On The Cost of ASIC Hardware Crackers: A SHA-1 Case Study

Citation for published version:
Chattopadhyay, A, Khairallah, M, Leurent, G, Najm, Z, Peyrin, T & Velichkov, V 2021, On The Cost of ASIC Hardware Crackers: A SHA-1 Case Study. in *Topics in Cryptology – CT-RSA 2021.* Lecture Notes in Computer Science, vol. 12704, Springer International Publishing, pp. 657 – 681, The Cryptographer's Track at the RSA Conference 2021, San Francisco, United States, 17/05/21. https://doi.org/10.1007/978-3-030-75539-3_27

Digital Object Identifier (DOI):
10.1007/978-3-030-75539-3_27

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Topics in Cryptology – CT-RSA 2021

On The Cost of ASIC Hardware Crackers: A SHA-1 Case Study

Anupam Chattopadhyay[1], Mustafa Khairallah[1,4], Gaëtan Leurent[2], Zakaria Najm[1,3], Thomas Peyrin[1,4], Vesselin Velichkov[5]

[1] Nanyang Technological University, Singapore, Singapore
[2] Inria, Paris, France
[3] TU Delft, Delft, Netherlands
[4] Temasek Labs @ NTU, Singapore, Singapore
[5] University of Edinburgh, Edinburgh, UK

Abstract. In February 2017, the `SHA-1` hashing algorithm was practically broken using an identical-prefix collision attack implemented on a GPU cluster, and in January 2020 a chosen-prefix collision was first computed with practical implications on various security protocols. These advances opened the door for several research questions, such as the minimal cost to perform these attacks in practice. In particular, one may wonder what is the best technology for software/hardware cryptanalysis of such primitives. In this paper, we address some of these questions by studying the challenges and costs of building an ASIC cluster for performing attacks against a hash function. Our study takes into account different scenarios and includes two cryptanalytic strategies that can be used to find such collisions: a classical generic birthday search, and a state-of-the-art differential attack using neutral bits for `SHA-1`.
We show that for generic attacks, GPU and ASIC poses a serious practical threat to primitives with security level ~ 64 bits, with rented GPU a good solution for a one-off attack, and ASICs more efficient if the attack has to be run a few times. ASICs also pose a non-negligible security risk for primitives with 80-bit security. For differential attacks, GPUs (purchased or rented) are often a very cost-effective choice, but ASIC provides an alternative for organizations that can afford the initial cost and look for a compact, energy-efficient, reusable solution. In the case of `SHA-1`, we show that an ASIC cluster costing a few millions would be able to generate chosen-prefix collisions in a day or even in a minute. This extends the attack surface to TLS and SSH, for which the chosen-prefix collision would need to be generated very quickly.

Keywords: `SHA-1`, Cryptanalysis, ASIC, Birthday Problem, Hash Functions

1 Introduction

Hardware cryptanalysis has always been an important part of modern cryptography. It studies building application-specific electronic machines for performing cryptanalytic attacks. These machines can use different technologies, starting from mechanical computers during World War II, to FPGA, GPU or ASIC in the modern days. A full discussion of the history and state of the art of this field can be found in [11]. A widely held belief is that FPGAs and GPUs are suited for small-scale or low-budget computations, while ASIC is predicted to be better for heavy computational tasks or if the attacker has an important budget to spend. It is intuitive that a chip that is designed for a specific task is much more efficient than a general-purpose chip for the same task. However, since ASIC design has a huge non-recurring cost for fabrication, it is only competitive when a huge amount of chips is required. Besides, unlike the cryptographic algorithms themselves, which are usually optimized for hardware implementations, the cryptanalytic algorithms are usually designed for general-purpose computing machines. Hence, it is not necessarily true that ASIC implementations of such algorithms are more efficient. In other words, ASIC can always be at least as efficient as general-purpose CPUs or GPUS, as in the worst case the ASIC designer can simply design a circuit that is similar to the general-purpose one, but the gap in efficiency between the ASIC and the general-purpose circuit depends on the algorithm being implemented.

In general, ASIC provides an unfair advantage to players with bigger budgets. This has led to speculation that large intelligence entities may already possess ASIC hardware crackers that can break some of the widely used cryptographic schemes. In this paper, we address the question of the feasibility of such machines and whether it is more beneficial to use ASIC for cryptanalysis. The answer to this question is yes, but only for generic attacks of very large complexities, e.g. $> 2^{64}$. For low scale or more complicated cryptanalytic attacks, GPUs provide a very competitive option, due to re-usability, mass production and/or the possibility of renting them.

A relevant topic to our study is blockchain mining. As discussed earlier, big players can gain a huge advantage by using expensive ASICs. This has been a trend for Bitcoin specifically, where the introduction of a new ASIC machine lowers the profitability of older machines significantly. To maintain fairness of blockchain and cryptocurrency mining, memory-bound and ASIC-resistant hashing algorithms have been used, such as Ethash [24] for the Ethereum cryptocurrency and the X16R algorithm [25].

Related Work COPACOBANA [12] was introduced in CHES 2006 as an FPGA cluster consisting on 120 FPGAs. It is considered to be the first publicly reported configurable platform built specifically for cryptanalysis. The design philosophy behind the architecture depends on three assumptions:

1. Cryptanalytic algorithms are parallelisable.
2. Different nodes need to communicate with each other only for a very limited amount of time.
3. Since the target algorithms are computationally intensive, the communication with the host is very limited compared to the time spent on the computational tasks.

These assumptions are satisfied by both brute force (generic) and a lot of cryptanalytic attacks. Hence, the COPACOBANA has been used to accelerate several attacks [6]. In our study we follow the same assumptions and add one more assumption:

4. Each node requires a constant/low amount of storage. The overall attack can be implemented using an almost memory-less algorithm.

This assumption needs to be satisfied by the attack algorithm in order to make sure that the efficiency due to parallelisation is not lost due to memory operations. For example, a naive approach to implementing a generic birthday collision search on m nodes, can lead to only $\sqrt{m}$ speed up compared to a single node if the algorithm doesn't satisfy this assumption.

Our Contributions. This paper is an attempt at answering three important research questions:

– *Can the cost of the collision attacks against* `SHA-1` *be reduced?* There has been major breakthroughs in the cryptanalysis of `SHA-1` over the past few years, with the first practical identical-prefix collision (IPC) found in February 2017 [17] and the first chosen-prefix collision (CPC) found in January 2020 [14]. While these attacks are practical on general-purpose GPUs, they still take a few months to generate one collision, by both academic and industrial entities. Interestingly, the authors of [14] remarked that TLS and SSH connections using `SHA-1` signatures to authenticate the handshake could be attacked with the SLOTH attack [2] if the chosen-prefix collision can be generated quickly. Hence, we would like to check if ASIC can provide a better alternative to speed up the attacks, using larger budgets. We actually show that chosen-prefix collisions could be generated within a day or even a minute using an ASIC cluster costing a few dozen Million USD (the amortized cost per chosen-prefix collision is then much lower).
– *What is the difference between generic attacks and cryptanalytic attacks in terms of cost and implementation?* When analyzing a new cipher, any algorithm that has a theoretical time complexity lower than the generic attacks is considered a successful attack and the cipher is considered broken. For example, an n-bit hash function that is collision resistant up to the birthday bound is considered insecure if there is a cryptanalytic attack that requires less than $2^{0.9n/2}$ hash calls. Most of the time, researchers only measure time complexity in terms of function calls and ignore other operations required to perform the attack if they are much smaller. However, in practice, it can be a lot harder to implement a cryptanalytic attack compared to a generic attack, even with lower theoretical complexity. There are countless attacks published every year with a complexity very close to the generic one, but a natural example of such scenarios is the biclique attack against `AES` [4], where the brute force complexity is reduced only by a small factor from 2^{128} to $2^{126.1}$. However, one can question if implementing the simple brute force

attack would actually be much less complex in practice. In this paper, we compare the generic 64-bit birthday CPC attack over a 128-bit hash function to the cryptanalytic CPC attack against SHA-1 (which costs close to $2^{63.6}$ operations on GPUs, and of a lower complexity in theory) showing that in practice, the generic attack cost is more than 5 times cheaper than the ad-hoc CPC attack. Attacks like biclique or complex cryptanalysis are even more difficult to implement than the ad-hoc CPC attack and might require a huge memory, which probably makes the gap even larger. Hence, we argue that for a cryptanalytic attack to be competitive against a generic algorithm in practice, one must ensure a sufficiently large gap, at least of a factor 5, if not more (only an actual hardware implementation testing or estimation could give accurate bounds on that factor).

– *How secure is an 80-bit collision-resistant hash function?* In the NIST Lightweight Cryptography Workshop 2019, Tom Brostöm proposed an application for lightweight cryptography where the SIMON cipher [1] is used in the Davis-Meyer construction as a secure compression function which is collision-resistant at most up to 2^{64} computations [19]. Besides, it remains a common belief that SHA-1 is insecure due to the cryptanalytic attacks against it, but it would have still been acceptable otherwise. Actually, it is only since 2011 that 80-bit security is not recommended anymore by the NIST, and 80-bit security for data already encrypted with this level of protection is deemed acceptable as a legacy feature, accepting some inherent risk. Hence, we study the cost of implementing the generic 2^{80} birthday collision attack against SHA-1, showing that it is within our reach in the near future, costing ≈ 61 million USD to implement the attack in 1 month, which is not out of reach of large budget players, *e.g.* large government entities, and with the decreasing cost of ASICs, this will even be within reach of academic/industrial entities in the near future.

Finally, we argue that ASIC provides the most efficient technology for implementing high complexity and generic attacks, while GPU provides a competitive option for cryptanalytic and medium/low cost attacks.

2 Hash Functions and Cryptanalysis

Cryptographic hash functions are one of the main and most widely used primitives in symmetric key cryptography. One of their key applications is to provide data integrity by ensuring each message will lead to a seemingly random digital *fingerprint.* They are also used as building blocks of some digital signature and authentication schemes. A cryptographic hash function takes a message of arbitrary length as input and returns a fixed-size string, which is called the hash value/tag. In order for the function to be considered secure, it must be hard to find collisions, i.e. two or more different messages that have the same tag. More specifically, a n-bit cryptographically secure hash function must satisfy at least the security notion of collision resistance, *i.e.* finding a pair (M_1, M_2) of distinct messages, such that $H(M_1) = H(M_2)$ must require about $2^{n/2}$ computations.

2.1 SHA-1 and Related Attacks.

The `SHA-1` hash function defines a generalized-Feistel-based compression function used inside the Merkle-Damgård (MD) algorithm. It was selected in 1995 as a replacement for the `SHA-0` hash function after some weaknesses have been discovered in the latter. While the two functions are relatively similar, `SHA-1` was considered collision resistant till 2005, when Wang *et al.* proposed the first cryptanalytic attack on `SHA-1` [23]. Since then, a lot of efforts have been targeted towards making the attack more efficient. In 2015, the authors of [7] provided an estimation for finding near collisions on `SHA-1`, which is a critical step in the collision attacks. The authors provided a design of an Application-Specific Instruction-set Processor (ASIP), named Cracken, which executes specific parts of the attack. It was estimated that to execute the free-start collision and real collision attacks from [16], the attacks will take 46 and 65 days and cost 15 and 121 Million Euros respectively. At Eurocrypt 2019, Leurent and Peyrin [13] provided a chosen-prefix attacks which uses two parts: first a birthday search to reach an acceptable set of differences in the chaining variable, and then a differential cryptanalysis part that successively generate near-collision blocks to eventually reach the final collision. The attack was implemented on GPUs and a first chosen-prefix collision was published in January 2020 [14].

2.2 Birthday Search in Practice.

The efficient design of a collision search algorithm is not a trivial task, especially if the attacker wants to use parallelization over a set of computing machines. This issue is discussed in details in [21]. The collision search problem can be treated as a graph search problem, where the attacker is looking for two edges with the same endpoint but with different starting points. Pollard's rho method [15] helps finding a collision in the functional graph with a small memory requirement. The underlying idea is to start at any vertex and perform a random walk in the graph until a cycle is found. Unless the attacker is unlucky to have chosen a starting point that is part of the cycle, he ends up with a graph that resembles the Greek letter ρ and the collision is detected. Unfortunately, this method is not efficiently parallelizable, as it provides only $\mathcal{O}(\sqrt{m})$ speed-up when m cores are used. In [21], the authors proposed a method to achieve $\mathcal{O}(m)$ speed-up, using limited memory and communication requirements. This algorithm leads to very efficient parallel implementations, and is the basis for our study.

However, in the chosen-prefix collision attack against `SHA-1`, it is not applied directly to the compression function of `SHA-1`, but to a helper function. Let IV_i represent a chaining value to the compression function (reached after processing a prefix), x a message block, and $H(IV_i, x)$ the application of the `SHA-1` compression function. The goal of the birthday phase of CPC attack is to find many solutions x_1 and x_2 such that $L(H(IV_1, x_1)) = L(H(IV_2, x_2))$, where $L(x)$ is a linear function applied to a word x, in order to select some of the output bits of the compression function. The helper function is defined as:

$$f(x) = \begin{cases} L(H(IV_1, x)), & \text{if } x = 1 \pmod 2 \\ L(H(IV_2, x)), & \text{otherwise.} \end{cases} \tag{1}$$

When a collision $f(x_1) = f(x_2)$ is found, we have $x_1 \neq x_2 \pmod 2$ with probability one half, and in this case we obtain $L(H(IV_1, x_1)) = L(H(IV_2, x_2))$.

2.3 Differential Cryptanalysis.

In this section we briefly describe the algorithms involved in the second part of the chosen-prefix collision attack: the generation of successive near-collision blocks to reach the final collision. The details of this differential attack can be found in [23, 16, 10, 18, 17, 13, 14]. For each new near-collision block, the attacker has to go through three main steps:

1. Preparing a fully defined differential path for the `SHA-1` compression function (in particular a non-linear part has to be generated for the first few steps of the `SHA-1` compression function)
2. Find base solutions for the first few steps of this differential path (a base solution is simply two messages inputs that verify the planned differential path in the internal state up to the starting step of the neutral bits).
3. Expand those solutions into many solutions using what is known as neutral bits (in order to amortize the cost of the base solution), and check whether any of these solutions verify the differential path until the output of the compression function.

A neutral bit for a step i is a bit (or a combination of bits) of the message such that when its value is flipped on a base solution valid until step i, the differential path is still satisfied with high probability until step i. Most of the time, a neutral bit is a single bit, but it can sometimes be composed of a combination of bits. A neutral bit for a step i allows to amortize the cost of finding a solution to the differential path until step i.

The hardware cluster we consider consists of one master node and many slave nodes. The master builds a proper differential path for the compression function steps, based on the incoming chaining values, and generates base solutions based on this path. The slave is then required to expand these base solutions into a wider set of potential solutions and find out which of them satisfy the differential path until a certain step r (we selected $r = 40$ for ASIC for implementation efficiency purposes, but we remark that $r = 61$ was selected for GPU even though it does not have much impact) in the `SHA-1` compression function. The master then aggregates all the solutions that are valid up to step r and exhaustively search for solutions that are valid up to step 80. This is repeated several times until a valid solution for the differential path is found. Consequently, we define a slave as a dedicated core that is responsible for extending a base solution found by the master into a set of potential solutions by traversing the tree of solutions defined by the neutral bits.

Unfortunately, this attack is not hardware-friendly and needs a lot of control logic. The master has to send to the slave:

1. A base solution, which consists of two message blocks M_1 and M_2.
2. A set $[DP]$ of differential specifications for the slave to check conformance.
3. A group of neutral bit sets N_i, where the neutral bits in N_i are supposed to be neutral up to step i.

Combining a base solution (M_1, M_2) valid at step i and the set N_i, we get about $2^{|N_i|}$ new solutions that are valid up to step i, simply by trying all the possible combinations of the neutral bits in the set. In a naive approach, each of these partial solutions is expended to $2^{|N_{i+1}|}$ by applying combinations of the next set. Eventually, we would end up with $2^{\sum_i N_i}$ partial solutions, organised in a tree as shown in Figure 1. However, the neutral bits N_{i+1} are defined such that they don't impact the path up to step $i+1$. Therefore, if the partial solution does *not* satisfy the conditions at step $i+1$, there is no need to apply the neutral bits N_{i+1}, and we can instead cut the corresponding branch from the tree. Indeed, there is a certain probability that a solution valid at step i will be valid at step $i+1$, according to the `SHA-1` differential path selected. With the parameters used in `SHA-1` collision attacks, most subtrees fail.

We can generate the partial solutions using a graph search algorithm to start navigating the tree from its root, and neglect complete subtrees that are failing. In this paper we choose Depth-First Search (DFS) graph search, with some modifications to suit our specific problems, in order to satisfy our assumptions for the cryptanalytic algorithm, as DFS has low memory requirements.

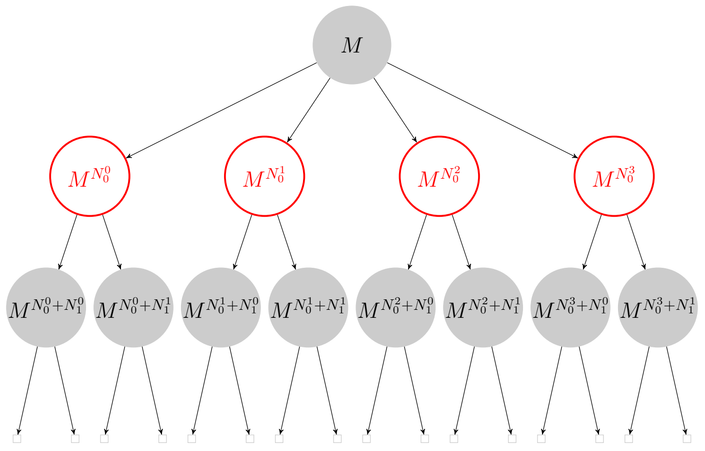


Fig. 1: Building partial solutions with neutral bits

Our attack scenarios. In this paper we consider three attack scenarios:

1. A plain 2^{64} birthday search: a generic birthday attack against a 128-bit hash function, constructed by selecting only 128 bits out of the 160 output bits of the SHA-1 compression function.
2. A plain 2^{80} birthday search: a generic birthday search over the full space of the SHA-1 compression function.
3. The chosen-prefix collision attack on SHA-1 from Leurent and Peyrin [13, 14].

These three scenarios cover two generic attacks against two security levels used in practice and one cryptanalytic attack.

3 Hardware Birthday Cluster

In this section, we describe the hardware core that handles the birthday attack. First, we define the nodes used in the proposed cluster. Then, we describe the design of the slave nodes and the communication requirements.

3.1 Cluster Nodes

The cluster used to apply the parallel birthday search attack consists of two types of nodes:

1. *Master*: a software-based CPU that manages the attack from high level and performs some jobs including choosing the initial prefixes, distributing the attack loads among slaves, sorting of the outputs and identifying colliding traces.
2. *Birthday Slaves*: dedicated cores that can perform different parts of the parallel birthday search. Specifically, it compute traces in the functional graph of the function in question, and once the master has identified colliding traces, the core also can locate the exact collision in these traces.

3.2 Hardware Design of Birthday Slaves

The design of the proposed birthday slave is shown in Figure 2. It's main role is to iterate the helper function of Equation 1. It consists of a reconfigurable ROM, where the initial trace value x_0, IV_1 and IV_2 are loaded, a logic SHA-1 core which performs the step function of SHA-1, a comparator to compute $L(x)$, $x \pmod 2$ and check whether a given x is a distinguished point (see [21]) or not, a memory to store distinguished points and a control unit to handle the communications with the master, and measure the lengths of different traces.

In order to estimate the cost of the proposed core, the area and speed are compared to a single, step-based SHA-1 core, which is a standard practice in estimating the cost of SHA-1 cryptanalytic attacks. We have implemented a full SHA-1 core and it has an area of 6.2 KGE and 0.21 ns critical path. The implementation of the core in Figure 2 using a step-based SHA-1 core requires

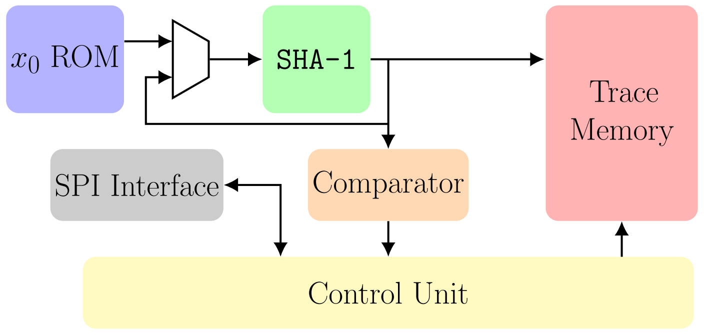


Fig. 2: Birthday slave for the parallel collision algorithm

at least twice this area. Moreover, its critical path is dominated by the memory and counters in the control logic. Besides, it is not expected that a huge ASIC cluster will run at a speed higher than 1 GHz, due to the power consumption. Hence, in order to regain the efficiency lost due to the extra control logic and memories, it is a good approach to try to use this logic with as many `SHA-1` steps as possible. Given these experiments and the huge cost of the control overhead, we increase the efficiency by cascading 4 `SHA-1` steps instead of one in the `SHA-1` core. This makes the critical path around 1ns, but a full `SHA-1` computations takes only 20 cycles instead of 80, and the overhead 25% instead of 100%.

4 Verification

We have verified the attack by finding collisions on a small number of bits using functional simulation of the hardware implementations. Specifically, we found collisions on $20 \sim 330$ bits of the output. We have also generated traces for larger number of bits and compared them to traces generated using software implementations.

5 Hardware Differential Attack Cluster Design

In this section we discuss the challenges and different trade-offs when implementing the neutral bit search algorithm in ASIC and give a description of the circuit. The cluster architecture uses 3 types of nodes: master nodes, birthday slaves (BC), and neutral bit slaves (NB).

5.1 Neutral Bits

One of the trade-offs when implementing the attack is whether to consider neutral bits as only single-bits or to use the more general sets of multi-bits. The first approach leads to a very small circuit, but it strongly limits the number of usable neutral bits. This increases the overall work load, as more base solutions need to be generated, and more time is spend applying neutral bits. The second approach is more complex, because multi-bit neutral bits must be represented by a bit-vector. However, the single-bit neutral bits are not sufficient to implement an efficient attack, and we have to use the second option:

1. Our simulation results show that the success probability of single-bits is very low. Hence, any gain achieved by using them is offset due to the huge work load and high communication cost between the master and slave.
2. In order to achieve significant results, multi-bits are inevitable. In particular, *boomerangs* [9] (which can be seen as multi-bit neutral bits with extra conditions to reach a later step) are crucial cryptanalytic tools for a low-complexity attack against `SHA-1`. Hence, avoiding multi-bits can lead to a drastic loss in terms of attack efficiency.

5.2 Storage

Each multi-bit neutral bit is represented by a 512-bit vector, which indicates the location of the involved bits in the message block (a `SHA-1` message block is indeed 512-bit long). However, we noticed that almost all the neutral bits involve bits only in the last 6 32-bit words of the message block. Therefore, we reduced the representation to only 192 bits. Yet, since the original chosen-prefix collision attack against `SHA-1` uses ~ 60 neutral bits, including boomerangs, this requires a representation of $\sim 11,520$ bits. Besides, the last few levels of the tree requires 320 bits per neutral bits as the boomerangs can be located as early as step 6. In addition, for each level of the tree search we need a counter to trace which node we are testing. The tree used in the attack has ~ 10 levels, and our experiments show that the maximum number of neutral bits in one level is ~ 26 bits. Hence, the overall size of the counters is ~ 260 bits. In order to design the circuit that handles this tree search algorithm, we tried out four different approaches:

1. Generic approach: we assume that each tree level can have ~ 28 neutral bits (slightly higher than our experiments for tolerance). Also, assume that these levels can be related to any step of the `SHA-1` compression function between 10 and 26, i.e. 16 possible steps. In total, this requires $\sim 63,670$ memory locations (Flip-Flops).
2. Statistical approach: from the software experiments and simulations, we identified an average number of neutral bits per level. In the design, we use the maximum number of neutral bits we observe for each level (in addition to two extra bits for tolerance). We observed that only the first few levels require such a huge storage, while the later levels usually have $3 \sim 7$ bits per level. In addition, boomerangs are usually $3 \sim 4$ per level. This reduces the memory requirement by about 50%. However, it remains a huge requirement.
3. Configurable approach: our experiments showed that not only the number of neutral bits per level can be predicted, but also the values of these bits. In other words, very few bits have different values for different blocks. Hence, we

can fix each neutral bit to two or three choices and use flip-flops to configure which choice is selected during execution. This reduces the cost significantly. However, the cost is still high as a multiplexer has an area only $\sim 50\%$ of a flip-flop. Besides, we still need flip-flops for configuring these multiplexers.

4. Another approach is to reduce the cost by fixing the the neutral bit values to a set of statistically dominant values. Indeed, [14] reports using the same neutral bits for each near-collision bloc. This eliminates the need to store the neutral-bit reference values.

At the end, we chose the third approach, since our analysis shows that it captures the reality, while allowing some level of freedom for the attacker to adjust the attack parameters after fabrication.

5.3 Architecture

Figure 3 shows the architecture of the neutral-bit slave. It consists of a register file to store the differential path for comparison, a configurable ROM to store the base solution, a unit to enumerate the different neutral bit patterns and maintains the tree level for the graph search algorithm, and the `SHA-1` step logic.

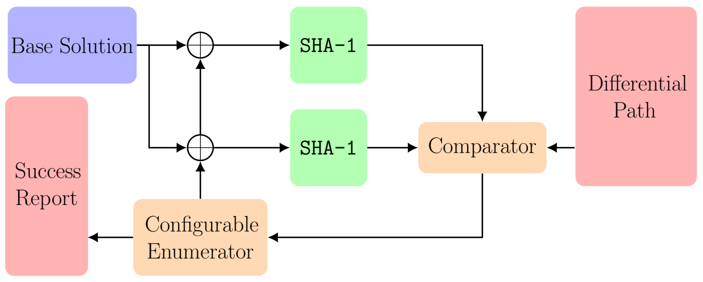


Fig. 3: Neutral-bit slave hardware architecture

6 Chip Design

In this section, we describe our process for simulating the proposed chips and the results in terms of power, area and performance for each.

6.1 Chip Architecture

A challenge when designing this cluster is the communication overhead between the master and the slaves. A 100MHz SPI bus interface is used as a one-to-one communication interface with the attack server. A set of ASICs can also be daisy-chained, thanks to this interface, in such a way to lower the number of interconnects with the master. It provides enough bandwidth to handle the data exchanges between the BD/NB slave cluster and the attack server. The CU (Control Unit) is responsible for dispatching the 32-bit de-serialized packets sent by the attack sever to configure the BD/NB slaves. It is also responsible for daisy-chaining and demultiplexing the output traces of the different BD/NB slaves to the SPI bus interface before the serialization. Each ASIC also outputs an asynchronous interrupt signal. The interrupt signal is 1 when at least one BD/NB slave is done, and an output trace is available. Those interrupt signals are managed by a set of ZYNQ board cluster interfaces.

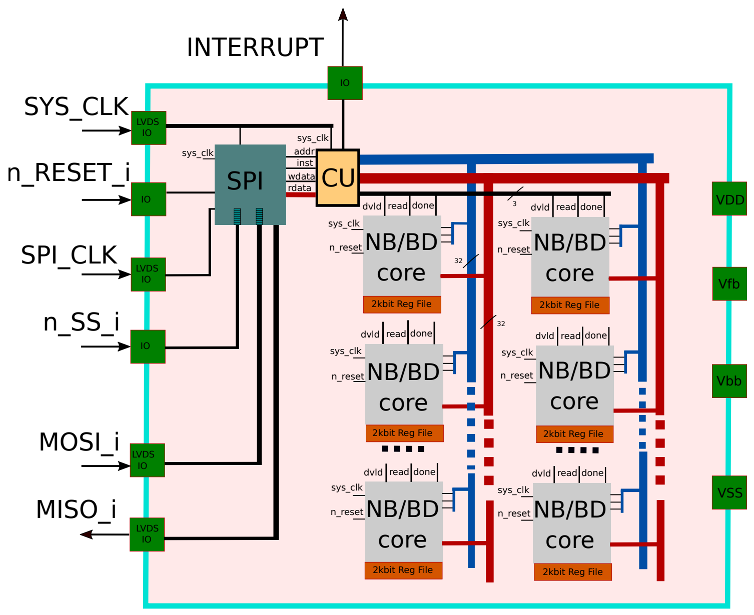


Fig. 4: System architecture of the ASIC cluster chip

6.2 ASIC Fabrication and Running Cost

Estimating the cost of fabricating and running an ASIC cluster can be challenging as many parameters are confidential to the fabs. In order to estimate the costs of the attacks considered, we developed a methodology based on the information available publicly. We considered the FD-SOI 28nm technology from ST-Microelectronics. For small scale academic projects, the price of a small batch of up to 100 die, the fabrication cost in US $ can be estimated by:

$$p_{100} = \begin{cases} 125400 + (A - 12) * 7700, & \text{if } A > 12mm^2 \\ 20900 + (A - 2) * 9900, & \text{if } 2mm^2 \leq A \leq 12mm^2 \end{cases}$$

where A is the die area in mm^2 and p_{100} is the price of the first 100 die in US Dollars (USD). For small scale projects with more than 100 die, the price for a lot of 100 extra die is between 21,120 and 38,500 USD depending on the die area and the number of reticules in a wafer. MPW runs uses Multi Layer Reticule technic to reduce the overall cost of the mask and additional dies. For our purposes, we consider a small scale project to be a project with at most 25 wafers [20] . For large scale projects, a market study published at the FDSOI Forum in 2018 showed that the die manufacturing cost per 40 mm^2 is 0.9 USD for the 28nm technology [8]. Hence, our methodology for estimating the costs consists of the three parts we explained. In reality, a more accurate methodology is probably available for the fabs to fill in the gaps. However, we believe that the overall cost will be in the same range.

On top of the fabrication cost, we need to consider the running cost of the ASIC cluster, which includes the energy consumption and cooling. We have performed post-layout extraction and simulation in order to estimate the power consumption of the different chips. In order to simplify the cost analysis, we use a figure of 18 cents/KWh, which is higher than the electricity consumption price in most countries [5]. Hence, we only consider the energy consumption of the chips and not the cooling cost or other factors that will be added after fabrication. The performances and power result are provided in Table 1.

6.3 Results

Two different architectures of `SHA-1` crackers are compared here. The first architecture is based on 2 separate ASIC slaves that handle the two parts of the attack, *i.e.,* the birthday search (BD) and the neutral bits part (NB). The two phases are performed sequentially. Figure 5 depicts the overall cost required to build the machine and find the first chosen-prefix collision depending on the time ratio between the two phases. For ASIC, the overall minimum cost is not perfectly at 50% ratio. Hence, we consider a two-stage pipeline architecture at the cost of slightly more hardware to balance the birthday and neutral-bit parts.

Our birthday (BD) core uses 16927.1 gate equivalents (GEs) per `SHA-1` rounds., while our neutral-bit core (NB) uses 170442.7 GEs. Our best implementation is a 4-round `SHA-1` unrolled compression function that can be clocked at 900 MHz at V_{core}=0.92V and V_{fbb}=0V. Using body biasing and LVT transistors for the critical path, we can further decrease the threshold voltage and increase the running frequency. With V_{fbb}=+2.0V we can increase the running frequency of our fastest core by 40%, reaching 1262 MHz with a 2% increase in dissipated power. The chip can be further over-clocked by increasing V_{core} but at the cost of a quadratic increase in the dissipated power, so a more costly cooling system. The results of our implementations are shown in Table 1. As shown in

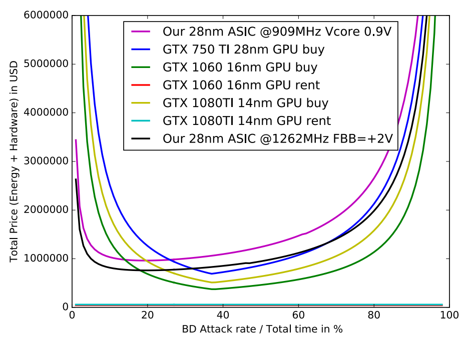


Fig. 5: Impact of the BD/NB time ratio on the cost

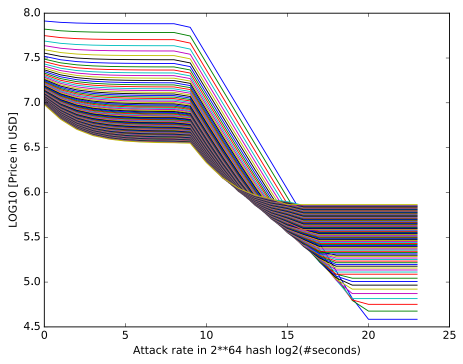


Fig. 6: Impact of the die size and latency on the HW cost (4 to 100 mm^2 28nm FD-SOI). The top left line in blue represents 4 mm^2 and the bottom left is 100 mm^2.

Figure 21, a BD slave contains up to $\sim$ 15 BD cores per mm^2 while an NB slave contains $\sim$ 1.5 NB cores per mm^2.

Version	**900 MHz** V_{fbb}**=0V**		**1262 MHz** V_{fbb}**=+2V**	
	BD	NB	BD	NB
Power (in mW)	71.1	289	72.6	294
CP delay (in ps)	1110	1110	792	792
Area (in mm^2)	0.0650	0.6545	0.0650	0.6545

Table 1: ASIC implementation performances for 2 corners cases : high performance at 900 MHz and high performance with FBB at 1262 MHz.

In our study, the overall cost is calculated without the cooling and infrastructure. Note that as shown in Figure 6, the total cost required to build an ASIC-based cracker greatly depends on the die size. This is due to the fact that the initial cost is predominant when the die size is large. The overall hardware cost tends to the same for any die size when the attack is fast.

6.4 Attack Rates and Execution Time

As shown in Table 2, a single NB slave of $16mm^2$ contains up to 24 NB cores and can generate up to 976 solutions up to step 40 of `SHA-1` per second. Each solution A_{40} requires 31 Million cycles, on average. A single BD slave of $16mm^2$ contains up to 245 BD cores and provides a hash rate of 20.6 GH/s for the fastest version of our design. As a comparison, as shown in Table 3 and taken from [14], a single GTX 1060 GPU provides a hash rate of 4.0GH/s and can generate 2000 A_{40} solutions per second. If we take the birthday part of the attack as a reference, the neutral bit part is ten time less efficient in hardware than on GPU.

Parameter	**900 MHz**	**1262 MHz**
`SHA-1`**/core/sec**	$2^{25.8}$	$2^{26.3}$
`SHA-1`**/core/month**	$2^{47.1}$	$2^{47.6}$
`SHA-1`**/chip/month**	$2^{55.1}$	$2^{55.6}$
A_{40} **Solutions/core/sec**	$2^{4.9}$	$2^{5.3}$
A_{40} **Solutions/core/month**	$2^{26.1}$	$2^{26.7}$
A_{40} **Solutions/chip/month**	$2^{30.8}$	$2^{31.2}$

Table 2: Our best $16mm^2$ ASIC implementation performances for 2 corners

GPU	arch	Hash Rate	A_{33} rate	A_{40} rate	Price	Power	Rental
GTX 750 Ti	Maxwell	0.9GH/s	62k/s	250/s	$144	60W	
GTX 1060	Pascal	4.0GH/s	470k/s	2k/s	$300	120W	$35/month
GTX 1080 Ti	Pascal	12.8GH/s	1500k/s	6.2k/s	$1300	250W	

Table 3: `SHA-1` hash rate from hashcat for various GPU models, as well as measured rate of solutions at step 33 (A_{33}-solutions). Data taken from [14].

The second architecture is based on GPU. For GPU, it is cost-wise more interesting to take advantage of its reconfigurability to minimize the cost. Hence, we consider in our cost analysis that the chosen-prefix collision is performed serially by reusing the same GPU for the two attack phases. In Table 4, the cost of the three attack scenarios is provided. We give in this table the cost to build the ASIC- and GPU-based clusters for 3 different speeds, *i.e.,* one attack per month, one attack per day and one attack per minute. The latency corresponds to the delay to get the first collision. For instance, a two-stage ASIC-based machine able to generate one `SHA-1` collision every months, will generate the first collision in two months. A GPU-based machine generates the first collision in one month for the same attack rate. Our ASIC-based two stage pipelined architecture has twice the latency of a sequential GPU-based machine for the same attack rate.

Our benchmark (Figures 15 and 16) provides a comparison between our ASIC cluster and two of the most widely spread GPU based machines, *i.e.,* the GTX 1080TI (CMOS 14nm) and the GTX 1060 (CMOS 16nm) for different attack rates. The numbers for the GTX 750 TI (CMOS 28nm technology) are also added to the benchmark as it provides an idea of the performance obtained with a GPU based on a similar technology node as our ASIC.

Platform	**ASIC**			**GPU rent**			**GPU buy**		
Attack	64	CPC	80	64	CPC	80	64	CPC	80
Energy Cost	$776	$1.6k	$50.9M	-	-	-	$18k	$12k	$1.2B

Cluster for 1 attack per month

Latency (month)	1	2	1	1	1	1	1	1	1
Hardware Cost	$257k	$1.1M	$11M	-	-	-	$715k	$490k	$47B
First Attack Cost	$257k	$1.1M	$61.9M	$61k	$43k	$4B	$733k	$502k	$48B
Amortized Cost	$7.9k	$32.1k	$51.2M	$61k	$43k	$4B	$38k	$26k	$2.5B

Cluster for 1 attack per day

Latency (day)	1	2	1	1	1	1	1	1	1
Hardware Cost	$1.4M	$3.7M	$218M	-	-	-	$22M	$15M	$1.4T
First Attack Cost	$1.4M	$3.7M	$269M	$61k	$43k	$4B	$22M	$15M	$1.4T
Amortized Cost	$2k	$5k	$51.1M	$61k	$43k	$4B	$38k	$26k	$2.5B

Cluster for 1 attack per minute

Latency (minute)	1	2	1	1	1	1	1	1	1
Hardware Cost	$8.5M	$48M	$263B	-	-	-	$31B	$21B	$2Q
First Attack Cost	$8.5M	$48M	$263B	$61k	$43k	$4B	$31B	$21B	$2Q
Amortized Cost	$781	$1.6k	$51M	$61k	$43k	$4B	$38k	$26k	$2.5B

Table 4: Comparison of attack costs with various parameters. Costs are given in USD (k stands for thousand, M for Million, B for Billion, T for Trillion, Q for Quadrillion). Amortized cost is the cost per attack assuming that the hardware is used continuously during three years. Note that it is possible to get slightly more energy efficient platforms and implementations at the cost of more expensive hardware. We list the cheapest platform after one attack, energy included.

Note on the use of FPGAs Our ASIC design have been tested on FPGA platform. FPGA can be considered as a good alternative to ASIC thanks to its reconfigurability property. However, one of the largest FPGAs from Xilinx, namely the Virtex 7 xc7vx330t-3ffg1157 can fit only 20 instances of the Birthday core running at 135MHz in one chip. The same FPGA can fit only 16 instances of the Neutral Bit core running at 133MHz. In order to do the 2^{64} generic birthday search, we need $2^{36.6}$ FPGA-seconds, *i.e.,* in order to do it in one month we need $2^{15.3}$ FPGAs. As a single FPGA costs around 8000 USD, this attack would cost around 319 Million USD. This is more than one thousand times the cost of the same attack on ASIC and 440 times the cost on GPU, making it irrelevant for the purpose of analyzing SHA-1. Even if FPGAs can be rented, a similar factor is expected compared to renting GPUs. It is worth mentioning that FPGA-based clusters, such as COPACABANA, use cheaper FPGAs. However, they are usually used for smaller projects and will face the same challenge to scale up to the level of attacks awe are considering.

7 Cost Analysis and Comparisons

As explained throughout the paper, we have performed several experiments to identify the different implementation trade-offs for the attack scenarios we consider. In this section, we analyze the cost estimates of implementing these attacks in ASIC vs. consumer GPU. We consider three attack scenarios that fall into two categories: generic birthday attacks and differential cryptanalysis of SHA-1. Before discussing the analysis in more details, here are a few general conclusions that we reached through our experiments, which can be helpful for building future hardware crackers:

1. The cost of implementing memoryless generic attacks, such as the parallel collision search of [21], in hardware can range from 20% to 50% of the overall ASIC implementation, while the rest is dedicated to the attacked primitive, e.g. the SHA-1 hash function.
2. For iterative cryptographic algorithms, such as hash functions and block ciphers, a way to reduce the attack cost is to use unrolling. This approach is similar to using memoryless algorithms. Instead of computing one step of the function every clock cycle, we compute several steps in the same cycle. This amortizes the costs of the attack logic among several steps. For example, implementing the birthday attack using a single-step iterative SHA-1 core leads to a circuits where only 20% of the area is used by the SHA-1 logic and 80% of the area is due to the attack logic, registers and comparisons. On the other hand, using a core that computes 4 steps every clock cycles leads to a circuit with a 50%/50% ratio. While this technique may increase the critical path of the circuit and reduce the frequency, it also reduces the overall number of cycles, so the overall time to compute a single SHA-1 per core is almost constant.
3. For cryptanalytic attacks, the cost is dominated by the attack logic, which may include a huge number of comparisons, modifications and registers. These extra operations are usually different from one step to another, so they consume a huge area. Besides, the state machine of these attacks can be very costly. In such scenarios, the advantage of using ASICs becomes diminished compared to consumer GPUs, except for very high budgets, especially as the GPUs are reusable and can be rented.

7.1 2^{64} Birthday Attack

The first attack scenario we consider is attacking a hash function with 2^{64} birthday collision complexity. The hash function used is the `SHA-1` compression function reduced to only 128 output bits, as explained in Section 2. A single ASIC core is described in Section 3. The time to finish such an attack depends on the number of chips fabricated and the size of each chip. A single ASIC core running at 1262 MHz contributes $2^{26.33}$ `SHA-1` computations per second. The attack costs $2^{37.67}$ core-seconds. To reach this complexity, Figure 6 shows the price required vs. the estimated time needed to finish the attack, including the fabrication cost of chips of different sizes and the energy consumption.

To put these numbers into perspective, the NVIDIA GeForce GTX 1080 TI GPU (14nm technology) can do about $2^{33.6}$ `SHA-1` computations per second, so implementing the attack on GPU would require $2^{30.4}$ GPU-seconds. In order to implement this attack in one month, we need to buy around 550 GPUs costing around 715k USD and around 18k USD in energy. As shown in Figure 7, a GTX 1060-based machine is a bit less expensive, costing 525k USD but consuming around 28k in energy for the same job (using 1750 GPUs).

Besides, as shown in Figure 7, for any attack rate it is cheaper to buy an ASIC cluster than a GPU-based cluster. The difference reaches 1 order of magnitude from a rate of 1 attack per week. Furthermore, the ASIC-based cluster consumes 1 to 2 order of magnitude less energy than any GPU-based solution. As shown in Table 4, the minimum cost in energy per attack on ASIC is as low as 776 USD. An ASIC-based cracker able to generate one collision per month would cost 257k USD. For an attack rate of 1 attack per minute, it would cost 8.5 million USD.

An alternative option is to rent the GPUs. This would cost around \$61k per attack, assuming a rental price of \$209/month for a machine with 6 GTX 1060 GPUs. This makes the GPU rental very competitive for a single attack, around 4 times cheaper than an ASIC cluster. However, the ASIC cluster quickly become much more cost effective when the attack is repeated (see Figure 9).

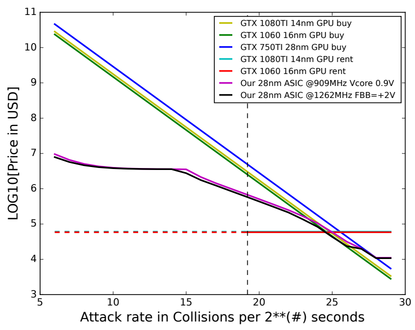


Fig. 7: 2^{64} BD machine price for different attack rates: ASIC vs GPU

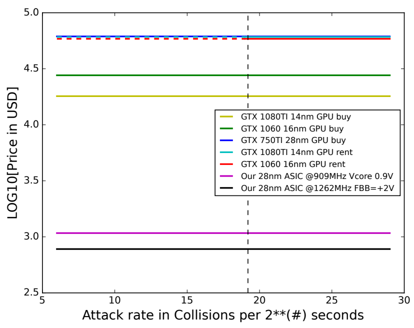


Fig. 8: Energy cost per 2^{64} BD attack: ASIC vs GPU

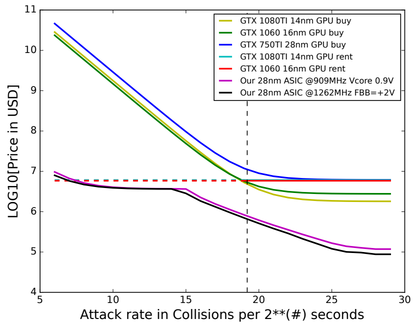


Fig. 9: Total cost (HW+E) for 100 2^{64} BD attack at a given attack rate: ASIC vs GPU

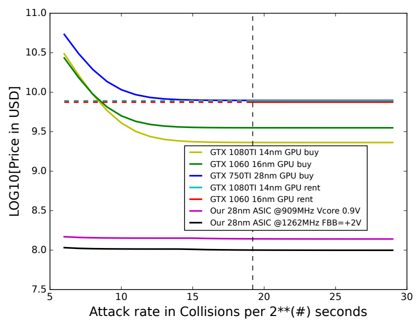


Fig. 10: Total cost (HW+E) for 100k 2^{64} BD attack at a given attack rate: ASIC vs GPU

7.2 2^{80} Birthday Attack

In this section, we look at the cost of implementing a generic birthday collision search for the full `SHA-1` output, which requires around 2^{80} `SHA-1` computations. The algorithm is the same as the previous attack, except that we use the full output of the `SHA-1` compression function. Since a single ASIC core performs $2^{26.33}$ `SHA-1` computations per second, the birthday collision search costs $2^{53.67}$ core-seconds, or around 454 million years on a single core. Fortunately, for a powerful attacker with enough money, the cost for producing ASICs grows slowly for large number of chips. The fabrication cost of a hardware cluster to perform the attack in one month costs only 11 million USD, as opposed to around 34 billion USD for GTX 1060. Hence in this case, for any attack rate as shown in Graphs 13 and 14 the only realistic option is to build an ASIC cluster.

Running the attack costs around 50.9 million USD in energy, which matches the order of magnitude estimated from the bitcoin network: the network currently computes about $2^{70.2}$ `SHA-256` every ten minutes, for a reward of 12.5 bitcoin, or roughly $85k at the time of writing. This would price a 2^{80} computation at 75 million USD.

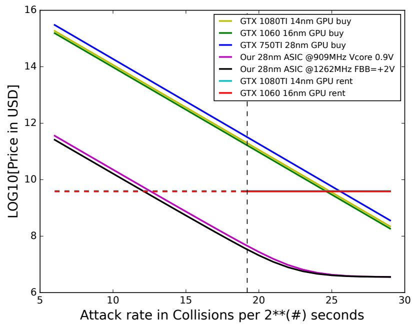


Fig. 11: 2^{80} BD machine price for different attack rates: ASIC vs GPU

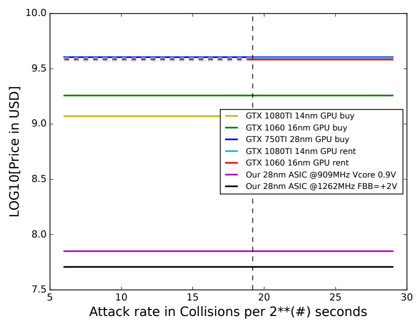


Fig. 12: Energy cost per 2^{80} BD attack: ASIC vs GPU

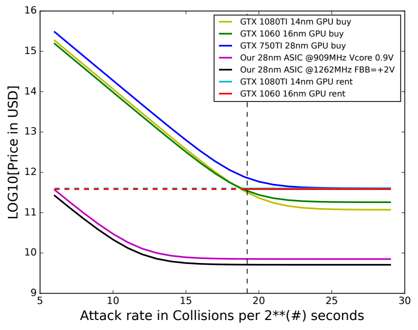


Fig. 13: Total cost (HW+E) for 100 2^{80} BD attack at a given attack rate: ASIC vs GPU

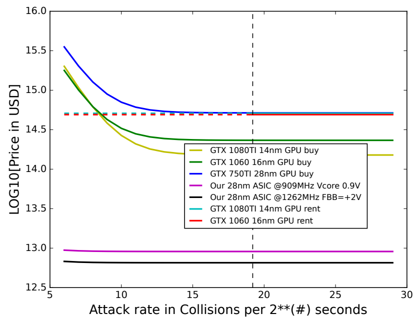


Fig. 14: Total cost (HW+E) for 100k 2^{80} attack at a given attack rate: ASIC vs GPU

7.3 Chosen Prefix Differential Collision Attack

The chosen-prefix collision attack proposed by Leurent and Peyrin [14] consists of two main parts: a birthday search attack, and a differential collision attack. The authors provide different trade-offs between the complexity of the two parts. In their paper, the number of solutions required for the neutral bits up to step 33 is provided. This number of solutions corresponds to the number of solutions required to get a valid solution with high probability. Step 33 is chosen because there is a zero difference at this state, so there is a single path at this step, and solutions are generated fast enough to measure the rate easily. This configuration requires to generate about $2^{62.05}$ `SHA-1` computations for the birthday part and $2^{49.78}$ solutions up to step 33. In this paper, it is cost-wise more interesting for ASIC to generate solutions for the neutral bits up to step A_{40}.There is a factor $2^{7.91}$ difference in the number of solutions to generate between step A_{33} and step A_{40}. Hence a chosen-prefix collision requires to generate $2^{41.87}$ solutions. Table 3 provides the hash rates and solution rates numbers used in our estimate for the cost on GPU. This gives 38 GPU-years for the birthday, and 65 years for the neutral bits. The estimated cost per attack using GTX 1060 GPU, assuming 209 USD per month for 6 GPU is about 43k USD. The cost of running the attack in GPU is dominated by the energy consumption. ASIC is much more energy efficient, as shown in Figure 16. It can be up to 2 order of magnitude less than using common consumer GPU. As shown in Figure 15, ASIC-based `SHA-1` cracker that generate one collision per month, costs about 1.1 million USD, about the same as the cheapest GPU-based cracker from our benchmark. However, a single attack on GPU costs about 19000 USD in energy. Hence from 100 attacks as shown in Figure 17 and 18 as well as for attack rates greater than 1 attack per week, an ASIC-based `SHA-1` cracker is the only realistic option.

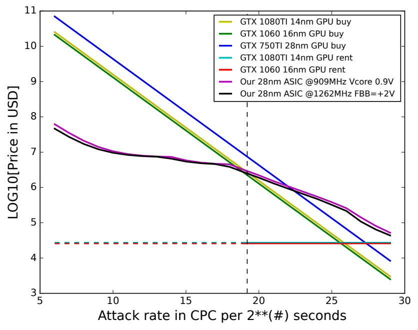


Fig. 15: CPC machine price for different attack rates: ASIC vs GPU

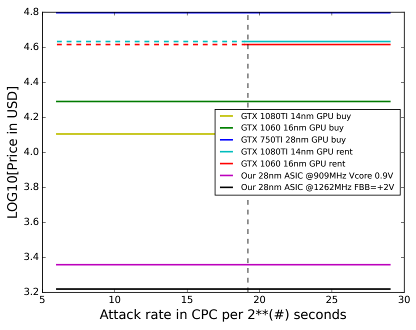


Fig. 16: Energy cost per CPC attack: ASIC vs GPU

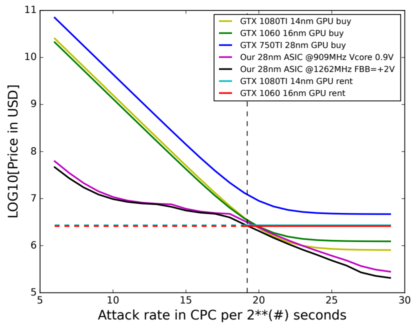


Fig. 17: Total cost (HW+E) for 100 CPC attack at a given attack rate: ASIC vs GPU

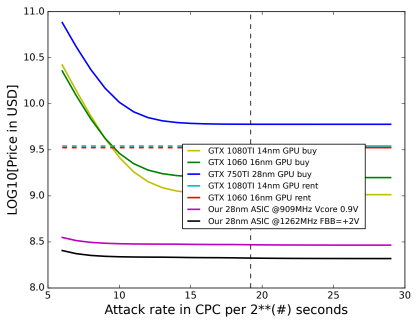


Fig. 18: Total cost (HW+E) for 100k CPC attack at a given attack rate: ASIC vs GPU

7.4 Limitations

While we did our best to estimate the price of the attacks as accurately as possible, our figures should only be considered as orders of magnitude because the pricing of hardware and energy can vary significantly. ASIC pricing is not completely public, and energy prices depend on the country. Moreover, our estimate only include hardware cost and energy, neglecting other operating costs such as cooling and servers to control the cluster (however, the energy price we use is somewhat high, so it can be considered as including some operating costs).

Another caveat is that we only consider the computation part of the attacks. In reality, there is some need for communication between the nodes, and some steps of the attacks must be done sequentially. Concretely, the generic birthday attacks must sort the data after computing all the chains, and the CPC attack must compute several near-collision blocks sequentially. This will likely add some latency to the computation, and running the attack in one minute will be be a huge challenge, even when the required computational power is available.

8 Conclusion

Our paper provides a precise comparison between ASIC-based and GPU-based solutions for cryptanalysis, with a case study on generic birthday search and a case study on the recent chosen-prefix collision on `SHA-1`. For the former, we show that generic birthday attacks can be performed very easily with ASICs against a 128-bit hash function, and that even a 160-bit hash function would not stand against a huge, yet potentially affordable, ASIC cluster. For the latter, we created two independent ASICs that handle the two parts separately. Our comparisons with GPU-based solutions show a clear advantage of ASIC-based solutions. In particular, we remark that the chosen-prefix collisions for `SHA-1` can be generated in under a minute, with an ASIC cluster that costs a few dozen Millions dollars. Such ability would allow an attacker to apply the SLOTH attack [2] on TLS or SSH connections using `SHA-1`.

In the introduction, we posed three research questions; the first question is related to the cost of attacks on `SHA-1`. Our study showed that ASIC is clearly the best choice for very high complexities attacks, or for attacks that need to be performed in a short amount of time. However, for proof-of- concept or cryptographic research in general, where complexities of 2^{64} or less can be computed in a month or so, renting a set of GPUs seems to be the best solution. If the attack needs to be repeated multiple times, or if the speed of the attack is critical, then the initial hardware cost might be amortized and the energy cost per attack might become important. We note that the energy cost will be very high on GPU compared to a dedicated ASIC solution. For a chosen-prefix collision on `SHA-1`, the energy cost per attack for our speed-optimized ASIC is 1.6k USD. The best GPU based solution from our benchmarks consumes about 12k USD per attack. Hence, the cost of the ASIC-based solution is amortized. Furthermore, when the CPC attack rate becomes higher than 100 attacks per month, the ASIC solution is cheaper than any GPU-based solution in our benchmarks. In this case, the cost of the GPU rent is prohibitive and the ASIC is the only realistic threat.

In the second question, we target the comparison between generic attacks and cryptanalytic attacks for similar theoretical level of numeric complexity. In our study, we show that for a similar level of $\sim 2^{64}$ computations, it is $\sim 75 \sim 82\%$ cheaper to implement a generic birthday search, compared to the differential CPC attack on `SHA-1`. This means that for these two attacks, the generic attack has an advantage of $5\times$. One can study more advanced brute force attacks, such as the biclique technique, in order to compare with generic ones. A preliminary study have been published on this topic [3], where the authors compare the cost of building a brute force machine for AES vs. implementing the biclique. They find that the cost of implementing the biclique attack is cheaper than brute force, but slightly worse than what is theoretically expected. However, they only consider one extreme architecture for the brute force machine, and we believe that this can be optimized bringing the cost of brute force down to lower than the biclique attack. However, we leave this hypothesis for future work.

Last but not least, the third question is whether the 80-bit security level is still adequate for practical use in less demanding applications. Our study is a warning, showing that not only `SHA-1` is indeed practically fully broken, but also that search-based and memory-less generic attacks with complexity $\leq 2^{80}$ are within practical reach.

References

1. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The simon and speck lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference. DAC '15, Association for Computing Machinery, New York, NY, USA (2015), https://doi.org/10.1145/2744769.2747946
2. Bhargavan, K., Leurent, G.: Transcript collision attacks: Breaking authentication in TLS, IKE and SSH. In: NDSS 2016. The Internet Society (Feb 2016)
3. Bogdanov, A., Kavun, E., Paar, C., Rechberger, C., Yalcin, T.: Better than brute-force—optimized hardware architecture for efficient biclique attacks on aes-128. In: ECRYPT Workshop, SHARCS-Special Purpose Hardware for Attacking Cryptographic Systems (2012)
4. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique cryptanalysis of the full aes. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. pp. 344–371. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
5. globalpetrolprices.com: https://www.globalpetrolprices.com
6. Güneysu, T., Kasper, T., Novotnỳ, M., Paar, C., Rupp, A.: Cryptanalysis with COPACOBANA. IEEE Transactions on Computers 57(11), 1498–1513 (2008)
7. Hassan, M., Khalid, A., Chattopadhyay, A., Rechberger, C., Güneysu, T., Paar, C.: New asic/fpga cost estimates for sha-1 collisions. In: Digital System Design (DSD), 2015 Euromicro Conference on. pp. 669–676. IEEE (2015)
8. Jones, H.: FINFET AND FD SOI:MARKET AND COST ANALYSIS. FDSOI Forum 2018. http://soiconsortium.eu/wp-content/uploads/2018/08/MS-FDSOI9.1818-cr.pdf (2018)
9. Joux, A., Peyrin, T.: Hash Functions and the (Amplified) Boomerang Attack. In: Menezes, A. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 4622, pp. 244–263. Springer (2007)
10. Karpman, P., Peyrin, T., Stevens, M.: Practical free-start collision attacks on 76-step sha-1. In: Annual Cryptology Conference. pp. 623–642. Springer (2015)
11. Khairallah, M., Najm, Z., Chattopadhyay, A., Peyrin, T.: Crack me if you can: Hardware acceleration bridging the gap between practical and theoretical cryptanalysis?: A survey. In: Proceedings of the 18th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation. pp. 167–172. SAMOS '18, ACM, New York, NY, USA (2018), http://doi.acm.org/10.1145/3229631.3239366
12. Kumar, S., Paar, C., Pelzl, J., Pfeiffer, G., Schimmler, M.: Breaking ciphers with COPACOBANA–a cost-optimized parallel code breaker. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 101–118. Springer (2006)
13. Leurent, G., Peyrin, T.: From collisions to chosen-prefix collisions application to full sha-1. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 527–555. Springer (2019)

14. Leurent, G., Peyrin, T.: Sha-1 is a shambles - first chosen-prefix collision on sha-1 and application to the pgp web of trust. Cryptology ePrint Archive, Report 2020/014 (2020), https://eprint.iacr.org/2020/014
15. Pollard, J.M.: Monte carlo methods for index computation. Mathematics of computation 32(143), 918–924 (1978)
16. Stevens, M.: New collision attacks on sha-1 based on optimal joint local-collision analysis. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 245–261. Springer (2013)
17. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y.: The first collision for full sha-1. In: Annual International Cryptology Conference. pp. 570–596. Springer (2017)
18. Stevens, M., Karpman, P., Peyrin, T.: Freestart collision for full sha-1. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 459–483. Springer (2016)
19. Tom Brostöm: Lightweight Trusted Computing. https://www.nist.gov/news-events/events/2019/11/lightweight-cryptography-workshop-2019 (2019)
20. Tu, Y.M., Lu, C.W.: The influence of lot size on production performance in wafer fabrication based on simulation. In: Procedia Engineering. vol. 174, pp. 135 – 144 (2017), http://www.sciencedirect.com/science/article/pii/S1877705817301807, 13th Global Congress on Manufacturing and Management Zhengzhou, China 28-30 November, 2016
21. Van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. Journal of cryptology 12(1), 1–28 (1999)
22. Vivek, D., Narendra, S., Haycock, M., Govindarajulu, V., Erraguntla, V., Wilson, H., Vangal, S., Pangal, A., Seligman, E., Nair, R., et al.: 1. 1 v 1 ghz communications router with on-chip body bias in 150 nm cmos. In: DIG TECH PAP IEEE INT SOLID STATE CIRCUITS CONF. pp. 270-271+ 466+ 263. 2002 (2002)
23. Wang, X., Yao, A.C., Yao, F.: Cryptanalysis on sha-1. In: Cryptographic Hash Workshop hosted by NIST (2005)
24. Wiki, E.: Ethash. GitHub Ethereum Wiki. https://github.com/ethereum/wiki/wiki/Ethash (2017)
25. X16R: https://en.bitcoinwiki.org/wiki/X16R

A Verification

In order to verify the functionality of the ASIC implementation of the neutral bit algorithm, we have implemented it also in software and we have checked that the outputs and the intermediate values from the two implementations match. This process is described in more detail next.

In Algorithm 1, we give a description of the neutral bit search algorithm in pseudo-code as implemented in software. It follows the high-level description given in Section 2. In more detail, all neutral bits are provided as an input to the algorithm in the form of 512-bit masks. Each mask selects one or more bits from the original 16 byte message, resulting resp. in a single neutral bit or a set of neutral bits (multibits). Note that the latter also include the boomerangs.

The set of all neutral bit (NB) masks is partitioned in n_k subsets, such that all NB from the n_k-th subset are neutral up to step k inclusive. Such order allows to apply the neutral bits recursively in a breadth-first manner from n_k to n_{k+1}. If a neutral (multi)bit from subset n_k fails (i.e. results in a message that does not follow the differential path), then the search does not explore any neutral bits from subset n_{k+1} for the particular failing combination at n_k. In this way failing branches of the search tree (Fig. 1) are abandoned early during the search.

An equivalent implementation as the one described above was developed also in hardware. The verification of the equivalence of the two implementations was performed as follows. The execution of the software program is stored in the form of a trace containing the following information: step k, neutral bit mask m_k from subset n_k applied at step k and a list of all differential pairs of modified internal states (A_i, A'_i) at steps $i = k, k+1, \ldots, k'$, where step k' is the step at which the pair of internal states $(A_{k'}, A'_{k'})$ has failed to follow the differential path. The hardware implementation takes as input the trace produced by the software together with a list of all neutral bit masks and verifies that the values of all internal state pairs (A_k, A'_k) match the ones produced by the hardware and fail at the same step k' given in the software trace.

B Chip design

Early studies [22] demonstrated the effectiveness of body biasing in reducing leakage, improving performances, and worst case power consumption. This is an interesting feature for high performance computing, and practical cryptanalysis. Indeed this feature allows to get the best possible performance at given desired energy point. For a single targeted attack, the energy cost is not the critical factor in the overall attack cost. However it has a direct impact on the complexity of the cooling infrastructure when the attack complexity gets high. Moreover, for multiple attacks scenario, the energy becomes a critical factor.

The STMicroelectronics CMOS FD-SOI 28nm technology has been chosen for our simulations for its very good *power* $\times$ *performance* $\times$ *cost* product capability compared to the its earlier predecessors CMOS 40nm and 65nm, its availability in our testing environment and the availability of enough public information

Algorithm 1 Apply neutral bits.

```
Input: —
    $i$:  Message step $i \geq 13$ (correspond to internal state step $i + 1$)
    $P$:  Path (composed of internal state $A$ and expanded message $W$)
    $S_i$:  Base solution with fully instantiated first 16 message words $W_0, \ldots, W_{15}$
    and following path $P$ up to and including message step $i \geq 13$
    $N_k[0 \ldots n_k - 1]$:  an array of $n_k$ 512-bit masks. Each mask is a single bit or a
    multibit neutral bit (NBit) combination that is neutral up to and including message
    step $k : 13 \leq k \leq 18$. (Note: a multibit is a collection of several bits that
    have to be flipped together)
Output: —
    $S_j$:  Base solution following path $P$ up to and including message step $j > i$
 1: apply_neutral_multibits($i, S_i, P$)
 2: // If no more NBits to assign, keep computing step by step until solution fails $P$
 3: if $i > 18$ then
 4:     while $S_i$ follows $P$ up to step $i$ inclusive do
 5:         compute $S_{i+1}$
 6:         $i \leftarrow i + 1$
 7:     end while
 8:     return $S_{i-1}$
 9: end if
10: // Get the original 16 message words to be modified by the NBbits
11: $(W_0||W_1||...||W_{15}) \leftarrow S_i$
12: // For all $2^{n_i}$ combinations of (multi)bits neutral up to step $i$ inclusive
13: for all $l = 0, 1 \ldots, (2^{n_i} - 1)$ combinations of NBits up to message step $i$ do
14:     // Apply the $l$-th combination of NBits up to step $i$ by XOR-ing all masks
15:     // that compose it to the initial 16 message words
16:     for all $(N_i[q] : 0 \leq q < n_i)$ that belong to combination $l$ do
17:         $(W_0||W_1||...||W_{15}) \leftarrow N_i[q] \oplus (W_0||W_1||...||W_{15})$
18:     end for
19:     // Store the modified 16 message words back to the solution
20:     $S_i \leftarrow (W_0||W_1||...||W_{15})$
21:     // Neutral bit probability
22:     if $S_i$ follows $P$ up to message step $i$ inclusive then
23:         // Compute next message step and call recursively the function
24:         compute $S_{i+1}$
25:         // Differential step probability
26:         if $S_{i+1}$ follows $P$ up to message step $i + 1$ inclusive then
27:             apply_neutral_multibits($i + 1, S_{i+1}, P$)
28:         end if
29:     end if
30: end for
```

regarding its pricing. The ASIC chip in Figure 4 is composed of slave cores, which can either be birthday or neutral-bit slaves. Our digital design flow is shown in below Figure 19. Each slave has been synthesized with a top-down strategy using cadence RTL-compiler v14.8, while placement and routing were done using Cadence Innovus. A Power-Aware Synthesis and Placement-And-Routing are used. Power simulations are performed with the pre- and post-placement and routing back-annotated netlist using Cadence Voltus. The slave is then imported as hard macro in Cadence Virtuoso and instantiated from the top-level RTL. The slave and the interface are then placed and routed in Virtuoso GXL.

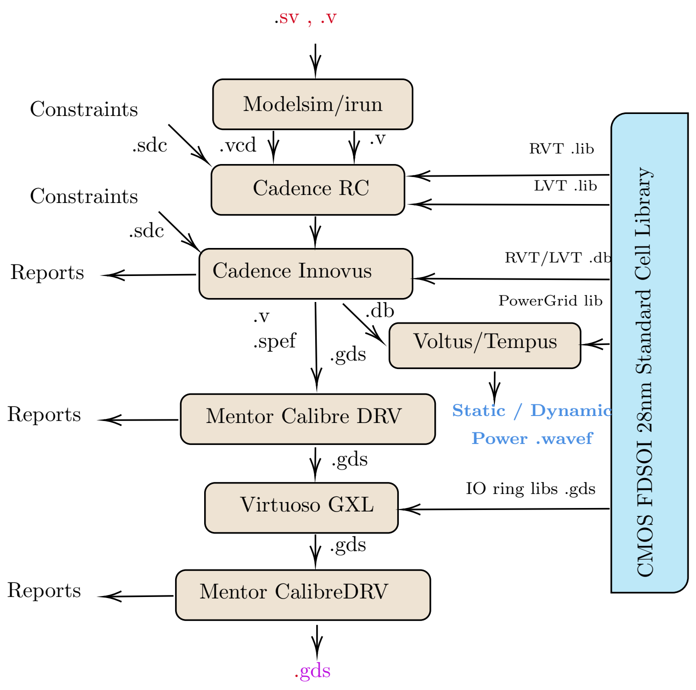


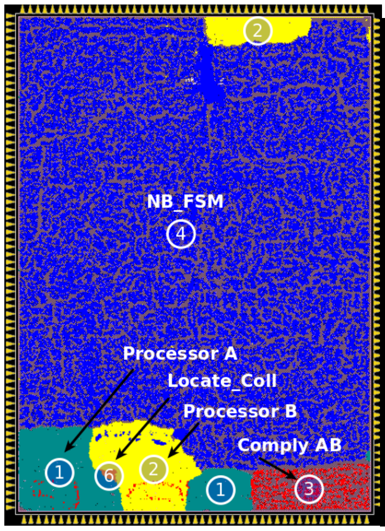
Fig. 19: Our Bottom-Up ASIC digital design flow

The power rail and clock tree are routed with large tracks from the closest power supply and clock pins so as to reduce local voltage drop effect. The RC parasitic extraction of the NB/BD core GDS and final layout is done using Cadence QRC.

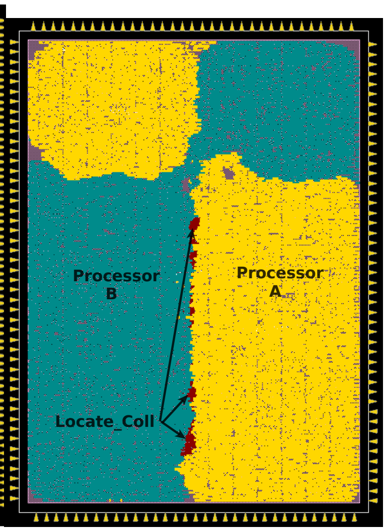
Mentor CalibreDRV is then used for the sign-off DRC and LVS checks. Our design mixes both Regular-Vt (RVT) and Low-Vt (LVT) cells. LVT cells are used without poly-biasing (PB0) for the critical path. RVT cells with poly-biasing up to PB16 are used for the rest of the circuit in order to minimize the leakage power.

Nominal process variation for both PMOS and NMOS for the pre/post-placement-and-routing power simulations with 0.92V supply voltage at 25 degree Celcius are used as parameter for the high performance version of our design. The circuit is first synthesized to reach the maximal operating frequency. Our high speed version reaches 909MHz with V_{fbb}=0V and 1262MHz with V_{fbb}=+2V. LVT cells have then been chosen for the critical path of the NB/BD core. The rest of the circuit have been synthesized with RVT cells so that to balance the performance and power consumption. Each slave is isolated using triple-well isolation to reduce parasitic substrate noise between the slaves that reduces the overall performances. Power simulations show that our $16mm^2$ die requires 140 power supply pins and a plastic-ceramic package to dissipate the power. The effect of body biasing on power and delay after place and route is simulated using Cadence Genus and Voltus. Parasitic extraction with QRC is done with typical parameters. The performances and power result are provided in Table 1.

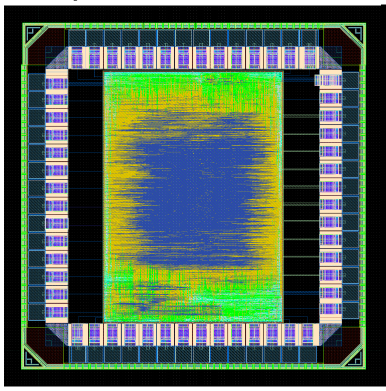
C Chip layout



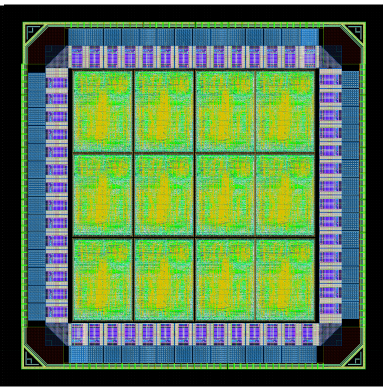
NB Slave ASIC CMOS 28nm FD-SOI layout.



Layout Birthday core.



Sample $1mm^2$. asic layout with 1 NB core.



Sample $1mm^2$ ASIC layout with 12 BD core.

Fig. 21: SHA-1 cryptanalysis accelerator ASIC Layouts