project4part2

April 7, 2025

1 Part II: Essay Questions

1.1 Mathematical inevitability

> Explain why hash collisions are a mathematical inevitability.

Hash collisions are a mathematical inevitability due to the pigeonhole principle, which states that if you map *items* into *containers*, where the number of items exceeds the number of containers, at least one container must hold more than one item. Here, the *items* are inputs into the hash function, and the *containers* are the hashes outputted by the hash functions. Therefore, there must be two inputs that result in the same hashes, given a sufficiently large input set and a sufficiently small output set. This is then exaggerated by the birthday paradox that implies that the probability of collisions increases exponentially and dictates the probability of collisions given the input space.

This can be generalized with m containers and n items as

$$p(n) = 1 - \frac{(m)_n}{m^n}$$

where $(m)_n$ is the falling factorial

$$\prod_{k=0}^{n-1} m - k$$

which can be reduced to

$$p(n) = 1 - \prod_{k=0}^{n-1} \frac{m-k}{m}$$

1.2 Shared birthday with Trudy

> 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%?

Each person's birthday is assumed to be equally likely on any of 365 days. There are $n-1$ other people besides Trudy. Each of those people has a $\frac{1}{365}$ chance of having Trudy's birth, or, $\frac{364}{365}$ chance of not having her birthday.

Generally, this is

$$q(n;d) = 1 - \left(\frac{d-1}{d}\right)^n$$

which specifically will be

$$q(n;365) = 1 - \left(\frac{364}{365}\right)^{n-1} \geq 0.5$$

We find that $n = 254$ satisfies the problem.

$$q(254;365) = 1 - \left(\frac{364}{365}\right)^{254-1} \approx 0.50047 \quad (50.1\%)$$

1.3 Any two shared birthdays

> In a room of n people ($n \leq 365$), what's the probability of any two sharing a birthday, and what's the minimum n for this probability to be over 50%?

The probability of any two sharing a birthday, $p(n) = 1 - \bar{p}(n)$

$$p(n) = 1 - \prod_{k=0}^{n-1} \frac{365-k}{365}$$

For

$$p(n) = 1 - \prod_{k=0}^{n-1} \frac{365-k}{365} \geq 0.5$$

we find $n = 23$.

$$p(23) = 1 - \prod_{k=0}^{23-1} \frac{365-k}{365} \approx 0.5072 \quad (50.7\%)$$

1.4 Birthday attack efficiency

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

The birthday attack leverages the probability of finding two inputs that hash to the same value (a *collision*). In a brute-force search, the probability of finding a collision reaches 50% with 2^n attempts. The birthday paradox implies that the probability of finding *any* collision (albiet not a target collision) grows exponentially faster than that, taking only $2^{n/2}$ attempts to reach a 50% probability (which is the classical preimage resistance). This implies a greater attack efficiency compared to brute-forcing by taking less attempts to find hash collisions. The resulting collision can be exploited in replay attacks or break trust in digital signatures. This attack highlights the importance of choosing strong, collision-resistant hash functions with sufficiently large output sizes to make such attacks infeasible. For example, SHA-256 would take up to 2^{256} attempts to find with brute forcing, while the birthday paradox suggests that 2^{128} attempts would be enough.

1.5 Merkle–Damgård construction issues [1]

Discuss the main issues associated with hash functions created using the Merkle–Damgård construction process.

1.5.1 Length extension attack

A major weakness of Merkle–Damgård-based hashes is their susceptibility to length extension attacks. If an attacker knows the hash $h(m)$ of a message m, they can compute the hash of m appended with additional data m', $h(m\|m')$, without knowing the original message. Because the Merkle–Damgård construction is done in a sequential, iterative manner, the internal state after processing m is the same as the initial state for processing m', allowing attackers to extend the message and generate a valid hash.

1.5.2 Herding attack [2]

Due to how the chaining process works, once a single collision is found, it can be extended to generate multiple collisions without additional effort, called herding attacks. They combine a collision-finding attack against to build a diamond structure, which then follows with searches for a string s such that $m\|s$ collides with one of the diamond structure's intermediate states. Having found such a string s, we can construct a sequence of message blocks q from the diamond structure, and build a suffix $s' = s\|q$ such that $h(m\|s') = h$, requiring a negligible amount of additional work.

1.5.3 Long second preimage attacks [3]

A second preimage attack aims to find a different message m' that produces the same hash as a given message m, $h(m) = h(m')$. With a sufficiently long message, and thus many intermediary values, an attacker may find an inermediate value with a collision that results in the same final hash. In general, a brute-force second preimage attack requires about 2^n operations for an output size n, however, for long messages, Merkle–Damgård allows an attack that reduces this complexity to around $2^{n/2}$.

1.5.4 References

1. Tiwari, H. (2017). Merkle–Damgård Construction Method and Alternatives: A Review. *Journal of Information and Organizational Sciences*, 41, 283-304.

2. Kelsey, J., & Kohno, T. (2006). Herding Hash Functions and the Nostradamus Attack. *In Advances in Cryptology - EUROCRYPT 2006* (pp. 183–200). Springer Berlin Heidelberg.

3. John Kelsey, & Bruce Schneier. (2004). Second Preimages on n-bit Hash Functions for Much Less than 2n Work.