Update on SHAvite 3: Secure, Efficient, and Flexible Hash Function Proposal

Orr Dunkelman

Faculty of Mathematics and Computer Science
Weizmann Institute of Science

Joint work with Eli Biham

SHAvite 3

The Most Suitable

Candidate for SHA-3

SHAvite 3

Among the 13 Most Suitable

Candidates for SHA-3

Mistake in the (Reference) Implementation

All SHAvite-3 implementations use the following AES round implementation:

```
output[0] = Table0[input[0] >> 24] ^ Table1[(input[1] >> 16) & 0xff] ^ \
            Table2[(input[2] >> 8) & 0xff] ^ Table3[input[3] & 0xff];
output[1] = Table0[input[1] >> 24] ^ Table1[(input[2] >> 16) & 0xff] ^ \
            Table2[(input[3] >> 8) & 0xff] ^ Table3[input[0] & 0xff];
output[2] = Table0[input[2] >> 24] ^ Table1[(input[3] >> 16) & 0xff] ^ \
            Table2[(input[0] >> 8) & 0xff] ^ Table3[input[1] & 0xff];
output[3] = Table0[input[3] >> 24] ^ Table1[(input[0] >> 16) & 0xff] ^ \
            Table2[(input[1] >> 8) & 0xff] ^ Table3[input[2] & 0xff];
```

Mistake in the (Reference) Implementation (cont.)

The implementation should be:

```
output[0] = Table0[input[0] & 0xff ] ^ Table1[(input[1] >> 8) & 0xff] ^ \
            Table2[(input[2] >> 16) & 0xff] ^ Table3[(input[3] >>24)];
output[1] = Table0[input[1] & 0xff ] ^ Table1[(input[2] >> 8) & 0xff] ^ \
            Table2[(input[3] >> 16) & 0xff] ^ Table3[(input[0] >>24)];
output[2] = Table0[input[2] & 0xff ] ^ Table1[(input[3] >> 8) & 0xff] ^ \
            Table2[(input[0] >> 16) & 0xff] ^ Table3[(input[1] >>24)];
output[3] = Table0[input[3] & 0xff ] ^ Table1[(input[0] >> 8) & 0xff] ^ \
            Table2[(input[1] >> 16) & 0xff] ^ Table3[(input[2] >>24)];
```

We would like to thank **Olivier Billet** and **Thomas Pornin**.

Software Performance SHAvite 3_{256}

Hash Function	32 Bit
MD5	5.67
SHA-1	9.38
SHA-256	27.29
SHA-512	78.38
SHAvite-3_{256} (measured)	32.83
SHAvite-3_{256} (eBASH, Intel/AMD)	28.73–84.42
SHAvite-3_{256} (eBASH, PowerPC)	20.62–43.99
SHAvite-3_{256} (with AES inst. [B+09]*)	
SHAvite-3_{256} (with AES inst. [B10]†)	
SHAvite-3_{256} (with AES inst. [B+10])	
SHAvite-3_{256} (with AES inst. eBASH)	

Software Performance SHAvite 3_{512}

Hash Function	32 Bit
MD5	5.67
SHA-1	9.38
SHA-256	27.29
SHA-512	78.38
SHAvite-3_{512} (measured)	55.90
SHAvite-3_{512} (eBASH, Intel/AMD)	55.30–242.09
SHAvite-3_{512} (eBASH, PowerPC)	32.00–184.78
SHAvite-3_{512} (with AES inst. [B+09]*)	
SHAvite-3_{512} (with AES inst. [B10]†)	
SHAvite-3_{512} (with AES inst. [B+10])	
SHAvite-3_{512} (with AES inst. eBASH)	
SHAvite-3_{512} (with AES inst. our)	

FPGA Implementations [HRG10]

Platform	SHAvite 3_{256} Throughput (Mbps)	SHAvite 3_{256} Area (CLBs/ALUTs)	Area (CLBs/ALUTs)
Xilinx Spartan 3	1173.52	4114	838
Xilinx Virtex 4	2104.74	4114	838
Xilinx Virtex 5	2885.89	1130	433
Altera Cyclone II	1320.72	9400	1655
Altera Cyclone III	1420.85	9323	1653
Altera Stratix II	2353.39	2501	973
Altera Stratix III	3528.65	2497	963

FPGA Implementations [HRG10]

Platform	SHAvite 3_{512} Throughput (Mbps)	SHAvite 3_{512} Area (CLBs/ALUTs)	Area (CLBs/ALUTs)
Xilinx Spartan 3	1354.83	7997	1367
Xilinx Virtex 4	2901.30	8544	1403
Xilinx Virtex 5	3847.44	1951	646
Altera Cyclone II	1560.94	20441	2916
Altera Cyclone III	1859.57	20434	2915
Altera Stratix II	2533.38	5507	1639
Altera Stratix III	3858.84	5605	1620

FPGA Implementations [K+10]

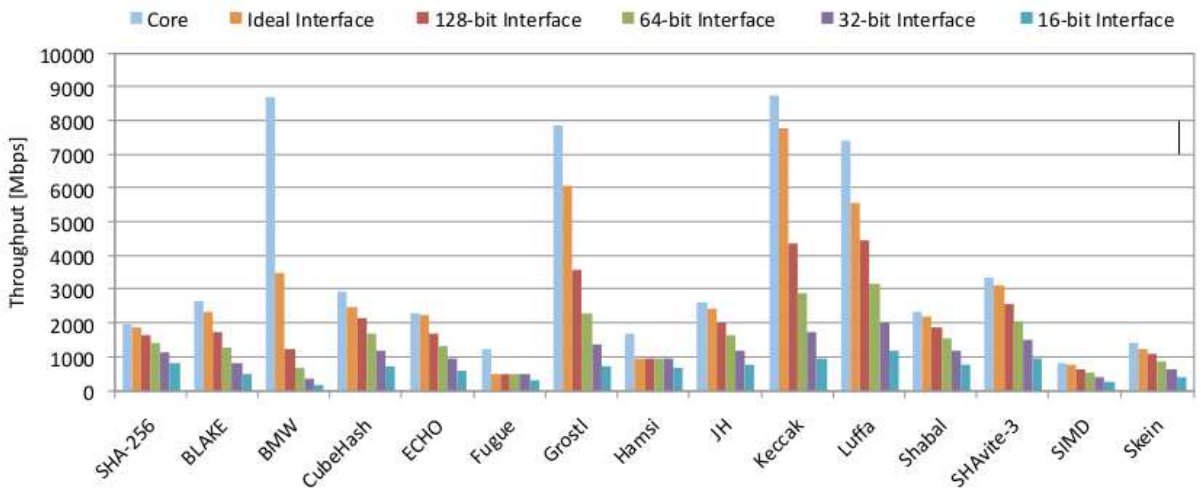

Fig. 5. Maximum Throughput for Various Types of Interface with $I_w = 3$. Target Platform: Virtex 5 (xc5vlx30-3ff324) FPGA Board.

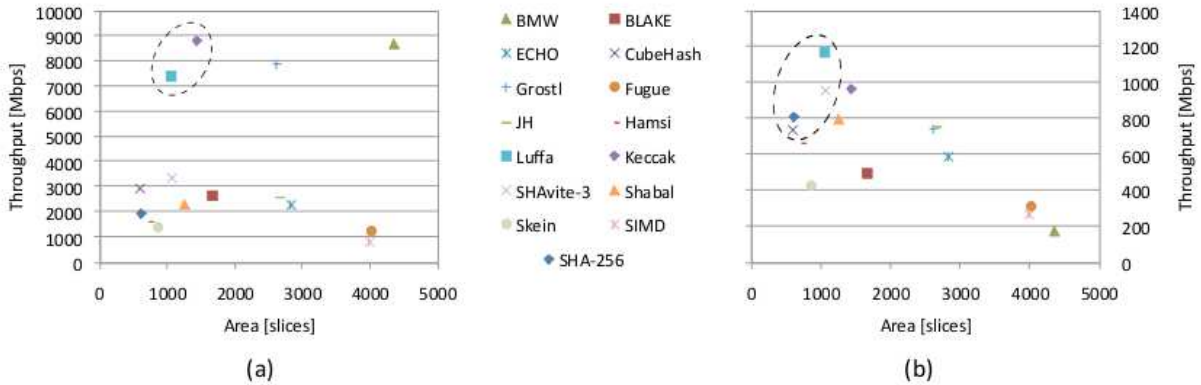

Fig. 6. Throughput versus Area graph: (a) Core Function only. (b) Fixed Interface with $w = 16$ bits and $I_w = 3$. Target Platform: Virtex 5 (xc5vlx30-3ff324) FPGA Board.

ASIC Implementations [K+10]

Fig. 13. Latency versus Area graph: (a) Core Function only. (b) Fixed Interface with $w = 16$ bits and $I_w = 3$. Target Platform: STM 90 nm CMOS Technology, Synthesis Results.

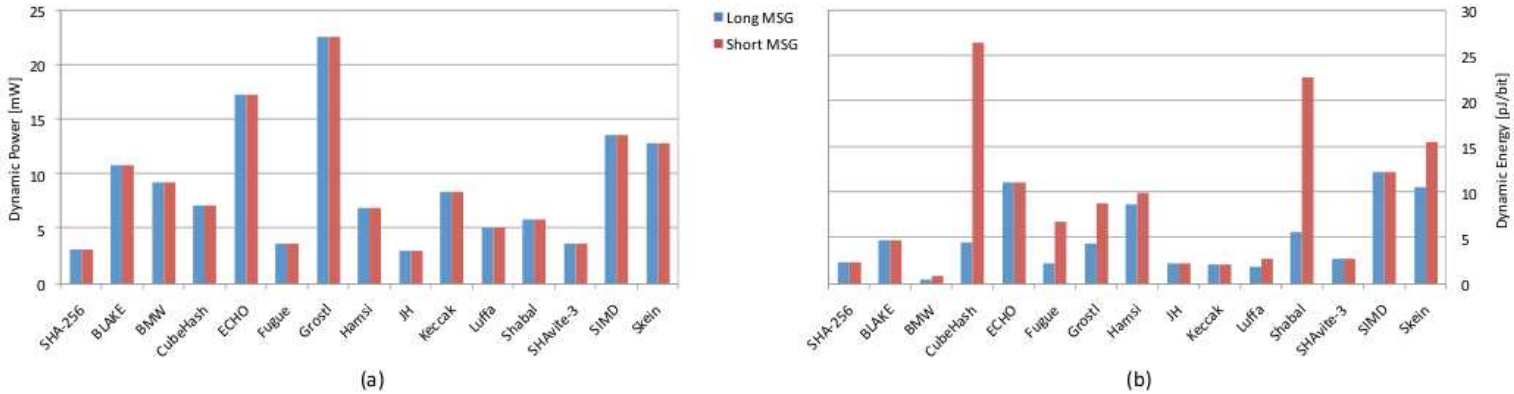

Fig. 14. (a) Dynamic Power Consumption. (b) Dynamic Energy Consumption. Target Platform: STM 90 nm CMOS Technology, Synthesis Results.

Hardware Implementations Two Comments

- Remember that when both AES & SHAvite-3 are needed, it is sufficient to implement the round function once, and re-use it twice.
- If no salt functionality is needed (or a fixed salt is used), the memory consumption is greatly reduced.

Attacks on SHAvite 3_{256} [M+00]

Rounds	Attack	Time	Memory	Adversary s control
6	free-start collision	2^{120}	2^{56}	chaining value, message
7	comp. func. distinguisher	2^{120}	2^{56}	chaining value, message
7	free-start near-collision	2^{25}	2^{14}	chaining value, message, salt
8	comp. func. distinguisher	2^{25}	2^{14}	chaining value, message, salt

Attacks on SHAvite 3_{512}

Rounds	Attack	Time	Memory	Adversary s control
9	Second Preimage (comp. func.) [B+10]	2^{384}	—	message block
	Second Preimage (hash. func.) [B+10]	2^{448}	2^{64}	message block
10	Second Preimage (comp. func.) [G+10]	2^{448}	—	message block
	Second Preimage (comp. func.) [G+10]	2^{416}	2^{64}	message block
	Second Preimage (comp. func.) [G+10]	2^{384}	2^{128}	message block
	Second Preimage (hash. func.) [G+10]	2^{480}	2^{32}	message block
	Second Preimage (hash. func.) [G+10]	2^{464}	2^{64}	message block
	Second Preimage (hash. func.) [G+10]	2^{448}	2^{128}	message block
14	Collision (comp. func.) [G+10]	2^{192}	2^{128}	message block, counter, salt, chaining value
	Preimage (comp. func.) [G+10]	2^{384}	2^{128}	message block, counter, salt, chaining value
	Preimage (comp. func.) [G+10]	2^{448}	—	message block, counter, salt, chaining value

Effects of the Full Compression Function Attacks

- One counter value is problematic.
- Making the compression non-optimal for that counter value.

Effects of the Full Compression Function Attacks

- One counter value is problematic.
- Making the compression non-optimal for that counter value.

Does this affect the hash function?

Is SHAvite 3_{512} Secure?

- Following [I10]:
  SHAvite-3_{512} is collision-resistant if C_{512} is two-step collision resistant.

Is SHAvite 3_{512} Secure?

- Following [I10]:
  SHAvite-3_{512} is collision-resistant if C_{512} is two-step collision resistant.
- Following [BFL10]:
  The indifferentiability of the hash function is

$$16 \cdot \frac{q^2}{2^n} + 4|\mathcal{W}| \cdot \frac{q}{2^n}$$

  where $|\mathcal{W}|$ is the number of weak states, which for SHAvite-3_{512} translates to $16q^2/2^{512} + 4q/2^{384}$ for q queries.

Is SHAvite 3_{512} Secure?

- Following [I10]:
  SHAvite-3_{512} is collision-resistant if C_{512} is two-step collision resistant.
- Following [BFL10]:
  The indifferentiability of the hash function is

$$16 \cdot \frac{q^2}{2^n} + 4|\mathcal{W}| \cdot \frac{q}{2^n}$$

  where $|\mathcal{W}|$ is the number of weak states, which for SHAvite-3_{512} translates to $16q^2/2^{512} + 4q/2^{384}$ for q queries.

Conclusion: SHAvite-3_{512} is a secure hash function

Tweaking SHAvite 3_{512}

- SHAvite-3_{512} is secure.

Tweaking SHAvite 3_{512}

- SHAvite-3_{512} is secure, but the security margins may be conceived as insufficient.
- Hence, we propose a tweak to SHAvite-3_{512}:
  1. Increasing the number of rounds from 14 to 16.
  2. Mixing the counter in each nonlinear update step.
  3. In the four new locations — use the negated value from the previous nonlinear update step.

Effects of the Tweak:

- Reduce adversarial control of the internal state.
- Impossible to fix the expanded message to a specific value in more than four consecutive rounds.
- Larger security margins.

Example: Setting Expanded Message Words to 0

SHAvite-3_{512} [G+10]:

Round i	$RK_{0,i}$				$RK_{1,i}$			
	$k^0_{0,i}$	$k^1_{0,i}$	$k^2_{0,i}$	$k^3_{0,i}$	$k^0_{1,i}$	$k^1_{1,i}$	$k^2_{1,i}$	$k^3_{1,i}$
0	?	?	?	?	?	?	?	?
1	?*	?	?	?	?	?	?	0
2	0	?	?	?	?	0	0	0
3	0	?	?	?	0	0	0	0
4	0	?	0	0	0	0	0	0
5	0	0*	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0*
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	?*	?

Tweaked SHAvite-3_{512}:

Round i	$RK_{0,i}$				$RK_{1,i}$			
	$k^0_{0,i}$	$k^1_{0,i}$	$k^2_{0,i}$	$k^3_{0,i}$	$k^0_{1,i}$	$k^1_{1,i}$	$k^2_{1,i}$	$k^3_{1,i}$
0	?	?	?	?	?	?	?	?
1	?†*	?	?	?	?	?	?†	0
2	0	?	?	?	?	0	0	0
3	0	?	?†	?†	0*	0	0	0
4	0	?	0	0	0	0	0	0
5	0	0*	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0*	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	?†	0*
10	?†	?†	?†	0	?†	0	?†	?†
11	?	?	?	?*	?	?	?	?
12	?	?	?	?	?	?	?	?
13	?	?	?	?	?	?	?*	?
14	?	?	?	?	?	?	?	?
15	?	?	?*	?	?	?	?	?

Reminder: SHAvite 3 MAC

- As SHAvite-3 is secure, we can define

$$\text{SHAvite-3-MAC}_k(M) = \text{SHAvite-3}_k(M).$$

- Of course, the user needs to keep the key secret!
- No need for HMAC or any other constructions!

Thank you for your Attention!

SHAvite-3:
The Most Suitable
Candidate for SHA-3

Questions?

http://www.cs.technion.ac.il/~orrd/SHAvite-3