Journal of Engineering and Applied Sciences 11 (1): 162-166, 2016
ISSN: 1816-949X
© Medwell Journals, 2016

Design and Implementation of the Lighter Version of Skein Cryptographic Hash Function Using Verilog HDL

Aparna Lakshmi Mooragondi, Sushrut Prabhakar, Avinash Yadlapati and Ranjan K. Senapati
Department of Electronics and Communication Engineering, KL University,
Vaddeswaram, Guntur, Andhra Pradesh, India

Abstract: Secure Hashing set of rules-1 (SHA-1), created through National Institutes of Standards and Technology in 1993 is a hashing algorithm that was used to supply message digest. In 2005, cryptanalysts determined attacks on SHA-1 suggesting that the algorithm may not be comfy sufficient for ongoing use. The drawbacks of SHA-1 result in invention of new algorithm, SHA-2 which possessed excessive stage of protection. One of the drawbacks of this algorithm becomes no longer likeminded with running systems. In 2012, NIST performed a hash feature opposition to select a standard for the latest SHA-3 cryptosystem of which skein was into one of the five finalists. This research is aimed towards implementing "Lighter version of Skein" which is based on the skein hash function in Verilog and its FPGA simulation using the Xilinx Virtex 7. The design for both encryption and decryption of lighter version of skein has been discussed in this study. The additives, it uses are threefish block cipher and the unique block iteration. The overall performance attributes of lighter version of skein are discussed below. The principle goal is to examine and compare the latency, throughput and delay of lighter version of skein with skein-256 and various other traditional block ciphers and cryptosystems.

Key words: Skein, cryptography, network security, Verilog, FPGA

INTRODUCTION

Because, the internet and different forms of electric communication become widely wide-spread, electronic protection of digital data is becoming increasingly vital. Cryptography is used to shield E-mail messages, passwords, credit card information, corporate statistics and so forth. A cryptographic hash function permits one to effortlessly verify that a few input data fits a stored hash value but makes it tough to assemble any information that might hash to the same value or find any two unique data pieces that hash to the same value. The ciphers in hash functions are built for hashing: they use large keys and blocks can effectively change keys every block and have been designed for providing resistance to attacks on the key.

Skein is a family of hash functions with three different state sizes: 256, 512 and 1024 bits (Aumasson *et al*., 2009). Each of these state sized can support any output size (Ferguson, 2010). Lighter version of skein, the cryptosystem discussed in this study has been derived from skein-256 (A few components have been changed to improve speed and throughput). It uses the threefish cipher with some minor changes as the main module for encryption of data in UBI chaining mode, the same as skein. The main principle behind lighter version of skein is that a higher number of simple rounds provide more security than a lower number of complex rounds (Bellare *et al*., 1990). This lighter version of skein employs 72 rounds of encryption before it computes the cipher text. Like skein, it makes the use of 18 different keys (which are derived from the user key) thus making brute force attack on the system highly difficult.

MATERIALS AND METHODS

Architecture: Lighter version of skein is a hash function based on skein-256 and uses similar components (with slight changes in functionality). Lighter version of skein, like skein is a low memory cryptosystem and requires <100 bytes of memory for execution.

Components of lighter version of skein

Threefish: Threefish is the tweakable block cipher at the core of skein, denied with a 256, 512 and 1024-bit block size. The threefish block takes three inputs, plaintext, userkey (256 bits each) and a tweak of 128 bits. The output of the function is a ciphertext of 256 bits.

Unique Block Iteration (UBI): UBI is a chaining mode that uses threefish to build a compression function that maps an arbitrary input size to a fixed output size (Ferguson *et al*., 2010).

Corresponding Author: Aparna Lakshmi Mooragondi, Department of Electronics and Communication Engineering, KL University, Vaddeswaram, Guntur, Andhra Pradesh, India

Table 1: Permulation LUT

LUT points															
No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7	No. 8	No. 9	No. 10	No. 11	No. 12	No. 13	No.14	No. 15	No. 16
91	168	83	119	194	23	45	54	149	155	203	216	115	71	11	3
106	140	231	243	97	147	151	101	89	94	124	185	166	117	100	122
41	74	42	30	21	17	198	218	250	254	87	68	33	38	102	154
163	177	112	32	133	247	253	111	47	233	103	130	69	139	217	183
28	135	78	37	49	65	171	204	35	196	227	76	136	201	143	191
215	7	79	0	10	134	114	242	169	153	70	20	237	164	131	137
105	121	184	150	64	208	162	129	99	82	132	199	235	26	14	175
239	222	219	109	107	135	170	167	63	207	22	4	12	200	232	238
84	67	34	19	5	179	245	224	144	95	72	44	27	31	16	226
211	180	157	138	104	123	252	240	160	159	181	96	81	86	6	29
13	202	187	85	36	53	80	176	182	210	241	209	113	88	6	29
93	58	61	1	249	161	146	145	225	18	178	2	66	228	212	116
148	197	229	214	213	118	230	246	9	24	8	25	73	152	90	59
43	62	46	248	126	110	141	125	142	158	173	190	174	188	221	172
223	156	220	255	15	77	92	75	251	236	186	189	234	127	40	39
192	193	57	60	55	56	98	128	48	206	195	205	52	50	51	224

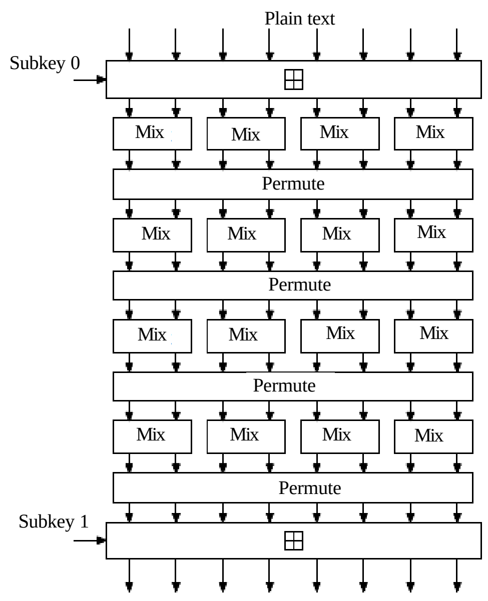


Fig. 1: Threefish block cipher for encryption

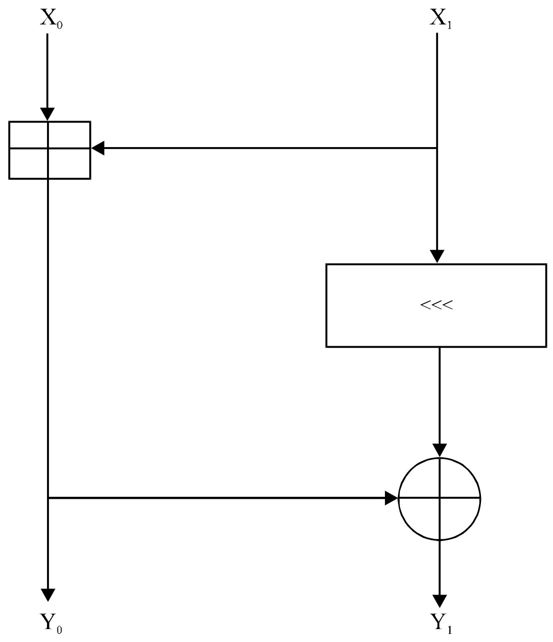


Fig. 2: Mix function

The threefish block cipher is the heart of the Lighter version of skein cryptosystem. The main components of threefish are the mix and permute function as shown in Fig. 1. The Mix and Permute functions have been described in detail below Eq. 1 show MIX operation::

$$Y_0 = (X_0+X_1)\text{mod}^{264};\ Y_1 = (X_1{<}{<}{<}16) \oplus Y_0$$

Encryption; MIX function: Figure 2 shows the mix function. The MIX function is an integral part of the threefish ciphers. It takes two inputs of 64-bit each and performs three basic operations on the data words; Addition modulo 2^{64}, arithmetic shift (Wakerly, 2006) and XOR operation.

Permutation operation: Table 1 shows a sample Look Up Table (LUT) used in the threefish cipher. The permutation function is used to diffuse the outputs obtained from the previous rounds of MIX function and scramble them to create new inputs for the following MIX round Eq. 2 show key schedule algorithm:

$$sk_0 = k_0;\ sk_1 = k_1 + t_0$$
$$sk_2 = k_2 + t_1;\ sk_3 = k_3 >> \{2\}$$

Key scheduling algorithm: After every four rounds of MIX and permute, a subkey is added to current threefish state. For the first round, user key and plaintext are XORed and then after every four rounds, Key Scheduling algorithm uses tweak and the key of previous rounds to generate a subkey (Burr, 2006).

Table 2: Inverse permutation LUT

LUT points															
No. 1	No. 2	No. 3	No. 4	No. 5	No.6	No. 7	No. 8	No. 9	No. 10	No. 11	No. 12	No. 13	No. 14	No. 15	No.16
83	179	187	15	123	132	158	81	202	20	84	14	124	160	110	228
142	37	185	131	91	36	122	5	201	203	109	140	64	159	35	141
51	44	130	72	164	67	45	239	238	32	34	208	139	6	210	56
248	68	253	254	252	165	7	244	245	242	177	207	243	178	209	120
100	69	188	129	43	60	90	13	138	204	33	231	75	229	66	82
166	156	105	2	128	163	157	42	173	24	206	0	230	176	25	137
155	20	246	104	30	23	46	58	148	96	16	116	175	115	218	55
50	172	86	12	191	29	197	3	174	97	31	149	26	215	212	237
247	103	59	94	106	52	85	117	76	95	147	61	17	214	216	78
136	183	182	21	192	8	99	22	205	59	47	9	225	146	217	153
152	181	102	48	93	65	28	119	1	88	118	70	223	218	220	111
167	49	186	133	145	154	168	63	98	27	234	162	221	235	219	79
240	241	4	250	73	193	38	107	125	77	161	10	71	251	249	121
101	171	169	144	190	196	195	80	11	62	39	114	226	222	113	224
135	184	143	74	189	194	198	18	126	57	236	108	233	92	127	112
151	170	87	19	255	134	199	53	211	180	40	232	150	54	41	227

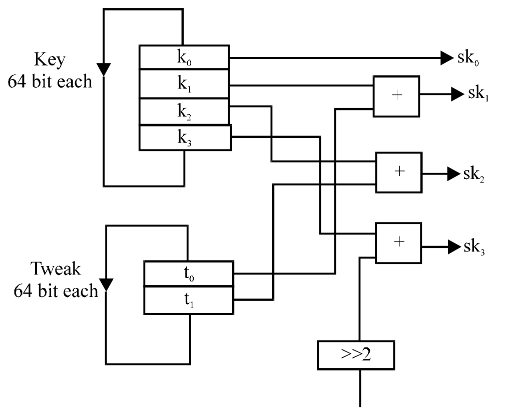


Fig. 3: Key schedule algorithm

The key from the previous round is divided into four 64 bit words and the tweak is divided into two 64 bit words. Key Scheduling algorithm makes use of modular addition 2^{64} and logical right shift of a constant arbitrary value decided by the designer. The 464 bit words from these operations concatenate to form a new subkey show in Fig. 3.

Thus, key schedule algorithm generates a new subkey every time. It can be thus concluded that due to the addition of a 256 bit subkey after every four rounds, a high level of security can be achieved by making the cryptosystem immune to cryptanalysis.

Decryption: Decryption is the exact inverse of the encryption algorithm. To decrypt the ciphertext generated from the threefish block cipher, the text is passed through the inverse threefish cipher which consists of the inverse mix function and the inverse permutation table. Also, the keys are supplied in reverse order, i.e., the 18th subkey of the encryption system is the 1st subkey of the decryption algorithm and vice versa. The inverse MIX function and the inverse permutation table have been discussed below.

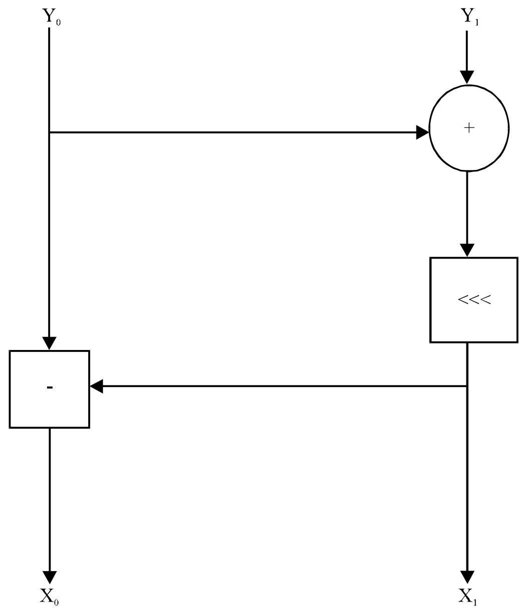


Fig. 4: Inverse MIX function

The point to be noted is that there is no change in the key scheduling algorithm. The keys used for encryption are used for decryption as well, in reverse order.

Inverse mix function: The inverse MIX function is an integral part of the inverse threefish cipher. It is created by using the inverse of the operators of the MIX function (Tilich, 2009). It consists of three digital systems, XOR operator, left shifter and subtractor. The subtraction in this module is performed with the help of a carry look ahead adder. Figure 4 shows the block diagram of the inverse MIX function.

Inverse permutation operation: Table 2 shows a sample Look Up Table (LUT) used in the inverse threefish cipher.

The inverse permutation function is used to diffuse the outputs obtained from the previous rounds of inverse MIX function and scramble them to create new inputs for the following inverse MIX round.

RESULTS AND DISCUSSION

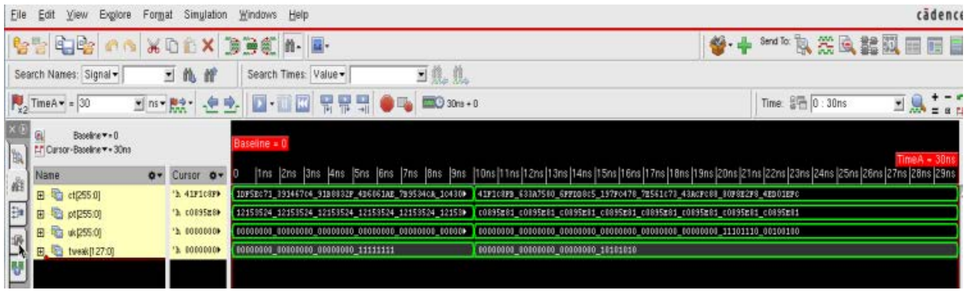
The design was simulated using structural and dataflow modelling. The components defined have been used to design the threefish cipher. It can be said that the threefish module is the top module of the design. This section shows the Verilog simulation and synthesis results of the threefish cipher as well as the inverse threefish cipher. Figure 5 and 6 shows the software simulation of both encryption and decryption of data in the lighter version of skein cryptosystem. The tool used for the implementation was NC simulator by cadence.

Figure 5 shows the simulation results of 72 rounds of encryption in which an XOR operation, right shift operation, carry look ahead addition, permutation look up table and a key scheduling algorithm are used. For this plain text of 256 bits, tweak o f 128 bits and user key of 256 bits are fed as inputs. The output is the cipher text (encrypted data) which is of 256 bits.

Figure 6 shows the simulation results of 72 rounds of decryption in which an XNOR operation, left shift operation, carry look ahead subtraction, inverse permutation look up table are used. The keys that are used in the encryption are shared here. For this cipher text of 256 bits, tweak of 128 bits and user key of 256 bits are fed as inputs. The output is the plain text (decrypted data) which is of 256 bits. From Fig. 5 and 6, it is clear that the text that is fed for encryption is the data that is observed after decryption. Upon successful simulation of the Verilog code, the same system was implemented in hardware using the Xilinx Virtex 7 FPGA simulator.

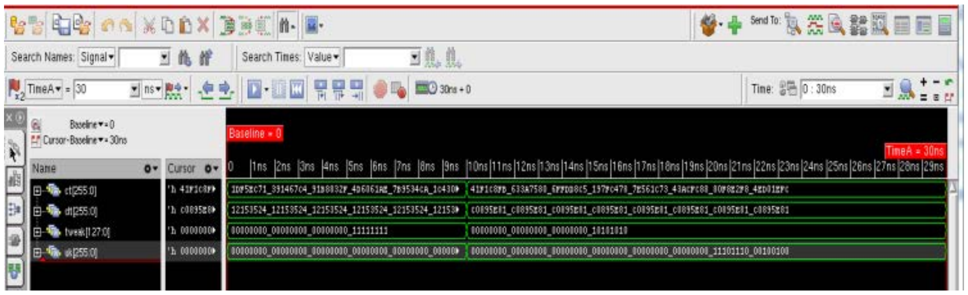
After successful simulation in NC Simulator (Cadence), the code was synthesized for FPGA implementation using Xilinx ISE 10.1. Then, the program was dumped into the memory into the FPGA board Xilinx Virtex 7. Since, this is the lightweight version of the skein hash cryptosystem (Kamal and Hossain, 2004); it was easily implemented on the FPGA device, Xilinx Virtex 7 (Webster and Lukowiak, 2011). Details of the hardware implementation have been shown below:

- Lighter version of skein-256 (encryption)
- Selected device: 7v×1140tflg1930-2
- Number of slice LUTs: 25301 out of 712000 (3%)
- Number used as logic: 25301 out of 712000 (3%)
- Number of fully used LUT-FF pairs: 0 out of 25301 (0%)
- Number of IOs: 896
- Number of bonded IOBs: 896 out of 1100 (81%)
- Delay: 920.721ns (Levels of logic = 1954)
- Total real time to Xst completion: 121.00 sec



```
Name          Cursor
ct[255:0]     'h 41F1C8F▸  1DF5EC71_391467C4_91B8832F_4B6061AE_7B9534CA_1C430▸  41F1C8FB_633A7580_6FFDD8C5_197FC478_7E561C73_43ACFC88_80F8E2F8_4ED01EFC
pt[255:0]     'h c0895E8▸  12153524_12153524_12153524_12153524_12153524_12153▸  c0895E81_c0895E81_c0895E81_c0895E81_c0895E81_c0895E81_c0895E81_c0895E81
uk[255:0]     'h 0000000▸  00000000_00000000_00000000_00000000_00000000_00000▸  00000000_00000000_00000000_00000000_00000000_00000000_11101110_00100100
tweak[127:0]  'h 0000000▸  00000000_00000000_00000000_11111111                   00000000_00000000_00000000_10101010
```

Fig. 5: Software simulation of the encryption algorithm (threefish cipher)



```
Name          Cursor
ct[255:0]     'h 41F1C8F▸  1DF5EC71_391467C4_91B8832F_4B6061AE_7B9534CA_1C430▸  41F1C8FB_633A7580_6FFDD8C5_197FC478_7E561C73_43ACFC88_80F8E2F8_4ED01EFC
dt[255:0]     'h c0895E8▸  12153524_12153524_12153524_12153524_12153524_12153▸  c0895E81_c0895E81_c0895E81_c0895E81_c0895E81_c0895E81_c0895E81_c0895E81
tweak[127:0]  'h 0000000▸  00000000_00000000_00000000_11111111                   00000000_00000000_00000000_10101010
uk[255:0]     'h 0000000▸  00000000_00000000_00000000_00000000_00000000_00000▸  00000000_00000000_00000000_00000000_00000000_00000000_11101110_00100100
```

Fig. 6: Software simulation of the decryption algorithm (inverse threefish cipher)

- Total CPU time to Xst completion: 120.71 sec
- Total memory usage: 374344 kb
- Lighter version of skein-256 (decryption)
- Selected device: 7v×1140tflg1930-2
- Number of slice LUTs: 22369 out of 712000 (3%)
- Number used as logic: 22369 out of 712000 (3%)
- Number of fully used LUT-FF pairs: 0 out of 22369 (0%)
- Number of unique control sets: 0
- Number of Ios: 896
- Number of bonded IOBs: 896 out of 1100 (81%)
- Delay: 1284.911ns
- Total real time to Xst completion: 254 sec
- Total CPU time to Xst completion: 253.56 sec
- Total memory usage: 546248 kb

CONCLUSION

Lighter version of skein provides slightly higher processing speeds and computational abilities compared to its heavier counterpart, skein-256. The main difference between skein-256 and lighter version of skein was visible during the HDL simulations of the two modules in NC Simulator. Lighter version of skein has minimal or no delay in software simulation in computation of the ciphertext. For seventy-two rounds of the skein cryptosystem, the delay found in skein-256 was 1126.389 ns while the delay in Lighter version of skein was just 920.721 ns in the hardware. The Lighter version of skein model is almost 1.5 times faster as compared to skein-256. The most critical thing is designing the decryption part as the data that is fed as input for encryption and the data that is obtained after decryption should match. A comparison of throughput versus number of rounds was calculated to make a thorough comparison between lighter version of skein and skein-256. Table 3 shows the data accumulated for throughput of various rounds of the lighter version of skein cryptosystem. Table 4 shows the data accumulated for comparison of various specifications of lighter version of skein with skein-256. Hence, on comparing the memory usage, delay, throughput and speed of Lighter version of skein with skein-256 proves that this design is a lighter version of skein (less delay, more throughput and high speed).

Table 3: Comparison of throughput for encryption and decryption for various rounds of lighter version of skein

Rounds	Throughput for Encryption (mbps)	Throughput decryption (mbps)
1	31,800	13,996
8	4,670	3,606
18	2,150	1,818
36	1,380	960
72	695	498

Table 4: Comparison of various specifications of skein and lighter version of skein cryptosystem

Specifications	Skein-256	Lighter version of skein
Memory usage (MB)	487.568	460.21
CPU time to XST completion (sec)	210.25	186.855
Delay (Nano sec)	1126.389	920.721
Throughput (mbps)	568.1	600

ACKNOWLEDGEMENTS

The researcher would like to thank the entire semiconductor team at cyient for their immense support and motivation. Without their guidance this study would not have seen the light of day. A special thanks to Mr. Ram Gollapudi for allowing the researchers to use the companies valuable resources to complete the project.

REFERENCES

Aumasson, J.P., C. Calik, W. Meier, O. Ozen, R.C.W. Phan and K. Varici, 2009. Improved cryptanalysis of skein. Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security, December 6-10, 2009, Tokyo, Japan, pp: 542-559.

Bellare, M., K. Joe and R. Phillip, 1990. The Security of Cipher Block Chaining. In: Advances in Cryptology-Crypto 94, Desmedt, Y.G. (Eds.). Springer-Verlag, New York, pp: 341-358.

Burr, W.E., 2006. Cryptographic hash standards: Where do we go from here?. IEEE Secur. Privacy, 2: 88-91.

Ferguson, N., S. Lucks, B. Schneier, D. Whiting and M. Bellare *et al.*, 2010. The Skein hash function family. Submission NIST., 7: 1-86.

Kamal, A.H.M. and G. Hossain, 2004. A new approach of image encryption a part of cryptography. Asia J. Inform. Technol., 3: 607-610.

Tilich, S., 2009. Hardware implementation of the SHA-3 candidate Skein. Institute for Applied Information Processing and Communication, Graz University of Technology, Austria.

Wakerly, J.F., 2006. Digital Design Principles and Practices. Pearson Education, New Jersey, United States, Pages: 261.

Webster, D.M. and M. Lukowiak, 2011. Versatile FPGA architecture for skein hashing algorithm. Proceedings of the 2011 International Conference on Reconfigurable Computing and FPGAs (ReConFig), November 30-December 2, 2011, IEEE, Cancun, Mexico, ISBN: 978-1-4577-1734-5, pp: 268-273.