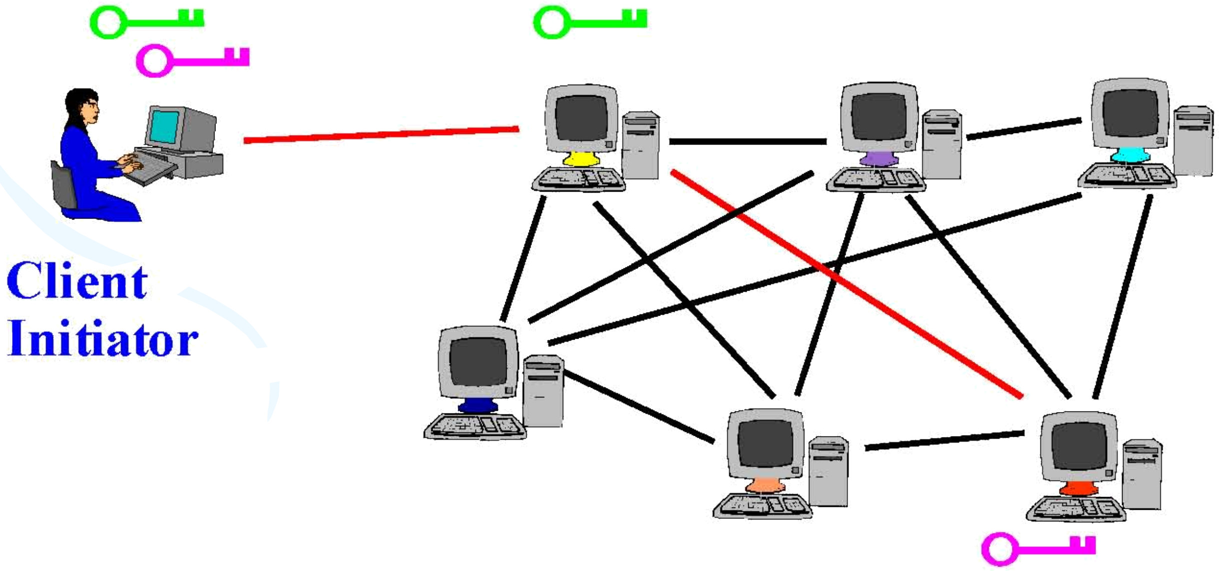
Tor Circuit Setup (1)

- Client proxy establish a symmetric session key and circuit with Onion Router #1



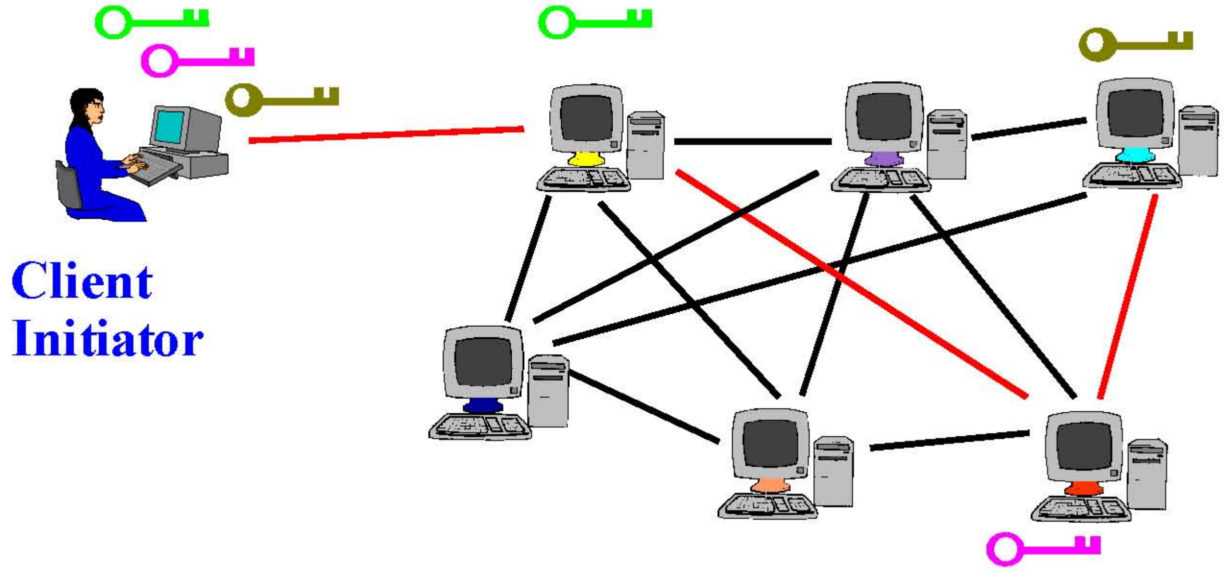
Tor Circuit Setup (2)

- Client proxy extends the circuit by establishing a symmetric session key with Onion Router #2
  - Tunnel through Onion Router #1

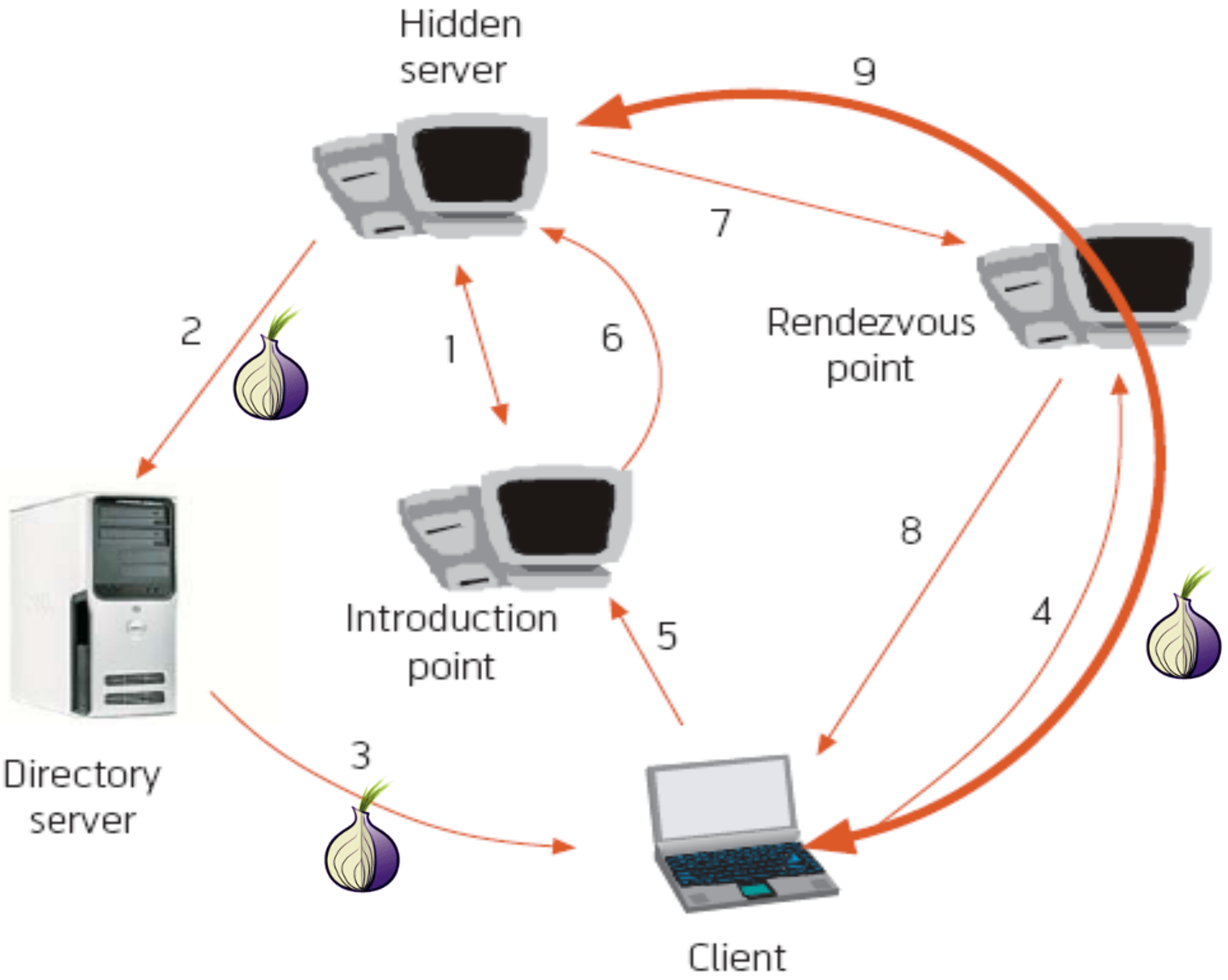


Tor Circuit Setup (3)

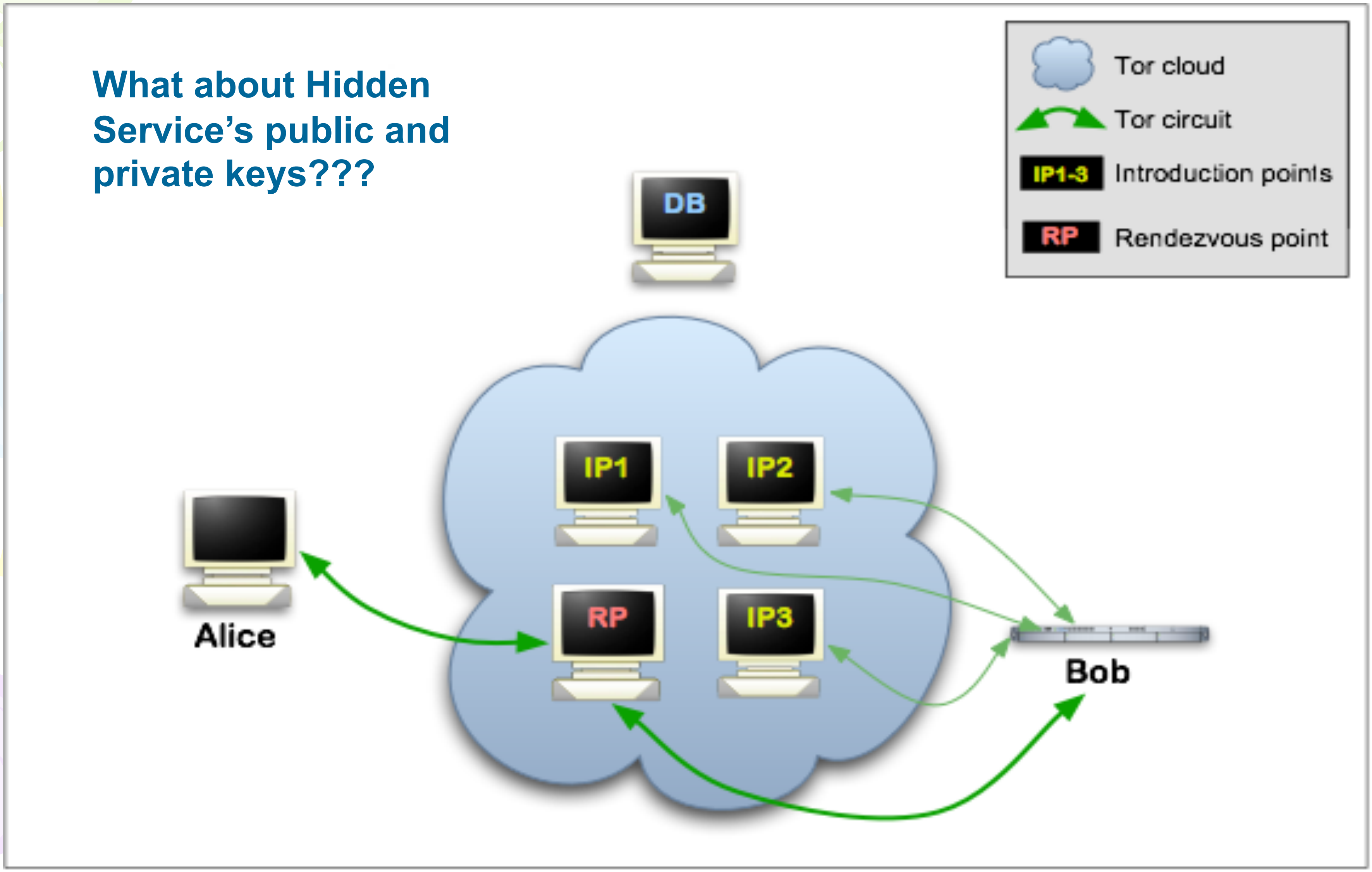
- Client proxy extends the circuit by establishing a symmetric session key with Onion Router #3
  - Tunnel through Onion Routers #1 and #2



Tor Hidden Services



Tor Hidden Services



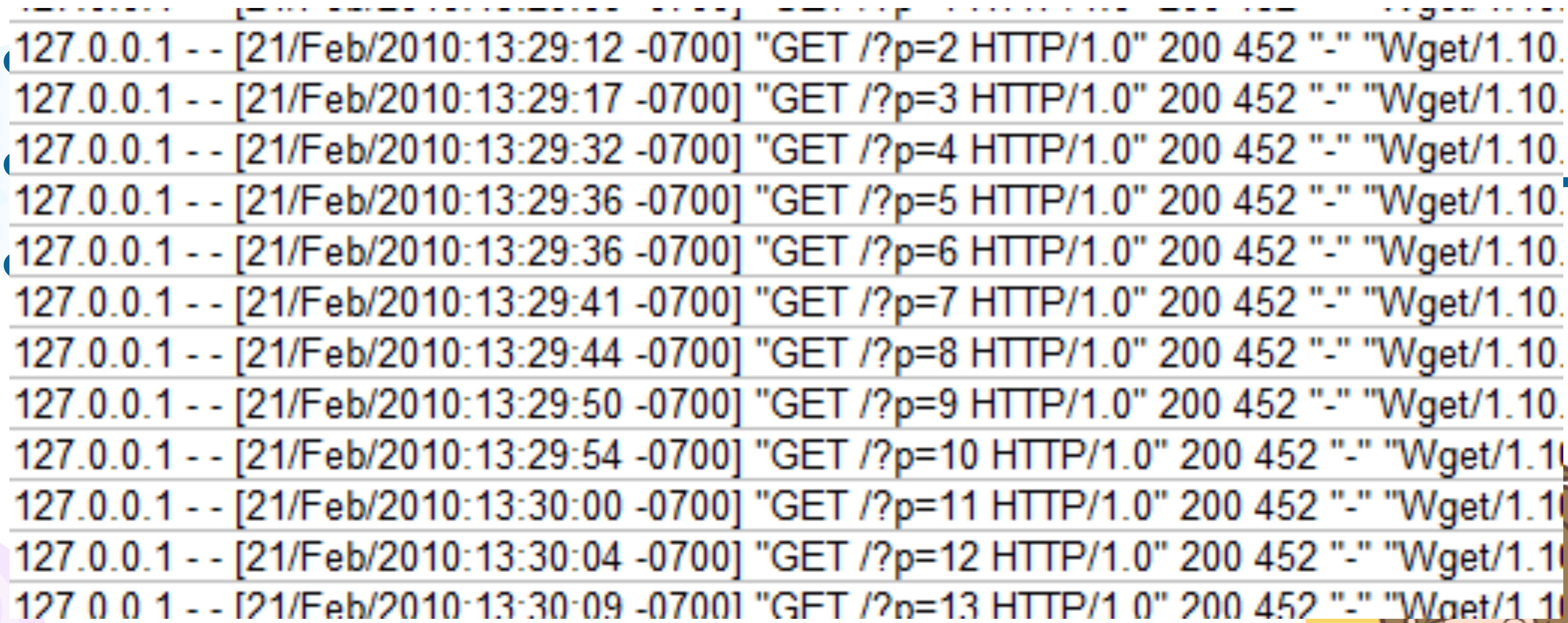
What about Hidden Service's public and private keys???

Threat Model & Challenges

- Our fingerprinting timestamps look exactly the same as those normal requests
- We don’t want to make it look suspicious
- Some portions of the log file were deleted
- False positive fingerprints may exist (solved by repeating our algorithm more than once as well as using RS-ecc)
- We are not claiming our method is impossible to detect, especially if the host knows our approach

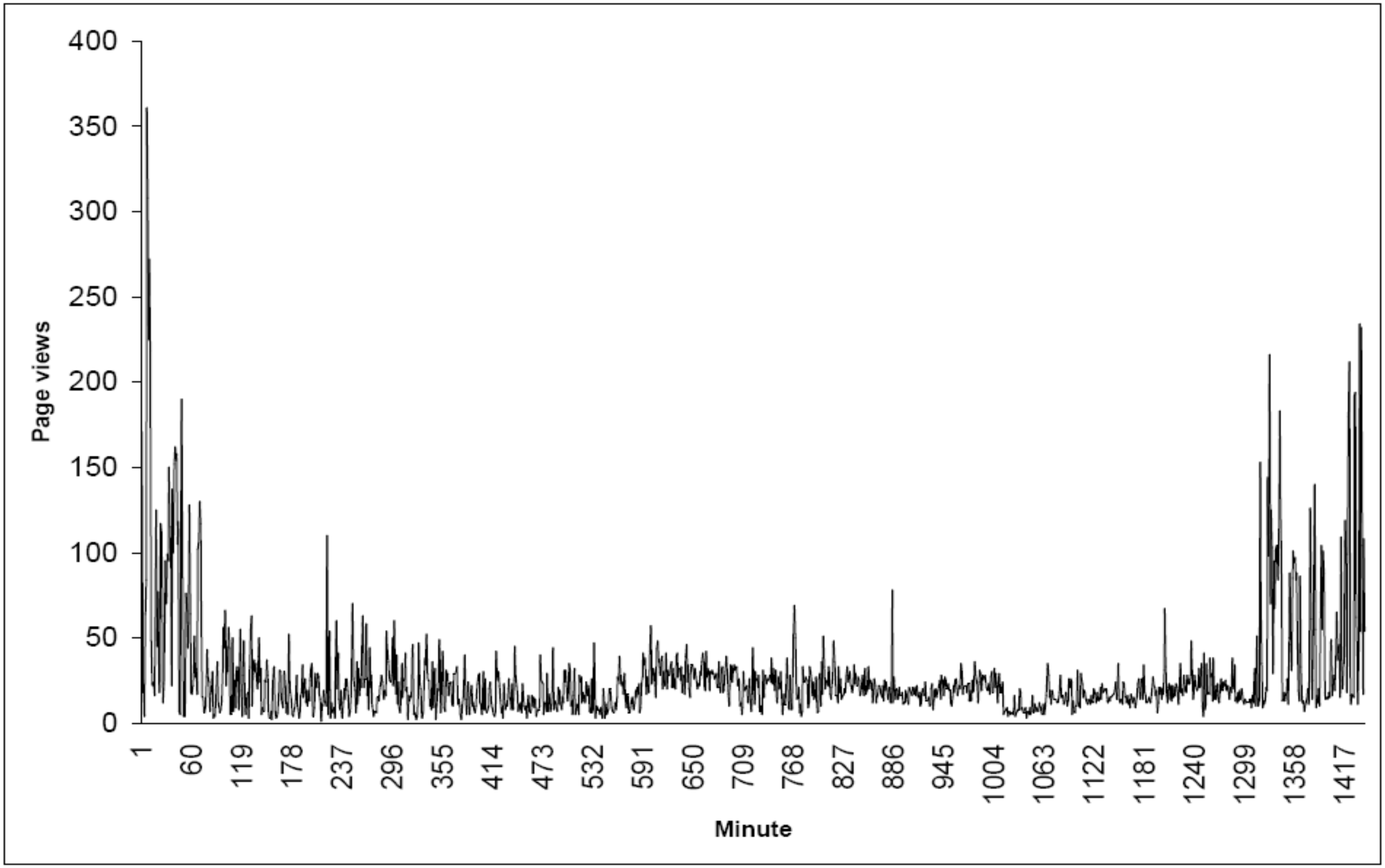
Threat Model & Challenges

- Our fingerprinting timestamps look exactly the same as those normal requests

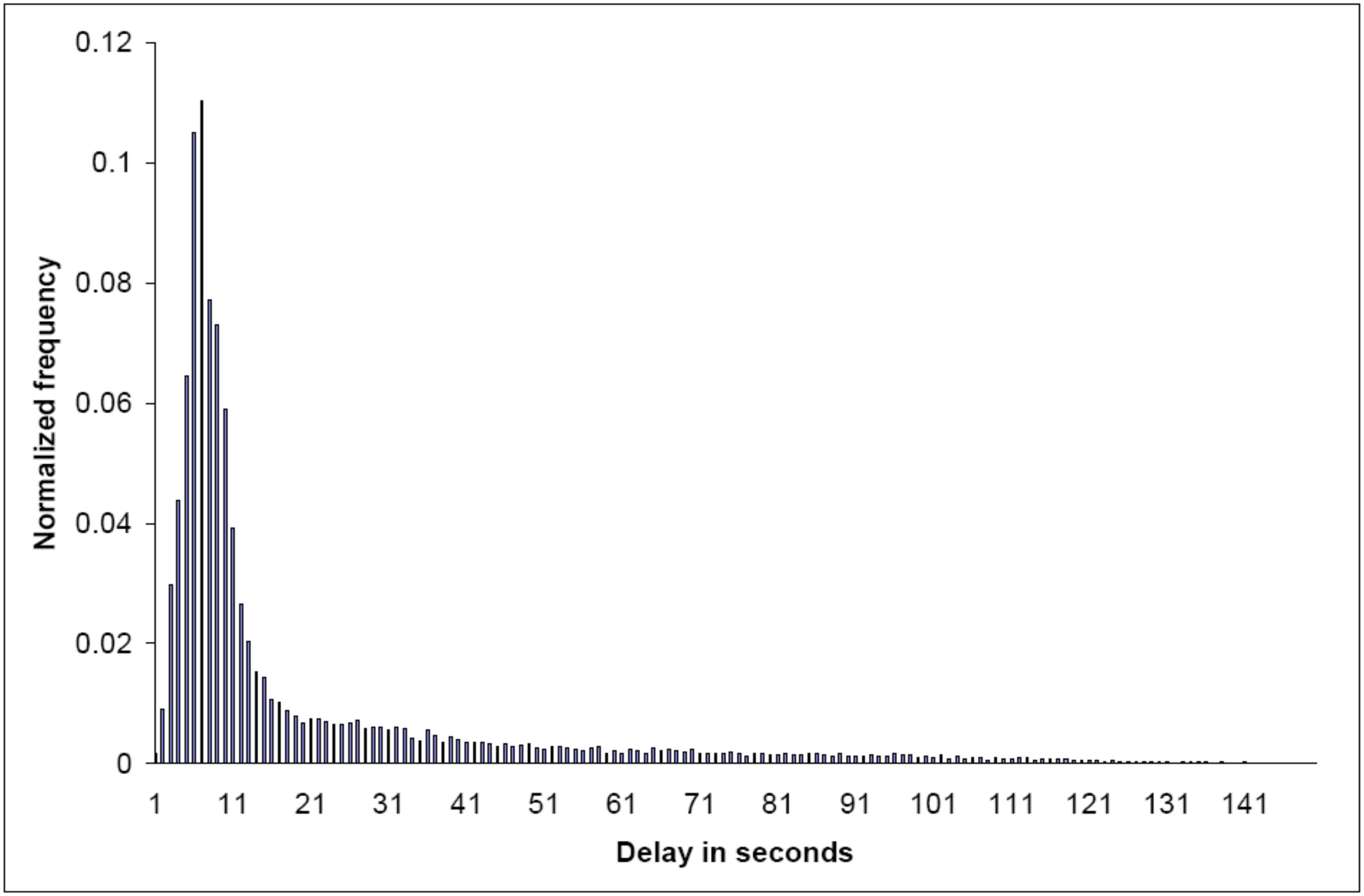


```
127.0.0.1 - - [21/Feb/2010:13:29:12 -0700] "GET /?p=2 HTTP/1.0" 200 452 "-" "Wget/1.10.
127.0.0.1 - - [21/Feb/2010:13:29:17 -0700] "GET /?p=3 HTTP/1.0" 200 452 "-" "Wget/1.10.
127.0.0.1 - - [21/Feb/2010:13:29:32 -0700] "GET /?p=4 HTTP/1.0" 200 452 "-" "Wget/1.10.
127.0.0.1 - - [21/Feb/2010:13:29:36 -0700] "GET /?p=5 HTTP/1.0" 200 452 "-" "Wget/1.10.
127.0.0.1 - - [21/Feb/2010:13:29:36 -0700] "GET /?p=6 HTTP/1.0" 200 452 "-" "Wget/1.10.
127.0.0.1 - - [21/Feb/2010:13:29:41 -0700] "GET /?p=7 HTTP/1.0" 200 452 "-" "Wget/1.10.
127.0.0.1 - - [21/Feb/2010:13:29:44 -0700] "GET /?p=8 HTTP/1.0" 200 452 "-" "Wget/1.10.
127.0.0.1 - - [21/Feb/2010:13:29:50 -0700] "GET /?p=9 HTTP/1.0" 200 452 "-" "Wget/1.10.
127.0.0.1 - - [21/Feb/2010:13:29:54 -0700] "GET /?p=10 HTTP/1.0" 200 452 "-" "Wget/1.1
127.0.0.1 - - [21/Feb/2010:13:30:00 -0700] "GET /?p=11 HTTP/1.0" 200 452 "-" "Wget/1.1
127.0.0.1 - - [21/Feb/2010:13:30:04 -0700] "GET /?p=12 HTTP/1.0" 200 452 "-" "Wget/1.1
127.0.0.1 - - [21/Feb/2010:13:30:09 -0700] "GET /?p=13 HTTP/1.0" 200 452 "-" "Wget/1.1
```

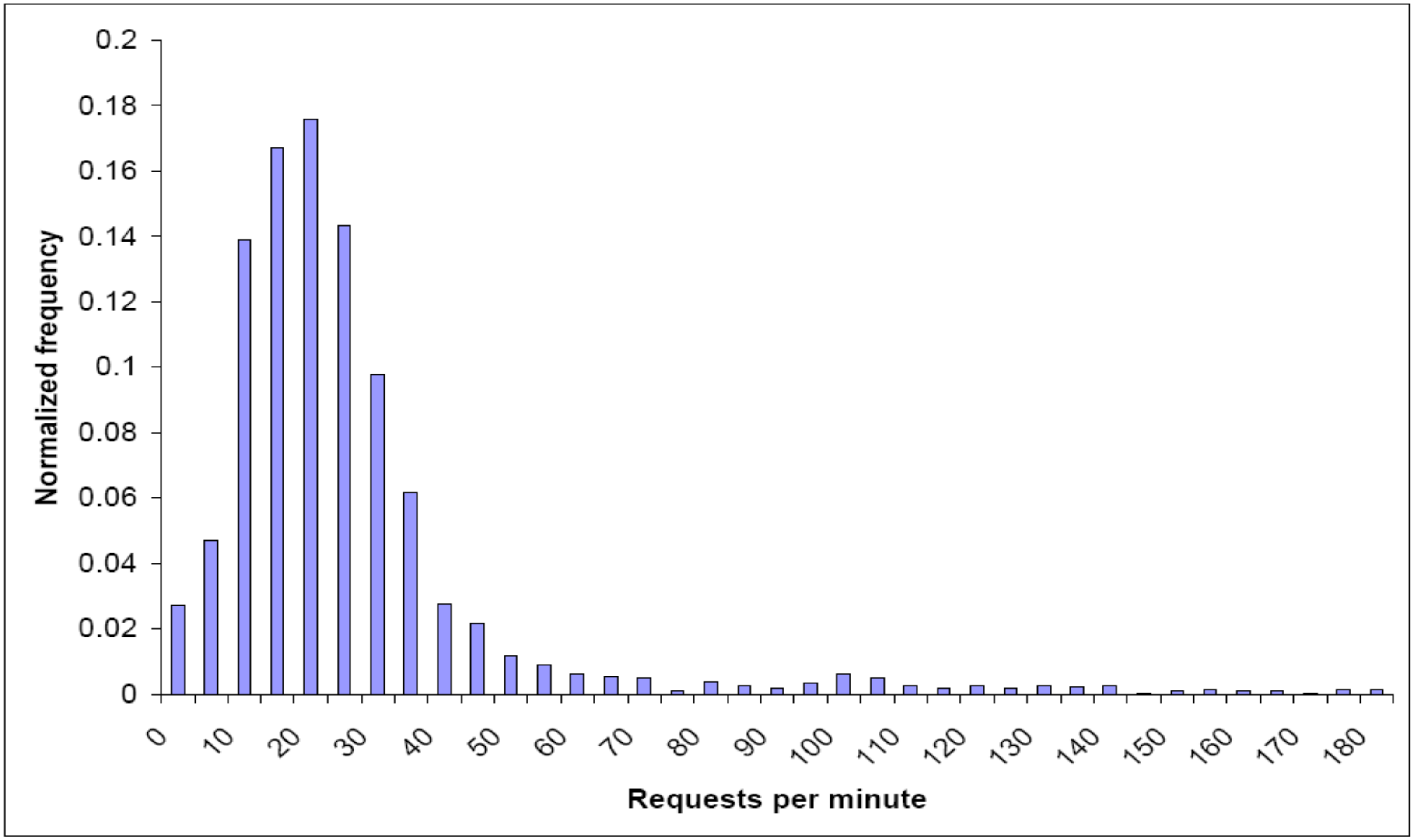
Web server traffic for a 24-hour period



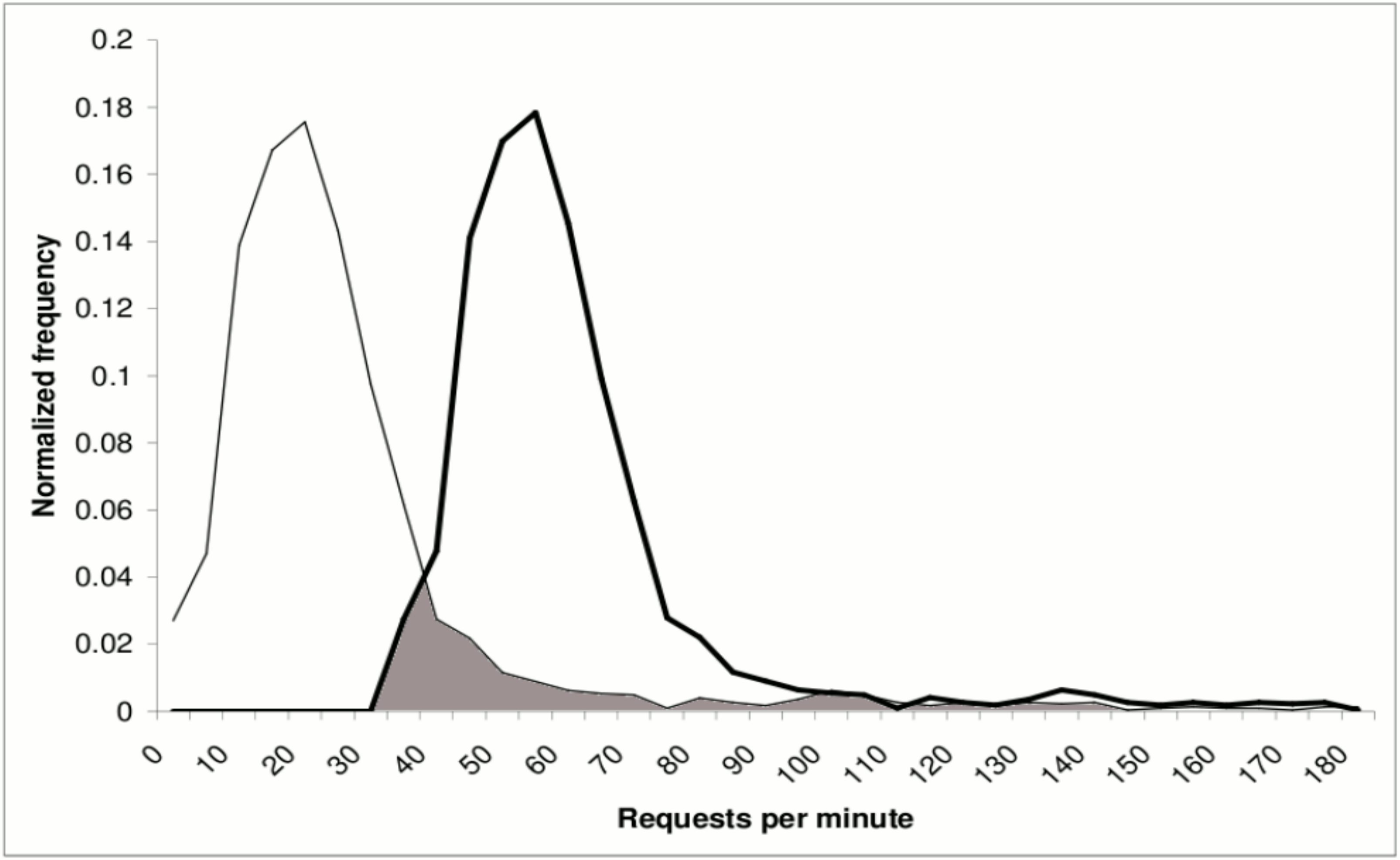
Histogram for Tor delays in seconds



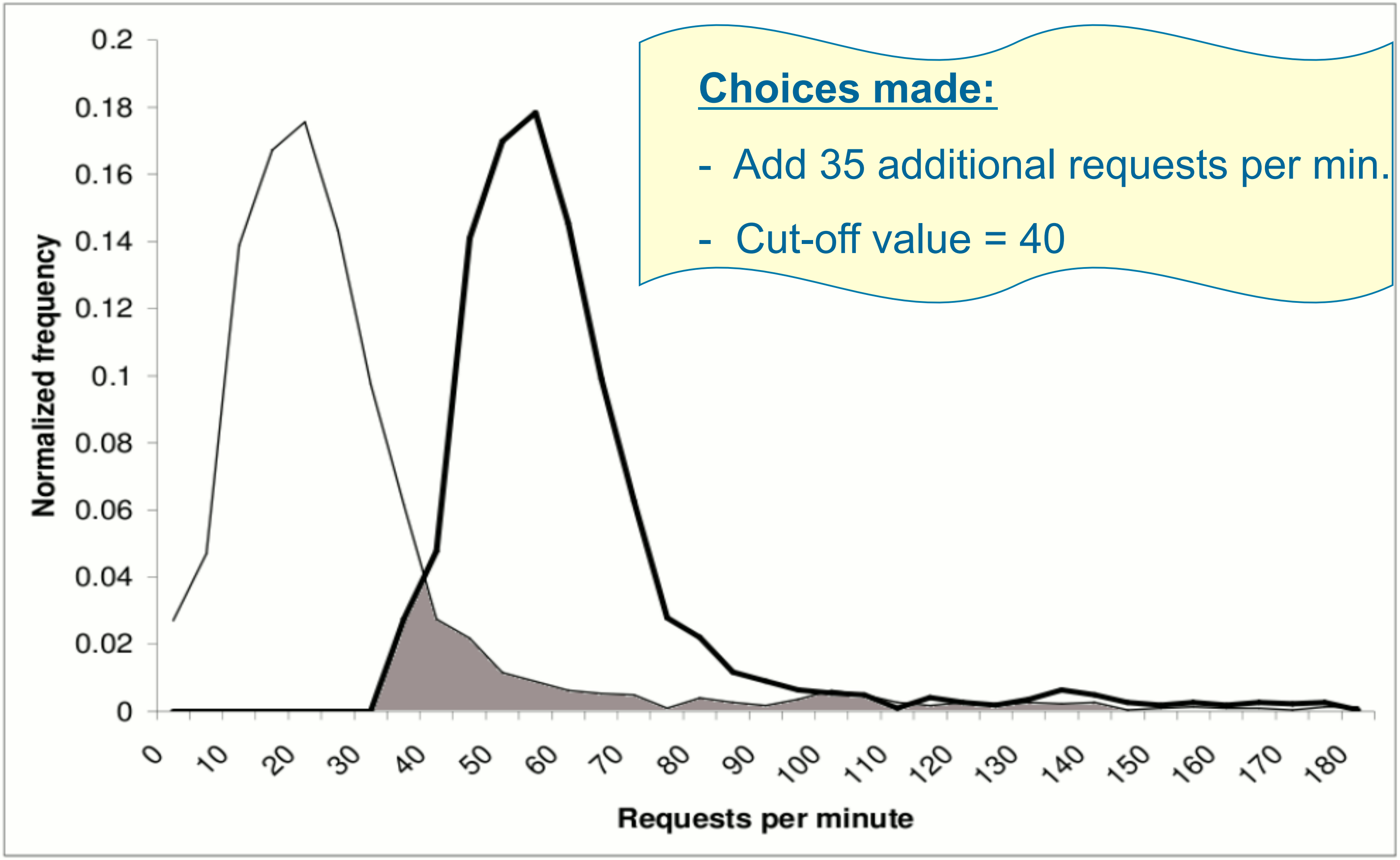
Histogram for existing HTTP GET requests per minute



Histogram for additional HTTP GET requests per minute



Histogram for additional HTTP GET requests per minute



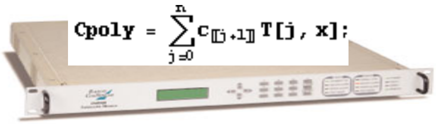
Choices made:

- Add 35 additional requests per min.
- Cut-off value = 40

36 bit

0	0	1	1	0	1	0	1

RS encoder



$$Cpoly = \sum_{j=0}^{n} c_{[[j+1]]}\, T[j, x];$$

60 bit

60 bit

$$Cpoly = \sum_{j=0}^{n} c_{[[j+1]]}\, T[j, x];$$

RS decoder

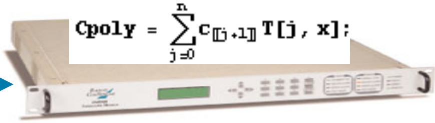
36 bit

0	0	1	1	0	1	0	1

36 bit

0 0 1 1 0 1 0 1

RS encoder



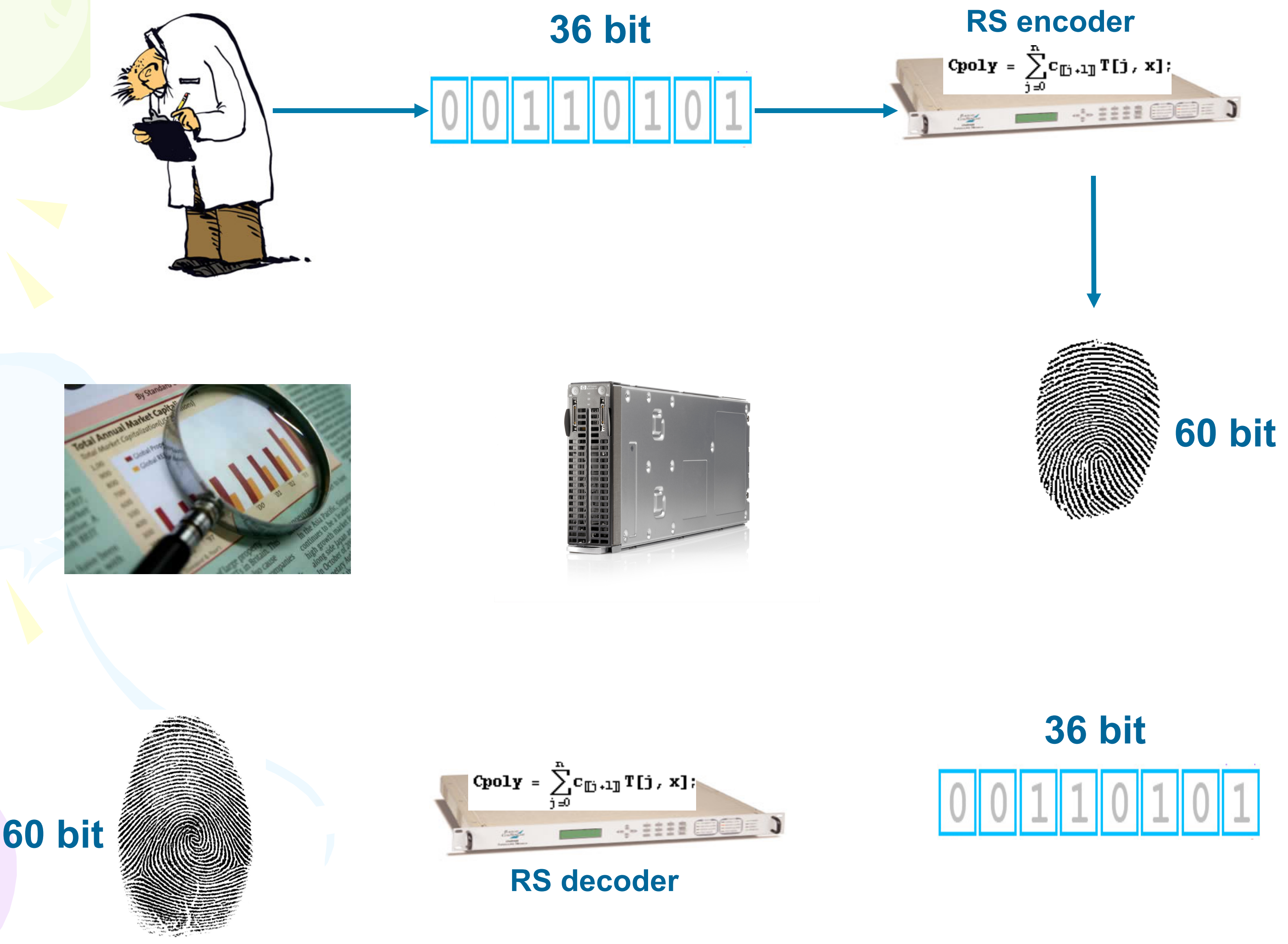
60 bit

60 bit

RS decoder

36 bit

0 0 1 1 0 1 0 1



36 bit

0 0 1 1 0 1 0 1

RS encoder

$\text{Cpoly} = \sum_{j=0}^{n} c_{[[j+1]]} T[j, x];$

60 bit

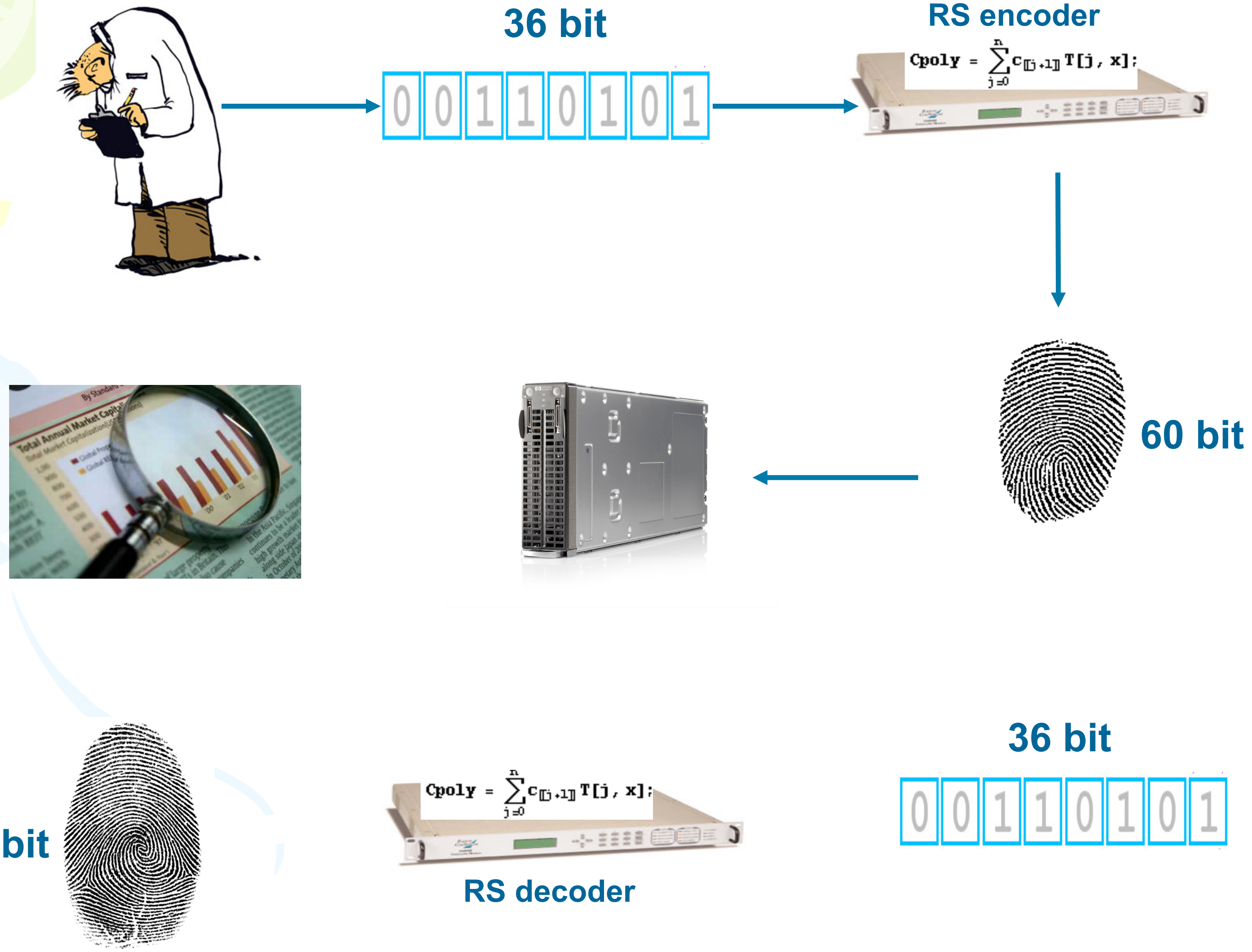
60 bit

$\text{Cpoly} = \sum_{j=0}^{n} c_{[[j+1]]} T[j, x];$

RS decoder

36 bit

0 0 1 1 0 1 0 1



36 bit

0 0 1 1 0 1 0 1

RS encoder

$Cpoly = \sum_{j=0}^{n} c_{[[j+1]]} T[j, x];$

60 bit

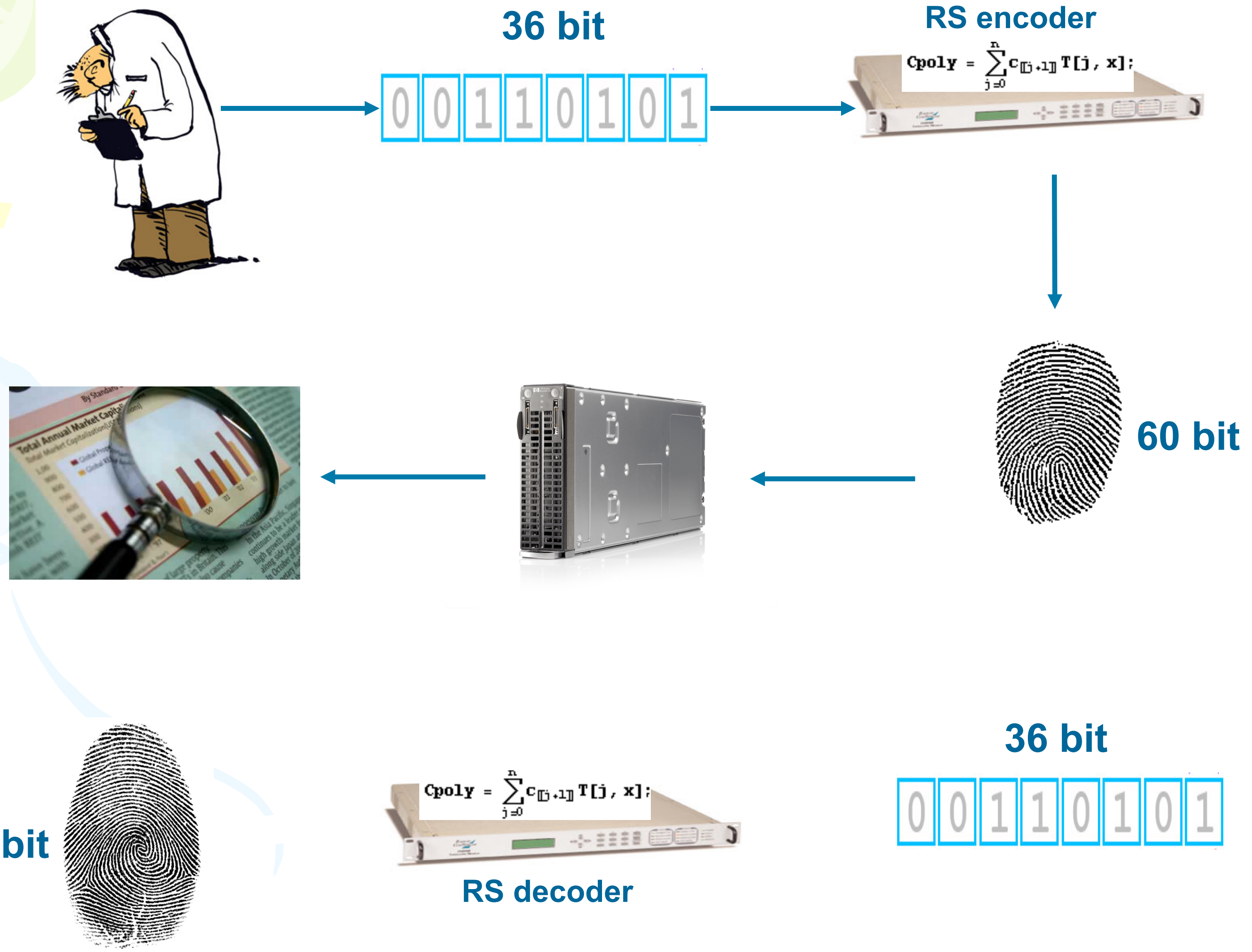
60 bit

$Cpoly = \sum_{j=0}^{n} c_{[[j+1]]} T[j, x];$

RS decoder

36 bit

0 0 1 1 0 1 0 1



36 bit

0 0 1 1 0 1 0 1

RS encoder

$Cpoly = \sum_{j=0}^{n} c_{[[j+1]]} T[j, x];$

60 bit

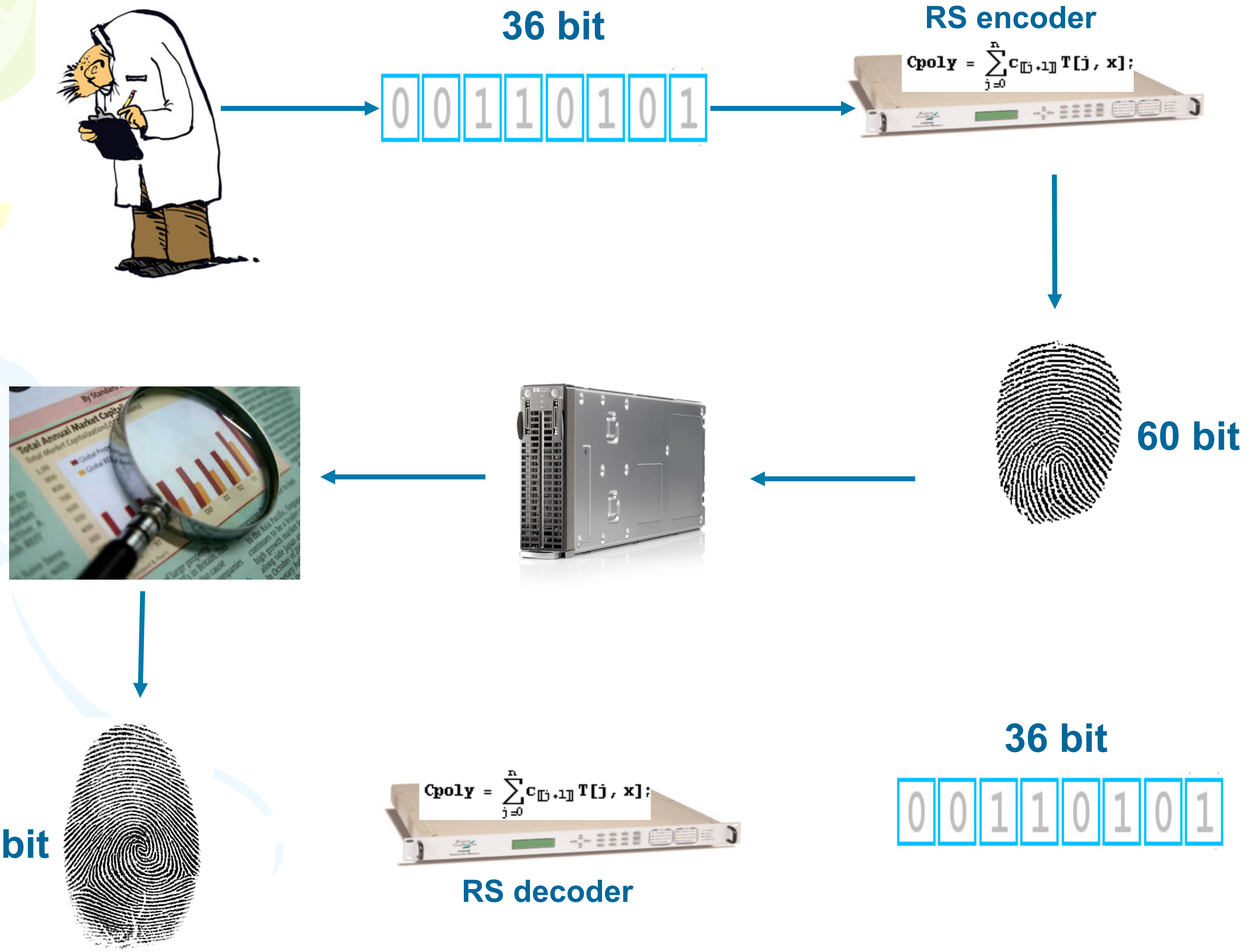
60 bit

$Cpoly = \sum_{j=0}^{n} c_{[[j+1]]} T[j, x];$

RS decoder

36 bit

0 0 1 1 0 1 0 1



36 bit

0 0 1 1 0 1 0 1

RS encoder

Cpoly = $\sum_{j=0}^{n} c_{[[j+1]]} T[j, x]$;

60 bit

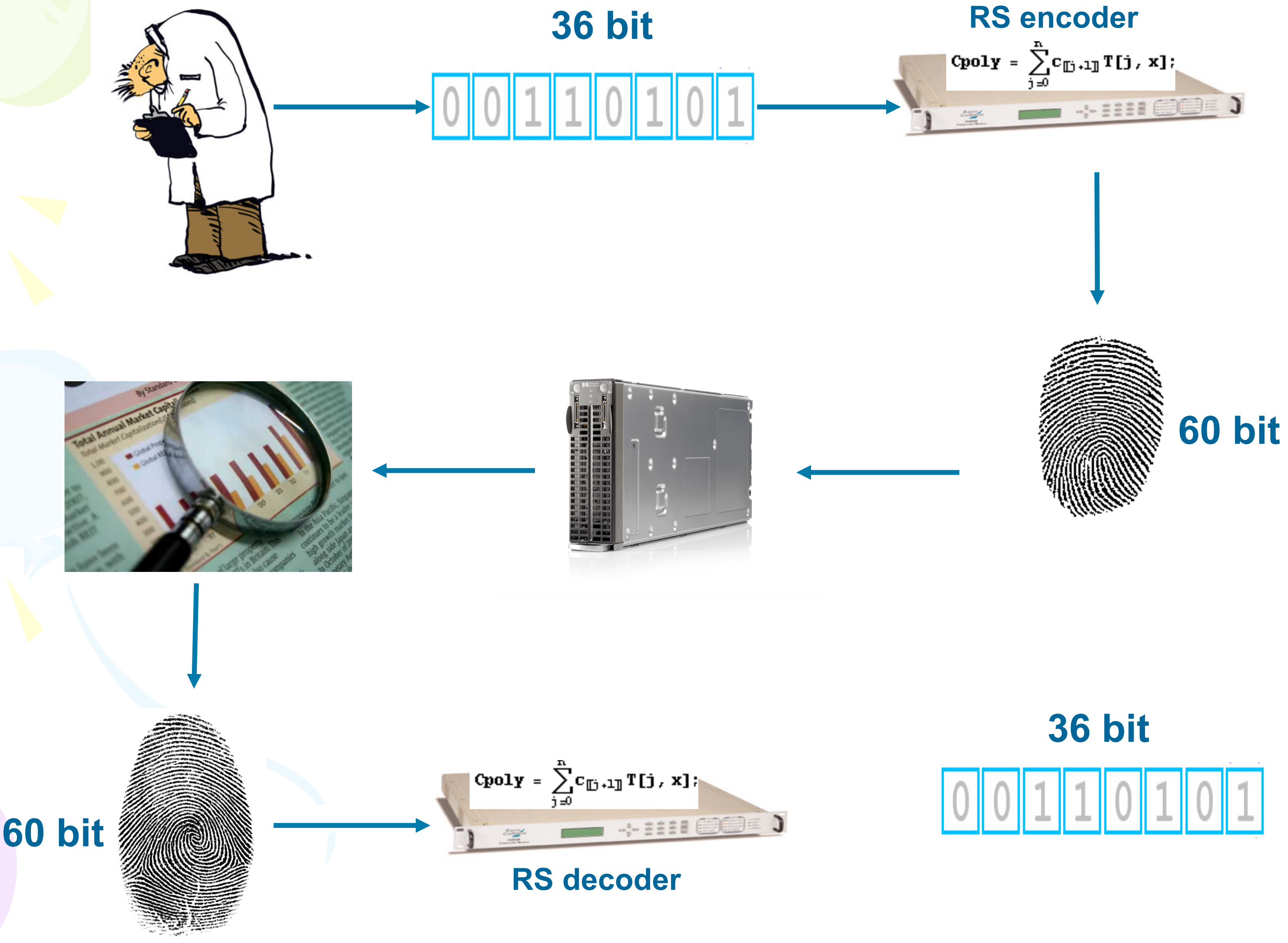
60 bit

Cpoly = $\sum_{j=0}^{n} c_{[[j+1]]} T[j, x]$;

RS decoder

36 bit

0 0 1 1 0 1 0 1



36 bit

0 0 1 1 0 1 0 1

RS encoder

$Cpoly = \sum_{j=0}^{n} c_{[[j+1]]} T[j, x];$

60 bit

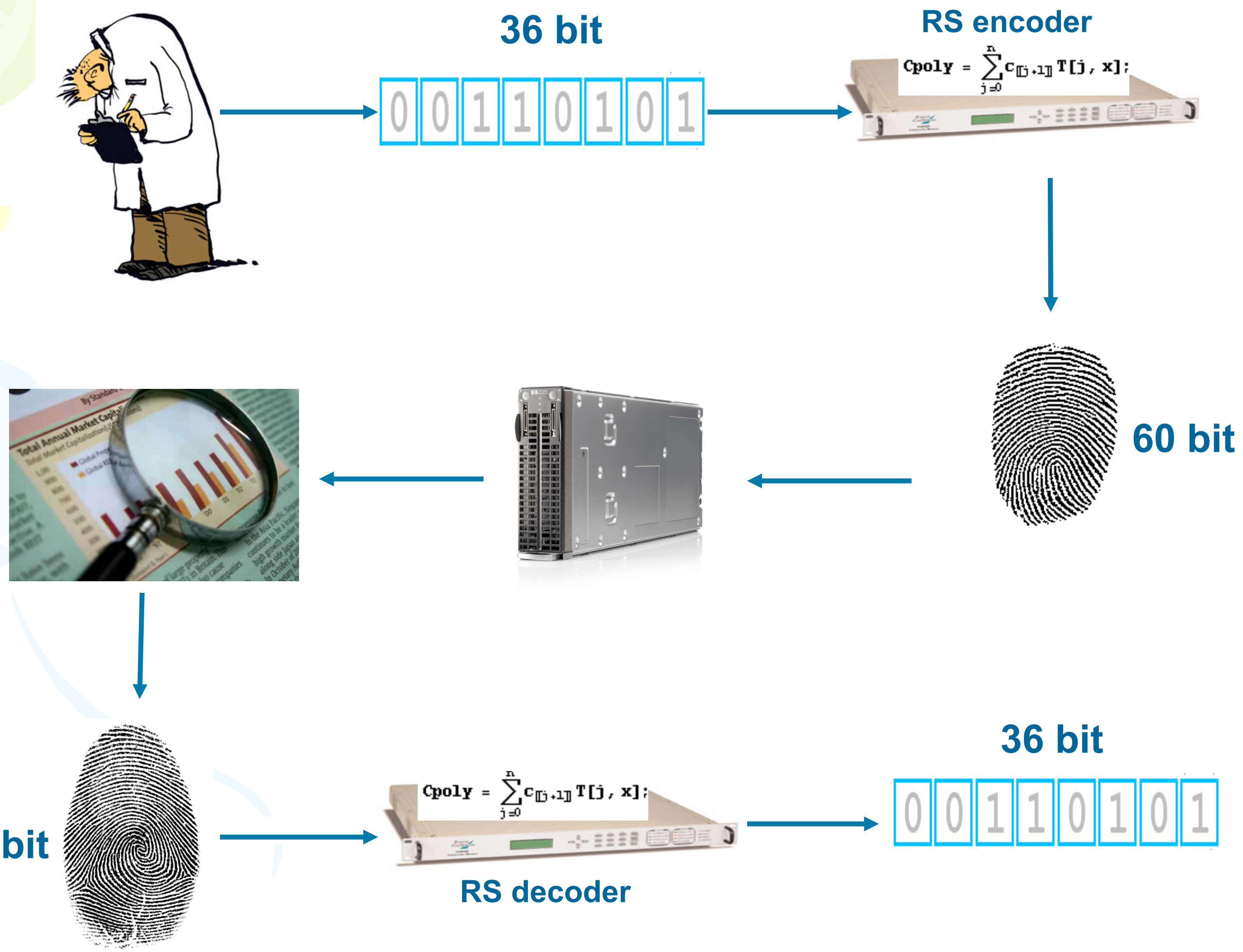
60 bit

$Cpoly = \sum_{j=0}^{n} c_{[[j+1]]} T[j, x];$

RS decoder

36 bit

0 0 1 1 0 1 0 1



36 bit

0 0 1 1 0 1 0 1

RS encoder

$$Cpoly = \sum_{j=0}^{n} c_{[[j+1]]} T[j, x];$$

60 bit

60 bit

$$Cpoly = \sum_{j=0}^{n} c_{[[j+1]]} T[j, x];$$

RS decoder

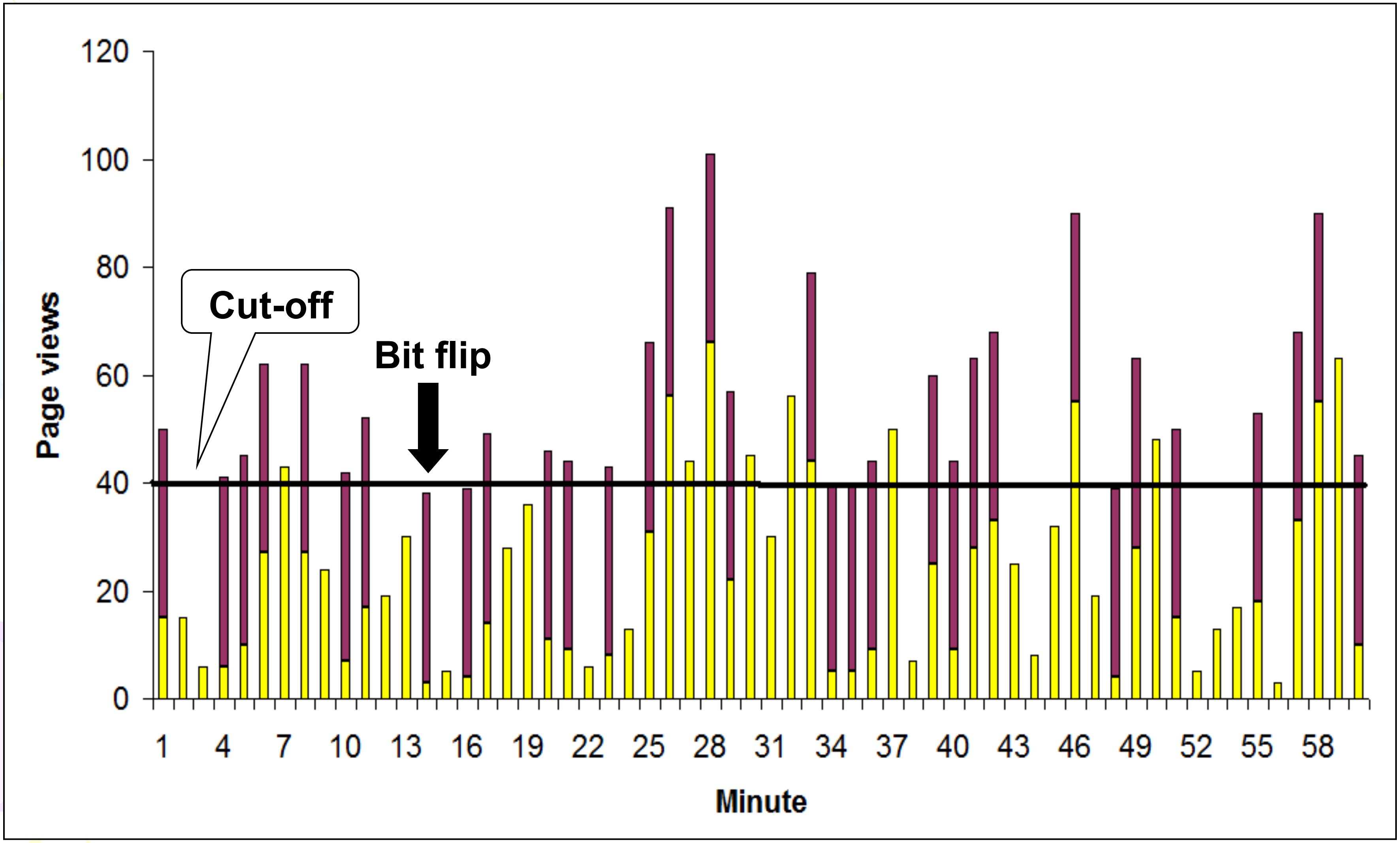
36 bit

0 0 1 1 0 1 0 1

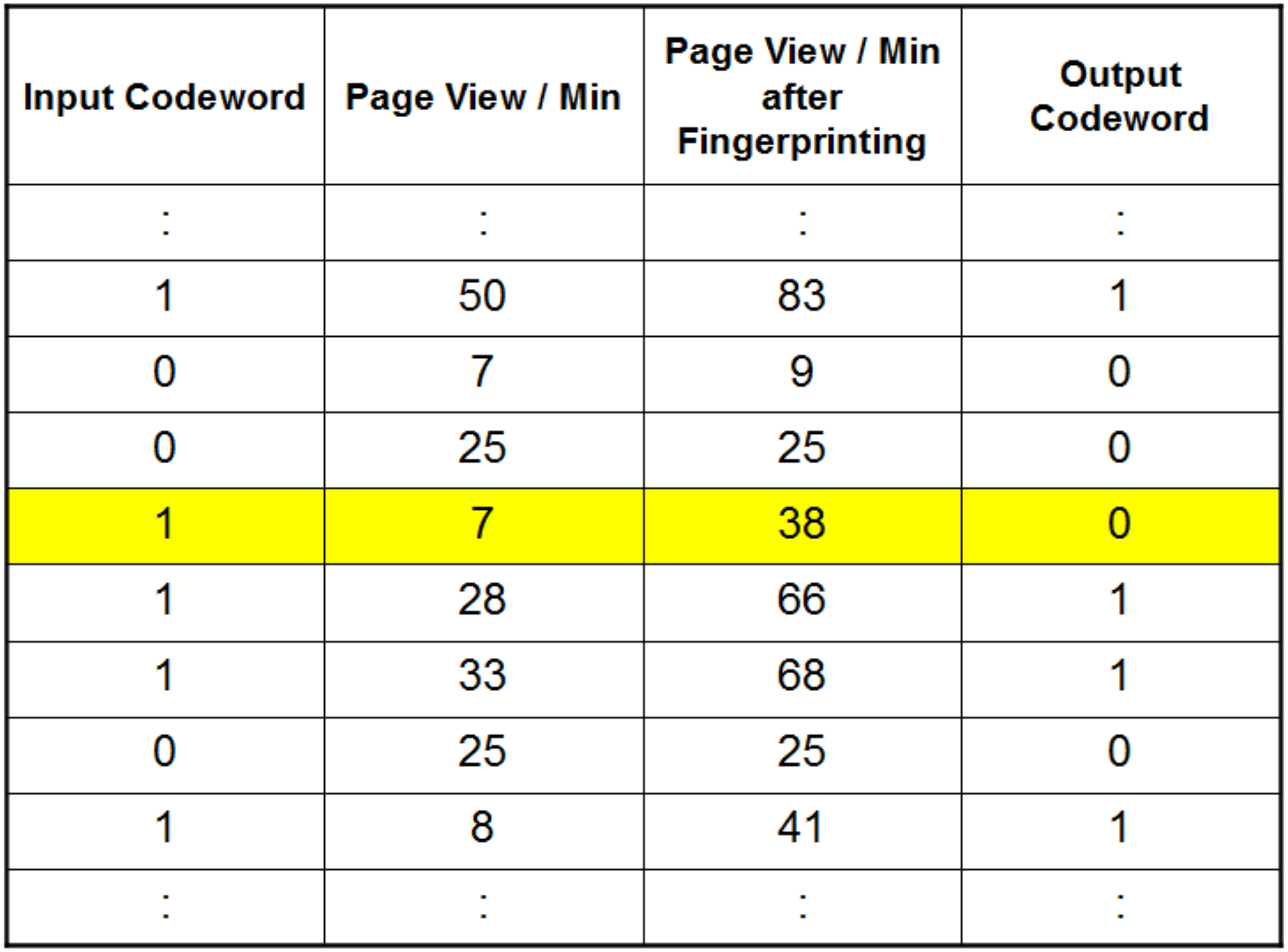
Fingerprinting Log file

- RS will produce a 60 bit codeword
- Each bit is given 60 sec
- If that bit is 0 then do nothing
- If that bit is 1 then do additional 35 requests to the hidden service
- Recover a 60 bit codeword from the log file (by comparing each minute to the cut-off value: 40 requests in our case)

How a 60-bit codeword appears in the log file

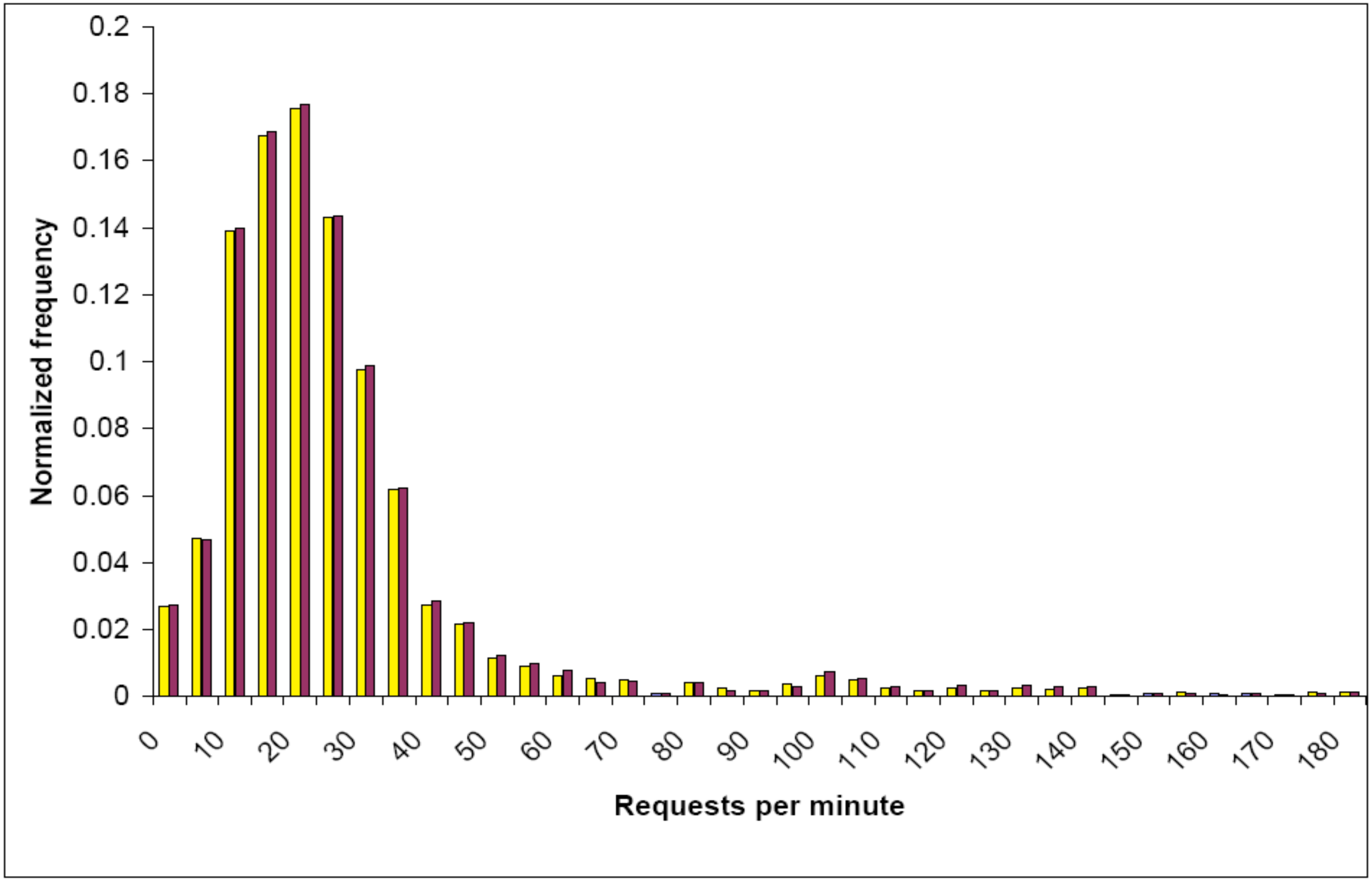


Details on a bit error from the example



Input Codeword	Page View / Min	Page View / Min after Fingerprinting	Output Codeword
:	:	:	:
1	50	83	1
0	7	9	0
0	25	25	0
1	7	38	0
1	28	66	1
1	33	68	1
0	25	25	0
1	8	41	1
:	:	:	:

Histograms with and without fingerprinting



Discussion

- Robust technique for leaving timing channel fingerprints in hidden service log files
- Ability to recover a 36-bit fingerprint with a 60-minute fingerprinting process
- Tradeoff in terms of how long it takes to leave a fingerprint vs. how much traffic must be added per minute
- We are not breaking the anonymity of Tor, our assumptions doesn't break in to Tor's main goal

Future Work

- Explore the tradeoffs in a stronger threat model
- Faster fingerprinting with less requests or additional timestamps
- Modeling the probability distribution of the network delays
- Time domain analysis of the gathered data will provide useful elements for the design of channel coding mechanisms

Acknowledgments

- NSF #0905201
- UNM CS support
  - George Kelbley
  - Jeff Bowles
- Tor Project People

NSF National Science Foundation
WHERE DISCOVERIES BEGIN

unm Computer SCIENCE

Tor
tor.eff.org