Technical University of Denmark

Hash3: Proofs, Analysis and Implementation

Report on ECRYPT II Event on Hash Functions

Gauravaram, Praveen

Publication date:
2009

Document Version
Early version, also known as pre-print

Citation (APA):
Gauravaram, P. (2009). Hash3: Proofs, Analysis and Implementation: Report on ECRYPT II Event on Hash Functions. (MAT REPORT; No. 2009-03).

Report regarding the winter school on Hash^3: Proofs, Analysis and Implementation

Praveen Gauravaram

Department of Mathematics,
Technical University of Denmark
Building S303, Matematiktorvet
Kgs.Lyngby, Denmark-2800
Email: p.gauravaram@mat.dtu.dk

December 3, 2009

Abstract

This report outlines the talks presented at the winter school on Hash^3: Proofs, Analysis, and Implementation [9]. In general, speakers may not write everything what they talk on the slides. So, this report also outlines such findings following the understanding of the author of this report. The author of this report would like to disclaim that any mistakes in this report are solely due to author of this report as not all of the technical details are verified with the speakers. The findings presented in this report are solely due to the author's understanding of the talks at the winter school. For many of the talks, the author of this report has spent some time in understanding some technical details (using prior knowledge and (re)visiting the literature) and explained that in this report. Of course, not all the details are covered while exploring the literature.

0.1 First day: 16/11/2009

0.1.1 Perspective on hash functions

Bart discussed overall state of art of hash functions with an emphasis on the history and the current state of art of hash functions. The idea of one-way hash functions was known in 1873 itself due to Jevons who proposed a kind of one-way hash function which easily multiplies two prime numbers whereas it is difficult to factor. This was noted in [15]. Hash functions were known as modification detection codes (MDCs) in 70s and 80s due to IBM. In Belgium, the banking system in 80s used a 128-bit hash function for information authentication wherein the hash value transferred with a courier on the magnetic strip is verified via a phone after the recipient obtains the hash value. This ensures that the courier person has not modified the message. Prior to 1990, cryptology was in hardware and moved to software in 90s. Hash functions have become Swiss army knife of cryptography as we expect many properties from them and due to their diverse applications.

Most of the hash functions in the late 70s and 80s are based on DES (e.g., Rabin'78, Davies-Meyer and Matyas-Meyer-Oseas). During that time, people did not think outside the box and apart from few researchers like Lai and Massey, research in the new block cipher proposals was not considered. For cryptographic algorithms, if we aim for two out of cost, performance and security, we actually end up sacrificing the third one. The first time ever the initial value (IV) was mentioned in a hash function was for MDC-2. Rivest was the first one who coined the term "strengthening" (Need to be verified.). A new reference for [Dean's'99] second preimage attack on hash functions based on fixed points is [Dean-Felten-Hu'99] (This reference reflects the submission to Crypto 1999 conference but the paper was not accepted for the publication).

In the last few years, we have seen several researchers claiming a real collision on SHA-1 (e.g., Wang et al in 2^{63}, Mendel et al in 2^{62}, McDonald et al. in 2^{52}) without any publication of their result. All these researchers have made NIST to win on its claim that SHA-1 is fine for the collision resistance applications such as digital signatures till 2010 [16] which was made 3 years before the start of SHA-3 competition. Researchers should share their ideas to really come up with a collision on SHA-1.

There is a need for light weight hash functions. In Belgium, public keys are generated in smart cards. The need for hash functions was highlighted in mid 80s itself at Eurocrypt 1986 in a 2-page paper "On the need for hash functions".

0.1.2 Definitions: Relations and Preservation

Tom discussed about the need for the formalising definitions of hash functions and discussed about some of them following his FSE 2004 paper with Rogaway [18]. Tom also gave a detailed explanation regarding the collision resistance preservation for Damgård's hash function [6]. Note that in hash function padding, a bit 1 is needed when we do length encoding of the message blocks. It is not needed when we do length encoding of the bits. eSec is already known as TCR and UOWHF and aSec (which refers to concrete hash functions like SHA-1) is completely new. In Slide 37, Coll → Pre, when the compression function is not doing much compression. eSec hash functions are sufficient for the domain extension of signatures. We can't define choosing uniformly at random from $\{0, 1\}^*$. His slides are quite self-explanatory.

0.1.3 Rebound Cryptanalysis

The fundamental idea of attack is discussed. The idea makes use of the known 4-round differential on the wide-trail design to exploit the available degrees of freedom in the AES based/derived hash functions in order to mount collision attacks (may be semi-free start and free-start). The easily understood example is a 4-round collision attack on the Whirlpool hash function [14]. This attack with further improvements has been extensively applied to the compression function of Whirlpool and LANE compression functions that are going to appear in Asiacrypt 2009.

In most of the applications of this idea, the average complexity for the candidate found (the possible exact value) for the inbound phase is either low or close to 1. The interesting point to note here is that no one has applied this method to attack AXR based designs like SHA-1. In principle, the idea of rebound attack works on these designs as well. The research problem is about how to find good differential paths, what are the good paths and how many rounds can be covered with an average complexity of 1.

0.1.4 Security Proofs: Possibilities

This talk is mostly related to the paper on "Blockcipher-Based Hashing Revisited" by Martijn Stam at FSE 2009. Martijn discussed about generalizing Black, Rogaway and Shrimpton analysis of twelve provably collision and (second) preimage resistant compression functions of PGV. Preneel pointed out there are actually only two schemes (I think one is Matyas-Meyer-Oseas (MMO) and the other is Miyaguchi-Preneel) and all others are derived from them.

0.2 Second day: 17/11/2009

0.2.1 Software Benchmarking

Daniel explained the reasons for Rijndael block cipher to win the AES competition over Serpent. The main reason for Serpent to lose in the competition is its software speed and Serpent is generally is the slowest of the AES finalists. An overview of the reasons that might contribute to the speed of the hash functions in software are given. The reasons include hash function parameters (for example, speed varies depending on the hash value size), number of cores used for hashing, CPU (that's why we use a standard way of measuring speed in clock cycles), length of the message we want to hash (hashing is faster for short messages than for the long ones), implementation, compiler and compiler options. The other interesting note is that the main goal of the benchmark reports is to accurately predict the speed that the user will see. Benchmarking of cryptographic algorithms in software during NESSIE, ECRYPT I and ECRYPT II process and hash functions in the SHA-3 competition was discussed.

0.2.2 Herding and Generic Collision Attacks

Elena presented the applications of the generic multicollision attack of Joux [10]. They include building multicollisions for the cascaded hash functions, expandable messages to mount long message second preimages [7, 12], building diamond structure [12] to mount herding [11] and long message second priemage attacks [1] on the Merkle-Damgård hash functions and their variants like Dithering hash functions.

0.2.3 Cryptanalysis of Stream-based Hashes

Thomas gave a brief comparison of block based (e.g.SHA family) and stream based hash functions based on some observations. For block-based hashes, the bit-rate $r \geq c$ where c is the capacity of the hash function. For stream based hashes, $r < c$. For block based designs, finding a differential differential path is not that hard but using degrees of freedom is hard and this opposite for the stream based hashes.

In the stream based hashes (whose internal function is in general permutation), the increase in the factor $\frac{c}{r}$ gives less control to the attacker and increasing the number of rounds R leads to less good differential paths. In these hash functions, after message is processed, there are some blank rounds followed by the output transformation. The output transformation is either just truncation of the internal state to build hash function as in Grindahl, CubeHash or extract the chunks from the blank rounds to build the hash value as in RadioGatun, Lux and Keccak.

The generic attacks on some of these structures include meet-in-the-middle and slide attacks. Note that slide attacks on these structures (Grindahl and Lux with chosen salt) presented in literature do not seem to be useful to find collisions but can distinguish hash function from a random oracle and secret key can be recovered in some MAC settings. Linear differential paths for CubeHash 1/36 and truncated differential paths for CubeHash 2/36 and Grindahl are presented.

0.2.4 Security Proofs: Impossibilities

Martijn discussed about the impossibility results in the hash function proofs. They include the impossibility of designing rate-1 hash functions based on block ciphers following the paper by [3], security/efficiency trade-offs for the permutation based hash functions [19] and [21]. Some attack based designs are also discussed. They include Combining Compression Functions and Block Cipher-Based Hash Functions [17] and Security Analysis of Constructions Combining FIL Random Oracles [20].

0.3 Third day: 18/11/2009

0.3.1 Hashing and The Intel AES Instructions Set

Shay presented a study on the impact of the new AES instruction set (AES-NI) (to appear on the Westmere processors in 2010) on the performance of the AES based (those that use AES as it is) and AES inspired (those that use AES with some modifications) hash functions in the SHA-3 competition. Performance of the candidates using AES-NI is measured using a novel software technique based on the publicly available information on the Nehalem processor.

In the AES-NI, there are 4 instructions to perform AES encryption and decryption and 2 instructions for the key expansion. AES-NI can be directly used for the AES based hash functions such as ECHO, LANE, VORTEX and SHAvite-3 and all these directly benefit in the improved performance (For example, without AES-NI, speed of Vortex is 46.3 cycles/byte whereas with AES-NI it raises to 4.4 cycles/byte). AES inspired schemes like Grøstl, Fugue and Twister are unlikely to benefit from the AES-NI as it can be used only for the S-box. Only 256-bit versions of Cheetah and Lux might benefit from AES-NI. NIST used this study in its evaluation of SHA-3 candidates.

0.3.2 Analysis Techniques for AXR and MD4-like constructions

Christophe presented techniques for collision attacks in AXR and MD4-like hash functions. He presented a high-level overview of the dedicated collision attack on hash functions. The goal in the collision attack on the compression function (resp. hash function) is to find a pair of distinct message blocks (resp. messages) which minimize the differences of the internal values. That is, the hamming distance between the internal values of the messages should be as minimum as possible and hopefully close to zero. This problem is similar to constructing a pair of codewords with a short hamming distance for a given linear code. The solution is to use a probabilistic algorithm by Leo [13] and Canteaut and Chabaud [5] or its cyclic variant. These algorithms can be applied to hash functions by linearizing the hash function (replace non-linear components with xor operations). When the structure is linear, it is easy to to keep track of all solution which satisfy the restrictions that are imposed during the low-weight search. An illustration on SHA-1 was explained. The other approach is the non-linear approach.

0.3.3 Survey of Dedicated Preimage Attacks

Christian presented a brief survey of the known techniques to mount preimage attack on a compression function (which leads to a pseudo preimage for an MD hash function) and how to extend it to the full hash function. Of all the techniques used to find preimages for the compression functions, meet-in-the-middle strategy has become quite effective and several MD4 like compression functions were shown to be vulnerable against this method in the last couple of years.

0.4 Fourth day: 19/11/2009

0.4.1 Hardware Benchmarking

The hardware benchmarking is influenced by the factors of speed or throughput (Gbits/sec), area (gate or transistor count and memory), power or energy consumption (for cooling an transmission) and security (side channel resistance). The power required for the HW circuit is directly related to the heat consumed.

Wastage of power can be reduced by reducing the power consumed by the HW processor and the idea is to use multi-core designs. Benchmarking results in hardware depend on ASIC, FPGA, Hardware API, Bandwidth, gap between application and architecture and Transformations.

Hardware design is a translation from the specification into Register Transfer Level (RTL, e.g, VHDL, Verilog). The definition of one gate refers to 4 transistor NAND gates. Gate count refers to the number of gates needed to implement a microprocessor design. Benchmarking on FPGA is cheaper than AISC (tools are almost free at least in the universities).

For hash function implementation in hardware, integration of hash module is necessary. Three types of hardware reportings were mentioned in SHA-3 zoo for hash functions without interface. For example, it is important to measure how much it costs to talk from HW to SW and back to HW.

0.4.2 Indifferentiability and Related Proof Techniques

Hash functions indifferentiable from Random oracles can be instiantiated in the protocols that are proven secure in the Random oracle model. This notion is good because of the composition. The cost of replacing a random oracle with an indifferentiable hash function provides an upper bound of indifferentiability. Hence, we just have to prove whether a particular hash function construction is indifferentiable without the need to prove the cost involved in replacing a random oracle with an indifferentiable hash function.

Some important points to note here are: In the slides in the Figure related to composition, the Environment watches the interaction of various parties and distinguisher runs the environment. When we prove indifferentiability of a hash scheme, we must present an efficient simulator. Alternatively, proving indifferentiability means showing an efficient simulator. Simulator works by maintaining a tree structure which starts with the IV of the hash function. It develops the tree when it gets queries. Whenever a query does not have a response in the tree, the simulator returns a random value to the distinguisher. The simulator queries the random oracle when required. This necessity depends whether the distinguisher is querying the last block which depends on the structure of the message block (for example, composition of the padding bits) and also on the ideal component to which it queries. For example, the function g in NMAC and a special padding method for the last block in PFMD determines the last message block query. So, when the simulator gets such queries for the first time and establishes a connection to the current block with the previous message blocks in the tree, it extracts the message from its tree and queries RO with that message. By doing so, the simulator maintains its consistency with the Random Oracle.

The popular Merkle-Damgård construction fails to be indifferentiable from a random oracle due to the length extension attack (for a hash function, knowing the hash value and length of the message, we can compute the new hash value). For this hash function, if we try to present a simulator in order to prove its indifferentiability, the simulator needs to invert the random oracle. An easy fix to MD is the NMAC construction where the output of the MD is processed using another independent function g. This construction is indifferentiable from a random oracle.

The recent research [8] has identified a property of hash functions called preimage awareness (PrA) which is similar to the plaintext awareness [2] and extractability for perfectly one-way functions [4] that are used in cryptographic protocols. If an attacker can announce a range point y of a hash function and subsequently produce a preimage M of y then the attacker almost certainly "knew" the preimage M when it announced y. This "knowledge of preimage" can be efficiently "extracted". Extractor is a deterministic function whose job is to extract the knowledge. The MD construction is preimage-aware preserving and by combining it with a fixed input-length random oracle (FIL-RO) g, one can obtain a variable input length random oracle (VIL-RO). Note that if we find a collision in the preimage-aware hash function, we can break PrA property. There are no proofs yet that show that when an FIL-RO compression function in a PrA construction is also used as the external function g, the hash function is indifferentiable from a Random Oracle.

Bibliography

[1] E. Andreeva, C. Bouillaguet, P.-A. Fouque, J. J. Hoch, J. Kelsey, A. Shamir, and S. Zimmer. Second preimage attacks on dithered hash functions. In N. P. Smart, editor, *Advances in Cryptology - EUROCRYPT Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 270–288. Springer-Verlag, 2008.

[2] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. D. Santis, editor, *Advances in Cryptology—EUROCRYPT 94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1994.

[3] J. Black, M. Cochran, and T. Shrimpton. On the impossibility of highly-efficient blockcipher-based hash functions. In R. Cramer, editor, *Advances in Cryptology*, volume 3494 of *Lecture Notes in Computer Science*, pages 526–541. Springer, 2005.

[4] R. Canetti and R. R. Dakdouk. Extractable Perfectly One-Way Functions. In *Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part II*, pages 449–460, Berlin, Heidelberg, 2008. Springer-Verlag.

[5] A. Canteaut and F. Chabaud. A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, 1998.

[6] I. Damgård. A Design Principle for Hash Functions. In G. Brassard, editor, *Advances in Cryptology: CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer-Verlag, 1989.

[7] R. D. Dean. *Formal Aspects of Mobile Code Security.* PhD thesis, Princeton University, 1999.

[8] Y. Dodis, T. Ristenpart, and T. Shrimpton. Salvaging Merkle-Damgård for Practical Applications. In A. Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 371–388. Springer, 2009.

[9] ECRYPT-II. Winter school on Hash^3 Proofs, Analysis and Implementation, 2009. Website at `http://crypto.rd.francetelecom.com/echo/` (Accessed on 27/11/2009).

[10] A. Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In M. Franklin, editor, *Advances in Cryptology-CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316, Santa Barbara, California, USA, Aug. 15–19 2004. Springer-Verlag.

[11] J. Kelsey and T. Kohno. Herding Hash Functions and the Nostradamus Attack. In S. Vaudenay, editor, *Advanes in Cryptology-EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 183–200. Springer-Verlag, 2006.

[12] J. Kelsey and B. Schneier. Second Preimages on n-bit Hash Functions for Much Less than 2^n Work. In R. Cramer, editor, *Advances in Cryptology–EUROCRYPT'05*, volume 3494 of *Lecture Notes in Computer Science*, pages 474–490. Springer-Verlag, 2005.

[13] J. S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354, 1988.

[14] F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen. The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In O. Dunkelman, editor, *Fast Software Encryption*, volume 5665 of *Lecture Notes in Computer Science*, pages 260–276. Springer, 2009.

[15] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*, chapter 9, pages 321–383. The CRC Press Series on Discrete Mathematics and its Applications. CRC Press, 1997.

[16] National Institute of Standards and Technology (NIST). NIST Comments on Cryptanalytic Attacks on SHA-1, January 2006. This short notice by NIST is available at `http://csrc.nist.gov/groups/ST/hash/statement.html`(Accessed on 27/11/2009).

[17] T. Peyrin, H. Gilbert, F. Muller, and M. J. B. Robshaw. Combining compression functions and block cipher-based hash functions. In X. Lai and K. Chen, editors, *Advances in Cryptology - ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 315–331. Springer, 2006.

[18] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In B. K. Roy and W. Meier, editors, *Fast Software Encryption (FSE)*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer-Verlag, 2004.

[19] P. Rogaway and J. P. Steinberger. Security/efficiency tradeoffs for permutation-based hashing. In N. P. Smart, editor, *Advances in Cryptology - EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 220–236. Springer, 2008.

[20] Y. Seurin and T. Peyrin. Security analysis of constructions combining FIL random oracles. In A. Biryukov, editor, *Fast Software Encryption*, volume 4593 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2007.

[21] M. Stam. Another Glance At Blockcipher Based Hashing. Cryptology ePrint Archive, Report 2008/071, 2008. Paper is available at `http://eprint.iacr.org/2008/071`.