Network Working Group
Request for Comments: 1810
Category: Informational

J. Touch
ISI

June 1995

Report on MD5 Performance

Status of this Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Abstract

MD5 is an authentication algorithm, which has been proposed as the default authentication option in IPv6. When enabled, the MD5 algorithm operates over the entire data packet, including header. This RFC addresses how fast MD5 can be implemented in software and hardware, and whether it supports currently available IP bandwidth. MD5 can be implemented in existing hardware technology at 256 Mbps, and in software at 87 Mbps. These rates cannot support current IP rates, e.g., 100 Mbps TCP and 130 Mbps UDP over ATM. If MD5 cannot support existing network bandwidth using existing technology, it will not scale as network speeds increase in the future. This RFC is intended to alert the IP community about the performance limitations of MD5, and to suggest that alternatives be considered for use in high speed IP implementations.

Introduction

MD5 is an authentication algorithm, which has been proposed as one authentication option in IPv6 [1]. RFC 1321 describes the MD5 algorithm and gives a reference implementation [3]. When enabled, the MD5 algorithm operates over the entire data packet, including header (with dummy values for volatile fields). This RFC addresses how fast MD5 can be implemented in software and hardware, and whether it supports currently available IP bandwidth.

This RFC considers the general issue of checksumming and security at high speed in IPv6. IPv6 has no header checksum (which IPv4 has [5]), but proposes an authentication digest over the entire body of the packet (including header where volatile fields are zeroed) [1]. This RFC specifically addresses the performance of that authentication mechanism.

Measurements

The performance of MD5 was measured. The code was an optimized version of the MD5 reference implementation from the RFC [3], and is available for anonymous FTP [7]. The following are the results of the performance test "md5 -t", modified to prohibit on-chip caching of the data block:

87 Mbps	DEC Alpha (190 Mhz)
33 Mbps	HP 9000/720
48 Mbps	IBM RS/6000 7006 (PPC 601 @80 Mhz)
31 Mbps	Intel i486/66 NetBSD
44 Mbps	Intel Pentium/90 NeXTStep
52 Mbps	SGI/IP-20 IRIX 5.2
37 Mbps	Sun SPARC-10/51, SPARC-20/50 SunOS 4.1.3
57 Mbps	Sun SPARC-20/71 SunOS 4.1.3

These rates do not keep up with currently available IP bandwidth, e.g., 100 Mbps TCP and 130 Mbps UDP over a Fore SBA-200 ATM host interface in a Sun SPARC-20/71.

Values as high as 100 Mbps have been reported for the DEC Alpha (190 Mhz). These values reflect on-chip caching of the data. It is not clear at this time whether in-memory, off-chip cache, or on-chip cache performance measures are more relevant to IP performance.

Analysis of the MD5 Algorithm

The MD5 algorithm is a block-chained hashing algorithm. The first block is hashed with an initial seed, resulting in a hash. The hash is summed with the seed, and that result becomes the seed for the next block. When the last block is computed, it’s "next-seed’ value becomes the hash for the entire stream. Thus, the seed for block depends on both the hash and the seed of its preceding block. As a result, blocks cannot be hashed in parallel.

Each 16-word (64-byte) block is hashed via 64 basic steps, using a 4-word intermediate hash, and collapsing the intermediate hash at the end. The 64 steps are 16 groups of 4 steps, one step per intermediate hash word. This RFC uses the following notation (as from RFC-1321 [3]):

A,B,C,D	intermediate hash words
X[i]	input data block
T[i]	sine table lookup
<< i	rotate i bits
F	logical functions of 3 args

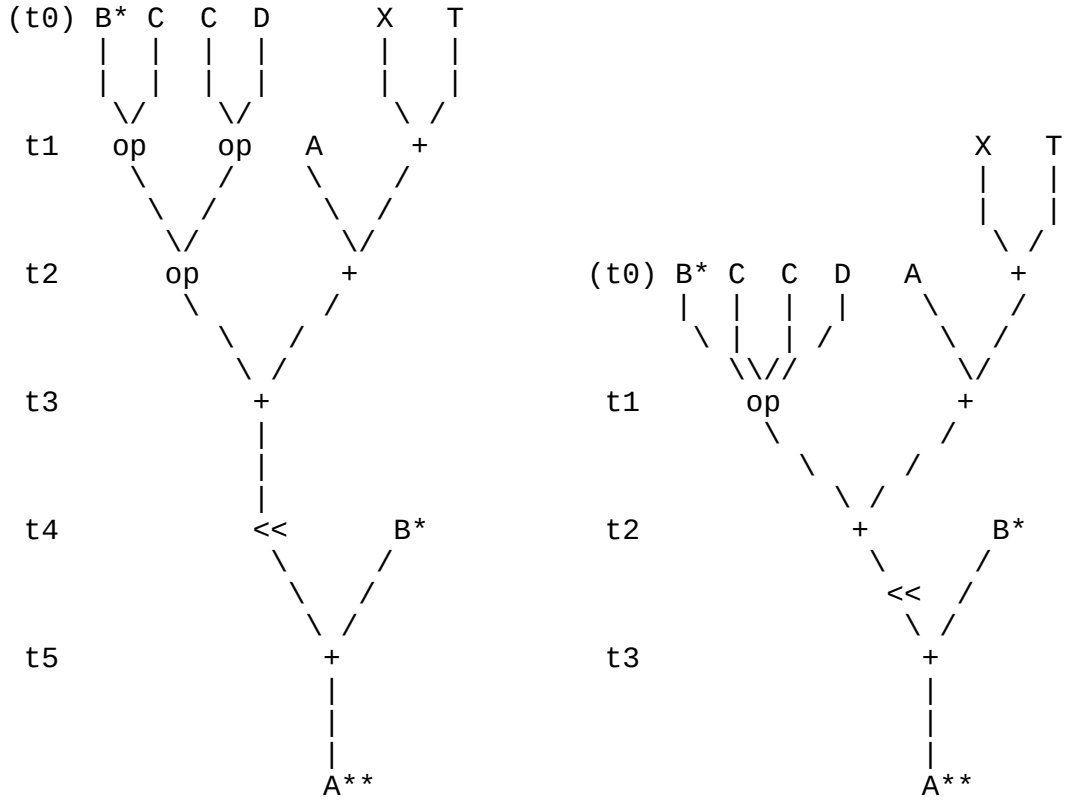
The subscripts to X, I, and << are fixed for each step, and are omitted here. There are four different logical functions, also omitted. Each 4-step group looks like:

```
A = B + ((A + F(B,C,D) + X[i] + T[i]) << i)
D = A + ((D + F(A,B,C) + X[i] + T[i]) << i)
C = D + ((C + F(D,A,B) + X[i] + T[i]) << i)
B = C + ((B + F(C,D,A) + X[i] + T[i]) << i)
```

Note that this has the general form shown below. Due to the complexity of the function 'f', these equations cannot be transformed into a less serial set.

```
A = f(D); B = f(A); C = f(B); D = f(C)
```

Each steps is composed of two table lookups, one rotation, a 3-component logical operation, and 4 additions. The best parallelization possible leaves F(x,y,z) to the last step, waiting as long as possible for the result from the previous step. The resulting tree is shown below.



Binary operation tree Optimized hardware tree

This diagram assumes that each operation takes one unit time. The tree shows the items that depend on the previous step as B*, and the item that the next step depends on as A**. Sequences of the binary operation tree cannot be overlapped, but the optimized hardware tree can (by one time step).

There are 4 steps processed per word of input, ignoring inter-block processing. The speed of the overall algorithm depends on how fast we can process these 4 steps, vs. the bandwidth of one word of input being processed.

The binary tree takes 5 time units per step of the algorithm, and permits at best 3-way parallelism (at time t1). In software, this means it takes 5 * 4 = 20 instructions per word input. A computer capable of M MIPS can support a data bandwidth of M/20 * 32 Mbps, i.e., bits per second equal to 1.6x its MIPS rate. Therefore, a 100 MIPS machine can support a 160 Mbps stream.

Parallel software rate in Mbps = 1.6 * MIPS rate

This assumes that register reads and writes are overlapped with computation entirely. Without any parallelism, there are 8 operations per step, and 4 steps per word, so 32 operations per word, i.e., the data rate in Mbps would be identical to the MIPS rate:

Serial software rate in Mbps = MIPS rate

Predictions using SpecInt92 numbers as MIPS estimators can be compared with measured rates [2]:

Spec-Int92	Predicted Upper-Bound	MD5 Measured	Machine
122	122-195	87 Mbps	DEC Alpha (190 Mhz)
48	48- 77	33 Mbps	HP 9000/720
88	88-141	48 Mbps	IBM RS/6000 7006 (PPC 601 @80 Mhz)
32	32- 51	31 Mbps	Intel i486/33 NetBSD
90	90-144	44 Mbps	Intel Pentium/90 NeXTStep
90	90-144	52 Mbps	SGI/IP-20 IRIX 5.2
65	65-104	37 Mbps	Sun SPARC-10/51 SunOS 4.1.3
126	126-202	57 Mbps	Sun SPARC-20/71 SunOS 4.1.3

The hardware rate takes 3 time units per step, i.e. 3 * 4 = 12 time units per word of input. Hardware capable of doing an operation (e.g., 32-bit addition) in N nanoseconds can support a data bandwidth of 32/12/N bps, i.e., 2/3N bps.

Hardware rate in Mbps = 8/3N * 1,000

For CMOS, an operation (32-bit addition, including register retrieval and storage) costs about 5.2 ns (2.6 ns per add, 2 ns for latching) [6]. There are 6 clocks through the most highly-parallelized implementation, resulting in 31.2 ns per 32-bit word, or 256 Mbps [6]. This will not keep pace with existing hardware, which is capable of link speeds in excess of 622 Mbps (ATM).

By comparison, IPv4 uses the Internet Checksum [5]. This checksum can be performed in 32-bit-wide units in excess of 1 Gbps in an existing, low-cost PLD. The checksum can also be parallelized by computing partial sums and reducing the result.

One Proposed Solution

There are several ways to increase the performance of the IPv6 authentication mechanism. One is to increase the hardware performance of MD5 by slightly modifying the algorithm, the other is to propose a replacement algorithm. This RFC discusses briefly the modification of MD5 for high-speed hardware implementation. Alternate algorithms, capable of 3.5x the speed of MD5, have been discussed elsewhere [6].

MD5 uses block chaining to ensure sensitivity to block order. Block chaining also prevents arbitrary parallelism, which can be as much a benefit to the spoofer as to the user. MD5 can be slightly altered to accommodate a higher bandwidth data rate. There should be a predetermined finite number of blocks processed from independent seeds, such that the I-th block is part of the "I mod K"-th chain. The resulting K digests themselves form a message, which can be MD5-encoded using a single-block algorithm. This idea was proposed independently by the author and by Burt Kaliski of RSA.

The goal is to support finite parallelism to provide adequate bandwidth at current processing rates, without providing arbitrary power for spoofing. It would require further analysis to ensure that it provides an adequate level of security.

For current technology and network bandwidth, a minimum of 4-way parallel chaining would suffice, and 16-way chaining would be preferable. This would support network bandwidth of 1 Gbps with 4-way chaining, in CMOS hardware. The chaining parallelism should be a multiple of 4-way, to generate a complete block of digests (4 words per digest, 16 words per block). This modification is believed to achieve the goals of MD5, without the penalties of implementation of the current MD5 algorithm.

Security Considerations

This entire document addresses a mechanism for providing security in IPv6. MD5 is the proposed default optional authentication mechanism for IPv6 traffic. This RFC specifically addresses the concern that security mechanisms such as MD5 that cannot support high bandwidth with available hardware will compromise their deployment, and ultimately, the security of the systems they are intended to maintain.

The IPv6 requirements document emphasizes that IPv6 implementations should not compromise performance, compared to IPv4. This is presumably despite IPv6's increased functionality. "Required optional" components of IPv6 should be held to this same standard. MD5 compromises performance, and so its use as a required default option in IPv6 should be reconsidered.

The use of MD5 as the default to the required authentication option may compromise security in high-bandwidth systems, because enabling the option causes performance degradation, defeating its inclusion as an IPv6 option. As a result, the authentication option may be disabled entirely.

It is important to the use of authentication in high-performance systems that an alternative mechanism be available in IPv6 from the outset. This may require the specification of multiple "required" authentication algorithms - one that's slower but believed strong, and one that's faster but may inspire somewhat less confidence.

Conclusions

MD5 cannot be implemented in existing technology at rates in excess of 256 Mbps in hardware, or 86 Mbps in software. MD5 is a proposed authentication option in IPv6, a protocol that should support existing networking technology, which is capable of 130 Mbps UDP.

As a result, MD5 cannot be used to support IP authentication in existing networks at existing rates. Although MD5 will support higher bandwidth in the future due to technological advances, these will be offset by similar advances in networking. If MD5 cannot support existing network bandwidth using existing technology, it will not be able to scale as network speeds increase in the future. This RFC proposes that MD5 be modified to support a 16-way block chaining, in order to allow existing technology (CMOS hardware) to support existing networking rates (1 Gbps). It further proposes that alternatives to MD5 be considered for use in high-speed networks.

Acknowledgements

The author would like to thank Steve Kent at BBN, Burt Kaliski, Victor Chang, and Steve Burnett at RSA, Ran Atkinson at the NRL, and the HPCC Division at ISI for reviewing the contents of this document. Burt independently suggested the block-parallel modification proposed here.

References

[1] Atkinson, R., "IPv6 Authentication Header", Work in Progress, Naval Research Lab, February 1995.

[2] DiMarco, J., "Spec Benchmark table, V. 4.12", <ftp://ftp.cfd.toronto.edu/pub/spectable>.

[3] Rivest, R., "The MD5 Message-Digest Algorithm", RFC1321, MIT LCS & RSA Data Security, Inc., April 1992.

[4] Partridge, C., and F. Kastenholz, "Technical Criteria for Choosing IP The Next Generation (IPng)", RFC 1726, BBN Systems and Technologies, FTP Software, December 1994.

[5] Postel, J., "Internet Protocol - DARPA Internet Program Protocol Specification," STD 5, RFC 791, USC/Information Sciences Institute, September 1981.

[6] Touch, J., "Performance Analysis fo MD5," to appear in ACM Sigcomm ’95, Boston.

[7] Touch, J., Optimized MD5 software, <ftp://ftp.isi.edu/pub/hpcc-papers/touch/md5-opt.tar>.

Author’s Address

Joe Touch
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292-6695
USA

Phone: +1 310-822-1511 x151
Fax: +1 310-823-6714
URL: ftp://ftp.isi.edu/pub/hpcc-papers/touch
EMail: touch@isi.edu