International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 5, May - 2014

Design and Implementation of Cryptography Hash Function

Prashant M Hullatti
M.Tech
CMRIT, Bangalore

Sridevi S
Asst. Professor
CMRIT, Bangalore

***Abstract*—SHA (Secure Hash Algorithm) is a famous message compress standard used in computer cryptography, it can compress a long message to become a short message abstract. The algorithm can be used in many protocols or**
Secure Algorithm, especially for DSS. In this paper, the improved version SHA-1 is analysised, then improved and implemented in HDL (Hardware Description Language) and FPGA. QuartusII is used to compile and generate the function modules, RTL level description circuit and simulated waveform. RTL level description is the circuit connection in FPGA chip. It shows the connection of the modules. Simulated waveform shows us the timing and the function of the SHA-1 module. The algorithm is implied easily. And the SHA-1 module that design in this paper used less memory units and logic elements. It can be used in DSA or any protocols or secure algorithm.

INTRODUCTION

Most cryptographic algorithms are inherently based on the unavailability of sufficient computing resources to break them, since they normally rely on some *hard problems*, supposed to be intractable with ordinary computing platforms.

Consequently, cryptanalysis, i.e., the study of methods for breaking cryptographic algorithms, can greatly benefit from special-purpose machines as a key aspect enabling high performance security attacks. Not surprisingly, the most remarkable results obtained in the past in the field of cryptanalysis relied on some custom hardware or application-specific computing platform [1], [2]. In this scenario, reconfigurable hardware technologies, now very popular in a large variety of general applications, can provide some new, crucial advantages. In fact modern Field-Programmable Gate Arrays (FPGAs) provide designers with the possibility of speeding up pure software applications by

means of hardware-implemented kernels. Importantly, unlike Application Specific Integrated Circuits (ASICs), FPGAs provide a cost-effective approach to building hardware-accelerated computing platforms, as they require much lower development efforts and incur marginal nonrecurring engineering costs Additionally, as quantitatively proven in this paper, reconfigurable hardware technologies also achieve much higher computational power/cost ratios compared to standard software high-performance computing (HPC) platforms. This has some important implications for cryptanalysis, where economic factors play an essential role. Cost, however, is not the only factor that may give reconfigurable technologies
a special role for cryptanalysis. FPGA-based systems, in fact, retain the flexibility of software development as they can be reconfigured at will, changing the hardware design over time and adapting it to the problem under study

ARCHITECTURE

We extensively experimented with algorithm variants and different opportunities for speeding up the collision search process by using an ad-hoc FPGA-based hardware platform. depicts the high-level organization of the basic building block of the platform. The system can process concurrently two messages, storing the differential characteristic, performing a configurable compliance check, and tracing statistics related to the collision search process. The fundamental components include two shift registers used to store and update the last sixteen words of the message expansion sequence as well as two groups of registers used to store the last five values of variable for both messages. The system also includes some memory blocks for the part and the part of the differential characteristic (shared among the two messages) and two memory blocks storing the messages themselves, as obtained during message enumeration. The latter process is supported by a Linear Feedback Shift Register used to generate pseudorandom bit sequences. The *evaluate* block is used to compare the results against the constraints imposed by the characteristic upon each round. The blocks used to generate words are controlled by a further hardware component, labeled AP in the figure keeping into account the position of the Auxiliary Paths and the bit-flipping operations they require. Notice that the system can start comparing message pairs from an arbitrary round in order to explore the behavior of the characteristic in any of its regions.

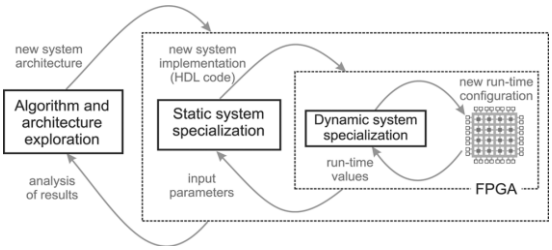
Related Works

The input of SHA is a message which is no longer than 2^{64} bit, and it can generate a 160 bit message abstract. If a message no longer than 2_{64} bit, it needs to be added zeros to make the message become a 2_{64} bit one. And if a message

IJERT

longer than 264 bit, it need to be separated into several groups. Every group contains 264 bit. Then the message groups will be converted into message abstract groups by SHA algorithm. When message abstract is generated, five 32 bit initial values A, B, C, D, E will be used.

A□0x67452301
B□0xefcdab89
C□0x98badcfe
D□0x10325476



Every time SHA-1 operate, non-linear function F_t, constant W_t and K_t are different if t is different value According to parameter t, the non-linear function F_t is

$$F_t(x,y,z) = (x \wedge y) \vee (\overline{x} \wedge z) \quad (t = 0 \sim 19)$$
$$F_t(x,y,z) = x \veebar y \veebar z \quad (t = 20 \sim 39)$$
$$F_t(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z) \quad (t = 40 \sim 59)$$
$$F_t(x,y,z) = x \veebar y \veebar z \quad (t = 60 \sim 79)$$

CRYPTANALYSIS OF THE SHA-1 HASH FUNCTION

This section deals with the hash function chosen as a case study for our work, SHA-1.Standardized by NIST as a Federal Information Processing Standard [6], the function takes a message of length less than bits and produces a 160-bit hash value. The input message is padded and then processed in 512-bit blocks in the Merkle Damgård iterative structure. Each iteration invokes a so-called *compression function* which takes five 32-bit chaining values along with a 512-bit message block and uses them to compute five new 32-bit values summed word-wise with the input 32-bit numbers to produce the subsequent five chaining values. The initial chaining values (called *IV*) are a set of fixed constants, and the last chaining values constitute the hash of the message.

The compression function of SHA-1 works as follows. For each 512-bit block of the padded message, divide it into sixteen 32-bit words,

Quantitative Performance Model for SHA-1 Attacks

In order to evaluate and compare quantitatively the effectiveness of the SHA-1 cryptanalysis techniques introduced here, we resorted to a detailed performance model, based on estimating

the expected number of executions for each round , denoted a , and the clock count they require. can be computed from and starting from the bottom of the characteristic, by using the recursive relationship

$$N(i+1) = N(i) \cdot P_u(i) \cdot 2^{D_w(i+1)} \Rightarrow$$
$$N(i) = N(i+1) \cdot 2^{-D_w(i+1)} / P_u(i)$$

with the initial value of , i.e., the number of executions of the last round (72 for the characteristic in the figure) equal to 1, since we will stop the search process as soon as we reach

the last round for the first time. This relationship is confirmed by, listing the values of , and for some rounds. Notice that the values of and are expressed as base-2 logarithms, so that the value of at a certain round can simply be obtained as the negated sum of all values

Since the execution of each round, i.e., the computation of and for and and the corresponding register updates, can be performed concurrently in a single clock cycle for a hardware implementation, we call this computation an *Elementary Operation* (EO), and the expected total number of round executions, i.e., the summation of parameters for all rounds , the *Mean number of EOs to Collision* (MEOC). For example, the characteristic of Fig. 2 has aMEOC equal to 54.51 (expressed as a base-2 logarithm).

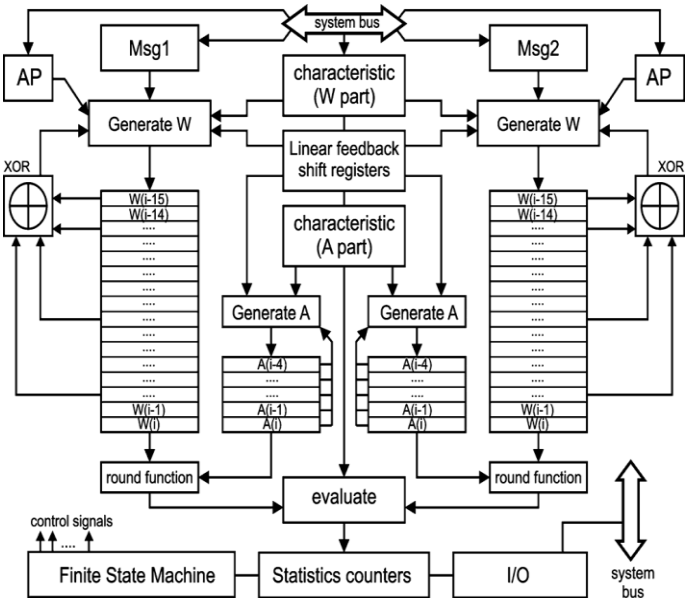
NEW ALGORITHM-LEVEL TECHNIQUES FOR SHA1 CRYPTANALYSIS

This section essentially deals with the outer loop in Fig, i.e., it presents the new algorithmic techniques that were identified by relying on FPGA-based acceleration. We extensively experimented with algorithm variants and different opportunities for speeding up the collision search process by using an ad-hoc FPGA-based hardware platform. Fig. 3 depicts the high-level organization of the basic building block of the platform. The system can process concurrently two messages, storing the differential characteristic, performing a configurable compliance check, and tracing statistics related to the collision search process.

The fundamental components include two shift registers used to store and update the last sixteen words of the message expansion sequence as well as two groups of registers used to store the last five values of variable for both messages. The system also includes some memory blocks for the part and the part of the differential characteristic (shared among the two messages) and two memory blocks storing the messages themselves, as obtained during message enumeration. The latter process is supported by a Linear Feedback Shift Register used to generate pseudorandom bit sequences. The *evaluate* block is used to compare the results against the constraints imposed by the characteristic upon each round keeping into account the position of the Auxiliary Paths and the bit-flipping operations they require. Notice that the system can start comparing message pairs from an arbitrary round in order to explore the behavior of the characteristic in any of its regions.

IJERT

To do so, it essentially reconstructs a random state in words at round and starts applying the computation and evaluation operations from that round on. The message enumeration logic makes sure that, albeit random (and hence not leading to an actual collision) the initial state at round is consistent with the overall structure of the differential characteristic.



IMPACT OF INTERBIT CONSTRAINTS (IBCS) FOR SOME ROUNDS

	without IBCs			with IBCs		
Round i	$\log_2 P_u(i)$	$\log_2 N(i)$	$C(i)$	$\log_2 P_u(i)$	$\log_2 N(i)$	$C(i)$
14	-3.273	50.093	1	-3.273	41.673	< 8
15	0	60.820	1	0	52.4	1
...	...	...	...	...	...	...
64	-3	28	1	-2	20	1
65	-2	25	1	-1	18	1
66	-3	23	1	-3	17	1
67	-3	20	1	-3	14	1
68	-5	17	1	-3	11	1
69	-5	12	1	-3	8	1
70	-4	7	1	-4	5	1
71	-3	3	1	-1	1	1

IMPACT OF CONSTRAINT RELAXATION (CR) FOR SOME ROUNDS

	without CR			with CR		
Round i	$\log_2 P_u(i)$	$\log_2 N(i)$	$C(i)$	$\log_2 P_u(i)$	$\log_2 N(i)$	$C(i)$
...	...	...	...	...	...	...
64	-2	20	1	-1	15.14	1
65	-1	18	1	-1	14.14	1
66	-3	17	1	-2	13.14	1
67	-3	14	1	-2	11.14	1
68	-3	11	1	-2.58	9.14	1
69	-3	8	1	-2.56	6.56	1
70	-4	5	1	-3	4	1
71	-1	1	1	-1	1	1

RECONFIGURABLE HARDWARE FOR SHA-1 CRYPTANALYSIS

As we emphasized above, reconfigurable technologies allow the *ad-hoc* generation of highly optimized systems tailored on specific input parameters, as well as the on-the-fly reconfiguration based on run-time values. This may constitute an especially relevant opportunity for cryptanalysis, as summarized in the inner loops of Fig. 1. This section shows how these opportunities were applied to our case-study

a) Static System Specialization: We identified numerous situations for the collision search process where static reconfiguration has a considerable impact on the implementation efficiency. One example is represented by the logic for the fast flipping of Auxiliary Paths bits [25]. As explained in Section III-B, flipping these bits immediately spawns a new message pair compliant to the characteristic up to the current round. This operation involves bits located at some particular positions depending on the specific characteristic and can be performed on-the-fly by a dedicated circuit like the AP block in Fig. 3. Other examples of characteristic-dependent optimizations include the XOR network for applying interbit constraints, the logic for the compliance check and constraint relaxation, and the implementation of what we called "segmented counters", used to efficiently enumerate the degrees of freedom ' ' in the part of the characteristic. On an FPGA device, this enumeration could be efficiently handled by hardware counters exploiting the optimized
carry propagation logic usually available on reconfigurable devices. Unfortunately, however, many ' ' bits in the characteristic are not physically consecutive, preventing the use of standard FPGA counters. The segmented counter technique, firstly described in overcomes this limitation by controlling the behavior of the circuit at very low level, in such a way as to skip

Proposed Solution

We essentially rely on two metrics to assess and compare the profitability of the proposed approach:
• the *computational density*, i.e., amount of delivered computation
per hardware unit, and
• the *absolute collision time* under a given cost budget to implement the cryptanalysis computing platform.

The computational density can be measured by means of the EOC of the SHA-1 cryptanalytic machine. The EOC reached by the designed platform is here much higher than other existing software and hardware solutions. Looking at hardware solutions in the literature, presents a microprocessor with minimal area requirements for speeding up collision search for the MD-4 hash function The most interesting comparisons, of course, consider solutions based on HPC facilities able to deliver the highest level of performance currently available.

IJERT

CONCLUSIONS

This paper presented a case for the role of hardware reconfigurability for cryptanalytic applications. Relying on FPGA-based hardware acceleration, we identified a few novel techniques at the algorithm level, in addition to a variety of architectural optimizations based on static and dynamic circuit specialization, playing a key role to maximize the performance and minimize the hardware complexity of the cryptanalytic platform. The FPGA-based SHA-1 cryptanalysis system presented in this work as a case-study achieved an EOC much higher than
other existing software and hardware solutions. Compared to a couple of supercomputing facilities based on cutting-edge architectures, such as hybrid multicore/SIMD and GPU-based parallel machines, a single SHA-1 core performs a collision search faster in spite of the difference in the clock frequency, while its extremely reduced footprint and cost allow improved levels of parallelism. As a result, under the same cost budget, the estimated times for a collision achieved by the FPGA-based platform are at least one order of magnitude lower than the HPC facilities, reaching the highest performance/cost ratio for SHA-1 collision search and providing a striking confirmation of the impact of hardware reconfigurability

REFERENCES

[1] E. Foundation, *Cracking DES. Secrets of Encryption Research, Wiretap Politics & Chip Design*. Sebastopol, CA, USA: O'Reilly Media, 1998.
[2] A. Sotirov,M. Stevens, J.Appelbaum, A. Lenstra, D.Molnar,D.Osvik, and B. de Weger, "MD5 considered harmful today: Creating a rogue CA certificate," in *Proc. 25th Chaos Communications Congress*
[3] X.Wang and H. Yu, "How to break MD5 and other hash functions," in *Proc. Advances in Cryptology (EUROCRYPT 2005)*, 2005, vol. 3494, pp. 19–35, ser. LNCS, Springer.
[4] M. Stevens, On Collisions for MD5 Jun. 2007
[5]G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, The KECCAK Sponge Function Family Oct. 2012 [Online]. Available: http://keccak.
noekeon.org
[6] A. Cilardo, L. Esposito, A. Veniero, A. Mazzeo, V. Beltran, and E. Ayguadé, "A CellBE-based HPC application for the analysis of vulnerabilities in cryptographic hash functions," in *Proc.* 12th Conf.High
Performance Computing and Communications (HPCC'10), 2010, pp.
450–457, IEEE.
[7] Deng An-Wen. Cryptography(in Chinese). China WaterPower Press. 2006:150-152
[8] Wenbo Mao.Modern Cryptography: Theory and Practice. Pearson Education. 2004:184-190
[9] A. Cilardo, G. P. Saggese, A. Mazzeo, and L. Romano, "Exploring the design-space for FPGA-based implementation of RSA," *Elsevier Microprocessors andMicrosystems*, vol. 28, no. 4, pp. 183–191,May2004
[10] A. Joux and T. Peyrin, "Hash functions and the (amplified) boomerang attack," in *Proc. Advances in Cryptology (CRYPTO 2007)*, 2007, vol. 4622, pp. 244–263, ser. LNCS, Springer.

IJERT