PHOTON LIGHTWEIGHT HASH FUNCTION

FINAL REPORT

CS532: Cryptography and Data Security

Dr. Sherif Hashem

Joseph Killian, Brandon Horton

INTRODUCTION

- Lightweight cryptography is developed with the implementation into devices with small memory devices in mind, such as RFID tags, sensor networks, embedded systems, or any device with microcontrollers.
- Hash functions are a common cryptography algorithm, used to encrypt numerical data, taking an input of arbitrary size, and producing a message with fixed size as an output, known as the hash value.
- There are a number of lightweight hash algorithms, such as Quark, SPONGENT, Lesmanta-LW, and PHOTON. We have chosen PHOTON to implement.

LITERATURE REVIEW

1. Paper one focuses on the PHOTON hash algorithm and how it is implemented and constructed, and how the function performs security-wise and performance-wise.
2. Paper two is an overview of hash functions, lightweight hash functions, and how they work and are implemented.
3. This paper gives an overview of lightweight cryptography, their requirements, challenges imposed by these functions, and well-known examples of different lightweight functions
4. Paper four introduces sponge functions and how they work, and how they fit into hash functions.
5. The fifth reference used is a website dedicated to PHOTON and it’s various implementations, how they work, and code examples in C for each.
6. The sixth paper is another reference on how sponge functions work and are constructed.

LIGHTWEIGHT HASH FUNCTIONS

- Hash functions are a well-known technique in programming and cryptography. It works by taking a numerical input and converting it into a compressed value, known as a hash value.
- There are two main parts of any hash function: Construction, and compression.
  - Construction exists to iterate the compression function
  - The compression function compresses many inputs into much fewer outputs, effectively compressing the data down
- The lightweight version of this is designed to take in inputs of much smaller sizes and also itself be computationally effective, but not overly so, as the systems it is implemented on usually do not have much computing power.

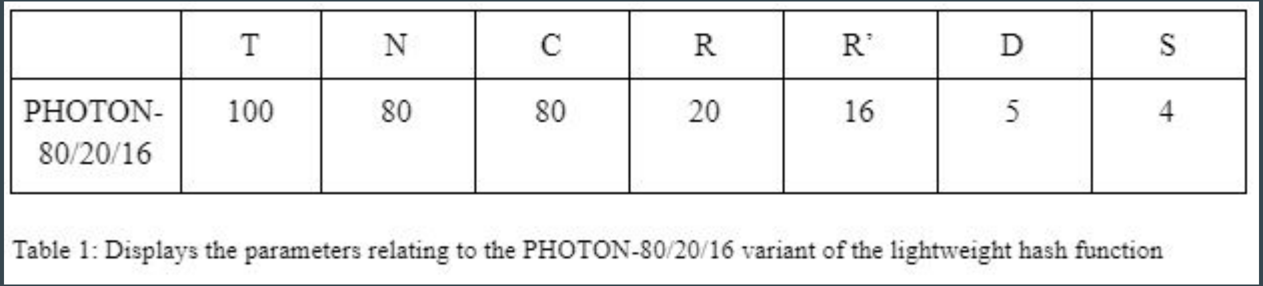
APPLICATIONS/SECURITY

There are three main applications of lightweight hash functions: Digital signatures, MAC, and Authenticated Encryption

1. The hash function encrypts a message with a private key, which results in a digital signature
2. Message Authentication Codes (MAC) take in the message and secret key, and generates a tag for the message, to verify the integrity of the message
3. Authenticated Encryption means that the plaintext is encrypted separately, and fed through the hash with a key to generate a MAC, which is appended to the cipher text

The variant of PHOTON we implemented is the smallest version, so it works best when there are lower security requirements, such as an RFID tag.

PARAMETERS FOR PHOTON



	T	N	C	R	R’	D	S
PHOTON-80/20/16	100	80	80	20	16	5	4

Table 1: Displays the parameters relating to the PHOTON-80/20/16 variant of the lightweight hash function

T = Bitsize of the internal state matrix || N = Bitsize of the output

C = Capacity part of the internal state matrix

R = Bitsize of the bitrate part of the internal state matrix || R’ = Bitrate of the output

D = Number of rows and columns in the internal state matrix || S = Bitsize of each cell in the internal state matrix

T = C + R

CONSTRUCTION

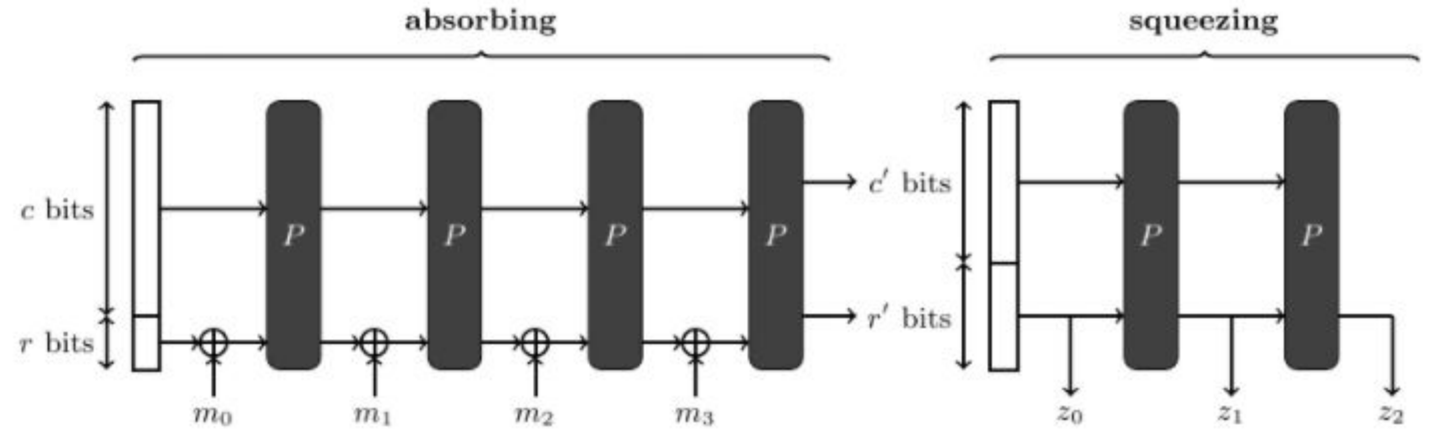


Figure 1: Illustrates the domain extension algorithm based off of the classical sponge framework, and where the permutations occur at each stage of the algorithm. Image taken from [5]

INTERNAL PERMUTATION

- AddConstants: This function applies round specific constants to the first column of the state matrix, each element of the first column of the internal state is XORed with the constants corresponding to the round of the permutation, and returns the internal state
- SubCells: This function applies a 4 bit S-Box to each cell of the internal state, this implementation uses the PRESENT Sbox for 4 bits as provided in the appendix by the authors of reference [1], as well as in their reference implementation in [5], and returns the internal state
- ShiftRows: This function performs a left circular shift of row each i, by i positions, and returns the internal state
- MixColumnsSerial: For this function, each column is multiplied by the aMatrix to produce the mixed column, So the formula for the new column is column[i] = aMatrix * column[i]. The values for the matrix were taken from the appendix of reference [1]

DOMAIN EXTENSION ALGORITHM

- Padding: The message is read in as a binary string, then a ‘1’ bit is appended to the end of the message, as well as, as many zeros as necessary so that the padded message is a multiple of the bitrate(20)
- Chunking: The message is broken up into chunks of 20 bits each
- Absorbing: A message chunk is XORed with the first row of the state matrix, and then the permutation function is applied
- Squeezing: 16 bits are output, which come from the first four cells of the first row of the state matrix, then the permutation function is applied

PHOTON-80/20/16

- Internal state is initialized, absorbing bits are set as the first row of the matrix, and squeezing bits are the first four cells of the first row of the matrix
- Message is padded, then broken up into chunks of 20 bits
- Each message chunk is absorbed, and the permutation function is performed in between each
- Once the message has been fully absorbed, the first 16 bits of output are squeezed out, then another permutation is performed
- This is repeated until the output size of 80 bits is reached

CONCLUSION

- We successfully implemented the PHOTON-80/20/16 variant of the lightweight hash function in Python
- Successfully constructed the function of the domain extension algorithm, which mirrors a classical sponge function
- This version of PHOTON is applicable to lower levels of security, such as small devices like RFID tags.

REFERENCES

[1] J. Guo, T. Peyrin, and A. Poschmann, “The photon family of lightweight hash functions,” in Annual Cryptology Conference. Springer, 2011, pp. 222–239.

[2] Baraa Tareq Hammad, Norziana Jamil, Mohd Ezanee Rusli and Muhammad Reza Z`aba, A survey of Lightweight Cryptographic Hash Function, International Journal of Scientific & Engineering Research Volume 8, Issue 7, July-2017

[3] McKay, K. , Bassham, L. , Sonmez, M. and Mouha, N. (2017), Report on Lightweight Cryptography, NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, Gaithersburg, MD, [online], https://doi.org/10.6028/NIST.IR.8114 (Accessed March 19, 2021)

[4] Bertoni, G.M. & Daemen, Joan & Peeters, Michael & Assche, Gilles. (2007). Sponge Functions. ECRYPT Hash Workshop 2007.

[5] Guo, J., Peyrin, T., and Poschmann, A., The PHOTON Family of Lightweight Hash Functions. http://sites.google.com/site/photonhashfunction/

[6] Guido Bertoni, Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche and Ronny Van Keer, Cryptographic sponge functions. https://keccak.team/sponge_duplex.html