SECURING THE WEB USING KECCAK (SHA-3)

V. Sukrutha[1] and Y. Madhavi Latha[2]
[1] Electronics & Computer Engineering, IV/IV, K. L. University, Vaddeswaram, Guntur, India
[2] Asst. Prof., Dept. of Electronics & Computer Engineering,
K. L. University, Vaddeswaram, Guntur, India

ABSTRACT

Information Security requirements have changed in recent times. Traditionally, provided by physical and administrative mechanisms. A mechanism that is designed to detect, prevent, or recover from a security attack. Cryptographic hash functions, which is widely used in a great number of protocols and security mechanisms. SHA-3, originally known as Keccak, is a Cryptographic hash function. SHA-3 is a compliment of SHA-1 and SHA-2.Keccak has higher performance in hardware implementations than SHA-2. It supports the same hash lengths as SHA-2, and its internal structure differs significantly from the rest of the SHA family.

KEYWORDS: *sponge construction, Keccak permutation, hashing ALGORITHMS.*

I. INTRODUCTION

Cryptography deals with the construction and analyzing protocols that helps in solving the difficulties that arise with the intrusion of third parties. It is also related to various concepts of information security like data confidentiality, data integrity, authentication, and non-repudiation. The cryptographic techniques of modern age are based on mathematical theory as well as computer science practice. The computational hardness assumptions on which the algorithms are designed make them hard to break by the intruders. The other important category of cryptography is the hash functions. The National Institute of Standards and Technology (NIST) has published a family of cryptographic hash functions called the secure hash algorithm which is recognized as a U.S. Federal Information Processing Standard (FIPS).It refers to :-

- **SHA-0**: The original version of 160 bit hash algorithm published in 1993 called SHA was withdrawn right after it was released because of an undisclosed problem. SHA -1 was released which was a modified version of SHA-0.
- **SHA-1**: It was designed by the National Security Agency (NSA) to be part of digital signature algorithm. It is a160 bit hash function which similar to MD5 algorithm. Nevertheless, its standard was no longer approved for most of the cryptographic uses after 2010 because of its weaknesses.
- **SHA-2**: It has two similar hash functions called SHA -256 and SHA-512.They have different block sizes and also different word sizes.SHA-256 uses 32-bit words whereas SHA-512 uses 64-bit words. There are also modified versions of both the above algorithms called SHA-224 and SHa-384 which were also designed by the NSA.
- **SHA-3**: It is also known as Keccak. It was chosen in 2012 from a competition among non-NSA designers. It uses same hash length as SHA-2 and the internal structure of Keccak differs from the SHA family.
- Most cryptographic applications that help in ensuring the authenticity of digital documents like digital signatures and message authentication codes use hash algorithms. These hash algorithms take an electronic file and generate a short "digest," a sort of digital fingerprint of the content. If a small modification is made to the original message, a change in the digest also follows. For any file and digest, it is impossible for a forger to create a different file with the same digest.

The Keccak algorithm was appreciated by the NIST for its amiable qualities like the elegant design and the ability to run well on many different computing devices. Among the many that participated in the competition Keccak has an easy way of analysis owing to its clarity in construction also compared to SHA-2 or other finalists Keccak has higher performance in hardware implementations.

II. KECCAK

- KECCAK is a family of sponge functions. The *sponge function* is a generalization of the concept of cryptographic hash function with infinite output and can perform quasi all symmetric cryptographic functions, from hashing to pseudo-random number generation to authenticated encryption.
- For a quick introduction, we propose a *pseudo-code* description of KECCAK. The reference specification, analysis, reference and optimized code and test vectors for KECCAK can be found in the file section.
- As primitive used in the sponge construction, the KECCAK instances call one of seven permutations named KECCAK-*f*[*b*], with *b*=25, 50, 100, 200, 400, 800 or 1600. In the scope of the SHA-3 contest, we proposed the largest permutation, namely KECCAK-*f*[1600], but smaller (or more "lightweight") permutations can be used in constrained environments. Each permutation consists of the iteration of a simple round function, similar to a block cipher without a key schedule. The choice of operations is limited to bitwise XOR, AND and NOT and rotations. There is no need for table-lookups, arithmetic operations, or data-dependent rotations.

2.1 Sponge Functions

Sponge function maps variable length input to variable length output. It is built upon a fixed length permutation and on a padding rule. This type of operation is known by the name sponge construction. The input is an element of $(\mathbf{Z}_2)^*$, i.e., a binary string of any length, and returns a binary string with any requested length, i.e., an element of $(\mathbf{Z}_2)^n$ with n a user-supplied value. A sponge function has two hash functions with a fixed output length and stream ciphers that have a fixed input length. It is a generalization of the above mentioned hash functions.[5] It operates on a finite state by iteratively applying the inner permutation to it, interleaved with the entry of input or the retrieval of output.
A wide spectrum of symmetric functions can be implemented using the sponge construction as well as its sister construction called duplex constructions. This includes hashing, reseedable pseudo random bit sequence generation, key derivation, encryption, message authentication code (MAC) computation and authenticated encryption.

2.2 The Sponge Construction

The sponge construction is a simple iterated construction for building a function F with variable-length input and arbitrary output length based on a fixed-length permutation (or transformation) f operating on a fixed number b of bits. Here b is called the **width**. The sponge construction operates on a state of b=sr+c bits. The value r is called the **bitrate** and the value c the **capacity**.

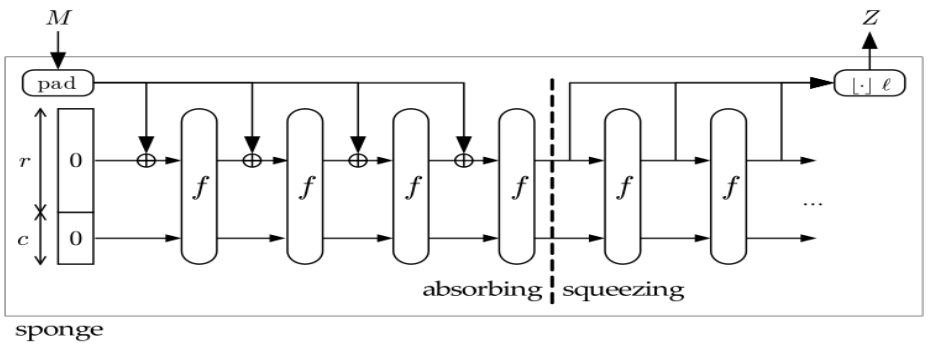

Figure 1: Sponge Construction

2.3 Sponge as a Reference of Security Claims

Random sponge functions act an alternative to random oracle function for security claims. A **random sponge** is an instance of the sponge construction with *f* chosen randomly from the set of transformations (or of permutations) over *b* bits. It is proved that a random sponge function is strong as a random oracle function save the effects induced by infinite memory. It can serve as a reference model for expressing compact security claims for iterated hash functions and stream ciphers. It also allows compact security claims that deal with all the possible properties that were not mentioned in the exhaustive list. When considering the success of the random sponge function, its success probability must be credited to the nature of the attack and also to the chosen parameters of the random sponge which may include its bitrate, capacity and whether it calls a random permutation or a random transformation.

2.4 Fixed-Length Permutation as a Versatile Cryptographic Primitive

The sponge construction allows building various primitives such as a hash function, a stream cipher or a MAC. The duplex construction that can be cited as the alternation of the input and output blocks at the same rate as the sponge construction like a full duplex construction. The duplex construction whose construction is similar to the sponge construction is said to have the same security as the latter. This allows one to implement an efficient reseedable pseudo random bit sequence generation and an authenticated encryption scheme requiring only one call to *f* per input block.

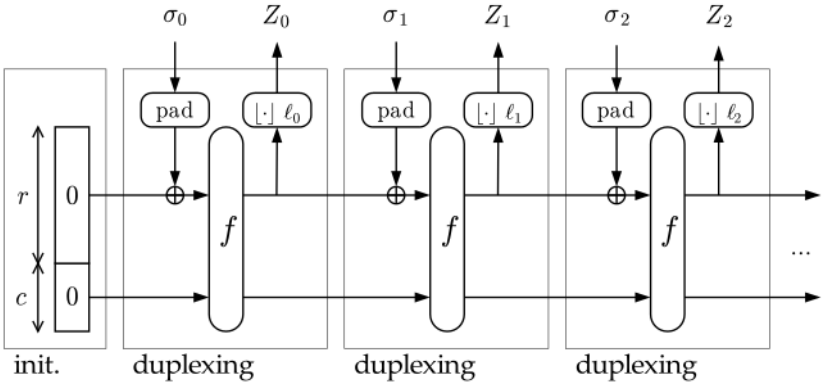

Figure 2: Duplex Construction

2.5. Keccak Tools

Keccak Tools is a set of C++ classes that can help analyze the **Keccak** sponge function family, designed by Guido Bertoni, Joan Daemen, Michael Peeters and Gilles Van Assche.
Keccak Tools provides the following features:

- the parameterized implementation of the seven Keccak-*f* permutations, from Keccak-*f*[25] to Keccak-*f*[1600], possibly with a specific number of rounds;
- the implementation of the *inverses* of the Keccak-*f* permutations;
- the generation of look-up tables for Keccak-*f*[25];
- the generation of GF(2) equations of the round functions and step mappings in the Keccak-*f* permutations and their inverses;
- the generation of optimized C code for the Keccak-*f* round functions, including several implementation techniques described in our document ***Keccak** implementation overview*:
    - bit interleaving,
    - lane complementing,
    - plane-per-plane processing,
    - early parity,
    - in-place processing;
- the implementation of the sponge construction using any transformation or permutation, and of the **Keccak** sponge function family;
- many classes and methods to assist differential and linear cryptanalysis (DC, LC).

Regarding DC and LC, KeccakTools supports the processing of linear and differential *trails*. This includes:

- for χ, the representation as an affine space of
  - the output differences compatible with a given input difference, and
  - the input masks compatible with a given output mask;
- for θ:
  - the representation of column parities in runs;
  - lower bounding the weight of any 2-round trail core with a given parity;
  - the exhaustive generation of 2-round trail cores with a given parity;
- for the round function, the iteration through all
  - the output differences compatible with a given input difference,
  - the input differences compatible with a given output difference (possibly up to a specified weight),
  - the input masks compatible with a given output mask,
  - the output masks compatible with a given input mask (possibly up to a specified weight);
- the representation and serialization of linear and differential trails;
  - including trail prefixes and trail cores;
- the generation of the conditions, expressed as equations([1]) in GF(2), for a pair to follow a given differential trail;
- the exhaustive forward and backward extension of trails up to a given weight and given number of rounds;
- the exhaustive generation of 2-round trail cores with a small number of active rows;
- the exhaustive generation of 3-round trail cores in the kernel up to a given weight:
  - the generation of knots and chains between knots;
  - the generation of vortices and their combination with knots and chains;
  - the implementation of a lower bound on the weight while adding knots, chains and vortices to limit the search.

2.6 Implementation

- KECCAK excels in **hardware performance**, with speed/area trade-offs, and outperforms SHA-2 by an order of magnitude. See for instance the works of Gürkaynak et al., Gaj et al., Latif et al., Kavun et al., Kaps et al. and Jungk presented at the Third SHA-3 Candidate Conference.
- KECCAK has overall **good software performance**. Although one could say that the SHA-3 finalists BLAKE and Skein have better software performance on PC, KECCAK shines when used in a mode exploiting parallelism. On AMD™ Bulldozer™, 128-bit and 256-bit security hashing tops at 4.8 and 5.9 cycles/byte, respectively. On Intel™ Sandy Bridge™, the same functions reach 5.4 and 6.9 cycles/byte. On constrained platforms, KECCAK has moderate code size and RAM consumption requirements.
- For modes involving a key, protecting the implementation against **side-channel attacks** is wanted. The operations used in KECCAK allow for **efficient countermeasures** against these attacks. Against cache-timing attacks, the most efficient implementations involve no table lookups. Against power analysis attacks and variants, countermeasures can take advantage of the quadratic round function.

2.7 Strengths of Keccak

KECCAK inherits the flexibility of the sponge and duplex constructions.

- As a sponge function, KECCAK has **arbitrary output length**. This allows to simplify modes of use where dedicated constructions would be needed for fixed-output-length hash functions. It can be natively used for, e.g., hashing, full domain hashing, randomized hashing, stream encryption, MAC computation. In addition, the arbitrary output length makes it suitable for tree hashing.

- As a duplex object, KECCAK can be used in **clean and efficient modes** as a reseedable pseudo-random bit generator and for authenticated encryption. Efficiency of duplexing comes from the **absence of output transformation**.
- KECCAK has a **simple security claim**. One can target a given security strength level by means of choosing the appropriate capacity, i.e., for a given capacity *c*, KECCAK is claimed to stand any attack up to complexity $2^{c/2}$(unless easier generically). This is similar to the approach of *security strength* used in NIST's SP 800-57.
- The security claim is **disentangled from the output length**. There is a minimum output length as a consequence of the chosen security strength level (i.e., to avoid generic birthday attacks), but it is not the other way around, namely, it is not the output length that determines the security strength level. For an illustration with the classical security requirements of hashing (i.e., collision and (second) preimage resistance), we refer to our interactive page.

The instances proposed for SHA-3 make use of a **single permutation** for all security strengths. This cuts down implementation costs compared to hash function families making use of two (or more) primitives, like the SHA-2 family. And with the same permutation, one can make **performance-security trade-offs** by way of choosing the suitable appropriate capacity-rate pair.

2.8 Design and Security

- KECCAK has a **thick safety margin**. In [KECCAK reference, Section 5.4], we estimate that the KECCAK sponge function should stand by its security claim even if the number of rounds is almost divided by two (i.e., from 24 down to 13 in the case of KECCAK-*f*[1600]).
- KECCAK was scrutinized by **third-party cryptanalysis**. For more details, we refer to the cryptanalysis page.
- We showed that the KECCAK-*f* permutations have provable lower **bounds on the weight of differential trails**.
- The design of the permutations follows the **Matryoshka principle**, where the security properties of the seven permutations are linked. The cryptanalysis of the smaller permutations, starting from the "toy" KECCAK-*f*[25], is meaningful to the larger permutations, and vice-versa. In particular, differential and linear trails in one KECCAK-*f* instance extend to symmetric trails in larger instances.
- The sponge and duplex constructions used by KECCAK are **provably secure against generic attacks**. This covers also the joint use of multiple KECCAK instances with different rate/capacity parameters.
- Unlike SHA-1 and SHA-2, KECCAK does not have the length-extension weakness, hence **does not need the HMAC nested construction**. Instead, MAC computation can be performed by simply pretending the message with the key.
- From the mode down to the round function, our design choices are fairly different from those in the SHA-1 and SHA-2 hash functions or in the Advanced Encryption Standard (AES). KECCAK therefore provides **diversity with respect to existing standards**.

III. CONCLUSIONS

The Keccak sha-3 hash algorithm is a tribute of the remaining hash algorithms like sha-1, sha-2.This hashing algorithm is mainly used for the securing the web from the third parities. The Keccak sha-3 is flexible and easy to design. The fundamental function of Keccak (sha-3) ,which consists of a number of simple rounds with logical operations and bit permutation. In future this Keccak algorithm can be implemented in such a way that, it can be more useful in security purpose on web and hard to break in practice by any adversary.

The proposed method involves security methods only. It prevents the effects from the third parties in the internet accessing and threats over it. By adopting these type of methods the damage by the third parties will be reduced but not fully rectified. Due to no rectification of the threats some damage will be there by implementing Keccak type of methods the damage will be reduced and one day all the threats will be rectified and a safe internet services will be enjoyed by everyone.

ACKNOWLEDGEMENTS

Vemu. Samson Devakumar, Project Manager, and V.Saikrishna, Programmer, E-world software development training center, SCRWWO, Vijayawada.

REFERENCES

[1]. G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, The Keccak Sponge Function Family, May2011.
[2]. "Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance", Department. Of Computer Science, University of California, Davis, California, 95616, USA, Appears in 11th International Workshop, Fast Software Encryption (FSE 2004), Delhi, India, February 5-7, 2004, LNCS 3017, pp. 371-388, 2004. www. cs. ucdavis. edu/~rogaway/papers/relates. Pdf
[3]. G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, The Sponge Functions Corner," Sponge Functions".
[4]. Hardware implementation of Kђèèюjin VHDL, 2011, http://keccak.noekeon.org
[5]. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, Sponge functions, Ecrypt Hash Workshop 2007, May 2007, also available as public comment to NIST from http://www.csrc.nist.gov/pki/HashWorkshop/Public Comments/2007 May.html.
[6]. http://en.wikipedia.org/wiki/Sponge_function.
[7]. S. Tillich, M. Feldhofer, M. Kirschbaum, T. Plos, J.-M. Schmidt, and A. Szekely, High-speed Hardware Implementations of BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein, Cryptology ePrint Archive, Report 2009/510, 2009.
[8]. Cryptographic sponge functions, January 2011, http://sponge.noekeon.org/.
[9]. The KECCAK sponge function family .http://keccak.noekeon.org/.

AUTHORS

V. Sukrutha, Pursuing B-Tech final year in K. L. University in Electronics and computer Engineering. Email Id: sukrutha3121@gmail.com