Dynamic-Sized Secure Hash Algorithm with Memory-Hard Capabilities for Enhanced Cryptographic Security

Under the guidance of

Dr. Ripon Patgiri

Submitted by :

Tolemy Kashyap, 2112130

Jugya Kamal Gogoi, 2112098

Kumar Ashish Ranjan, 2112144

Arnav Raj, 2112116

TABLE OF CONTENTS

INTRODUCTION

Secure hash algorithms (SHAs) are essential tools in cryptography, widely used across various applications. **Keccak**, selected as the SHA-3 standard by *NIST*, introduced the sponge construction with absorb and squeeze functions, advancing secure hashing techniques. However, while Keccak attempts to randomize bit outputs, it lacks formal proof of true randomness and is not designed for variable-sized outputs or random number generation.

Conventional hash functions face challenges with fixed-sized outputs, which are predictable and vulnerable to attacks like rainbow tables and brute-force methods. Additionally, the absence of mechanisms to counter parallelism increases their susceptibility to adversarial attacks. To overcome these limitations, we propose a memory-hard, randomized algorithm that generates dynamic, variable-sized hash values.

LITERATURE SURVEY

Paper Name	Publication Details	Methodology	Takeaways
Stronger Key Derivation via Sequential Memory-Hard Function	Colin Percival, Published in 2009, available on Tarsnap	- Introduces sequential memory-hard functions. - Presents the scrypt key derivation function, designed to resist brute-force attacks by requiring significant memory resources. - Analyzes computational costs for brute-force attacks and compares scrypt to other key derivation functions (PBKDF2, bcrypt).	- Scrypt offers strong resistance against brute-force attacks by leveraging memory-hardness. - It provides future adaptability by allowing tunable parameters based on memory, CPU cost, and parallelism. - Memory usage limits hardware attack advantages, making it more secure than bcrypt and PBKDF2.

Paper Name	Publication Details	Methodology	Takeaways
Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications	2016 IEEE European Symposium on Security and Privacy	- Introduces Argon2, a memory-hard hash function designed to resist attacks using specialized hardware like GPUs and ASICs. - Two variants: Argon2d (data-dependent) for cryptocurrencies and Argon2i (data-independent) for password hashing and key derivation. - Uses a tunable amount of memory and computational resources to thwart trade-off attacks.	- Provides strong protection against both time-memory and computation-memory trade-off attacks. - Focuses on simplicity and clarity to enhance security and ease of analysis. Argon2i is slower but safer from side-channel attacks, while Argon2d is faster but vulnerable to side-channel attacks. - The function is highly scalable in both time and memory, making it suitable for a wide range of applications from backend servers to cryptocurrency mining

Paper Name	Publication Details	Methodology	Takeaways
Enhancing Password Security with Salt and Pepper	Niyaa Meganathan. Volume 9 - 2024, Issue 9 - September	- Analysis of cryptographic techniques: Explores how salt and pepper are implemented to enhance password hashing and protect against rainbow table attacks. - Evaluation of security enhancements: Examines the impact of salt, pepper, and key-stretching algorithms. - Future advancements discussion: Proposes future improvements, including post-quantum cryptography and passwordless authentication.	- Salt and pepper defense: Using salt ensures unique password hashes, while pepper adds a system-wide secret, making attacks more difficult. - Not foolproof: While effective, these methods need additional security layers, like key-stretching and multi-factor authentication. - Future directions: Post-quantum cryptography and passwordless authentication are potential future advancements to enhance password security further

Paper Name	Publication Details	Methodology	Takeaways
Balloon Hashing: A Memory-Hard Function Providing Provable Protection Against Sequential Attacks	Proceedings of the 22nd International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2016), Springer, Lecture Notes in Computer Science, Volume 10031, 2016	- Introduced the Balloon hashing algorithm, a memory-hard function designed for password hashing. - Utilizes random sandwich graphs to analyze memory-hardness, a novel approach in the domain. - Compares the performance of Balloon against other memory-hard functions like Argon2i and scrypt, showing strong results under different cryptographic primitives (e.g., Blake2b).	- Provides stronger theoretical guarantees than other practical memory-hard functions, ensuring resistance to sequential attacks. - Outperforms similar algorithms (Argon2i, Catena) in specific configurations, making it suitable for diverse environments. - Although it is robust against sequential attacks, Balloon may not be fully optimal against parallel attacks. Proposals are suggested to address this vulnerability.

Paper Name	Publication Details	Methodology	Takeaways
Towards Practical Attacks on Argon2i and Balloon Hashing	Joel Alwen, Jeremiah Blocki. 2017 IEEE European Symposium on Security and Privacy.	- The paper introduces novel heuristics to extend theoretical attacks on Argon2i-A and Balloon Hashing algorithms to Argon2i-B. - Simulations of the attacks on Argon2i-A, Argon2i-B, and Balloon Hashing for various memory parameters were conducted.	- Memory hardness is critical: Argon2i-B must use more than 10 memory passes to prevent attacks, exceeding the “paranoid” setting in the current IRTF proposal. - Security deficiencies in practical settings: The Alwen-Blocki attack can overcome the security measures of Argon2i-B and Balloon Hashing when applied under realistic hardware constraints, especially for limited memory parameters.

Paper Name	Publication Details	Methodology	Takeaways
Cryptographic Hash Functions: Recent Design Trends and Security Notions	Saif Al-Kuwari, James H. Davenport, Russell J. Bradford 2011, Department of Computer Science, University of Bath, UK	- Provides a detailed overview of the evolution of cryptographic hash functions in response to advances in cryptanalysis, such as attacks on MD5 and SHA-1. - Discusses key properties of hash functions, including collision resistance, preimage resistance, and second pre-image resistance, with practical implications. - Analyzes security notions like indifferentiability from Random Oracle (RO) and indistinguishability from Pseudorandom Function.	- Emphasizes the inherent challenges in balancing efficiency and security in hash function designs, as demonstrated by the limitations of Merkle-Damgård construction. - Highlights the importance of addressing emerging attack vectors, such as multi-collision and herding attacks, through advanced constructions like sponge and tree-based methods. - Stresses the role of competitions like SHA-3 in fostering innovation and setting new standards for cryptographic hash functions.

Paper Name	Publication Details	Methodology	Takeaways
On the Indifferentiability of the Sponge Construction	Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche EUROCRYPT 2008, Lecture Notes in Computer Science, Springer	- Explores the sponge construction, an iterative cryptographic hash function that uses a single transformation or permutation rather than a compression function. - Provides a formal proof that the sponge construction is indifferentiable from a random oracle when utilizing random transformations or permutations. - Introduces two simulators to analyze the construction: one for random transformations and another for random permutations, ensuring sponge-consistent outputs.	- Sponge construction's security is bounded by its capacity, with collision resistance achievable for capacities above the required output length. - Avoids the complexities of compression functions by using a random-looking permutation, simplifying design and analysis. - Supports variable-length outputs, making it suitable for hash functions, message authentication codes (MACs), and stream ciphers.

Paper Name	Publication Details	Methodology	Takeaways
Keccak. In: Advances in Cryptology - EUROCRAT 2013	Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche EUROCRYPT 2013, Lecture Notes in Computer Science, Springer	- Developed as the winner of the SHA-3 Cryptographic Hash Algorithm Competition organized by NIST in 2012. - Utilizes the sponge construction, a flexible cryptographic framework enabling variable-length outputs for hashing, masking, and key derivation. - Introduces the Keccak-f permutations with tunable widths ranging from 25 to 1600 bits, optimized for efficiency and security	- Supports diverse cryptographic applications, including hash functions, MACs, stream ciphers, and authenticated encryption. - Provides multiple security levels and performance trade-offs by varying the Keccak-f permutation width, accommodating lightweight to high-security use cases. - Established as the SHA-3 standard, Keccak introduced a paradigm shift in cryptographic hash functions, moving from block-cipher-based to permutation-based designs.

Paper Name	Publication Details	Methodology	Takeaways
The Making of KECCAK. Cryptologia	By Bertoni, G., Daemen, J., Peeters, M., Van Assche and Gilles Van Assche Published Jan 2014	- Utilizes the sponge construction, a flexible cryptographic framework enabling variable-length outputs for hashing, masking, and key derivation. - Introduces the Keccak-f permutations with tunable widths ranging from 25 to 1600 bits, optimized for efficiency and security. - Incorporates operations such as non-linear layers (χ), linear mixing layers (θ), dispersion steps (ρ, π), and round constants (ι) for robust cryptographic transformations.	- Proven resistance to collision, preimage, and side-channel attacks due to its innovative design and thorough public scrutiny during the SHA-3 competition. - Streamlined for hardware and software implementations with short critical paths, enabling high performance and energy efficiency. - Supports diverse cryptographic applications, including hash functions, MACs, stream ciphers, and authenticated encryption.

MOTIVATION

- **Challenges with Fixed-Sized Hash Values**: Fixed-size hash values are predictable, making them vulnerable to rainbow table and brute-force attacks.

- **Limitations of Current Algorithms**: Conventional algorithms often lack the capability to restrict high levels of parallelism, which adversaries exploit to speed up attacks.

- **Advantages of Variable-Sized and Memory-Hard Algorithms**: Memory-hard algorithms impose significant resource requirements on adversaries while being efficient for legitimate users. Variable-sized outputs further enhance security by increasing computational complexity for attackers.

- **Vision for the Proposed Solution**: A memory-hard, variable-sized secure hash function aims to address these challenges, ensuring resistance to advanced computational threats and diverse cryptographic applications.

PROBLEM STATEMENT

Conventional secure hash algorithms face significant limitations in addressing modern cryptographic challenges. Fixed-sized hash values are a critical drawback, as their predictability makes them susceptible to rainbow table attacks and brute-force methods. Adversaries can precompute and store vast amounts of hash values, exploiting the fixed size to significantly reduce attack time. Furthermore, these algorithms often allow high degrees of parallelism, enabling attackers to leverage advanced computational resources to generate hash values faster, undermining the security of the system. The need for a secure, memory-hard, and variable-sized hashing mechanism remains unresolved, thus highlighting the urgency for this solution.

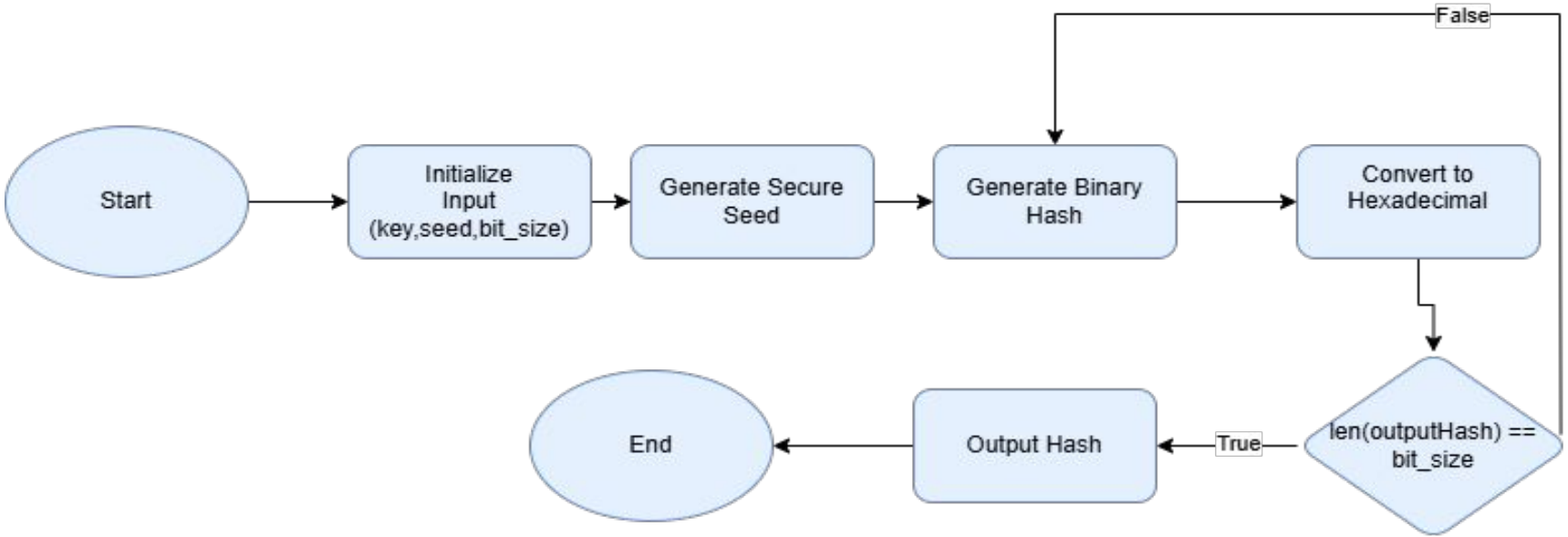
OBJECTIVE

The key objectives of our project are outlined below-

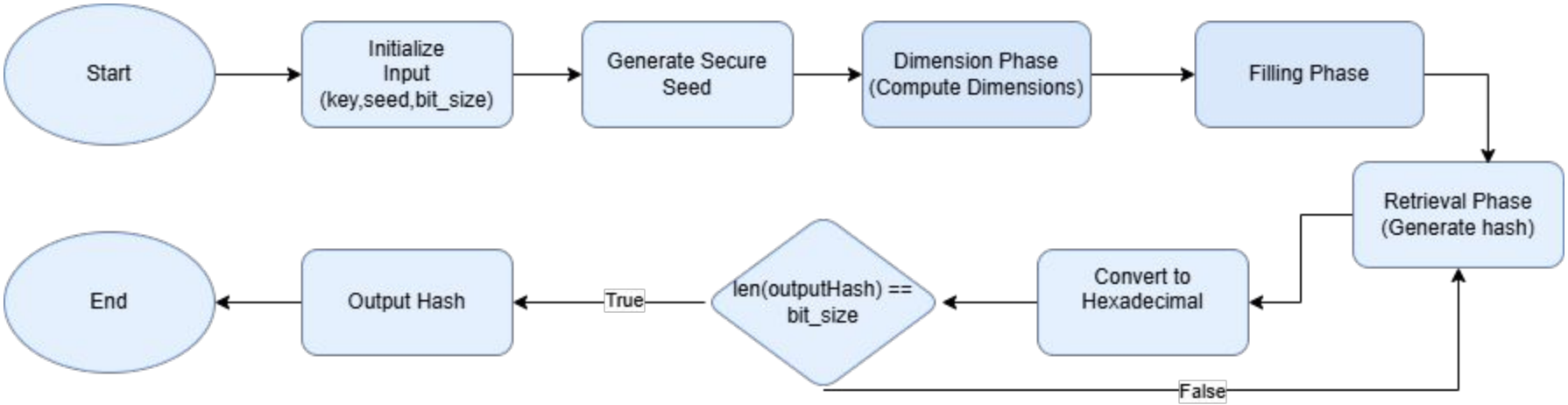
- To develop a secure hash algorithm that produces a randomized secure hash value based on an input string and a seed value.
- To demonstrate how keeping the size variable of the output hash value makes a brute-force attack nearly impossible.
- To develop a memory-hard secure hash algorithm.

APPROACH

- **DSSHA-1D**



- **DSSHA-2D/3D**


Start
Initialize Input (key,seed,bit_size)
Generate Secure Seed
Dimension Phase (Compute Dimensions)
Filling Phase
Retrieval Phase (Generate hash)
Convert to Hexadecimal
len(outputHash) == bit_size
True
False
Output Hash
End

Step 1: Input Handling (1D, 2D, 3D)

- Accept key, seed, and bit_size from the user.
- Use default values if the user does not provide inputs.

Step 2: Seed Initialization (1D, 2D, 3D)

- Compute the initial seed using the murmur hash function.
- Adjust the seed value with predefined constants or tables.

Step 3: Dimension Phase (2D and 3D)

- Calculate dimensions (X and Y for 2D, and X, Y, Z for 3D) using prime numbers and the seed.
- Ensure dimensions are pseudo-random and distinct.

Step 4: Filling Phase (2D and 3D Only)

- Allocate memory for a 2D or 3D vector (matrix or tensor).
- Initialize the vector with pseudo-random bits using the seed.

Step 5: Retrieval Phase (1D, 2D, 3D)

- For 1D, generate a sequence of pseudo-random bits directly using the seed.
- For 2D/3D, select bits pseudo-randomly from the vector based on the updated seed.

Step 6: Hexadecimal Conversion/ Output Generation (1D, 2D, 3D)

- Convert the generated bit sequence into hexadecimal format.
- If the given output length in not divisible by 4, then add one extra hexadecimal position.

OUTPUTS

- DSSHA-1D

```
######
DSSHA: Dynamic-Sized Secure Hash Algorithm
######
Enter a key (or press Enter to use default "DSSHA"): Group 27
Enter a seed (or press Enter to use default 198899): 198893
Enter a bit size (or press Enter to use default 512): 267

Choose the hashing algorithm:
1. DSSHA-1D
2. DSSHA-2D
3. DSSHA-3D
Enter your choice (1, 2, or 3): 1

Bit size is not multiple of 4!
The last hexadecimal code, 6, is not 4 bits!
It is 3 bits!

The generated hash value is:
c5705928afe28e05a28bf223e3ff8b22f64cc519644341390bc4f58865b3019da26

Time taken by the algorithm: 22.000 ms

Successfully completed...
```

- DSSHA-2D

```
######
DSSHA: Dynamic-Sized Secure Hash Algorithm
######
Enter a key (or press Enter to use default "DSSHA"): Group 27
Enter a seed (or press Enter to use default 198899): 198893
Enter a bit size (or press Enter to use default 512): 342

Choose the hashing algorithm:
1. DSSHA-1D
2. DSSHA-2D
3. DSSHA-3D
Enter your choice (1, 2, or 3): 2
Vector dimensions are X:457  Y:277

Bit size is not multiple of 4!
The last hexadecimal code, 0, is not 4 bits!
It is 2 bits!

The generated hash value is:
7c6711c18cb94b11f4c34f91d751c74c24367c05f58ea079ceb8cf0c8a053f31e4cafe557c8cfb0bfba750

Time taken by the algorithm: 36.000 ms

Successfully completed...
```

- DSSHA-3D

```
######
DSSHA: Dynamic-Sized Secure Hash Algorithm
######
Enter a key (or press Enter to use default "DSSHA"): Group 27
Enter a seed (or press Enter to use default 198899): 198893
Enter a bit size (or press Enter to use default 512): 256

Choose the hashing algorithm:
1. DSSHA-1D
2. DSSHA-2D
3. DSSHA-3D
Enter your choice (1, 2, or 3): 3
Vector dimensions are X:97  Y:127  Z:167

The generated hash value is:
5403cb608c84db5fcde9116217e1bc49a1b34cec52db029971099bf0eb6d7c3d

Time taken by the algorithm: 1549.000 ms

Successfully completed...
```

CONCLUSION

- This paper introduces a novel secure hash algorithm capable of generating variable-sized, randomized, and one-way hash values.
- Three variants—1D, 2D, and 3D—are presented to cater to diverse application requirements with varying balances of performance and security.
- The algorithm integrates memory-hardness to counter adversarial parallel processing, enhancing its resistance to attacks.
- The variability in hash values demonstrates its ability to thwart advanced attacks, such as rainbow table attacks, by increasing unpredictability.
- Overall, the proposed approach significantly enhances cryptographic security through adaptive features and robust randomization.

REFERENCE

- Balloon Hashing: A Memory-Hard Function Providing Provable Protection Against Sequential Attacks

  By Dan Boneh, Henry Corrigan-Gibbs , and Stuart Schechter (Revised 2017)

  https://link.springer.com/chapter/10.1007/978-3-662-53887-6_8

- Towards Practical Attacks on Argon2i and Balloon Hashing

  By Jeremiah Blocki and Joel Alwen (Published 2017)

  https://ieeexplore.ieee.org/document/7961977

- Stronger key derivation via sequential memory hard function

  By Colin Percival (Published 2009)

  bsdcan.org/2009/schedule/attachments/87_scrypt.pdf

- Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications

  By Alex Biryukov, Daniel Dinu, Dmitry Khovratovich (Published 2016)

  https://ieeexplore.ieee.org/document/7467361/

- Cryptographic hash functions: Recent design trends and security notions. Cryptology ePrint Archive
  By l-Kuwari, S., Davenport, J.H., Bradford, R.J. (Published 2011)
  https://eprint.iacr.org/2011/565

- On the Indifferentiability of the Sponge Construction. In: Advances in Cryptology - EUROCRAT 2008
  By Bertoni, G., Daemen, J., Peeters, M., Van Assche and Gilles Van Assche (Published 2008)
  https://link.springer.com/chapter/10.1007/978-3-540-78967-3_11

- Keccak. In: Advances in Cryptology - EUROCRAT 2013
  By Bertoni, G., Daemen, J., Peeters, M., Van Assche and Gilles Van Assche (Published 2013)
  https://link.springer.com/chapter/10.1007/978-3-642-38348-9_19

- The Making of KECCAK. Cryptologia
  By Bertoni, G., Daemen, J., Peeters, M., Van Assche and Gilles Van Assche (Published Jan 2014)
  https://www.tandfonline.com/doi/abs/10.1080/01611194.2013.856818

THANK YOU