International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2020/191796 Volume 7, Issue 1, January – February (2020)

RESEARCH ARTICLE

Performance Evaluation of SHA-3(KECCAK) on ARM Cortex-A9 and Comparison with ARM 7TDMI and Cortex-M4

Pooja Kaplesh
Department of Computer Science and Engineering, Chandigarh University, India.
pooja.e7943@cumail.in

Published online: 04 February 2020

Abstract – Cryptography is the main area which deals with security. Cryptographic hash functions are essential elements of cryptography which is helpful for various security application areas. This paper presents cryptography, cryptographic hash functions, their properties, applications and some popular cryptographic hash standards. In this paper, KECCAK (SHA-3) algorithm has been tested on ARM based platforms (Cortex-A9) and analyzed its performance on the same. Comparative study is also discussed between Cortex-A9, Cortex-M4 and ARM7TDMI to check on which ARM platform KECCAK performs well. KECCAK has strong design construction that is "sponge construction" which differentiates it from other SHA-3 candidates.

Index Terms – Cryptography, ARM, KECCAK (SHA-3).

1. INTRODUCTION

This era may be termed as Data Age (like storage age and ice age). We need to implement security at each step to secure the data. Thousands of millions of data travel over the network for accomplishing successful transmission of data sent across the world. This data needs an assurance that the content sent must not be altered or accessed by any unauthorized party. And the area that deals with security issues of data is Cryptography. Cryptography is a field which is useful for planning, designing and implementing the cryptographic systems or cryptosystems. These systems contain different strategies for encryption and decryption purpose. The role of cryptography is to give secure solutions to a set of parties who wish to do a distributed task and don't confide in one another. The information is secured as: An original information or message that is to be encrypted is called plaintext and after applying encryption, the coded or encrypted message obtained is called cipher text [1]. So, Encryption or encipherment is the process of conversion of plaintext to cipher-text and the reverse process i.e. restoring the plaintext from cipher text is called Deciphering (decoding) or Decryption. At the same time, a cryptanalyst is concerned with the designing of attack methodologies that can break a cryptographic algorithm, for example, unauthorized or third person access to secret data or information. Handbook of Applied Cryptography lists out the basic security services provided by cryptography which are [3]:

1.1. Confidentiality

This service assures that private data remains confidential i.e. the data sent across the network aren't disclosed to the adversary.

1.2. Authentication

This service assures the identity of all users want to access system or resources.

1.3. Authorization

This service assuring that a certain user should have permissions to access a system or to perform any function.

1.4. Data Integrity

Data Integrity assuring that data is not modified illegally.

1.5. Non-Repudiation

This assuring against a person that denying a data or a communication that is originated by that person only.

In cryptography, mainly three basic cryptographic schemes are used namely symmetric cryptography systems (also called secret key cryptography), secondly, asymmetric cryptography systems (also named as public-key cryptography), and also hash functions or standards. Secret or Symmetric key cryptography provides confidentiality or privacy and Public-key Cryptography provides means for user authentication or verification and non-repudiation. On the other side, Hash standards or functions are also used in number of security related applications.[1] This paper describes about various cryptographic hash functions its categories and types. Section 2 explains about different type of Hash Functions and its applications. Section 3 focus is on KECCAK Hash algorithm its sponge structure, features and moreover its strength and weaknesses. In section 4, purpose of using ARM platform is discussed along with its various variants like Cortex-M4, Cortex-A9 and ARM7TDMI etc. ARM is considered among more powerful and highly used processor. Reason to select

IAR ARM workbench is also discussed. In section 5 methodology, tools and techniques are defined like CCStudio-v5 and IAR workbench. These tools are used to analyze the performance of KECCAK algorithm. Section 6 discuss about results obtained of KECCAK (in tabular and graphical form) performance analysis (in cycles per byte) on Cortex-A9. In section 7, a discussion on KECCAK performance on Cortex A9, ARM 7TDMI and Cortex-M4 is done to understand its performance on various ARM variants. Section 8 discuss about conclusion of performance comparison of KECCAK.

2. CRYPTOGRAPHIC HASH FUNCTIONS

Hash Function or hash algorithm is an important tool in cryptography and is used to achieve various security objectives such as Data Integrity, Message authentication and Digital Signatures. Hash Function in Cryptographic is an algorithm which takes variable size messages (M) as input and generates a fixed size message as output (h), called hash digest(hash value) or message digest (MD). Hash functions are introduced in cryptography in the 1976 by Diffie and Hellman [16]. The difference between Hashing and Encryption is that Hashing process is basically a one-way operation means it is not feasible to recover the actual data from the message digest or hash value. But Encryption is a two-way process which converts a plaintext into a cipher-text and at receiver side the cipher-text is transformed into its original plaintext, which is called decryption. Figure 1 illustrate about basic block diagram of Hash Function which takes a variable length input (M) and produces a fixed length output (h).

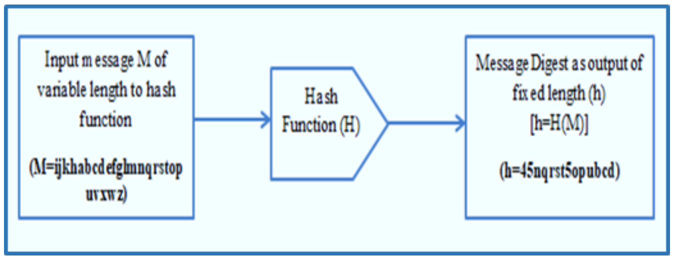

Input message M of variable length to hash function
(M=ijkhabcdefglmnqrstop uvxwz)
Hash Function (H)
Message Digest as output of fixed length (h)
[h=H(M)]
(h=45nqrst5opubcd)

Figure 1: Block Diagram of Cryptographic Hash Function

2.1. Security Requirements of Hash Algorithm

- A hash algorithm or function should be easy to calculate.
- Preimage Resistance: For each and every output of a hash function, it is mathematically impractical to obtain any input hash for that output.
- Weak collision/Second Preimage Resistance: It is mathematically inapplicable for a given input, to extract a second input hash for the same output.
- Collision Resistance: It is mathematically impractical to obtain two colliding inputs, i.e., z and z' such that z'! = z with h(z) = h(z'). This collision based characteristic of a hash algorithm is also known as Strong Collision Resistance. [3]

2.2. Cryptographic Hash Functions Standards

In all hash functions or algorithms, input is seen as a group or sequence of n bit blocks. This sequence of input is processed in an iterative fashion one block at a time to generate a output of n-bit hash function. There are many hash standards in cryptography like MD4, MD5 and SHA family etc.[4][5] All have their own construction e.g. MD5 and SHA-1, SHA-2 is totally based on the design of Merkle-Damgard construction. To design a collision resistant hash function, Merkle and Damgard presented an iterative structure that depends on fix length input collision resistant compression function (f). Till 1980s, there were few hash functions designed and most of them were broken very quickly after their introduction. During 1990s, various hash functions/standards were proposed or implemented, but very few of them have been broken [11]. Table 1 shows some of the Cryptographic Hash Standards along with their structure type and output length as:

Name	Output Length(in bits)	Structure Type
SHA-3(KECCAK)	Arbitrary	Sponge Construction
Skein	Arbitrary	Unique Block Iteration
SHA-512	512	Merkle and Damgard
SHA-384	384	Merkle and Damgard
SHA-256	256	Merkle and Damgard
SHA-224	224	Merkle and Damgard
SHA-1	160	Merkle and Damgard
MD5	128	Merkle and Damgard
MD4	128	Hash Standard
MD2	128	Hash Standard
Tiger	192/160/128	Hash Standard
Whirlpool	512	Hash Standard

Table 1: Some Cryptographic Hash Functions Standards

2.3 Application Areas of hash functions

Hash functions have wide range of applications based on Symmetric key as well as on Asymmetric keys. Bart Preneel in his Ph.D thesis has defined many applications few of which are highlighted here [13].

2.3.1. Authenticate Users

While authenticating or verify the identity of a user at the time of login process, hash functions are used. To avoid access to system, the entered passwords are saved in message digest (or called hash value) form. When user attempts to login and put the password, the hash or message digest of these passwords is calculated and then compared with the message or hash digest saved in the database. If the digest value matches, then user is considered to be authenticated and then login process is done successfully, otherwise user is not considered to be authenticated.

2.3.2. Digital Signature Implementation

To achieve the goal of authenticity and non-repudiation, digital signatures play an important role. Only MAC and Hash Functions cannot fulfill the security objective of Digital Signatures. To avoid contention between source and receiver Diffie and Hellman realized the requirement for a message dependent digital signature (or fingerprint).

Hash algorithms play a very useful role to optimize the use of digital signature schemes. Signature must have same size as message if hash functions are not used. The advantage is that there is no need to generate signature for the whole transmitted message which needs to be authenticated instead the sender only signs the hash digest of the message by using a hash function(or signature generating algorithm). After that the sender/source sends the message and the signature to the intended receiver. When receiver gets this information he compute the digest or hash value of the message with the help of same hash function as used by sender and verifies the signature by comparing digest values with the help of signature verification algorithm. This technique basically saves a lot of computational overhead included to sign and verify the messages in the absence of hash functions [14].

2.3.3. Digital Time Stamping

Most of documents like text, audio and video are provided in digital form nowadays. A lot of various simple tools and techniques are provided to modify these digital documents. Therefore there is need of a mechanism to verify when these documents were created to last modified. Hash functions and digital signatures are possible techniques that could be used to implement these mechanisms. To index data in hash tables, to determine duplicate data and uniquely identify files, for fingerprinting, for generating random numbers and as checksums to determine incidental data corruption, Hash functions play essential role. They are mostly used in all places in the field of cryptology where effective information processing is desired. [40].

3. SECURE HASH FAMILY (SHA)

SHA is a class or family of cryptography hash algorithms or functions that is released by National Institute of Standards and Technology (NIST) organization. This SHA family mainly refers to:

SHA-0: It was federal information processing standard (FIPS) 180 under name SHA, released by NIST in 1993.[24]

SHA-1: It is updated version of SHA. SHA-1 is basically a 160-bit hash function which is similar to MD5 algorithm [17].

SHA-2: It contains four hash functions, each having different block sizes, called SHA-256, SHA-224, SHA-384 and SHA-512[18].

SHA-3: After the attacks on SHA-0 and SHA-1 were published, NIST felt need of a more secure standard which can overcome the weaknesses of earlier hash standards. Two open workshops and a public consultation period was planned to achieve the goal. NIST on November 2, 2007 made an open call for SHA-3 competition to define a new crypto-graphic hash family [18] [25].

NIST got outstanding feedback from worldwide cryptographic community throughout the selection process. On the basis of internal assessment of the second-round competitors and the public feedback, NIST selected Grøstl Blake, JH, KECCAK, and Skein (CSRC, 2010) as finalists of competition on December 9, 2010. KECCAK is finalized as the *winner* of SHA-3 family on October 3, 2012[39].

3.1. KECCAK Specification

3.1.1. Design Principle

KECCAK is a cryptographic hash algorithm implemented by Joan Daemen , Guido Bertoni , Gilles Van Assche and Michael Peeters. [34]. This algorithm is a group of sponge functions/construction. The basic role of sponge construction is that it can produce outputs (hash digests) of any size length and therefore a single function may be used to generate different output lengths. KECCAK works in two stages: absorption and squeezing [10]. It makes use of a state ‘b’ throughout the hashing process and the main methods of the algorithm consist of the sponge inherited functions pad, absorb and squeeze.

These functions rely on the permutation f denoted by KECCAK-f[]. There are seven KECCAK- f permutations are applied on this as Keccak-f[b], with values b=25, 50, 100, 200, 400, 800 and 1600. In SHA-3 contest, the largest permutation like Keccak-f[1600] is proposed and smaller permutations are useful to implement in constrained environments. Every permutation comprises of the iteration or cycle of a basic round function. The selection of operations is restricted to bitwise operations like AND, NOT, XOR, and rotations. [6] [24]

3.1.2. Specifications

Guido Bertoni and team members [21] has defined that KECCAK is based on sponge construction having members KECCAK[r, c] use two parameters named:

- Bit-rate (r)
- Capacity (c)

The input data or message is fragmented into pieces of length r bits each. The importance of r and c is that higher values of capacity(c) improve its security level and likewise higher values of bit rate(r) improve its speed [21]. The combined bits of r and c define the width (b) of the KECCAK- f permutation (also called width of state in the sponge structure and are confined to values in {25, 50, 100, 200, 400, 800, 1600}. The state is managed as an array representation of 5×5 lanes, each having length w ∈ {1, 2, 4, 8, 16, 32, 64}. The suggested length for each lane is 6. The total number of rounds algorithm has, depends upon the width of permutation, and calculated using nr = 12+2l that is 24 rounds for KECCAK-f[1600].[28]

3.1.3. Sponge Construction

Cayrel P.L and his team members [29] have defined the design of sponge construction consisting of a fixed permutation function (f) taking arbitrary or variable length input and outputs. This construction is based on following steps:

- Firstly, reset the state (combination of r and c bits) to initial value zero.
- To make input message length multiple of r bits, Pad or insert extra bits in input message.
- Absorbing phase: In this phase, the r bits of each block (p_i) are XORed with the first r bits of the state, and then forwarded to permutation function f. When all input blocks are processed, construction is switched to the squeezing phase.
- Squeezing phase: The first r(bit rate) bits of the state are reverted as output block (z_i) when all blocks got processed in the absorbing phase. On the state this process is continued till all the blocks are processed to get required output length. The final c bits of the state are modified by permutation function f only, but not by XOR-operation in the absorbing phase. These bits are additionally never used as output in the next squeezing phase.

In Figure 2, P_i is the message blocks, function f defines the permutation or compression function, and Z_i in squeezing phase are output blocks of the hash function.

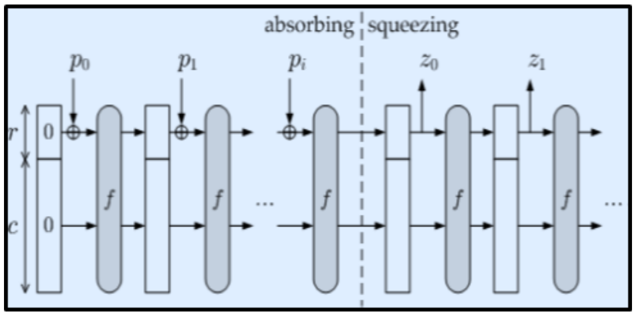
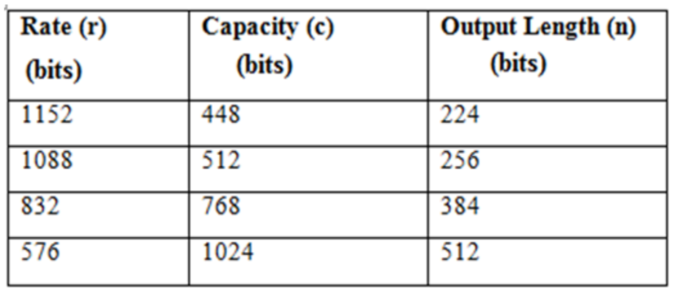


Figure 2: The Sponge Construction [21]

3.1.4. Proposals for the KECCAK standards

According to the requirement of NIST, the candidate algorithm should hold at least four different output length variants like n ∈ {224, 256, 384, and 512}. Hence, the following four fixed-output-length variants are defined:



Rate (r) (bits)	Capacity (c) (bits)	Output Length (n) (bits)
1152	448	224
1088	512	256
832	768	384
576	1024	512

Table 2: Proposed Parameters for Algorithm

Table 2 defines input and output variants of KECCAK algorithm. In addition, KECCAK is also proposed with default parameters, where one may reduce the output to get the desired output length. For this arbitrary output bit-rate is taken as 1024 and capacity is as 576 [28].

3.1.5. KECCAK- f Permutations

The KECCAK-f function takes the current state as input and performs a number of calls to the underlying permutation f. Each call updates the state and passes it as argument on to the next call. Each call is known as a round and the number of rounds is calculated by the permutation width b as $n_r = 12 + 2l$, where $b = 25w$ and $w = 2^l$. The Permutation works in five steps named as: theta, chi, pi, rho and iota. These permutation steps are repeated in each round of KECCAK permutation [33]. All these steps operate on the state (b) so it is necessary to introduce about state and its parts.

3.2. Strengths and Weaknesses

3.2.1. Strengths

The sponge construction has many advantages over other constructions using a compression function and these are as follows [29]:

a. Variable length output: The sponge construction generates outputs of any length according to user’s choice using a single function.

b. Simple Construction: The sponge construction is having a simple design. It is a very secure function and has proved secure against generic attacks.

c. Flexibility: Security of the sponge function can be enhanced at the expense of speed by managing bit-rate for capacity utilizing the same permutation function.

d. Functionality: A sponge construction can be used as a MAC function, stream cipher and a pseudorandom bit generator because of its large length outputs and proven security limits with respect to general attacks.

e. This algorithm has better performance in terms of hardware implementation.

f. KECCAK has a special support for parallel implementation.

g. This algorithm is considered to be slowest of the NIST finalists but still possess one of the smallest outlines, with small implementation, and more than sufficient performance in real world. [33]

3.2.2. Weaknesses

a. KECCAK is slowest in software implementation amongst the SHA-3 finalists.

b. Since KECCAK follows sponge construction it is secure to upper limit of $2^{c/2}$ where c is the capacity. So if a higher capacity is chosen, naturally higher security but a decrease in performance [28].

4. ARM (ADVANCED RISC MACHINES)

ARM is a 32-bit processor depends on RISC (Reduced Instruction Set Computer) computer structure and a product of ARM Holdings. The first ARM processor was developed in 1985 at Acorn Computers Limited named as Acorn RISC Machine until the emergence of Advanced RISC Machines. To widen the utilization of its technology ARM Limited was renowned as a separate company in 1990.[9] And it has developed to turned into the biggest microchip IP organization on the planet and the ARM processor covers each region of microprocessor applications, from extremely low cost installed microcontrollers, up to exceptionally elite multi-center processors. ARM core is not a single core, but an entire group of structures having identical design principles and a basic instructions set design [10].

ARM has three classes of processor: Classic, Embedded and Application based. ARM has developed a variety of processors that are combined into different classes according to the core they operate on. Some of them are Cortex-M3, Cortex-A9, Cortex-M4, Cortex-R4 and ARM7 as shown in figure 3. Among the selected platforms ARM7TDMI is the oldest but still covers most of the market and besides that Cortex-M3, Cortex M4 and Cortex A9 support majority of Hashing Applications.[7] This paper presents the assessment of Keccak on Cortex A9. Therefore some features of Cortex-A9 are highlighted as:

Cortex-A9 is one of the application specific processors supported by a huge set of functions to provide a high level performance and low-power solution over both general purpose and application specific designs. This processor runs 32-bit ARM instructions, 8-bit Java bytecodes in Jazelle state and 16-bit and 32-bit sized thumb instructions. It has shipped various digital TV, smart phones, consumer and enterprise applications. Key functionalities of the Cortex-A9 core are [32]:

- High performance floating point structure module
- To maximize performance and minimize power consumption, power optimized L1 caches combine minimum access latency techniques.
- Ensures reliable development of security applications like electronic payment and digital rights management applications called TrustZone Technology.
- Use Cortex-A9 NEON Media Processing Engine (MPE) to enhance application specific performance.

Figure 3 below shows various variants of ARM processor along with its performance parameter and capacity. Application based cortex processors are better in performance than classic and embedded processors.

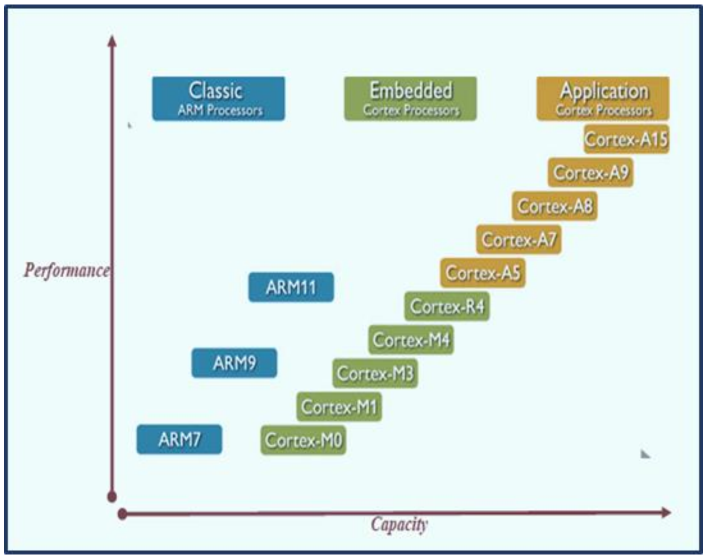


Figure 3: Classes of ARM Processor [10]

4.1. IAR workbench for ARM

IAR Workbench provides an integrated development environment to design and debug ARM-based embedded applications. It provides extended support for various ARM devices, and hardware debug systems and produce high efficient code [42]. It supports various ARM cores. Some of them are: Cortex-A9, Cortex-R7, Cortex-M3, ARM11, ARM9 and ARM7TDMI. It is a high-performance platform, C/C++ compiler and debugger tool suitable for applications that are based on 8-bit, 16-bit and 32-bit microcontrollers. Following are the several reasons to choose the IAR Embedded Workbench [35]:

- Reliable and most commonly used embedded tool in the market.
- Fully integrated software debugger with performance assessment and power visualization.
- The market's broadest range of processors - more than 7400 devices
- It complies with industry standards, such as C++, and is frequently used in certified applications development.
- Clean, robust user interface made up for embedded development with an easy workflow in mind.
- Excellent Performance in Code Quality and Optimizations in Speed and Size.

5. METHODOLOGY

Tools of data collection and assessment

The tools used are:

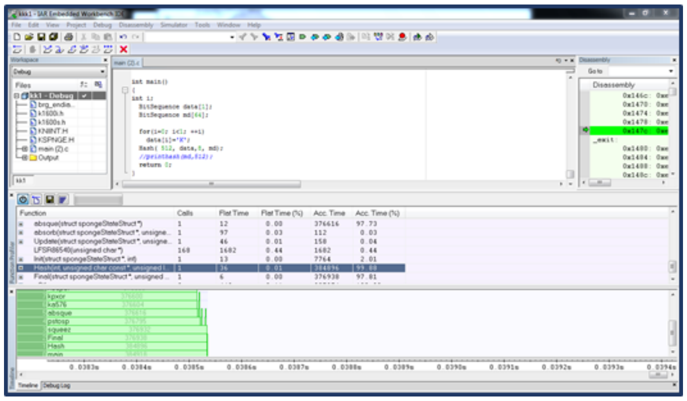
- IAR Embedded Workbench for ARM
- CU(LM4F232H5QD-Evaluation Board)
- CCStudio-v5 (Code Composer Studio)

When the code was debugged on IAR Embedded Workbench in Simulator mode, there were multiple facilities available through the interface to see the performance of the code. To obtain the cycles/byte and the total time taken by the function running in the machine environment, I chose Function profiler and Timeline. KECCAK was given 1418 sets of input bytes (from 1 byte to 4288 bytes), on Cortex A9 environment. The screenshot are as follows:

5.1. Cortex-A9

Figure 4 above shows various performance evaluation parameters of KECCAK on Cortex-A9 for 1 byte input. Here main focus is on execution time which is approximate 668 cycles per byte.

Figure 5 shows execution time for 4288 input byte which is approximate 654 cycles per byte.

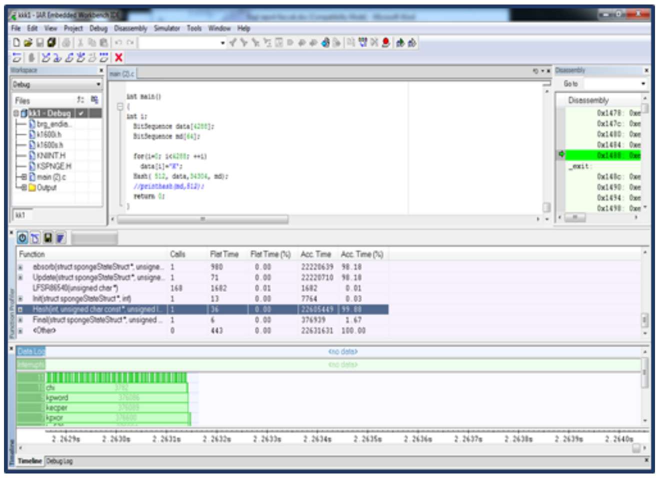


```
int main()
{
int i;
  BitSequence data[1];
  BitSequence md[64];

  for(i=0; i<1; ++i)
    data[i]='X';
  Hash( 512, data,8, md);
  //printhash(md,512);
  return 0;
}
```

Function	Calls	Flat Time	Flat Time (%)	Acc. Time	Acc. Time (%)
absque(struct spongeStateStruct*)	1	12	0.00	376616	97.73
absorb(struct spongeStateStruct*, unsigne...	1	97	0.03	112	0.03
Update(struct spongeStateStruct*, unsigne...	1	46	0.01	158	0.04
LFSR86540(unsigned char*)	168	1682	0.44	1682	0.44
Init(struct spongeStateStruct*, int)	1	13	0.00	7764	2.01
Hash(int, unsigned char const*, unsigned l...	1	36	0.01	384896	99.88
Final(struct spongeStateStruct*, unsigned ...	1	6	0.00	376938	97.81

Figure 4: 1 Byte Input for Cortex-A9



```
int main()
{
int i;
  BitSequence data[4288];
  BitSequence md[64];

  for(i=0; i<4288; ++i)
    data[i]='X';
  Hash( 512, data,34304, md);
  //printhash(md,512);
  return 0;
}
```

Function	Calls	Flat Time	Flat Time (%)	Acc. Time	Acc. Time (%)
absorb(struct spongeStateStruct*, unsigne...	1	980	0.00	22220639	98.18
Update(struct spongeStateStruct*, unsigne...	1	71	0.00	22220710	98.18
LFSR86540(unsigned char*)	168	1682	0.01	1682	0.01
Init(struct spongeStateStruct*, int)	1	13	0.00	7764	0.03
Hash(int, unsigned char const*, unsigned l...	1	36	0.00	22605449	99.88
Final(struct spongeStateStruct*, unsigned ...	1	6	0.00	376939	1.67
<Other>	0	443	0.00	22631631	100.00

Figure 5: 4288 Byte Input for Cortex-A9

6. RESULTS

KECCAK has variable output sizes: 224, 256, 384, 512-bits. I have analyzed its performance for output size 512 (KECCAK-512). The results for Cortex-A9 are calculated on simulator and when the actual machine is used for implementation, the results may vary. I have calculated results on Cortex-A9 simulation and results for Cortex M4 and ARM7TDMI are taken from reference [42]. KECCAK-512 gives an average speed of 657 cycles/byte on Cortex-A9. For evaluating the performance of algorithm, I have given 1418 no. of input bytes (from 1 byte to 4288 byte). Then I formed a set of 50 inputs and calculated their average access time in cycles/byte. The execution time for 1 input byte is 668.2222222 cycles/byte and for input byte 4288 it is 654.092853cycles/byte. The table and graph representation between number of bytes and cycles/byte depicts that with the increasing input size, the execution time of the algorithm was reduced.

Table 3 below shows performance of KECCAK-512 in cycle per bytes of set of 50 inputs. Here, some of input values provided to KECCAK-512 to calculate its execution time on IAR workbench tool for Cortex-A9.

6.1. Results Obtained on Cortex A9

Sr. No.	Input Size (Bytes)	Hash() Access Time (Cycles)	Cycles per byte
1	1	384896	668.2222222
2	2	384901	668.2309028
3	3	384900	668.2291667
4	4	384915	668.2552083
5	5	384915	668.2552083
6	6	384914	668.2534722
7	7	384913	668.2517361
8	8	384918	668.2604167
9	9	384920	668.2638889
10	10	384919	668.2621528
11	20	384925	668.2725694
12	30	384921	668.265625
13	40	384929	668.2795139
14	50	384931	668.2829861
15	60	384927	668.2760417
16	70	384939	668.296875
17	80	761633	661.1397569
18	90	761638	661.1440972
19	100	761636	661.1423611
20	110	761641	661.1467014
21	120	761641	661.1467014
22	130	761646	661.1510417
23	140	761653	661.1571181
24	150	1138251	658.7100694
25	160	1138260	658.7152778
26	170	1138257	658.7135417
27	180	1138263	658.7170139
28	190	1138259	658.7146991
29	200	1138260	658.7152778
30	250	1514878	657.4991319
31	300	1891494	656.76875
32	350	1891499	656.7704861
33	400	2268118	656.2841435
34	450	2644733	655.9357639
35	500	2644741	655.937748
36	550	3021355	655.6759983
37	600	3397968	655.4722222
38	650	3774571	655.3074653
39	700	3774589	655.3105903
40	750	4151199	655.1766098
41	800	4527815	655.0658275
42	850	4527822	655.0668403
43	900	4904443	654.9736912
44	950	5281046	654.8916171
45	1000	5281063	654.8937252
46	1390	7540757	654.5796007
47	2178	11683858	654.3379256
48	2965	15826331	654.1968833
49	3753	19969415	654.1344012
50	4288	22605449	654.092853

Table 3: Cortex-A9 Simulator Results for KECCAK-512

Figure 6 shows the graphical representation of Keccak-512 results in cycle per bytes. The execution time reduced with increased input size.

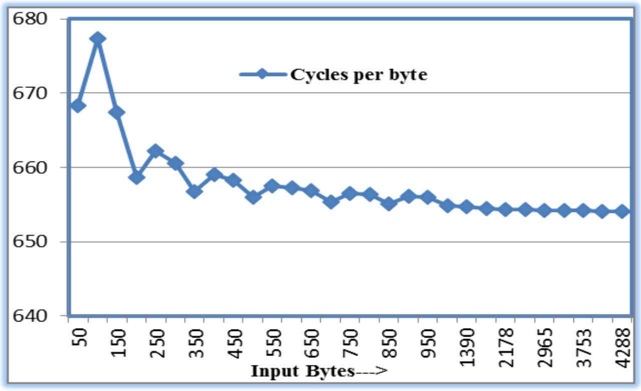


Figure 6: Input Byte vs. Cycles/Byte for Cortex-A9

7. KECCAK PERFORMANCE ON CORTEX A9, ARM 7TDMI AND CORTEX-M4

Table 4 below gives an average results obtained on Cortex A9, ARM 7TDMI and Cortex-M4. On Cortex-A9, KECCAK-512 has an average speed of 657 cycles per byte (as discussed in result section). These results are calculated on simulator (IAR workbench). On Cortex M4 and ARM7TDMI, KECCAK-512 takes 739 cycles per byte and 213 cycles per byte respectively [42]. So in comparison I can say that Keccak-512 performs well on Cortex A9 than ARM7TDMI (simulation results). It gives better performance on Cortex M4.

Processor	Average Speed (Cycles/Byte)
Cortex A9	657
ARM7TDMI	739[43]
Cortex M4	213 [41]

Table 4: Average Processor Speed

8. CONCLUSION

ARM is one of the major players in the current mobile device market. Increasing market value of ARM processors direct us to choose the platform for the observations and testing of performance of the algorithm on them. According to the results presented in result section (both tabular and graphical form), KECCAK-512 works faster on Cortex A9 with increased input size. These results are calculated on simulation but if evaluation is analyzed on hardware results may vary. While comparing to other ARM variants, KECCAK-512 gives better result on Cortex M4 hardware or evaluation board but works slow on ARM7TDMI simulation. Cortex-A9 is observed fast as compared to ARM7TDMI. As number of input size is increasing, there is decrease in cycles/byte. So, in conclusion, KECCAK-512 does not give good results on ARM7TDMI but it performs well on Cortex-A9.

REFERENCES

[1] Diffie, W. & Hellman, M.E. (1976) “New directions in Cryptography” IEEE Transactions on Information Theory, Vol. IT-22, No.-6.
[2] Demgard M, R.C. (1989). Advances in Cryptology - CRYPTO '89 Proceedings, Lecture Notes in Computer Science Vol. 435. A Certified Digital Signature. G. Brassard, Springer-Verlag.
[3] Preneel B. &Dobbertin H. (1990) “The Cryptographic Hash Function RIPEMD-160” , Katholieke Universiteit Leuven, ESAT-COSIC K. Mercierlaan 94, B-3001 Heverlee, Belgium 2 German Information Security Agency, Germany.
[4] Rivest R. (1990)“The MD4 message digest algorithm”, MIT Laboratory for Computer Science; http://tools.ietf.org/html/rfc1186.
[5] Rivest R. (April, 1992) “The MD5 message digest algorithm”, IETF RFC 1321; www.rfc-editor.org/rfc/rfc1320.txt.
[6] Federal Information Processing Standards Publication 180-1(April 1995) “Announcing the Secure Hash Standard”, NIST,17,Retrieved from: http://www.umich.edu/~x509/ssleay/fip180/fip180-1.htm
[7] Technical Reference Manual ARM7TDMI (2001), Rev3, April-Retrieved from http://infocenter.arm.com/help/topic/com.arm.doc.../DDI0210B.pdf
[8] Federal Information Processing Standards Publication 180-2 , (August 2002) “Specifications for the Secure Hash Standard”, Retrieved from: https://csrc.nist.gov/csrc/media/publications/fips/180/2/archive/2002-08-01/documents/fips180-2.pdf
[9] Andrew N. Sloss, Dominic Symes(2004) *ARM System Developer's Guide*- Designing and Optimizing System Software, Morgan Kaufmann Publishers
[10] Andrew N. Sloss & Chris Wright “ARM System Developer's Guide-Designing and Optimizing System Software”(2004), Retrieved from https://doc.lagout.org/electronics/Game%20boy%20advance/ARM_BOOKS/ARM_System_Developers_Guide-Designing_and_Optimizing_System_Software.pdf.
[11] Preneel B. (2003) “Assessment and Design of Cryptographic Hash Functions” https://www.cosic.esat.kuleuven.be/publications/thesis-2.pdf
[12] Damgård I.,(2004),”A Design Principle for Hash Functions", Retrieved from *https://www.iacr.org/archive/eurocrypt2001/20450165.pdf*.
[13] Preneel B. *(2005)* “Cryptographic Hash Functions and MAC Algorithms Based on Block Ciphers” Katholieke Universiteit Leuven (Belgium).
[14] Stallings W. (2006), *Cryptography and Network Security*, Pearson Education, Inc, Edition – 5
[15] Gauravaram P.(2007), “Cryptography hash functions:Cryptassessment, design and its applications” Information security Institute, Queensland university of technology.
[16] Von D. & Knopf C.(November 2007) “Cryptographic Hash Functions”, Retrieved from https://www.thi.uni-hannover.de/fileadmin/forschung/arbeiten/knopf-da.pdf.
[17] Manuel, S. & Peyrin, T. (2008). Lecture Notes in Computer Science Collisions on SHA-0 in One Hour, Retrieved from: https://link.springer.com/chapter/10.1007/978-3-540-71039-4_2
[18] White paper ARM “The ARM Cortex-A9 Processors” (September 2009),Document Revision 2.0.,Retrieved from www.arm.com/files/pdf/armcortexa-9processors.pdf
[19] Kelsey J, Nandi M. & Paul S. (September 2009) “Status Report on the First Round of the SHA-3 Cryptographic Hash Algorithm Competition”- NIST.
[20] Arpan M. (Augus,2010) “Efficient software implementation of SHA-3 candidates on 8-bit AVR microcontrollers”, Indian Institute of Technology, Kanpur.
[21] Bertoni G., Daemen J. & Michaël Peeters et.al (June, 2010) “Keccak sponge function family main document” version 2.1.
[22] Ros A. & Sigurjonsson C. (2010) “Assessment of SHA-3 Candidates CubeHash and keccak” Bachelor of Science Thesis Stockholm, Sweden.
[23] Technical Reference Manual Cortex-M4 (2010) Revision r0p1, June-Retrieved from http://infocenter.arm.com/.../DDI0439C_cortex_m4_r0p1_trm.pdf
[24] Mourad Gouicem, (2010) “Comparison of seven SHA-3 candidates software implementations” Retrieved from https://eprint.iacr.org/2010/531.pdf
[25] Andreicheva L.(2011) “Attacks on SHA-3 candidate functions: Keccak and Blue MidnightWish (BMW)” Department of Computer Science B. Thomas Golisano College of Computing and Information Sciences, Rochester Institute of Technology Rochester, New York, 3rd January.
[26] Bertoni G., Daemen J. & Michaël Peeters et.al (2011)” The KECCAK SHA-3 submission”, STMicroelectronics and NXP Semiconductors.
[27] Bertoni G., Daemen J. & Michaël Peeters et.al (January, 2011)” The KECCAK SHA-3 References”, STMicroelectronics and NXP Semiconductors.
[28] Bertoni G., Daemen J. & Michaël Peeters et.al(2011) “Cryptographic sponge functions” Version 0.1 .

RESEARCH ARTICLE

[29] Cayrel P.L, Hoffmann G. & Schneider M, (2011),"GPU Implementation of the Keccak Hash Function Family", CASED – Center for Advanced Security Research Darmstadt, Germany

[30] Chen R. (2011), "New Techniques for Cryptassessment of Cryptographic Hash Functions", the Senate of the Technion, Israel Institute of Technology, Ph. D. Thesis.

[31] Manuel S.,(2011) "Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1,Retrieved from: https://link.springer.com/article/10.1007/s10623-010-9458-9 .

[32] Cortex™-A Series Version: 2.0 Programmer's Guide (2011) Received from http://www.dsi.fceia.unr.edu.ar/...0184%20Curso%20CORTEX%20

[33] Federal Information Processing Standards Publication 180-4, (February 2011) Draft "Announcing the Secure Hash Standard (SHS)", , Retrieved from: https://ws680.nist.gov/publication/get_pdf.cfm?pub_id=910977

[34] R. Sobti, S.Anand & Dr. G. Geetha, (2012): "Performance comparison of Grøestl, JH and BLAKE – SHA-3 Final Round Candidate Algoritms on ARM Cortex M3 Processors" Computer Science, Lovely Professional University, Punjab, India.

[35] Getting Started with IAR Embedded Workbench" Tool 6.4, (April 2012), GSEW-3,IAR Systems. Retrieved from http://supp.iar.com/FilesPublic/UPDINFO/005691/common/doc/EW_GettingStarted.ENU.pdf

[36] National Institute of Standards and Technology (NIST), (2012) "SHA-3 Selection Announcement". Retrieved from : https://www.nist.gov/news-events/news/2012/10/nist-selects-winner-secure-hash-algorithm-sha-3-competition

[37] Sadhu S. ,"Keccak discussion" 9th January(2012) http://spiegel.cs.rit.edu/~hpb/public_html/Lectures/20112//S_T/Src/32/keccak.pdf

[38] Sterling J. G., (2014) "Hash Functions In Cryptography" Master of science thesis shodhganga.inflibnet.ac.in/jspui/bitstream/10603/199858/1/aroy-thesis.pdf.

[39] Chris Bentivenga, Frederick Christie & Michael Kitson , (2017),"Keccak Final Paper" , Retrieved from https://technodocbox.com/66430335-Network_Security/Cryptographic-hash-functions.html.

[40] Preneel B. "Cryptographic Hash Functions: An Overview" ESAT-COSIC Laboratory, K.U.Leuven K. Mercierlaan 94,. Retrieved from: https://www.esat.kuleuven.be/cosic/publications/article-289.pdf

[41] W. Stornetta & S. Haber (2017), "How to time-stamp a digital document", *Journal of Cryptology*, vol. 3(2), pp. 99-111,.

[42] Rajeev Sobti and Geetha Ganesan, (2018) "Performance Evaluation of SHA-3 Final Round Candidate Algorithms on ARM Cortex–M4 Processor" Retrieved from https://www.igi-global.com/gateway/article/190857.

Author

Pooja Kaplesh is working as Assistant Professor in department of Computer Science and Engineering at Chandigarh University. She did her B. Tech –M. Tech from Lovely Professional University, Punjab. She has published four research papers in international journal and published a chapter in IGI Global. Her research areas are mainly networking and cryptography & Network Security.