Design of a New Cryptographic Hash Function – Titanium

Mohammad A. AlAhmad
Public Authority for Applied Education and Training, College of Basic Education, Kuwait

Article Info

Article history:

Received Nov 7, 2017
Revised Jan 29, 2018
Accepted Feb 12, 2018

Keywords:

AES
Hashfunction
SF
Sponge
Titanium

ABSTRACT

This paper introduces a new cryptographic hash function that follows sponge construction. Paper begins with outlining the structure of the construction. Next part describes the functionality of Titanium and cipher used. A competition between block cipher and stream cipher is presented and showed the reason of using block cipher rather than stream cipher. Speed performance is calculated and analyzed using state-of-art CPUs.

Corresponding Author:

Mohammad A. AlAhmad,
Public Authority for Applied Education and Training
College of Basic Education
Computer Science Department, P.O. Box 34567 Adliyah, 73205, Kuwait City, Kuwait.
Email: malahmads@yahoo.com

1. INTRODUCTION

The usage of the internet connections was confined in communication between peers. With the new technology revolution, the internet connections became widely used for all purposes especially for commerce. The development of the internet and the connection mediums led to reduction in the costs as well as the increase of users and uses. Banks, organization and companies around the world are depending on the internet connections to make their deals. This environment is very fertile for crackers and it is risky for those organizations to keep using the internet with no defense lines since the backbone if their information technology infrastructure is the internet connections. Information security was implemented for this purpose and to protect and immune systems to crack. There are many security procedures to ensure data integrity, confidentiality and availability such as encrypting, digital signatures and MACs. However, hash functions are light powerful ways to protect and verify data. Hash function is non-invertible function and it has two main components. The construction and cipher used. Hash functions should go in one direction such that crackers cannot obtain plain text from final result. It is heavily used in passwords databases in production servers. Hash functions used to verify the integrity of data over peers. It takes arbitrary input size, processes it and produces fixed length output known as “Digest”. It should be resist to preimage, second pre-image and collisions. There are several constructions for hash function. Each one of them has its own scheme to produces the digest.

In this paper, Titanium hash function is introduced. It follows sponge construction which has a state width of bitrate and the capacity denoted by “w”, “r” and “c” respectively (w=r+c).

Capacity is a security parameter stored in the state matrix and its value is protected from changes over the operations and never affect the bitrate values. The usage of capacity is to split the digest length from the security level of hash function. Bitrate is where the data is being processed. Sponge construction has three stages, absorbing, squeezing and truncation stage. Absorbing is the operation of preparing and inserting data to the blender while squeezing is the operation of building the digest and preparing it to the final stage lastly, truncation is the operation of cutting the digest to rightmost or leftmost. According to Claude's theory, there are two primitives for hash functions, Diffusion and Confusion. Diffusion is to spread the influence of one bit insertion to the message in order to hide the statistical properties of the construction. Confusion is hide the relationship between the input and the output and keep it obscure. To carry out those two primitives, there are three main components. Permutations\Transformations: applying several layers on S-boxes in order to maintain higher diffusion, Logical Functions: The most common arithmetic operation used is XOR. It is not possible to disclose the inputs from a given output after XORing. Titanium hash function is a newly constructed hash function that follows sponge construction and uses SP block cipher. It has a state of 1600bit. Bitrate is 576bit and the capacity is 1024bit. The capacity should be 2 times of bitrate to preserve the basic security criteria. Table1 shows all the hash functions that used a sponge as its construction.

Table 1. Shows All Sponge Function Hashes Regardless Its Construction and Cipher Mode

Sponge F.	Year	structure	Cipher mode	Ciphers name
GLUON[4]	2012	T-sponge	Stream	X-FCSR-v2 and F-FCSR-H-v3
PHOTON[5]	2011	P-Sponge	Block	AES , PRESENT, LED
QUARK[6]	2010	P-Sponge	Block and Stream	KATAN / Grain
SipHash[7]	2012	JH-style T-Sponge	-	BLAKE and Skein
SPN-Hash[8]	2012	JH-style P-Sponge	Block	AES , LED and PHOTON
SPONGENT [9]	2011	P-Sponge	Block	PRESENT
Spritz[10]	2014	Sponge	Stream	RC4
Keccak[11]	2008	P-Sponge	Block	Noekeon and Rijndael
LHash[12]	2013	Feistel-PG	-	Extended sponge function
DOUBLE-A	2015	Sponge	Stream	Salsa20

2. RESEARCH METHDOLOGY

2.1 Block and Stream ciphers

Ciphers should work within rules called mapping function. Data is processed either if it is block mode or stream mode. Stream mode is efficient with large amount of data input when the input length is unknown and it deals with bits rather than a blocks while block cipher is dealing with chunks of data. In terms of costs, block cipher is cheaper to implement and easy to manipulate, on the other hand stream cipher is much harder to implement and verifying outputs but it is faster in executing. The flexibility of block mode allows building anything from stream ciphers to hash functions or MACs. Both, stream and block ciphers look secure enough to use. However in our case, block mode is our choice. Input data is known, thanks to padding rule used, easier to manipulate and add extra operations to the construction without touch its properties.

2.2 Sponge construction choice

Sponge construction is an iterated function that operates on fixed size internal states (state width). It goes through three main operations. Firstly, padding rule by adding enough bits to the message to let it fit the construction and initialize the initial value to zeros. Secondly, absorbing by XORing the message blocks into the state then return the first r bits as a part of output and lastly, squeezing phase which is used for finalizing the round as Figure 1 shown below.

2.3 Titanium.

Titanium is a new hash function that follows sponge construction. It has 1600bit state. 1024bit for capacity which is a security parameter and 576bit for bitrate where the function will operate. It produces 576bit digest output. The 1024bit capacity is used for security claims such that no attack will be applicable under complexity of 2n with considering the hash performance on microprocessors. Titanium uses SP block cipher. The state width of Titanium is 1600bit, all initialized to zeros distributed in 10*20 matrix state. Each element is one block and four blocks representing one word size.

Bitrate is distributed in 8*9 matrix. There is no different if the distribution was between any numbers of cells since it is blocking and at the end of the operation, the difference will be in the digest distribution with the same security level. For security reasons, length padding has been used. It is to append "1" then zeros until the last number which will be the message length

$$P(m) = (M||P) + 1 + 0 * + mn \text{ where mn is the message length.}$$

The simplest padding rule is to append zeros to the message, but it is risky to use since collisions in this padding rule could be easily obtained. To harden the padding rule, length padding rule has been implemented.

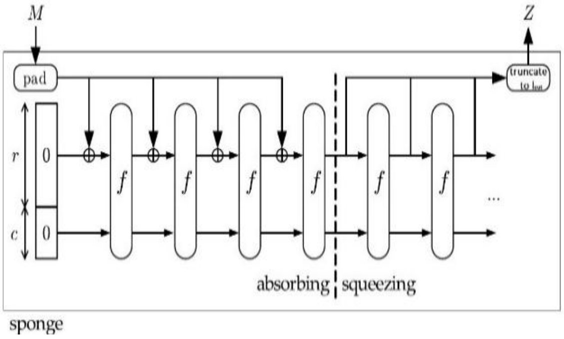


Figure 1 Sponge Construction

r	r	r	r	r	r	r	r	r	c	c	c	c	c	c	c	c	c	c	c	c
r	r	r	r	r	r	r	r	r	c	c	c	c	c	c	c	c	c	c	c	c
r	r	r	r	r	r	r	r	r	c	c	c	c	c	c	c	c	c	c	c	c
r	r	r	r	r	r	r	r	r	c	c	c	c	c	c	c	c	c	c	c	c
r	r	r	r	r	r	r	r	r	c	c	c	c	c	c	c	c	c	c	c	c
r	r	r	r	r	r	r	r	r	c	c	c	c	c	c	c	c	c	c	c	c
r	r	r	r	r	r	r	r	r	c	c	c	c	c	c	c	c	c	c	c	c
r	r	r	r	r	r	r	r	r	c	c	c	c	c	c	c	c	c	c	c	c
c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c
c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c

Figure2. Highlighter cells is bitrate, C is capacity

2.3 Function

After padding and preparing the message to the blender, Titanium operates literately using sub-byte, Convert row, shifting and add round key.

2.4 Cipher.

SF is a 512bit block cipher. It iterates on four operations: sub-byte, Convert row, shifting and add round key. It is based on AES algorithm with substitution permutation network design. It takes a block of plain text and key of 512bit input and applies its operations on S-boxes to produce the ciphertext as an output. Message block and key are distributed in 4*16 matrix let SP iterates on 24 rounds to produce the final ciphertext. SP operates on 512bit input in the state as Figure 3 shown below. Each state element is one byte and four elements equals one word size as shown below.

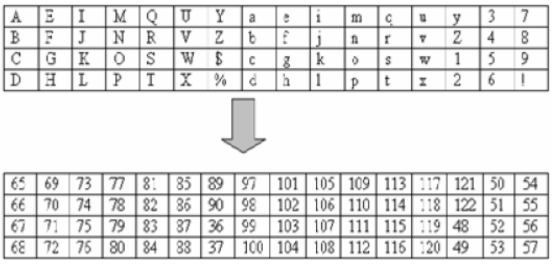
11	22	33	44	55	66	77	88	99	0	AA	BB	CC	DD	EE	FF
GG	HH	II	JJ	KK	LL	MM	NN	OO	PP	QQ	RR	SS	TT	UU	VV
WW	XX	YY	ZZ	11	22	33	44	55	66	77	88	99	0	11	22
AA	BB	CC	DD	EE	FF	AA	NN	OO	PP	QQ	RR	SS	TT	VV	VV

Figure 3 SP State

Cipher steps:

1. Sub-byte round. Data location is substituted from S-box. This process repeats on all element's location as shown below in figure 4.
2. Convert Row Round. In this stage each data element is converted to binary form after that reading from right to left then and converted back to hexadecimal values as shown in figure 5.
3. Shifting Round. In this stage, shifting will be performed on the state that was the output from Convert Row Round stage. Two rows will be shifted then, XORed with the mother state as in figure 6.

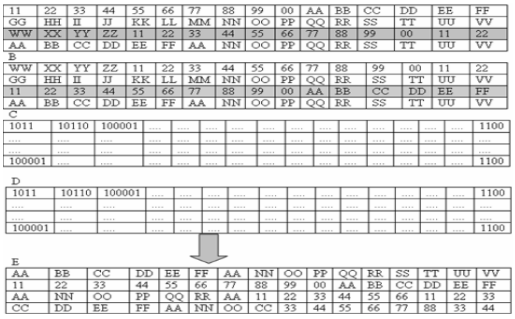
4. Add Round Key. In this stage, each byte is XORed with sub key. Sub key is obtained from Key depending on key expansion schedule as shown in figure 7.



A	E	I	M	Q	U	Y	a	e	i	m	q	u	y	3	7
B	F	J	N	R	V	Z	b	f	j	n	r	v	Z	4	8
C	G	K	O	S	W	$	c	g	k	o	s	w	1	5	9
D	H	L	P	T	X	%	d	h	l	p	t	x	2	6	!

65	69	73	77	81	85	89	97	101	105	109	113	117	121	50	54
66	70	74	78	82	86	90	98	102	106	110	114	118	122	51	55
67	71	75	79	83	87	36	99	103	107	111	115	119	48	52	56
68	72	76	80	84	88	37	100	104	108	112	116	120	49	53	57

Figure 4 Sub Byte



11	22	33	44	55	66	77	88	99	00	AA	BB	CC	DD	EE	FF
GG	HH	II	JJ	KK	LL	MM	NN	OO	PP	QQ	RR	SS	TT	UU	VV
WW	XX	YY	ZZ	11	22	33	44	55	66	77	88	99	00	11	22
AA	BB	CC	DD	EE	FF	AA	NN	OO	PP	QQ	RR	SS	TT	UU	VV

B

WW	XX	YY	ZZ	11	22	33	44	55	66	77	88	99	00	11	22
GG	HH	II	JJ	KK	LL	MM	NN	OO	PP	QQ	RR	SS	TT	UU	VV
11	22	33	44	55	66	77	88	99	00	AA	BB	CC	DD	EE	FF
AA	BB	CC	DD	EE	FF	AA	NN	OO	PP	QQ	RR	SS	TT	UU	VV

C

1011	10110	100001	....	....	....	....	....	....	....	....	....	....	....	....	1100
....	....	....	....	....	....	....	....	....	....	....	....	....	....	....	....
....	....	....	....	....	....	....	....	....	....	....	....	....	....	....	....
100001	....	....	....	....	....	....	....	....	....	....	....	....	....	....	1100

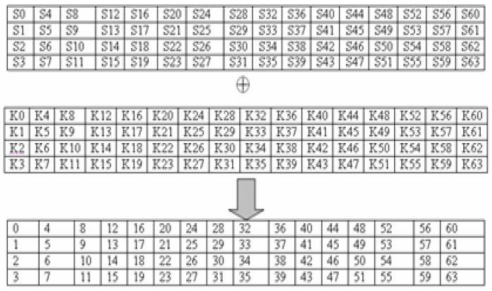
D

1011	10110	100001	....	....	....	....	....	....	....	....	....	....	....	....	1100
....	....	....	....	....	....	....	....	....	....	....	....	....	....	....	....
....	....	....	....	....	....	....	....	....	....	....	....	....	....	....	....
100001	....	....	....	....	....	....	....	....	....	....	....	....	....	....	1100

E

AA	BB	CC	DD	EE	FF	AA	NN	OO	PP	QQ	RR	SS	TT	UU	VV
11	22	33	44	55	66	77	88	99	00	AA	BB	CC	DD	EE	FF
AA	NN	OO	PP	QQ	RR	AA	11	22	33	44	55	66	11	22	33
CC	DD	EE	FF	AA	NN	OO	CC	33	44	55	66	77	88	33	44

Figure 5 Convert Round Key



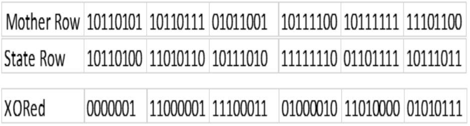
S0	S4	S8	S12	S16	S20	S24	S28	S32	S36	S40	S44	S48	S52	S56	S60
S1	S5	S9	S13	S17	S21	S25	S29	S33	S37	S41	S45	S49	S53	S57	S61
S2	S6	S10	S14	S18	S22	S26	S30	S34	S38	S42	S46	S50	S54	S58	S62
S3	S7	S11	S15	S19	S23	S27	S31	S35	S39	S43	S47	S51	S55	S59	S63

$\oplus$

K0	K4	K8	K12	K16	K20	K24	K28	K32	K36	K40	K44	K48	K52	K56	K60
K1	K5	K9	K13	K17	K21	K25	K29	K33	K37	K41	K45	K49	K53	K57	K61
K2	K6	K10	K14	K18	K22	K26	K30	K34	K38	K42	K46	K50	K54	K58	K62
K3	K7	K11	K15	K19	K23	K27	K31	K35	K39	K43	K47	K51	K55	K59	K63

0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61
2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62
3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63

Figure 6 Shift Round



Mother Row	10110101	10110111	01011001	10111100	10111111	11101100
State Row	10110100	11010110	10111010	11111110	01101111	10111011
XORed	0000001	11000001	11100011	01000010	11010000	01010111

Figure 7 Add Round Key

Table 2 Speed Analysis SP Cipher

Input Size	Time(seconds)		
	AES	Blowfish	SF
49	65	36	61
59	45	36	43
100	89	61	79
247	120	90	112
321	167	134	168
694	243	256	212
899	223	256	259
963	243	187	206
5345	1224	1376	1216
7310	1435	1543	1363
Averege	388	395	377

2.5 Performance Analysis On Cipher.

Cost of the algorithm is a precious circumstance especially in real time ciphering. The security of algorithm is better with bigger security parameters. However, the cost of implementation will be high and using bigger parameters might be not reasonable. Different file sizes have been tested on the SP algorithm to test its performance. According to SP speed study [reference here], 100KB data takes 79 Milliseconds to encrypt. Comparing it with AES and Blowfish, it is faster on bigger files.

2.6 Titanium

Since the capacity will never affect or enter the operations, Titanium is distributed in 10*20 forge 1600 bits matrix to preserve the cipher criteria. Bitrate will be 8*9 and the rest of the elements will be the capacity. Half of the element is spirited out and sub-byte with the elements row from the previous state as Figure 8.

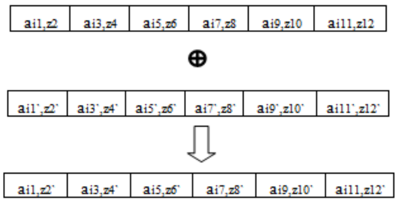


Figure 8 Sub-Byte Operation Titanium

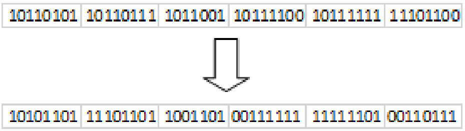


Figure 9. Binary Row

Same for the cipher steps. Row is converted to binary values then read from right to left then reconvert it to hexadecimal values. Same operation will repeat for all rows as Figure 9. The state rows are converted to binary form then XORed with the mother state rows. Figure 10 below illustrates the process. Systems needed a small powerful function tool to verify the data over peers without overloading it with operations. One of the best tools to do that is hash function. It has been designed to verify data over peers without overloading servers with many operations. Since there is no key in hashes, in this stage, the state matrix will be XORed with the previous one as Figure 11.

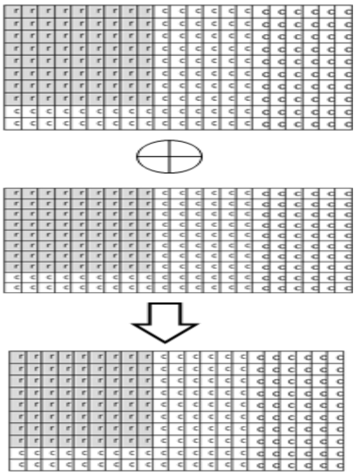


Figure 10 XOR states

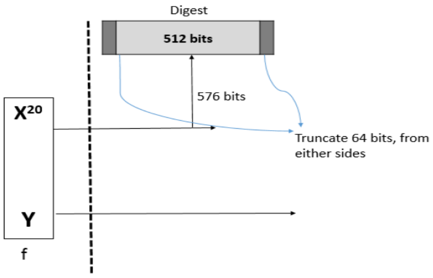


Figure 11 Squeezing Phase

2.7 Truncation

After obtaining 576bit output digest, truncation will take its place to produce 512bit final digest after 24 rounds by spiriting out the bitrate to the right most or left most full digest as Figure 11 shown. Since 100KB takes around 79 milliseconds, that means 1KB takes 0.1 milliseconds. With some calculations, 576bit takes around 0.444 milliseconds.

3. CONCLUSION

Titanium is a sponge hash function. It acts as a random sponge by following and achieving the designing principles. It has a state width of 1600bit matrix distributed to 576bit to bitrate and 1024bit to capacity. Titanium uses SP block cipher which consist of three operations and follows sponge construction structure. Titanium final digest is truncated and its preserves the confusion and diffusion primitives.

ACKNOWLEDGMENT

This work supported and funded by The Public Authority of Applied Education and Training, Research project No (B-17-06), Research Title (Design and Analysis of a New Hash Function- Titanium hash function).

REFERENCES

[1] Narayanan, Arvind; Bonneau, Joseph; Felten, Edward; Miller, Andrew; Goldfeder, Steven (2016). Bitcoin and cryptocurrency technologies: a comprehensive introduction. Princeton: Princeton University Press. ISBN 978-0-691-17169-2.G. Bertoni, J. Daemen, M. Peeters, and

[2] Jakobsson, Markus; Juels, Ari (1999). "Proofs of Work and Bread Pudding Protocols". Communications and Multimedia Security. Kluwer Academic Publishers: 258–272.

[3] Dwork, Cynthia; Naor, Moni (1993). "Pricing via Processing, Or, Combatting Junk Mail, Advances in Cryptology". CRYPTO'92: Lecture Notes in Computer Science No. 740. Springer: 139–147.

[4] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

[5] Grinberg, R. (2012). Bitcoin: An innovative alternative digital currency. Hastings Sci. & Tech. LJ, 4, 159.

[6] Blockgeeks, blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/.

[7] A. Back, "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.

[8] King, S. (2013). Primecoin: Cryptocurrency with prime number proof-of-work. July 7th.

[9] Buterin, Vitalik. "What Proof of Stake Is And Why It Matters." Bitcoin Magazine, Bitcoin Magazine, 26 Aug. 2013, bitcoinmagazine.com/articles/what-proof-of-stake-is-and-why-it-matters-1377531463/.

[10] Alyssa Hertig. "What Is Ether?" CoinDesk, 21 Apr. 2017, www.coindesk.com/information/what-is-ether-ethereum-cryptocurrency/.

[11] Crane, Fabian Brian. "Proof of Work, Proof of Stake and the Consensus Debate." Cointelegraph, 20 Dec. 2014, cointelegraph.com/news/proof-of-work-proof-of-stake-and-the-consensus-debate.

[12] Alahmad, M. A., I. Al-shaikhli, et al. (2013). *"Jouxmulticollisions attack in sponge construction"*. The 6th International Conference on Security of Information and Networks (SIN), 2013 6th International Conference on, ACM.

[13] AlAhmad, M. A., & Alshaikhli, I. F. (2013). Broad view of cryptographic hash functions. *International Journal of Computer Science Issues*, 10(4), 239-246.

[14] S. Wu, D. Feng, W. Wu, J. Guo, L. Dong, and J. Zou. (Pseudo) preimage attack on round-reduced Grøstl hash function and others. In Canteaut [9], pages 127–145.

[15] Wang, Xiaoyun, Hongbo Yu, and Yiqun Lisa Yin. "Efficient collision search attacks on SHA-0." *Advances in Cryptology–CRYPTO 2005*. Springer Berlin Heidelberg, 2005.

[16] Nandi, M. and S. Paul (2010). "Speeding up the wide-pipe: Secure and fast hashing." Progress in Cryptology-INDOCRYPT 2010: 144-162.

[17] Eli Biham and Orr Dunkelman, "A Framework for Iterative Hash Functions - HAIFA," Cryptology ePrint Archive, 2007. [Online]. http://eprint.iacr.org/2007/278

BIOGRAPHY OF AUTHOR

Mohammad Abdulateef AlAhmad received his bachelor degree in computer engineering from university of the pacific in 2002, his master in computer engineering from Gulf university in Bahrain in 2011, and his PhD degree in computer science from international Islamic University Malaysia (IIUM) in 2015. His research area is information security which focuses on cryptographic algorithms and protocols. My favourite specific research topics are designing and analysis of hash functions, cryptocurrency and cryptography in general. My favorite hash functions are Gear, Double A and Titanium cryptographic hash functions.