CSCI262 – System Security

Assignment 2

Timothy Mah

8750634

[txamah001@mymail.sim.edu.sg](mailto:txamah001@mymail.sim.edu.sg)

1 (a)

Puzzle A: m=1 , k=6

Columns needed =

1 possible combination for all 64 hash counts

Puzzle A: m=6 , k=4

Columns needed =

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hash** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** |
| Combi | 0 | 0 | 0 | 0 | 0 | 1 | 6 | 21 | 56 | 126 | 252 | 462 | 792 | 1287 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hash** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** |
| Combi | 2002 | 3003 | 4368 | 6188 | 8568 | 11628 | 15504 | 20343 | 26298 | 33523 | 42168 | 52374 | 64268 | 77958 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hash** | **29** | **30** | **31** | **32** | **33** | **34** | **35** | **36** | **37** | **38** | **39** | **40** | **41** | **42** |
| Combi | 93528 | 111033 | 130494 | 151893 | 175168 | 200208 | 226848 | 254864 | 283968 | 313818 | 344028 | 374178 | 403824 | 432508 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hash** | **43** | **44** | **45** | **46** | **47** | **48** | **49** | **50** | **51** | **52** | **53** | **54** | **55** | **56** |
| Combi | 459768 | 485148 | 508208 | 528534 | 545748 | 559518 | 569568 | 575688 | 577744 | 575688 | 569568 | 559518 | 545748 | 528534 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hash** | **57** | **58** | **59** | **60** | **62** | **63** | **64** | **65** | **66** | **67** | **68** | **69** | **70** |
| Combi | 508208 | 485148 | 459768 | 432508 | 374178 | 344028 | 313818 | 283968 | 254864 | 226848 | 200208 | 175168 | 151893 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hash** | **71** | **72** | **73** | **74** | **75** | **76** | **77** | **78** | **79** | **80** | **81** | **82** | **83** | **84** |
| Combi | 130494 | 111033 | 93528 | 77958 | 64268 | 52374 | 42168 | 33523 | 26298 | 20343 | 15504 | 11628 | 8568 | 6188 |

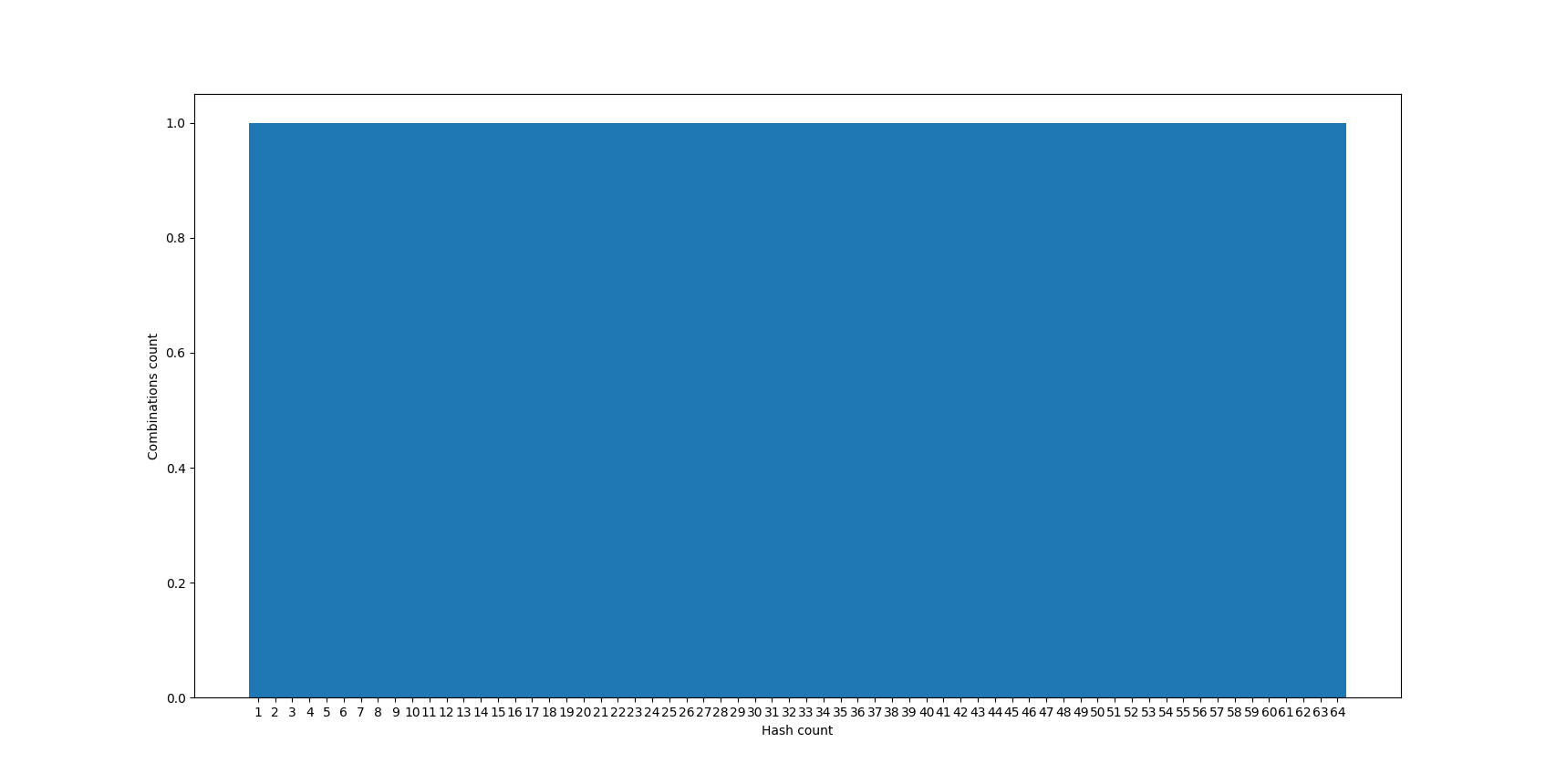
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hash** | **85** | **86** | **87** | **88** | **89** | **90** | **91** | **92** | **93** | **94** | **95** | **96** |
| Combi | 4368 | 3003 | 2002 | 1287 | 792 | 462 | 252 | 126 | 56 | 21 | 6 | 1 |

1 (b)

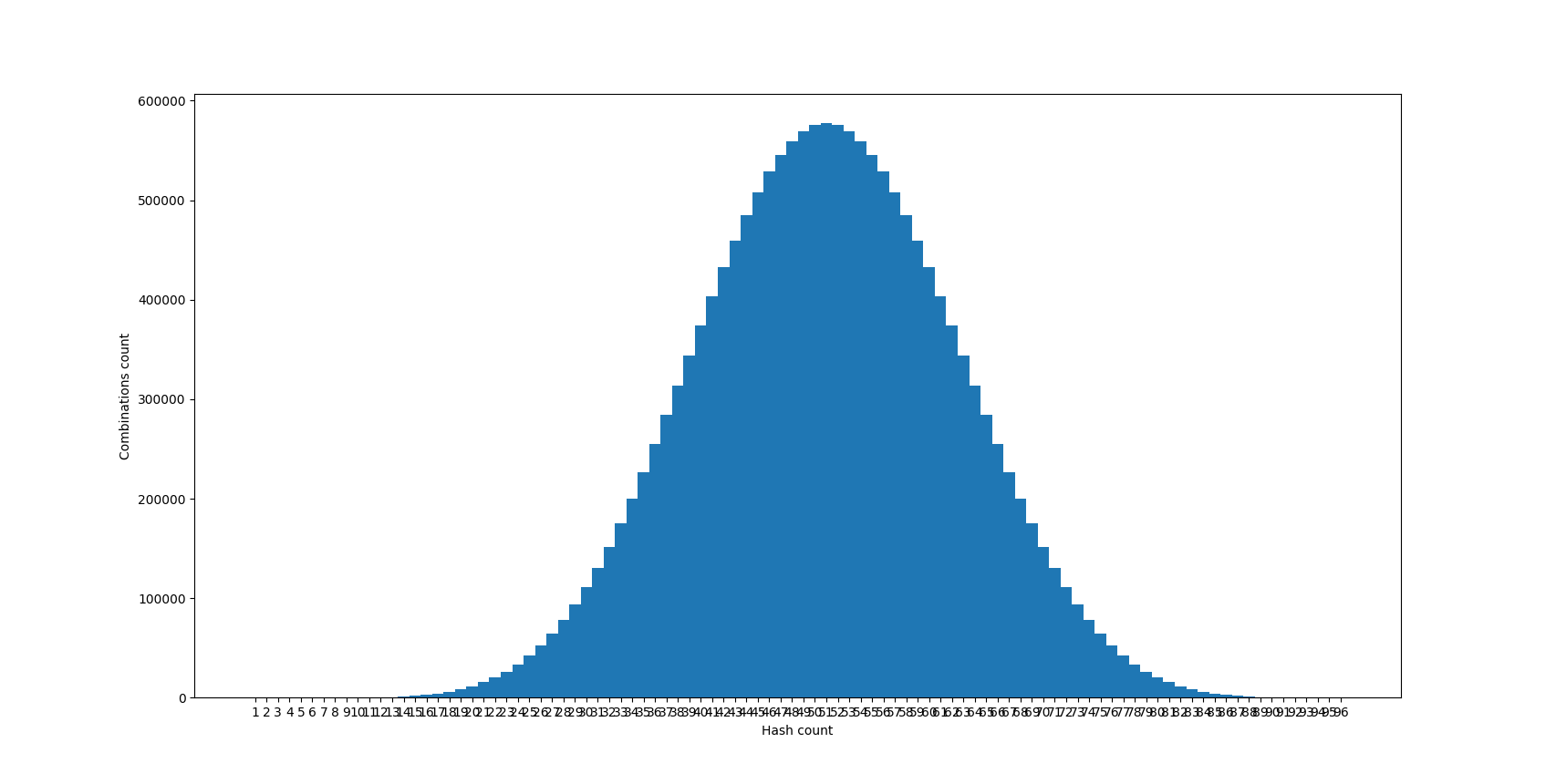
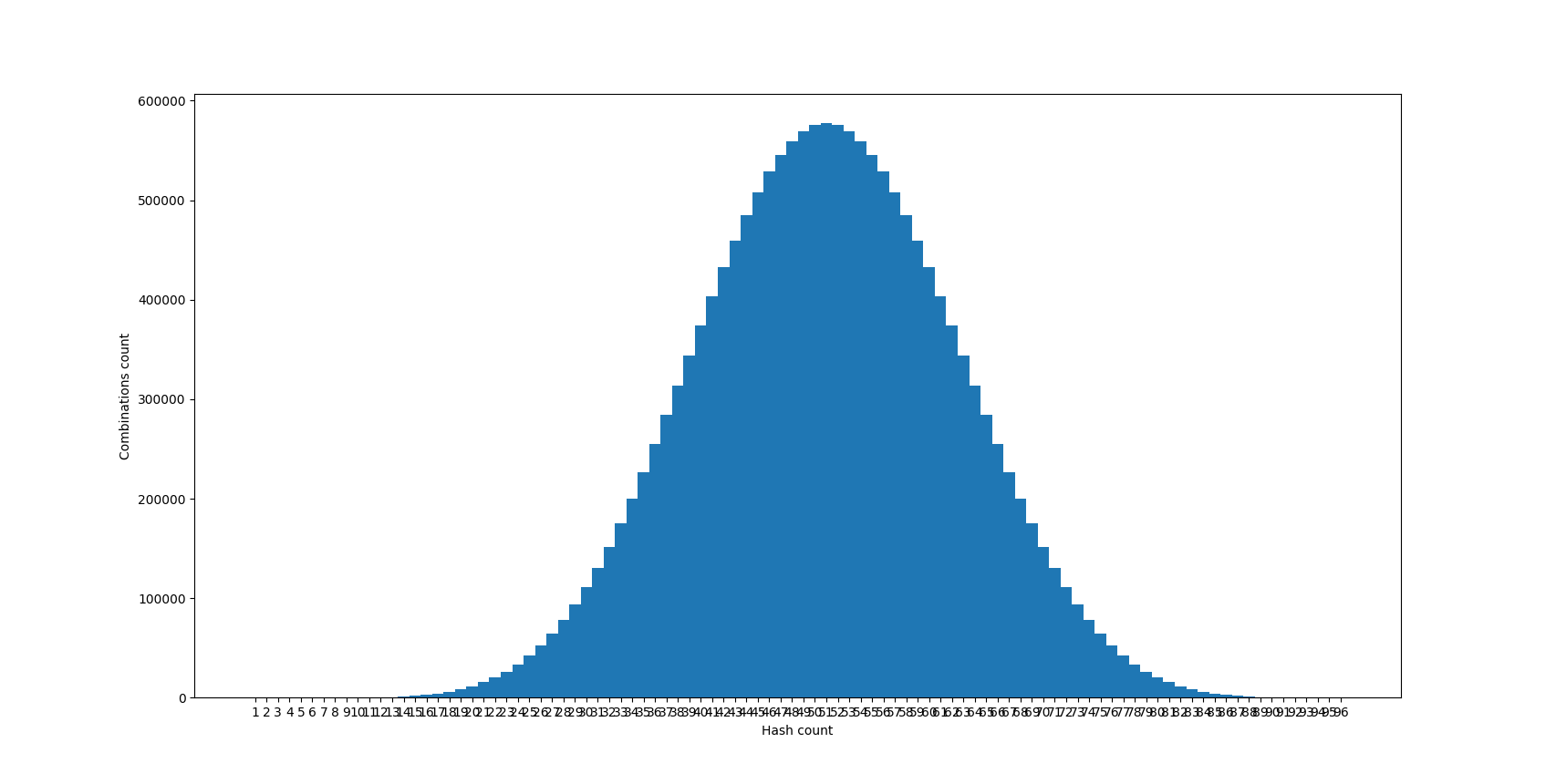
I used a python program that calculates these values. The app calculates how many combinations to reach a target value. Each number added reduces the target by that number and the process repeats until no more numbers are left to add. If at that point the sum equals the target, it counts as a valid combination. This is looped for each possible target sum from 1 up to maximum.

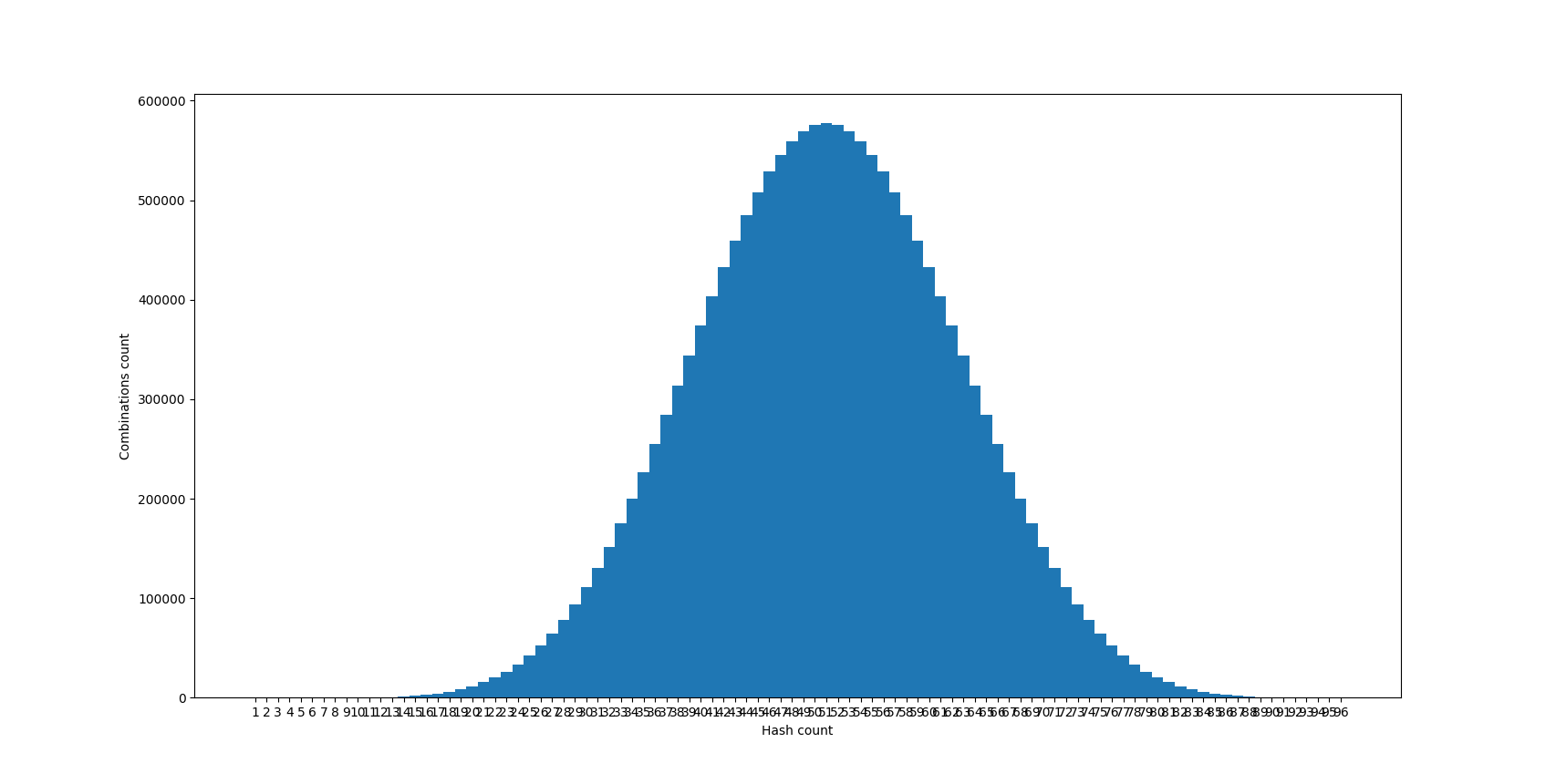
1 (c)

Puzzle A:



Puzzle B:





1 (d)

Puzzle A: m=1 k=6:

No. of columns (n) = = 64

Average = (n \* (n + 1)) / (2 \* n) = **32.5**

Puzzle B: m=6 k=4:

No. of columns (n) = = 96

Average = (n \* (n + 1)) / (2 \* n) = **48.5**

1(e)

Puzzle A: m=1 k=6:

No. of columns (n) = = 64

variance (v) = (- 1) / 12

SD = = **147.78**

Puzzle A: m=1 k=6:

No. of columns (n) = = 96

variance (v) = (- 1) / 12

SD = = **271.51**

2 (i)

Max req = 512 requests

Max time = 3 mins

Attack rate requirement = 512 / 3 ≈ **171 requests/min**

2 (ii)

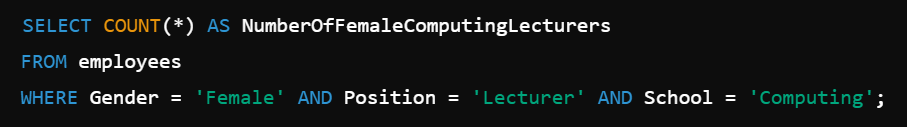
Packet size = 64 bytes = 512 bits

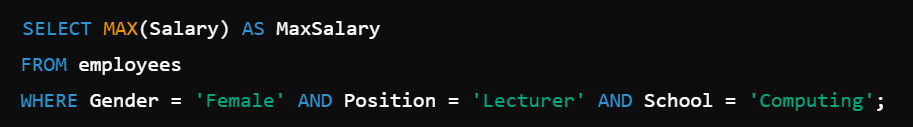
1 request = 1 packet

Attack rate = 171 requests/min = 171 requests/60 seconds ≈ 2.85 requests/second

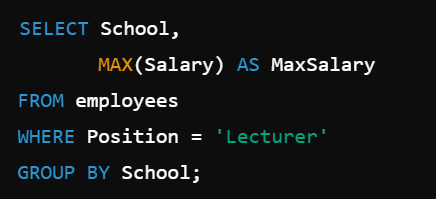
Bandwidth consumed = 2.85 x 512 = **1459.2 bits/ second**

3 (a) <minimum number of rows returned is 2>





3(b)



4(a)

SYN cookies store information about a TCP handshake. This allows the server to drop the connection and continue when the client is ready to resume the connection. By doing so, the server does not hog resources waiting for the client’s reply. By extension this mitigates DOS attacks

Client puzzles are sent to clients by a server to slow down the connection in a reasonable time and serves as liveliness probe. These two features mitigate DOS attacks

As stated, both are similar in their purpose of mitigating DOS attacks. They are different in their approach since for SYN cookies, the server gives a “cookie” back to the client while for client puzzles, the server gives a puzzle to the client. They are different in application since SYN cookies work on the network layer while client puzzles work on application layer.

4(b)

Main property of client puzzles is computationally taxing without sacrificing user experience.

When client puzzles are computationally taxing, it takes longer for a client to connect to a server. This mitigates DOS by increasing the time taken to crash the server. And even more so DDOS. Even if the connection delay may be seemingly small, in a distributed attack, the delay scales up proportionately to the attack. The client puzzle also serves as a liveliness probe so that zombie botnets cannot connect to the server.

However at the same time, the computation difficulty cannot be too taxing else legitimate users would take a long time to connect to the server. This degrades user experience thus there needs to be a balance.

5(a)

Infected =

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hour** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| Infected | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hour** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** |
| Infected | 8192 | 16384 | 32768 | 65536 | 131072 | 262144 | 524288 | 1048576 | 2097152 | 4194304 | 8388608 | 16777216 |

5(b)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hour** | **6.5** | **7** | **7.5** | **8** | **8.5** | **9** | **9.5** | **10** | **10.5** | **11** | **11.5** | **12** | **12.5** | **13** | **13.5** | **14** |
| X infected | 63 | 126 | 124 | 248 | 242 | 484 | 466 | 932 | 878 | 1756 | 1594 | 3188 | 2702 | 5404 | 3946 | 7892 |

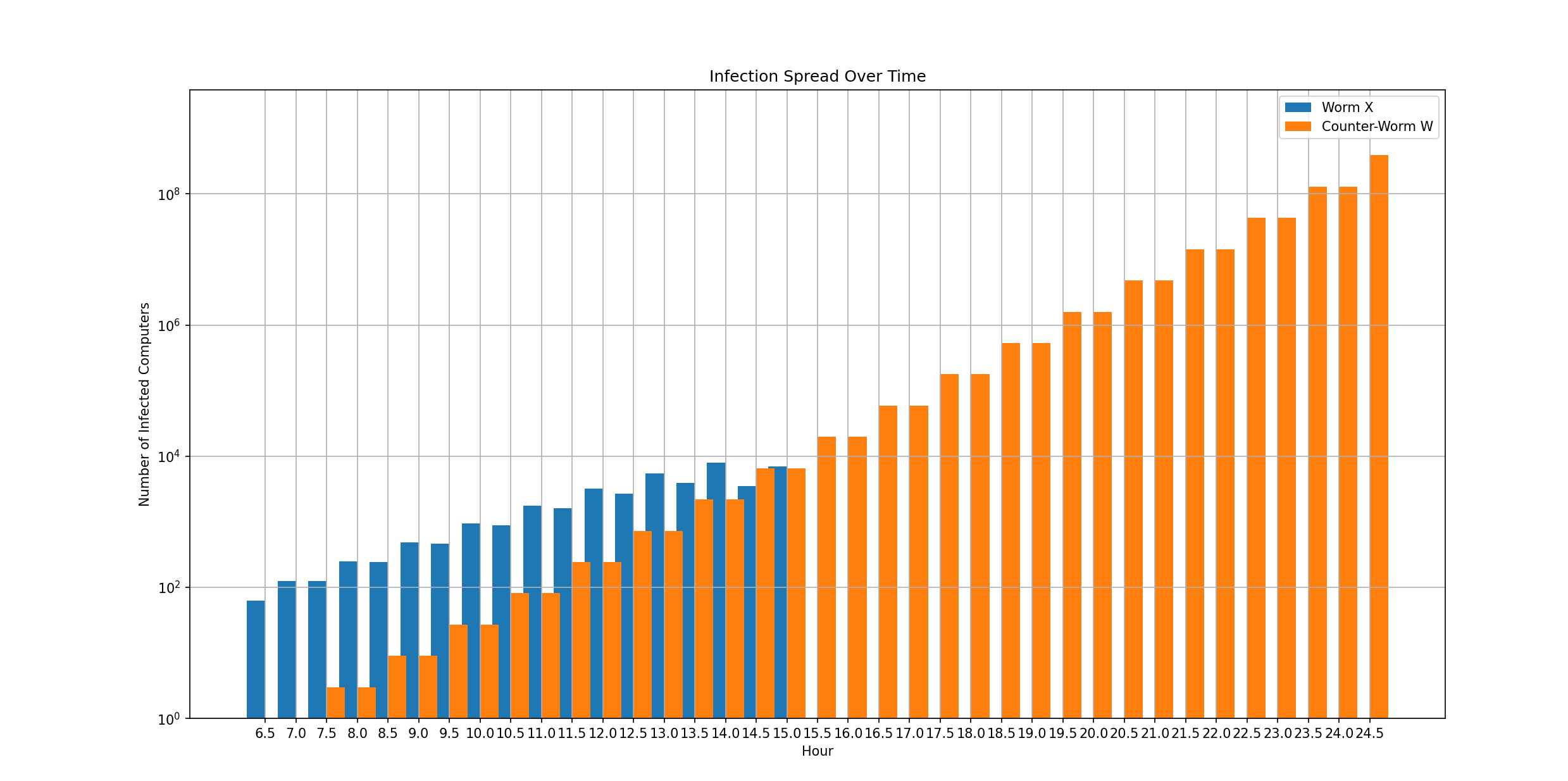
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hour** | **14.5** | **15** | **15.5** | **16** | **16.5** | **17** | **17.5** | **18** | **18.5** | **19** | **19.5** | **20** | **20.5** | **21** | **21.5** | **22** | **22.5** | **23** | **23.5** | **24** |
| X infected | 3518 | 7036 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hour** | **6.5** | **7** | **7.5** | **8** | **8.5** | **9** | **9.5** | **10** | **10.5** | **11** | **11.5** | **12** | **12.5** | **13** | **13.5** | **14** | **14.5** | **15** | **15.5** | **16** |
| W infected | 1 | 1 | 3 | 3 | 9 | 9 | 27 | 27 | 81 | 81 | 243 | 243 | 729 | 729 | 2187 | 2187 | 6561 | 6561 | 19683 | 19683 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hour** | **16.5** | **17** | **17.5** | **18** | **18.5** | **19** | **19.5** | **20** | **20.5** | **21** | **21.5** | **22** | **22.5** | **23** | **23.5** | **24** |
| W infected | 59049 | 59049 | 177147 | 177147 | 531441 | 531441 | 1.59432e+06 | 1.59432e+06 | 4.78297e+06 | 4.78297e+06 | 1.43489e+07 | 1.43489e+07 | 4.30467e+07 | 4.30467e+07 | 1.2914e+08 | 1.2914e+08 |

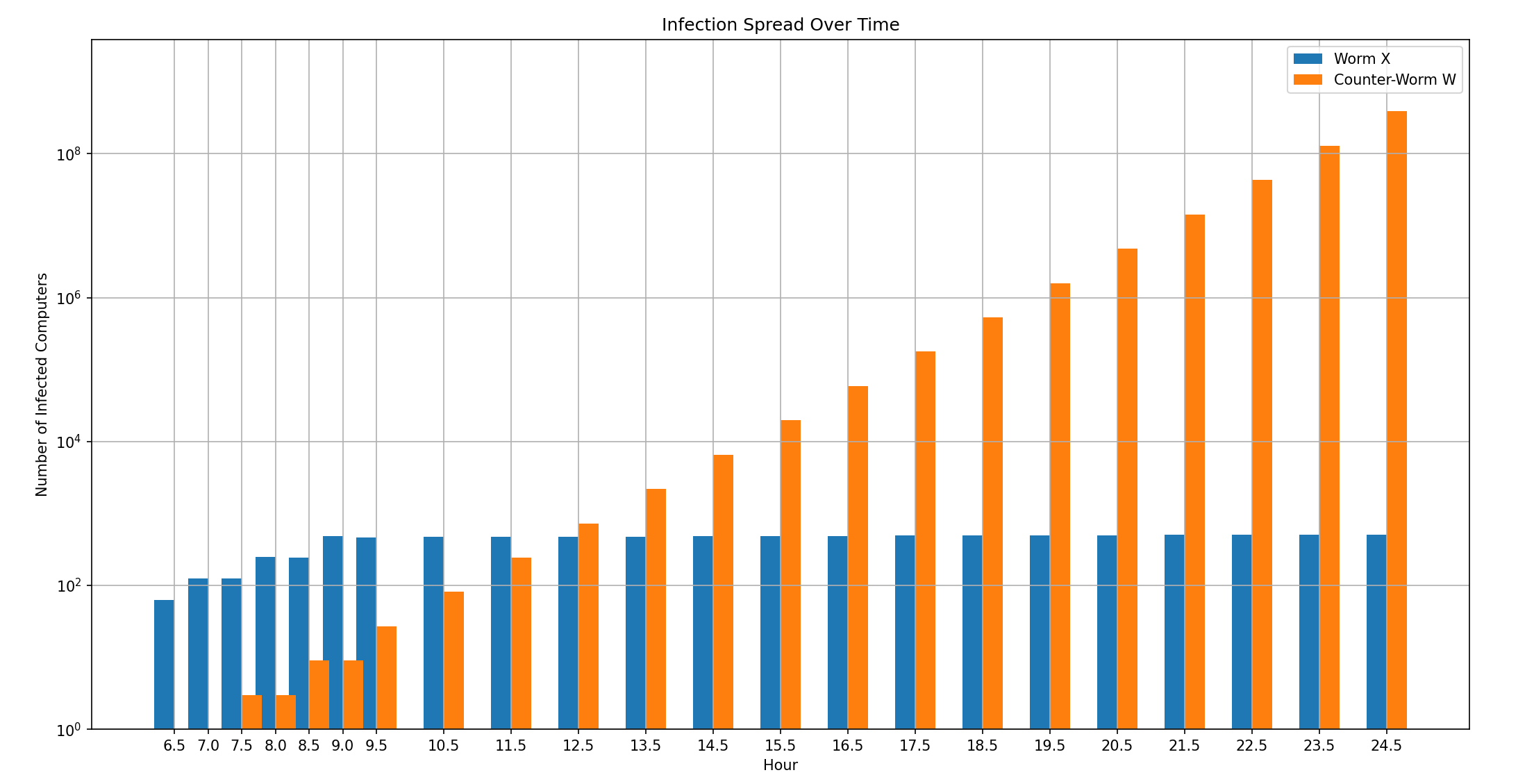
5(c)

the number of 𝑋 infections will increase on the hour and decrease on the half-hour after 𝑡 = 6.5 . Hence at hour 15.5 there are 0 𝑋 infections



5(d)

The number of x infections would stagnate



6

Type of information to gather: access logs, error logs, hardware metrics, network activity. Access logs to know if any sensitive data is being access, and by who, unusually. Error logs to know if any unusual errors have occurred from possible intrusion. Hardware metrics to detect DOS attacks. Network activity to know if any unusual data is coming inward or going outward

When or where you would gather the information: application layer metrics, hardware metrics from OS. Constantly gather information. Applications can output their own access logs and error logs. OS can output hardware utilisation. Network logs file format is determined by the Firewall solution used. Constantly gather information at small intervals such as 5 seconds to ensure real time logging

The format you would store the information in and where this information would be stored: all logs can be stored as JSON or CSV to be ingested by observability tools. Can be stored locally within the public transport system and replicated in cloud, within a different availability zone and another in different region, for redundancy

The types of problems you would look for in analysing the data: Unauthorised access, DOS, sensitive data exfiltration, access to suspicious websites

How you would identify some of those problems based on the storage: Based on access logs, can determine if any user has accessed files that are not expected. This identify unauthorised access. Based on hardware metrics, can determine if DOS is occurring. Based on network logs, can determine if data exfiltration and access to suspicious websites has occurred by looking at destination address