Project #2: Hash Attack report

Background

In a hash attack, an adversary tries to take advantage of flaws or vulnerabilities in cryptographic hash functions, which are algorithms that produce fixed-size, one-of-a-kind hash values from input data. Hash attacks can take many different forms, such as preimage attacks (identifying an input that creates a certain hash), collision attacks (finding two separate inputs with the same hash), and more. Data integrity, authentication, or security systems that depend on the dependability and uniqueness of hash values are the main targets of hash attacks. Such attacks are a major problem in cybersecurity because they can have substantial repercussions, especially when used to modify digital signatures, passwords, or authentication systems.

The National Institute of Standards and Technology (NIST) first released the Secure Hash Algorithm-1 (SHA-1) in 1993. SHA-1 was created by the National Security Agency (NSA) of the United States. Initially, it was intended to offer data authenticity and integrity in a number of applications, such as digital signatures and certificate authorities.SHA-1 generates a fixed-size 160-bit hash value, which is commonly shown as a 40-character hexadecimal integer, from an input (message) of any length. Preimage resistance (given a hash, it should be computationally impossible to find the original input), second preimage resistance (given an input, it should be computationally impossible to find a different input with the same hash), and collision resistance (it should be computationally impossible to find two different inputs with the same hash) are the key characteristics of a secure cryptographic hash function.SHA-1 is now insecure for many cryptographic tasks due to numerous weaknesses and collision attacks that have been found against it over the years. As a result, more secure hash algorithms like SHA-256 and SHA-3 are frequently employed in place of SHA-1 for applications that require security. In fact, because of these flaws, a lot of organizations and standards bodies have discouraged or forbidden the use of SHA-1 in favor of more robust substitutes.

According to the BYU CS465 project 2 project background. A collision attack in hashing is when an attacker finds two separate source messages that both hash to the same value. The expected time for this attack is 2^(n/2) where n is the number of bits in the hash digest. A pre-image attack in hashing is when an attacker is given a specific hash value (usually the hash value of an intercepted message) and is asked to find a source message that hashes to that value. The expected time for this attack is 2^n where n is the number of bits in the digest. For this project, we will try to use brute-force hash attacks to see if the complexity of pre-image and collision attacks against SHA-1 matches the expected values.

Experiment abstract

The purpose of this lab is to evaluate how challenging collision and preimage attacks on a truncated SHA-1 hash with different bit sizes are. The actions listed below can help you accomplish this:

1. Wrapper for SHA-1: Create a custom wrapper for the SHA-1 hash function that takes two inputs: the string to be hashed and the desired number of bits (n) for the hash output. The wrapper will produce the SHA-1 hash of the input string, which is then truncated to the specified number of bits (n).
2. Conducting Attacks: Perform a series of collision and preimage attacks at different bit sizes. Choose at least four-bit sizes within the range of 8 to 32. It's suggested to include values like 8, 10, 16, 20, and 24. For bit sizes between 8-15, limit the number of trials to a maximum of two to ensure efficiency. Avoid attempting attacks that would require an impractical amount of time.
3. Data Gathering: Keep track of the number of attempts (hashes) for each attack to succeed. This data should be collected for each of the selected bit sizes, with a minimum of 50 samples for each bit size.

Ultimately, the goal of this lab is to offer empirical evidence on the practical difficulty of collision and preimage attacks on truncated SHA-1 hashes with different bit sizes. The findings will aid in assessing the security ramifications of employing truncated hashes for various applications and serve as a reference for choosing the right hash algorithm and bit size.

Experiment Process

Collision Attack

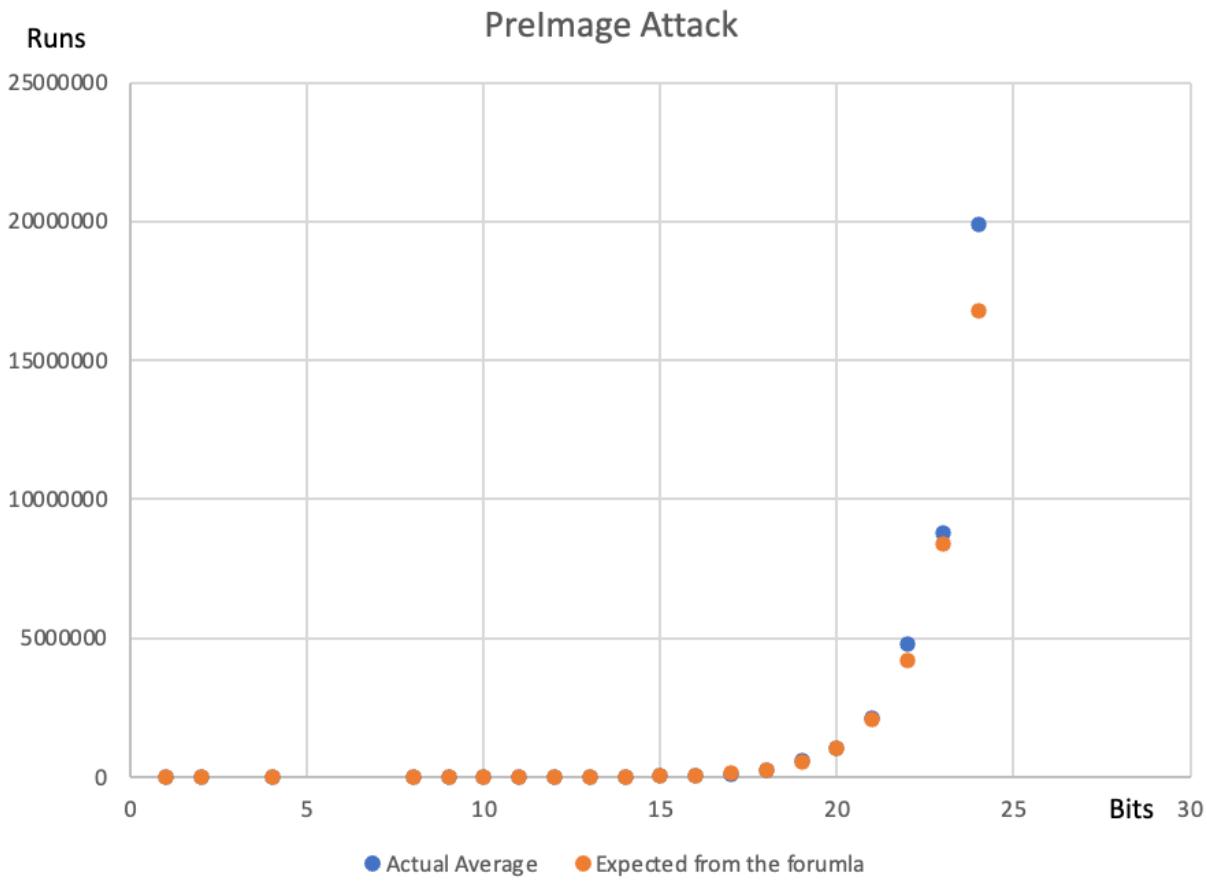
A collision attack is launched and executed 100 times using the collisionAttack function.
It keeps a dictionary hash to retain the truncated hash values and corresponding input strings inside the loop. It uses the generateRandomString function to produce a random string. It uses the truncateDigest function to truncate the text after computing its SHA-1 hash. It determines whether the input string differs from the one already saved and determines whether the truncated hash is already present in the hashes dictionary if it is. The loop ends if a collision is discovered. Each time the inner loop iterates, the number of tries is increased. The overall count of attempts and the average number of efforts to find a collision is calculated and printed after 100 iterations.

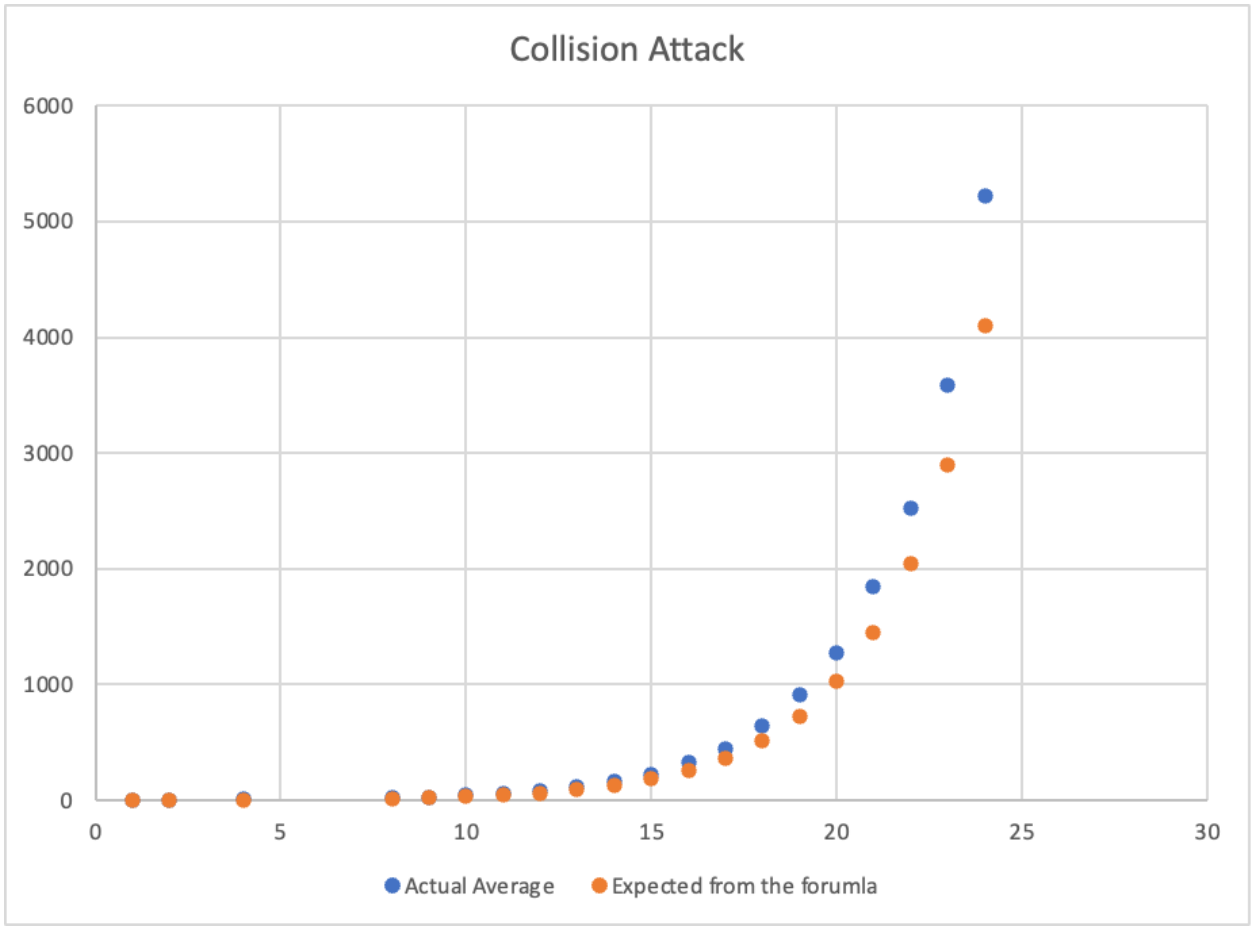
Preimage attack

Preimage attacks are launched and repeated 100 times using the preImageAttack function. It first computes the SHA-1 hash of the original input string self.string and then uses the truncateDigest method to truncate this hash to a specified amount of bits.
It starts a counter inside the loop and looks for input strings that provide the same truncated hash value as the target hash found in step 2. Utilizing the generateRandomString method, it creates random strings and computes the SHA-1 hash of each string, trimming the hash to the required bit length. It contrasts the target truncated hash acquired in step 2 with the truncated hash of the created string (newDigest). The loop ends if a match is discovered. Each time the inner loop

iterates, the number of tries is increased. The total count of attempts and the average number of tries to discover a preimage match for the target truncated hash are calculated and printed after 100 iterations.

Result





Due to time, I was able to run the collision attach for (1,2,4,8,9,10-24) bits for about 400 times total. Each actual average column will represent the average 100 times. And since the pre-image attack is 2^n I was only able to run (1,2,4,8,9,10-20) for about 400 times and (21-24) for 100 times since each will double the time from the previous bit.

The outcomes of this experiment confirm my expectations. As I mentioned previously, the collision attack only has a complexity of 2^(n/2) compared to the pre-image attack's complexity of 2^n. This is seen in the graph, as both curves almost closely match the expected complexities.

Conclusion

This experiment's data demonstrates that the initial complexity estimates of 2^n and 2^(n/2) for pre-image and collision assaults, respectively, were accurate.

Reviewed by Yirui Sun(My wife who is a data scientist working at Credit One Bank)