Efficient Hash Collision Search Strategies on Special-Purpose Hardware

Tim Güneysu, Christof Paar, Sven Schäge

Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
`{gueneysu,cpaar}@crypto.rub.de`, `sven.schaege@nds.rub.de`

Abstract

Hash functions play an important role in various cryptographic applications. Modern cryptography relies on a few but supposedly well analyzed hash functions, most of which are part of the so called MD4-family. This work shows whether it is possible, using special-purpose hardware, to significantly speedup collision search for MD4-family hash functions. A thorough analysis of the computational requirements of MD4-family hash functions and their corresponding collision attacks reveals that a microprocessor based architecture is best suited for the implementation of collision search algorithms. Consequently, we designed and implemented a general-purpose microprocessor for MD4-family hash-functions with minimal area requirements and, based on this, a full collision search integrated circuit. Comparing the performance characteristics of both ASICs with standard PC processors and networks, it turns out that our design, massively parallelized, is nearly four times more cost-efficient than parallelized standard PCs. We believe that there is much room for further improvements left.

Keywords. Hash functions, special-purpose hardware, crypto attacks.

1 Introduction

In February 2005 Wang et al. presented a new attack method against the popular Secure Hash Algorithm (SHA-1). It reduces the computational attack complexity to find a collision from $O(2^{80})$ to approximately $O(2^{69})$ [10] leading to the announcement that SHA-1 has been broken in theory. Soon it was further improved to $O(2^{63})$ [9]. However, still this attack is supposed to be theoretic in nature, because the necessary number of computations is very high.
For practical attacks, all theoretical results have to be mapped to an executable algorithm, which subsequently has to be launched on an appropriate architecture.
Basically, there are two ways to design such architectures, namely standard and special-purpose hardware. Generally, both FPGA and ASIC architectures require higher development costs than PC based systems. However, when manufactured at high volumes they quickly excel PC clusters with respect to cost-efficiency.
The main issue of this thesis is whether it is possible to develop alternative hardware architectures for collision search which offer better efficiency than standard PC architectures. Given a certain amount of money, which hardware architecture should be invested in to gain best performance results for collision search? Our solution is a highly specialized, full collision search unit called μCS which is based on a very compact ASIC microprocessor referred to as μMD.
In the cryptographic literature, there are few works that deal with practical aspects of collision search for MD5-family hash functions. Most of the contributions on collision search are dedicated to rather theoretical problems. To our best knowledge, this is the first work that analyzes implementation requirements for collision search algorithms from a computational perspective.
This work is structured as follows. Section 2 gives an introduction to the basic features of MD4-family hash functions and their attacks. In contrast to this, Section 3 presents the design details of μMD and μCS. In Section 4, we develop a metric for adequately describing the cost-efficiency of hardware units with respect to collision search. We then use this metric to compare our solution with a standard Pentium 4 based PC system. We close with a short conclusion.

2 Attacks on Hash Functions of the MD4-family

A (cryptographic) hash function is an efficiently evaluable mapping h which maps arbitrary-sized messages to fixed-size hash values [2]. There are at least three features a secure hash function is expected to have; (first) preimage resistance, second preimage resistance, and collision resistance. Successful attacks on collision resistance, i.e. finding two distinct messages that map to the same hash value, are much more promising, and so most attacks in the literature focus hereon.
As it is hard to efficiently describe algorithms that directly process inputs of variable length, hash functions of the MD4-family first divide the input message M into fixed-size message blocks, which are successively processed by a so called compression function. To impose chaining dependencies between consecutive message blocks, the compression function also processes the output of the immediately preceding computation, which is in this context also called chaining value. The first chaining value is a fixed initialization vector IV. The output of the computation of the last message block is defined to be the output of the entire hash function. Hash functions of the MD4-family mainly differ in their initialization vectors and their compression functions.
Generally, there are two types of attacks on MD4-family hash functions, generic and specific attacks. Generic attacks are attacks that are applicable to all (even ideal) hash functions. Specific attacks try to exploit the knowledge of the inner structure of the hash function and its inherent weaknesses. In this way, it is possible to *construct* collisions to a certain extent. Specific attacks are always dedicated to a single concrete hash function. However, there are some general design principles for developing specific attacks for MD4-family hash functions. Such attacks can be divided into two phases. The first phase launches a differential attack [1] on the inner structure of the compression function. The second phase consists of usefully utilizing the remaining freedom of choice for the concrete message bits. This freedom can of course be used to predefine parts of the input messages. However, another and very popular application is to exploit it for a *significant* acceleration of the collision search. These techniques are called single step modifications, multi step modifications, and tunneling [4, 5, 11].
For MD5 [7], there exist several efficient collision search algorithms [3, 4, 8] constructed according to these principles. Joŝĉák [3] compares their performances in more detail, while Klima's collision search (CS) approach [4] turns out to be the fastest. In contrast to the other ones, this algorithm extensively makes use of tunneling. As an example and for comparison reasons, we fully implemented CS into our μCS unit.

3 Collision Search Processor Design

When analyzing MD4-family hash functions and their corresponding collision search algorithm, one can find several important hints on how a suitable hardware architecture should look like. First, MD4-family hash functions have been developed for 32-bit processor systems. Our target architecture should also support 32-bit operands, otherwise expensive correction operations have to be applied. Second, collision search algorithms can hardly be parallelized on lower hierarchical levels, since most operations also compute the result of their predecessor. Third, tunneling will, besides multi step modifications, become a standard means for improving collision search based on differential patterns. Unfortunately, tunneling highly parameterizes the computation path using loop constructions. This requires efficient resource reuse, while at the same time making usual hardware acceleration techniques like pipelining hardly useful. We finally decided to implement the collision search algorithm CS on a 32-bit microprocessor based ASIC architecture. The central microprocessor we call μMD, the final collision search circuit μCS. μMD uses a very small instruction set, consisting of not more than sixteen native commands. As we aimed at a entirely compact solution, we maximized reuse of program code wherever possible. As a result, we also implemented a sufficiently large hardware stack and indirect load and store operations.

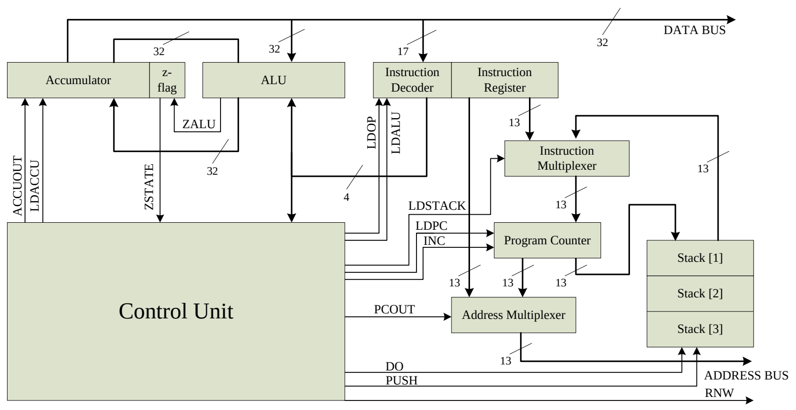

Figure 1: Inner structure of processor

In combination, they provide for a comfortable access to parameterized subfunctions. μMD is based on the design approach from [6]. The instruction set is adapted according to the requirements of the target application. Therefore, several instructions have been deleted, others have been added.
μCS is our final integrated circuit for collision search. Roughly spoken, it consists of a single μMD unit, additional memory and I/O logic. To start a computation, it just needs an initial seed for its integrated PRNG. When a collision is found, the corresponding message words are returned. Except for the PRNG initialization phase and the collision output sequence, there is no further I/O communication required. This decisively supports parallelization approaches making the overhead for additional control logic negligible.

4 Implementation

Implemented on an FPGA, μMD is comparably small. It uses about 9 percent of the slices of a Spartan3 xc3s1000 FPGA. The final clock frequency is about 95 MHz after synthesis. Synthesizing μCS for standard cells (UMC 130 nm) requires 0.959618 mm^2 chip area. Contrarily, μMD can be realized with only 0.026626 mm^2. This is just 2.77% of the chip area for the full μCS unit. μMD requires about 6000 gate equivalents (GE), μCS circa 210000 GE. For comparison, we also tested collision search on a standard PC processor. We used a Pentium 4, 2.0 GHz machine (Northwood core) with approximately 55 million transistors build in 130 nm circuit technology. Its die size is 146 mm^2.
We define the time $T = \frac{C}{f}$ to be the average time for a single unit to find a collision. Unfortunately, μMD and μCS are not available for a practical test run. For μCS, T is computed based on the average number C of cycles required to find a collision and the frequency f.
After implementing an assembler version of CS, we estimate CS executed on μMD to require roughly eight times more clock cycles than on a Pentium 4. Assuming equal production constraints, meaning the same price per chip area, each of our solution is much more effective than a comparable Pentium 4 architecture. The higher speed of a Pentium 4 processor is achieved only with much more area. The area-time product P reflects this fact appropriately.
ting an assembler version of CS, we estimate CS executed on μMD to require roughly eight times more clock cycles than on a Pentium 4. Assuming equal production constraints, meaning the same price per chip area, each of our solution is much more effective than a comparable Pentium 4 architecture. The higher speed of a Pentium 4 processor is achieved only with much more area. The area-time product P reflects this fact appropriately.

	Pentium 4	μCS	μMD
Q_A	$0.3425\frac{€}{mm^2}$	$0.3425\frac{€}{mm^2}$	$0.3425\frac{€}{mm^2}$
C	$60 \cdot 10^9$	$480 \cdot 10^9$	$480 \cdot 10^9$
f	2GHz	102.9 MHz	228.8 MHz
T	30 s	4660.8 s	2097.6 s
A	146 mm^2	0.959618 mm^2	0.026626 mm^2
P	4380	4472.6	55.9
Q_s	50 €	0.3287 €	-
O_p	300 %	5 %	-
Q_p	200 €	0.3451 €	-
R	6000	1608.4	-

Table 1: Cost overview

It is also interesting to compare the Pentium 4 with our *full* collision search unit μCS. One can see that a single unit of μCS is just 1.021 times less efficient than a Pentium 4 processor, although it includes all logic and data required to immediately start a collision search.
When we use an off-the-shelf standard PC for parallelization, we have to consider the costs for I/O, ROMs, motherboards, fans, and additional equipment for parallelization like network cards and cables. Altogether, we believe the costs to be at least 200 €, whereas we assume the Pentium 4 2.0 GHz to have a price of roughly 50 €. As a consequence, we estimate the price per chip Q_A area to be $Q_A = \frac{50\,€}{146\ mm^2} = 0.3425\frac{€}{mm^2}$. Using such off-the-shelf standard PCs, the processors are connected to each other by standard network equipment. The costs per unit will increase four-fold. This means, that for each single parallelized Pentium 4 processor one can buy four boxed ones. In other words, the parallelization overhead O_p is 300 %.
The parallelization of μCS requires only few additional logic per unit. For our solution, we assume the overhead in area to almost negligible, i.e. ranging below 5 %.
Based on these considerations, we believe that our *full* collision search solution is noticeably more effective for collision search than parallelized Pentium 4 processors. Our estimates are summed up in Table 1. Q_s reflects the price for a single standalone unit of the corresponding architecture. In contrast, Q_p is the average price for a single unit after parallelization. $R = Q_p \cdot T$ then reflect the amount of money required to find a collision within one second.
Obviously, for finding a single MD5 collision per second one has to spend 6000 € in (30) parallelized standard PCs. Assuming similar constraints for manufacturing ASICs, the price for the same performance invested in parallelized μCS units is 1608.4 €. So, it is almost four times more cost-efficient than the Pentium 4 standard PC architecture.

4.1 Estimations for SHA-1

We believe that a comparable implementation of a SHA-1 collision search algorithm in dedicated and parallelized collision search units has similar performance characteristics.
The average number of required clock cycles to find a collision is primarily dependent on the available theoretical results. Currently, attacks on SHA-1 have a complexity of about $O(2^{63})$. For practical attacks, we believe this number to be still very large.
Given 1 mio. €, an attacker can buy enough standard PC equipment to on average find a MD5 collision within $\frac{6000}{1000000} = 0.006$ s. Invested in our collision search units, it would take only $\frac{1608.4}{1000000} = 0.0016084$ s. Finding MD5 collisions based on CS has in [3] reported to be have a complexity of about $7.7 \cdot 2^{30}$ MD5 step operations. The current bound for SHA-1 collisions is $O(2^{63})$. Assuming a complexity of exactly 2^{63} step operations and a similar execution time for a

single step operation, finding collisions for SHA-1 takes about 2^{30} times more time than for MD5. We can thus conclude how long it would take to find a single SHA-1 collision with equipment for 1 mio. €. Invested in standard PCs, it would take $0.006\ \text{s} \cdot 2^{30} = 6442450\ \text{s}$ or almost 75 days. Using our collision search units, this time would be only $0.0016084\ \text{s} \cdot 2^{30} = 1727006\ \text{s}$ or roughly 20 days.

5 Conclusion

In this work we analyzed the hardware requirements of current and future collision search algorithms for hash functions of the MD4-family. We used our results to develop an appropriate hardware platform which executes collision search algorithms about four times faster than standard PC architectures.
The heart of our design is a very small microprocessor μMD with only sixteen instructions. At the same time, it provides very effective means to support program code reuse, what greatly helps to keep the size of our overall collision search unit μCS small.
In the context of MD4-family hash functions, μMD is general-purpose, meaning that it is appropriate for the execution of all MD4-family hash functions and also of all corresponding current and future collision search algorithms.
In contrast to standard PCs, the final collision search unit needs only very little additional logic. This reduces its price and greatly eases parallelization approaches.
We believe that our design approach is much better suited for collision search than standard PCs. When money is spent on collision search, our design, massively parallelized, is nearly four times more cost-efficient than parallelized P4 standard PCs. Given 1 million €, our solution can so find a SHA-1 collision in about 20 days, whereas a Pentium 4 based architecture would take nearly 75 days.

References

[1] E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*, 1993.

[2] M. Daum. *Cryptanalysis of Hash Functions of the MD4-Family.* PhD thesis, Ruhr-Universität Bochum, 2005.

[3] D. Joŝĉák. Finding collisions in cryptographic hash functions. Master's thesis, Univerzita Karlova v Praze, 2006.

[4] V. Klima. *Tunnels in Hash Functions: MD5 Collisions Within a Minute.* Cryptology ePrint Archive, Report 2006/105, 2006.

[5] J. Liang and X. Lai. *Improved Collision Attack on Hash Function MD5.* Cryptology ePrint Archive, Report 2005/425, November 2005.

[6] J. Reichardt and B. Schwarz. *VHDL-Synthese.* Oldenbourg, third edition, 2003.

[7] R. Rivest. *The MD5 Message-Digest Algorithm, RFC 1321*, 1992.

[8] M. Stevens. *Fast Collision Attack on MD5.* Cryptology ePrint Archive, Report 2006/104, 2006.

[9] X. Wang, A. Yao, and F. Yao. *Cryptanalysis of SHA-1.* Presented at the Cryptographic Hash Workshop hosted by NIST, October 2005.

[10] X. Wang, Y. L. Yin, and X. Yu. *Finding Collisions in the Full SHA-1.* In *Advances in Cryptology — EUROCRYPT 2005.* Springer Verlag, August 2005.

[11] X. Wang and X. Yu. *How to Break MD5 and other Hash Functions.* In *Advances in Cryptology — EUROCRYPT 2005.* Springer Verlag, May 2005.