FSOC Semester Project WS21/22

Design and Implementation of a SHA-1 Accelerator

Report submitted by:

PREM JASON MENDONCA
Matrikelnummer : 1112318

March/18/2022

Content

1 Theoretical Part

1. *What is a one-way function?*

   In computer science, a one-way function is a function that is easy to compute on every input, but hard to invert given the image of a random input.
   In mathematical terms, for any variable 'x', there exists a unique value 'X' computed through a function 'f(x)' such that, it is very difficult to find a reverse function to find 'x'.

2. *What are the typical applications of one-way functions?*

   Cryptographic applications like Private-key encryption schemes.
   Authentication of Messages.
   Authentication of Digital signatures.
   Pseudorandom generators.
   Cryptocurrency verification, authentication and mining.

3. *Define pre-image resistance and the secondary preimage resistance characteristic of a one-way function.*

   Pre-image resistance characteristic makes it hard and time-consuming for an attacker to find an original message 'm', given its respective hash value, H_m. It wards off brute force attacks from powerful machines.

   Second pre-image resistance characteristic is granted by SHA such that, when a message is known as m_1, yet it is hard to find another message, m_2 that hashes to the same value as $H_{m1} = H_{m2}$.
   Without this characteristic, two different passwords would yield the same hash value, deeming the original password unnecessary in order to access secured data.

4. *What is a collision and how does it affect the security of a hash function?*

   Since hash functions have infinite input length and a predefined output length, there is inevitably going to be the possibility of two different inputs that produce the same output hash. However, if two separate inputs do indeed produce the same hash output, it is called a **collision**.
   This collision can then be exploited by any application that compares two hashes together – such as password hashes, file integrity checks, etc.

5. *Does the Boolean XOR function represent a valid way to verify the integrity of a message. Justify your answer!*

   The XOR operator is simple to implement, computationally inexpensive and extremely common as a component in more complex ciphers. However, the problem with XOR encryption is that for long runs of the same characters, it is very easy to see the password. When the frequently occurring characters are XORed with the password shorter in length that the text, the output will have repeating sequences of characters. The attacker would just look for any such repeating characters, try to guess the character in the original file (normally, 'space' would be the first candidate to try), and derive the length of the password from length of repeating groups.

But, XOR encryption can be reasonably strong to verify the integrity of a message if the following conditions are met:

a. The plain text and the password are about the same length.
b. The password is not reused for encrypting more than one message.
c. The password cannot be guessed. This means the bits are randomized.

6. *Why do collisions necessarily exist?*

   By pigeonhole principle, on which the hash functions are based, there exist collisions invariably. A hash function with large sized input (say 2^{64}-bits) producing limited sized output (say 256-bits) will necessarily has a very large number of collisions. However, the probability of running across one by accident is negligible and for a secure hash function, the cost of constructing one deliberately is unrealistically large. Hence the disregard for the possibility of a collision until a hash function starts to make the transition from 'secure' to 'broken'.

7. *Explain the birthday problem and the state the relation to the hash collisions*.

   In probability theory, the ‘birthday paradox’ or ‘birthday problem’ considers the probability that some paired people in a set of ‘n’ randomly chosen of them, will have the same birthday. Therefore, the ‘birthday paradox’ challenges the collision resistance; if a hash function produces N bits of output, an attacker who computes only $2^{N/2}$ hash operations on random input is likely to find two matching outputs.
   However, the birthday attack is used to create hash collisions to test the flaw in the hash functions. Just as matching the birthdays is difficult, finding a specific input with a hash that collides with another input is difficult. Similarly, just like matching any birthday is easier, finding any input that creates a colliding hash with any other input is easier due to the birthday attack. Henceforth, if the attacker uses a brute-force attack such that, it is broken within $2^{N/2}$ hash operations, it is typically considered a flaw in the hash function.

8. *What role do mathematical one-way functions play in the context of cryptocurrency Bitcoin?*

   In cryptocurrency Bitcoin, one-way functions are extensively used as either hash functions and/or digital signatures for the following important purposes:
   a. Authorizing transactions with digital signatures
   A Bitcoin transaction is a message with instructions, and by signing it with a digital signature, it is simultaneously proved that the private key is truly owned by the user and ensures that the transaction details intended. If the signature is missing or doesn't match the public key, nodes on the Bitcoin network will consider the transaction invalid and will not add it to the blockchain.
   b. Verifying the validity of the transaction history
   Each block in the blockchain contains a list of transactions and a hash of the transactions in the previous block in the chain. Hence, it's impossible to alter just one transaction in a block in the middle without causing a mismatch between the expected block hash and the hash recorded in the next block. This verification provides an efficient way for a single node to check whether transactions in its copy of the blockchain have been tampered with.

Rather than checking every transaction in the entire transaction history, a node can simply check whether the hash of each block matches the recorded hash of subsequent blocks.

c. Proof-of-work in Bitcoin mining
The one-way function used in Bitcoin mining is a cryptographic hash function. Miners are given the output criteria (the output can be any number less than some threshold), but by design, cryptographic hash functions make it difficult to reverse calculate what the corresponding input needs to be. Therefore, Bitcoin miners can only randomly choose different inputs, hoping that the output will be a number below the threshold. Once a miner determines the correct input value, it is easy to prove his or her work to others by sharing that value with others who can easily recalculate the hash.

d. Extra protection for Bitcoin private keys
Authorizing transactions with digital signatures requires sharing the public key with others. Although it is thought to be computationally infeasible to calculate a private key from a public key, it is similarly difficult to calculate the input of a hash function given just the output, and it is doubly difficult to do both. Instead of sharing a public key, users share a Bitcoin address, which is a cryptographic hash of a public key. In fact, the public key is hashed twice using two different cryptographic hash functions to create a Bitcoin address. This extra protection ensures that no amount of analysis of a Bitcoin address can reveal the underlying private key.

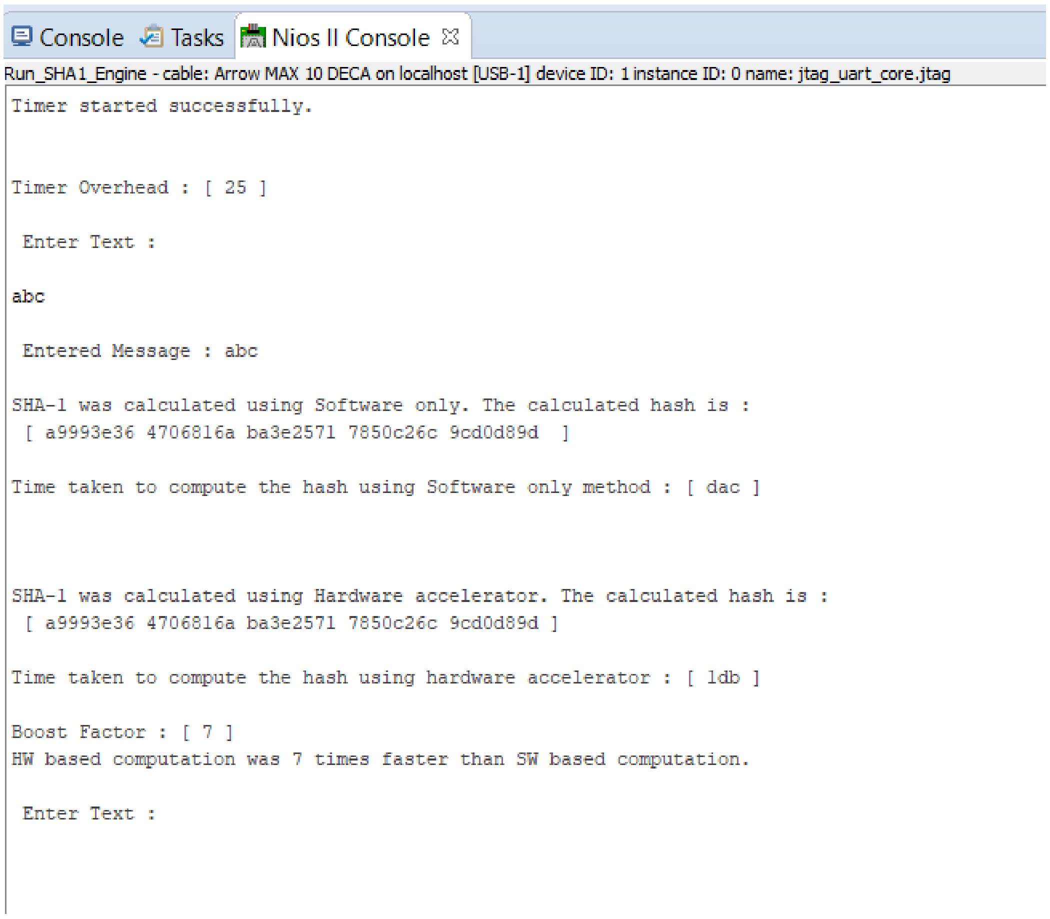
2 Boost Factor Calculation

An input message ‚abc‘ was given.

Number of cycles taken to calculate SHA-1 hash using software only was 3500 cycles.

Number of cycles taken to calculate SHA-1 hash using custom hardware acceleration was 475 cycles.

Boost factor is 7, implying that the SHA-1 compuation using hardware accelerator was 7 times faster than computation using software only.



Run_SHA1_Engine - cable: Arrow MAX 10 DECA on localhost [USB-1] device ID: 1 instance ID: 0 name: jtag_uart_core.jtag

```
Timer started successfully.


Timer Overhead : [ 25 ]

 Enter Text :

abc

 Entered Message : abc

SHA-1 was calculated using Software only. The calculated hash is :
 [ a9993e36 4706816a ba3e2571 7850c26c 9cd0d89d  ]

Time taken to compute the hash using Software only method : [ dac ]



SHA-1 was calculated using Hardware accelerator. The calculated hash is :
 [ a9993e36 4706816a ba3e2571 7850c26c 9cd0d89d ]

Time taken to compute the hash using hardware accelerator : [ 1db ]

Boost Factor : [ 7 ]
HW based computation was 7 times faster than SW based computation.

 Enter Text :
```