**Project 4 – Cryptographic Hash**

**Part 1: Literature Review**

Merkle Trees:

* Kuznetsov, O., Rusnak, A., Yezhov, A., Kuznetsova, K., Kanonik, D., & Domin, O. (2024). *Merkle Trees in Blockchain: A Study of Collision Probability and Security Implications.* <http://arxiv.org.proxy.library.nyu.edu/abs/2402.04367>
  + Summary: The research paper assesses the security through probability of hash collisions in Merkle Trees that underpins blockchain systems today by summarizing all transactions in a block and Merkle Proofs used to efficiently verify data without accessing the entire Merkle Tree. Through theoretical analysis, review of the birthday strategy paradox, experimentation through python code to estimate probabilities the research found increasing hash length decreases collisions, increasing Merkle Tree path length increases collisions, and emphasizes the decision tradeoffs between efficient computing and security for varying parameters.
* Paris, J.-F., & Schwarz, T. (2020). Merkle Hash Grids Instead of Merkle Trees. 2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 2020 28th International Symposium On, 1–8. <https://doi-org.proxy.library.nyu.edu/10.1109/MASCOTS50786.2020.9285942>
  + Summary: The research paper explores Merkle Grids that organize data in a square array to provide similar function to Merkle Trees in a more efficient way (e.g., reduces transmission and storage costs by 50%). The research concludes that extended Merkle grids, which incorporates two added auxiliary Merkle Trees to basic Merkle grid, require much less space than Merkle trees and perform as well or better in explored cases, such as locating and authenticating non-conforming objects.

Hash Collisions:

* Smith, P., Sarkar, S., Kasera, S., & Patwari, N. (2024). On Passive Privacy-Preserving Exposure Notification Using Hash Collisions. IEEE Internet of Things Journal, 1-1–1. <https://doi-org.proxy.library.nyu.edu/10.1109/JIOT.2024.3353255>
  + Summary: The research paper explores a COVID-19 contact tracing solution based on users cellphone proximity to WiFi access points that employs a hash collision filter to allow users to lookup proximity to positive test cases while maintaining privacy and security of sensitive identifying information, such as BSSID and SSID of WiFi access points. The servers stores obfuscated group ID and if a user matches the group ID, they can determine based on RSS values if close proximity to known positive test user reported by a health care provider. There are collisions (e.g., same hashes) across various group IDs and RSS values by design but would not necessarily indicate a true match. The experiment implementation and threat analysis shows effectiveness against information leaks.

Hash Puzzles:

* Ali, I. M., Caprolu, M., & Di Pietro, R. (2021). Foundations, Properties, and Security Applications of Puzzles: A Survey. ACM Computing Surveys, 53(4), 72–72:38. <https://doi-org.proxy.library.nyu.edu/10.1145/3396374>
  + Summary: The research paper examines various types of hash puzzles that serve many purposes, such as proof-of-work support cryptocurrencies like Bitcoin. The paper defines a puzzle as moderately hard problems that are easier to verify than solve, resulting in use of resources to solve the puzzle which yields additional time and cost. The paper looks at three high-level macro types of puzzles: application, resource (CPU, memory), and verification (explicit, implicit). For example, the paper discussed four types of application puzzles: pricing puzzle (to address low-cost of using a service and price-deterrent to large-scale automated attacks), delaying puzzle (time dependency to solve), timing puzzle (computations with increasingly large efforts to meter traffic), and AI-hard puzzles (hard for a computer to solve but humans could) which feature elements of varying computation, timing, and human interaction. For application puzzles the paper explores feasibility and applicability to different use cases through evaluation of nine different requirements: Ease to construct, fine-grained, deterministic computation, non-parallelizable, non-interactive, publicly verifiable, stateless, trapdoorless, fair. For example when applied to cryptocurrencies the key criteria for a puzzle are easy to construct, non-interactive, publicly verifiable, stateless, and trapdoorless (the issuer does not have advantage to solve or forge proof). The research concludes there are benefits and limitations to each type of puzzle and whether a type of puzzle is right depends on the specific use case applied, whether different application, resource, and verification type puzzles.

**Part 2: Discussion Questions**

1. Explain why hash collisions are a mathematical inevitability.

Hash collisions are a mathematical inevitability due to input data of infinite amount into hash function that produces a fixed length output. Once the input data length (bits) vastly exceeds the hash output length, it is only a matter of time before a hash collision occurs. However, the longer the hash length (e.g., 128 bits as opposed to 32 bits), the lower the risk of hash collisions.

1. Considering a room with N people, including Trudy, what's the probability that at least one other person shares Trudy's birthday? At what minimum N does this probability exceed 50%?

There are 365 days in a year, so each additional person selected must not have the same birthday as prior selections multiplied by each successive selection options, so ((365)\*(364)\*…(365-n+1)). This is divided by the total numbers of days in the year n times representing the n selections, so 365^n. Then take the n person selections calculated first ((365)\*(364)\*…(365-n+1)) and divide by the 365 days n times (365^n) to get the probability of not selecting a person with the same birthday as any prior selection. As we are looking for at least one person with the same birthday of Trudy we want the inverse, so probability of same birthday as Trudy is 1 – (((365)\*(364)\*…(365-n+1))/365^n).

Formula: probability of n with same birthday as Trudy = 1 – (((365)\*(364)\*…(365-n+1))/365^n)

At a minimum N of 23 results in a probability of greater than 50% of selecting a person with the same birthday as Trudy (50.729723% probability).

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **Numerator**  ((365)\*(364)\*…(365-n+1)) | **Denominator**  (365^n) | **p(Bday same as Trudy)**  1-(Numerator/Denominator), then converted to % by multiplying by 100. |
| 1 | 365 | 365 | 0.000000% |
| 2 | 132860 | 133225 | 0.273973% |
| 3 | 48228180 | 48627125 | 0.820417% |
| 4 | 17458601160 | 17748900625 | 1.635591% |
| 5 | 6.30256E+12 | 6.47835E+12 | 2.713557% |
| 6 | 2.26892E+15 | 2.3646E+15 | 4.046248% |
| 7 | 8.14542E+17 | 8.63078E+17 | 5.623570% |
| 8 | 2.91606E+20 | 3.15023E+20 | 7.433529% |
| 9 | 1.04103E+23 | 1.14984E+23 | 9.462383% |
| 10 | 3.70608E+25 | 4.1969E+25 | 11.694818% |
| 11 | 1.31566E+28 | 1.53187E+28 | 14.114138% |
| 12 | 4.65743E+30 | 5.59132E+30 | 16.702479% |
| 13 | 1.64407E+33 | 2.04083E+33 | 19.441028% |
| 14 | 5.78714E+35 | 7.44904E+35 | 22.310251% |
| 15 | 2.03129E+38 | 2.7189E+38 | 25.290132% |
| 16 | 7.1095E+40 | 9.92398E+40 | 28.360401% |
| 17 | 2.48122E+43 | 3.62225E+43 | 31.500767% |
| 18 | 8.63463E+45 | 1.32212E+46 | 34.691142% |
| 19 | 2.99622E+48 | 4.82575E+48 | 37.911853% |
| 20 | 1.03669E+51 | 1.7614E+51 | 41.143838% |
| 21 | 3.57658E+53 | 6.4291E+53 | 44.368834% |
| 22 | 1.23034E+56 | 2.34662E+56 | 47.569531% |
| 23 | 4.22008E+58 | 8.56517E+58 | 50.729723% |

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **Numerator** | **Denominator** | **p(Bday same as Trudy)**  Note: Conversion to % by multiplying by 100 not shown in formula below |
| 1 | 365 | =365 | =1-(P14/Q14) |
| 2 | =P14\*C14 | =Q14\*365 | =1-(P15/Q15) |
| 3 | =P15\*C15 | =Q15\*365 | =1-(P16/Q16) |
| 4 | =P16\*C16 | =Q16\*365 | =1-(P17/Q17) |
| 5 | =P17\*C17 | =Q17\*365 | =1-(P18/Q18) |
| 6 | =P18\*C18 | =Q18\*365 | =1-(P19/Q19) |
| 7 | =P19\*C19 | =Q19\*365 | =1-(P20/Q20) |
| 8 | =P20\*C20 | =Q20\*365 | =1-(P21/Q21) |
| 9 | =P21\*C21 | =Q21\*365 | =1-(P22/Q22) |
| 10 | =P22\*C22 | =Q22\*365 | =1-(P23/Q23) |
| 11 | =P23\*C23 | =Q23\*365 | =1-(P24/Q24) |
| 12 | =P24\*C24 | =Q24\*365 | =1-(P25/Q25) |
| 13 | =P25\*C25 | =Q25\*365 | =1-(P26/Q26) |
| 14 | =P26\*C26 | =Q26\*365 | =1-(P27/Q27) |
| 15 | =P27\*C27 | =Q27\*365 | =1-(P28/Q28) |
| 16 | =P28\*C28 | =Q28\*365 | =1-(P29/Q29) |
| 17 | =P29\*C29 | =Q29\*365 | =1-(P30/Q30) |
| 18 | =P30\*C30 | =Q30\*365 | =1-(P31/Q31) |
| 19 | =P31\*C31 | =Q31\*365 | =1-(P32/Q32) |
| 20 | =P32\*C32 | =Q32\*365 | =1-(P33/Q33) |
| 21 | =P33\*C33 | =Q33\*365 | =1-(P34/Q34) |
| 22 | =P34\*C34 | =Q34\*365 | =1-(P35/Q35) |
| 23 | =P35\*C35 | =Q35\*365 | =1-(P36/Q36) |

1. In a room of N people (N ≤ 365), what's the probability of any two sharing a birthday, and what's the minimum N for this probability to be over 50%?

See formula and calculation in response 2 above. As N approaches closer to 365, the probability increases towards 100%. For example, when selecting a 365th person there is a 0.2739726027% probability that person does not have a similar birthday as the prior 364 selections, with this example selection probability being dependent on the first 364 people all not having the same birthday, which is near 0%.

Similar to the calculation for Trudy above the minimum N is 23 for the probability of selecting N people with any two sharing a birthday to exceed 50%. If N were 366 there would have to be at least two sharing the a birthday at a minimum.

1. Describe the principle of the birthday attack on hashing and how it offers efficiency over brute-force attacks.

The principle of the birthday attack on hashing is similar in that if you select a number of data inputs (for hash operations) increasingly closer to the total possible options (e.g., 365 days for possible birthdays only so many combinations) the probability of hash collisions (different input producing same hash) increases. The goal of the birthday attack in hashing is to select two different inputs that result in the same hash value output. If doing brute force attack would need to compute more hashes to find a collision, if you follow the birthday attack method it will take computing fewer hashing operations to get a collision and increased efficiency due to taking advantage of the birthday attack mathematics and increased probability of finding hash collisions sooner.

1. Discuss the main issues associated with hash functions created using the Merkle-Damgård Construction process.

Merkle-Damgård Construction Vulnerabilities

* Length Extension Attack: Allows calculating hash of extended message without original message, compromising integrity.
* Collision Resistance: Potential for collisions as computational power increases, undermining security.
* Padding Vulnerabilities: Exploits in padding schemes like Padding Oracle Attack.
* Initialization Vector (IV) Vulnerabilities: Poorly chosen or predictable IV enables attacks.
* Compression Function Security: Weaknesses in underlying compression function undermine hash security.
* Cryptanalytic Attacks: New attacks discovered over time as techniques evolve.
* Implementation Issues: Poor implementation or non-compliance with standards introduces risks.