Crypto/Math Paper

SAND2020-8190R

# Side Channel Considerations for SHA-512

W.R. Cordwell

Sandia National Laboratories

SAND xxxxxxx

July 2020

U.S. DEPARTMENT OF ENERGY NNSA

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S.Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND No. 20XX-XXXX.

Sandia National Laboratories

# Summary

We consider a theoretical side-channel attack on SHA-512; the attack should easily generalize to other algorithms in the SHA-2 family.

Rather than looking at a side-channel attack on an HMAC [1], which has been done in various papers [6][7], we assume that the targeted device is applying the hash function as a pseudo-random function (prf) in order to generate a secret key from a secret seed, as recommended by NIST [2]. The analyst uses side-channel information to try to recover the secret seed. We use entropy/information theory to show how one might judge whether or not a side-channel attack might be possible and/or feasible, and we show how the design of the implementation can affect the feasibility of an attack.

# Entropy and Introduction

Entropy can be used to quantify the theoretical amount of work that an adversary must perform to recover a secret key and defeat an encryption system. For example, if we say that a secret key has 256 bits of entropy, we mean that an adversary must recover 256 bits of hidden information. If the adversary is reduced to performing an exhaustive search ("brute force attack") over a 256-bit key space, then the expected number of tries to recover the secret key would be $\frac{1}{2} \cdot 2^{256}$. (The factor of one-half comes from the fact that, on average, the adversary will need to guess half of the total number of keys before he tries the correct one.) Shannon entropy, $H_1$, the best known of the various measures of entropy, gives the average amount of information that is hidden (or revealed), while min-Entropy, $H_\infty$, gives the amount of information hidden or revealed in the most likely case. min-Entropy puts a lower bound on the amount of information that an adversary will gain when performing a measurement, and min-Entropy tends to be the value favored by cryptographers for estimating the security of a system. For a uniform probability distribution, $H_1 = H_\infty$.

Entropy can be understood in various ways. One interpretation is that it is the amount of uncertainty in the outcome of a measurement. It is usually defined in terms of a probability distribution. In a discrete probability distribution, the entropy of an event $i$ can be quantified as $-lg(p_i)$, with units of "bits", where $lg(.)$ is the base two logarithm, and $p_i$ is the probability that the event $i$ occurred. For example, if there is a secret 8-bit byte value chosen at random from all 256 possible values, each value has a probability of $\frac{1}{256} = 2^{-8}$ of occurring, and the amount of uncertainty in selecting or measuring any particular byte value is 8 bits.

Entropy can also be considered as the amount of information in a quantity or in a measurement. Information can be hidden or it can be revealed. If there is a secret byte value involved, there are eight hidden bits of information that we should like to discover. If the byte value is directly revealed to the analyst, he obtains $-lg(\frac{1}{256}) = 8$ bits of information. Instead of learning everything, some partial information about the secret byte might be revealed to the analyst: if the analyst is told that the first bit is a 1, he knows one bit of information. Correspondingly, the uncertainty in the number of possible values has gone down by a factor of $2^{-1} = \frac{1}{2}$ to 128 possibilities (vs. 256), as the probability of such a byte being chosen is $\frac{128}{256} = \frac{1}{2}$. Alternately, if the analyst discovers that the byte value is a multiple of 8, then he knows that the last three bits of the secret byte are all zeros, and he has gained 3 bits of information. This corresponds to a reduction in the search space of $2^{-3} = \frac{1}{8}$, or the number of possibilities being reduced from $2^8 = 256$ to $2^{(8-3)} = 2^5 = 32$ possibilities for the value of the byte.

**Hamming Weights**

The Hamming weight (HW) of a byte is the number of 1s that it contains in the binary representation. A byte value of 0 looks like 0000 0000 (in binary), and it has $HW(0) = 0$. A byte value of 255 looks like 1111 1111, and $HW(255) = 8$, as all eight bits are 1. The

number of bytes of a particular HW corresponds to the probability that a byte has a particular Hamming weight. For example, there are $\binom{8}{1} = 8$ different bytes that have HW = 1, namely 0000 00001, 0000 0010, 0000 0100, ..., 1000 0000. The most likely HW is 4, with four 1s and four 0s in the byte, and there are $\binom{8}{4} = 70$ such bytes. This means that, if the analyst discovers that a (secret) byte has a HW = 4, that there are 70 possibilities, and each one occurs with a probability of $\frac{1}{256}$. The information that is given away by narrowing the possibilities from 256 to 70 is $-lg(\frac{70}{256}) \approx 1.87$ bits. If the analyst discovers that a secret byte has HW = 0, he knows that it is the all-zeros byte, and he has gained 8 bits of information. Table 1 gives a listing of byte Hamming weights and the information revealed by their measured values.

| **HW** | **Probability** | **Revealed Information** | **Num Bytes** |
|---|---|---|---|
| 0 | $^1/_{256}$ | 8.0 | 1 |
| 1 | $^8/_{256}$ | 5.0 | 8 |
| 2 | $^{28}/_{256}$ | 3.19 | 28 |
| 3 | $^{56}/_{256}$ | 2.19 | 56 |
| 4 | $^{70}/_{256}$ | 1.87 | 70 |
| 5 | $^{56}/_{256}$ | 2.19 | 56 |
| 6 | $^{28}/_{256}$ | 3.19 | 28 |
| 7 | $^8/_{256}$ | 5.0 | 8 |
| 8 | $^1/_{256}$ | 8.0 | 1 |

Table 1

Note that at least 1.87 bits of information will always be revealed by measuring the Hamming weight of a byte, corresponding to the min-Entropy of the distribution.

### Shannon Entropy

The Shannon entropy of the distribution is the weighted sum of the individual entropies, or $H_1 = -\sum_i p_i \cdot lg(p_i)$. It is the expected value (or average) of the individual entropies. For the byte Hamming weight probability distribution, as above, the Shannon entropy is approximately 2.54 bits. Thus, *on average*, if a HW value is provided to the analyst, it will give away about 2.5 bits of information about the (secret) byte value. It can be shown that, for a 16-bit word, $H_1$ is about 3 bits; for a 32-bit word, $H_1 \approx 3.5$ bits; and for a 64-bit word, the Shannon entropy is about 4 bits per HW.[†]

# Theoretical Analysis

## SHA-512 Structure

SHA-512 is one of the SHA-2 family of Secure Hash Algorithms defined in FIPS 180-4 [3]. We review some definitions and the structure of SHA-512.

### Definitions and Functions

| | |
|---|---|
| $a,\ b,\ c, \ldots, h$ | The internal registers of the hash function, each register is 64 bits for SHA-512 |
| $K_t$ | A round constant for the $t^{th}$ iteration |
| $w$ | number of bits in a word, for SHA-512 $w = 64$ |
| $W_t$ | the $t^{th}$ $w$-bit word of the message schedule |
| $\wedge$ | bitwise AND operation |
| $\vee$ | bitwise OR operation |
| $\oplus$ | bitwise XOR operation |
| $\neg$ | bitwise complement (not) operation |
| $+$ | addition modulo $2^w$ ; mod $2^{64}$ for SHA-512 |

[†]For an $n$-bit word, $H_1 \approx -\frac{1}{2} + \frac{1}{2} \cdot lg(\pi \cdot e \cdot n)$ [4]

| | |
|---|---|
| $x \gg n$ | shift word $x$ right by $n$ bits, filling in with zeros on the left |
| $x \ggg n$ | right circular shift of word $x$ by $n$ bits |

| | |
|---|---|
| $Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$ | operating on words $x$, $y$, and $z$ |
| $Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$ | Majority function, bitwise |
| $\Sigma_0(x) = (x \ggg 28) \oplus (x \ggg 34) \oplus (x \ggg 39)$ | |
| $\Sigma_1(x) = (x \ggg 14) \oplus (x \ggg 18) \oplus (x \ggg 41)$ | |
| $\sigma_0(x) = (x \ggg 1) \oplus (x \ggg 8) \oplus (x \gg 7)$ | |
| $\sigma_1(x) = (x \ggg 19) \oplus (x \ggg 61) \oplus (x \gg 6)$ | |

SHA-512 and SHA-384 both have eighty iterations (rounds), and they use the same eighty round-constant words, $K_t$. They represent the first 64 bits of the fractional parts of the cube roots of the first eighty prime numbers.
The initial internal register states $a$, $b, \ldots,$ $h$ are the first 64 bits of the fractional parts of the square roots of the first eight prime numbers for SHA-512.

The message schedule is built from an input of 1024 bits. Of the eighty words, the first 16 words are the input 1024 bits broken up into (consecutive) 64-bit words, $W_t$, for $0 \leq t \leq 15$. For $16 \leq t \leq 79$, $W_t = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}$.

**Rounds and Information**

The round structure can be viewed as in Figure 1, which follows the diagram in Wikipedia. [5]

where the red squares with the plus sign are addition mod $2^w$.

Each of the modular addition, Ch, Maj, $\Sigma_0$, and $\Sigma_1$ operations work with 64-bit inputs and outputs for SHA-512. For the most part, we shall not consider the component opera-

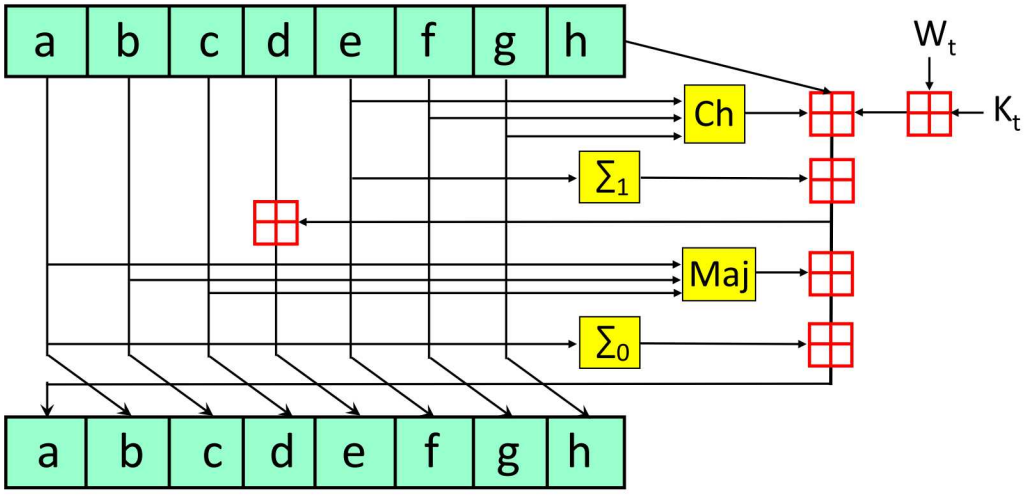


Figure 1

tions such as shifts, XORs, logical ands, etc., that make up these larger operations, although those, too, could be a source of information.

Assuming that the analyst measures Hamming weights, how much information is (theoretically) provided by an operation depends on whether the operation is performed byte-wise, in 16-bit pieces, 32-bit pieces, or in 64-bit words. If the operations are performed byte-wise, about 2.54 bits of entropy (information) is revealed per byte HW measurement. At the other extreme, the HW of a 64-bit word will give, on average, a little more than 4 bits of information.

We note that, even though information exists, it is not necessarily exploitable. For example, AES-128 uses a 128-bit key, takes in a 128-bit plaintext, and produces a 128-bit ciphertext. If an analyst is given a single plaintext-ciphertext pair, he has been given $\sim 127.17$ bits of information [4] (even without side-channel information). However, even though this is enough to narrow down the possible secret key to two choices, the analyst does not have the necessary computational power, absent a quantum computer, to determine the two possibilities.

## Information Theoretic Analysis

The goal of the analyst is to recover the 1024-bit input block, which comprises the first sixteen input words $W_t$.

We shall focus our analysis on the first sixteen rounds (iterations) of SHA-512. The later rounds have input words corresponding to shifts and XORings of previous input words. We note that information is available from measuring HWs of later rounds, but it is less obvious how to recover the original input block. It could be useful, though, for verifying derived possibilities of the input block, if the analyst can narrow down the choices sufficiently. Initially, we shall only consider HW measurements from the modular addition functions.

The first round can be particularly useful, because there are six modular additions, and only 64 bits of unknown input, $W_0$, the first input word. The other values that enter into the modular additions are known values of the register initial contents.

The best case for the analyst is if the HW values can be measured by bytes. Since the Shannon entropy is $\sim 2.5$ bits per HW, and there are $6 \times 8 = 48$ bytes involved, there are 120 bits of information available, with 64 bits of unknown $W_0$, which should be more than enough to determine the value of $W_0$. The following example illustrates just one way that the analyst might proceed.

Suppose that the low bytes of the six known modular addition inputs are, respectively, 10111011, 11100111, 10111101, 10011010, 10001000, and 10001110. These were generated at random, but they would actually be known values. Suppose also, that the HW measurements corresponding to the low bytes of the six modular additions are 1, 4, 2, 6, 4, 4.

(1) The only byte values that combine with 10111011 and give a HW of 1 are 01000110, 01000111, 01001001, 01001101, 01010101, 01100101, 10000101, 11000101. These are

the candidates for the first byte of $W_0$, after using the first HW measurement.

(2) We test the byte values of the previous sum (eight possibilities; dropping any carry bit) by adding it to the second known modular addition input and saving only the answers that have a HW of 4. This narrows the possibilities to 01000110, 11000101.

(3) Adding the cumulative (byte) sum possibilities to the third modular addition byte input for a HW of 4 gives just one possibility: 11000101.

Now the first byte of the input $W_0$ is known, and we can use it in doing the same process for the second byte (knowing the lower byte allows us to compute any carry bit that might go into the second byte of a modular addition). In this fashion, we can recover all of the bytes of $W_0$. Once the first input word is recovered, we know the initial state values for the second input word, and we proceed in the same manner, recovering the entire 1024-bit input.

It becomes more difficult for the analyst as the Hamming weight word size increases. Although, for a 16-bit word, there are $\sim 3$ bits of information per HW, the number of bits to be determined (per word) has gone up substantially. Still, for 16-bit measurements, there are, in the first round, $6 \times 4 = 24$ HW measurements available. At 3 bits per HW (on average), there are 72 bits of information available to determine 64 unknown bits.

At 32 bit HW measurements, there are $\sim 3\frac{1}{2}$ bits of information available per HW. There are 12 such measurements available, so there are only 42 bits of information available, leaving a gap of 22 bits. If HW values can be measured during the second round at the outputs of the $Ch$, $Maj$, $\Sigma_0$, and $\Sigma_1$ functions, where the $W_0$ value comes in as an input from the (updated) state registers $a$ and $e$, that would give an additional $8 \times 3.5 = 28$ bits of information, making this theoretically possible, but quite possibly infeasible.

For 64 bit HW measurements, there are only six modular addition HW measurements,

and perhaps an extra four from the $Ch$, $Maj$, $\Sigma_0$, and $\Sigma_1$ functions in the second round, giving only 40 bits of information to determine 64 unknown bits. The only way to improve this might be if the HWs from the individual operations that go into the $Ch$, $Maj$, $\Sigma_0$, and $\Sigma_1$ functions were available. Note that a circular shift preserves the Hamming weight, so it would not give any additional information. We note that [6] proposed attacking the lower-level logical and ($\wedge$) function (as well as the modular additions) in their description of a DPA attack on SHA-256 HMAC. (The HMAC attack is a bit different, as it assumes that the secret key (scrambled) comprises the initial register values, so that the initial register values are unknown, but that the 1024-bit (or more) inputs are known. The HMAC standard requires two hashings involving the secret key, which provides opportunities for additional information recovery).

Optimally, for the analyst, six modular additions, 2 logical ands, one "not" operation, one XOR for the Ch function (the Ch function HWs would be measured in the second round, where the updated register $e$ has the first input word, $W_0$ mixed in it, but none of the inputs have any other unknowns. The $\Sigma_1$ operator also acts on the updated $e$ register in round 2, and has two XOR operations involving it. The Maj function has three logical ands and two XORs, which give HWs to be measured in the second round, after register $a$ is updated, except for the last logical and, the HW of which would be measured in the third round, when the updated $b$ register would have $W_0$ mixed in it, but no later unknown input words would be used. The $\Sigma_0$ operation HWs, with two XORs, would be measured in the second round, after its input has been updated.

This gives, in the best case for the attacker, nineteen 64-bit HW measurements involving the first 64-bit input word, giving an average of 76 bits of information. This would say that such an attack would be theoretically possible, but the measurements would need to be reasonably precise, and all of the mentioned operations would need to be registered.

### Changing the Implementation

If, instead of implementing according to the Wikipedia diagram, the designer aggregates the operations of $Ch(e, f, g) + \Sigma_1(e) + K_t$ before adding with $W_t$, and $Maj(a, b, c) + \Sigma_0(a)$ before the final modular addition in the first round, then it might appear that he reduces the number of modular additions that act with the input word to three, rather than six. If the analyst is depending primarily on the modular addition HW measurements, that would seem to cut the amount of available information in half.[†] However, the analyst can recover the use of those by operations by measuring the $Ch(e, f, g) + \Sigma_1(e) + K_t$ and the $Maj(a, b, c) + \Sigma_0(a)$ modular additions in the second round, before the second input word affects them. If the implementation were in software, say, the designer could aggregate operations in the first round, but go back to the Wikipedia version for later rounds, when they would be mixing with the second input word.

It seems as though the designer is more likely to preclude a successful side-channel attack by focusing on the hardware aspects of the implementation. If the designer is able to use 64-bit computations, that should help to minimize the amount of side-channel information exposed. If it is possible to perform the lower-level operations without giving away HW information, that could help prevent such an attack as well.

## Comments and Conclusions

By using Entropy/Information Theory arguments, we see that an analyst may be able to use side-channel information to recover a secret seed that is input to SHA-512, if he can measure Hamming weights as the message is being processed. The attack would be the same for SHA-512, and should extend to any of the SHA-2 family of algorithms.

[†]This way of implementing is suggested by the description in [3]

The side-channel attack is more likely to succeed if the hash algorithm processes the operations in smaller word sizes. If the design processes the input in bytes, we judge that the attack is feasible. If the design processes the information in 64-bit words (as for SHA-512 and SHA-384), the attack becomes more problematic.

A major assumption is that the analyst can measure perfect (or near-perfect) Hamming weights. This assumption may be possible to achieve if the analyst is able to make repeated measurements of the identical hash operation, allowing him to average the recovered traces/signals. If the word size used in the hashing operations is relatively small, this assumption can be somewhat relaxed. We note that, at least in theory, there is information available from later rounds, as there are 80 rounds of processing, but only 20 independent input words.

As always, whether or not a side-channel attack is successful will depend on the implementation and the device characteristics.

# References

[1] Federal Information Processing Standards Publication, FIPS-198-1, The Keyed-Hash Message Authentication Code (HMAC). NIST. July 2008.

[2] Lily Chen. Recommendation for Key Derivation Using Pseudorandom Functions, Special Publication 800-108. NIST. October 2009. (Revised)

[3] Federal Information Processing Standards Publication, FIPS-180-4, Secure Hash Standard (SHS). NIST. August 2015.

[4] Advanced Crypto-Math Class. *Sandia National Laboratories.*

[5] https://en.wikipedia.org/wiki/SHA-2

[6] Sonia Belaïd, et al. *Differential Power Analysis of HMAC SHA-2 in the Hamming Weight Model.* SECRYPT 2013 - 10th International Conference on Security and Cryptography (2013).

[7] Cordwell, William. *HMAC Susceptibility to DPA.* Sandia National Laboratories. 01 October 2008.

[8] C.E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal.* 27:379-423,623-646. 1948.

[9] R.M. Gray. Entropy and Information Theory. Springer-Verlag, New York, NY. 1990.