Volume 6, Issue 10, October 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A J-Lanes Tree Hashing Mode and J-Lanes SHA-512

[1]Nandu Nandini, [2]Janjanam Madhu Babu
[1] (M.Tech) –CSE, Vasireddy Venkatadri Institute of Technology (VVIT), Namburu, Guntur, Andhra Pradesh, India
[2] Assistant Professor, Dept of CSE, Vasireddy Venkatadri Institute of Technology (VVIT), Namburu, Guntur, Andhra Pradesh, India

Abstract: ***In this paper we describe a method for efficiently hashing multiple messages of different lengths. Such computations occur in various scenarios, and one of them is when an operating system checks the integrity of its components during boot time. These tasks can gain performance by parallelizing the computations and using SIMD architectures. For such scenarios, we compare the performance of a new 4-buffers SHA-512 S-HASH implementation among 3rd Generation Intel® Core™ and 5th generation Intel core i5 Processors, 32-bit platforms vs 64-bit platforms can make a significant difference, as well as the amount of data you're hashing. 64-bit machine, SHA-512 bits SHA-256 for hashing anything more than 16 bytes of data at a time. And generally, the more data being hashed at once, the bigger the performance improvement. SHA-512 has 25% more rounds than SHA-256. On a 64-bit processor each round takes the same amount of operations, yet can process double the data per round, because the instructions process 64-bit words instead of 32-bit words. Therefore, 2 / 1.25 = 1.6, which is how much faster SHA-512 can be under optimal conditions.***

Keywords: j-lanes hashing, SHA-256, SHA-512

I. INTRODUCTION

Most cryptographic hash functions operate on a fixed-size internal state and process the message to be hashed block by block, where a block is a number of bits ,The performance of hash functions is important in various situations and platforms. One example is a server workload: authenticated encryption in SSL/TLS sessions, where hash functions which provide message authentication, in HMAC mode. This is one reason why the performance of SHA-256 on modern x86_64 architectures was defined as a baseline for the SHA3 competition. When an operating system checks the integrity of its components during boot time. These tasks can gain performance by parallelizing the computations and using SIMD architectures.

Single instruction, multiple data (SIMD), is a class of parallel computers in Flynn's taxonomy. It describes computers with elements that perform the same operation on multiple data points simultaneously. Thus, such machines exploit data level parallelism, but not concurrency: there are simultaneous (parallel) computations, but only a single process (instruction) at a given moment. SIMD is particularly applicable to common tasks like adjusting the contrast in a digital image or adjusting the volume of digital audio. Most modern CPU designs include SIMD instructions in order to improve the performance of multimedia use. SHA- 256 and SHA-512 are most widely used and popular hash functions. Therefore, for comparison with Cayley hash function, we only consider these two. SHA-256 processes blocks of size 512 bits in every cycle of compression function, whereas SHA-512 takes in blocks of size 1024 bits. Both the hash functions pad input message if the size of message is smaller than required block size. SHA-256 produces digest of size 256 bits, whereas SHA-512 produces output digest of size 512 bits. Word sizes used by SHA-256 and SHA-512 are 32 and 64 respectively.

II. PRELIMINARIES

J-lanes tree hashing:

The j-lanes tree hashing is a tree mode that splits an input message to j slices, computes j independent digests of each slice, and outputs the hash value of their concatenation

SHA-256:

The detailed definition of SHA-256 can be found in FIPS180-2 publication [9]. Schematically, the computational flow of SHA-256 can be viewed as follows: “Init” (setting the initial values), a sequence of “Update” steps (compressing a 64 bytes block of the message, and updating the digest value), and a “Finalize” step (takes care of the message padding). The padding requires either one or two calls to the Update function, depending on the message’s length (see more details in [2]). For SHA-256, the performance is almost linearly proportional to the number (N) of Update function calls, which. For a message of length bytes, the value of N is:

© 2016, IJARCSSE All Rights Reserved

$$N = \begin{cases} \left\lfloor \frac{\text{length}}{64} \right\rfloor + 2 & \text{length mod } 64 \geq 56 \\ \left\lfloor \frac{\text{length}}{64} \right\rfloor + 1 & \text{else} \end{cases}$$

For sufficiently long messages, we can approximate N ~ floor (length/64). For example, this approximation for a 4 KB message gives floor (length/64) = 64, while actual hashing of a 4 KB message requires 65 Update function calls (i.e., a ~ 1.5% deviation).

SHA-512

Guaranteeing message and code integrity is very important for the security of applications, operating systems and the network infrastructure of the future Internet. Protection against intentional alteration of data may be supported using one way hash functions. A one way hash function is a mathematical construct that accepts as input a message of some length and returns a digest of much smaller length. One way hash functions are designed in such a way that it is computationally infeasible to find the input message by knowing only the digest. Federal Information Processing Standard (FIPS) 180-2, published by the National Institute of Standards and Technology (NIST), defines a variety of one way hash functions: SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512, which are widely used in a variety of secure internet protocols, for example, TLS, SSL, PGP, SSH, S/MIME and IPsec protocols.

III. SYSTEM STUDY

Properties of Hash Function

- These hash functions can be applied to any size data producing a fixed-length output.
- Hash functions H(x) are relatively easy to compute for any given message x
- Follows one-way property where it is computationally infeasible to find x such that H(x) = h
- Have weak collision resistance where it is computationally infeasible to find $y \neq x$ such that H(y) = H(x)
- Hash functions are strong collision resistance computationally infeasible to find any pair (x, y) such that H(x) = H(y)

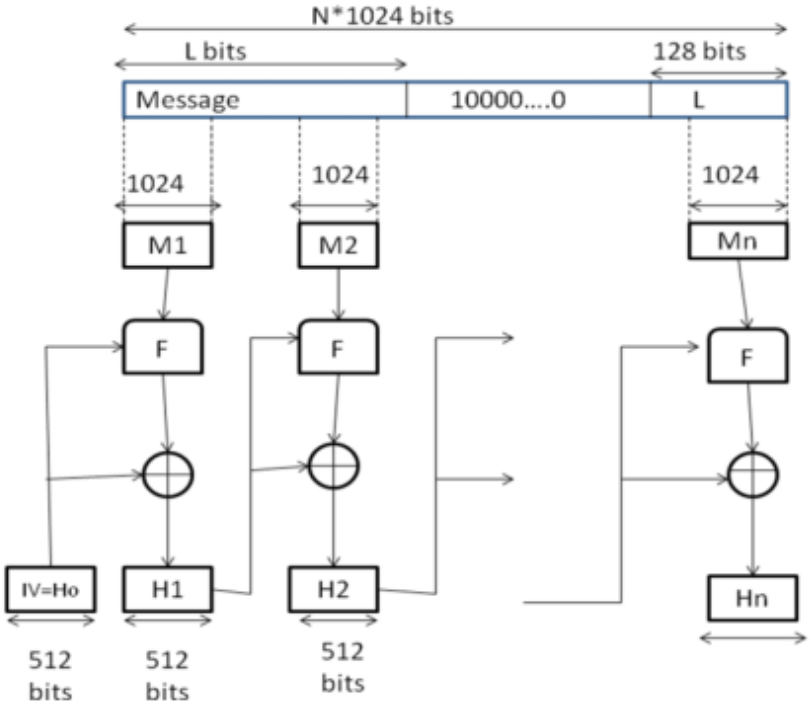

Fig 1. SHA-512 BASIC structure

SHA-512 is identical in structure to SHA-256, but:

- the message is broken into 1024-bit chunks,
- the initial hash values and round constants are extended to 64 bits,
- there are 80 rounds instead of 64,
- the message schedule array w has 80 64-bit words instead of 64 32-bit words,
- to extend the message schedule array w, the loop is from 16 to 79 instead of from 16 to 63,
- the round constants are based on the first 80 primes 2..409,
- the word size used for calculations is 64 bits long,
- the appended length of the message (before pre-processing), in *bits*, is a 128-bit big-endian integer, and
- The shift and rotate amounts used are different.

© 2016, IJARCSSE All Rights Reserved

S-HASH(Simultaneous hashing) of multiple messages

SIMD architectures [7] are designed to execute, in parallel, the same operations on several independent chunks of data (called "elements"). Modern architectures have variants of SIMD instructions that operate on elements of sizes 16,64,256,1024 and 8192 bytes. By the nature of the algorithms, SHA-256 (and SHA-1) requires operations on 4 bytes elements, while SHA-512 requires operations on 8 bytes elements. Fig. 1 describes the Simultaneous Hashing algorithm (S-HASH) that hashes k messages and generates k digests, with some hash function. Suppose that the implemented hash function operates on t-bit "words" (elements), and that the architecture has s-bit SIMD registers. Then, the number of words that fit into a SIMD register is m = s/t, which we assume to be an integer. We also assume that k > m.

SHA 512 takes as input a message of any arbitrary length, then processes this input into fixed length blocks of 1024 bits and finally produces a hash value of 512 bits.

Step I: Padding: The message is appended with padding with a string of 1 followed by 0's.Padding is done to make the variable length message a multiple of 1024 bits.

Step II: Append Length: A block of 128 bits is appended to the message after padding field. This block consists of actual length of message before padding is done.

Step III: Initialize Buffer: Eight 512 bit buffer is designed to hold intermediate and final result of hash function. Each buffer is represented as 64 bit register (a, b, c, d, e, f, g, h) a=6A09E667F3BCC908 b=BB67AE8584CAA73B c=3C6EF372FE94F82 d=A54FF53A5F1D36F1 e=510E527FADE682D1 f=9B05688C2B3E6C1F g=1F83D9ABFB41BD6B h=5BE0CD19137E2179

Step IV: Process message in 1024 bit blocks: The message is processed in blocks of 1024 bits. Each block goes through a compression function reducing it into 512 bits. This compression function consists of a total 80 rounds increasing the complexity of hash function produced.

Step V: Output: After all N 1024 bit blocks have been processed the output obtained from nth stage is the message digests.

Example

```
$ uname -m
x86_64

$ openssl speed sha256 sha512
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
sha256           29685.74k    79537.52k   148376.58k   186700.77k   196588.36k
sha512           23606.96k    96415.90k   173050.74k   253669.59k   291315.50
```

SHA-512 has 25% more rounds than SHA-256. On a 64-bit processor each round takes the same amount of operations, yet can process double the data per round, because the instructions process 64-bit words instead of 32-bit words. Therefore, 2 / 1.25 = 1.6, which is how much faster SHA-512 can be under optimal conditions.

Of course there is memory overhead, instruction latency, and other factors involved; on an Intel Ivy Bridge processor long message SHA-512 is 1.54 x faster, and on an AMD Piledriver it is 1.48 x faster.

For small messages (less than 448 bits) SHA-512 will be approx 1.25 x slower, because only a single hash iteration is performed. There are also various crossover points where one hash will need to process an extra iteration and the other will not, but these numbers are averages, the actual performance graph will be stepped at the iteration increment point.

Note: SHA-512 (and SHA-384) is usually faster on 64-bit platforms, and SHA-256 is usually faster on 32-bit platforms.

IV. RESULTS

The performance numbers labeled 'x86' were running using 32-bit code on 64-bit processors, whereas the 'x86-64' numbers are native 64-bit code. While SHA-256 is designed for 32-bit calculations, it does benefit from code optimized for 64-bit processors on the x86 architecture. 32-bit implementations of SHA-512 are significantly slower than their 64-bit counterparts. Variants of both algorithms with different output sizes will perform similarly, since the message expansion and compression functions are identical, and only the initial hash values and output sizes are different.

CPU architecture	Frequency	Algorithm	Word size (bits)	Cycles/byte x86	MiB/s x86	Cycles/byte x86-64	MiB/s x86-64
Intel Ivy Bridge	3.5 GHz	SHA-256	32-bit	16.80	199	13.05	256
		SHA-512	64-bit	43.66	76	8.48	394
AMD Piledriver	3.8 GHz	SHA-256	32-bit	22.87	158	18.47	196
		SHA-512	64-bit	88.36	41	12.43	292

© 2016, IJARCSSE All Rights Reserved

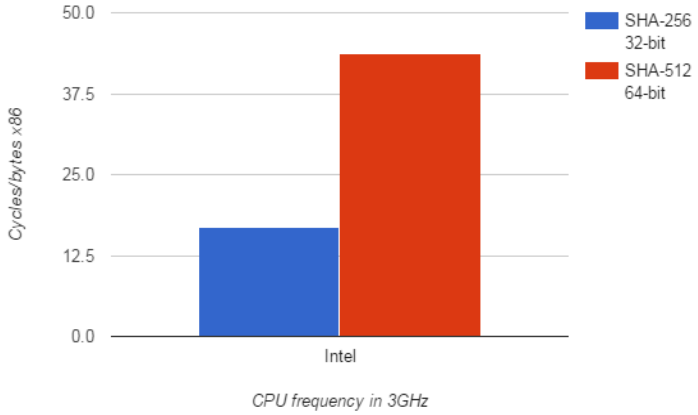

Fig 2. CPU Frequency in 3 GHz vs Cycles/bytes x86

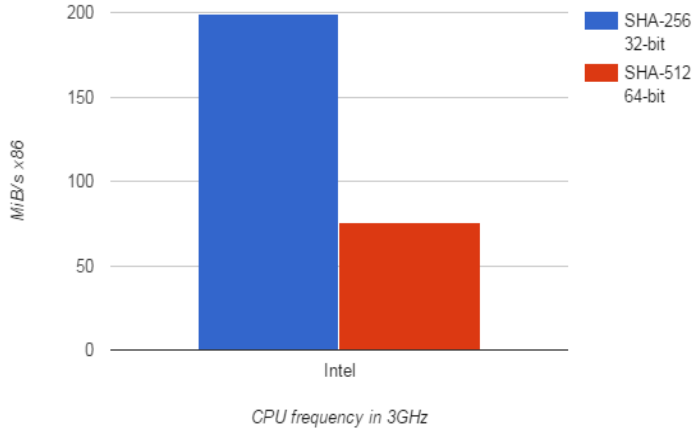

Fig 3. CPU Frequency in 3 GHz vs MiB/s x86

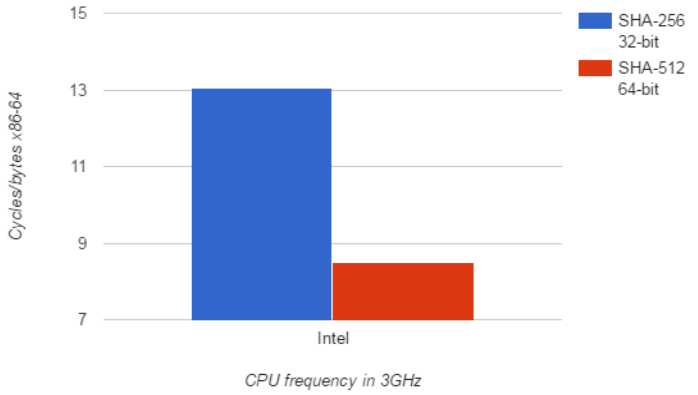

Fig 4. CPU Frequency in 3 GHz vs Cycles/bytes x86-64

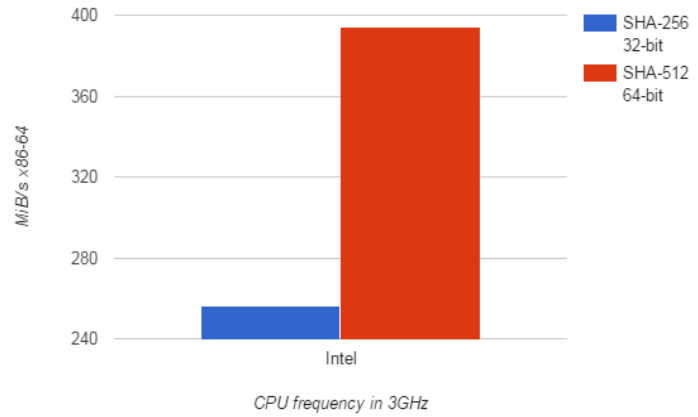

Fig 5. CPU Frequency in 3 GHz vs MiB/s x86-64

© 2016, IJARCSSE All Rights Reserved

Description:

We compare the performance of a new 4-buffers SHA-512 S-HASH implementation among 3rd Generation Intel® Core™ and 5th generation intel core i5 Processors, 32-bit platforms vs 64-bit platforms under Intel lvy Architecture can make a significant difference, as well as the amount of data you're hashing. 64-bit machine, SHA-512 bits SHA-256 for hashing anything more than 16 bytes of data at a time. And generally, the more data being hashed at once, the bigger the performance improvement. SHA-512 has 25% more rounds than SHA-256. On a 64-bit processor each round takes the same amount of operations, yet can process double the data per round, because the instructions process 64-bit words instead of 32-bit words. Therefore, 2 / 1.25 = 1.6, which is how much faster SHA-512 can be under optimal conditions.

V. CONCLUSION

We illustrated a method for efficiently hashing multiple messages of different lengths. These tasks can gain performance by parallelizing the computations and using Intel lvy Architecture. In this connection we compare the performance of 32-bit platforms vs 64-bit platforms can make a significant difference, as well as the amount of data you're hashing. 64-bit machine, SHA-512 bits SHA-256 for hashing anything more than 16 bytes of data at a time performance improvement is 1.6% than SHA-256

REFERENCES

[1] NIST, "Cryptographic Hash Algorithm Competition," 2012. http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

[2] S. Gueron and V. Krasnov, "Parallelizing Message Schedules to Accelerate the Computations of Hash Functions," 2012. http://eprint.iacr.org/2012/067.pdf

[3] Federal Information Processing Standards Publication 180-2: Secure Hash Standard. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[4] Gaudet, D.: SHA1 using SIMD techniques, http://arctic.org/~dean/crypto/sha1.html.

[5] Gueron, S., Krasnov, V.: Parallelizing message schedules to accelerate the computations of hash functions (2012), http://eprint.iacr.org/2012/067.pdf

[6] Dodis, Y., Reyzin, L., Rivest, R.L., Shen, E.: Indifferentiability of Permutation-Based Compression Functions and Tree-Based Modes of Operation, with Applications to MD6. Proceedings of FSE 2009, Lecture Notes in Computer Science, 5665 104-121 (2009).

[7] Merkle, R. C.: A certified digital signature. Advances in Cryptology, Proceedings of CRYPTO '89, Lecture Notes in Computer Science, 435: 218-238 (1990).

[8] P. Sarkar, P. Schellenberg, P. J.: A parallelizable design principle for cryptographic hash functions. Cryptology ePrint Archive (2002), http://eprint.iacr.org/2002/031

[9] "Federal Information Processing Standards Publication 180-2: Secure Hash Standard." http://csrc.nist.gov/publications/fips/fips180-2/fips180-2. pdf

© 2016, IJARCSSE All Rights Reserved